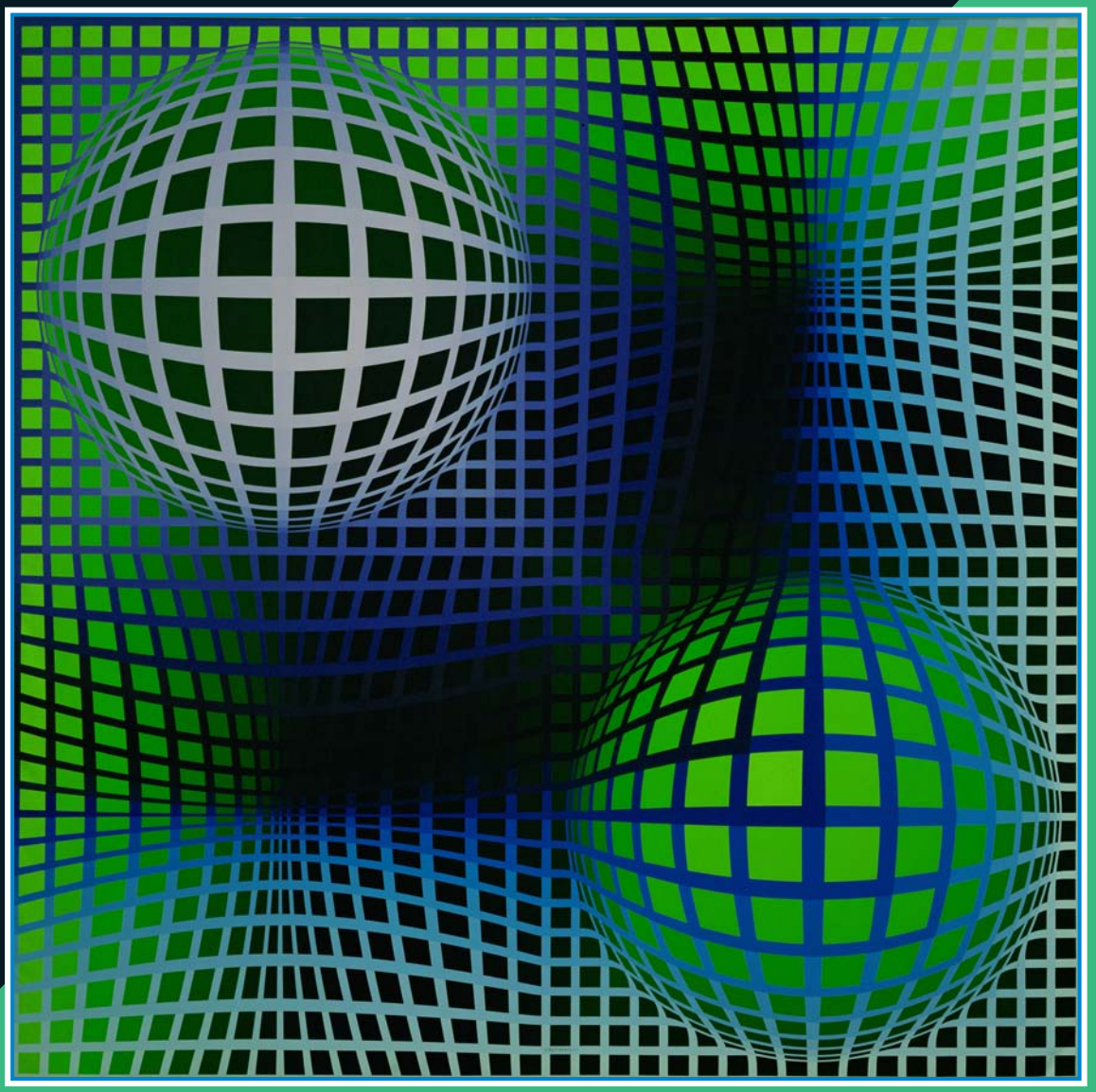
MATEMÁTICAS DISCRETAS

Sexta edición



PEARSON
Prentice Hall

Richard Johnsonbaugh

LISTA DE SÍMBOLOS

LÓGICA

$p \vee q$	p o q; p. 2
$p \wedge q$	p y q; p. 2
$\neg p$	no p; p. 5
$p \rightarrow q$	si p, entonces q; p. 8
$p \leftrightarrow q$	p si y sólo si q; p. 12
$P \equiv Q$	P y Q son lógicamente equivalentes; p. 12
$\forall$	para todo; p. 19
$\exists$	existe; p. 22
$\backslash$	por lo tanto; p. 43

NOTACIÓN DE CONJUNTOS

$\{x_1,\ldots, x_n\}$	conjunto que consta de los elementos $x_1,\ldots, x_n$; p. 76
$\{x \mid p(x)\}$	conjunto de los elementos x que satisfacen la propiedad $p(x)$; p. 77
$x \in X$	x es un elemento de X; p. 77
$x \notin X$	x no es un elemento de X; p. 77
$X = Y$	igualdad de conjuntos (X y Y tienen los mismos elementos); p. 77
$\lvert X \rvert$	número de elementos en X; p. 77
$\emptyset$	conjunto vacío; p. 77
$X \subseteq Y$	X es un subconjunto de Y; p. 77
$X \subset Y$	X es un subconjunto propio de Y; p. 79
$\mathcal{P}(X)$	conjunto potencia de X (todos los subconjuntos de X); p. 79
$X \cup Y$	X unión Y (todos los elementos en X o Y); p. 80
$\bigcup_{i=1}^{n} X_i$	unión de $X_1,\ldots, X_n$ (todos los elementos que pertenecen al menos a un conjunto de $X_1,\ldots, X_n$); p. 83
$\bigcup_{i=1}^{\infty} X_i$	unión de $X_1, X_2,\ldots$ (todos los elementos que pertenecen al menos a uno de $X_1, X_2,\ldots$); p. 83
$\cup\mathcal{S}$	unión de $\mathcal{S}$ (todos los elementos que pertenecen al menos a un conjunto en $\mathcal{S}$); p. 83
$X \cap Y$	X intersección Y (todos los elementos en X y Y); p. 80
$\bigcap_{i=1}^{n} X_i$	intersección de $X_1,\ldots, X_n$ (todos los elementos que pertenecen a todos los conjuntos $X_1, X_2,\ldots, X_n$); p. 83
$\bigcap_{i=1}^{\infty} X_i$	intersección de $X_1, X_2,\ldots$ (todos los elementos que pertenecen a todos los conjuntos $X_1, X_2,\ldots$); p. 83
$\cap\mathcal{S}$	intersección de $\mathcal{S}$ (todos los elementos que pertenecen a todos los conjuntos de $\mathcal{S}$); p. 83
$X - Y$	diferencia de conjuntos (todos los elementos en X pero no en Y); p. 80
$\overline{X}$	complemento de X (todos los elementos que no están en X); p. 80
(x, y)	par ordenado; p. 83
$(x_1,\ldots, x_n)$	n-eada; p. 84
$X \times Y$	producto cartesiano de X y Y [pares (x, y) con x en X y y en Y]; p. 83

Relaciones

$x\ R\ y$	(x, y) está en R (x está relacionada con y mediante la relación R); p. 117
$[x]$	clase de equivalencia que contiene a x; p. 127
R^{-1}	relación inversa [todo (x, y) que está en R]; p. 122
$R_2 \circ R_1$	composición de relaciones; p. 122
$x \preceq y$	$x\ R\ y$; p. 121

Funciones

$f(x)$	valor asignado a x; p. 88
$f: X \to Y$	función de X a Y; p. 87
$f \circ g$	composición de f y g; p. 97
f^{-1}	función inversa [todo (y, x) con (x, y) que está en f]; p. 96
$f(n) = O(g(n))$	$\lvert f(n)\rvert \leq C\lvert g(n)\rvert$ para n suficientemente grande; p. 158
$f(n) = \Omega(g(n))$	$c\lvert g(n)\rvert \leq \lvert f(n)\rvert$ para n suficientemente grande;p. 158
$f(n) = \Theta(g(n))$	$c\lvert g(n)\rvert \leq \lvert f(n)\rvert \leq C\lvert g(n)\rvert$ para n suficientemente grande; p. 158

Conteo

$C(n, r)$	número de combinaciones r de un conjunto de n elementos $(n!/[(n-r)!r!])$; p. 232
$P(n, r)$	número de permutaciones r de un conjunto de n elementos $[n(n-1)\cdot\cdot\cdot(n-r+1)]$; p. 231

Gráficas

$G = (V, E)$	gráfica G con conjunto de vértices V y conjunto de aristas E; p. 320
(v, w)	arista; p. 320
$\delta(v)$	grado del vértice v; p. 333
$(v_1,\ldots, v_n)$	trayectoria de v_1 a v_n; p. 330
$(v_1,\ldots, v_n)$, $v_1 = v_n$	ciclo; p. 332
K_n	gráfica completa en n vértices; p. 325
$K_{m, n}$	gráfica completa bipartita en m y n vértices; p. 326
$w(i, j)$	peso de la arista (i, j); p. 347
F_{ij}	flujo en la arista (i, j); p. 445
C_{ij}	capacidad de la arista (i, j); p. 445
$(P, \overline{P})$	cortadura en una red; p. 457

Probabilidad

$P(x)$	probabilidad del resultado x; p. 250
$P(E)$	probabilidad del evento E; p. 251
$P(E\lvert F)$	probabilidad condicional de E dado F $[P(E \cap F)/P(F)]$; p. 255

MATEMÁTICAS DISCRETAS

MATEMÁTICAS DISCRETAS

SEXTA EDICIÓN

Richard Johnsonbaugh
DePaul University, Chicago

TRADUCCIÓN:
MARCIA AÍDA GONZÁLEZ OSUNA
Facultad de Ciencias
Universidad Nacional Autónoma de México

REVISIÓN TÉCNICA:
ARIADNE SÁNCHEZ RUIZ
Facultad de Ingeniería, Mecánica y Eléctrica
Universidad Autónoma de Nuevo León

PEARSON Educación®

México • Argentina • Brasil • Colombia • Costa Rica • Chile • Ecuador
España • Guatemala • Panamá • Perú • Puerto Rico • Uruguay • Venezuela

Datos de catalogación bibliográfica

MATEMÁTICAS DISCRETAS. Sexta edición

Johnsonbaugh, Richard

PEARSON EDUCACIÓN, México, 2005

ISBN: 970-26-0637-3

Área: Universitarios

Formato: 21 × 27 cm Páginas 696

ISBN 0-13-117686-2

Edición en español

Editor: Enrique Quintanar Duarte
e-mail: enrique.quintanar@pearsoned.com
Editor de desarrollo: Felipe Hernández Carrasco
Supervisor de producción: Rodrigo Romero Villalobos

Edición en inglés
Executive Acquisitions Editor: George Lobell
Editor-in-Chief: Sally Yagan
Vice President/Director of Production and Manufacturing: David W. Riccardi
Production Editor: Debbie Ryan
Senior Managing Editor: Linda Mihatov Behrens
Assistant Managing Editor: Bayani Mendoza de Leon
Executive Managing Editor: Kathleen Schiaparelli
Assistant Manufacturing Manager/Buyer: Michael Bell
Manufacturing Manager: Trudy Pisciotti
Marketing Manager: Halee Dinsey
Marketing Assistant: Rachel Beckman
Art Director: John Christiana
Interior Design: Abigail Bass
Cover Designer: Anthony Gemmellaro
Creative Director: Carole Anson
Director of Creative Services: Paul Belfanti
Art Editor: Thomas Benfatti
Editorial Assistant: Joanne Weldelken
Front/Back Cover Image: Vasarely, Victor (1908-1977) © ARS, NY Boo. 1978.
Location: Private Collection, Monaco/Photo Credit: Erich Lessing / Art Resource, NY

SEXTA EDICIÓN, 2005

D.R.© 2005 por Pearson Educación de México, S.A. de C.V.
Atlacomulco No. 500, 5° piso
Col. Industrial Atoto
53519, Naucalpan de Juárez, Edo. de México
E-mail: editorial.universidades@pearsoned.com

Cámara Nacional de la Industria Editorial Mexicana. Reg. Núm. 1031

ISBN 970-26-0637-3

Impreso en México. *Printed in Mexico.*

1 2 3 4 5 6 7 8 9 0 – 08 07 06 05

CONTENIDO

PREFACIO

Este libro fue diseñado para un curso de introducción a las matemáticas discretas, basado en mi experiencia como profesor de la asignatura durante muchos años. Los prerrequisitos de matemáticas formales son mínimos; no se requiere cálculo. No hay requisitos de ciencias de la computación. El libro incluye ejemplos, ejercicios, figuras, tablas, secciones de solución de problemas, secciones que contienen sugerencias para resolver problemas, secciones de repaso, notas, revisión del capítulo, autoevaluaciones y ejercicios para realizar en computadora con la finalidad de ayudar al estudiante a dominar las matemáticas discretas.

A principios de la década de 1980, había pocos libros de texto adecuados para un curso de introducción a las matemáticas discretas. Sin embargo, era necesario un curso que consolidara la madurez matemática de los estudiantes y su habilidad para manejar la abstracción, que además incluyera temas útiles como combinatoria, algoritmos y gráficas. La edición original de este libro (1984) atendió esta necesidad e influyó de manera significativa en los cursos de matemáticas discretas. Con el paso del tiempo, los cursos de matemáticas discretas se justificaron para diferentes grupos, que incluyeron estudiantes de matemáticas y ciencias de la computación. Un panel de la Mathematical Association of America (MAA) apoyó un curso de un año de matemáticas discretas. El Comité de Actividades Educativas del Institute of Electrical and Electronics Engineers (IEEE) recomendó un curso de matemáticas discretas en el primer año. Las guías de acreditación de la Association for Computing Machinery (ACM) y la IEEE hacen obligatorio un curso de matemáticas discretas. Esta edición, al igual que las anteriores, incluye temas como algoritmos, combinatoria, conjuntos, funciones e inducción matemática para cubrir las necesidades de estos grupos. También toma en cuenta la comprensión y construcción de pruebas y, en general, el reforzamiento de la madurez matemática.

Cambios respecto a la quinta edición

Los cambios en la sexta edición de este libro surgieron a partir de comentarios y peticiones de numerosos usuarios y revisores de las ediciones anteriores. Estos cambios son los siguientes:

- El primer capítulo de lógica y demostraciones fue ampliado en forma considerable. La sección de cuantificadores en la quinta edición fue dividida en dos secciones. La primera sección de cuantificadores (sección 1.3) estudia las afirmaciones de los cuantificadores sencillos, y la siguiente (sección 1.4) analiza los cuantificadores anidados. La sección de inducción matemática en la quinta edición también fue dividida en dos secciones. La primera (sección 1.7) introduce la inducción en la que el paso inductivo consiste en suponer S(n) y probar S(n + 1). Se agregó a esta sección un análisis de las invariantes de un ciclo. La segunda sección (1.8) continúa con la inducción fuerte y la propiedad de buen orden. Como ejemplo del uso de la propiedad del buen orden, se demuestra el teorema del cociente-residuo.

  Los materiales de exposición, ejemplos y motivación se ampliaron en todo el capítulo. Se agregaron ejemplos de lógica en lenguajes de programación (por ejemplo, el uso de and, or, not y las leyes de De Morgan). Con el fin de dar mayor claridad, la barra superior para denotar negación se sustituyó por ¬. Aparece una explicación más

detallada de las condiciones necesaria y suficiente. Se añadieron ejemplos para mostrar la relación entre el lenguaje normal y la lógica simbólica. En todo el capítulo hay más ejemplos de demostración de afirmaciones y de cómo construir las demostraciones. En este capítulo se aumentó el número de ejemplos resueltos de 59 a 90, y el número de ejercicios de 391 a 521.

- En el capítulo 2, se agregaron varios ejemplos para mostrar cómo es posible usar el material del capítulo 1 para probar afirmaciones referentes a conjuntos, funciones, sucesiones y cadenas (por ejemplo, cómo probar que una función específica es uno a uno).
- Ahora se presentan varios ejemplos de algoritmos antes de llegar a la notación de O mayúscula y otras relacionadas (secciones 4.1 y 4.2), que proporcionan una introducción más suave y motivada al formalismo que le sigue. Se menciona que muchos algoritmos modernos no tienen todas las propiedades de los algoritmos clásicos (por ejemplo, muchos algoritmos modernos no son generales, determinísticos o finitos). Para ilustrar el punto, se da un ejemplo de un algoritmo aleatorizado (ejemplo 4.2.4).
- Un nuevo capítulo (el 5) de introducción a la teoría de números combina material de la quinta edición (como la representación de enteros, el máximo común divisor) y amplía algunos temas (como la teoría algorítmica de números). Este capítulo incluye resultados clásicos (como la divisibilidad, el carácter infinito de los primos, el teorema fundamental de la aritmética), así como los algoritmos de teoría de números: por ejemplo, el algoritmo euclidiano para encontrar el máximo común divisor, cómo elevar a un exponente usando cuadrados repetidos, cómo calcular s y t tales que mcd(a, b) = sa + tb, y cómo calcular el inverso del módulo de un entero. La aplicación principal es el sistema criptográfico de llave pública RSA (sección 5.4). Los cálculos requeridos por el sistema criptográfico se pueden realizar usando los algoritmos desarrollados en el capítulo.
- El apartado de sugerencias para resolver problemas se agregaron al final de muchas secciones, en especial en los primeros capítulos. Como el nombre lo dice, ayudan al estudiante a centrarse en las técnicas requeridas para resolver los problemas. Las sugerencias para resolver problemas, que aparecen al final de las secciones, enfatizan y ayudan a comprender las técnicas para resolver los problemas de la sección.
- Hay nuevas secciones de solución de problemas para funciones y teoría de números.
- El estilo del seudocódigo se ha actualizado del tipo Pascal al tipo Java (que también se parece a C y C++). Es más probable que el estudiante esté familiarizado con este estilo. Además, la descripción del seudocódigo se ha cambiado a los apéndices (apéndice C), lo que hace posible dar ejemplos de seudocódigos antes (para quienes estén interesados).
- Se agregaron diversos libros y artículos recientes a la lista de referencias. Las referencias de varios libros se actualizaron para considerar las últimas ediciones.
- El número de ejemplos resueltos aumentó a cerca de 600. (Había aproximadamente 500 en la quinta edición).
- El número de ejercicios aumentó a casi 4000. (Había cerca de 3500 en la quinta edición).

Contenido y estructura

Este libro incluye:

- Lógica (incluyendo cuantificadores), demostraciones, pruebas por resolución e inducción matemática (capítulo 1). En el ejemplo 1.4.15 se presenta un juego de lógica, que ofrece una manera alternativa para determinar si una función proposicional cuantificada es verdadera o falsa.
- Conjuntos, funciones, sucesiones, notaciones de suma y producto, cadenas y relaciones (capítulo 2 y 3), incluyendo ejemplos que despiertan la motivación, como el de la introducción a las funciones de dispersión (hashing) y los generadores de números seudoaleatorios (sección 2.2), una aplicación de órdenes parciales en la programación de tareas (sección 3.1) y las bases de datos relacionales (sección 3.4).

- Un análisis exhaustivo de algoritmos, algoritmos recursivos y análisis de algoritmos (capítulo 4). Además, se da un enfoque algorítmico en todo el libro. Los algoritmos están escritos en una forma flexible de seudocódigo. (El libro no supone requisitos de computación; la descripción del seudocódigo que se usa se presenta en el apéndice C). Entre los algoritmos presentados están el de enlosado (sección 4.4), el algoritmo euclidiano para encontrar el máximo común divisor (sección 5.3), el algoritmo para codificar la llave pública RSA (sección 5.4), la generación de combinaciones y permutaciones (sección 6.3), el merge sort (sección 7.3), el algoritmo de la ruta más corta de Dijkstra (sección 8.4), los algoritmos de regreso (sección 9.3), la búsqueda a lo ancho y a profundidad (sección 9.3), el recorrido de árboles (sección 9.6), la evaluación de un árbol de juego (sección 9.9), el flujo máximo en una red (sección 10.2), el par más cercano de puntos (sección 13.1) y el cálculo del casco convexo (sección 13.2).
- Un análisis completo de las notaciones O mayúscula, omega y theta para el crecimiento de las funciones (sección 4.3). Disponer de todas estas notaciones hace posible hacer afirmaciones precisas acerca del crecimiento de funciones y el tiempo y espacio requeridos por los algoritmos.
- Una introducción a la teoría de números (capítulo 5).
- Combinaciones, permutaciones, probabilidad discreta y el principio del palomar (capítulo 6). Dos secciones opcionales (6.4 y 6.5) presentan la probabilidad discreta.
- Relaciones de recurrencia y su aplicación al análisis de algoritmos (capítulo 7).
- Gráficas, incluyendo modelos de cobertura de una gráfica de computadoras paralelas, recorrido del caballo, ciclos de Hamilton, isomorfismos de gráficas y gráficas planas (capítulo 8). El teorema 8.4.3 da una demostración elegante, breve y sencilla de que el algoritmo de Dijkstra es correcto.
- Árboles, que incluyen árboles binarios, recorridos del árbol, árbol de expansión mínima, tiempo mínimo para ordenar e isomorfismos de árboles (capítulo 9).
- Redes, el algoritmo del flujo máximo y acoplamiento (capítulo 10).
- Un análisis de álgebras booleanas que hace hincapié en la relación de éstas con los circuitos combinatorios (capítulo 11).
- Un enfoque de autómata que resalta el modelado y las aplicaciones (capítulo 12). El circuito flip-flop SR se estudia en el ejemplo 12.1.11. Los fractales, incluyendo el copo de nieve de Von Koch, se describen por gramáticas de tipo especial (ejemplo 12.3.19).
- Una introducción a la geometría para el cálculo (capítulo 13).
- Apéndices de matrices, álgebra básica y seudocódigo.
- Se resalta la importancia de la interrelación de los diferentes temas. Como ejemplos, la inducción matemática tiene una relación estrecha con los algoritmos recursivos (sección 4.4); la sucesión de Fibonacci se usa en el análisis del algoritmo euclidiano (sección 5.3); muchos ejercicios a lo largo del libro requieren inducción matemática; se demuestra cómo se caracterizan las componentes de una gráfica mediante la definición de una relación de equivalencia sobre el conjunto de vértices (vea el análisis que sigue al ejemplo 8.2.13); se cuentan los vértices de árboles binarios de n vértices (teorema 9.8.12).
- Se hace hincapié en la lectura y las demostraciones. Casi todas las pruebas de los teoremas se ilustran con figuras que incluyen anotaciones. Algunas secciones separadas (los rincones de solución de problemas) muestran al estudiante cómo abordar y resolver problemas y cómo desarrollar demostraciones. Las secciones especiales de sugerencias para resolver problemas resaltan las técnicas principales de la sección.
- Un gran número de aplicaciones, en especial de computación.
- Figuras y tablas para ilustrar los conceptos, para mostrar cómo funcionan los algoritmos, para derivar pruebas y para aclarar el material. Las figuras ilustran las pruebas de los teoremas. Las leyendas de estas figuras brindan explicaciones adicionales y mayor comprensión de la demostración.
- Ejercicios de repaso de la sección.
- Secciones de notas con sugerencias para otras lecturas.

- Repasos de los capítulos.
- Autoevaluación en cada capítulo.
- Ejercicios para computadora.
- Una sección de referencias que contiene 159 referencias.
- Contraportadas que resumen la notación matemática y de los algoritmos utilizados en el libro.

Cada capítulo se organiza como sigue:

Descripción general
Sección
Sección de ejercicios de repaso
Sección de ejercicios
Sección
Sección de ejercicios de repaso
Sección de ejercicios
.
.
.
Notas
Repaso del capítulo
Autoevaluación del capítulo
Ejercicios para computadora

El apartado de ejercicios de repaso revisa los conceptos clave, definiciones, teoremas, técnicas, etcétera, de la sección. Todos los ejercicios de repaso tienen respuestas al final del libro. Aunque la intención es repasar cada sección, estos ejercicios también se pueden usar para la elaboración de exámenes muestra.

Las notas contienen sugerencias para otras lecturas. Las revisiones de los capítulos proporcionan listas de referencia de los conceptos importantes. Las autoevaluaciones contienen cuatro ejercicios por sección, con respuestas al final del libro.

Los ejercicios para computadora incluyen proyectos, implementaciones de algunos algoritmos y otras actividades relacionadas con la programación. Aunque no hay requisitos de programación para este libro, ni se pretende hacer una introducción a la programación, estos ejercicios se incluyen para quienes deseen explorar los conceptos de matemáticas discretas con una computadora.

Ejercicios

Por último, casi todos los capítulos incluyen una sección de solución de problemas.
El libro contiene cerca de 4000 ejercicios, 147 de los cuales son para computadora. Los ejercicios que parecen más difíciles que otros se indican con el símbolo ⋆. Los ejercicios con números en cursivas (cerca de un tercio de ellos) tienen una sugerencia o la solución al final del libro. Las soluciones a los ejercicios restantes se pueden encontrar en la Guía del instructor. Es claro que un puñado de ejercicios requieren del cálculo. No se usan conceptos de cálculo en el cuerpo principal del libro y, excepto por estos pocos ejercicios, no se necesita cálculo para resolverlos.

Ejemplos

El libro contiene cerca de 600 ejemplos resueltos, que muestran a los estudiantes cómo abordar problemas en matemáticas discretas, presentan las aplicaciones de la teoría, aclaran las demostraciones y ayudan como motivación del material.

Rincones de solución de problemas

Las secciones tituladas Rincón de solución de problemas ayudan al estudiante a atacar y resolver problemas, y le muestran cómo desarrollar demostraciones. Escritos en un estilo informal, cada "rincón" es una sección que, por sí misma, sigue el análisis del tema del problema. En lugar de presentar simplemente una prueba o una solución al problema, en estas secciones se intenta mostrar los caminos alternativos para atacar el problema, se analiza qué buscar al tratar de resolverlo y se presentan las técnicas para encontrar soluciones y para elaborar demostraciones.

Cada sección de solución de problemas comienza con el enunciado de un problema. Después de hacer el planteamiento se analizan las maneras de abordarlo. Siguen a este análisis las técnicas para encontrar una solución. Una vez que se encuentra una respuesta, se da una solución formal para mostrar la manera correcta de escribirla. Por último, se resumen las técnicas de solución de problemas presentadas en la sección, incluyendo un apartado de comentarios que analiza las relaciones con otros temas de matemáticas y computación, despierta la motivación para resolver el problema y da una lista de referencias para otras lecturas relacionadas. Algunos rincones de solución de problemas concluyen con ejercicios.

Complemento para el instructor

Guía del instructor (en inglés) sin costo para los profesores que adopten este libro. Debe solicitarse al representante local de Pearson Educación. La Guía del instructor contiene las soluciones de los ejercicios no incluidas en el libro.

Página de Internet

www.pearsoneducacion.net/johnsonbaugh

se ha enriquecido mucho respecto a la página de la edición anterior. El nuevo sitio contiene

WWW

- Mayores explicaciones del material difícil y vínculos a otros sitios para obtener información adicional de los temas de matemáticas discretas. El icono www al margen señala que la página Web del libro contiene más explicaciones o un vínculo.
- Diapositivas de PowerPoint.
- Material complementario
- Programas de computadora
- Una lista de erratas.

Agradecimientos

Recibí comentarios útiles de muchas personas, entre ellas Gary Andrus, Kendall Atkinson, André Berthiaume, Gregory Brewster, Robert Busby, David G. Cantor, Tim Carroll, Joseph P. Chan, Hon-Wing Cheng, I-Ping Chu, Robert Crawford, Henry D'Angelo, Jerry Delazzer, Br. Michael Driscoll, Carl E. Eckberg, Herber Enderton, Susana Epp, Gerald Gordon, Jerrold Grossman, Reino Hakala, Mark Herbster, Steve Jost, Martin Kalin, Nicholas Krier, Warren Krueger, Glenn Lancaster, Donald E. G. Malm, Nick Meshes, Kevin Phelps, Jenni Piane, Mansur Samadzadeh, Sigrid (Anne) Settle, James H. Stoddard, Chaim Goodman Strauss, Michael Sullivan, Edward J. Williams y Hanyi Zhang. Agradezco también a todos los usuarios de mi libro por sus valiosas cartas y correos electrónicos.

★

Un agradecimiento especial por esta edición es para George F. Bachelis, de Wayne State University, por las correcciones y la retroalimentación de su grupo de alumnos, y para Bob Fisher, mi colega en DePaul, por atraer mi atención a algunos ejercicios agradables de conjuntos convexos que se pueden resolver usando inducción matemática.

Por la revisión del manuscrito de esta edición, doy gracias a:

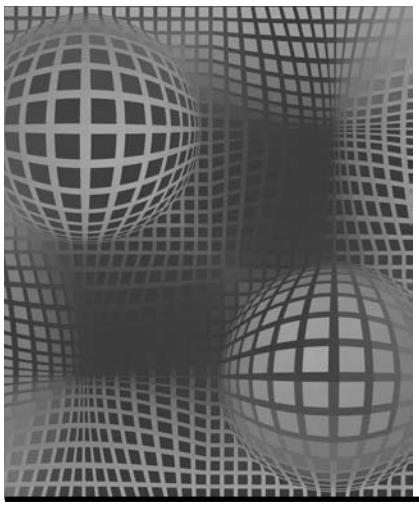
Scott Annin, California State University, Fullerton
Brendan Frey, University of Toronto
Dennis Garity, Oregon State University
Aaron Keen, California Polytechnic State University, San Luis Obispo
Miguel Lerma, Northwestern University
Truc Nguyen, Bowling Green State University
Craig Jensen, University of New Orleans
Randall Pruim, Calvin College
David Stewart, University of Iowa
Suely Oliveira, University of Iowa
Bogdan Suceava, California State University, Fullerton
Anthony S. Wojcik, Michigan State University

Agradezco a mi amable editora, Patricia Johnsonbaugh, por marcar cada uno de los cerca de 4000 ejercicios con codificación mística, por cuidar numerosos detalles, detectar texto que no pretendí escribir, mejorar la exposición y sugerir cambios que mejoraron el libro.

Estoy en deuda con Helmut Epp, decano de la Escuela de ciencias de la computación, telecomunicaciones y sistemas de información en DePaul University, por proporcionarme su tiempo y alentarme en el desarrollo de esta edición y las anteriores.

He recibido apoyo constante del personal de Prentice Hall. En especial, agradezco a George Lobell, editor ejecutivo, por su ayuda; a Jennifer Brady, asistente editorial, y a Debbie Ryan, supervisor de producción.

Richard Johnsonbaugh



Capítulo 1

LÓGICA Y DEMOSTRACIONES

WWW

Lógica, lógica, lógica. La lógica es el principio de la sabiduría, Valeria, no el fin.

STAR TREK VI: EL PAÍS SIN DESCUBRIR

Lógica es el estudio del razonamiento; se refiere específicamente a si el razonamiento es correcto. La lógica se centra en la relación entre las afirmaciones y no en el contenido de una afirmación en particular. Considere, por ejemplo, el siguiente argumento:

Todos los matemáticos usan sandalias

Cualquiera que use sandalias es un algebrista

Por lo tanto, todos los matemáticos son algebristas

En el sentido técnico, la lógica no ayuda a determinar si alguna de estas afirmaciones es cierta; sin embargo, si las primeras dos afirmaciones son ciertas, la lógica asegura que la afirmación

todos los matemáticos son algebristas

también es cierta.

Los métodos lógicos se usan en matemáticas para demostrar teoremas y, en las ciencias de la computación, para probar que los programas hacen lo que deben hacer. Suponga, por ejemplo, que se asigna a un estudiante el desarrollo de un programa para calcular las trayectorias más cortas entre ciudades. Es necesario que el programa acepte como entrada un número arbitrario de ciudades y las distancias entre las ciudades con conexión directa por carretera, y que produzca como salida las trayectorias (rutas) más cortas entre cada par distinto de ciudades. Después de escribir el programa, es fácil para el estudiante probarlo con un número reducido de ciudades. Con papel y lápiz, puede enumerar todas las trayectorias posibles entre pares de ciudades y encontrar las más cortas. Esta solución por "fuerza bruta" se compara con la salida del programa. Sin embargo, para un número grande de ciudades, la técnica de la "fuerza bruta" sería tardada. ¿Cómo puede el estudiante estar seguro de que el programa trabaja bien para muchos datos (casi seguro el tipo de entrada con la que el profesor probaría el programa)? Él tendrá que usar la *lógica* para argumentar que el programa es correcto. El argumento puede ser informal o formal usando las técnicas presentadas en este capítulo; pero se requerirá un argumento lógico.

Entender la lógica también resulta útil para aclarar la escritura común. Por ejemplo, en una ocasión, se publicó el siguiente decreto en Naperville, Illinois: "Será ilegal que una

† Esta sección se puede omitir sin pérdida de continuidad.

persona tenga más de tres perros y tres gatos en su propiedad dentro de la ciudad". Un ciudadano que tenía cinco perros y ningún gato, ¿violaba el decreto? Piense en esta pregunta ahora y analícela (vea ejercicio 54, sección 1.1) después de leer la sección 1.1.

1.1 ➜ Proposiciones

¿Cuáles oraciones de la *a)* a la *e)* son verdaderas o falsas (pero no ambas)?

a) Los únicos enteros positivos que dividen† a 7 son 1 y el mismo 7.
b) Alfred Hitchcock ganó un premio de la Academia en 1940 por la dirección de "Rebeca".
c) Para todo entero positivo n, existe un número primo‡ mayor que n.
d) La Tierra es el único planeta en el universo que tiene vida.
e) Compra dos boletos para el concierto de rock "Unhinged Universe" del viernes.

La oración *a)*, que es otra manera de decir que el 7 es primo, es verdadera.

La oración *b)* es falsa. Aunque "Rebeca" ganó el premio de la Academia por la mejor película de 1940, John Ford ganó el premio por dirigir "Las viñas de la ira". Es un hecho sorprendente que Alfred Hitchcock nunca haya ganado un premio de la Academia por mejor dirección.

La oración *c)*, que es otra forma de decir que el número de primos es infinito, es verdadera.

La oración *d)* puede ser verdadera o falsa (pero no ambas), sin embargo en este momento se ignora.

La oración *e)* no es verdadera ni falsa [esta oración es una orden].

Una oración que es verdadera o falsa, pero no ambas, se llama una **proposición.** Las oraciones *a)* a la *d)* son proposiciones, mientras que la oración *e)* no es una proposición. Es común que una proposición se exprese como una oración declarativa (y no como pregunta, orden, exclamación, etcétera). Las proposiciones son los bloques básicos de construcción de cualquier teoría de lógica.

Se usarán variables, como p, q y r, para representar las proposiciones, casi como se usan letras en álgebra para representar números. También se usará la notación

$$p\text{: } 1 + 1 = 3$$

para definir que p es la proposición $1 + 1 = 3$.

Al hablar y escribir de forma normal, las proposiciones se combinan usando conectores como *y* y *o*. Por ejemplo, las proposiciones "está lloviendo" y "hace frío" se pueden combinar para formar la proposición "está lloviendo y hace frío". A continuación se dan las definiciones formales de *y* y *o*.

Definición 1.1.1 ▶

Sean p y q proposiciones.

La *conjunción* de p y q, denotada por $p \wedge q$, es la proposición

$$p \quad \text{y} \quad q.$$

La *disyunción* de p y q, denotada por $p \vee q$, es la proposición

$$p \quad \text{o} \quad q.$$

◀

Un *operador binario* sobre un conjunto* X, asigna a cada par de elementos en X un elemento de X (vea la definición 2.2.44). El operador $\wedge$ asigna a cada par de proposiciones

†"Divide" se refiere a "división exacta". De manera más formal, se dice que un entero diferente de cero *d divide* a un entero m si existe un entero q tal que $m = dq$. A q se le llama el *cociente*. Se explorarán los enteros con detalle en el capítulo 5.

‡ Un entero $n > 1$ es *primo* si los únicos enteros positivos que dividen a n son 1 y el mismo n. Por ejemplo, 2, 3 y 11 son números primos.

* Un *conjunto* es una colección de objetos. Por ejemplo, el conjunto de enteros positivos consiste en los enteros 1, 2, … Los "conjuntos" se estudian con detalle en la sección 2.1.

p y q la proposición $p \wedge q$. Entonces, $\wedge$ es un operador binario sobre las proposiciones. El operador $\vee$ también es un operador binario sobre las proposiciones.

Ejemplo 1.1.2 ▶

Si

p: Está lloviendo,

q: Hace frío,

entonces la conjunción de p y q es

$p \wedge q$: Está lloviendo y hace frío.

La disyunción de p y q es

$p \vee q$: Está lloviendo o hace frío. ◀

El valor de verdad de la conjunción $p \wedge q$ está determinado por los valores verdaderos de p y q, y la definición se basa en la interpretación usual de "y". Considere la proposición

$p \wedge q$: Está lloviendo y hace frío

del ejemplo 1.1.2. Si está lloviendo (es decir, p es verdadera) y también hace frío (es decir, q también es verdadera), entonces la proposición

$p \wedge q$: Está lloviendo y hace frío

se consideraría verdadera. Sin embargo, si no está lloviendo (esto es, p es falsa) o si no hace frío (q es falsa) o ambas, entonces la proposición

$p \wedge q$: Está lloviendo y hace frío

se consideraría falsa.

Los valores de verdad de las proposiciones, tales como conjunciones o disyunciones, se pueden describir por las **tablas de verdad**. La tabla de verdad de una proposición *P,* formada por las proposiciones individuales $p_1, \ldots, p_n$, enumera todas las posibles combinaciones de los valores de verdad para $p_1, \ldots, p_n$, donde V denota verdadero y F denota falso, y da la lista de valores de verdad de P para cada combinación. Se usa una tabla de verdad para dar la definición formal de los valores de verdad de $p \wedge q$.

Definición 1.1.3 ▶

Los valores de verdad de la proposición $p \wedge q$ se definen por la tabla de verdad

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

◀

Observe que en la tabla de verdad de la definición 1.1.3 se dan las cuatro combinaciones posibles (cuatro alternativas posibles) de asignaciones de verdad para p y q.

La definición 1.1.3 establece que la conjunción $p \wedge q$ es verdadera siempre que p y q sean ambas verdaderas; de otra manera, $p \wedge q$ es falsa.

Ejemplo 1.1.4 ▶

Si

p: Una década tiene 10 años,

q: Un milenio tiene 100 años,

entonces p es verdadera, q es falsa (un milenio tiene 1000 años) y la conjunción

$p \wedge q$: Una década tiene 10 años y un milenio tiene 100 años

es falsa. ◀

Ejemplo 1.1.5 ▶

Casi todos los lenguajes de programación definen "y" justo como la definición 1.1.3. Por ejemplo, en el lenguaje de programación Java, el "y" (lógico) se denota por `&&`, y la expresión

```
x < 10 && y > 4
```

es verdadera precisamente cuando el valor de la variable `x` es menor que 10 (esto es, `x < 10` es cierta) y el valor de la variable y es mayor que 4 (es decir, `y > 4` se cumple). ◀

El valor de verdad de la disyunción $p \vee q$ también está determinado por los valores de verdad de p y q, y la definición se basa en la interpretación "inclusiva" de "o". Considere la proposición

$p \vee q$: Está lloviendo o hace frío,

del ejemplo 1.1.2. Si está lloviendo (es decir, p es verdadera) o si hace frío (es decir, q es verdadera) *o ambas*, entonces se consideraría que la proposición

$p \vee q$: Está lloviendo o hace frío

es verdadera (esto es, $p \vee q$ es verdadera). El **or-inclusivo** de las proposiciones p y q es verdadero si ambas, p y q, son verdaderas. Si no está lloviendo (o sea, p es falsa) y si no hace frío (q también es falsa), entonces se consideraría que la proposición

$p \vee q$: Está lloviendo o hace frío,

es falsa (esto es, $p \vee q$ es falsa). También existe el **or-exclusivo** (vea el ejercicio 53) que define *p exor q* como falsa si ambas, p y q, son verdaderas.

Definición 1.1.6 ▶

El valor de verdad de la proposición $p \vee q$ se define por la tabla de verdad

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

◀

La definición 1.1.6 establece que la disyunción $p \vee q$ es verdadera siempre que p o q (o ambas) sean verdaderas; de otra manera, $p \vee q$ será falsa (es decir, sólo si p y q son falsas la disyunción será falsa).

Ejemplo 1.1.7 ▶

Si

p: Un milenio tiene 100 años,

q: Un milenio tiene 1000 años,

entonces p es falsa, q es verdadera y la disyunción

$p \vee q$: Un milenio tiene 100 años o un milenio tiene 1000 años

es verdadera. ◀

Ejemplo 1.1.8 ▶

Casi todos los lenguajes de programación definen un or (inclusivo) justo como en la definición 1.1.6. Por ejemplo, en Java, el or (lógico) se denota por `||` y la expresión

```
x < 10  ||  y > 4
```

es verdadera precisamente cuando el valor de la variable `x` es menor que 10 (esto es, `x < 10` es cierta) o el valor de la variable *y* es mayor que 4 (es decir, `y > 4` se cumple) o ambas. ◀

En el lenguaje común, las proposiciones que se combinan (es decir, p y q combinadas para dar la proposición $p \vee q$) suelen estar relacionadas; pero en lógica, no se requiere que estas proposiciones hagan referencia al mismo asunto. Por ejemplo, en lógica se permiten proposiciones como

3 < 5 o París es la capital de Inglaterra.

La lógica se ocupa de la forma de las proposiciones y de la relación de las proposiciones entre sí, no del tema. (La proposición anterior es verdadera porque $3 < 5$ es verdadera).

El operador final en una proposición p que analizamos en esta sección es la **negación** de p.

Definición 1.1.9 ▶

La *negación* de p, denotada por $\neg p$, es la proposición

no p.

El valor de verdad de esta proposición $\neg p$ se define por la tabla de verdad

p	$\neg p$
V	F
F	V

◀

Algunas veces escribimos $\neg p$ para decir "no ocurre que p". Por ejemplo, si

p: París es la capital de Inglaterra,

la negación de p se escribe como

$\neg p$: No ocurre que París es la capital de Inglaterra.

o más fácil como

$\neg p$: París no es la capital de Inglaterra.

Un *operador unario* sobre un conjunto X asigna a cada elemento de X un elemento de X (vea la definición 2.2.46). El operador $\neg$ asigna a cada proposición p la proposición $\neg p$. Entonces, $\neg$ es un operador unario sobre las proposiciones.

Ejemplo 1.1.10 ▶

Si

p: π se calculó con 1,000,000 de dígitos decimales en 1954,

la negación de p es la proposición

$\neg p$: π no se calculó con 1,000,000 de dígitos decimales en 1954.

No fue sino hasta 1973 que se calculó π con 1,000,000 de dígitos decimales; entonces p es falsa. (Desde entonces se han calculado más de 200 mil millones de dígitos decimales de π). Puesto que p es falsa, $\neg p$ es verdadera. ◀

Ejemplo 1.1.11 ▶

Casi todos los lenguajes de programación definen "no" justo como en la definición 1.1.9. Por ejemplo, en Java el "no" se denota por !, y la expresión

```
! (x < 10)
```

es verdadera precisamente cuando el valor de la variable `x` no es menor que 10 (es decir, `x` es mayor que o igual a 10). ◀

En las expresiones que incluyen algunos o todos los operadores $\neg$, $\wedge$ y $\vee$, en la ausencia de paréntesis, primero se evalúa $\neg$, después $\wedge$ y luego $\vee$. Esta convención se conoce como **precedencia del operador**. En álgebra, la precedencia del operador indica que se evalúan $\cdot$ y / antes que $+$ y $-$.

Ejemplo 1.1.12 ▶

Puesto que la proposición p es falsa, la proposición q es verdadera y la proposición r es falsa, determine si la proposición

$$\neg p \vee q \wedge r$$

es falsa o verdadera.

Primero se evalúa $\neg p$, que es verdadera. Después se evalúa $q \wedge r$, que es falsa. Por último, se evalúa

$$\neg p \vee q \wedge r$$

que es verdadera. ◀

Ejemplo 1.1.13 ▶

Búsqueda en Internet

Se dispone de gran variedad de herramientas de búsqueda en Internet (como AltaVista, Google, Yahoo) que permiten al usuario introducir palabras clave que el portal de búsqueda intenta igualar con páginas Web. Por ejemplo, introducir *matemáticas* produce una lista (¡enorme!) de páginas que contienen la palabra "matemáticas". Algunos sitios de búsqueda permiten al usuario incluir operadores como *AND*, *OR* y *NOT* (*y*, *o* y *no*) junto con paréntesis para combinar las palabras clave (vea la figura 1.1.1), lo que admite búsquedas

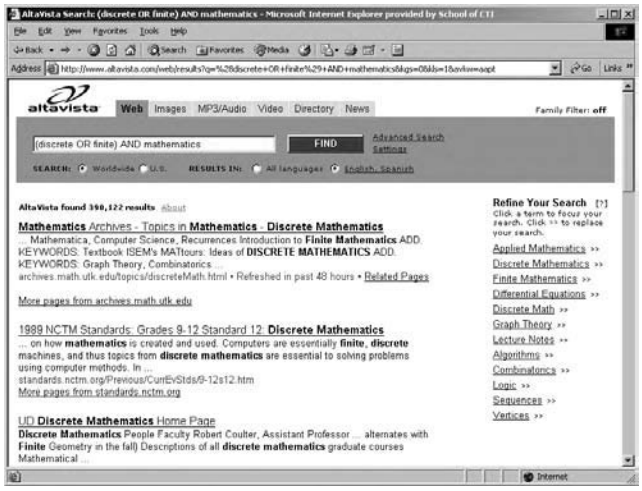


Figura 1.1.1 El portal de búsqueda AltaVista permite al usuario introducir expresiones con *AND*, *OR* y *NOT* junto con paréntesis. (En AltaVista, NOT debe ir precedido de otro operador como AND). En la figura, el usuario busca páginas que contengan "discrete mathematics" o "finite mathematics" ("matemáticas discretas" o "matemáticas finitas") escribiendo *(discrete OR finite) AND mathematics*. Como se muestra, AltaVista encontró cerca de 390,000 páginas de Internet que contienen matemáticas discretas o matemáticas finitas.

más complejas. Por ejemplo, para buscar páginas que contengan las palabras clave "discretas" *y* "matemáticas", el usuario escribiría *discretas AND matemáticas*. Para buscar páginas con las palabras clave "discretas" *y* "matemáticas" *o* las palabras clave "finitas" *y* "matemáticas", el usuario podría introducir (*discretas OR finitas) AND matemáticas*. ◀

Sugerencias para resolver problemas

Aunque tal vez haya un camino más corto para determinar los valores de verdad de una proposición P formada al combinar las proposiciones $p_1, \ldots, p_n$ usando operadores como $\neg$ y $\vee$, la tabla de verdad siempre proporcionará todos los valores de verdad posibles de P para diferentes valores de las proposiciones que la constituyen $p_1, \ldots, p_n$.

Sección de ejercicios de repaso

†***1.*** ¿Qué es una proposición?

2. ¿Qué es una tabla de verdad?

3. ¿Qué es la conjunción de p y q? ¿Cómo se denota?

4. Proporcione la tabla de verdad para la conjunción de p y q.

5. ¿Qué es la disyunción de p y q? ¿Cómo se denota?

6. Proporcione la tabla de verdad para la disyunción de p y q.

7. ¿Qué es la negación de p? ¿Cómo se denota?

8. Proporcione la tabla de verdad para la negación de p.

Ejercicios

Determine si cada oración en los ejercicios 1 a 8 es una proposición. Si la oración es una proposición, escriba su negación. (No se piden los valores de verdad de las oraciones que son proposiciones).

1. $2 + 5 = 19$.

2. Mesero, ¿serviría las nueces, quiero decir, serviría las nueces a los invitados?

3. Para algún entero positivo n, $19340 = n \cdot 17$.

4. Audrey Meadows fue la "Alice" original de la serie "The Honeymooners".

5. Pélame una uva.

6. La línea "Tócala otra vez, Sam" corresponde a la película "Casablanca".

7. Todo entero par mayor que 4 es la suma de dos primos.

8. La diferencia de dos primos.

Los ejercicios 9 a 12 se refieren a una moneda que se lanza 10 veces. Escriba la negación de la proposición.

9. Salieron 10 caras.

10. Salieron algunas caras.

11. Salieron algunas caras y algunas cruces.

12. Salió al menos una cara.

Puesto que la proposición p es falsa, la proposición q es verdadera y la proposición r es falsa, determine si cada proposición en los ejercicios 13 a 18 es falsa o verdadera.

13. $p \vee q$

14. $\neg p \vee \neg q$

15. $\neg p \vee q$

16. $\neg p \vee \neg(q \wedge r)$

17. $\neg(p \vee q) \wedge (\neg p \vee r)$

18. $(p \vee \neg r) \wedge \neg((q \vee r) \vee \neg(r \vee p))$

Escriba la tabla de verdad de cada proposición en los ejercicios 19 a 26.

19. $p \wedge \neg q$

20. $(\neg p \vee \neg q) \vee p$

21. $(p \vee q) \wedge \neg p$

22. $(p \wedge q) \wedge \neg p$

23. $(p \wedge q) \vee (\neg p \vee q)$

24. $\neg(p \wedge q) \vee (r \wedge \neg p)$

25. $(p \vee q) \wedge (\neg p \vee q) \wedge (p \vee \neg q) \wedge (\neg p \vee \neg q)$

26. $\neg(p \wedge q) \vee (\neg q \vee r)$

En los ejercicios 27 a 29, represente la proposición indicada simbólicamente definiendo

$$p\text{: } 5 < 9,\ \ q\text{: } 9 < 7,\ \ r\text{: } 5 < 7.$$

Determine si cada proposición es verdadera o falsa.

27. $5 < 9$ y $9 < 7$.

28. No ocurre que ($5 < 9$ y $9 < 7$).

29. $5 < 9$ o no ocurre que ($9 < 7$ y $5 < 7$).

En los ejercicios 30 a 35, formule la expresión simbólica en palabras usando

p: Leo toma ciencias de la computación.

q: Leo toma matemáticas.

30. $\neg p$

31. $p \wedge q$

32. $p \vee q$

33. $p \vee \neg q$

34. $p \wedge \neg q$

35. $\neg p \wedge \neg q$

En los ejercicios 36 a 40, formule la expresión simbólica en palabras usando

p: Hoy es lunes.

q: Está lloviendo.

r: Hace calor.

36. $p \vee q$

37. $\neg p \wedge (q \vee r)$

38. $\neg(p \vee q) \wedge r$

39. $(p \wedge q) \wedge \neg(r \vee p)$

† Los números de ejercicios en cursivas indican que se da una sugerencia o la solución al final del libro, después de la sección de referencias

40. $(p \wedge (q \vee r)) \wedge (r \vee (q \vee p))$

En los ejercicios 41 a 46, represente simbólicamente la proposición definiendo

p: *Hay huracán.*

q: *Está lloviendo.*

41. No hay huracán.

42. Hay huracán y está lloviendo.

43. Hay huracán, pero no está lloviendo.

44. No hay huracán y no está lloviendo.

45. Hay huracán o está lloviendo (o ambas).

46. Hay huracán o está lloviendo, pero no hay huracán.

En los ejercicios 47 a 52, represente simbólicamente la proposición definiendo

p: *Oíste el concierto de rock de "Flying Pigs".*

q: *Oíste el concierto de rock de "Y2K".*

r: *Tienes los tímpanos inflamados.*

47. Oíste el concierto de rock de "Flying Pigs" y tienes los tímpanos inflamados.

48. Oíste el concierto de rock de "Flying Pigs", pero no tienes los tímpanos inflamados.

49. Oíste el concierto de rock de "Flying Pigs", oíste el concierto de rock de "Y2K" y tienes los tímpanos inflamados.

50. Oíste el concierto de rock de "Flying Pigs" o el concierto de rock de "Y2K", pero no tienes los tímpanos inflamados.

51. No oíste el concierto de rock de "Flying Pigs" y no oíste el concierto de rock de "Y2K", pero tienes los tímpanos inflamados.

52. No ocurre que: oíste el concierto de rock de "Flying Pigs" o bien oíste el concierto de rock de "Y2K" o no tienes los tímpanos inflamados.

53. Proporcione una tabla de verdad para el or-exclusivo de *p* y *q* donde *p exor q* es verdadera si *p* o *q*, pero no ambas, son verdaderas.

54. En una ocasión se publicó el siguiente decreto en Naperville, Illinois: "Será ilegal que una persona tenga más de tres [3] perros y tres [3] gatos en su propiedad dentro de la ciudad". El señor Charles Marko tenía cinco perros y ningún gato, ¿violaba el decreto? Explique.

55. Escriba las instrucciones de búsqueda en Internet para encontrar parques nacionales en Dakota del Sur o del Norte.

56. Escriba las instrucciones de búsqueda en Internet para obtener información de enfermedades pulmonares que no sean cáncer.

57. Escriba las instrucciones de búsqueda en Internet para ver equipos de béisbol de las ligas menores que estén en la Liga del Medio Oeste.

1.2 ➜ Proposiciones condicionales y equivalencia lógica

El decano de la escuela anunció que

Si el departamento de matemáticas obtiene \$40,000 adicionales,
entonces contratará un nuevo académico. **(1.2.1)**

La afirmación (1.2.1) establece que con la condición de que el departamento de matemáticas obtenga \$40,000 adicionales, entonces contratará un nuevo académico. Este tipo de proposición se conoce como **proposición condicional.**

Definición 1.2.1 ▶

Si *p* y *q* son proposiciones, la proposición

si *p* entonces *q* **(1.2.2)**

se llama *proposición condicional* y se denota por

$$p \rightarrow q$$

La proposición *p* se llama *hipótesis* (o *antecedente*) y la proposición *q* recibe el nombre de *conclusión* (o *consecuente*). ◀

Ejemplo 1.2.2 ▶

Si se define

p: El departamento de matemáticas obtiene \$40,000 adicionales,

q: El departamento de matemáticas contrata un nuevo académico,

entonces la proposición (1.2.1) toma la forma (1.2.2). La hipótesis es la afirmación "el departamento de matemáticas obtiene \$40,000 adicionales" y la conclusión es la afirmación "el departamento de matemáticas contrata un nuevo académico". ◀

¿Cuál es el valor de verdad para la afirmación del decano (1.2.1)? Primero, suponga que el departamento de matemáticas obtiene \$40,000 adicionales. Si de hecho contrata otro académico, con seguridad la afirmación del decano es verdadera. (Usando la notación del

ejemplo 1.2.2, si p y q son ambas verdaderas, entonces $p \to q$ es verdadera). Por otra parte, si el departamento de matemáticas obtiene \$40,000 adicionales y *no* contrata un nuevo académico, el decano está equivocado, es decir, la oración (1.2.1) es falsa. (Si p es verdadera y q es falsa, entonces $p \to q$ es falsa). Ahora suponga que el departamento de matemáticas no obtiene \$40,000 adicionales. En este caso, el departamento de matemáticas puede o no contratar otro académico. (Quizá alguien del departamento se jubila y se contrata a alguien más para reemplazarlo. Por otro lado, el departamento puede no contratar a alguien). Por supuesto, no se consideraría falsa la afirmación del decano. Así, si el departamento de matemáticas *no* obtiene los \$40,000, la afirmación del decano debe ser verdadera, sin importar si el departamento contrata o no otro académico. (Si p es falsa, entonces $p \to q$ es verdadera sea q verdadera o falsa). Este análisis motiva la siguiente definición.

Definición 1.2.3 ▶

El valor verdadero de la proposición condicional $p \to q$ está definido por la siguiente tabla de verdad:

p	q	$p \to q$
V	V	V
V	F	F
F	V	V
F	F	V

◀

De manera formal, $\to$ es un operador binario sobre las proposiciones. El operador $\to$ asigna a cada par de proposiciones p y q la proposición $p \to q$.

Para quienes necesitan mayor evidencia de que $p \to q$ se debe definir como verdadera cuando p es falsa, se ofrece otra justificación. Casi todas las personas están de acuerdo en que la proposición

$$\text{Para todos los números reales } x, \text{ si } x > 0, \text{ entonces } x^2 > 0, \tag{1.2.3}$$

es verdadera. (En la sección 1.3 se hará el análisis formal y detallado de afirmaciones del tipo "para todos"). En la siguiente presentación, p denotada por $x > 0$ y q denotada por $x^2 > 0$. El hecho de que la proposición (1.2.3) sea verdadera significa que no importa con cuál número real se sustituya x, la proposición

$$\text{si } p \text{ entonces } q \tag{1.2.4}$$

resultante es verdadera. Por ejemplo, si $x = 3$, entonces p y q son ambas ciertas ($3 > 0$ y $3^2 > 0$ son ambas verdaderas) y, por la definición 1.2.3, (1.2.4) es verdadera. Ahora considere la situación donde p es falsa. Si $x = -2$, entonces p es falsa ($-2 > 0$ es falsa) y q es verdadera [$(-2)^2 > 0$ es verdadera]. Con objeto de que la proposición (1.2.4) sea verdadera en ese caso, debe definirse $p \to q$ como verdadera cuando p es falsa y q es verdadera. Esto es justo lo que ocurre en el tercer renglón de la tabla de verdad para la definición 1.2.3. Si $x = 0$, entonces p y q son ambas falsas ($0 > 0$ y $0^2 > 0$ son falsas). Para que la proposición (1.2.4) sea cierta en este caso, debe definirse $p \to q$ como verdadera cuando p y q son ambas falsas. Justo ocurre esto en el cuarto renglón de la tabla de verdad para la definición 1.2.3. En los ejercicios 52 y 53 se da una mayor motivación para definir $p \to q$ como verdadera cuando p es falsa.

Ejemplo 1.2.4 ▶

Sea

$$p\text{: } 1 > 2, \qquad q\text{: } 4 < 8.$$

Entonces p es falsa y q es verdadera. Por lo tanto,

$$p \to q \text{ es verdadera}, \qquad q \to p \text{ es falsa.}$$

◀

En las expresiones que incluyen a los operadores lógicos $\wedge$, $\vee$, $\neg$ y $\rightarrow$, el operador condicional $\rightarrow$ evalúa al final. Por ejemplo,

$$p \vee q \rightarrow \neg r$$

se interpreta como

$$(p \vee q) \rightarrow (\neg r).$$

Ejemplo 1.2.5 ▶ Suponiendo que p es verdadera, q es falsa y r es verdadera, encuentre el valor de verdad de cada proposición.

a) $p \wedge q \rightarrow r$ *b)* $p \vee q \rightarrow \neg r$ *c)* $p \wedge (q \rightarrow r)$ *d)* $p \rightarrow (q \rightarrow r)$

a) Primero se evalúa $p \wedge q$ porque $\rightarrow$ se evalúa al final. Como p es cierta y q es falsa, $p \wedge q$ es falsa. Por lo tanto, $p \wedge q \rightarrow r$ es verdadera (sin importar si r es cierta o falsa).

b) Primero se evalúa $\neg r$. Como r es verdadera, $\neg r$ es falsa. Después se evalúa $p \vee q$. Como p es verdadera y q es falsa, $p \vee q$ es verdadera. Por lo tanto, $p \vee q \rightarrow \neg r$ es falsa.

c) Como q es falsa, $q \rightarrow r$ es verdadera (sin importar si r es verdadera o falsa). Como p es verdadera, $p \wedge (q \rightarrow r)$ es verdadera.

d) Puesto que q es falsa, $q \rightarrow r$ es verdadera (sin importar si r es verdadera o falsa). Entonces, $p \rightarrow (q \rightarrow r)$ es verdadera (sin importar si p es verdadera o falsa). ◀

Una proposición condicional que es verdadera porque la hipótesis es falsa se dice que es **verdadera por omisión** o **superficialmente verdadera**. Por ejemplo, si la proposición,

> Si el departamento de matemáticas obtiene $40,000 adicionales, entonces contratará un nuevo académico,

es verdadera porque el departamento de matemáticas no obtuvo $40,000 adicionales, se dice que la proposición es verdadera por omisión o que es superficialmente verdadera.

Algunas afirmaciones no de la forma (1.2.2) pueden rescribirse como proposiciones condicionales, como ilustra el siguiente ejemplo.

Ejemplo 1.2.6 ▶ Reescriba cada proposición en la forma (1.2.2) de una proposición condicional.

a) María será una buena estudiante si estudia mucho.

b) Juan toma cálculo sólo si está en 2º, 3º o 4º grado de universidad.

c) Cuando cantas, me duelen los oídos.

d) Una condición necesaria para que los Cachorros ganen la Serie Mundial es que contraten a un pitcher suplente diestro.

e) Una condición suficiente para que María visite Francia es ir a la Torre Eiffel.

a) La hipótesis es la cláusula que sigue a *si*; entonces una formulación equivalente es

> Si María estudia mucho, entonces será una buena estudiante.

b) La afirmación significa que para que Juan tome cálculo debe estar en 2º, 3º o 4º año de universidad. En particular, si está en 1º, *no* puede tomar cálculo. Así, se concluye que si toma cálculo, entonces está en 2º, 3º o 4º año. Por lo tanto, una formulación equivalente sería

> Si Juan toma cálculo, entonces está en 2º, 3º o 4º año.
>
> Observe que
>
> Si Juan está en 2º, 3º o 4º año, entonces toma cálculo,

no es una formulación equivalente. Si Juan está en 2º, 3º o 4º año, puede o *no* tomar cálculo. (Aunque sea elegible para tomar cálculo, puede decidir no tomarlo).

La formulación "si p entonces q" hace hincapié en la hipótesis mientras que la formulación "p sólo si q" resalta la conclusión; la diferencia es nada más de estilo.

c) *Cuando* significa lo mismo que *si*; entonces una formulación equivalente es

Si cantas, me duelen los oídos.

d) Una **condición necesaria** es sólo eso: una condición que *se necesita* para lograr un resultado en particular. La condición *no* garantiza el resultado; pero si no se cumple, el resultado no se logrará. Aquí, la afirmación significa que si los Cachorros ganan la Serie Mundial, podemos estar seguros de que contrataron un pitcher suplente diestro, ya que sin ese contrato no habrían ganado. Así, una formulación equivalente de la afirmación es

Si los Cachorros ganan la Serie Mundial, entonces contrataron un pitcher suplente diestro.

La conclusión expresa una condición necesaria.
Observe que

Si los Cachorros contratan un pitcher suplente diestro, entonces ellos ganan la Serie Mundial,

no es una formulación equivalente. Contratar un pitcher suplente diestro no es garantía de que ganarán la Serie Mundial. Sin embargo, *no* contratarlo garantiza que no ganarán la Serie Mundial.

e) De manera similar, una **condición suficiente** es una condición que *basta* para garantizar un resultado en particular. Si la condición no se cumple, el resultado puede lograrse de otras formas o tal vez no se logre; pero si la condición se cumple, el resultado está garantizado. Aquí, para asegurar que María visite Francia, basta con que vaya a la Torre Eiffel. (Sin duda, hay otras maneras de asegurar que María visite Francia; por ejemplo, podría ir a Lyon). Así, una formulación equivalente a la afirmación en cuestión es

Si María va a la Torre Eiffel, entonces visita Francia.

La hipótesis expresa una condición suficiente.
Observe que

Si María visita Francia, entonces va a la Torre Eiffel,

no es una formulación equivalente. Como se observó, hay otras maneras de asegurar que María visite Francia que ir a la Torre Eiffel. ◀

El ejemplo 1.2.4 muestra que la proposición $p \rightarrow q$ puede ser verdadera mientras que la proposición $q \rightarrow p$ es falsa. La proposición $q \rightarrow p$ se llama la **recíproca** de la proposición $p \rightarrow q$. Así, una proposición condicional puede ser verdadera mientras que su recíproca es falsa.

Ejemplo 1.2.7 ▶

Escriba la proposición condicional

Si Jesús recibe una beca, entonces irá a la universidad,

y su recíproca en símbolos y en palabras. Además, suponga que Jesús no recibe la beca, pero gana la lotería y de todas formas va a la universidad, encuentre entonces el valor de verdad de la proposición original y su recíproca.

Sea

p: Jesús recibe una beca,

q: Jesús va la universidad.

La proposición se escribe en símbolos como $p \rightarrow q$. Como la hipótesis p es falsa, la proposición condicional es verdadera.

La recíproca de la proposición es

Si Jesús va a la universidad, entonces recibe una beca.

La recíproca se escribe en símbolos como $q \to p$. Puesto que la hipótesis q es verdadera y la conclusión p es falsa, la recíproca es falsa. ◀

Otra proposición útil es

$$p \text{ si y sólo si } q,$$

que se considera verdadera precisamente cuando p y q tienen el mismo valor de verdad (es decir, si p y q son ambas verdaderas o ambas falsas).

Definición 1.2.8 ▶

Si p y q son proposiciones, la proposición

$$p \text{ si y sólo si } q$$

se llama *proposición bicondicional* y se denota por

$$p \leftrightarrow q.$$

El valor de verdad de la proposición $p \leftrightarrow q$ se define por la siguiente tabla de verdad:

p	q	$p \leftrightarrow q$
V	V	V
V	F	F
F	V	F
F	F	V

◀

El operador $\leftrightarrow$ también es un operador binario sobre las proposiciones. Asigna a cada par de proposiciones p y q la proposición $p \leftrightarrow q$.

Una manera alternativa de establecer "p si y sólo si q" es "p es una condición necesaria y suficiente para q". La proposición "p si y sólo si q" algunas veces se escribe p ssi q.

Ejemplo 1.2.9 ▶

La proposición

$$1 < 5 \text{ si y sólo si } 2 < 8 \qquad \textbf{(1.2.5)}$$

se escribe en símbolos como

$$p \leftrightarrow q$$

si se define

$$p\colon\ 1 < 5, \qquad q\colon\ 2 < 8$$

Puesto que ambas, p y q, son verdaderas, la proposición $p \leftrightarrow q$ es verdadera. ◀

Una manera alternativa de establecer (1.2.5) es: Una condición necesaria y suficiente para que $1 < 5$ es que $2 < 8$.

En algunos casos, dos proposiciones diferentes tienen los mismos valores de verdad sin importar qué valores de verdad tengan las proposiciones que las constituyen. Tales proposiciones se conocen como **equivalentes lógicos.**

Definición 1.2.10 ▶

Suponga que las proposiciones P y Q están formadas por las proposiciones $p_1, \ldots, p_n$. Se dice que P y Q son *equivalentes lógicos* y se escriben

$$P \equiv Q$$

siempre que, a partir de cualesquiera valores de verdad de $p_1, \ldots, p_n$, o bien P y Q son ambas verdaderas o P y Q son ambas falsas. ◀

Ejemplo 1.2.11 ▶

Leyes de De Morgan para lógica

Se verificará la primera de las **leyes de De Morgan**

$$\neg(p \vee q) \equiv \neg p \wedge \neg q, \quad \neg(p \wedge q) \equiv \neg p \vee \neg q,$$

WWW y se dejará la segunda como ejercicio (vea el ejercicio 54).

Si se escriben las tablas de verdad para $P = \neg(p \vee q)$ y $Q = \neg p \wedge \neg q$, se puede verificar que, a partir de cualesquiera valores de verdad para p y q, P y Q son ambas verdaderas o P y Q son ambas falsas.

p	q	$\neg(p \vee q)$	$\neg p \wedge \neg q$
V	V	F	F
V	F	F	F
F	V	F	F
F	F	V	V

Entonces, P y Q son equivalentes lógicos. ◀

Ejemplo 1.2.12 ▶

Demuestre que, en Java, las expresiones

```
x < 10  ||  x > 20
```

y

```
!(x >= 10 && x <= 20)
```

son equivalentes. (En Java, `>=` significa ≥ y `<=` significa ≤.)

Si p denota la expresión `x >= 10` y q denota la expresión `x <= 20`, la expresión `!(x >= 10 && x <= 20)` se convierte en $\neg(p \wedge q)$. Por la segunda ley de De Morgan, $\neg(p \wedge q)$ es equivalente a $\neg p \vee \neg q$. Como $\neg p$ se traduce como x < 10 y $\neg q$ se traduce como `x > 20`, $\neg p \vee \neg q$ se traducen como `x < 10 || x > 20`. Por lo tanto, las expresiones `x < 10 || x > 20` y `!(x >= 10 && x <= 20)` son equivalentes. ◀

El siguiente ejemplo da una forma de equivalencia lógica para la negación de $p \rightarrow q$.

Ejemplo 1.2.13 ▶

Demuestre que la negación de $p \rightarrow q$ es equivalente lógico de $p \wedge \neg q$.

Debe demostrarse que

$$\neg\ (p \rightarrow q) \equiv p \wedge \neg q.$$

Al escribir las tablas de verdad para $P = \neg(p \rightarrow q)$ y $Q = p \wedge \neg q$, se puede verificar que, a partir de cualesquiera valores de verdad de p y q, o bien P y Q son ambas verdaderas o P y Q son ambas falsas:

p	q	$\neg(p \rightarrow q)$	$p \wedge \neg q$
V	V	F	F
V	F	V	V
F	V	F	F
F	F	F	F

Entonces P y Q son equivalentes lógicos. ◀

Ejemplo 1.2.14 ▶

Use la equivalencia lógica de $\neg(p \rightarrow q)$ y $p \wedge \neg q$ (vea el ejemplo 1.2.13) para escribir la negación de

Si Jesús recibe una beca, entonces va la universidad,

con símbolos y en palabras.

Sean

p: Jesús recibe una beca,

q: Jesús va la universidad.

La proposición se escribe con símbolos como $p \to q$. Su negación lógicamente equivalente a $p \wedge \neg q$. En palabras, esta última expresión es

Jesús recibe una beca y no va la universidad. ◀

Ahora se demostrará que, según estas definiciones, $p \leftrightarrow q$ es equivalente lógico de $p \to q$ y $q \to p$. En palabras,

p si y sólo si q

es lógicamente equivalente a

si p entonces q y si q entonces p.

Ejemplo 1.2.15 ▶ La tabla de verdad muestra que

$$p \leftrightarrow q \equiv (p \to q) \wedge (q \to p).$$

p	q	$p \leftrightarrow q$	$p \to q$	$q \to p$	$(p \to q) \wedge (q \to p)$
V	V	V	V	V	V
V	F	F	F	V	F
F	V	F	V	F	F
F	F	V	V	V	V

◀

Se concluye esta sección con la definición de la **contrapositiva** de una proposición condicional. Se verá (en el Teorema 1.2.18) que la contrapositiva es una forma alternativa, lógicamente equivalente de la proposición condicional. El ejercicio 55 da otra forma de equivalente lógico para la proposición condicional.

Definición 1.2.16 ▶ La *contrapositiva* (o *transposición*) de la proposición condicional $p \to q$ es la proposición $\neg q \to \neg p$. ◀

Observe la diferencia entre la contrapositiva y la recíproca. La recíproca de una proposición condicional simplemente invierte los papeles de p y q, mientras que la contrapositiva invierte los papeles de p y q *y* niega cada una de ellas.

Ejemplo 1.2.17 ▶ Escriba la proposición condicional,

Si se cae la red, entonces Darío no puede entrar a Internet,

con símbolos. Escriba la contrapositiva y la recíproca con símbolos y en palabras. Además, suponga que la red no se cayó y que Darío puede entrar a Internet; encuentre los valores de verdad de la proposición original, su contrapositiva y su recíproca.

Sean

p: La red se cae,

q: Darío no puede entrar a Internet.

La proposición se escribe en símbolos como $p \to q$. Como la hipótesis p es falsa, la proposición condicional es verdadera.

La contrapositiva se escribe en símbolos como $\neg q \rightarrow \neg p$ y, en palabras,

Si Darío puede entrar a Internet, entonces la red no se cayó.

Como la hipótesis $\neg q$ y la conclusión $\neg p$ son ambas verdaderas, la contrapositiva es verdadera. (El Teorema 1.2.18 mostrará que la proposición condicional y su contrapositiva son equivalentes lógicos, es decir, que siempre tienen el mismo valor de verdad).

La recíproca de la proposición se escribe simbólicamente como $q \rightarrow p$, y en palabras:

Si Darío no puede entrar a Internet, entonces la red se cayó.

Como la hipótesis q es falsa, la recíproca es cierta. ◀

Un hecho importante es que una proposición condicional y su contrapositiva son equivalentes lógicos.

Teorema 1.2.18

La proposición condicional $p \rightarrow q$ y su contrapositiva $\neg q \rightarrow \neg p$ son equivalentes lógicos.

Demostración La tabla de verdad

p	q	$p \rightarrow q$	$\neg q \rightarrow \neg p$
V	V	V	V
V	F	F	F
F	V	V	V
F	F	V	V

muestra que $p \rightarrow q$ y $\neg q \rightarrow \neg p$ son lógicamente equivalentes.

En lenguaje común, "si" con frecuencia se usa como "si y sólo si". Considere la afirmación

Si arreglas mi computadora, entonces te pagaré $50.

El significado que se pretende transmitir es

Si arreglas mi computadora, entonces te pagaré $50, y
si no la arreglas, entonces no te pagaré $50,

que es lógicamente equivalente a (vea el Teorema 1.2.18).

Si arreglas mi computadora, entonces te pago $50, y
si te pago $50, entonces arreglas mi computadora,

que, a su vez, es el lógico equivalente a (vea ejemplo 1.2.15).

Arreglas mi computadora si y sólo si te pago $50.

En un discurso ordinario, el significado que se pretende para las afirmaciones que incluyen operadores lógicos con frecuencia (¡pero no siempre!) se infiere. Sin embargo, en matemáticas y ciencias, se requiere precisión. Sólo con la definición cuidadosa del significado de los términos, como "si" y "si y sólo si", podremos obtener afirmaciones precisas y sin ambigüedad. En particular, la lógica distingue entre las proposiciones condicional, bicondicional, recíproca y contrapositiva.

Sugerencias para resolver problemas

En la lógica formal, "si" y "si y sólo si" son bastante diferentes. La proposición condicional $p \rightarrow q$ (si p entonces q) es verdadera excepto cuando p es verdadera y q es falsa. Por otro lado, la proposición bicondicional $p \leftrightarrow q$ (p si y sólo si q) es verdadera precisamente cuando p y q son ambas verdaderas o ambas falsas.

Para determinar si las proposiciones P y Q, formadas con las proposiciones $p_1, \ldots, p_n$, son equivalentes lógicos, escriba las tablas de verdad para P y Q. Si todos los elementos son al mismo tiempo verdaderos o falsos para P y Q, entonces P y Q *son* equivalentes. Si algún elemento es verdadero para una de las dos, P o Q, y falso para la otra, entonces P y Q *no* son equivalentes.

Las leyes de De Morgan para lógica

$$\neg(p \vee q) \equiv \neg p \wedge \neg q, \qquad \neg(p \wedge q) \equiv \neg p \vee \neg q$$

dan las fórmulas para negar "o" ($\vee$) y negar "y" ($\wedge$). A grandes rasgos, negar "o" da como resultado "y", lo mismo que al negar "y" se obtiene "o".

El ejemplo 1.2.13 establece una equivalencia muy importante

$$\neg(p \rightarrow q) \equiv p \wedge \neg q,$$

que se encontrará a lo largo del libro. Esta equivalencia muestra que la negación de la proposición condicional se puede escribir usando el operador "y" ($\wedge$). Observe que no aparece el operador condicional en el lado derecho de la ecuación.

Sección de ejercicios de repaso

1. ¿Qué es una proposición condicional? y ¿cómo se denota?
2. Escriba la tabla de verdad para la proposición condicional.
3. En una proposición condicional, ¿cuál es la hipótesis?
4. En una proposición condicional, ¿cuál es la conclusión?
5. ¿Qué es una condición necesaria?
6. ¿Qué es una condición suficiente?
7. ¿Cuál es la recíproca de $p \rightarrow q$?
8. ¿Qué es una proposición bicondicional? y ¿cómo se denota?
9. Escriba la tabla de verdad para la proposición bicondicional.
10. ¿Qué significa para P ser equivalente lógico de Q?
11. Establezca las leyes de De Morgan para lógica.
12. ¿Qué es la contrapositiva de $p \rightarrow q$?

Ejercicios

En los ejercicios 1 a 7, restablezca cada proposición en la forma (1.2.2) de una proposición condicional.

1. José pasará el examen de matemáticas discretas si estudia duro.
2. Rosa se graduará si tiene créditos por 160 horas-trimestre.
3. Una condición necesaria para que Fernando compre una computadora es que obtenga $2000.
4. Una condición suficiente para que Katia tome el curso de algoritmos es que apruebe matemáticas discretas.
5. Cuando se fabriquen mejores automóviles, Buick los fabricará.
6. La audiencia se dormirá si el maestro de ceremonias da un sermón.
7. El programa es legible sólo si está bien estructurado.
8. Escriba la recíproca de cada proposición en los ejercicios 1 al 7.
9. Escriba la contrapositiva de cada proposición en los ejercicios 1 al 7.

Suponiendo que p y r son falsas y que q y s son verdaderas, encuentre el valor de verdad para cada proposición en los ejercicios 10 al 17.

10. $p \rightarrow q$
11. $\neg p \rightarrow \neg q$
12. $\neg(p \rightarrow q)$
13. $(p \rightarrow q) \wedge (q \rightarrow r)$
14. $(p \rightarrow q) \rightarrow r$
15. $p \rightarrow (q \rightarrow r)$
16. $(s \rightarrow (p \wedge \neg r)) \wedge ((p \rightarrow (r \vee q)) \wedge s)$
17. $((p \wedge \neg q) \rightarrow (q \wedge r)) \rightarrow (s \vee \neg q)$

Los ejercicios 18 al 27 se refieren a las proposiciones p, q y r; p es verdadera, q es falsa y el estado de r no se conoce por ahora. Diga si cada proposición es verdadera, falsa o tiene un estado desconocido.

18. $p \vee r$
19. $p \wedge r$
20. $p \rightarrow r$
21. $q \rightarrow r$
22. $r \rightarrow p$
23. $r \rightarrow q$
24. $(p \wedge r) \leftrightarrow r$
25. $(p \vee r) \leftrightarrow r$
26. $(q \wedge r) \leftrightarrow r$
27. $(q \vee r) \leftrightarrow r$

En los ejercicios 28 al 31, represente con símbolos la proposición cuando

$$p\text{: } 4 < 2,\ q\text{: } 7 < 10,\ r\text{: } 6 < 6$$

28. Si $4 < 2$, entonces $7 < 10$.
29. Si ($4 < 2$ y $6 < 6$), entonces $7 < 10$.
30. Si no ocurre que ($6 < 6$ y 7 no es menor que 10), entonces $6 < 6$.
31. $7 < 10$ si y sólo si ($4 < 2$ y 6 no es menor que 6).

En los ejercicios 32 al 37, formule la expresión simbólica en palabras usando

p: Hoy es lunes,
q: Está lloviendo,
r: Hace calor.

32. $p \to q$ **33.** $\neg q \to (r \wedge p)$

34. $\neg p \to (q \vee r)$ **35.** $\neg(p \vee q) \leftrightarrow r$

36. $(p \wedge (q \vee r)) \to (r \vee (q \vee p))$

37 $(p \vee (\neg p \wedge \neg(q \vee r))) \to (p \vee \neg(r \vee q))$

En los ejercicios 38 a 41, escriba cada proposición condicional en símbolos. Escriba la recíproca y la contrapositiva de cada proposición en símbolos y en palabras. Encuentre también el valor de verdad para cada proposición condicional, su recíproca y su contrapositiva.

38. Si $4 < 6$, entonces $9 > 12$. **39.** Si $4 < 6$, entonces $9 < 12$.

40. $|1| < 3$ si $-3 < 1 < 3$. ***41.*** $|4| < 3$ si $-3 < 4 < 3$.

Para cada par de proposiciones P y Q en los ejercicios 42 al 51, establezca si $P \equiv Q$ o no.

42. $P = p, Q = p \vee q$

43. $P = p \wedge q, Q = \neg p \vee \neg q$

44. $P = p \to q, Q = \neg p \vee q$

45. $P = p \wedge (\neg q \vee r), Q = p \vee (q \wedge \neg r)$

46. $P = p \wedge (q \vee r), Q = (p \vee q) \wedge (p \vee r)$

47. $P = p \to q, Q = \neg q \to \neg p$

48. $P = p \to q, Q = p \leftrightarrow q$

49. $P = (p \to q) \wedge (q \to r), Q = p \to r$

50. $P = (p \to q) \to r, Q = p \to (q \to r)$

51. $P = (s \to (p \wedge \neg r)) \wedge ((p \to (r \vee q)) \wedge s), Q = p \vee t$

Los ejercicios 52 y 53 proporcionan mayor motivación para definir $p \to q$ como verdadera cuando p es falsa. Se considera cambiar la tabla de verdad de $p \to q$ cuando p es falsa. Para este primer cambio, el operador resultante recibe el nombre de imp1 (ejercicio 52), y para el segundo cambio el operador resultante es imp2 (ejercicio 53). En ambos casos, se obtienen patologías.

52. Defina la tabla de verdad para *imp1* como

p	q	p *imp1* q
V	V	V
V	F	F
F	V	F
F	F	V

Demuestre que p *imp1* $q \equiv q$ *imp1* p.

53. Defina la tabla de verdad para *imp2* como

p	q	p *imp2* q
V	V	V
V	F	F
F	V	V
F	F	F

a) Demuestre que

$$(p \textit{ imp2 } q) \wedge (q \textit{ imp2 } p) \not\equiv p \leftrightarrow q. \qquad \textbf{(1.2.6)}$$

b) Demuestre que (1.2.6) permanece verdadera si se cambia el tercer renglón de la tabla de verdad de *imp2* a F V F.

54. Verifique la segunda ley de De Morgan, $\neg(p \wedge q) \equiv \neg p \vee \neg q$.

55. Demuestre que $(p \to q) \equiv (\neg p \vee q)$

1.3 ➜ Cuantificadores

WWW

La lógica en las secciones 1.1 y 1.2 referente a proposiciones es incapaz de describir la mayoría de las afirmaciones en matemáticas y en ciencias de la computación. Considere, por ejemplo, la afirmación

p: n es un entero impar

Una proposición es una afirmación que es verdadera o falsa. La afirmación p no es una proposición, porque el hecho de que p sea verdadera o falsa depende del valor de n. Por ejemplo, p es verdadera si $n = 103$ y falsa si $n = 8$. Como casi todas las afirmaciones en matemáticas y ciencias de la computación usan variables, debe ampliarse el sistema de lógica para incluir estas afirmaciones.

Definición 1.3.1 ▶

Sea $P(x)$ una oración que incluye la variable x y sea D un conjunto. P se llama *función proposicional* o *predicado* (respecto a D) si para cada x en D, $P(x)$ es una proposición. D es el *dominio de discurso* (también llamado dominio de referencia) de P. ◀

Ejemplo 1.3.2 ▶

Sea $P(n)$ la afirmación

n es un entero impar,

y sea D el conjunto de enteros positivos. Entonces P es una función proposicional con dominio de discurso D ya que para cada n en D, $P(n)$ es una proposición [es decir, para cada n en D, $P(n)$ es verdadera o falsa pero no ambas]. Por ejemplo, si $n = 1$, se obtiene la proposición

$P(1)$: 1 es un entero impar

(que es verdadera). Si $n = 2$, se obtiene la proposición

$$P(2)\text{: 2 es un entero impar}$$

(que es falsa). ◀

Una proposición P, por sí misma, no es falsa ni verdadera. Sin embargo, para cada x en su dominio de discurso, $P(x)$ es una proposición y es, por lo tanto, verdadera o falsa. Se puede pensar que una función proposicional define una clase de proposiciones, una para cada elemento de su dominio de discurso. Por ejemplo, si P es una función proposicional con dominio de discurso igual al conjunto de enteros positivos, se obtiene una clase de proposiciones

$$P(1), P(2), \ldots.$$

Cada una de las $P(1), P(2), \ldots$ es verdadera o falsa.

Ejemplo 1.3.3 ▶

Las siguientes son funciones proposicionales.

a) $n^2 + 2n$ es un entero impar (dominio de discurso = conjunto de enteros positivos).

b) $x^2 - x - 6 = 0$ (dominio de discurso = conjunto de números reales).

c) El beisbolista bateó más de .300 en 2003 (dominio de discurso = conjunto de beisbolistas).

d) El restaurante tiene más de dos estrellas en la revista *Chicago* (dominio de discurso = restaurantes clasificados en la revista *Chicago*).

En la afirmación *a)*, para cada entero positivo n, se obtiene una proposición; por lo tanto la afirmación *a)* es una función proposicional.

De manera similar, en la afirmación *b)*, para cada número real x, se obtiene una proposición; por lo tanto, la afirmación *b)* es una función proposicional.

Se puede ver a la variable en la afirmación *c)* como "beisbolista". Siempre que se sustituya un beisbolista específico en lugar de la variable, la afirmación es una proposición. Por ejemplo, si se sustituye "Barry Bonds" en lugar de "beisbolista", la afirmación *c)* es

Barry Bonds bateó más de .300 en 2003,

que es verdadera. Si se sustituye "Alex Rodríguez" en lugar de "beisbolista", la afirmación *c)* es

Alex Rodríguez bateó más de .300 en 2003,

que es falsa. Así, la afirmación *c)* es una función proposicional.

La afirmación *d)* es similar en la forma a *c)*; aquí la variable es "restaurante". Al sustituir la variable por un restaurante clasificado en la revista *Chicago*, la afirmación es una proposición. Por ejemplo, si se sustituye "Yugo Inn" la afirmación *d)* es

Yugo Inn tiene más de dos estrellas en la revista *Chicago*,

que es falsa. Si se sustituye "Le Français" en lugar de "restaurante", la afirmación *d)* es

Le Français tiene más de dos estrellas en la revista *Chicago*,

que es verdadera. Así, la afirmación *d)* es una función proposicional. ◀

Casi todas las afirmaciones en matemáticas y ciencias de la computación usan términos como "para todo" y "para alguno". Por ejemplo, en matemáticas se tiene el siguiente teorema:

Para todo triángulo T, la suma de los ángulos de T es igual a 180°.

En ciencias de la computación, se tiene el siguiente teorema:

Para algún programa P, la salida de P es P mismo.

Ahora se extenderá el sistema lógico de las secciones 1.1 y 1.2 de manera que las afirmaciones que incluyen "para todo" y "para alguno" sean manejables.

Definición 1.3.4 ▶

Sea P una función proposicional con dominio de discurso D. Se dice que la afirmación

$$\text{para toda } x,\ P(x)$$

es una *afirmación cuantificada universalmente*. El símbolo ∀ significa "para toda", "Para cada", "Para cualquier". Entonces, la afirmación

$$\text{para toda } x,\ P(x)$$

se escribe

$$\forall\, x\, P(x).$$

El símbolo ∀ se llama *cuantificador universal*.

La afirmación

$$\forall x\, P(x)$$

es verdadera si $P(x)$ es verdadera para toda x en D. La afirmación

$$\forall x\, P(x)$$

es falsa si $P(x)$ es falsa para al menos una x en D. ◀

Ejemplo 1.3.5 ▶

Considere la afirmación cuantificada universalmente

$$\forall x\ (x^2 \geq 0)$$

con el conjunto de números reales como dominio de discurso. La afirmación es verdadera porque, *para todo* número real x, es cierto que el cuadrado de x es positivo o cero. ◀

De acuerdo con la definición 1.3.4, la afirmación cuantificada universalmente

$$\forall x\, P(x)$$

es falsa si *para al menos una* x en el dominio de discurso, la proposición $P(x)$ es falsa. Un valor x en el dominio de discurso que hace que $P(x)$ sea falsa se llama **contraejemplo** de la afirmación

$$\forall x\, P(x).$$

Ejemplo 1.3.6 ▶

Considere la afirmación cuantificada universalmente

$$\forall x\ (x^2 - 1 > 0)$$

con el conjunto de los números reales como dominio de discurso. La afirmación es falsa, ya que si $x = 1$, la proposición

$$1^2 - 1 > 0$$

es falsa. El valor 1 es un contraejemplo de la afirmación

$$\forall x\ (x^2 - 1 > 0).$$

Aunque existen valores de x que hacen que la función proposicional sea verdadera, el contraejemplo muestra que la afirmación cuantificada universalmente es falsa. ◀

Ejemplo 1.3.7 ▶

Suponga que P es una función proposicional cuyo dominio de discurso consiste en los elementos $d_1, \ldots, d_n$. El siguiente seudocódigo† determina si

$$\forall x\, P(x)$$

es verdadera o falsa:

```
for i = 1 to n
    if (¬P(d_i))
        return falsa
return verdadera
```

El ciclo "for" examina los miembros d_i del dominio de discurso uno por uno. Si encuentra un valor d_i para el que $P(d_i)$ es falsa, la condición $\neg P(d_i)$ en el estatuto "if" es verdadera; así, el código regresa a falsa [para indicar que $\forall x\, P(x)$ es falsa] y termina. En este caso, d_i es un contraejemplo. Si $P(d_i)$ es verdadera para toda d_i, la condición $\neg P(d_i)$ en el estatuto "if" es siempre falsa. En este caso, el ciclo "for" corre hasta completarse, después de lo cual el código regresa a verdadera [para indicar que $\forall x\, P(x)$ es verdadera] y termina.

Observe que si $\forall x\, P(x)$ es verdadera, el ciclo "for" necesariamente corre hasta el final, de manera que *cada* miembro del dominio se verifica para asegurar que $P(x)$ es verdadera para toda x. Si $\forall x\, P(x)$ es falsa, el ciclo "for" termina en cuanto se encuentra *un* elemento x del dominio de discurso para el que $P(x)$ es falsa. ◀

La variable x en la función proposicional $P(x)$ se llama *variable libre*. (La idea es que x es "libre" de recorrer el dominio de discurso). La variable x en la afirmación cuantificada universalmente

$$\forall x\, P(x) \qquad \textbf{(1.3.1)}$$

se llama *variable acotada*. (La idea es que x está "acotada" por el cuantificador $\forall$).

Se señaló ya que una función proposicional no tiene valor de verdad. Por otro lado, la definición 1.3.4 asigna un valor de verdad a la afirmación cuantificada universalmente (1.3.1). En suma, una afirmación con variables libres (no cuantificadas) no es una proposición, y una afirmación sin variables libres (sin variables no cuantificadas) es una proposición.

Otras maneras de escribir

$$\forall x\, P(x)$$

son

$$\text{para toda } x,\ P(x)$$

y

$$\text{para cualquier } x,\ P(x).$$

El símbolo $\forall$ se lee "para toda", "para todos" o "para cualquier".

Para demostrar que

$$\forall x\, P(x)$$

es *verdadera* debemos, de hecho, examinar *todos* los valores de x en el dominio de discurso y demostrar que para toda x, $P(x)$ es cierta. Una técnica para probar que

$$\forall x\, P(x)$$

es verdadera consiste en hacer que x denote un elemento *arbitrario* del dominio de discurso D. El argumento procede usando el símbolo x. Cualquier cosa que se asegure acerca de

† El seudocódigo usado en este libro se explica en el apéndice C.

x debe ser cierto *sin importar qué valor* pueda tener x en D. El argumento debe concluir con la prueba de que $P(x)$ es verdadera.

Algunas veces, para especificar el dominio de discurso D, se escribe la afirmación cuantificada universalmente como

$$\text{para toda } x \text{ en } D,\ P(x).$$

Ejemplo 1.3.8 ▶

La afirmación cuantificada universalmente

$$\text{para todo número real } x, \text{ si } x > 1, \text{ entonces } x + 1 > 1$$

es verdadera. Esta vez se debe verificar que la afirmación

$$\text{si } x > 1, \text{ entonces } x + 1 > 1$$

es verdadera *para todo* número real x.

Sea x cualquier número real. Es cierto que para cualquier número real x, o bien $x \leq 1$ o $x > 1$. Si $x \leq 1$, la proposición condicional

$$\text{si } x > 1, \text{ entonces } x + 1 > 1$$

es trivialmente cierta. (La proposición es cierta porque la hipótesis $x > 1$ es falsa. Recuerde que cuando la hipótesis es falsa, la proposición condicional es verdadera sin importar si la conclusión es falsa o verdadera). En la mayoría de los argumentos, el caso trivial se omite.

Ahora suponga que $x > 1$. Sea cual fuere el valor específico de x, $x + 1 > x$. Como

$$x + 1 > x \quad \text{y} \quad x > 1,$$

se concluye que $x + 1 > 1$, de manera que la conclusión es verdadera. Si $x > 1$, la hipótesis y la conclusión son ambas verdaderas; así, la proposición condicional

$$\text{si } x > 1, \text{ entonces } x + 1 > 1$$

es verdadera.

Se ha demostrado que para todo número real x, la proposición

$$\text{si } x > 1, \text{ entonces } x + 1 > 1$$

es verdadera. Por lo tanto, la afirmación cuantificada universalmente

$$\text{para todo número real } x, \text{ si } x > 1, \text{ entonces } x + 1 > 1$$

es verdadera. ◀

El método para desaprobar la afirmación

$$\forall x\, P(x)$$

es bastante diferente del método usado para probar que la afirmación es verdadera. Para demostrar que la afirmación cuantificada universalmente

$$\forall x\, P(x)$$

es *falsa*, es suficiente encontrar *un* valor de x en el dominio de discurso para el que la proposición $P(x)$ sea falsa. Tal valor, como se recordará, se llama contraejemplo de la afirmación cuantificada universalmente.

Ahora se analizarán las afirmaciones cuantificadas existencialmente.

Definición 1.3.9 ▶

Sea P una función proposicional con dominio de discurso D. Se dice que la afirmación

$$\text{existe } x,\ P(x)$$

es una *afirmación cuantificada existencialmente*. El símbolo ∃ significa "existe". Así, la afirmación

$$\text{existe } x,\ P(x)$$

se escribe

$$\exists x\, P(x)$$

El símbolo ∃ se llama *cuantificador existencial*.

La afirmación

$$\exists x\, P(x)$$

es verdadera si $P(x)$ es verdadera para al menos una x en D. La afirmación

$$\exists x\, P(x)$$

es falsa si $P(x)$ es falsa para toda x en D. ◀

Ejemplo 1.3.10 ▶

Considere la afirmación cuantificada existencialmente

$$\exists x \left(\frac{x}{x^2+1} = \frac{2}{5} \right)$$

con el conjunto de números reales como dominio de discurso. La afirmación es verdadera porque es posible encontrar *al menos un* número real x para el que la proposición

$$\frac{x}{x^2+1} = \frac{2}{5}$$

es verdadera. Por ejemplo, si $x = 2$, se obtiene la proposición verdadera

$$\frac{2}{2^2+1} = \frac{2}{5}.$$

No ocurre que *todo* valor de x dé una proposición verdadera. Por ejemplo, si $x = 1$, la proposición

$$\frac{1}{1^2+1} = \frac{2}{5}$$

es falsa. ◀

Según la definición 1.3.9, la afirmación cuantificada existencialmente

$$\exists x\, P(x)$$

es falsa si para toda x en el dominio de discurso, la proposición $P(x)$ es falsa.

Ejemplo 1.3.11 ▶

Para verificar que la afirmación cuantificada existencialmente

$$\exists x \left(\frac{1}{x^2+1} > 1 \right)$$

es falsa, debe demostrarse que

$$\frac{1}{x^2+1} > 1$$

es falsa para todo número real x. Ahora

$$\frac{1}{x^2+1} > 1$$

es falsa precisamente cuando

$$\frac{1}{x^2+1} \leq 1$$

es cierta. Así, debe demostrarse que

$$\frac{1}{x^2+1} \leq 1$$

es verdadera para todo número real x. Con este fin, sea x cualquier número real. Como $0 \leq x^2$, se puede sumar 1 en ambos lados de la desigualdad para obtener $1 \leq x^2 + 1$. Si se dividen ambos lados de esta desigualdad por $x^2 + 1$, se obtiene

$$\frac{1}{x^2+1} \leq 1.$$

Por lo tanto, la afirmación

$$\frac{1}{x^2+1} \leq 1$$

es verdadera para todo número real x. Entonces la afirmación

$$\frac{1}{x^2+1} > 1$$

es falsa para todo número real x. Se ha demostrado que la afirmación cuantificada existencialmente

$$\exists x \left(\frac{1}{x^2+1} > 1 \right)$$

es falsa. ◀

Ejemplo 1.3.12 ▶

Suponga que P es una función proposicional cuyo dominio de discurso consiste en los elementos $d_1, \ldots, d_n$. El siguiente seudocódigo determina si

$$\exists x\, P(x)$$

es verdadera o falsa:

```
for i = 1 to n
    if (P(d_i))
        return verdadera
return falsa
```

El ciclo "for" examina los miembros d_i del dominio de discurso uno por uno. Si encuentra un valor d_i para el que $P(d_i)$ es verdadera, la condición $P(d_i)$ en el estatuto "if" es verdadera; así, el código regresa a verdadera [para indicar que $\exists x\, P(x)$ es verdadera] y termina. En este caso, el código encuentra un valor en el dominio de discurso, a saber d_i, para el que $P(d_i)$ es verdadera. Si $P(d_i)$ es falsa para toda d_i, la condición $P(d_i)$ en el estatuto "if" es siempre falsa. En este caso, el ciclo "for" corre hasta completarse, después de lo cual, regresa a falsa [para indicar que $\exists x\, P(x)$ es verdadera] y termina.

Observe que si $\exists x\, P(x)$ es verdadera, el ciclo "for" termina en cuanto se encuentra un elemento x del dominio de discurso para el que $P(x)$ es verdadera. Si $\exists x\, P(x)$ es falsa, el ciclo "for" corre hasta completarse, de manera que se verifique *todo* miembro del dominio de discurso para asegurarse de que $P(x)$ es falsa para toda x. ◀

Otras formas de escribir

$$\exists x\, P(x)$$

son

existe x tal que, $P(x)$

y

para alguna x, $P(x)$

y

para al menos una x, $P(x)$.

El símbolo $\exists$ se lee como "existe", "para alguna" o "para al menos una".

Ejemplo 1.3.13 ▶

Considere la afirmación cuantificada existencialmente

para alguna n, si n es primo, entonces $n + 1$, $n + 2$, $n + 3$ y $n + 4$ no son primos

con el conjunto de enteros positivos como dominio de discurso. Esta afirmación es verdadera porque podemos encontrar *al menos un* entero positivo n para el que la proposición condicional

si n es primo, entonces $n + 1$, $n + 2$, $n + 3$ y $n + 4$ no son primos

es verdadera. Por ejemplo, si $n = 23$, se obtienen las proposiciones

si 23 es primo, 24, 25, 26 y 27 no son primos.

(Esta proposición condicional es verdadera porque tanto la hipótesis "23 es primo" como la conclusión "24, 25, 26 y 27 no son primos" son verdaderas). Algunos valores de n hacen que la proposición condicional sea verdadera (por ejemplo, $n = 23$, $n = 4$, $n = 47$), mientras que para otros es falsa (como $n = 2$, $n = 101$). El hecho es que se encontró *un* valor que hace que la proposición condicional

si n es primo, entonces $n + 1$, $n + 2$, $n + 3$ y $n + 4$ no son primos

sea verdadera. Por esta razón, la afirmación cuantificada existencialmente

si n es primo, entonces $n + 1$, $n + 2$, $n + 3$ y $n + 4$ no son primos

es verdadera. ◀

En el ejemplo 1.3.11, se demostró que una afirmación cuantificada existencialmente era falsa probando que la afirmación cuantificada universalmente relacionada era verdadera. El siguiente teorema hace precisa esa relación. El teorema generaliza las leyes de De Morgan de lógica (ejemplo 1.2.11).

Teorema 1.3.14

Leyes generalizadas de De Morgan para lógica

Si P es una función proposicional, cada par de proposiciones en a) y b) tiene el mismo valor de verdad (es decir, ambas son verdaderas o ambas son falsas).

a) $\neg(\forall x\, P(x))$; $\exists x \neg P(x)$

b) $\neg(\exists x\, P(x))$; $\forall x \neg P(x)$

Demostración Se prueba sólo el inciso *a)* y se deja la demostración del inciso *b)* al lector (ejercicio 51).

Suponga que la proposición $\neg(\forall x\, P(x))$ es verdadera. Entonces la proposición $\forall x\, P(x)$ es falsa. Por la definición 1.3.4, la proposición $\forall x\, P(x)$ es falsa precisamente cuando $P(x)$ es falsa para al menos una x en el dominio de discurso. Pero si $P(x)$ es falsa para al menos una x en el dominio de discurso, $\neg P(x)$ es verdadera para al menos una x en el dominio de discurso. Por la definición 1.3.9, cuando $\neg P(x)$ es verdadera para al menos una

x en el dominio de discurso, la proposición $\exists x\ \neg P(x)$ es verdadera. Entonces, si la proposición $\neg(\forall x\ P(x))$ es verdadera, la proposición $\exists x\ \neg P(x)$ es verdadera. De manera similar, si la proposición $\neg(\forall x\ P(x))$ es falsa, la proposición $\exists x\ \neg P(x)$ es falsa.

Por lo tanto, el par de proposiciones del inciso *a)* siempre tienen los mismos valores de verdad.

Ejemplo 1.3.15 ▶

Sea $P(x)$ la afirmación

$$\frac{1}{x^2+1} > 1.$$

En el ejemplo 1.3.11 se demostró que

$$\exists x\ P(x)$$

es falsa verificando que

$$\forall x\ \neg P(x) \qquad \textbf{(1.3.2)}$$

es verdadera.

La técnica se justifica recurriendo al Teorema 1.3.14. Después de probar que la proposición (1.3.2) es verdadera, se puede negar (1.3.2) y concluir que

$$\neg(\forall x\ \neg\ P(x))$$

es falsa. Por el Teorema 1.3.14, inciso *a)*,

$$\exists x\ \neg\ \neg P(x)$$

o de manera equivalente,

$$\exists x\ P(x)$$

también es falsa. ◀

Ejemplo 1.3.16 ▶

Escriba la afirmación

Todo amante del rock ama a U2,

de manera simbólica. Escriba su negación en símbolos y en palabras.

Sea $P(x)$ la función proposicional "x ama U2". La afirmación se escribe simbólicamente como

$$\forall x\ P(x).$$

El dominio de discurso es el conjunto de amantes del rock.

Por el Teorema 1.3.14, inciso *a)*, la negación de la proposición anterior $\neg(\forall x\ P(x))$ es equivalente a

$$\exists x\ \neg P(x).$$

En palabras, esta última proposición se enuncia como: Existe un amante del rock que no ama a U2. ◀

Ejemplo 1.3.17 ▶

Escriba la afirmación

Algunas aves no pueden volar,

simbólicamente. Escriba la negación en símbolos y en palabras.

Sea $P(x)$ la función proposicional "x vuela". La afirmación se escribe en símbolos como

$$\exists x\ \neg P(x)$$

[La afirmación también pudo escribirse $\exists x\, Q(x)$, donde $Q(x)$ es la función proposicional "x no puede volar". Al igual que en álgebra, existen muchas maneras de representar simbólicamente el texto.] El dominio de discurso es el conjunto de aves.

Por el Teorema 1.3.14, inciso *b)*, la negación de la proposición anterior $\neg(\exists x\, \neg P(x))$ es equivalente a

$$\forall x\, \neg\, \neg P(x)$$

o lo que es lo mismo,

$$\forall x\, P(x).$$

En palabras, esta última proposición se enuncia como: Toda ave puede volar. ◀

Una proposición cuantificada universalmente generaliza la proposición

$$P_1 \wedge P_2 \wedge \ldots \wedge P_n \tag{1.3.3}$$

en el sentido de que las proposiciones individuales $P_1, P_2, \ldots, P_n$ se sustituyen por una familia arbitraria $P(x)$, donde x es un miembro del dominio de discurso, y (1.3.3) se sustituye por

$$\forall x\, P(x). \tag{1.3.4}$$

La proposición (1.3.3) es verdadera si y sólo si P_i es verdadera para toda $i = 1, \ldots, n$. El valor de verdad de la proposición (1.3.4) se define de manera similar: (1.3.4) es verdadera si y sólo si $P(x)$ es verdadera para toda x en el dominio de discurso.

De manera similar, una proposición cuantificada existencialmente generaliza la proposición

$$P_1 \vee\ P_2 \vee\ \ldots \vee P_n \tag{1.3.5}$$

en el sentido de que las proposiciones individuales $P_1, P_2, \ldots, P_n$ se sustituyen por una familia arbitraria $P(x)$, donde x es un miembro del dominio de discurso, y (1.3.5) se sustituye por

$$\exists x\, P(x).$$

Las observaciones anteriores explican cómo el Teorema 1.3.14 generaliza las leyes de De Morgan para lógica (ejemplo 1.2.11). Recuerde que la primera ley de De Morgan para lógica establece que las proposiciones

$$\neg(P_1 \vee P_2 \vee \ldots \vee P_n) \quad \text{y} \quad \neg P_1 \wedge\ \neg P_2 \wedge \ldots \wedge \neg P_n$$

tienen los mismos valores de verdad. En el teorema 1.3.14, inciso *b)*,

$$\neg P_1 \wedge\ \neg P_2 \wedge \ldots \wedge \neg P_n$$

se sustituye por

$$\forall x\, \neg P(x)$$

y

$$\neg(P_1 \vee\ P_2 \vee \ldots \vee P_n)$$

se sustituye por

$$\neg(\exists x\, P(x)).$$

Ejemplo 1.3.18 ▶ Las afirmaciones en palabras con frecuencia tienen más de una interpretación posible. Considere la bien conocida cita de *El mercader de Venecia* de Shakespeare:

No todo lo que brilla es oro.

Una interpretación posible de esta cita es: Todo objeto que brilla no es oro. Sin embargo, seguro que esto no es lo que Shakespeare quiso decir. La interpretación correcta es: Algún objeto que brilla no es oro.

Si $P(x)$ es la función proposicional "x brilla" y $Q(x)$ es la función proposicional "x es oro", la primera interpretación se convierte en

$$\forall x(P(x) \rightarrow \neg Q(x)), \tag{1.3.6}$$

y la segunda interpretación se convierte en

$$\exists x(P(x) \wedge \neg Q(x)).$$

Usando el resultado del ejemplo 1.2.13, se ve que los valores de verdad de

$$\exists x(P(x) \wedge \neg Q(x))$$

y

$$\exists x \neg (P(x) \rightarrow Q(x))$$

son los mismos. Por el Teorema 1.3.14, los valores de verdad de

$$\exists x \neg (P(x) \rightarrow Q(x))$$

y

$$\neg(\forall x\, P(x) \rightarrow\ Q(x))$$

son los mismos. Así, una manera equivalente de representar la segunda interpretación es

$$\neg(\forall x\, P(x) \rightarrow\ Q(x)). \tag{1.3.7}$$

Al comparar (1.3.6) y (1.3.7), vemos la ambigüedad que resulta para establecer si la negación se aplica a $Q(x)$ (la primera interpretación) o a toda la afirmación

$$\forall x\ (P(x) \rightarrow Q(x))$$

(la segunda interpretación). La interpretación correcta de la afirmación

Todo lo que brilla no es oro

se obtiene al negar toda la afirmación.

En afirmaciones positivas, "cualquiera", "cada" y "todo" tienen el mismo significado. En afirmaciones negativas, la situación cambia:

No todas las x satisfacen $P(x)$.
No cada x satisface $P(x)$.
No cualquier x satisface $P(x)$.

se considera que tienen el mismo significado que

Para alguna x, $\neg P(x)$;

mientras que

Ni una x satisface $P(x)$

Ninguna x satisface $P(x)$

significan

Para toda x, $\neg P(x)$.

Vea otros ejemplos en los ejercicios 41 al 49. ◀

Sugerencias para resolver problemas

Para probar que la afirmación cuantificada universalmente

$$\forall x\, P(x)$$

es verdadera, demuestre que para *toda x* en el dominio de discurso, la proposición $P(x)$ es verdadera. Demostrar que $P(x)$ es verdadera para un valor *particular* de *x no* prueba que

$$\forall x\, P(x)$$

sea cierta.

Para probar que la afirmación cuantificada existencialmente

$$\exists x\, P(x)$$

es verdadera, encuentre *un* valor de x en el dominio de discurso para el que la proposición $P(x)$ es verdadera. *Un* valor es suficiente.

Para probar que la afirmación cuantificada universalmente

$$\forall x\, P(x)$$

es falsa, encuentre *un* valor de x (un contraejemplo) en el dominio de discurso para el que la proposición $P(x)$ es falsa.

Para probar que la afirmación cuantificada existencialmente

$$\exists x\, P(x)$$

es falsa, demuestre que para *toda x* en el dominio de discurso, la proposición $P(x)$ es falsa. Demostrar que $P(x)$ es falsa para un valor *particular x no* prueba que

$$\exists x\, P(x)$$

sea falsa.

Sección de ejercicios de repaso

1. ¿Qué es una función proposicional?
2. ¿Qué es un dominio de discurso?
3. ¿Qué es una afirmación cuantificada universalmente?
4. ¿Qué es un contraejemplo?
5. ¿Qué es una afirmación cuantificada existencialmente?
6. Establezca las leyes generalizadas de De Morgan para lógica.
7. Explique cómo probar que una afirmación cuantificada universalmente es verdadera.
8. Explique cómo probar que una afirmación cuantificada existencialmente es verdadera.
9. Explique cómo probar que una afirmación cuantificada universalmente es falsa.
10. Explique cómo probar que una afirmación cuantificada existencialmente es falsa.

Ejercicios

En los ejercicios 1 al 6, diga si la afirmación es una función proposicional. Para cada afirmación que sea una función proposicional, dé un dominio de discurso.

1. $(2n + 1)^2$ es un entero impar.
2. Seleccione un entero entre 1 y 10.
3. Sea x un número real.
4. La película ganó el premio de la Academia como mejor película en 1955.
5. $1 + 3 = 4$.
6. Existe x tal que $x < y$ (x, y números reales).

Sea P(n) la función proposicional "n divide a 77". Escriba cada proposición en los ejercicios 7 al 11 en palabras y diga si es verdadera o falsa. El dominio de discurso es el conjunto de enteros positivos.

7. $P(11)$
8. $P(1)$
9. $P(3)$
10. $\forall n\, P(n)$
11. $\exists n\, P(n)$

Sea P(x) la afirmación "x está en un curso de matemáticas". El dominio de discurso es el conjunto de todos los estudiantes. Escriba cada proposición en los ejercicios 12 al 17 en palabras.

12. $\forall x\, P(x)$
13. $\exists x\, P(x)$

14. $\forall x \neg P(x)$ **15.** $\exists x \neg P(x)$

16. $\neg(\forall x\ P(x))$ **17.** $\neg(\exists x\ P(x))$

18. Escriba la negación de cada proposición en los ejercicios 12 al 17 en símbolos y en palabras.

Sea P(x) la afirmación "x es un atleta profesional" y sea Q(x) la afirmación "x juega fútbol". El dominio de discurso es el conjunto de todas las personas. Escriba cada proposición en los ejercicios 19 al 26 en palabras. Determine el valor de verdad de cada afirmación.

19. $\forall x\,(P(x) \rightarrow Q(x))$ **20** $\exists x\,(P(x) \rightarrow Q(x))$

21. $\forall x\,(Q(x) \rightarrow P(x))$ ***22.*** $\exists x\,(Q(x) \rightarrow P(x))$

23. $\forall x\,(P(x) \vee Q(x))$ **24.** $\exists x\,(P(x) \vee Q(x))$

25. $\forall x\,(P(x) \wedge Q(x))$ **26.** $\exists x\,(P(x) \wedge Q(x))$

27. Escriba la negación de cada proposición en los ejercicios 19 al 26 en símbolos y en palabras.

Sea P(x)la afirmación "x es un contador" y sea Q(x) la afirmación "x tiene un Porsche". Escriba en símbolos y en palabras cada afirmación en los ejercicios 28 al 31.

28. Todos los contadores tienen un Porsche.

29. Algunos contadores tienen un Porsche.

30. Todos los dueños de Porsches son contadores.

31. Alguien que tiene un Porsche es contador.

32. Escriba la negación de cada proposición en los ejercicios 28 al 31 en símbolos y en palabras.

Determine el valor de verdad de cada afirmación en los ejercicios 33 al 38. El dominio de discurso es el conjunto de números reales. Justifique sus respuestas.

33. $\forall x(x^2 > x)$

34. $\exists x(x^2 > x)$

35. $\forall x(x > 1 \rightarrow x^2 > x)$

36. $\exists x(x > 1 \rightarrow x^2 > x)$

37. $\forall x(x > 1 \rightarrow x/(x^2 + 1) < 1/3)$

38. $\exists x(x > 1 \rightarrow x/(x^2 + 1) < 1/3)$

39. Escriba la negación de cada proposición en los ejercicios 33 al 38 en símbolos y en palabras.

40. ¿El seudocódigo del ejemplo 1.3.7 podría escribirse como sigue?

```
for i = 1 to n
    if(¬P(d_i))
        return falsa
else
        return verdadera
```

¿Cuál es el significado literal de cada afirmación en los ejercicios 41 al 49? ¿Cuál es el significado deseado? Aclare cada afirmación expresándola con otras palabras y en símbolos.

41. De *Querida Abby*: Todos los hombres no engañan a sus esposas.

42. De *San Antonio Express News:* Todas la cosas viejas no envidian a las cosas de veinte.

43. Todos los 74 hospitales no entregan su reporte cada mes.

44. El economista Robert J. Samuelson: Todo problema ambiental no es una tragedia.

45. Comentario del consejal de Door County: Esto todavía es Door County y todos nosotros no tenemos un título.

46. Título de una columna de Martha Stewart: Todas las pantallas de las lámparas no se pueden limpiar.

47. Titular en el *New York Times:* Un mundo donde no todo es dulzura y luz.

48. Encabezado de una historia de subsidio a la vivienda: Todos no pueden pagar una casa.

49. De *Newsweek:* Las investigaciones formales son una buena práctica en las circunstancias correctas, pero toda circunstancia no es correcta.

50. *a)* Use una tabla de verdad para probar que si p y q son proposiciones, al menos una de $p \rightarrow q$ o $q \rightarrow p$ es cierta.

b) Sea $P(x)$ la función proposicional "x es un entero" y sea $Q(x)$ la función proposicional "x es un número positivo". El dominio de discurso es el conjunto de todos los números reales. Determine si la siguiente prueba de que todos los enteros son positivos o todos los números reales positivos son enteros es correcta o no.

Por el inciso *a)*,

$$\forall x\,((P(x) \rightarrow Q(x)) \vee (Q(x) \rightarrow P(x)))$$

es verdadera. En palabras: Para toda x, si x es un entero, entonces x es positivo; o si x es positivo, entonces x es un entero. Por lo tanto, todos los enteros son positivos o todos los números reales positivos son enteros.

51. Demuestre el Teorema 1.3.14, inciso *b)*.

1.4 ➜ Cuantificadores anidados

Considere escribir la afirmación

La suma de cualesquiera dos números reales positivos es positiva,

simbólicamente. Primero se observa que se trata de dos números, se necesitan dos variables, digamos x y y. La aseveración se puede reestablecer como: Si $x > 0$ y $y > 0$, entonces $x + y > 0$. La afirmación dice que la suma de *cualesquiera* dos números reales positivos es positiva, de manera que se necesitan dos cuantificadores universales. Así, la afirmación se escribe simbólicamente como

$$\forall x \forall y((x > 0) \wedge (y > 0) \rightarrow (x + y > 0)).$$

En palabras, para cada x y para cada y, si $x > 0$ y $y > 0$, entonces $x + y > 0$. El dominio de discurso es el conjunto de números reales. Se dice que los cuantificadores múltiples como $\forall\, x\ \forall y$ son **cuantificadores anidados.** En esta sección se exploran con detalle los cuantificadores anidados.

Ejemplo 1.4.1 ▶

Replantee

$$\forall m \exists n\ (m < n)$$

en palabras. El dominio de discurso es el conjunto de enteros.

Primero se replantea esta afirmación como: Para toda m, existe n tal que $m < n$. De manera menos formal, esto significa que si toma cualquier entero m, hay un entero n mayor que m. Dicho de otra forma: No existe un entero que sea el más grande. ◀

Ejemplo 1.4.2 ▶

Escriba la aseveración

Todos aman a alguien,

en símbolos, donde $L(x, y)$ es la afirmación "x ama a y".

"Todos" requiere un cuantificador universal y "alguien" requiere un cuantificador existencial. Entonces, la afirmación se escribe en símbolos como

$$\forall x \exists y\ L(x,y).$$

En palabras, para cada persona x, existe una persona y, tal que x ama a y.

Observe que

$$\exists x \forall y\ L(x,y)$$

no es una interpretación correcta de la afirmación original. Esta última afirmación es: Existe una persona x tal que para toda y, x ama a y. De modo menos formal, alguien ama a todos. El orden de los cuantificadores es importante; cambiar el orden altera el significado. ◀

Por definición, la afirmación

$$\forall x \forall y\ P(x,y),$$

con dominio de discurso D, es verdadera si, para *toda* x y para *toda* y en D, $P(x, y)$ es verdadera. La afirmación

$$\forall x \forall y\ P(x,y)$$

es falsa si existe *al menos una* x y *al menos una* y en D tal que $P(x, y)$ es falsa.

Ejemplo 1.4.3 ▶

Considere la afirmación

$$\forall x \forall y((x > 0) \wedge (y > 0) \rightarrow (x + y > 0)),$$

con el conjunto de números reales como dominio de discurso. Esta afirmación es verdadera porque, para todo número real x y para todo número real y, la proposición condicional

$$(x > 0) \wedge (y > 0) \rightarrow (x + y > 0)$$

es verdadera. En palabras, para todo número real x y para todo número real y, si x y y son positivos, su suma es positiva. ◀

Ejemplo 1.4.4 ▶

Considere la afirmación

$$\forall x \forall y((x > 0)\ y < 0) \rightarrow (x + y \neq 0)),$$

con el conjunto de números reales como dominio de discurso. Esta afirmación es falsa porque si $x = 1$ y $y = -1$, la proposición condicional

$$(x > 0) \wedge (y < 0) \rightarrow (\mathrm{x} + \mathrm{y} \neq 0)$$

es falsa. Se dice que el par $x = 1$ y $y = -1$ es un contraejemplo. ◀

Ejemplo 1.4.5 ▶

Suponga que P es una función proposicional cuyo dominio de discurso consiste en los elementos d_1, ..., d_n. El siguiente seudocódigo determina si

$$\forall x \forall y\ P(x,y)$$

es verdadera o falsa:

```
for i = 1 to n
    for j = 1 to n
        if(¬P(d_i, d_j)
            return falsa
return verdadera
```

Los ciclos "for" examinan pares de miembros del dominio de discurso. Si encuentran un par d_i, d_j para el cual $P(d_i, d_j)$ es falsa, la condición $\neg P(d_i, d_j)$ en el estatuto "if" es verdadera; entonces, el código envía a falsa [para indicar que $\forall x \forall y\ P(x, y)$ es falsa] y termina. En este caso, el par d_i, d_j es un contraejemplo. Si $P(d_i, d_j)$ es verdadera para todo par d_i, d_j, la condición $\neg P(d_i, d_j)$ en el estatuto "if" es siempre falsa. En este caso, los ciclos "for" corren hasta terminar, después de lo cual, el código regresa a verdadera [para indicar que $\forall x \forall y\ P(x, y)$ es verdadera] y termina. ◀

Por definición, la afirmación

$$\forall x\ \exists y\ P(x, y),$$

con dominio de discurso D, es verdadera si, para *toda* x en D, existe *al menos una* y en D para la que $P(x, y)$ es verdadera. La afirmación

$$\forall x\ \exists y\ P(x, y)$$

es falsa si existe *al menos una* x en D tal que $P(x, y)$ es falsa para toda y en D.

Ejemplo 1.4.6 ▶

Considere la afirmación

$$\forall x \exists y (x + y = 0),$$

cuyo dominio de discurso es el conjunto de números reales. Esta afirmación es verdadera porque, para todo número real x, existe al menos una y (a saber $y = -x$) para la que $x + y = 0$ es verdadera. En palabras, para todo número real x, existe un número que, al sumarse a x hace la suma igual a cero. ◀

Ejemplo 1.4.7 ▶

Considere la afirmación

$$\forall x \exists y (x > y),$$

cuyo dominio de discurso es el conjunto de enteros positivos. Esta afirmación es falsa porque existe al menos una x, a saber $x = 1$, tal que $x > y$ es falsa para todo entero positivo y. ◀

Ejemplo 1.4.8 ▶

Suponga que P es una función proposicional cuyo dominio de discurso consiste en los elementos d_1, . . . , d_n. El siguiente seudocódigo determina si

$$\forall x \exists y\ P(x,y)$$

es verdadera o falsa:

```
for i = 1 to n
    if(¬existe_dj(i))
        return falsa
return verdadera
```

```
existe_dj(i)){
   for j = 1 to n
   if (P(d_i, d_j))
       return verdadera
   return falsa
}
```

Si para cada d_i, existe una d_j tal que $P(d_i, d_j)$ es verdadera, entonces para cada i, $P(d_i, d_j)$ es verdadera para alguna j. Entonces, *existe_dj*(i) regresa a verdadera para toda i. Como $\neg$*existe_dj*(i) siempre es falsa, el primer ciclo "for" termina eventualmente y regresa a verdadera para indicar que $\forall x \exists y\, P(x)$ es verdadera.

Si para alguna d_i, $P(d_i, d_j)$ es falsa para toda j, entonces, para esta i, $P(d_i, d_j)$ es falsa para toda j. En este caso, el ciclo "for" en *existe_dj*(i) corre hasta el final y regresa a falsa. Como $\neg$*existe_dj(i)* es verdadera, regresa a falsa para indicar que $\forall x \exists y\, P(x)$ es falsa. ◀

Por definición, la afirmación

$$\exists x \forall y\, P(x, y),$$

con dominio de discurso D, es verdadera si existe *al menos una* x en D tal que $P(x, y)$ es verdadera *para toda* y en D. La afirmación

$$\exists x \forall y\, P(x, y)$$

es falsa si, para *toda* x en D, existe *al menos una* y en D tal que $P(x, y)$ es falsa.

Ejemplo 1.4.9 ▶ Considere la afirmación

$$\exists x \forall y\, (x \leq y),$$

cuyo dominio de discurso es el conjunto de enteros positivos. Esta afirmación es verdadera porque hay al menos un entero positivo x (a saber $x = 1$) para el que $x \leq y$ es verdadera para todo entero positivo y. En otras palabras, existe un entero positivo más pequeño (el 1). ◀

Ejemplo 1.4.10 ▶ Considere la afirmación

$$\exists x \forall y\, (x \geq y),$$

con el conjunto de enteros positivos como dominio de discurso. Esta afirmación es falsa porque, para todo entero positivo x, existe al menos un entero positivo y, a saber $y = x + 1$, tal que $x \geq y$ es falsa. En otras palabras, no hay el entero positivo más grande. ◀

Por definición, la afirmación

$$\exists x \exists y\, P(x, y),$$

con dominio de discurso D, es verdadera si existe *al menos una* x en D y *al menos una* y en D tal que $P(x, y)$ es verdadera. La afirmación

$$\exists x \exists y\, P(x, y)$$

es falsa si, para *toda* x en D y para *toda* y en D, $P(x, y)$ es falsa.

Ejemplo 1.4.11 ▶ Considere la afirmación

$$\exists x \exists y\, ((x > 1) \wedge (y > 1) \wedge (xy = 6)),$$

con el conjunto de enteros positivos como dominio de discurso. Esta afirmación es verdadera porque existe al menos un entero positivo x (a saber $x = 2$) y al menos un entero positivo y (a saber $y = 3$) tales que $xy = 6$. En otras palabras, 6 es compuesto (es decir, no es primo). ◀

Ejemplo 1.4.12 ▶

Considere la afirmación

$$\exists x \exists y((x > 1) \wedge (y > 1) \wedge (xy = 7)),$$

cuyo dominio de discurso es el conjunto de enteros positivos. Esta afirmación es falsa porque para todo entero positivo x y para todo entero positivo y,

$$(x > 1) \wedge (y > 1) \wedge (xy = 7),$$

es falsa. En palabras, 7 es primo. ◀

Las leyes generalizadas de De Morgan para lógica (Teorema 1.3.14) se pueden usar para negar una proposición que contiene cuantificadores anidados.

Ejemplo 1.4.13 ▶

Usando las leyes de De Morgan para lógica, se encuentra que la negación de

$$\forall x \exists y\ P(x, y)$$

es

$$\neg(\forall x \exists y\ P(x, y)) \equiv \exists x \neg(\exists y\ P(x, y)) \equiv \exists x \forall y\ \neg P(x, y)$$

Observe que en la negación, $\forall$ y $\exists$ están intercambiados. ◀

Ejemplo 1.4.14 ▶

Escriba la negación de $\exists x \forall y(xy < 1)$, donde el dominio de discurso es el conjunto de números reales. Determine el valor de verdad de la afirmación y su negación.

Usando las leyes generalizadas de De Morgan para lógica, se encuentra que la negación es

$$\neg(\exists x \forall y(xy < 1)) \equiv \forall x \neg(\forall y(xy < 1)) \equiv \forall x \exists y \neg(xy < 1) \equiv \forall x \exists y(xy \geq 1).$$

La afirmación dada $\exists x \forall y(xy < 1)$ es verdadera porque existe al menos una x (a saber $x = 0$) tal que $xy < 1$ para toda y. Como la afirmación dada es verdadera, su negación es falsa. ◀

Se concluye con un juego de lógica que presenta una manera alternativa para determinar si una función proposicional cuantificada es verdadera o falsa. André Berthiaume contribuyó con este ejemplo.

Ejemplo 1.4.15 ▶

El juego de lógica

A partir de una función proposicional como

$$\forall x \exists y\ P(x, y),$$

usted y su oponente, a quien llamaremos Fernando, participan en un juego de lógica. Su meta es hacer $P(x, y)$ verdadera, y la de Fernando es tratar de que $P(x, y)$ sea falsa. El juego comienza con el primer cuantificador (izquierda). Si el cuantificador es $\forall$, Fernando elige un valor para esa variable; si el cuantificador es $\exists$, usted elige un valor para esa variable. El juego continúa con el segundo cuantificador. Después de elegir los valores para todas la variables, si $P(x, y)$ es verdadera, usted gana; si $P(x, y)$ es falsa, Fernando gana. Se mostrará que si usted gana siempre sin importar qué valores elija Fernando para las variables, la afirmación cuantificada será verdadera, pero si Fernando elige valores para las variables de manera que usted no pueda ganar, la afirmación cuantificada será falsa.

Considere la afirmación

$$\forall x \exists y(x + y = 0).$$

El dominio de discurso es el conjunto de números reales. Como el primer cuantificador es $\forall$, Fernando juega primero y elige un valor para x. Como el segundo cuantificador es $\exists$, usted juega. Sin importar qué valor elija Fernando, usted selecciona $y = -x$, que convierte la

afirmación $x + y = 0$ en verdadera. Usted siempre puede ganar este juego, de manera que la afirmación

$$\forall x \exists y (x + y = 0)$$

es verdadera.

Ahora considere la afirmación

$$\exists x \forall y (x + y = 0).$$

De nuevo, el dominio de discurso es el conjunto de números reales. Como el primer cuantificador es $\exists$, usted juega primero y elige un valor para x. Como el segundo cuantificador es $\forall$, Fernando juega después. Sin importar qué valor eligió usted, Fernando siempre puede seleccionar un valor de y que haga falsa la afirmación $x + y = 0$. (Si usted elige $x = 0$, Fernando opta por $y = 1$. Si usted elige $x \neq 0$, Fernando escoge $y = 0$). Fernando puede ganar siempre el juego, por lo tanto la afirmación

$$\exists x \forall y (x + y = 0)$$

es falsa.

Se analizará por qué el juego determina correctamente el valor de verdad de una función proposicional cuantificada. Considere

$$\forall x \forall y P(x, y).$$

Si Fernando gana siempre, significa que encuentra valores de x y y que hacen que $P(x, y)$ sea falsa. En este caso, la función proposicional es falsa; los valores que Fernando encuentra constituyen un contraejemplo. Si Fernando no puede ganar el juego, no existe un contraejemplo; en este caso, la función proposicional es verdadera.

Considere

$$\forall x \exists y P(x, y)$$

Fernando juega primero y elige valores para x. Usted le sigue en turno. No importa qué valor haya elegido Fernando, si usted elige valores para y que hagan a $P(x, y)$ verdadera, ganará siempre y la función proposicional será verdadera. Sin embargo, si Fernando elige un valor de x de manera que todo valor y que usted seleccione haga que $P(x, y)$ sea falsa, entonces usted siempre perderá el juego y la función proposicional será falsa.

Un análisis de los otros casos muestra que si usted gana el juego siempre, la función proposicional es verdadera; pero si Fernando gana siempre, será falsa.

El juego de lógica se extiende a funciones proposicionales de más de dos variables. Las reglas son las mismas y, de nuevo, si usted puede ganar siempre, la función proposicional es verdadera; si Fernando puede ganar siempre, la función proposicional es falsa. ◀

Sugerencias para resolver problemas

Para probar que

$$\forall x \forall y P(x, y)$$

es verdadera, debe demostrar que $P(x, y)$ es verdadera para todos los valores de x y y en el dominio de discurso. Una técnica es argumentar que $P(x, y)$ es verdadera usando los símbolos x y y para representar elementos *arbitrarios* en el dominio de discurso.

Para probar que

$$\forall x \exists y P(x, y)$$

es verdadera, debe demostrar que para toda x en el dominio de discurso, existe al menos una y en el dominio de discurso tal que $P(x, y)$ es verdadera. Una técnica consiste en que x represente un elemento arbitrario del dominio de discurso y después encontrar un valor para y (¡*un* valor es suficiente!) para el que $P(x, y)$ es verdadera.

Para probar que

$$\exists x \forall y\, P(x, y)$$

es verdadera, debe demostrar que para al menos una x en el dominio de discurso, $P(x, y)$ es verdadera para toda y en el dominio de discurso. Una técnica consiste en encontrar un valor de x (de nuevo, ¡*un* valor es suficiente!) que tenga la propiedad de que $P(x, y)$ sea verdadera para toda y en el dominio de discurso. Una vez que se ha escogido un valor para x, deje que y represente un elemento arbitrario del dominio de discurso y demuestre que $P(x, y)$ es siempre verdadera.

Para probar que

$$\exists x \exists y\, P(x, y)$$

es verdadera, encuentre un valor de x y un valor de y (*dos* valores son suficientes, uno para x y uno para y) para los que $P(x, y)$ es verdadera.

Para negar una expresión con cuantificadores anidados, se usan las leyes generalizadas de De Morgan para lógica. En palabras comunes, $\forall$ y $\exists$ se intercambian. No olvide que la negación de $p \rightarrow q$ es equivalente a $p \wedge \neg q$.

Sección de ejercicios de repaso

1. ¿Cuál es la interpretación de $\forall x \forall y\, P(x, y)$? ¿Cuándo es verdadera esta expresión cuantificada? ¿Cuándo es falsa?
2. ¿Cuál es la interpretación de $\forall x \exists y\, P(x, y)$? ¿Cuándo es verdadera esta expresión cuantificada? ¿Cuándo es falsa?
3. ¿Cuál es la interpretación de $\exists x \forall y\, P(x, y)$? ¿Cuándo es verdadera esta expresión cuantificada? ¿Cuándo es falsa?
4. ¿Cuál es la interpretación de $\exists x \exists y\, P(x, y)$? ¿Cuándo es verdadera esta expresión cuantificada? ¿Cuándo es falsa?
5. Dé un ejemplo para mostrar que, en general, $\forall x \exists y\, P(x, y)$ y $\exists x \forall y\, P(x, y)$ tienen significados diferentes.
6. Escriba la negación de $\forall x \forall y\, P(x, y)$ usando las leyes generalizadas de De Morgan para lógica.
7. Escriba la negación de $\forall x \exists y\, P(x, y)$ usando las leyes generalizadas de De Morgan para lógica.
8. Escriba la negación de $\exists x \forall y\, P(x, y)$ usando las leyes generalizadas de De Morgan para lógica.
9. Escriba la negación de $\exists x \exists y\, P(x, y)$ usando las leyes generalizadas de De Morgan para lógica.
10. Explique las reglas para jugar el juego de lógica. ¿Cómo puede usarse este juego para determinar el valor de verdad de una expresión cuantificada?

Ejercicios

Sea $T(x, y)$ la función proposicional "x es más alto que y, y $x \neq y$". El dominio de discurso consiste en tres estudiantes: Gerardo, que mide 5 pies 11 pulgadas; Ernesto, que mide 5 pies 6 pulgadas, y Martín que mide 6 pies. Escriba cada proposición en los ejercicios 1 al 4 en palabras y diga si ésta es verdadera o falsa.

1. $\forall x \forall y\, T(x, y)$
2. $\forall x \exists y\, T(x, y)$
3. $\exists x \forall y\, T(x, y)$
4. $\exists x \exists y\, T(x, y)$
5. Escriba la negación de cada proposición en los ejercicios del 1 al 4 en palabras y con símbolos.

Sea $T(x, y)$ la función proposicional "x es más alto o de la misma altura que y". El dominio de discurso consiste en tres estudiantes: Gerardo, que mide 5 pies 11 pulgadas; Ernesto, que mide 5 pies 6 pulgadas, y Martín que mide 6 pies. Escriba cada proposición en los ejercicios 6 al 9 en palabras y diga si ésta es verdadera o falsa.

6. $\forall x \forall y\, T(x, y)$
7. $\forall x \exists y\, T(x, y)$
8. $\exists x \forall y\, T(x, y)$
9. $\exists x \exists y\, T(x, y)$
10. Escriba la negación de cada proposición en los ejercicios 6 al 9 en palabras y con símbolos.

Sea $L(x, y)$ la función proposicional "x ama a y". El dominio de discurso es el conjunto de todas las personas amantes. Escriba cada proposición en los ejercicios 11 al 14 con símbolos. ¿Cuál piensa que sea verdadera?

11. Alguien ama a todos.
12. Todos aman a todos.
13. Alguien ama a alguien.
14. Todos aman a alguien.
15. Escriba la negación de cada proposición en los ejercicios 11 al 14 en palabras y con símbolos.

Sea $P(x, y)$ la función proposicional $x \geq y$. El dominio de discurso es el conjunto de todos los enteros positivos. Diga si cada proposición en los ejercicios 16 al 19 es verdadera o falsa.

16. $\forall x \forall y\, P(x, y)$
17. $\forall x \exists y\, P(x, y)$
18. $\exists x \forall y\, P(x, y)$
19. $\exists x \exists y\, P(x, y)$
20. Escriba la negación de cada proposición en los ejercicios 16 al 19.

Determine el valor de verdad de cada afirmación en los ejercicios 21 al 38. El dominio de discurso es el conjunto de números reales. Justifique sus respuestas.

21. $\forall x \forall y(x^2 < y + 1)$ ***22.*** $\forall x \exists y(x^2 < y + 1)$

23. $\exists x \forall y(x^2 < y + 1)$ ***24.*** $\exists x \exists y(x^2 < y + 1)$

25. $\exists y \forall x(x^2 < y + 1)$ **26.** $\forall y \exists x(x^2 < y + 1)$

27. $\forall x \forall y(x^2 + y^2 = 9)$ **28.** $\forall x \exists y(x^2 + y^2 = 9)$

29. $\exists x \forall y(x^2 + y^2 = 9)$ ***30.*** $\exists x \exists y(x^2 + y^2 = 9)$

31. $\forall x \forall y(x^2 + y^2 \geq 0)$ **32.** $\forall x \exists y(x^2 + y^2 \geq 0)$

33. $\exists x \forall y(x^2 + y^2 \geq 0)$ **34.** $\exists x \exists y(x^2 + y^2 \geq 0)$

35. $\forall x \forall y((x < y) \rightarrow (x^2 < y^2))$

36. $\forall x \exists y((x < y) \rightarrow (x^2 < y^2))$

37. $\exists x \forall y((x < y) \rightarrow (x^2 < y^2))$

38. $\exists x \exists y((x < y) \rightarrow (x^2 < y^2))$

39. Escriba la negación de cada proposición en los ejercicios 21 al 38.

40. Suponga que P es una función proposicional cuyo dominio de discurso consiste en los elementos d_1, ..., d_n. Escriba el seudocódigo que determina si

$$\exists x \forall y\, P(x, y)$$

es verdadera o falsa.

41. Suponga que P es una función proposicional cuyo dominio de discurso consiste en los elementos d_1, ..., d_n. Escriba el seudocódigo que determina si

$$\exists x \exists y\, P(x, y)$$

es verdadera o falsa.

42. Explique cómo determina el juego de lógica (ejemplo 1.4.15) si cada proposición en los ejercicios 21 al 38 es verdadera o falsa.

43. Use el juego de lógica (ejemplo 1.4.15) para determinar si la proposición

$$\forall x \forall y \exists z((z > x) \wedge (z < y))$$

es verdadera o falsa. El dominio de discurso es el conjunto de todos los enteros.

44. Use el juego de lógica (ejemplo 1.4.15) para determinar si la proposición

$$\forall x \forall y \exists z((z < \mathrm{x}) \wedge (z < \mathrm{y}))$$

es verdadera o falsa. El dominio de discurso es el conjunto de todos los enteros.

45. Use el juego de lógica (ejemplo 1.4.15) para determinar si la proposición

$$\forall x \forall y \exists z((x < y) \rightarrow ((z > x) \wedge (\mathrm{z} < y)))$$

es verdadera o falsa. El dominio de discurso es el conjunto de todos los enteros.

46. Use el juego de lógica (ejemplo 1.4.15) para determinar si la proposición

$$\forall x \forall y \exists z((x < y) \rightarrow ((z > x) \wedge (z < y)))$$

es verdadera o falsa. El dominio de discurso es el conjunto de todos los números reales.

Suponga que $\forall x \forall y\, P(x, y)$ es verdadera y que el dominio de discurso es no vacío. ¿Cuál de los ejercicios entre el 47 y el 49 debe ser verdadero también? Si la afirmación es verdadera, explique, de otra manera, dé un contraejemplo.

47. $\forall x \exists y\, P(x, y)$

48. $\exists x \forall y\, P(x, y)$

49. $\exists x \exists y\, P(x, y)$

Suponga que $\forall x \exists y\, P(x, y)$ *es verdadera y que el dominio de discurso es no vacío. ¿Cuál de los ejercicios entre el 50 y el 52 debe ser verdadero también? Si la afirmación es verdadera, explique de otra manera, dé un contraejemplo.*

50. $\forall x \forall y\, P(x, y)$

51. $\exists x \forall y\, P(x, y)$

52. $\exists x \exists y\, P(x, y)$

Suponga que $\exists x \forall y\, P(x, y)$ *es verdadera y que el dominio de discurso es no vacío. ¿Cuál de los ejercicios entre el 53 y el 55 debe ser verdadero también? Si la afirmación es verdadera, explique; de otra manera, dé un contraejemplo.*

53. $\forall x \forall y\, P(x, y)$

54. $\forall x \exists y\, P(x, y)$

55. $\exists x \exists y\, P(x, y)$

Suponga que $\exists x \exists y\, P(x, y)$ *es verdadera y que el dominio de discurso es no vacío. ¿Cuál de los ejercicios entre el 56 y el 58 debe ser verdadero también? Si la afirmación es verdadera, explique; de otra manera, dé un contraejemplo.*

56. $\boldsymbol{\forall x \forall y\, P(x, y)}$

57. $\boldsymbol{\forall x \exists y\, P(x, y)}$

58. $\boldsymbol{\exists x \forall y\, P(x, y)}$

¿Cuál de los ejercicios entre el 59 y el 62 es lógicamente equivalente a $\neg(\forall x \exists y\, P(x, y))$? Explique.

59. $\exists x \neg(\forall y\, P(x, y))$

60. $\forall x \neg(\exists y\, P(x, y))$

61. $\exists x \forall y \neg P(x, y)$

62. $\exists x \exists y \neg P(x, y)$

1.5 ➜ Demostraciones

WWW

Un **sistema matemático** consiste en **axiomas, definiciones** y **términos no definidos**. Se supone que los axiomas son verdaderos. Las definiciones se usan para crear nuevos conceptos en términos de los existentes. Algunos términos no están definidos de forma explícita por los axiomas. Dentro de un sistema matemático es posible derivar teoremas. Un **teorema** es una proposición que se ha probado que es verdadera. Algunos tipos especiales de teoremas reciben el nombre de lemas y corolarios. Un **lema** es un teorema que no suele ser muy interesante por sí mismo, pero que resulta útil para probar otro teorema. Un **corolario** es un teorema que se deriva con facilidad de otro teorema.

Un argumento que establece la verdad de un teorema se llama **demostración o prueba**. La lógica es la herramienta para el análisis de las demostraciones. En esta sección se describen algunos métodos generales de demostración y se usa la lógica para analizar

argumentos válidos e inválidos. En las secciones 1.6 a la 1.8, se estudia la solución y la inducción matemática, que son técnicas especiales de demostración. Se comenzará por dar algunos ejemplos de sistemas matemáticos.

Ejemplo 1.5.1 ▶

La geometría euclidiana proporciona un ejemplo de un sistema matemático. Entre los axiomas están los siguientes:

- Si se tienen dos puntos distintos, existe exactamente una línea que los contiene.
- Si se tienen una línea y un punto fuera de la línea, existe exactamente una línea paralela a la línea que pasa por el punto.

Los términos *punto* y *línea* son términos no definidos que están implícitamente definidos por los axiomas que describen sus propiedades.

Entre las definiciones se encuentran:

- Dos triángulos son *congruentes* si sus vértices se pueden aparear de manera que los lados correspondientes y los ángulos correspondientes sean iguales.
- Dos ángulos son *suplementarios* si la suma de sus medidas es 180°. ◀

Ejemplo 1.5.2 ▶

Los números reales proporcionan otro ejemplo de un sistema matemático. Entre los axiomas están los siguientes:

- Para todo número real x y y, $xy = yx$
- Existe un subconjunto **P** de números reales que satisface

a) Si x y y están en **P**, entonces $x + y$ y xy están en **P**.
b) Si x es un número real, entonces exactamente una de las siguientes afirmaciones es verdadera:

$$x \text{ está en } \mathbf{P}, \quad x = 0, \quad -x \text{ está en } \mathbf{P}.$$

La multiplicación está definida de forma implícita por el primer axioma y por otros que describen las propiedades que se supone que tiene la multiplicación.

Entre las definiciones están:

- Los elementos de **P** (del axioma anterior) se llaman *números reales positivos*.
- El *valor absoluto* $|x|$ de un número real x está definido como x si x es positivo o 0, y $-x$ de otra manera. ◀

Se darán varios ejemplos de teoremas, corolarios y lemas en la geometría euclidiana y en el sistema de números reales.

Ejemplo 1.5.3 ▶

Algunos ejemplos de teoremas en la geometría euclidiana son:

- Si dos lados de un triángulo son iguales, entonces los ángulos opuestos a ellos son iguales.
- Si las diagonales de un cuadrilátero se bisectan entre sí, entonces el cuadrilátero es un paralelogramo. ◀

Ejemplo 1.5.4 ▶

Un ejemplo de corolario en la geometría euclidiana es:

- Si un triángulo es equilátero, entonces es equiangular.

Este corolario se deriva de inmediato del primer teorema del ejemplo 1.5.3. ◀

Ejemplo 1.5.5 ▶

Dos ejemplos de teoremas acerca de números reales son:

- $x \cdot 0 = 0$ para todo número real x.
- Para todos los números reales x, y y z, si $x \leq y$ y $y \leq z$, entonces $x \leq z$. ◀

Ejemplo 1.5.6 ▶

Un ejemplo de un lema acerca de números reales es:

- Si n es un entero positivo, entonces $n - 1$ es un entero positivo o $n - 1 = 0$.

Por supuesto, este resultado no es interesante por sí mismo, pero puede usarse para probar otros resultados. ◀

Con frecuencia los teoremas son de la forma

Para toda $x_1, x_2, \ldots, x_n$, si $p(x_1, x_2, \ldots, x_n)$, entonces $q(x_1, x_2, \ldots, x_n)$.

Esta proposición cuantificada universalmente es verdadera siempre que la proposición condicional

$$\text{si } p(x_1, x_2, \ldots, x_n), \text{ entonces } q(x_1, x_2, \ldots, x_n) \qquad \textbf{(1.5.1)}$$

sea verdadera para todo $x_1, x_2, \ldots, x_n$ en el dominio de discurso. Para probar (1.5.1), se supone que $x_1, x_2, \ldots, x_n$ son miembros arbitrarios del dominio de discurso. Si $p(x_1, x_2, \ldots, x_n)$ es falsa, por la definición 1.2.3, (1.5.1) es trivialmente cierta; así, sólo se necesita considerar el caso en que $p(x_1, x_2, \ldots, x_n)$ es verdadera. Una **prueba directa** supone que $p(x_1, x_2, \ldots, x_n)$ es verdadera, y después, usando $p(x_1, x_2, \ldots, x_n)$ junto con otros axiomas, definiciones y teoremas derivados antes, demuestra directamente que $q(x_1, x_2, \ldots, x_n)$ es verdadera.

Todos "saben" qué es un entero par o impar, pero la siguiente definición hace estos términos precisos y proporciona una manera formal de usar los términos "entero par" y "entero impar" en las demostraciones.

Definición 1.5.7 ▶

Un entero n es *par* si existe un entero k tal que $n = 2k$. Un entero n es *impar* si existe un entero k tal que $n = 2k + 1$. ◀

Ejemplo 1.5.8 ▶

El entero $n = 12$ es par porque existe un entero k (a saber, $k = 6$) tal que $n = 2k$; es decir, $12 = 2 \cdot 6$. ◀

Ejemplo 1.5.9 ▶

El entero $n = -21$ es impar porque existe un entero k (a saber $k = -11$) tal que $n = 2k + 1$; es decir, $-21 = 2 \cdot -11 + 1$. ◀

Ejemplo 1.5.10 ▶

Se dará una prueba directa de la siguiente afirmación. Para cualesquiera enteros m y n, si m es impar, y n es par, entonces $m + n$ es impar.

Demostración Se supone que m y n son enteros arbitrarios y que

$$m \text{ es impar y } n \text{ es par}$$

es verdadera. Se puede probar que

$$m + n \text{ es impar}$$

es verdadera. Por definición, como m es impar, existe un entero k_1 tal que $m = 2k_1 + 1$. También por definición, como n es par, existe un entero k_2 tal que $n = 2k_2$. (Observe que *no* se supone que $k_1 = k_2$). Ahora la suma es

$$m + n = (2k_1 + 1) + (2k_2) = 2(k_1 + k_2) + 1$$

Entonces, existe un entero k (a saber $k = k_1 + k_2$) tal que $m + n = 2k + 1$. Por lo tanto, $m + n$ es impar. ◀

Ejemplo 1.5.11 ▶

Si a y b son números reales, se define mín$\{a, b\}$ como el *mínimo de a* y b o el valor común si son iguales. De modo más preciso,

$$\text{mín}\{a, b\} = \begin{cases} a \text{ si } a < b \\ a \text{ si } a = b \\ b \text{ si } b < a. \end{cases}$$

Se presentará una prueba directa de la siguiente afirmación. Para cualesquiera números reales d, d_1, d_2, x,

$$\text{si } d = \text{mín}\{d_1, d_2\} \text{ y } x \le d, \text{ entonces } x \le d_1, \text{ y } x \le d_2.$$

Demostración Se supone que d, d_1, d_2 y x son números reales arbitrarios y que

$$d = \text{mín}\{d_1, d_2\} \text{ y } x \le d$$

es verdadera. Entonces se prueba que

$$x \le d_1 \text{ y } x \le d_2$$

es verdadera.

De la definición de mín, se tiene que $d \le d_1$ y $d \le d_2$. De $x \le d$ y $d \le d_1$, se puede derivar $x \le d_1$ de un teorema anterior (el segundo teorema del ejemplo 1.5.5). De $x \le d$ y $d \le d_2$, se puede derivar $x \le d_2$ a partir del mismo teorema anterior. Por lo tanto, $x \le d_1$ y $x \le d_2$. ◀

Una segunda técnica de demostración es la **prueba por contradicción**. Una prueba por contradicción establece (1.5.1) suponiendo que la hipótesis p es cierta y que la conclusión q es falsa y entonces, usando p y $\neg q$ junto con otros axiomas, definiciones y teoremas derivados antes, llega a una **contradicción**. Una contradicción es una proposición de la forma $r \wedge \neg r$ (r puede ser cualquier proposición). Una prueba por contradicción en ocasiones se llama **prueba indirecta** ya que para establecer (1.5.1) usando la prueba por contradicción, se sigue una ruta indirecta: se deriva $r \wedge \neg r$ y después se concluye que (1.5.1) es verdadera.

La única diferencia entre las suposiciones en una prueba directa y una prueba por contradicción es la conclusión negada. En una prueba directa no se supone la conclusión negada, mientras que en una prueba por contradicción la conclusión negada es una suposición.

La prueba por contradicción se justifica observando que las proposiciones

$$p \to q \quad \text{y} \quad (p \wedge \neg q) \to (r \wedge \neg r)$$

son equivalentes. La equivalencia es inmediata a partir de la tabla de verdad:

p	q	r	$p \to q$	$p \wedge \neg q$	$r \wedge \neg r$	$(p \wedge \neg q) \to (r \wedge \neg r)$
V	V	V	V	F	F	V
V	V	F	V	F	F	V
V	F	V	F	V	F	F
V	F	F	F	V	F	F
F	V	V	V	F	F	V
F	V	F	V	F	F	V
F	F	V	V	F	F	V
F	F	F	V	F	F	V

Ejemplo 1.5.12 ▶

Se dará una prueba por contradicción de la siguiente afirmación:

Para todos los números reales x y y, si $x + y \ge 2$, entonces $x \ge 1$ o $y \ge 1$.

Demostración Primero, sean x y y números reales arbitrarios. Se supondrá que la conclusión es falsa, es decir, que $\neg(x \ge 1 \vee y \ge 1)$ es verdadera. Por las leyes de De Morgan de lógica (vea el ejemplo 1.2.11),

$$\neg(x \ge 1 \vee y \ge 1) \equiv \neg(x \ge 1) \wedge \neg(y \ge 1) \equiv (x < 1) \wedge (y < 1).$$

En palabras, se está suponiendo que $x < 1$ y que $y < 1$. Usando un teorema anterior, se pueden sumar estas desigualdades para obtener

$$x + y < 1 + 1 = 2.$$

En este punto, se ha obtenido la contradicción $p \wedge \neg p$, donde

$$p\text{: } x + y \geq 2.$$

Por lo tanto, concluimos que la afirmación es verdadera. ◀

Suponga que se da una prueba por contradicción de (1.5.1) en donde, igual que en el ejemplo 1.5.12, se deduce $\neg p$. De hecho, se probó

$$\neg q \rightarrow \neg p.$$

Este caso especial por contradicción se llama **prueba por contrapositiva**.

Ejemplo 1.5.13 ▶

Use la prueba por contrapositiva para demostrar que

para todo entero m, si m^2 es impar, entonces m es impar.

Demostración Primero, sea m un entero arbitrario. La contrapositiva de

si m^2 es impar, entonces m es impar

es

si m es par, entonces m^2 es no es impar

o, de manera equivalente,

si m es par, entonces m^2 es par.

Así, suponga que m es par. Entonces $m = 2k$ para algún entero k. Ahora, $m^2 = (2k)^2 = 2 \cdot (2k^2)$. Como m^2 es de la forma $2 \times$ un entero (a saber $2k^2$), m^2 es par. La prueba queda completa. ◀

La **prueba por casos** se emplea cuando la hipótesis original se divide de manera natural en varios casos. Por ejemplo, la hipótesis "x es un número real" se puede dividir en casos: a) x es un número real no negativo y b) x es un número real negativo. Suponga que la tarea es probar $p \rightarrow q$ y que p es equivalente a $p_1 \vee p_2 \vee \ldots \vee p_n$ ($p_1, \ldots, p_n$ son los casos). En lugar de probar

$$(p_1 \vee p_2 \vee \ldots \vee p_n) \rightarrow q, \tag{1.5.2}$$

se prueba

$$(p_1 \rightarrow q) \wedge (p_2 \rightarrow q) \wedge \ldots \wedge (p_n \rightarrow q). \tag{1.5.3}$$

Como se verá, la prueba por casos se justifica porque las dos afirmaciones son equivalentes.

Primero suponga que alguna p_i es verdadera. En especial, se supone que p_j es cierta. Entonces

$$p_1 \vee p_2 \vee \ldots \vee p_n$$

es verdadera. Si q es verdadera, entonces (1.5.2) es verdadera. Ahora, $p_i \rightarrow q$ es verdadera para toda i; de manera que (1.5.3) también es verdadera. Si q es falsa, entonces (1.5.2) es falsa. Como $p_j \rightarrow q$ es falsa, (1.5.3) también es falsa.

Ahora suponga que ninguna p_i es verdadera; esto es, todas las p_i son falsas. Entonces ambas, (1.5.2) y (1.5.3), son ciertas. Por lo tanto, (1.5.2) y (1.5.3) son equivalentes.

Ejemplo 1.5.14 ▶

Se probará que para todo número real x, $x \leq |x|$.

Demostración Observe que "x es un número real" es equivalente a $(x \geq 0) \vee (x < 0)$. Una afirmación "o" con frecuencia lleva por sí misma a una prueba por casos. El caso 1 es $x \geq 0$

y el caso 2 es $x < 0$. La prueba se divide en dos casos porque la propia definición de valor absoluto está dividida en los casos $x \geq 0$ y $x < 0$ (vea el ejemplo 1.5.2).

Si $x \geq 0$, por definición $|x| = x$. Así, $|x| \geq x$. Si $x < 0$, por definición $|x| = -x$. Como $|x| = -x > 0$ y $0 > x$, $|x| \geq x$. En cualquier caso, $|x| \geq x$, y esto completa la prueba. ◀

Algunos teoremas son de la forma

$$p \text{ si y sólo si } q.$$

Esos teoremas se demuestran usando la equivalencia (vea el ejemplo 1.2.15)

$$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$$

es decir, para probar "p si y sólo si q" se prueba "si p entonces q" y "si q entonces p".

Ejemplo 1.5.15 ▶ Demuestre que para todo entero n, n es impar si y sólo si $n - 1$ es par.

Demostración Sea n un entero arbitrario. Debe probarse que

$$\text{si } n \text{ es impar, entonces } n - 1 \text{ es par}$$

y

$$\text{si } n - 1 \text{ es par, entonces } n \text{ es impar.}$$

Primero se demostrará que

$$\text{si } n \text{ es impar, entonces } n - 1 \text{ es par.}$$

Si n es impar, entonces $n = 2k + 1$ para algún entero k. Ahora

$$n - 1 = (2k + 1) - 1 = 2k.$$

Por lo tanto, $n - 1$ es par.

Ahora se demostrará que

$$\text{si } n - 1 \text{ es par entonces } n \text{ es impar.}$$

Si $n - 1$ es par, entonces $n - 1 = 2k$ para algún entero k. Ahora

$$n = 2k + 1,$$

por lo tanto, n es par y la demostración está completa. ◀

Ejemplo 1.5.16 ▶ Demuestre que para todo número real x y todo número real positivo d,

$$|x| < d \text{ si y sólo si } -d < x < d.$$

Demostración Sea x un número real arbitrario y d un número real positivo. Debemos demostrar que

$$\text{si } |x| < d \text{ entonces } -d < x < d$$

y

$$\text{si } -d < x < d \text{ entonces } |x| < d.$$

Para demostrar

$$\text{si } |x| < d \text{ entonces } -d < x < d,$$

se usa la prueba por casos. Se supone que $|x| < d$. Si $x \geq 0$, entonces

$$-d < 0 \leq x = |x| < d.$$

Si $x < 0$, entonces

$$-d < 0 < -x = |x| < d;$$

es decir,

$$-d < x < d.$$

Multiplicando por –1, se obtiene

$$d > x > -d.$$

En cualquier caso, se ha probado que

$$-d < x < d.$$

Para demostrar que

$$\text{si } -d < x < d \text{ entonces } |x| < d,$$

también se usa la prueba por casos. Se supone que $-d < x < d$. Si $x \geq 0$, entonces

$$|x| = x < d.$$

Si $x < 0$, entonces $|x| = -x$. Como $-d < x$, se multiplica por –1 para obtener $d > -x$. Combinando $|x| = -x$ y $d > -x$ se tiene

$$|x| = -x < d.$$

En cualquier caso, se ha probado que

$$|x| < d.$$

La prueba queda completa ◀

Las técnicas de demostración presentadas hasta ahora se aplican a afirmaciones con cuantificadores universales. En la sección 1.3 se muestra que para probar

$$\exists x\, P(x)$$

simplemente se necesita encontrar un elemento x del dominio de discurso que haga $P(x)$ verdadera. Tal prueba se llama **prueba existencial.**

Ejemplo 1.5.17 ▶ Sean a y b números reales con $a < b$. Pruebe que existe un número real x que satisface $a < x < b$.

Demostración Es suficiente encontrar un número real x que satisfaga $a < x < b$. El número real

$$x = \frac{a+b}{2},$$

a la mitad entre a y b, sin duda satisface $a < x < b$. ◀

Ejemplo 1.5.18 ▶ Demuestre que existe un número primo p tal que $2^p - 1$ es compuesto (es decir, *no es primo*).

Demostración Por prueba y error, se encuentra que $2^p - 1$ es primo para $p = 2, 3, 5, 7$ pero no para $p = 11$, ya que

$$2^{11} - 1 = 2048 - 1 = 2047 = 23 \cdot 89.$$ ◀

WWW Un número primo de la forma $2^p - 1$, donde p es primo, se llama *primo de Mersenne* [en honor de Marin Mersenne (1588–1648)]. Los primos más grandes conocidos son primos de Mersenne. A fines de 2003, se encontró el cuadragésimo número primo de Mersenne, $2^{20,996,011}$, un número que tiene 6,320,430 dígitos decimales. Este número fue encontrado por

el Great Internet Mersenne Prime Search (GIMPS). GIMPS es un programa de computadora distribuido en muchas computadoras personales que mantienen voluntarios. Usted puede participar. Sólo entre al enlace en Internet. ¡Podría encontrar el siguiente primo de Mersenne!

Recuerde (vea la sección 1.3) que para *refutar*

$$\forall x\, P(x)$$

simplemente se necesita encontrar un miembro x del dominio de discurso para el que $P(x)$ es falsa. Ese valor de x se llama *contraejemplo*.

Ejemplo 1.5.19 ▶ La afirmación

$$\forall n(2^n + 1 \text{ es primo})$$

es falsa. Un contraejemplo es $n = 3$ ya que $2^3 + 1 = 9$, que no es primo. ◀

Al construir una prueba, debemos asegurarnos de que los argumentos que se usan son **válidos**. Se precisará el concepto de argumento válido y se analizará con cierto detalle.

Considere la siguiente secuencia de proposiciones.

La falla está en el módulo 17 o en el módulo 81.

La falla es un error numérico.

El módulo 81 no tiene error numérico. **(1.5.4)**

Suponiendo que estas afirmaciones son verdaderas, es razonable concluir

La falla está en el módulo 17. **(1.5.5)**

Este proceso de obtener conclusiones de una secuencia de proposiciones se llama **razonamiento deductivo**. Las proposiciones que se dan, como (1.5.4), se llaman **hipótesis** o **premisas**, y la proposición que se deriva de la hipótesis, como (1.5.5), se llama **conclusión**. Un **argumento (deductivo)** consiste en hipótesis junto con una conclusión. Muchas demostraciones en matemáticas y ciencias de la computación son argumentos deductivos.

Cualquier argumento tiene la forma

Si p_1 y p_2 y ... y p_n, entonces q. **(1.5.6)**

Se dice que el argumento (1.5.6) es válido si la conclusión se obtiene de la hipótesis; es decir, si p_1 y p_2 y ... y p_n son verdaderas, entonces q también debe ser verdadera. Este análisis motiva la siguiente definición.

Definición 1.5.20 ▶ Un *argumento* es una secuencia de proposiciones escritas

$$\begin{array}{l} p_1 \\ p_2 \\ \vdots \\ \underline{p_n} \\ \therefore q \end{array}$$

o

$$p_1, p_2, \ldots, p_n / \therefore q.$$

El símbolo $\therefore$ se lee "por lo tanto". Las proposiciones $p_1, p_2, \ldots, p_n$ se conocen como *hipótesis* (o *premisas*), y la proposición q recibe el nombre de *conclusión*. El argumento es *válido* siempre y cuando, si p_1 y p_2 y $\ldots$ y p_n son todas verdaderas, entonces q también es verdadera; de otra manera, el argumento es *inválido* (o una *falacia*). ◀

WWW

En un argumento válido, a veces se dice que la conclusión se deriva de la hipótesis. Observe que no se está afirmando que la conclusión es cierta; sólo se dice que si se garan-

tiza la hipótesis, también se debe garantizar la conclusión. Un argumento es válido por su forma no por su contenido.

Ejemplo 1.5.21 ▶ Determine si el argumento

$$\begin{array}{l} p \to q \\ p \\ \hline \therefore q \end{array}$$

es válido.

[Primera solución] Se construye una tabla de verdad para todas las proposiciones implicadas:

p	q	$p \to q$	p	q
V	V	V	V	V
V	F	F	V	F
F	V	V	F	V
F	F	V	F	F

Se observa que cuando las hipótesis $p \to q$ y p son verdaderas, la conclusión q también es verdadera; por lo tanto, el argumento es válido.

[Segunda solución] Es posible evitar escribir la tabla de verdad con una verificación directa de que cuando las hipótesis son verdaderas, la conclusión también lo es.

Suponga que $p \to q$ y p son verdaderas. Entonces q debe ser verdadera, pues de otra manera $p \to q$ sería falsa. Por lo tanto, el argumento es válido. ◀

Ejemplo 1.5.22 ▶ Represente el argumento

Si $2 = 3$, entonces yo me comí mi sombrero.
Me comí mi sombrero.
$\therefore 2 = 3$

simbólicamente y determine si el argumento es válido.

Si se hace

p: $2 = 3$, q: Me comí mi sombrero,

el argumento se escribe como

$$\begin{array}{l} p \to q \\ q \\ \hline \therefore p \end{array}$$

Si el argumento es válido, entonces siempre que $p \to q$ y q sean ciertas ambas, p también debe ser cierta. Suponga que $p \to q$ y q son verdaderas. Esto es posible si p es falsa y q es verdadera. En este caso, p no es verdadera; así, el argumento es inválido. ◀

También es posible determinar la validez del argumento en el ejemplo 1.5.22 examinando la tabla de verdad del ejemplo 1.5.21. En el tercer renglón de la tabla, las hipótesis son verdaderas y la conclusión es falsa; así, el argumento es inválido.

Como se explicó antes, una prueba usa las hipótesis, axiomas, definiciones, etcétera para llegar a una conclusión. Cada paso de la prueba incluye sacar conclusiones inmediatas. Para que la prueba como un todo sea válida, cada paso debe dar como resultado una conclusión intermedia válida. Cuando se construye una prueba, con frecuencia se usa la intuición como guía para sacar conclusiones intermedias válidas; sin embargo, es factible formalizar el proceso. Este capítulo concluye con la observación cuidadosa de las **reglas de inferencia**, que son argumentos válidos breves que se utilizan dentro de argumentos más largos como una demostración.

Regla de inferencia	Nombre	Regla de inferencia	Nombre
$p \to q$ p $\therefore q$	*Modus ponens*	p q $\therefore p \wedge q$	Conjunción
$p \to q$ $\neg q$ $\therefore \neg p$	*Modus tollens*	$p \to q$ $q \to r$ $\therefore p \to r$	Silogismo hipotético
p $\therefore p \vee q$	Suma	$p \vee q$ $\neg p$ $\therefore q$	Silogismo disyuntivo (*Tollens ponens*)
$p \wedge q$ $\therefore p$	Simplificación		

Figura 1.5.1 Reglas de inferencia para proposiciones.

Reglas de inferencia para proposiciones

En el ejemplo 1.5.21, se vio que el argumento

$$\begin{array}{l} p \to q \\ p \\ \hline \therefore q \end{array}$$

es válido. Esta regla de inferencia se llama **modus ponens** o **ley de separación**. Otras reglas de inferencia para proposiciones, que pueden verificarse usando las tablas de verdad (vea los ejercicios 59 al 64), se enumeran en la figura 1.5.1.

Ejemplo 1.5.23 ▶

¿Cuál regla de inferencia se usa en el siguiente argumento?

Si la computadora tiene 32 megabytes de almacenamiento, entonces puede correr "como ráfaga". Si la computadora puede trabajar "como ráfaga", entonces el sonido será impresionante. Por lo tanto, si la computadora tiene 32 megabytes de almacenamiento, entonces el sonido será impresionante.

Sea p la proposición "la computadora tiene 32 megabytes de almacenamiento", sea q la proposición "la computadora puede correr 'como ráfaga', y sea r la proposición "el sonido será impresionante". El argumento se escribe en símbolos como

$$\begin{array}{l} p \to q \\ q \to r \\ \hline \therefore p \to r \end{array}$$

Por lo tanto, el argumento usa la regla de inferencia de silogismo hipotético. ◀

Ejemplo 1.5.24 ▶

Se tiene la siguiente hipótesis: si los Cargadores obtienen un buen defensa, entonces los Cargadores pueden derrotar a los Broncos. Si los Cargadores pueden derrotar a los Broncos, entonces los Cargadores pueden derrotar a los Jets. Si los Cargadores pueden derrotar a los Broncos, entonces los Cargadores pueden derrotar a los Delfines. Los Cargadores obtienen un buen defensa. Demuestre usando las reglas de inferencia (vea la figura 1.5.1) que la conclusión, los Cargadores pueden derrotar a los Jets y los Cargadores pueden derrotar a los Delfines, se deriva de la hipótesis.

Sea p la proposición "los Cargadores pueden obtener un buen defensa", sea q la proposición "los Cargadores pueden derrotar a los Broncos", sea r la proposición "los Cargadores

pueden derrotar a los Jets" y sea s la proposición "los Cargadores pueden derrotar a los Delfines". Entonces las hipótesis son:

$$p \rightarrow q$$
$$q \rightarrow r$$
$$q \rightarrow s$$
$$p.$$

A partir de $p \rightarrow q$ y $q \rightarrow r$, se usa el silogismo hipotético para concluir que $p \rightarrow r$. A partir de $p \rightarrow r$ y p, se utiliza el modus ponens para concluir r. A partir de $p \rightarrow q$ y $q \rightarrow s$, se emplea el silogismo hipotético para concluir que $p \rightarrow s$. A partir de $p \rightarrow s$ y p, se usa el modus ponens para concluir s. A partir de r y s, se utiliza la conjunción para concluir $r \wedge s$. Como $r \wedge s$ representa la proposición "los Cargadores pueden derrotar a los Jets y los Cargadores pueden derrotar a los Delfines", se sabe que la conclusión sí se deriva de la hipótesis. ◀

Reglas de inferencia para afirmaciones cuantificadas

Suponga que $\forall x\, P(x)$ es verdadera. Por la definición 1.3.4, $P(x)$ es verdadera para toda x en D, el dominio de discurso. En particular, si d está en D, entonces $P(d)$ es cierta. Se ha demostrado que el argumento

$$\frac{\forall x\, P(x)}{\therefore P(d) \text{ si } d \text{ está en } D}$$

es válido. Esta regla de inferencia se llama particularización **universal**. Argumentos similares (vea los ejercicios 65 al 67) justifican las otras reglas de inferencia que aparecen en la figura 1.5.2.

Reglas de inferencia	Nombre
$\forall x\, P(x)$ / $\therefore P(d)$ si d está en D	Particularización universal
$P(d)$ para toda d en D / $\therefore \forall x\, P(x)$	Generalización universal
$\exists x\, P(x)$ / $\therefore P(d)$ para alguna d en D	Particularización existencial
$P(d)$ para alguna d en D / $\therefore \exists x\, P(x)$	Generalización existencial

Figura 1.5.2 Reglas de inferencia para afirmaciones cuantificadas. El dominio de discurso es D.

Ejemplo 1.5.25 ▶ Se tienen las siguientes hipótesis: Todos aman a Microsoft o a Apple. Lina no ama a Microsoft. Demuestre que la conclusión, Lina ama a Appple se deriva de las hipótesis.

Sea $P(x)$ la función proposicional "x ama a Microsoft" y sea $Q(x)$ la función proposicional "x ama a Apple". La primera hipótesis es $\forall x\, P(x) \vee Q(x)$. Por la particularización universal, se tiene que $P(\text{Lina}) \vee Q(\text{Lina})$. La segunda hipótesis es $\neg P(\text{Lina})$. La regla de inferencia del silogismo disyuntivo (vea la figura 1.5.1) ahora da $Q(\text{Lina})$, que representa la proposición "Lina ama a Apple". Determinamos entonces que la conclusión sí se obtiene de las hipótesis. ◀

Sugerencias para resolver problemas

Elaborar una demostración es más que un procedimiento metódico. La práctica y la experiencia son importantes. De cualquier manera, existen algunas sugerencias generales.

Determine si la afirmación que se va a probar está cuantificada universalmente (lo más común) o existencialmente. Si la afirmación está universalmente cuantificada (es decir, para toda x ...), para construir una prueba directa haga que x denote un elemento *arbitrario* del dominio de discurso y argumente en términos de x. Aunque los valores específicos de x resultan útiles para comprender mejor lo que se debe probar, demostrar que esa afirmación es cierta para valores específicos de x no es una demostración.

Si la afirmación está cuantificada existencialmente (es decir, existe x . . .), la demostración consistirá en encontrar un valor de x para el que la afirmación sea verdadera o en dar una prueba indirecta de que ese valor existe. En este caso, un valor específico de x es justo lo que se necesita.

Antes de intentar una demostración, recuerde lo que conoce acerca de los términos, valores, etcétera. Por ejemplo, si la hipótesis se refiere a todo entero par n, sabemos que n es de la forma $2k$ para algún entero k.

Revise las diferentes técnicas de demostración de esta sección. Si desea construir una prueba directa de una afirmación de la forma $p \rightarrow q$ y parece estar atorado, intente una prueba por contradicción. Con ella tendrá más elementos con los cuales trabajar: además de suponer p, puede también suponer $\neg q$.

La prueba por casos es útil si las hipótesis se separan en partes de manera natural. Por ejemplo, si la afirmación que va a probar contiene el valor absoluto de x, tal vez quiera considerar los casos $x \geq 0$ y $x < 0$ ya que $|x|$ mismo se define por los casos $x \geq 0$ y $x < 0$.

Por último, recuerde que para probar p si y sólo si q, debe probar dos afirmaciones: 1) si p entonces q y 2) si q entonces p.

Sección de ejercicios de repaso

1. ¿Qué es un sistema matemático?
2. ¿Qué es un axioma?
3. ¿Qué es una definición?
4. ¿Qué es un término no definido?
5. ¿Qué es un teorema?
6. ¿Qué es una demostración?
7. ¿Qué es un lema?
8. ¿Qué es una prueba directa?
9. ¿Cuál es la definición formal de "entero par"?
10. ¿Cuál es la definición formal de "entero impar"?
11. ¿Qué es una prueba por contradicción?
12. ¿Qué es una prueba indirecta?
13. ¿Qué es una prueba por contrapositiva?
14. ¿Qué es una demostración por casos?
15. ¿Qué es una prueba de existencia?
16. ¿Qué es el razonamiento deductivo?
17. ¿Qué es una hipótesis en un argumento?
18. ¿Qué es una premisa en un argumento?
19. ¿Qué es una conclusión en un argumento?
20. ¿Qué es un argumento válido?
21. Qué es un argumento inválido?
22. Establezca la regla de inferencia *modus ponens*.
23. Establezca la regla de inferencia *modus tollens*.
24. Establezca la regla de inferencia de suma.
25. Establezca la regla de inferencia de simplificación.
26. Establezca la regla de inferencia de conjunción.
27. Establezca la regla de inferencia de silogismo hipotético.
28. Establezca la regla de inferencia de silogismo disyuntivo.
29. Establezca la regla de inferencia del particularización universal.
30. Establezca la regla de inferencia de generalización universal.
31. Establezca la regla de inferencia del particularización existencial.
32. Establezca la regla de inferencia de generalización existencial.

Ejercicios

1. Dé un ejemplo (diferente a los del ejemplo 1.5.1) de un axioma de la geometría euclidiana.
2. Dé un ejemplo (diferente a los del ejemplo 1.5.2) de un axioma del sistema de los números reales.
3. Dé un ejemplo (diferente a los del ejemplo 1.5.1) de una definición en la geometría euclidiana.
4. Dé un ejemplo (diferente a los del ejemplo 1.5.2) de una definición en el sistema de los números reales.
5. Dé un ejemplo (diferente a los del ejemplo 1.5.3) de un teorema en la geometría euclidiana.
6. Dé un ejemplo (diferente a los del ejemplo 1.5.5) de un teorema en el sistema de los números reales.

7. Pruebe que para todos los enteros m y n, si m y n son pares, entonces $m + n$ es par.

8. Pruebe que para todos los enteros m y n, si m y n son impares, entonces $m + n$ es par.

9. Pruebe que para todos los enteros m y n, si m y n son pares, entonces mn es par.

10. Pruebe que para todos los enteros m y n, si m y n son impares, entonces mn es impar.

11. Pruebe que para todos los enteros m y n, si m es par y n es impar, entonces mn es par.

12. Si a y b son números reales, se define máx$\{a, b\}$ como el máximo entre a y b o el valor común si son iguales. Pruebe que para todos los números reales d, d_1, d_2, x,

si $d =$ máx$\{d_1, d_2\}$ y $x \geq d$, entonces $x \geq d_1$ y $x \geq d_2$.

13. Justifique cada paso de la siguiente prueba directa, que muestra que si x es un número real, entonces $x \cdot 0 = 0$. Suponga que los siguientes son teoremas previos: Si a, b y c son números reales, entonces $b + 0 = b$ y $a(b + c) = ab + ac$. Si $a + b = a + c$, entonces $b = c$.

Demostración $x \cdot 0 + 0 = x \cdot 0 = x \cdot (0 + 0) = x \cdot 0 + x \cdot 0$; por lo tanto, $x \cdot 0 = 0$.

14. Justifique cada paso de la siguiente demostración por contradicción, que muestra que si $xy = 0$, entonces $x = 0$ o $y = 0$. Suponga que si a, b y c son números reales con $ab = ac$ y $a \neq 0$, entonces, $b = c$.

Demostración Suponga que $xy = 0$ y $x \neq 0$ y $y \neq 0$. Como $xy = 0 = x \cdot 0$ y $x \neq 0$, $y = 0$, que es una contradicción.

15. Demuestre, por contradicción, que si se colocan 100 pelotas en nueve urnas, alguna urna contiene 12 pelotas o más.

16. Demuestre, por contradicción, que si se distribuyen 40 monedas en nueve bolsas de manera que cada bolsa contenga al menos una moneda, al menos dos bolsas contienen el mismo número de monedas.

17. Sea

$$A = \frac{s_1 + s_2 + \cdots + s_n}{n}$$

el promedio de los números reales $s_1, \ldots, s_n$. Demuestre, por contradicción, que existe i tal que $s_i \geq A$.

18. Sea

$$A = \frac{s_1 + s_2 + \cdots + s_n}{n}$$

el promedio de los números reales $s_1, \ldots, s_n$. Pruebe o desapruebe: Existe i tal que $s_i > A$. ¿Qué técnica de demostración utilizó?

19. Sea

$$A = \frac{s_1 + s_2 + \cdots + s_n}{n}$$

el promedio de los números reales $s_1, \ldots, s_n$. Suponga que existe i tal que $s_i < A$. Pruebe o desapruebe: Existe j tal que $s_j > A$. ¿Qué técnica de demostración usó?

20. Utilice la prueba por casos para demostrar que $|xy| = |x|\,|y|$ para todos los números reales x y y.

21. Utilice la prueba por casos para demostrar que $|x + y| \leq |x| + |y|$ para todos los números reales x y y.

22. Defina el *signo del número real* x, sgn(x), como

$$\text{sgn}(x) = \begin{cases} 1 & \text{si } x > 0 \\ 0 & \text{si } x = 0 \\ -1 & \text{si } x < 0. \end{cases}$$

Use la prueba por casos para demostrar que $|x| = \text{sgn}(x)x$ para todo número real x.

23. Utilice la prueba por casos para demostrar que sgn$(xy) =$ sgn(x)sgn(y) para todos los números reales x y y (sgn se define en el ejercicio 22).

24. Use los ejercicios 22 y 23 para dar otra prueba de que $|xy| = |x|\,|y|$ para todos los números reales x y y.

25. Use la prueba por casos para demostrar que máx$\{x, y\}$ + mín$\{x, y\} = x + y$ para todos los números reales x y y.

26. Utilice la demostración por casos para probar que

$$\text{máx}\{x, y\} = \frac{x + y + |x - y|}{2}$$

para todos los números reales x y y.

27. Utilice la demostración por casos para probar que

$$\text{mín}\{x, y\} = \frac{x + y - |x - y|}{2}$$

para todos los números reales x y y.

28. Utilice los ejercicios 26 y 27 para probar que máx$\{x, y\}$ + mín$\{x, y\} = x + y$ para todos los números reales x y y.

†★**29.** Sea $s_1, \ldots, s_n$ una secuencia‡ que satisface

a) s_1 es un entero positivo y s_n es un entero negativo.

b) para toda i, $1 \leq i < n$, $s_{i+1} = s_i + 1$ o $s_{i+1} = s_i - 1$.

Pruebe que existe i, $1 < i < n$, tal que $s_i = 0$.

Los estudiantes de cálculo reconocerán este ejercicio como la versión discreta del teorema de cálculo: Si f es una función continua sobre $[a, b]$ y $f(a) > 0$ y $f(b) < 0$, entonces $f(c) = 0$ para alguna c en (a, b). Existen pruebas similares de las dos afirmaciones.

30. Desapruebe la afirmación: Para todo entero positivo n, $n^2 \leq 2^n$.

Formule con símbolos los argumentos de los ejercicios 31 al 35 y determine si cada uno es válido. Sean

p: Estudio duro. q: Obtengo 10. r: Me hago rico.

31. Si estudio duro, entonces obtengo 10.
Estudio duro.

∴ obtengo 10.

32. Si estudio duro, entonces obtengo 10.
Si no me hago rico, entonces no obtengo 10.

∴ Me hago rico.

33. Estudio duro si y sólo si me hago rico.
Me hago rico

∴ estudio duro.

† El símbolo ★ antepuesto al número de un ejercicio indica un problema con dificultad más alta que el promedio.

‡ De manera informal, una *secuencia* es una lista de elementos en la que se toma en cuenta el orden, de forma que s_2 es el primer elemento, s_2 es el segundo, y así sucesivamente. La definición formal se presenta en la sección 2.3.

34. Si estudio duro o me hago rico, entonces obtengo 10.
Obtengo 10.

$\therefore$ si no estudio duro, entonces me hago rico.

35. Si estudio mucho, entonces obtengo 10 o me hago rico.
No obtengo 10 y no me hago rico.

$\therefore$ no estudio duro.

En los ejercicios 36 al 40, escriba el argumento en palabras y determine si cada argumento es válido. Sean

p: 4 megabytes es mejor que sin memoria.

q: Compraremos más memoria.

r: Compraremos una computadora nueva.

36. $p \to r$
$p \to q$

$\therefore p \to (r \wedge q)$

37. $p \to (r \vee q)$
$r \to \neg q$

$\therefore p \to r$

38. $p \to r$
$r \to q$

$\therefore q$

39. $\neg r \to \neg p$
r

$\therefore p$

40. $p \to r$
$r \to q$
p

$\therefore q$

Determine si cada argumento en los ejercicios 41 al 45 es válido.

41. $p \to q$
$\neg p$

$\therefore \neg q$

42. $p \to q$
$\neg q$

$\therefore \neg p$

43. $p \wedge \neg p$

$\therefore q$

44. $p \to (q \to r)$
$q \to (p \to r)$

$\therefore (p \vee q) \to r$

45. $(p \to q) \wedge (r \to s)$
$p \vee r$

$\therefore q \vee s$

46. Demuestre que si

$$p_1, p_2 / \therefore p \quad \text{y} \quad p, p_3, \ldots, p_n / \therefore c$$

son argumentos válidos, el argumento

$$p_1, p_2, \ldots, p_n / \therefore c$$

también es válido.

47. Comente el siguiente argumento:
Almacenar en disco flexible es mejor que nada.
Nada es mejor que un disco duro.

$\therefore$ Almacenar en disco flexible es mejor que un disco duro.

48. Analice los siguientes comentarios del crítico de cine Roger Ebert: Ninguna buena película es demasiado larga. Ninguna mala película es demasiado corta. "Amor en realidad" es buena, pero es demasiado larga.

Para cada argumento en los ejercicios 49 al 52, diga qué regla de inferencia se usó.

49. La pesca es un deporte popular. Por lo tanto, pescar es un deporte popular o el lacrosse es muy popular en California.

50. Si la pesca es un deporte popular, entonces el lacrosse es muy popular en California. La pesca es un deporte popular. Por lo tanto, el lacrosse es muy popular en California.

51. La pesca es un deporte popular o el lacrosse es muy popular en California. El lacrosse no es muy popular en California. Por lo tanto, la pesca es un deporte popular.

52. Todo número racional es de la forma p/q, donde p y q son enteros. Por lo tanto, 9.345 es de la forma p/q.

En los ejercicios 53 al 58, dé un argumento usando las reglas de inferencia para demostrar que la conclusión se deriva de las hipótesis.

53. Hipótesis: Si el auto tiene gasolina, entonces iré a la tienda. Si voy a la tienda, entonces compraré un refresco. El auto tiene gasolina. Conclusión: Compraré un refresco.

54. Hipótesis: Si el auto tiene gasolina, entonces iré a la tienda. Si voy a la tienda, entonces compararé un refresco. No compro un refresco. Conclusión: El auto no tiene gasolina o la transmisión del auto está defectuosa.

55. Hipótesis: Si Julio puede cantar o Daniel puede jugar, entonces compraré el CD. Julio puede cantar. Compraré el reproductor de CD. Conclusión: Compraré el CD y el reproductor de CD.

56. Hipótesis: Todos en clase tienen una calculadora que grafica. Todos los que tienen calculadora que grafica entienden las funciones trigonométricas. Conclusión: Rafael, que está en la clase, entiende las funciones trigonométricas.

57. Hipótesis: Ken, un miembro de los Titanes, puede batear lejos. Todos los que pueden batear lejos pueden ganar mucho dinero. Conclusión: Algún miembro de los Titanes puede ganar mucho dinero.

58. Hipótesis: Todos en la clase de matemáticas discretas aman las demostraciones. Alguien en la clase de matemáticas discretas nunca ha tomado cálculo. Conclusión: Alguien que ama las demostraciones nunca ha tomado cálculo.

59. Demuestre que el *Modus tollens* (vea la figura 1.5.1) es válido.

60. Demuestre que la Suma (vea la figura 1.5.1) es válida.

61. Demuestre que la Simplificación (vea la figura 1.5.1) es válida.

62. Demuestre que la Conjunción (vea la figura 1.5.1) es válida.

63. Demuestre que el Silogismo Hipotético (vea la figura 1.5.1) es válido.

64. Demuestre que el Silogismo Disyuntivo (vea la figura 1.5.1) es válido.

65. Demuestre que la generalización universal (vea la figura 1.5.2) es válida.

66. Demuestre que la particularización existencial (vea la figura 1.5.2) es válida.

67. Demuestre que la generalización existencial (vea la figura 1.5.2) es válida.

1.6 ➜ Pruebas por resolución†

En esta sección, $a \wedge b$ se escribirá como ab.

Resolución es una técnica de prueba propuesta por J. A. Robinson en 1965 (vea [Robinson]) que depende de una sola regla:

Si $p \vee q$ y $\neg p \vee r$ son ambas verdaderas, entonces $q \vee r$ es verdadera. **(1.6.1)**

La afirmación (1.6.1) se verifica escribiendo la tabla de verdad (vea el ejercicio 1). Como la resolución depende de esta única regla sencilla, es la base de muchos programas de computadora que razonan y prueban teoremas.

En una prueba por resolución, las hipótesis y la conclusión se escriben como **cláusulas**. Una cláusula consiste en términos separados por o, donde cada término es una variable o la negación de una variable.

Ejemplo 1.6.1 ▶ La expresión

$$a \vee b \vee \neg c \vee d$$

es una cláusula, ya que los términos a, b, $\neg c$ y d están separados por o, y cada término es una variable o la negación de una variable. ◀

Ejemplo 1.6.2 ▶ La expresión

$$xy \vee w \vee \neg z$$

no es una cláusula aunque los términos estén separados por o, ya que el término xy tiene dos variables, no una sola. ◀

Ejemplo 1.6.3 ▶ La expresión

$$p \rightarrow q$$

no es una cláusula, ya que los términos están separados por $\rightarrow$. Sin embargo, cada término es una variable. ◀

Una prueba directa por resolución procede aplicando (1.6.1), repetidas veces, a pares de afirmaciones para derivar nuevas afirmaciones hasta llegar a la conclusión. Cuando se aplica (1.6.1), p debe ser una variable sencilla, pero q y r pueden ser expresiones. Observe que cuando se aplica (1.6.1) a cláusulas, el resultado $q \vee r$ es una cláusula. (Como q y r están formadas por términos separados por o, donde cada término es una variable o la negación de una variable, $q \vee r$ también consiste en términos separados por o, donde cada término es una variable o la negación de una variable.)

Ejemplo 1.6.4 ▶ Se demostrará lo siguiente usando resolución:

1. $a \vee b$
2. $\neg a \vee c$
3. $\neg c \vee d$

$\therefore b \vee d$

Al aplicar (1.6.1) a las expresiones 1 y 2, se deriva

4. $b \vee c$.

† Esta sección se puede omitir sin pérdida de continuidad.

Al aplicar (1.6.1) a las expresiones 3 y 4, se obtiene

5. $b \vee d$,

la conclusión deseada. A partir de las hipótesis 1, 2 y 3, se probó la conclusión $b \vee d$. ◀

Algunos casos especiales de (1.6.1) son los siguientes:

Si $p \vee q$ y $\neg p$ son verdaderas, entonces q es verdadera.
Si p y $\neg p \vee r$ son verdaderas, entonces r es verdadera. **(1.6.2)**

Ejemplo 1.6.5 ▶

Se demostrará lo siguiente usando resolución:

1. a
2. $\neg a \vee c$
3. $\underline{\neg c \vee d}$
$\therefore d$

Al aplicar (1.6.2) a las expresiones 1 y 2, se tiene

4. c.

Al aplicar (1.6.2) a las expresiones 3 y 4, se deriva

5. d,

la conclusión deseada. A partir de las hipótesis 1, 2 y 3, se ha probado la conclusión d. ◀

Si una hipótesis no es una cláusula, debe sustituirse por una expresión equivalente que sea una cláusula o el *y* de cláusulas. Por ejemplo, suponga que una de las hipótesis es $\neg(a \vee b)$. Como la negación se aplica a más de un término, se usa la primera ley de De Morgan (vea el ejemplo 1.2.11)

$$\neg(a \vee b) \equiv \neg a\, \neg b, \qquad \neg(ab) \equiv \neg a \vee \neg b \tag{1.6.3}$$

para obtener una expresión equivalente con la negación aplicada sólo a variables:

$$\neg(a \vee b) \equiv \neg a\, \neg b.$$

Después se sustituye la hipótesis original $\neg(a \vee b)$ por las dos hipótesis $\neg a$ y $\neg b$. Esta sustitución se justifica si se recuerda que las hipótesis individuales h_1 y h_2 son equivalentes a $h_1 h_2$ (vea la definición 1.5.20 y el análisis que le precede). El uso repetido de las leyes de De Morgan dará como resultado cada negación aplicada sólo a una variable.

Una expresión que consiste en términos separados por *o*, donde cada término consiste en el *y* de *varias* variables, puede sustituirse por una expresión equivalente que consiste en el *y* de varias cláusulas usando la equivalencia

$$a \vee bc \equiv (a \vee b)(a \vee c). \tag{1.6.4}$$

En este caso, podemos sustituir la hipótesis individual $a \vee bc$ por las dos hipótesis $a \vee b$ y $a \vee c$. Usando las leyes de De Morgan (1.6.3) y después (1.6.4), podemos obtener hipótesis equivalentes, cada una de las cuales es una cláusula.

Ejemplo 1.6.6 ▶

Se demostrará lo siguiente usando resolución:

1. $a \vee \neg bc$
2. $\underline{\neg(a \vee d)}$
$\therefore \neg b$

Se usa (1.6.4) para reemplazar la hipótesis 1 con las dos hipótesis

$$a \vee \neg b$$
$$a \vee c.$$

Se usa la primera ley de De Morgan (1.6.3) para sustituir la hipótesis 2 con las dos hipótesis

$$\neg a$$
$$\neg d.$$

el argumento se convierte en

1. $a \vee \neg b$
2. $a \vee c$
3. $\neg a$
4. $\neg d$

$\therefore \neg b$

Aplicando (1.6.1) a las expresiones 1 y 3, de inmediato se deriva la conclusión

$$\neg b.$$

◀

En los sistemas de razonamiento automatizado, la prueba por resolución se combina con la prueba por contradicción. Escribimos la conclusión negada como cláusulas y agrega las cláusulas a la hipótesis. Después aplicamos repetidamente (1.6.1) hasta obtener la contradicción.

Ejemplo 1.6.7 ▶

Probamos de nuevo el ejemplo 1.6.4 combinando la resolución con la prueba por contradicción.

Primero se niega la conclusión y se usa la primera ley de De Morgan (1.6.3) para obtener

$$\neg(b \vee d) \equiv \neg b \neg d.$$

Después se agregan las cláusulas $\neg b$ y $\neg d$ a las hipótesis para obtener

1. $a \vee b$
2. $\neg a \vee c$
3. $\neg c \vee d$
4. $\neg b$
5. $\neg d$

Aplicando (1.6.1) a las expresiones 1 y 2, se deriva

6. $b \vee c.$

Aplicando (1.6.1) a las expresiones 3 y 6, se deriva

7. $b \vee d.$

Aplicando (1.6.1) a las expresiones 4 y 7, se deriva

8. $d.$

Ahora se combinan 5 y 8 para dar una contradicción, y la prueba queda completa. ◀

Es posible demostrar que la resolución es *correcta* y la *refutación* queda *completa*. "La resolución es correcta" significa que si la resolución deriva una contradicción de un conjunto de cláusulas, las cláusulas son incongruentes (es decir, el conjunto de cláusulas no es todo cier-

to). "La resolución es la refutación completa" significa que la resolución será capaz de derivar una contradicción a partir del conjunto de cláusulas incongruentes. Así, si una conclusión se obtiene de un conjunto de hipótesis, la resolución podrá derivar una contradicción a partir de las hipótesis y una negación de la conclusión. Por desgracia, la resolución no dice qué cláusulas combinar para deducir la contradicción. Un reto clave para automatizar un sistema de razonamiento es ayudar a guiar la búsqueda de cláusulas a combinar. Las referencias acerca de la resolución y el razonamiento automatizado son [Gallier; Genesereth; y Wos].

Sugerencias para resolver problemas

Para construir una prueba por resolución, primero se sustituye cualquiera de las hipótesis o la conclusión que no sea una cláusula por una o más cláusulas. Después se reemplazan los pares de hipótesis de la forma $p \vee q$ y $\neg p \vee r$ con $q \vee r$ hasta llegar a la conclusión. Recuerde que es posible combinar la resolución con la prueba por contradicción.

Sección de ejercicios de repaso

1. ¿Qué regla de lógica usa la prueba por resolución?

2. ¿Qué es una cláusula?

3. Explique cómo procede una prueba por resolución.

Ejercicios

1. Escriba la tabla de verdad que demuestra (1.6.1).

Use la resolución para derivar cada combinación en los ejercicios 2 al 6. Sugerencia: En los ejercicios 5 y 6, sustituya $\rightarrow$ y $\leftrightarrow$ con expresiones lógicamente equivalentes que usan o e y.

2.
$\neg p \vee q \vee r$
$\neg q$
$\neg r$
$\therefore \neg p$

3.
$\neg p \vee r$
$\neg r \vee q$
p
$\therefore q$

4.
$\neg p \vee t$
$\neg q \vee s$
$\neg r \vee st$
$p \vee q \vee r \vee u$
$\therefore s \vee t \vee u$

5.
$p \rightarrow q$
$p \vee q$
$\therefore q$

6.
$p \leftrightarrow r$
r
$\therefore p$

7. Use resolución y prueba por contradicción para demostrar de nuevo los ejercicios 2 al 6.

8. Use resolución y prueba por contradicción para demostrar de nuevo el ejemplo 1.6.6.

1.7 ➜ Inducción matemática

WWW

Suponga que una secuencia de bloques numerados 1, 2, . . . están en una mesa (infinitamente) larga (vea la figura 1.7.1) y que algunos bloques están marcados con una "X". (Todos los bloques visibles en la figura 1.7.1 están marcados). Suponga que

El primer bloque está marcado. **(1.7.1)**

Para toda n, si el bloque n está marcado, entonces el bloque $n + 1$ también está marcado. **(1.7.2)**

Se asegura que (1.7.1) y (1.7.2) implican que todo bloque está marcado.

Se examinan los bloques uno por uno. La afirmación (1.7.1) establece, de modo explícito, que el bloque 1 está marcado. Considere el bloque 2. Como el bloque 1 está marcado, por (1.7.2) (con $n = 1$), el bloque 2 también está marcado. Considere el bloque 3. Como el bloque 2 está marcado, por (1.7.2) (con $n = 2$), el bloque 3 también está marcado. Continuando de esta manera, se demuestra que todo bloque está marcado. Por ejemplo, suponga que se ha verificado que los bloques 1 al 5 están marcados, como se indica en la figura 1.7.1. Para probar que el bloque 6 (que no se ilustra en la figura 1.7.1) está marcado, se observa que como el bloque 5 está marcado, por (1.7.1) (con $n = 5$), el bloque 6 también está marcado.

El ejemplo anterior ilustra el **principio de inducción matemática.** Para mostrar cómo se utiliza la inducción matemática en una forma más profunda, sea S_n la suma de los primeros n enteros positivos:

$$S_n = 1 + 2 + \ldots + n \qquad \textbf{(1.7.3)}$$

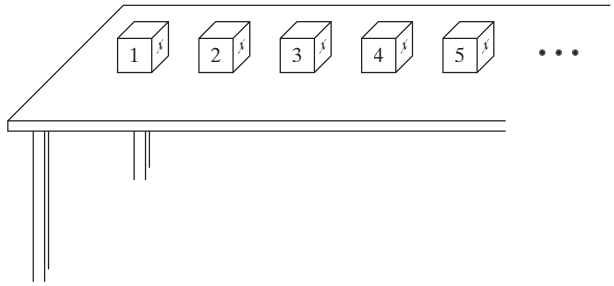


Figura 1.7.1 Bloques numerados en una mesa.

Suponga que alguien asegura que

$$S_n = \frac{n(n+1)}{2} \quad \text{para toda } n \geq 1. \tag{1.7.4}$$

En realidad ésta es una secuencia de afirmaciones, a saber,

$$S_1 = \frac{1(2)}{2} = 1$$
$$S_2 = \frac{2(3)}{2} = 3$$
$$S_3 = \frac{3(4)}{2} = 6$$
$$\vdots$$

$$\begin{array}{rl} S_1 = \dfrac{1(2)}{2} & \times \\ S_2 = \dfrac{2(3)}{2} & \times \\ \vdots & \\ S_{n-1} = \dfrac{(n-1)n}{2} & \times \\ S_n = \dfrac{n(n+1)}{2} & \times \\ S_{n+1} = \dfrac{(n+1)(n+2)}{2} & ? \\ \vdots & \end{array}$$

Figura 1.7.2 Una secuencia de afirmaciones. Las afirmaciones verdaderas están marcadas con ×.

Suponga que cada ecuación verdadera tiene una "×" colocada a su lado (vea la figura 1.7.2). Como la primera ecuación es verdadera, está marcada. Ahora suponga que podemos demostrar que para toda n, si la ecuación n está marcada, entonces la ecuación $n + 1$ también está marcada. Entonces, igual que en el ejemplo de los bloques, todas la ecuaciones están marcadas; es decir, todas las ecuaciones son verdaderas y la fórmula (1.7.4) queda verificada.

Debe demostrarse que para toda n, si la ecuación n es verdadera, entonces la ecuación $n + 1$ también es verdadera. La ecuación n es

$$S_n = \frac{n(n+1)}{2}. \tag{1.7.5}$$

Suponiendo que esta ecuación es verdadera, debe probarse que la ecuación $n + 1$

$$S_{n+1} = \frac{(n+1)(n+2)}{2}$$

es verdadera. Según la definición (1.7.3),

$$S_{n+1} = 1 + 2 + \cdots + n + (n+1).$$

Observe que S_n está contenido en S_{n+1}, en el sentido de que

$$S_{n+1} = 1 + 2 + \cdots + n + (n+1) = S_n + (n+1). \tag{1.7.6}$$

Por (1.7.5) y (1.7.6), tenemos

$$S_{n+1} = S_n + (n+1) = \frac{n(n+1)}{2} + (n+1).$$

Como

$$\begin{aligned} \frac{n(n+1)}{2} + (n+1) &= \frac{n(n+1)}{2} + \frac{2(n+1)}{2} \\ &= \frac{n(n+1) + 2(n+1)}{2} = \frac{(n+1)(n+2)}{2}, \end{aligned}$$

tenemos

$$S_{n+1} = \frac{(n+1)(n+2)}{2}.$$

Por lo tanto, suponiendo que la ecuación n es cierta, se ha probado que la ecuación $n+1$ es cierta. Se concluye que todas las ecuaciones son ciertas.

La prueba usando inducción matemática consiste en dos pasos. Primero, se verifica que la afirmación correspondiente a $n = 1$ sea verdadera. Segundo, se *supone* que la afirmación n es verdadera y después se *prueba* que la afirmación $n+1$ también es verdadera. Al probar la afirmación $n+1$, se puede usar la afirmación n; sin duda, el truco al desarrollar una prueba usando inducción matemática es relacionar la afirmación n con la afirmación $n+1$.

Ahora se establece formalmente el principio de inducción matemática.

Principio de inducción matemática

Suponga que se tiene una función proposicional S(n) cuyo dominio de discurso es el conjunto de enteros positivos. Suponga que

S(1) *es verdadera;* **(1.7.7)**

para toda $n \geq 1$, *si S(n) es verdadera, entonces S(n + 1) es verdadera.* **(1.7.8)**

Entonces S(n) es verdadera para todo entero positivo n.

La condición (1.7.7) en ocasiones recibe el nombre de **paso base** y la condición (1.7.8) el de **paso inductivo**. En adelante, por "inducción" entenderemos "inducción matemática".

Después de definir el factorial de n, se ilustra el principio de inducción matemática con otro ejemplo.

Definición 1.7.1 ▶

El *factorial de n* (o *n factorial*) se define como

$$n! = \begin{cases} 1 & \text{si } n = 0 \\ n(n-1)(n-2)\cdots 2 \cdot 1 & \text{si } n \geq 1. \end{cases}$$

Esto es, si $n \geq 1$, $n!$ es igual al producto de todos los enteros entre 1 y n inclusive. Como caso especial, 0! se define como 1. ◀

Ejemplo 1.7.2 ▶

$$0! = 1! = 1, \quad 3! = 3 \cdot 2 \cdot 1 = 6, \quad 6! = 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 720$$ ◀

Ejemplo 1.7.3 ▶

Utilice inducción matemática para demostrar que

$$n! \geq 2^{n-1} \text{ para toda } n \geq 1. \quad \textbf{(1.7.9)}$$

Paso base

[Condición (1.7.7)] Debe demostrarse que (1.7.9) es verdadera si $n = 1$. Esto es sencillo ya que $1! = 1 \geq 1 = 2^{1-1}$.

Paso inductivo

[Condición (1.7.8)] Suponemos que la desigualdad es cierta para n; es decir, suponemos que

$$n! \geq 2^{n-1} \quad \textbf{(1.7.10)}$$

es cierta. Debemos probar que la desigualdad es cierta para $n+1$; es decir, debemos probar que

$$(n+1)\,! \geq 2^n \quad \textbf{(1.7.11)}$$

es cierta. Podemos relacionar (1.7.10) y (1.7.11) si observamos que

$$(n+1)! = (n+1)(n!).$$

Ahora bien

$$\begin{aligned}(n+1)! &= (n+1)(n!) \\ &\geq (n+1)2^{n-1} && \text{por (1.7.10)} \\ &\geq 2\cdot 2^{n-1} && \text{ya que } n+1 \geq 2 \\ &= 2^n.\end{aligned}$$

Por lo tanto, (1.7.11) es cierta. Esto completa el paso inductivo.

Como se verificaron el *paso base* y el paso inductivo, el principio de inducción matemática dice que (1.7.9) es verdadera para todo entero positivo n. ◀

Si se desea verificar que las afirmaciones

$$S(n_0),\ S(n_0+1),\ \ldots,$$

donde $n_0 \neq 1$, son verdaderas, debe cambiarse el paso base a

$$S(n_0) \textit{ es verdadera.}$$

Entonces, el paso inductivo se convierte en

$$\textit{para toda } n \geq n_0\textit{, si } S(n) \textit{ es verdadera, entonces } S(n+1) \textit{ es verdadera.}$$

Ejemplo 1.7.4 ▶

Suma geométrica

Use la inducción para demostrar que si $r \neq 1$,

$$a + ar^1 + ar^2 + \cdots + ar^n = \frac{a(r^{n+1}-1)}{r-1} \qquad \textbf{(1.7.12)}$$

para toda $n \geq 0$.

La suma en el lado izquierdo se llama **suma geométrica**. En la suma geométrica en la que $a \neq 0$ y $r \neq 0$, la razón de términos adyacentes $[(ar^{i+1})/(ar^i) = r]$ es constante.

Paso base

El paso base, que en este caso se obtiene haciendo $n = 0$, es

$$a = \frac{a(r^1-1)}{r-1},$$

que es verdadera.

Paso inductivo

Suponga que la afirmación (1.7.12) es verdadera para n. Ahora

$$\begin{aligned}a + ar^1 + ar^2 + \cdots + ar^n + ar^{n+1} &= \frac{a(r^{n+1}-1)}{r-1} + ar^{n+1} \\ &= \frac{a(r^{n+1}-1)}{r-1} + \frac{ar^{n+1}(r-1)}{r-1} \\ &= \frac{a(r^{n+2}-1)}{r-1}.\end{aligned}$$

Como se verificaron el paso base y el paso inductivo, el principio de inducción matemática dice que (1.7.12) es verdadera para toda $n \geq 0$. ◀

Como ejemplo del uso de la suma geométrica, si tomamos $a = 1$ y $r = 2$ en (1.7.12), obtenemos la fórmula

$$1 + 2 + 2^2 + 2^3 + \cdots + 2^n = \frac{2^{n+1}-1}{2-1} = 2^{n+1} - 1.$$

Sin duda el lector ha observado que para probar las fórmulas anteriores, debe contar con las fórmulas correctas desde el principio. Una pregunta razonable es: ¿cómo se obtienen las fórmulas? Existen muchas respuestas a esta pregunta. Una técnica para derivar una fórmula consiste en experimentar con pequeños valores e intentar descubrir un patrón. (Otra técnica se ve en los ejercicios 64 al 67.) Por ejemplo, considere la suma $1+3+ \ldots +(2n–1)$. La siguiente tabla da los valores de esta suma para $n = 1, 2, 3, 4$.

n	$1 + 3 + \ldots + (2n-1)$
1	1
2	4
3	9
4	16

Como la segunda columna consiste en los cuadrados, se conjetura que

$$1 + 3 + \ldots + (2n-1) = n^2 \text{ para todo entero positivo } n.$$

La conjetura es correcta y la fórmula se puede probar por inducción matemática (vea el ejercicio 1).

En este momento, quizá el lector quiera leer el rincón de solución de problemas que sigue a esta sección. Este cuadro proporciona una exposición amplia y detallada de cómo desarrollar las demostraciones por inducción matemática.

Los ejemplos finales muestran que la inducción no está limitada a probar fórmulas para sumas y verificar desigualdades.

Ejemplo 1.7.5 ▶

Utilice la inducción para demostrar que $5^n – 1$ es divisible entre 4 para toda $n \geq 1$.

Paso base

Si $n = 1$, $5^n – 1 = 5^1 – 1 = 4$, que es divisible entre 4.

Paso inductivo

Se supone que $5^n – 1$ es divisible entre 4. Debemos demostrar que $5^{n+1} – 1$ es divisible entre 4. Se usa el hecho de que si p y q son cada uno divisibles entre k, entonces $p+q$ también es divisible entre k. En este caso, $k = 4$. Se deja la prueba de este hecho como ejercicio (vea el ejercicio 68).

El caso $(n + 1)$ se relaciona con el caso n escribiendo

$$5^{n+1} – 1 = 5^n — 1 + \text{por determinar.}$$

Ahora, por la suposición de inducción, $5^n – 1$ es divisible entre 4. Si "por determinar" también es divisible entre 4, entonces la suma anterior, que es igual a $5^{n+1} – 1$, también será divisible entre 4, y el paso inductivo quedará completo. Debe encontrarse el valor de "por determinar".

Ahora

$$5^{n+1} - 1 = 5 \cdot 5^n - 1 = 4 \cdot 5^n + 1 \cdot 5^n - 1.$$

Así, "por determinar" es $4 \cdot 5^n$, que es divisible entre 4. De manera formal, el paso inductivo se escribe como sigue.

Por la suposición de inducción, $5^n – 1$ es divisible entre 4 y, como $4 \cdot 5^n$ es divisible entre 4, la suma

$$(5^n - 1) + 4 \cdot 5^n = 5^{n+1} - 1$$

es divisible entre 4.

Como se verificaron el paso base y el paso inductivo, el principio de inducción matemática dice que $5^n – 1$ es divisible entre 4 para toda $n \geq 1$. ◀

Ejemplo 1.7.6 ▶

Un problema de enlosado

Un *tromino derecho*, en adelante llamado sólo *tromino*, es un objeto formado por tres cuadros como el que se observa en la figura 1.7.3. Un tromino es un tipo de poliomino. Desde que Solomon W. Golomb introdujo los poliominos en 1954 (vea [Golomb, 1954]), han sido un tema favorito de matemáticas recreativas. Un *poliomino de orden s* consiste en *s* cuadros unidos en las orillas. Un tromino es un poliomino de orden 3. Tres cuadros en una fila forman sólo otro tipo de poliomino de orden 3. (No se ha encontrado una fórmula sencilla para el número de poliominos de orden *s*). Se han diseñado numerosos problemas usando poliominos (vea [Martin]).

WWW

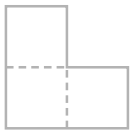


Figura 1.7.3 Un tromino.

Presentamos la prueba inductiva de Golomb (vea [Golomb, 1954]) que indica que si se quita un cuadro de un tablero de $n \times n$, donde n es una potencia de 2, se puede enlosar con trominos el resto de los cuadros (vea la figura 1.7.4). *Enlosar* una figura con trominos significa un cubrimiento exacto de la figura con los trominos sin que ninguno de los trominos se traslape con otro o se extienda fuera de la figura. Un tablero al que le falta un cuadro se llama *tablero deficiente*.

Ahora se usará inducción matemática sobre k para probar que es posible enlosar un tablero deficiente de $2^k \times 2^k$ con trominos.

Paso base

Si $k = 1$, el tablero deficiente de 2×2 es en sí un tromino y por lo tanto se puede enlosar con un tromino.

Paso inductivo

Suponga que se puede enlosar un tablero deficiente de $2^k \times 2^k$. Se debe probar que es posible enlosar un tablero deficiente de $2^{k+1} \times 2^{k+1}$.

Considere un tablero deficiente de $2^{k+1} \times 2^{k+1}$. Divida el tablero en cuatro tableros de $2^k \times 2^k$, como se indica en la figura 1.7.5. Rote el tablero de manera que el cuadro que falta esté en el cuadrante superior izquierdo. Por la suposición inductiva, este tablero arriba a la izquierda de $2^k \times 2^k$ se puede enlosar. Coloque un tromino T en el centro como se ve en la figura 1.7.5, de manera que cada cuadro de T esté en cada uno de otros los cuadrantes. Si se consideran los cuadros cubiertos por T como faltantes, cada cuadrante es un tablero deficiente de $2^k \times 2^k$. De nuevo, por la suposición inductiva, estos tableros se pueden enlosar. Ahora tenemos un enlosado del tablero $2^{k+1} \times 2^{k+1}$. Por el principio de inducción matemática, se concluye que cualquier tablero deficiente de $2^k \times 2^k$ se puede enlosar con trominos, $k = 1, 2, \ldots$.

Si se puede enlosar un tablero deficiente de $n \times n$, donde n no necesariamente es una potencia de 2, entonces el número de cuadros, $n^2 - 1$, debe ser divisible entre 3. [Chu] mostró que el inverso es cierto, excepto cuando n es 5. Con mayor precisión, si $n \neq 5$, cualquier tablero deficiente de $n \times n$ se puede enlosar con trominos si y sólo si 3 es divisor de $n^2 - 1$ (vea los ejercicios 23 y 24, sección 1.8). [Algunos tableros deficientes de 5×5 se pueden enlosar y otros no (vea los ejercicios 29 y 30)].

Es posible modelar algunos problemas de la vida real como problemas de enlosado. Un ejemplo es el *problema de la distribución de VLSI*, el problema de empacar muchos componentes en un chip de computadora (vea [Wong]). (VSLI son las siglas de *very large*

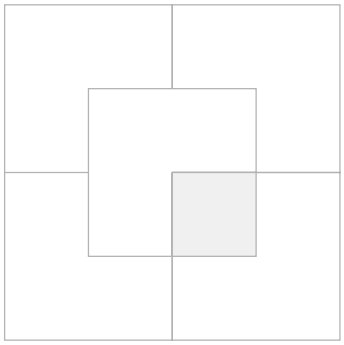


Figura 1.7.4 Enlosado de un tablero deficiente de 4×4 con trominos.

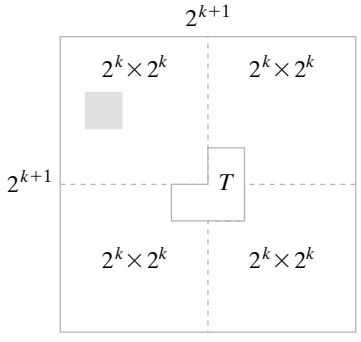


Figura 1.7.5 Empleo de la inducción matemática para enlosar un tablero deficiente de $2^{k+1} \times 2^{k+1}$ con trominos.

scale integration o integración a gran escala). El problema es enlosar un rectángulo de área mínima con los componentes deseados. Algunas veces, los componentes se modelan como rectángulos y figuras en forma de L similares a los trominos. En la práctica, se imponen otras restricciones, como la proximidad de diferentes componentes que deben interconectarse y las restricciones sobre la relación entre la base y la altura del rectángulo resultante. ◀

Un **ciclo invariante** es una afirmación acerca de variables de programa que es verdadera justo antes de que inicie la ejecución del ciclo y también después de cada iteración del ciclo. En particular, un ciclo invariante es verdadero cuando termina el ciclo, punto en el que el invariante nos dice algo acerca del estado de las variables. De manera ideal, esta afirmación dice que el ciclo produce el resultado esperado, es decir, que el ciclo es correcto. Por ejemplo, un ciclo invariante para un ciclo "while" (o mientras)

```
while (condición)
   //cuerpo del ciclo
```

es verdadero justo antes de evaluar la *condición* la primera vez, y también es verdadero cada vez que se ejecuta el cuerpo del ciclo.

Mediante la inducción matemática es posible probar que un invariante tiene el comportamiento deseado. El paso base demuestra que el invariante es verdadero antes de que la condición que controla el ciclo se pruebe la primera vez. El paso inductivo supone que el invariante es verdadero y después demuestra que si la condición que controla el ciclo es verdadera (de manera que el cuerpo del ciclo se ejecuta de nuevo), el invariante es verdadero después de ejecutar el cuerpo del ciclo. Como un ciclo itera un número finito de veces, la forma de la inducción matemática usada aquí demuestra que una secuencia *finita* de afirmaciones es verdadera, en lugar de una secuencia infinita de afirmaciones como en el ejemplo anterior. Sea finita o infinita la secuencia de afirmaciones, los pasos necesarios para la demostración por inducción matemática son los mismos. Se ilustra un ciclo invariante con un ejemplo.

Ejemplo 1.7.7 ▶

Se usa un ciclo invariante para demostrar que cuando el seudocódigo

```
i = 1
fact = 1
while (i < n){
   i = i + 1
   fact = fact * 1
}
```

termina, *fact* es igual a $n!$.

Se demostrará que $fact = i!$ es un invariante para el ciclo "while". Justo antes de que el ciclo "while" comience a ejecutarse, $i = 1$ y $fact = 1$, o sea, $fact = 1!$. Esto prueba el paso base.

Suponga que $fact = i!$. Si $i < n$ es verdadera (y el cuerpo del ciclo se ejecuta otra vez), i se convierte en $i + 1$ y *fact* se convierte en

$$fact * (i + 1) = i! * (i + 1) = (i + 1)!.$$

Se probó el paso inductivo. Por lo tanto, $fact = i!$ es un invariante para el ciclo "while".

El ciclo "while" termina cuando $i = n$. Como $fact = i!$ es un invariante, en este punto, $fact = n!$. ◀

Sugerencias para resolver problemas

Para demostrar que

$$a_1 + a_2 + \cdots + a_n = F(n),$$

donde $F(n)$ es la fórmula para la suma, primero se verifica la ecuación para $n = 1$

$$a_1 = F(1)$$

(Paso base). Éste suele ser directo.

Ahora suponga que la afirmación es verdadera para n; es decir, suponga que

$$a_1 + a_2 + \ldots + a_n = F(n).$$

Sume a_{n+1} a ambos lados para obtener

$$a_1 + a_2 + \cdots + a_n + a_{n+1} = F(n) + a_{n+1}.$$

Por último, demuestre que

$$F(n) + a_{n+1} = F(n+1).$$

Para verificar la ecuación anterior, use álgebra para manejar el lado izquierdo de la ecuación $[F(n) + a_{n+1}]$ hasta obtener $F(n+1)$. *Observe $F(n+1)$ para que sepa hacia dónde va.* (¡Es algo como ver la respuesta al final del libro!) Usted demostró que

$$a_1 + a_2 + \cdots + a_{n+1} = F(n+1),$$

que es el paso inductivo. La demostración queda completa.

Probar una desigualdad se maneja de manera similar. La diferencia es que en lugar de obtener la igualdad $[F(n) + a_{n+1} = F(n+1)$ del análisis anterior], se obtiene una *desigualdad*.

En general, la clave para idear una demostración por inducción es encontrar el caso n "dentro" del caso $n + 1$. Revise el problema del enlosado (ejemplo 1.7.6), que proporciona un ejemplo notable del caso n "dentro" del caso $n + 1$.

Sección de ejercicios de repaso

1. Establezca el principio de inducción matemática.

2. Explique cómo procede una demostración por inducción matemática.

3. Dé una fórmula para la suma $1 + 2 + \ldots + n$.

4. ¿Qué es una suma geométrica? Dé una fórmula para calcularla.

Ejercicios

En los ejercicios 1 al 11, mediante inducción matemática, verifique que cada ecuación es verdadera para todo entero positivo n.

1. $1 + 3 + 5 + \cdots + (2n-1) = n^2$

2. $1 \cdot 2 + 2 \cdot 3 + 3 \cdot 4 + \cdots + n(n+1) = \dfrac{n(n+1)(n+2)}{3}$

3. $1(1!) + 2(2!) + \cdots + n(n!) = (n+1)! - 1$

4. $1^2 + 2^2 + 3^2 + \cdots + n^2 = \dfrac{n(n+1)(2n+1)}{6}$

5. $1^2 - 2^2 + 3^2 - \cdots + (-1)^{n+1} n^2 = \dfrac{(-1)^{n+1} n(n+1)}{2}$

6. $1^3 + 2^3 + 3^3 + \cdots + n^3 = \left[\dfrac{n(n+1)}{2}\right]^2$

7. $\dfrac{1}{1 \cdot 3} + \dfrac{1}{3 \cdot 5} + \dfrac{1}{5 \cdot 7} + \cdots + \dfrac{1}{(2n-1)(2n+1)} = \dfrac{n}{2n+1}$

8. $\dfrac{1}{2 \cdot 4} + \dfrac{1 \cdot 3}{2 \cdot 4 \cdot 6} + \dfrac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6 \cdot 8} + \cdots + \dfrac{1 \cdot 3 \cdot 5 \cdots (2n-1)}{2 \cdot 4 \cdot 6 \cdots (2n+2)} = \dfrac{1}{2} - \dfrac{1 \cdot 3 \cdot 5 \cdots (2n+1)}{2 \cdot 4 \cdot 6 \cdots (2n+2)}$

9. $\dfrac{1}{2^2-1} + \dfrac{1}{3^2-1} + \cdots + \dfrac{1}{(n+1)^2-1} = \dfrac{3}{4} - \dfrac{1}{2(n+1)} - \dfrac{1}{2(n+2)}$

★10. $\cos x + \cos 2x + \cdots + \cos nx = \dfrac{\cos[(x/2)(n+1)]\operatorname{sen}(nx/2)}{\operatorname{sen}(x/2)}$ siempre que $\operatorname{sen}(x/2) \neq 0$.

★11. $1\operatorname{sen} x + 2\operatorname{sen} 2x + \cdots + n\operatorname{sen} nx = \dfrac{\operatorname{sen}[(n+1)x]}{4\operatorname{sen}^2(x/2)} - \dfrac{(n+1)\cos[(2n+1)x/2]}{2\operatorname{sen}(x/2)}$ siempre que $\operatorname{sen}(x/2) \neq 0$.

En los ejercicios 12 al 17, use inducción para verificar la desigualdad.

12. $\dfrac{1}{2n} \leq \dfrac{1 \cdot 3 \cdot 5 \cdots (2n-1)}{2 \cdot 4 \cdot 6 \cdots (2n)}, \; n = 1, 2, \ldots$

★13. $\dfrac{1 \cdot 3 \cdot 5 \cdots (2n-1)}{2 \cdot 4 \cdot 6 \cdots (2n)} \leq \dfrac{1}{\sqrt{n+1}}, \; n = 1, 2, \ldots$

14. $2n + 1 \leq 2^n, \; n = 3, 4, \ldots$

★15. $2^n \geq n^2, \; n = 4, 5, \ldots$

★**16.** $(a_1 a_2 \cdots a_{2^n})^{1/2^n} \le \dfrac{a_1 + a_2 + \cdots + a_{2^n}}{2^n}$, $n = 1, 2, \ldots,$ y las a_i son números positivos.

17. $(1 + x)^n \ge 1 + nx$, para $x \ge -1$ y $n \ge 1$

18. Use la suma geométrica para probar que

$$r^0 + r^1 + \cdots + r^n < \frac{1}{1 - r}$$

para toda $n \ge 0$ y $0 < r < 1$.

★**19.** Demuestre que

$$1 \cdot r^1 + 2 \cdot r^2 + \cdots + nr^n < \frac{r}{(1 - r)^2}$$

para toda $n \ge 1$ y $0 < r < 1$. *Sugerencia:* Use el resultado del ejercicio anterior, compare la suma de los términos en

$$\begin{array}{llllll} r & r^2 & r^3 & r^4 & \cdots & r^n \\ r^2 & r^3 & r^4 & \cdots & r^n & \\ r^3 & r^4 & \cdots & r^n & & \\ r^4 & \cdots & & & & \\ \vdots & \vdots & & & & \\ & r^n & & & & \\ r^n & & & & & \end{array}$$

en la diagonal ($\swarrow$) con la suma de los términos por columna.

20. Pruebe que

$$\frac{1}{2^1} + \frac{2}{2^2} + \frac{3}{2^3} + \cdots + \frac{n}{2^n} < 2$$

para toda $n \ge 1$.

En los ejercicios 21 al 24, use la inducción para probar la afirmación.

21. $7^n - 1$ es divisible entre 6, para toda $n \ge 1$.

22. $11^n - 6$ es divisible entre 5, para toda $n \ge 1$.

23. $6 \cdot 7^n - 2 \cdot 3^n$ es divisible entre 4, para toda $n \ge 1$.

★***24.*** $3^n + 7^n - 2$ es divisible entre 8, para toda $n \ge 1$.

25. Experimentando con valores pequeños de n, adivine una fórmula para la suma

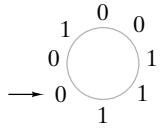
$$\frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \cdots + \frac{1}{n(n + 1)};$$

después use inducción para verificar su fórmula.

26. Use inducción para demostrar que n líneas rectas en el plano lo dividen en $(n^2 + n + 2)/2$ regiones. Suponga que no hay dos líneas paralelas y que no hay tres líneas con un punto en común.

27. Demuestre que las regiones del ejercicio anterior pueden colorearse de rojo y verde de modo que no haya dos regiones que compartan una orilla que sean del mismo color.

28. Dados n ceros y n unos distribuidos de cualquier manera alrededor de un círculo (vea la figura siguiente), demuestre, por inducción sobre n, que es posible comenzar en algún número y proceder en el sentido de las manecillas del reloj alrededor del círculo hasta la posición original de inicio, de manera que en cualquier punto durante el ciclo se hayan visto al menos la misma cantidad de ceros que de unos. En la siguiente figura, un punto de inicio posible está marcado con una flecha.



29. Dé un enlosado de un tablero de 5×5 con trominos en los que falte el cuadro superior izquierdo.

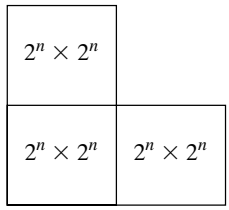
30. Demuestre un tablero deficiente de 5×5 que es imposible enlosar con trominos. Explique por qué no se puede enlosar.

31. Demuestre que cualquier tablero de $(2i) \times (3j)$, donde i y j son enteros positivos, sin cuadro faltante, se puede enlosar con trominos.

★***32.*** Demuestre que cualquier tablero deficiente de 7×7 se puede enlosar con trominos.

33. Demuestre que cualquier tablero deficiente de 11×11 se puede enlosar con trominos. *Sugerencia:* Subdivida el tablero en tableros que se traslapan de 7×7 y 5×5 y dos tableros de 6×4. Después remítase a los ejercicios 29, 31 y 32.

34. Este ejercicio y el siguiente se deben a Anthony Quass. Una *forma-L* de $2^n \times 2^n$, $n \ge 0$ es una figura de la forma



sin cuadros faltantes. Demuestre que cualquier forma-L de $2^n \times 2^n$ se puede enlosar con trominos.

35. Use el ejercicio anterior para dar una prueba diferente de que cualquier tablero deficiente de $2^n \times 2^n$ se puede enlosar con trominos.

Un tromino recto es *un objeto hecho de tres cuadros en fila:*

36. ¿Qué tableros deficientes de 4×4 se pueden enlosar con trominos rectos? *Sugerencia:* Numere los cuadros del tablero de 4×4, de izquierda a derecha y de arriba abajo: 1, 2, 3, 1, 2, 3, etcétera. Observe que si existe un enlosado, cada tromino recto cubre exactamente un 2 y exactamente un 3.

37. ¿Qué tableros deficientes de 5×5 se pueden enlosar con trominos rectos?

38. ¿Qué tableros deficientes de 8×8 se pueden enlosar con trominos rectos?

39. Use un ciclo invariante para probar que cuando el seudocódigo

```
i=1
pow=1
while(i ≤ n){
    pow = pow * a
    i=i+1
}
```

termina, *pow* es igual a a^n.

40. Pruebe que, después de que termina el siguiente seudocódigo, $a[h] = val$; para toda p, $i \le p < h$, $a[p] < val$; y para toda p, $h < p \le j$, $a[p] \ge val$. En particular, *val* está en la posición en el arreglo $a[i], \ldots, a[j]$ donde estaría si el arreglo fuera ordenado.

```
val = a[i]
h = i
for k = i + 1 to j
    if (a[k] < val) {
```

```
        h = h + 1
        swap(a[h], a[k])
    }
swap(a[i], a[h])
```

Sugerencia: Use el ciclo invariante: para toda p, $i < p \le h$, $a[p] < val$; y para toda p, $h < p < k$, $a[p] \ge val$. (Un dibujo ayuda).

Esta técnica se llama *particionar*. Esta versión particular se debe a Nico Lomuto. Las particiones se emplean para encontrar el k-ésimo, elemento más pequeño en cualquier arreglo y para construir un algoritmo para ordenar llamado *quicksort*.

Un septomino en 3D es un *cubo de tres dimensiones de* $2 \times 2 \times 2$ *sin una esquina de* $1 \times 1 \times 1$. Un cubo deficiente es *un cubo de* $k \times k \times k$ *sin un cubo de* $1 \times 1 \times 1$.

41. Pruebe que un cubo deficiente de $2^n \times 2^n \times 2^n$ se puede enlosar con septominos en 3D.

42. Demuestre que si un cubo deficiente de $k \times k \times k$ se puede enlosar con septominos en 3D, entonces 7 divide uno de los $k-1$, $k-2$, $k-4$.

43. Suponga que $S_n = (n + 2)(n - 1)$ se propone (incorrectamente) como fórmula para

$$2 + 4 + \ldots + 2n.$$

a) Demuestre que el paso inductivo se satisface pero que el paso base falla.

★ *b)* Si S'_n es una expresión arbitraria que satisface el paso inductivo, ¿qué forma debe tener S'_n?

★ **44.** ¿Qué está mal en el siguiente argumento, que se supone que prueba que cualesquiera dos enteros positivos son iguales?

Se usa inducción sobre n para "probar" que si a y b son enteros positivos y $n = \text{máx}\{a, b\}$, entonces $a = b$.

Paso base ($n = 1$). Si a y b son enteros positivos y $1 = \text{máx}\{a, b\}$, debe tenerse $a = b = 1$.

Paso inductivo Suponga que si a' y b' son enteros positivos y $n = \text{máx}\{a', b'\}$, entonces $a' = b'$. Suponga que a y b son enteros positivos y que $n + 1 = \text{máx}\{a, b\}$. Ahora $n = \text{máx}\{a–1, b–1\}$. Por la hipótesis inductiva, $a–1 = b–1$. Por lo tanto, $a = b$.

Como se verificaron el paso base y el paso inductivo, por el principio de inducción matemática, ¡cualesquiera dos enteros positivos son iguales!

45. ¿Qué está mal en la siguiente "prueba" de que

$$\frac{1}{2} + \frac{2}{3} + \cdots + \frac{n}{n+1} \ne \frac{n^2}{n+1}$$

para toda $n \ge 2$?

Suponga, a manera de contradicción, que

$$\frac{1}{2} + \frac{2}{3} + \cdots + \frac{n}{n+1} = \frac{n^2}{n+1}. \qquad \textbf{(1.7.13)}$$

Entonces también

$$\frac{1}{2} + \frac{2}{3} + \cdots + \frac{n}{n+1} + \frac{n+1}{n+2} = \frac{(n+1)^2}{n+2}.$$

Se podría probar la afirmación (1.7.13) por inducción. En particular, el paso inductivo daría

$$\left(\frac{1}{2} + \frac{2}{3} + \cdots + \frac{n}{n+1}\right) + \frac{n+1}{n+2} = \frac{n^2}{n+1} + \frac{n+1}{n+2}.$$

Por lo tanto,

$$\frac{n^2}{n+1} + \frac{n+1}{n+2} = \frac{(n+1)^2}{n+2}.$$

Al multiplicar cada lado de esta ecuación por $(n + 1)(n + 2)$ se tiene

$$n^2(n+2) + (n+1)^2 = (n+1)^3.$$

Esta última ecuación se rescribe como

$$n^3 + 2n^2 + n^2 + 2n + 1 = n^3 + 3n^2 + 3n + 1$$

o

$$n^3 + 3n^2 + 2n + 1 = n^3 + 3n^2 + 3n + 1,$$

lo cual es una contradicción. Por lo tanto,

$$\frac{1}{2} + \frac{2}{3} + \cdots + \frac{n}{n+1} \ne \frac{n^2}{n+1},$$

como se aseguró.

46. Use inducción matemática para probar que

$$\frac{1}{2} + \frac{2}{3} + \cdots + \frac{n}{n+1} < \frac{n^2}{n+1}$$

para toda $n \ge 2$. Esta desigualdad da una prueba correcta de la afirmación del ejercicio anterior.

En los ejercicios 47 al 51, suponga que $n > 1$ personas se colocan en un campo (plano euclidiano) de manera que cada una tiene un vecino más cercano único. Aún más, suponga que cada persona tiene un pastel que lanza a su vecino más cercano. Un sobreviviente es una persona a la que no le pega un pastel.

47. Proporcione un ejemplo para mostrar que si n es par, puede no haber un sobreviviente.

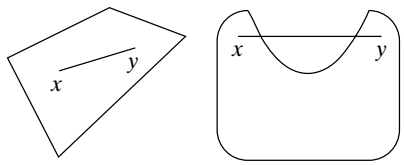
48. Dé un ejemplo para demostrar que puede haber más de un sobreviviente.

★ **49.** [Carmony] Use inducción sobre n para demostrar que si n es impar, siempre habrá al menos un sobreviviente.

50. Pruebe o desapruebe: Si n es impar, una de las dos personas más separadas es un sobreviviente.

51. Pruebe o desapruebe: Si n es impar, la persona que lanza un pastel la mayor distancia es un sobreviviente.

Los ejercicios 52 al 55 manejan conjuntos convexos planos. Un conjunto convexo plano, *en adelante abreviado "conjunto convexo", es un conjunto no vacío X en el plano que tiene la propiedad de que si x* y *y son cualesquiera dos puntos en X, el segmento de recta de x a y también está en X. Las siguientes figuras ilustran esto.*



conjunto convexo conjunto no convexo

52. Pruebe que si X y Y son conjuntos convexos y $X \cap Y$ (los puntos comunes a X y Y) es no vacío, $X \cap Y$ es un conjunto convexo.

★**53.** Suponga que X_1, X_2, X_3, X_4 son conjuntos convexos, cada tres de los cuales tienen un punto en común. Pruebe que los cuatro tienen un punto en común.

★**54.** *Pruebe el Teorema de Helly:* Suponga que $X_1, X_2, \ldots, X_n$, $n \ge 4$, son conjuntos convexos, cada tres de los cuales tienen un punto en común. Pruebe que los n conjuntos tienen un punto en común.

55. Suponga que $n \geq 3$ puntos en el plano tienen la propiedad de que cada tres de ellos están contenidos en un círculo de radio 1. Pruebe que existe un círculo de radio 1 que contiene a todos los puntos.

56. Si a y b son números reales con $a < b$, un *intervalo abierto* (a, b) es el conjunto de números reales x tales que $a < x < b$. Pruebe que si $I_1, \ldots, I_n$ es un conjunto de $n \geq 2$ intervalos abiertos tales que cada par tiene una intersección no vacía, entonces

$$I_1 \cap I_2 \cap \cdots \cap I_n$$

(los puntos comunes a $I_1, \ldots, I_n$) es no vacío.

Flavius Josephus era un soldado e historiador judío que vivió en el siglo I (vea [Graham, 1994; Schumer]). Era uno de los líderes de una revuelta judía contra Roma en el año 66. El siguiente año, se encontraba entre un grupo de soldados atrapados que decidieron suicidarse antes de que los capturaran. Una versión de la historia dice que, antes que ser capturados, formaron un círculo y procedieron a matar a cada tercera persona alrededor del círculo. Josephus, que tenía conocimientos de matemáticas discretas, se dio cuenta dónde debían pararse él y un amigo para evitar que los mataran.

Los ejercicios 57 al 63 se refieren a una variante del problema de Josephus en el que se elimina a una persona cada dos. Se supone que n personas se colocan en un círculo y se numeran 1, 2, . . . , n en el sentido de las manecillas del reloj. Se elimina 2, se elimina 4, etcétera, hasta que hay un sobreviviente, denotado por J(n).

57. Calcule $J(4)$.

58. Calcule $J(6)$.

59. Calcule $J(10)$.

60. Use la inducción para demostrar que $J(2^i) = 1$ para toda $i \geq 1$.

61. Con un valor de $n \geq 2$, sea 2^i la potencia más grande de 2 tal que $2^i \leq n$. (Ejemplos: Si $n = 10$, $i = 3$. Si $n = 16$, $i = 4$.) Sea $j = n - 2^i$. (Después de restar de n, 2^i, la potencia más grande de 2 menor o igual que n, j es lo que queda.) Usando el resultado del ejercicio 60 o de otra manera, pruebe que

$$J(n) = 2j + 1.$$

62. Use el resultado del ejercicio 61 para calcular $J(1000)$.

63. Use el resultado del ejercicio 61 para calcular J (100,000).

Si $a_1, a_2, \ldots$ es una secuencia, se define el operador diferencia Δ como

$$\Delta a_n = a_{n+1} - a_n.$$

La fórmula del ejercicio 64 se utiliza en ocasiones para encontrar *una fórmula para una suma en lugar de usar la inducción para* probar *una fórmula para la suma (vea los ejercicios 65 al 67).*

64. Suponga que $\Delta a_n = b_n$. Demuestre que

$$b_1 + b_2 + \cdots + b_n = a_{n+1} - a_1.$$

Esta fórmula es análoga a la fórmula de cálculo $\int_c^d f(x)\,dx = g(d) - g(c)$, donde $Dg = f$ (D es el operador derivada). En la fórmula de cálculo, la suma se sustituye por la integral y Δ se sustituye por la derivada.

65. Sea $a_n = n^2$, calcule Δa_n. Use el ejercicio 64 para encontrar una fórmula para

$$1 + 2 + 3 + \ldots + n.$$

66. Use el ejercicio 64 par encontrar una fórmula para

$$1(1!) + 2(2!) + \cdots + n(n!).$$

(Compare con el ejercicio 3).

67. Use el ejercicio 64 para encontrar una fórmula para

$$\frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \cdots + \frac{1}{n(n+1)}.$$

(Compare con el ejercicio 25).

68. Pruebe que si p y q son divisibles entre k, entonces $p + q$ es divisible entre k.

Rincón de solución de problemas

Inducción matemática

Problema

Defina

$$H_k = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{k} \qquad (1)$$

para toda $k \geq 1$. Los números $H_1, H_2, \ldots$ se llaman *números armónicos*. Pruebe que

$$H_{2^n} \geq 1 + \frac{n}{2} \qquad (2)$$

Para toda $n \geq 0$.

Cómo atacar el problema

Con frecuencia es una buena idea comenzar a atacar un problema viendo algunos ejemplos concretos de las expresiones que se consideran. Veamos H_k para algún valor pequeño de k. El valor más pequeño de k para el que H_k está definida es $k = 1$. En este caso, el último término $1/k$ en la definición de H_k es igual a $1/1 = 1$. Como el primero y último términos coinciden,

$$H_1 = 1.$$

Para $k = 2$, el último término $1/k$ en la definición de H_k es igual a 1/2, entonces

$$H_2 = 1 + \frac{1}{2}.$$

De manera similar, se encuentra que

$$H_3 = 1 + \frac{1}{2} + \frac{1}{3},$$

$$H_4 = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4}.$$

Se observa que H_1 aparece como el primer término de H_2, H_3 y H_4, que H_2 aparece como los dos primeros términos de H_3 y H_4, y que H_3 aparece como los tres primeros términos de H_4. En general, H_m aparece como los m primeros términos de H_k si $m \leq k$. Esta observación será útil más adelante porque el paso inductivo en una prueba por inducción debe relacionar las instancias más pequeñas de un problema con las instancias más grandes del problema.

En general, es una buena estrategia para retrasar la combinación de términos y simplificar lo más tarde posible; ésta es, por ejemplo, la razón para dejar H_4 como la suma de cuatro términos en lugar de escribir $H_4 = 25/12$. Como se dejó H_4 como la suma de cuatro términos, se pudo ver que cada una de H_1, H_2 y H_3 aparece en la expresión para H_4.

Cómo encontrar una solución

El paso base consiste en probar la afirmación que se da para el valor más pequeño de n, que aquí es $n = 0$. Para $n = 0$, la desigualdad (2) que debemos probar se convierte en

$$H_{2^0} \geq 1 + \frac{0}{2} = 1.$$

Se ha observado que $H_1 = 1$. Así, la desigualdad (2) es verdadera cuando $n = 0$; de hecho, la desigualdad es una igualdad. (Recuerde que por definición, si $x = y$ es verdadera, entonces $x \geq y$ también es verdadera).

Sigamos al paso inductivo. Es una buena idea escribir lo que se supone (aquí, el caso n).

$$H_{2^n} \geq 1 + \frac{n}{2}, \tag{3}$$

y lo que se debe probar (aquí el caso $n + 1$),

$$H_{2^{n+1}} \geq 1 + \frac{n+1}{2}. \tag{4}$$

También es una buena idea escribir las fórmulas para cualquier expresión que ocurra. Usando la ecuación (1), se escribe

$$H_{2^n} = 1 + \frac{1}{2} + \cdots + \frac{1}{2^n} \tag{5}$$

y

$$H_{2^{n+1}} = 1 + \frac{1}{2} + \cdots + \frac{1}{2^{n+1}}.$$

No es tan evidente de la última ecuación que $H_2{}^n$ aparece como los primeros 2^n términos de $H_2{}^{n+1}$. Escribamos la última ecuación como

$$H_{2^{n+1}} = 1 + \frac{1}{2} + \cdots + \frac{1}{2^n} + \frac{1}{2^n + 1} + \cdots + \frac{1}{2^{n+1}} \tag{6}$$

para hacer claro que H_{2n} aparece como los primeros 2^n términos de H_{2n+1}.

Por claridad, se escribe el término que sigue a $1/2^n$. Observe que el denominador aumenta en 1, por lo que el término que sigue a $1/2^n$ es $1/(2^n + 1)$. Además, observe que hay una gran diferencia entre $1/(2^n + 1)$, el término que sigue a $1/2^n$, y $1/2^{n+1}$, el último término de la ecuación (6).

Usando las ecuaciones (5) y (6) se puede relacionar $H_2{}^n$ con $H_2{}^{n+1}$ explícitamente escribiendo

$$H_{2^{n+1}} = H_{2^n} + \frac{1}{2^n + 1} + \cdots + \frac{1}{2^{n+1}}. \tag{7}$$

Combinando (3) y (7) se obtiene

$$H_{2^{n+1}} \geq 1 + \frac{n}{2} + \frac{1}{2^n + 1} + \cdots + \frac{1}{2^{n+1}}. \tag{8}$$

Esta desigualdad muestra que $H_2{}^{n+1}$ es mayor o igual que

$$1 + \frac{n}{2} + \frac{1}{2^n + 1} + \cdots + \frac{1}{2^{n+1}},$$

pero nuestra meta (4) es demostrar que $H_{2^{n+1}}$ es mayor o igual que $1 + (n + 1)/2$. Se logrará la meta si se demuestra que

$$1 + \frac{n}{2} + \frac{1}{2^n + 1} + \cdots + \frac{1}{2^{n+1}} \geq 1 + \frac{n+1}{2}.$$

En general, para probar una desigualdad, se sustituyen los términos en la expresión más grande con términos más pequeños de manera que la expresión que se obtiene sea igual a la expresión más pequeña; o se sustituyen términos en la expresión más pequeña con términos más grandes de modo que la expresión resultante sea igual a la expresión más grande. Se sustituirá cada término de la suma

$$\frac{1}{2^n + 1} + \cdots + \frac{1}{2^{n+1}}$$

con el término más pequeño $1/2^{n+1}$ en la suma. Se obtiene

$$\frac{1}{2^n + 1} + \cdots + \frac{1}{2^{n+1}} \geq \frac{1}{2^{n+1}} + \cdots + \frac{1}{2^{n+1}}.$$

Como existen 2^n términos en la última suma, cada uno igual a $1/2^{n+1}$, la desigualdad anterior se rescribe como

$$\frac{1}{2^n + 1} + \cdots + \frac{1}{2^{n+1}} \geq \frac{1}{2^{n+1}} + \cdots + \frac{1}{2^{n+1}} = 2^n \frac{1}{2^{n+1}} = \frac{1}{2}. \tag{9}$$

Combinando (8) y (9),

$$H_{2^{n+1}} \geq 1 + \frac{n}{2} + \frac{1}{2} = 1 + \frac{n+1}{2}.$$

Se tiene el resultado deseado y el paso inductivo queda completo.

Solución formal

La solución formal se escribe como sigue.

Paso base $(n = 0)$.

$$H_{2^0} = 1 \geq 1 = 1 + \frac{0}{2}.$$

Paso inductivo Se supone (2). Ahora

$$\begin{aligned} H_{2^{n+1}} &= 1 + \frac{1}{2} + \cdots + \frac{1}{2^n} + \frac{1}{2^n + 1} + \cdots + \frac{1}{2^{n+1}} \\ &= H_{2^n} + \frac{1}{2^n + 1} + \cdots + \frac{1}{2^{n+1}} \\ &\geq 1 + \frac{n}{2} + \frac{1}{2^{n+1}} + \cdots + \frac{1}{2^{n+1}} \\ &= 1 + \frac{n}{2} + 2^n \frac{1}{2^{n+1}} \\ &= 1 + \frac{n}{2} + \frac{1}{2} = 1 + \frac{n+1}{2}. \end{aligned}$$

Resumen de técnicas de solución de problemas

- Observe ejemplos concretos de las expresiones bajo consideración, en general, para valores pequeños de las variables.
- Busque expresiones para valores pequeños de n que aparezcan dentro de las expresiones para valores más grandes de n. En particular, el paso inductivo depende de relacionar el caso n con el caso $n + 1$.

- Retrase la combinación y simplificación de términos lo más posible para ayudar a descubrir las relaciones entre las expresiones.
- Escriba completos los casos específicos para probar, en particular, el valor más pequeño de n para el paso base, el caso n que se supone en el paso inductivo, y el caso $n + 1$ para probar el paso inductivo. Escriba las fórmulas para las diferentes expresiones que surjan.
- Para probar una desigualdad, sustituya términos en la expresión más grande con términos más pequeños, de manera que la expresión que obtenga sea igual a la expresión más pequeña, o sustituya términos en la expresión más pequeña con términos mayores, de modo que la expresión obtenida sea igual a la expresión más grande.

Comentarios

La serie

$$1 + \frac{1}{2} + \frac{1}{3} + \cdots,$$

que surge en cálculo, se llama *serie armónica*. La desigualdad (2) muestra que los números armónicos aumentan sin límite. En terminología de cálculo, la serie armónica diverge.

Ejercicios

1. Pruebe que $H_{2^n} \leq 1 +$ para toda $n \geq 0$.

2. Pruebe que

$$H_1 + H_2 + \cdots + H_n = (n + 1)H_n - n$$

para toda $n \geq 1$.

3. Pruebe que

$$H_n = H_{n+1} - \frac{1}{n+1}$$

para toda $n \geq 1$.

4. Pruebe que

$$1 \cdot H_1 + 2 \cdot H_2 + \cdots + nH_n = \frac{n(n+1)}{2} H_{n+1} - \frac{n(n+1)}{4}$$

para toda $n \geq 1$.

1.8 ➜ Forma fuerte de inducción y la propiedad del buen orden

En el paso inductivo de la inducción matemática presentado en la sección 1.7, se supone que la afirmación n es cierta y después se prueba que la afirmación $n + 1$ es cierta. En otras palabras, para probar que una afirmación es verdadera (afirmación $n + 1$), se supone la verdad de su predecesor inmediato (afirmación n). En algunos casos en el paso inductivo, para probar que una afirmación es verdadera, resulta útil suponer la verdad de *todas* las afirmaciones precedentes (no sólo del predecesor inmediato). La **forma fuerte de inducción matemática** permite suponer la verdad de todas las afirmaciones precedentes. Siguiendo la convención usual, la afirmación que se va a probar se denota por n en lugar de $n + 1$. Se establecerá de manera formal la forma fuerte de inducción matemática.

Forma fuerte de inducción matemática

Suponga que se tiene una función proposicional $S(n)$ cuyo dominio de discurso es el conjunto de enteros mayores o iguales que n_0. Suponga que

$S(n_0)$ es verdadera;
para toda $n > n_0$, si $S(k)$ es verdadera para toda k, $n_0 \leq k < n$, entonces $S(n)$ es verdadera.

Entonces $S(n)$ es verdadera para todo entero $n \geq n_0$.

Las dos formas de inducción matemática son lógicamente equivalentes (vea el ejercicio 29).

Se presentarán varios ejemplos que ilustran el uso de la forma fuerte de inducción matemática.

Ejemplo 1.8.1 ▶

Utilice inducción matemática para demostrar que el importe postal de 4 centavos o más se puede lograr usando sólo timbres de 2 y 5 centavos.

Antes de dar una demostración formal, se analiza la idea de la prueba por inducción matemática. Considere el paso inductivo, donde se quiere probar que el importe de n centavos se puede lograr sólo con timbres de 2 y 5 centavos. Sería en particular sencillo probar esta afirmación si se pudiera suponer que se puede lograr un importe de $n – 2$ centavos. Entonces, con sólo sumar un timbre de 2 centavos se tendría el importe de n centavos. ¡Qué sencillo! Si usamos la forma fuerte de inducción matemática, *se puede* suponer la verdad

de la afirmación para toda $k < n$. En particular, se supone la verdad de la afirmación para $k = n - 2$. Así, la forma fuerte de inducción matemática permite dar una demostración correcta basada en el razonamiento informal.

Existe un punto sutil al que debe ponerse atención. Se están considerando sólo importes de 4 centavos o más. Entonces, cuando $n = 5$, $n - 2$ no es un valor válido; esto es, como $n - 2 < 4$, no se puede suponer que se logrará un importe de $n - 2$ centavos. Por lo tanto, además del caso $n = 4$, debe verificarse explícitamente el caso $n = 5$. Sólo cuando $n \geq 6$, $n - 2$ es válido. Así, deben verificarse de manera explícita los casos $n = 4$ y $n = 5$, que ahora se convierten en los pasos base.

Pasos base ($n = 4$, $n = 5$)

Se pueden reunir importes de 4 centavos usando dos timbres de 2 centavos. Se logran importes de 5 centavos usando un timbre de 5 centavos. El paso base queda verificado.

Paso inductivo

Se supondrá que $n \geq 6$ y que el importe de k centavos o más se puede lograr usando sólo timbres de 2 y 5 centavos para $4 \leq k < n$.

Por la suposición de inducción, se puede lograr un importe de $n - 2$ centavos. Se agrega un timbre de 2 centavos para reunir el importe de n centavos. El paso inductivo queda completo. ◀

Ejemplo 1.8.2 ▶

Cuando un elemento de una secuencia se define en términos de alguno de sus predecesores, la forma fuerte de inducción matemática resultará útil para probar una propiedad de la secuencia. Por ejemplo, suponga que la secuencia $c_1, c_2, \ldots$ está definida por las ecuaciones†

$$c_1 = 0, \qquad c_n = c_{\lfloor n/2 \rfloor} + n \quad \text{para toda } n > 1.$$

Como ejemplos,

$$\begin{aligned} c_2 &= c_{\lfloor 2/2 \rfloor} + 2 = c_{\lfloor 1 \rfloor} + 2 = c_1 + 2 = 0 + 2 = 2, \\ c_3 &= c_{\lfloor 3/2 \rfloor} + 3 = c_{\lfloor 1.5 \rfloor} + 3 = c_1 + 3 = 0 + 3 = 3, \\ c_4 &= c_{\lfloor 4/2 \rfloor} + 4 = c_{\lfloor 2 \rfloor} + 4 = c_2 + 4 = 2 + 4 = 6, \\ c_5 &= c_{\lfloor 5/2 \rfloor} + 5 = c_{\lfloor 2.5 \rfloor} + 5 = c_2 + 5 = 2 + 5 = 7. \end{aligned}$$

Se usa la inducción fuerte para probar que

$$c_n < 4n \qquad \text{para toda } n \geq 1.$$

Paso base ($n = 1$)

$$c_1 = 0 < 4 = 4 \cdot 1.$$

El paso base queda verificado.

Paso inductivo

Se supone que

$$c_2 < 4k$$

para toda $k < n$, y se prueba que

$$c_n < 4n.$$

† El piso de x, $\lfloor x \rfloor$, es el entero más grande menor o igual que x (vea la sección 2.2). De manera informal, se está "redondeando hacia abajo". Ejemplos: $\lfloor 2.3 \rfloor = 2$, $\lfloor 5 \rfloor = 5$, $\lfloor -2.7 \rfloor = -3$

Observe que $\lfloor n/2 \rfloor < n$. Por lo tanto, con $k = \lfloor n/2 \rfloor$, por la suposición de inducción

$$c_{\lfloor n/2 \rfloor} = c_k < 4k = 4\lfloor n/2 \rfloor.$$

Ahora

$$c_n = c_{\lfloor n/2 \rfloor} + n < 4\lfloor n/2 \rfloor + n \leq 4(n/2) + n = 3n < 4n;$$

y el paso inductivo queda completo. ◀

Ejemplo 1.8.3 ▶

Suponga que se insertan paréntesis y después se multiplican los n números $a_1 a_2 \ldots a_n$. Por ejemplo, si $n = 4$, podrían insertarse paréntesis como se muestra:

$$(a_1 a_2)(a_3 a_4). \tag{1.8.1}$$

Aquí, primero se multiplicaría a_1 por a_2 para obtener $a_1 a_2$ y a_3 por a_4 para obtener $a_3 a_4$. Después, se multiplicaría $a_1 a_2$ por $a_3 a_4$ para obtener $(a_1 a_2)(a_3 a_4)$. Note que el número de multiplicaciones es tres. Pruebe que si se insertan paréntesis de cualquier manera y después se multiplican los n números $a_1 a_2 \ldots a_n$, se realizan $n - 1$ multiplicaciones.

Se usará la inducción fuerte para probar el resultado.

Paso base ($n = 1$)

Se necesitan 0 multiplicaciones para calcular a_1. Esto verifica el paso base.

Paso inductivo

Se supone que para toda $k < n$ toma $k - 1$ multiplicaciones calcular el producto de k números si se insertan paréntesis de cualquier manera. Debe probarse que se necesitan n multiplicaciones para calcular el producto $a_1 a_2 \ldots a_n$ si se insertan paréntesis de cualquier manera.

Suponga que se insertan paréntesis en el producto $a_1 a_2 \ldots a_n$. Considere la multiplicación final, que se ve como

$$(a_1 \cdots a_t)(a_{t+1} \cdots a_n),$$

para alguna $t < n$. [Por ejemplo, en la ecuación (1.8.1), $t = 2$]. Existen $t < n$ términos en el primer conjunto de paréntesis y $n - t < n$ términos en el segundo conjunto de paréntesis. Por la suposición inductiva, se requieren $t - 1$ multiplicaciones para calcular $a_1 \ldots a_t$ y $n - t - 1$ multiplicaciones para calcular $a_{t+1} \ldots a_n$, sin importar cuántos paréntesis se inserten. Se necesita una multiplicación adicional para multiplicar $a_1 \ldots a_t$ por $a_{t+1} \ldots a_n$. Entonces, el número total de multiplicaciones es

$$(t - 1) + (n - t - 1) + 1 = n - 1.$$

El paso inductivo queda completo. ◀

Propiedad del buen orden

La **propiedad del buen orden para enteros no negativos** establece que todo conjunto no vacío de enteros no negativos tiene un elemento menor. Esta propiedad es equivalente a las dos formas de inducción (vea los ejercicios 27 al 29). Se usará la propiedad del buen orden para probar algo familiar de la división: cuando se divide un entero n entre un entero positivo d, se obtiene el cociente q y un residuo r que satisface $0 \leq r < d$ de manera que $n = dq + r$.

Ejemplo 1.8.4 ▶

Cuando se divide $n = 74$ entre $d = 13$

$$\begin{array}{r} 5 \\ 13\overline{)74} \\ 65 \\ 9 \end{array}$$

se obtiene el cociente $q = 5$ y el residuo $r = 9$. Observe que r satisface $0 \leq r < d$; es decir, $0 \leq 9 < 13$. Se tiene

$$n = 74 = 13 \cdot 5 + 9 = dq + r.$$ ◀

Teorema 1.8.5

Teorema del cociente-residuo

Si d y n son enteros, $d > 0$, existen enteros q (cociente) y r (residuo) que satisfacen

$$n = dq + r \qquad 0 \leq r < d.$$

Más aún, q y r son únicos; es decir, si

$$n = dq_1 + r_1 \qquad 0 \leq r_1 < d$$

y

$$n = dq_2 + r_2 \qquad 0 \leq r_2 < d$$

entonces $q_1 = q_2$ y $r_1 = r_2$.

Es posible idear una prueba del teorema 1.8.5 observando con cuidado la técnica usada en la división. ¿Por qué es 5 el cociente en el ejemplo 1.8.4? Porque $q = 5$ hace que el residuo $n - dq$ sea no negativo y tan pequeño como es posible. Si, por ejemplo, $q = 3$, el residuo sería $n - dq = 74 - 13 \cdot 3 = 35$, que es demasiado grande. Como otro ejemplo, si $q = 6$, el residuo sería $n - dq = 74 - 13 \cdot 6 = -4$, que es negativo. La existencia del residuo no negativo más pequeño $n - dq$ está garantizada por la propiedad del buen orden.

Demostración del teorema 1.8.5 Sea X el conjunto de todos los enteros de la forma $n - dk$ donde $n - dk \geq 0$ y k es un entero. Se demostrará que X es no vacío usando la prueba por casos. Si $n \geq 0$, entonces

$$n - d \cdot 0 = n \geq 0$$

así que n está en X. Suponga que $n < 0$. Como d es un entero positivo, $1 - d \leq 0$. Por tanto,

$$n - dn = n(1 - d) \geq 0.$$

En este caso, $n - dn$ está en X. Por lo tanto, X es no vacío.

Como X es un conjunto no vacío de enteros negativos, por la propiedad del buen orden, X tiene un elemento más pequeño que denotamos por r. Sea q el valor específico de k para el que $r = n - dq$. Entonces

$$n = dq + r.$$

Como r está en X, $r \geq 0$. Se usa la prueba por contradicción para demostrar que $r < d$. Suponga que $r \geq d$. Entonces

$$n - d(q + 1) = n - dq - d = r - d \geq 0.$$

Así, $n - d(q + 1)$ está en X. Además, $n - d(q + 1) = r - d < r$. Pero r es el entero más pequeño en X. Esta contradicción demuestra que $r < d$.

Se ha demostrado que si d y n son enteros, $d > 0$, existen enteros q y r que satisfacen

$$n = dq + r \qquad 0 \leq r < d.$$

Ahora se analiza la unicidad de q y r. Suponga que

$$n = dq_1 + r_1 \qquad 0 \leq r_1 < d$$

y

$$n = dq_2 + r_2 \qquad 0 \le r_2 < d.$$

Debe demostrarse que $q_1 = q_2$ y $r_1 = r_2$. Restando las ecuaciones anteriores se obtiene

$$0 = n - n = (dq_1 + r_1) - (dq_2 + r_2) = d(q_1 - q_2) - (r_2 - r_1),$$

que se rescribe como

$$d(q_1 - q_2) = r_2 - r_1.$$

La ecuación anterior demuestra que d divide a $r_2 - r_1$. Sin embargo, $0 \le r_1 < d$ y $0 \le r_2 < d$,

$$-d < r_2 - r_1 < d.$$

Pero el único entero estrictamente entre $-d$ y d divisible por d es 0. Por lo tanto,

$$r_1 = r_2.$$

Así,

$$d(q_1 - q_2) = 0;$$

y con ello

$$q_1 = q_2.$$

Y la prueba queda completa. ◀

Observe que, en el teorema 1.8.5, el residuo r es cero si y sólo si d divide a n.

Sugerencias para resolver problemas

En el paso inductivo de la forma fuerte de inducción matemática, la meta es probar el caso n. Para hacerlo, se pueden suponer *todos* los casos precedentes (no sólo el precedente inmediato como en la sección 1.7). Siempre es posible usar la forma fuerte de inducción matemática. Si ocurre que se necesita sólo el caso precedente inmediato en el paso inductivo, simplemente se usa la forma de inducción matemática de la sección 1.7. No obstante, suponer todos los casos anteriores potencialmente da más con qué trabajar para probar el caso n.

Sección de ejercicios de repaso

1. Enuncie la forma fuerte de inducción matemática.

2. Enuncie la propiedad del buen orden.

3. Enuncie el teorema del cociente-residuo.

Ejercicios

1. Demuestre que un importe postal de 6 centavos o más se logra usando sólo timbres de 2 y 7 centavos.

2. Demuestre que el importe postal de 24 centavos o más se logra usando sólo timbres de 5 y 7 centavos.

★ ***3.*** Utilice la forma

Si $S(n)$ es verdadera, entonces $S(n + 1)$ es verdadera

del paso inductivo para probar la afirmación del ejemplo 1.8.1.

★ **4.** Utilice la forma

Si $S(n)$ es verdadera, entonces $S(n + 1)$ es verdadera

del paso inductivo para probar la afirmación en el ejercicio 1.

★ **5.** Utilice la forma

Si $S(n)$ es verdadera, entonces $S(n + 1)$ es verdadera

del paso inductivo para probar la afirmación en el ejercicio 2.

Los ejercicios 6 y 7 se refieren a la secuencia c_1, c_2,... definida por las ecuaciones

$$c_1 = 0, \qquad c_n = c_{\lfloor n/2 \rfloor} + n^2 \text{ para toda } n > 1.$$

6. Calcule c_2, c_3, c_4 y c_5.

7. Pruebe que $c_n < 4n^2$ para toda $n \ge 1$.

Los ejercicios 8 al 10 se refieren a la secuencia $c_1, c_2, \ldots$ definida por las ecuaciones

$$c_1 = 0, \qquad c_n = 4c_{\lfloor n/2 \rfloor} + n \ \text{ para toda } n > 1.$$

8. Calcule c_2, c_3, c_4 y c_5.

9. Pruebe que $c_n \leq 4(n-1)^2$ para toda $n \geq 1$.

10. Pruebe que $(n+1)^2/8 < c_n$ para toda $n \geq 2$. *Pista:* Los pasos base son $n = 2, 3$. Además $\lfloor n/2 \rfloor \geq (n-1)/2$ para toda n.

11. Suponga que se tienen dos pilas de cartas cada una con n cartas. Dos jugadores juegan el siguiente juego. Cada jugador, en su turno, elige una pila y quita cualquier número de cartas, pero al menos una, de la pila elegida. El jugador que quita la última carta gana el juego. Muestre que el segundo jugador siempre puede ganar.

En los ejercicios 12 al 17, encuentre el cociente q y el residuo r como en el teorema 1.8.5 cuando n se divide entre d.

12. $n = 47, d = 9$

13. $n = -47, d = 9$

14. $n = 7, d = 9$

15. $n = -7, d = 9$

16. $n = 0, d = 9$

17. $n = 47, d = 47$

Los egipcios de la antigüedad expresaban una fracción como la suma de fracciones cuyos numeradores eran 1. Por ejemplo, 5/6 se expresaba como

$$\frac{5}{6} = \frac{1}{2} + \frac{1}{3}.$$

Decimos que la fracción p/q, donde p y q son enteros positivos, está en forma egipcia si

$$\frac{p}{q} = \frac{1}{n_1} + \frac{1}{n_2} + \cdots + \frac{1}{n_k}, \qquad \textbf{(1.8.2)}$$

donde $n_1, n_2, \ldots, n_k$ son enteros positivos que satisfacen $n_1 < n_2 < \ldots < n_k$.

18. Demuestre que la representación (1.8.2) no tiene que ser única representando 5/6 de dos maneras diferentes.

★**19.** Demuestre que la representación (1.8.2) nunca es única.

20. Siguiendo los pasos descritos, proporcione una prueba por inducción sobre p para demostrar que toda fracción p/q con $0 < p/q < 1$ puede expresarse en forma egipcia.

a) Verifique el paso base ($p = 1$).

b) Suponga que $0 < p/q < 1$ y que todas las fracciones i/q', con $1 \leq i < p$ y q' arbitrarios, se pueden expresar en la forma egipcia. Seleccione el menor entero positivo n con $1/n \leq p/q$. Demuestre que

$$n > 1 \qquad \text{y} \qquad \frac{p}{q} < \frac{1}{n-1}.$$

c) Demuestre que si $p/q = 1/n$, la prueba queda completa.

d) Suponga que $1/n < p/q$. Sea

$$p_1 = np - q \qquad \text{y} \qquad q_1 = nq.$$

Demuestre que

$$\frac{p_1}{q_1} = \frac{p}{q} - \frac{1}{n}, \quad 0 < \frac{p_1}{q_1} < 1, \qquad \text{y} \qquad p_1 < p.$$

Concluya que

$$\frac{p_1}{q_1} = \frac{1}{n_1} + \frac{1}{n_2} + \cdots + \frac{1}{n_k}$$

y $n, n_1, \ldots, n_k$ son diferentes.

e) Demuestre que $p_1/q_1 < 1/n$.

f) Demuestre que

$$\frac{p}{q} = \frac{1}{n} + \frac{1}{n_1} + \cdots + \frac{1}{n_k}$$

y $n, n_1, \ldots, n_k$ son diferentes.

21. Use el método del ejercicio anterior para encontrar formas egipcias para 3/8, 5/7 y 3/19.

★***22.*** Demuestre que cualquier fracción p/q, donde p y q son enteros positivos, se puede escribir en la forma egipcia. (No se está suponiendo que $p/q < 1$).

★***23.*** Demuestre que cualquier tablero deficiente de $n \times n$ se puede enlosar con trominos si n es impar, $n > 5$ y 3 divide a $n^2 - 1$. *Sugerencia:* Use las ideas mencionadas en la sugerencia del ejercicio 33, sección 1.7.

★**24.** Demuestre que cualquier tablero deficiente de $n \times n$ se puede enlosar con trominos si n es par, $n > 8$ y 3 divide a $n^2 - 1$. *Sugerencia:* Tome como base el hecho de que un tablero deficiente de 4×4 se puede enlosar con trominos; ejercicios 23 y 31, sección 1.7.

25. Proporcione una prueba alternativa de la existencia de q y r en el teorema 1.8.5 para el caso $n \geq 0$; para ello demuestre primero que el conjunto X de todos los enteros k donde $dk > n$ es un conjunto no vacío de enteros no negativos, después demuestre que X tiene un elemento menor, y por último analice el elemento menor de X.

26. Proporcione una prueba alternativa de la existencia de q y r en el teorema 1.8.5 usando la forma de inducción matemática donde el paso inductivo es "si $S(n)$ es verdadera, entonces $S(n+1)$ es verdadera". *Sugerencia:* Primero suponga que $n > 0$. Maneje el caso $n = 0$ por separado. Reduzca el caso $n < 0$ al caso $n > 0$.

★***27.*** Suponga la forma de inducción matemática donde el paso inductivo es "si $S(n)$ es verdadera, entonces $S(n+1)$ es verdadera". Pruebe la propiedad del buen orden.

★**28.** Suponga la propiedad del buen orden. Pruebe la forma fuerte de inducción matemática.

★**29.** Demuestre que la forma fuerte de inducción matemática y la forma de inducción matemática donde el paso inductivo es "si $S(n)$ es verdadera, entonces $S(n+1)$ es verdadera" son equivalentes. Es decir, suponga la forma fuerte de inducción matemática y pruebe la forma alternativa; después suponga la forma alternativa y pruebe la forma fuerte de inducción matemática.

Notas

Las referencias generales de matemáticas discretas son [Graham, 1994; Liu, 1985; Tucker]. [Knuth, 1997, 1998a, 1998b] es la referencia clásica para la mayor parte de este material.

[Barker; Copi; Edgar] son libros de introducción a la lógica. Un análisis más avanzado se encuentra en [Davis]. El primer capítulo del libro de geometría de [Jacobs] está dedicado a lógica básica. [D'Angelo; Solow] estudian el problema de cómo construir demostraciones. Si desea la historia de la lógica vea [Kline]. El papel de la lógica en el razonamiento acerca de programas de computadora se estudia en [Gries].

Enlosar con poliominos es el tema del libro de [Martin].

Repaso del capítulo

Sección 1.1

1. Lógica
2. Proposición
3. Conjunción: p y q, $p \wedge q$
4. Disyunción: p o q, $p \vee q$
5. Negación: no p, $\neg p$
6. Tabla de verdad
7. or-exclusivo de las proposiciones p, q: p o q, pero no ambos

Sección 1.2

8. Proposición condicional: si p, entonces q; $p \rightarrow q$
9. Hipótesis
10. Conclusión
11. Condición necesaria
12. Condición suficiente
13. Recíproca de $p \rightarrow q$: $q \rightarrow p$
14. Proposición bicondicional: p si y sólo si q, $p \leftrightarrow q$
15. Equivalencia lógica: $P \equiv Q$
16. Leyes de De Morgan para lógica $\neg(p \vee q) \equiv \neg p \wedge \neg q$, $\neg(p \wedge q) \equiv \neg p \vee \neg q$
17. Contrapositiva de $p \rightarrow q$: $\neg q \rightarrow \neg p$

Sección 1.3

18. Proposición funcional
19. Dominio de discurso
20. Cuantificador universal
21. Afirmación cuantificada universalmente
22. Contraejemplo
23. Cuantificador existencial
24. Afirmación cuantificada existencialmente
25. Leyes generalizadas de De Morgan para lógica:

$\neg(\forall x\, P(x))$ y $\exists x \neg P(x)$ tienen los mismos valores de verdad.

$\neg(\exists x\, P(x))$ y $\forall x \neg P(x)$ tienen los mismos valores de verdad.

26. Para probar que la afirmación cuantificada universalmente

$$\forall x\, P(x)$$

es verdadera, se demuestra que para toda x en el dominio de discurso, la proposición $P(x)$ es verdadera.

27. Para probar que la afirmación cuantificada existencialmente

$$\exists x\, P(x)$$

es verdadera, encuentre un valor de x en el dominio de discurso para el que $P(x)$ es verdadera.

28. Para probar que la afirmación cuantificada universalmente

$$\forall x\, P(x)$$

es falsa, encuentre un valor de x (contraejemplo) en el dominio de discurso para el que $P(x)$ es falsa.

29. Para probar que la afirmación cuantificada existencialmente

$$\exists x\, P(x)$$

es falsa, demuestre que para toda x en el dominio de discurso, la proposición $P(x)$ es falsa.

Sección 1.4

30. Para probar que

$$\forall x \forall y\, P(x, y)$$

es verdadera, demuestre que $P(x, y)$ es verdadera para todos los valores de x y y en el dominio de discurso.

31. Para probar que

$$\forall x \exists y P(x, y)$$

es verdadera, demuestre que para toda x en el dominio de discurso, existe al menos una y en el dominio de discurso tal que $P(x, y)$ es verdadera.

32. Para probar que

$$\exists x \forall y P(x, y)$$

es verdadera, demuestre que para al menos una x en el dominio de discurso, $P(x, y)$ es verdadera para toda y en el dominio de discurso.

33. Para probar que

$$\exists x \exists y P(x, y)$$

es verdadera, encuentre un valor de x y un valor de y en el dominio de discurso que hagan verdadera a $P(x, y)$.

34. Para probar que

$$\forall x \forall y P(x, y)$$

es falsa, encuentre un valor de x y un valor de y en el dominio de discurso que hagan falsa a $P(x, y)$.

35. Para probar que

$$\forall x \exists y P(x, y)$$

es falsa, demuestre que para al menos una x en el dominio de discurso, $P(x, y)$ es falsa para toda y en el dominio de discurso.

36. Para probar que

$$\exists x \forall y P(x, y)$$

es falsa, demuestre que para toda x en el dominio de discurso, existe un y en el dominio de discurso tal que $P(x, y)$ es falsa.

37. Para probar que

$$\exists x \exists y P(x, y)$$

es falsa, demuestre que $P(x, y)$ es falsa para todos los valores de x y y en el dominio de discurso.

38. Para negar una expresión con cuantificadores anidados, use las leyes generalizadas de De Morgan para lógica.

39. El juego de lógica

Sección 1.5

40. Sistema matemático

41. Axioma

42. Definición

43. Término no definido

44. Teorema

45. Demostración

46. Lema

47. Prueba directa

48. Entero par

49. Entero impar

50. Prueba por contradicción

51. Prueba indirecta

52. Prueba por contrapositiva

53. Prueba por casos

54. Prueba de existencia

55. Razonamiento deductivo

56. Hipótesis

57. Premisas

58. Conclusión

59. Argumento

60. Argumento válido

61. Argumento inválido

62. Reglas de inferencia para proposiciones: *modus ponens*, *modus tollens*, suma, simplificación, conjunción, silogismo hipotético, silogismo disyuntivo.

63. Reglas de inferencia para afirmaciones cuantificadas: particularización universal, generalización universal, particularización existencial, generalización existencial

Sección 1.6

64. Pruebas por resolución; usos: si $p \vee q$ y $\neg p \vee r$ son ambas verdaderas, entonces $q \vee r$ es verdadera.

65. Cláusula: consiste en términos separados por *o*, donde cada término es una variable o la negación de una variable.

Sección 1.7

66. Principio de inducción matemática

67. Paso base: se prueba que la primera instancia es verdadera.

68. Paso inductivo: suponga que la instancia n es verdadera; después se prueba que la instancia $n + 1$ es verdadera.

69. n factorial: $n! = n(n-1)\cdots 1, 0! = 1$

70. Fórmula para la suma de los n primeros enteros positivos:

$$1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$

71. Fórmula para la suma geométrica:

$$ar^0 + ar^1 + \cdots + ar^n = \frac{a(r^{n+1} - 1)}{r - 1}, \qquad r \neq 1$$

Sección 1.8

72. Forma fuerte de inducción matemática

73. Paso base para la forma fuerte de inducción matemática: se prueba que la primera instancia es verdadera.

74. Paso inductivo de la forma fuerte de inducción matemática: se supone verdadera para todas las instancias menores que n; después se prueba que la instancia n es verdadera.

75. Propiedad de buen orden: todo conjunto no vacío de enteros no negativos tiene un elemento menor.

76. Teorema del cociente-residuo: si d y n son enteros, $d > 0$, existen enteros q (cociente) y r (residuo) que satisfacen $n = dq + r$, $0 \leq r < d$. Más aún, q y r son únicos.

Autoevaluación del capítulo

Sección 1.1

1. Si p, q y r son verdaderas, encuentre la tabla de verdad de la proposición

$(p \vee q) \wedge \neg((\neg p \wedge r) \vee q)$.

2. Escriba la tabla de verdad de la proposición $\neg(p \wedge q) \vee (p \vee \neg r)$.

3. Formule la proposición $p \wedge (\neg q \vee r)$ en palabras usando

p: Tomo el curso de administración de hoteles.

q: Tomo el curso de supervisión recreativa.

r: Tomo el curso de cultura popular.

4. Suponga que a, b y c son números reales. Represente la afirmación

$$a < b \text{ o } (b < c \text{ y } a \geq c)$$

simbólicamente, cuando

$$p\text{: } a < b, \qquad q\text{: } b < c, \qquad r\text{: } a < c.$$

Sección 1.2

5. Restablezca la proposición "Una condición necesaria para que Luis obtenga 10 en matemáticas discretas es que estudie duro" en la forma de una proposición condicional.

6. Escriba la recíproca y la contrapositiva de la proposición del ejercicio 5.

7. Si p es verdadera y q y r son falsas, encuentre el valor de verdad de la proposición

$$(p \vee q) \rightarrow \neg r.$$

8. Represente la afirmación

Si ($a \geq c$ o $b < c$), entonces $b \geq c$

simbólicamente usando las definiciones del ejercicio 4.

Sección 1.3

9. ¿Es la afirmación:

El equipo ganó el campeonato de la Asociación Nacional de Básquetbol en 2004

una proposición? Explique.

10. La afirmación del ejercicio 9, ¿es una función proposicional? Explique.

Sea $P(n)$ la afirmación

$$n \text{ y } n + 2 \text{ son primos.}$$

En los ejercicios 11 y 12, escriba la afirmación en palabras y diga si es verdadera o falsa.

11. Para todo entero positivo n, $P(n)$.

12. Para algún entero positivo n, $P(n)$.

Sección 1.4

13. Sea $K(x, y)$ la función proposicional "x conoce a y". El dominio de discurso es el conjunto de estudiantes que toman matemáticas discretas. Represente la aseveración "alguien no conoce a nadie" en símbolos.

14. Escriba la negación de la aseveración del ejercicio 13 simbólicamente y en palabras.

15. Determine si la afirmación

$$\forall x \exists y (x = y^3)$$

es verdadera o falsa. El dominio de discurso es el conjunto de números reales. Explique su respuesta. Explique, en palabras, el significado de la afirmación.

16. Use las leyes generalizadas de De Morgan para lógica para escribir la negación de

$$\forall x \exists y \forall z P(x, y, z).$$

Sección 1.5

17. Demuestre, con una prueba por contradicción, que si cuatro equipos juegan siete juegos, algún par de equipos juega al menos dos veces.

18. Distinga entre los términos *axioma* y *definición*.

19. ¿Cuál es la diferencia entre una prueba directa y una prueba por contradicción?

20. Determine si el siguiente argumento es válido.

$$\begin{array}{l} p \rightarrow q \vee r \\ p \vee \neg q \\ r \vee q \\ \hline \therefore q \end{array}$$

Sección 1.6

21. Encuentre una expresión, que sea el y de las cláusulas, equivalente a $(p \vee q) \rightarrow r$.

22. Encuentre una expresión, que sea el y de las cláusulas, equivalente a $(p \vee \neg q) \rightarrow \neg r s$.

23. Use la resolución para probar

$$\begin{array}{l} \neg p \vee q \\ \neg q \vee \neg r \\ p \vee \neg r \\ \hline \therefore \neg r \end{array}$$

24. Realice de nuevo el ejercicio 23 usando la resolución y la prueba por contradicción.

Sección 1.7

Utilice inducción matemática para probar que las afirmaciones en los ejercicios 25 al 28 son verdaderas para todo entero positivo n.

25. $2 + 4 + \cdots + 2n = n(n + 1)$

26. $2^2 + 4^2 + \cdots + (2n)^2 = \dfrac{2n(n + 1)(2n + 1)}{3}$

27. $\dfrac{1}{2!} + \dfrac{2}{3!} + \cdots + \dfrac{n}{(n + 1)!} = 1 - \dfrac{1}{(n + 1)!}$

28. $2^{n+1} < 1 + (n + 1)2^n$

Sección 1.8

29. Encuentre el cociente q y el residuo r como en el teorema 1.8.5 cuando $n = 101$ se divide entre $d = 11$.

Los ejercicios 30 y 31 se refieren a la secuencia c_1, c_2, . . . definida por las ecuaciones

$$c_1 = 0, \qquad c_n = 2c_{\lfloor n/2 \rfloor} + n \text{ para toda } n > 1.$$

30. Calcule c_2, c_3, c_4 y c_5.

31. Pruebe que $c_n \leq n \log n$ para toda $n \geq 1$.

32. Defina una *cota superior* para un conjunto no vacío X de números como un número a que satisface $a \geq x$ para toda x en X. Utilice la propiedad del buen orden para demostrar que cualquier conjunto no vacío X de enteros no negativos que tiene una cota superior contiene un elemento que es el más grande. *Sugerencia:* Considere el conjunto de cotas superiores enteras de X.

Ejercicios para computadora

1. Escriba un programa que lea una expresión lógica en p y q e imprima la tabla de verdad de la expresión.

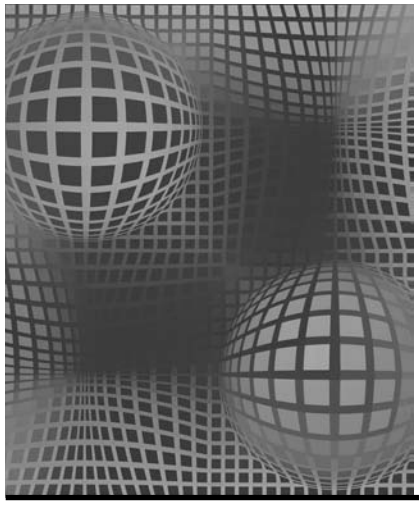
2. Escriba un programa que lea un expresión lógica en p, q y r e imprima la tabla de verdad de la expresión.

3. Escriba un programa que pruebe si dos expresiones lógicas en p y q son lógicamente equivalentes.

4. Escriba un programa que pruebe si dos expresiones lógicas en p, q y r son lógicamente equivalentes.

5. Implante una prueba por resolución a través de un programa.

6. Escriba un programa que dé una forma egipcia de una fracción.



Capítulo 2

EL LENGUAJE DE LAS MATEMÁTICAS

Cuando uso una palabra, significa sólo lo que elijo que quiera decir, ni más ni menos.

DE ALICIA EN EL PAÍS DE LAS MARAVILLAS

Este capítulo trata del lenguaje de las matemáticas. Los temas, algunos de los cuales resultarán familiares, son conjuntos, funciones, sucesiones y cadenas. Todas las matemáticas, al igual que las materias que se apoyan en ellas, como las ciencias de la computación y la ingeniería, utilizan estos conceptos fundamentales.

Un conjunto es una colección de objetos; el orden no se toma en cuenta. Las matemáticas discretas se ocupan de estructuras como gráficas (conjuntos de vértices y aristas) y álgebras boleanas (conjuntos con ciertas operaciones definidas en ellos).

Una función asigna a cada miembro de un conjunto X exactamente un miembro de un conjunto Y. Las funciones se utilizan con frecuencia en matemáticas discretas; por ejemplo, se emplean funciones para analizar el tiempo necesario para ejecutar algoritmos.

WWW Una sucesión es un tipo especial de función. Una lista de letras conforme aparecen en una palabra es un ejemplo de sucesión. A diferencia de un conjunto, una sucesión toma en cuenta el orden. (No cabe duda de que el orden es importante ya que, por ejemplo, *mala* y *alma* son palabras diferentes).

2.1 ➜ Conjuntos

El concepto de conjunto es básico en todas las matemáticas y sus aplicaciones. Un conjunto es simplemente una colección de objetos. En ocasiones se hace referencia a los objetos como elementos o miembros. Si un conjunto es finito y no muy grande, es posible describirlo por la lista de los elementos en él. Por ejemplo, la ecuación

WWW

$$A = \{1, 2, 3, 4\} \qquad \textbf{(2.1.1)}$$

describe un conjunto A integrado por cuatro elementos 1, 2, 3 y 4. Un conjunto se determina por sus elementos y no por el orden particular en el que se enumeren. Así, es lo mismo si A se especifica como

$$A = \{1, 3, 4, 2\}.$$

Se supone que los elementos que forman un conjunto son distintos, y aunque por alguna razón podemos tener duplicados en la lista, sólo una ocurrencia de cada uno está en el conjunto. Por esta razón, también podemos describir el conjunto A definido en (2.1.1) como

$$A = \{1, 2, 2, 3, 4\}.$$

Si un conjunto es finito grande o infinito, se describe mediante una propiedad necesaria para ser miembro. Por ejemplo, la ecuación

$$B = \{x \mid x \text{ es un entero par, positivo}\} \qquad \textbf{(2.1.2)}$$

describe el conjunto B formado por todos los enteros pares, positivos; es decir, B consiste en los enteros 2, 4, 6, etcétera. La barra vertical "|" se lee "tal que". La ecuación (2.1.2) se leería "B es igual al conjunto de todas las x tales que x es un entero par, positivo". Aquí, ser "un entero par, positivo" es la propiedad necesaria para pertenecer al conjunto. Observe que la propiedad aparece después de la barra vertical.

Si X es un conjunto finito, se define

$$|X| = \text{número de elementos en } X.$$

Por ejemplo, para el conjunto A en (2.1.1), se tiene $|A| = 4$.

A partir de una descripción de un conjunto X como (2.1.1) o (2.1.2) y un elemento x, es posible determinar si x pertenece o no a X. Si los miembros de X se enuncian como en (2.1.1), sólo tenemos que mirar si x aparece o no en la lista. En una descripción como (2.1.2) se verifica si el elemento x tiene la propiedad indicada. Si x está en el conjunto X, se escribe $x \in X$ y si x no está en X, se escribe $x \notin X$. Por ejemplo, si $x = 1$, entonces $x \in A$, pero $x \notin B$, donde A y B están definidos por las ecuaciones (2.1.1) y (2.1.2).

El conjunto sin elementos se llama **conjunto vacío** (o **nulo**)y se denota por $\emptyset$. Así, $\emptyset = \{\}$.

Dos conjuntos X y Y son **iguales** y escribimos $X = Y$ si X y Y tienen los mismos elementos. Dicho de otra manera, $X = Y$ si para cada x, si $x \in X$, entonces $x \in Y$ (de manera que todo elemento de X es un elemento de Y) y para toda x, si $x \in Y$, entonces $x \in X$ (de manera que todo elemento de Y es un elemento de X). En símbolos, $X = Y$ si y sólo si

$$\forall x((x \in X \to x \in Y) \wedge (x \in Y \to x \in X)).$$

Esta última caracterización es una manera de probar que los dos conjuntos son iguales.

Ejemplo 2.1.1 ▶

Se desea probar que si

$$A = \{x \mid x^2 + x - 6 = 0\} \qquad \text{y} \qquad B = \{2, -3\},$$

entonces $A = B$.

Se debe probar que para toda x, si $x \in A$, entonces $x \in B$ y si $x \in B$, entonces $x \in A$.

Suponga que $x \in A$. Entonces

$$x^2 + x - 6 = 0$$

Al despejar x se encuentra que $x = 2$ o $x = -3$. En cualquier caso, $x \in B$. Por lo tanto, para cada x, si $x \in A$, entonces $x \in$ B.

Ahora suponga que $x \in B$. Entonces, $x = 2$ o $x = -3$. Si $x = 2$, entonces

$$x^2 + x - 6 = 2^2 + 2 - 6 = 0.$$

Por lo tanto, $x \in A$. Si $x = -3$, entonces

$$x^2 + x - 6 = (-3)^2 + (-3) - 6 = 0.$$

De nuevo, $x \in A$. Así, para toda x, si $x \in B$, entonces $x \in A$. Se concluye que $A = B$. ◀

Suponga que X y Y son conjuntos. Si todo el elemento de X es un elemento de Y, se dice que X es un **subconjunto** de Y y se escribe $X \subseteq Y$.

Ejemplo 2.1.2 ▶

Si

$$C = \{1,3\} \quad \text{y} \quad A = \{1,2,3,4\}$$

entonces C es un subconjunto de A, y se escribe $C \subseteq A$. ◀

En símbolos, X es un subconjunto de Y si

$$\forall x(x \in X \rightarrow x \in Y).$$

En palabras, X es un subconjunto de Y si para toda x, si x está en X, entonces x está en Y.

Ejemplo 2.1.3 ▶

Sea

$$X = \{x \mid x^2 + x - 2 = 0\}, \qquad Y = \text{conjunto de enteros},$$

donde el dominio de discurso es el conjunto de números reales. Para probar que $X \subseteq Y$, se debe demostrar que para todo número real x,

$$x \in X \rightarrow x \in Y.$$

Sea x un elemento del dominio de discurso; es decir, suponga que x es un número real. Si $x \in X$, entonces

$$x^2 + x - 2 = 0.$$

Al despejar x, se obtiene $x = 1$ o $x = -2$. Si $x = 1$, x es un entero, de manera que $x \in Y$. Si $x = -2$, x es un entero, de manera que $x \in Y$. Por lo tanto, para toda x en el dominio de discurso,

$$x \in X \rightarrow x \in Y.$$

Se concluye que $X \subseteq Y$. ◀

Si

$$\forall x(x \in X \rightarrow x \in Y),$$

X es un subconjunto de Y. Así, X *no* es un subconjunto de Y si

$$\forall x(x \in X \rightarrow x \in Y)$$

es falsa. De acuerdo con la definición 1.3.4, la proposición anterior es falsa si *para al menos una* x en el dominio de discurso,

$$x \in X \rightarrow x \in Y$$

es falsa. Demostrar que la proposición condicional anterior es falsa es equivalente a probar que su negación

$$\neg(x \in X \rightarrow x \in Y)$$

es verdadera. Por ejemplo, 1.2.13, $\neg(p \rightarrow q)$ es equivalente a $p \wedge \neg q$. Entonces, para demostrar que X no es un subconjunto de Y, debe probarse que, para al menos una x en el dominio de discurso,

$$x \in X \wedge \neg(x \in Y) \equiv x \in X \wedge x \notin Y$$

es verdadera. En palabras, debe encontrarse al menos una x en el dominio de discurso que está en X pero no en Y.

Ejemplo 2.1.4 ▶

Sea

$$X = \{x \mid 3x^2 - x - 2 = 0\}, \qquad Y = \text{conjunto de enteros},$$

donde el dominio de discurso es el conjunto de números reales. Si $x \in X$, entonces

$$3x^2 - x - 2 = 0.$$

Al despejar x, se obtiene $x = 1$ o $x = -2/3$. Tomando $x = -2/3$, se tiene que $x \in X$ pero $x \notin Y$. Por lo tanto, X no es un subconjunto de Y. El valor $x = -2/3$ es un contraejemplo para la aseveración $X \subseteq Y$. ◀

Cualquier conjunto X es un subconjunto de sí mismo, ya que un elemento en X está en X. Además, el conjunto vacío es un subconjunto de todo conjunto. Para dar una prueba formal de este hecho, debe probarse que para cualquier conjunto X,

$$\forall x(x \in \emptyset \rightarrow x \in X)$$

es verdadera. Esto es inmediato puesto que para toda x, la proposición condicional

$$x \in \emptyset \rightarrow x \in X$$

es cierta porque la hipótesis $x \notin \emptyset$ es falsa. Por lo tanto, para cualquier conjunto X,

$$\forall x(x \in \emptyset \rightarrow x \in X)$$

es verdadera, y el conjunto vacío es un subconjunto de todo conjunto.

Si X es un subconjunto de Y y X no es igual Y, se dice que X es un **subconjunto propio** de Y y se escribe $X \subset Y$. El conjunto de todos los subconjuntos (propios o no) de un conjunto X, denotado por $\mathcal{P}(X)$, se llama el **conjunto potencia** de X.

Ejemplo 2.1.5 ▶

Si $A = \{a, b, c\}$, los miembros de $\mathcal{P}(A)$ son

$$\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}.$$

Todos menos $\{a, b, c\}$ son subconjuntos propios de A. Para este ejemplo,

$$|A| = 3, \quad |\mathcal{P}(A)| = 2^3 = 8.$$ ◀

Damos una demostración utilizando inducción matemática de que el resultado del ejemplo 2.1.5 se cumple en general; esto es, el conjunto potencia de un conjunto con n elementos tiene 2^n elementos.

Teorema 2.1.6

Si $|X| = n$, entonces

$$|\mathcal{P}(X)| = 2^n. \tag{2.1.3}$$

Demostración La prueba se hace por inducción sobre n.

Paso base

Si $n = 0$, X es el conjunto vacío. El único subconjunto del conjunto vacío es el conjunto vacío mismo; así,

$$|\mathcal{P}(X)| = 1 = 2^0 = 2^n.$$

Entonces, (2.1.3) es verdadera para $n = 0$.

Paso inductivo

Suponga que (2.1.3) se cumple para n. Sea X el conjunto con $n + 1$ elementos. Elija $x \in X$. Se afirma que exactamente la mitad de los subconjuntos de X contiene a x, y exactamente la mitad de los subconjuntos de X no contiene a x. Para ver esto, observe que cada subconjunto S de X que contiene a x se puede aparear de manera única con el subconjunto obtenido eliminando x de S (vea la figura 2.1.1). Entonces, justo la mitad de los subconjuntos de X contiene a x y justo la mitad de ellos no contiene a x.

Subconjuntos de X que contienen a	*Subconjuntos de X que no contienen* a
$\{a\}$	$\emptyset$
$\{a, b\}$	$\{b\}$
$\{a, c\}$	$\{c\}$
$\{a, b, c\}$	$\{b, c\}$

Figura 2.1.1 Subconjuntos de $X = \{a, b, c\}$ divididos en dos clases: los que contienen a a y los que no contienen a a. Cada subconjunto en la columna derecha se obtiene al quitar el elemento a del subconjunto correspondiente en la columna izquierda.

Si Y es el conjunto obtenido de X eliminando x, Y tiene n elementos. Por la suposición de inducción, $|\mathcal{P}(X)| = 2^n$. Pero los subconjuntos de Y son precisamente los subconjuntos de X que no contienen a x. A partir del argumento en el párrafo anterior se concluye que

$$|\mathcal{P}(Y)| = \frac{|\mathcal{P}(X)|}{2}.$$

Por lo tanto,

$$|\mathcal{P}(X)| = 2\,|\mathcal{P}(Y)| = 2 \cdot 2^n = 2^{n+1}.$$

Así, (2.1.3) se cumple para $n + 1$ y el paso inductivo queda completo. Por el principio de inducción matemática, (2.1.3) se cumple para toda $n \geq 0$.

Si se tienen dos conjuntos X y Y, existen varias operaciones de conjuntos que implican a X y Y que pueden producir un nuevo conjunto. El conjunto

$$X \cup Y = \{x \mid x \in X \text{ o } x \in Y\}$$

se llama **unión** de X y Y. La unión consiste en todos los elementos que pertenecen a X o a Y (o a ambos).

El conjunto

$$X \cap Y = \{x \mid x \in X \text{ y } x \in Y\}$$

se llama **intersección** de X y Y. La intersección consiste en todos los elementos que pertenecen a X y Y.

El conjunto

$$X - Y = \{x \mid x \in X \text{ y } x \notin Y\}$$

se llama **diferencia** (o **complemento relativo**). La diferencia $X - Y$ consiste en todos los elementos en X que no están en Y.

Ejemplo 2.1.7 ▶

Si $A = \{1, 3, 5\}$ y $B = \{4, 5, 6\}$, entonces

$$A \cup B = \{1, 3, 4, 5, 6\}$$
$$A \cap B = \{5\}$$
$$A - B = \{1, 3\}$$
$$B - A = \{4, 6\}.$$

Observe que $A - B \neq B - A$. ◀

Los conjuntos X y Y son **disjuntos** si $X \cap Y = \emptyset$. Se dice que una colección de conjuntos S es **disjunta por pares** si siempre que X y Y son conjuntos diferentes en S, X y Y son disjuntos.

Ejemplo 2.1.8 ▶

Los conjuntos

$$\{1, 4, 5\} \quad \text{y} \quad \{2, 6\}$$

son disjuntos. La colección de conjuntos

$$\mathcal{S} = \{\{1, 4, 5\}, \{2, 6\}, \{3\}, \{7, 8\}\}$$

son disjuntos por pares. ◀

En ocasiones, se trata con conjuntos donde todos son subconjuntos de un conjunto U. Este conjunto U se llama **conjunto universal** o **universo**. El conjunto U debe estar dado explícitamente o poder inferirse del contexto. Dado un conjunto universal U y un subconjunto X de U, el conjunto $U - X$ se llama **complemento** de X y se escribe $\overline{X}$.

Ejemplo 2.1.9 ▶

Sea $A = \{1, 3, 5\}$. Si U, un conjunto universal, se especifica como $U = \{1, 2, 3, 4, 5\}$, entonces $\overline{A} = \{2, 4\}$. Por otra parte, si un conjunto universal se especifica como $U = \{1, 3, 5, 7, 9\}$, entonces $\overline{A} = \{7, 9\}$. Es evidente que el complemento depende del universo en el que se está trabajando. ◀

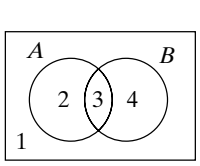


Figura 2.1.2 Diagrama de Venn.

Los **diagramas de Venn** proporcionan una vista pictórica de los conjuntos. En un diagrama de Venn, un rectángulo describe el conjunto universal (vea la figura 2.1.2). Los subconjuntos del conjunto universal se dibujan como círculos. El interior de un círculo representa los elementos de ese conjunto. En la figura 2.1.2 se observan dos conjuntos A y B dentro del conjunto universal U. Los elementos que no están en A ni en B están en la región 1. Los elementos en la región 2 están en A pero no en B. La región 3 representa $A \cap B$, los elementos comunes a A y B. La región 4 comprende los elementos en B pero no en A.

Ejemplo 2.1.10 ▶

Las regiones específicas de los diagramas de Venn se describen por sombreados. El conjunto $A \cup B$ se muestra en la figura 2.1.3, y la figura 2.1.4 representa el conjunto $A - B$. ◀

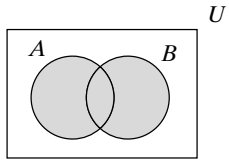


Figura 2.1.3 Un diagrama de Venn de $A \cup B$.

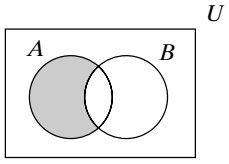


Figura 2.1.4 Un diagrama de Venn de $A - B$.

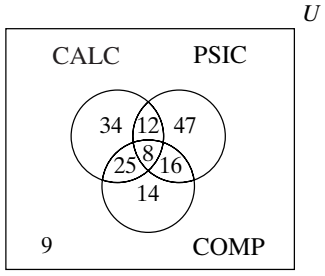


Figura 2.1.5 Un diagrama de Venn de tres conjuntos CALC, PSIC y COMP. Los números muestran cuántos estudiantes pertenecen a la región específica descrita.

Para representar tres conjuntos se usan tres círculos que se traslapan (vea la figura 2.1.5).

Ejemplo 2.1.11 ▶

En un grupo de 165 estudiantes, 8 toman cálculo, psicología y computación; 33 toman cálculo y computación; 20 toman cálculo y psicología; 24 toman psicología y computación; 79 están en cálculo; 83 están en psicología y 63 toman computación. ¿Cuántos estudiantes no toman ninguna de las tres asignaturas?

Sean CALC, PSIC y COMP los conjuntos de estudiantes que toman cálculo, psicología y computación, respectivamente. Sea U el conjunto de los 165 estudiantes (vea la figura 2.1.5). Como 8 estudiantes toman cálculo, psicología y computación, hay un 8 en la región que representa CALC $\cap$ PSIC $\cap$ COMP. De los 33 estudiantes que toman cálculo, 8 toman también psicología; entonces, 25 toman cálculo y computación pero no psicología. Escribimos 25 en la región que representa CALC $\cap$ PSIC $\cap$ $\overline{\text{COMP}}$. De manera similar, escribimos 12 en la región que representa CALC $\cap$ PSIC $\cap$ $\overline{\text{COMP}}$ y 16 en la región correspondiente a $\overline{\text{CALC}} \cap$ PSIC $\cap$ COMP. De 79 estudiantes que toman cálculo, ya se sabe qué hacen 45. Esto deja 34 estudiantes que toman sólo cálculo. Se escribe 34 en la región que representa CALC $\cap$ $\overline{\text{PSIC}} \cap \overline{\text{COMP}}$. De modo parecido, escribimos 47 en la región correspondiente a $\overline{\text{CALC}} \cap$ PSIC $\cap$ $\overline{\text{COMP}}$ y 14 en la que corresponde a $\overline{\text{CALC}} \cap$ PSIC $\cap$ COMP. Ahora se sabe qué hacen 156 estudiantes. Esto deja 9 estudiantes que no toman ninguna de estas tres asignaturas. ◀

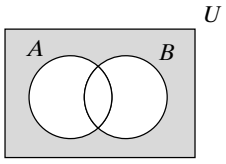


Figura 2.1.6 La región sombreada describe ambos casos $\overline{(A \cup B)}$ y $\overline{A} \cap \overline{B}$; entonces, estos conjuntos son iguales.

Los diagramas de Venn también resultan útiles para visualizar ciertas propiedades de los conjuntos. Por ejemplo, si se dibujan tanto $\overline{(A \cup B)}$ y $\overline{A} \cap \overline{B}$ (vea figura 2.1.6), se observa que estos conjuntos son iguales. Un argumento formal mostraría que para cada x, si $x \in \overline{(A \cup B)}$, entonces $x \in \overline{A} \cap \overline{B}$, y si $x \in \overline{A} \cap \overline{B}$, entonces $x \in \overline{(A \cup B)}$. En el teorema 2.1.12 se enuncian varias propiedades útiles de los conjuntos.

Teorema 2.1.12

Sea U un conjunto universal y sean A, B y C subconjuntos de U. Las siguientes propiedades se cumplen.

a) Leyes asociativas:

$$(A \cup B) \cup C = A \cup (B \cup C), \quad (A \cap B) \cap C = A \cap (B \cap C)$$

b) Leyes conmutativas:

$$A \cup B = B \cup A, \quad A \cap B = B \cap A$$

c) Leyes distributivas:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C), \quad A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

d) Leyes de identidad:

$$A \cup \emptyset = A, \quad A \cap U = A$$

e) Leyes de complemento:

$$A \cup \overline{A} = U, \quad A \cap \overline{A} = \emptyset$$

f) Leyes de idempotencia:

$$A \cup A = A, \quad A \cap A = A$$

g) Leyes de acotación:

$$A \cup U = U, \quad A \cap \emptyset = \emptyset$$

h) Leyes de absorción:

$$A \cup (A \cap B) = A, \quad A \cap (A \cup B) = A$$

i) Leyes de involución:

$$\overline{\overline{A}} = A$$

j) Leyes 0/1

$$\overline{\emptyset} = U, \quad \overline{U} = \emptyset$$

k) Leyes de De Morgan para conjuntos:

$$\overline{(A \cup B)} = \overline{A} \cap \overline{B}, \quad \overline{(A \cap B)} = \overline{A} \cup \overline{B}$$

Demostración Se prueba primero la ley distributiva del inciso *c)* y se dejan los incisos restantes como ejercicios (vea ejercicios 83 al 89).

Debemos probar que para todo x, si $x \in A \cap (B \cup C)$, entonces $x \in (A \cap B) \cup (A \cap C)$, y si $x \in (A \cap B) \cup (A \cap C)$, entonces $x \in A \cap (B \cup C)$.

Suponga que $x \in A \cap (B \cup C)$. Por la definición de intersección, $x \in A$ y $x \in B \cup C$. Por la definición de unión, $x \in B$ o $x \in C$. Si $x \in B$, entonces $x \in A$ y $x \in B$; de manera que $x \in A \cap B$. Por la definición de unión, $x \in (A \cap B) \cup (A \cap C)$. Si $x \in C$, entonces $x \in A$ y $x \in C$; de manera que $x \in A \cap C$. Por la definición de unión, de nuevo $x \in (A \cap B) \cup (A \cap C)$. Por lo tanto, si $x \in A \cap (B \cup C)$, entonces $x \in (A \cap B) \cup (A \cap)$.

Ahora suponga que $x \in (A \cap B) \cup (A \cap C)$. Por la definición de unión, $x \in A \cap B$ o $x \in A \cap C$. Si $x \in A \cap B$, entonces $x \in A$ y $x \in B$. Por la definición de unión, $x \in A$ y $x \in B \cup C$. Por la definición de intersección, $x \in A \cap (B \cup C)$. Si $x \in A \cap C$, entonces $x \in A$ y $x \in C$. Por la definición de unión, $x \in A$ y $x \in B \cup C$. Por la definición de intersección, de nuevo $x \in A \cap (B \cup C)$. Por lo tanto, si $x \in (A \cap B) \cup (A \cap C)$, entonces $x \in A \cap (B \cup C)$.

Se ha demostrado que

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C).$$

Se define la unión de una familia arbitraria de conjuntos, $\mathcal{S}$, como los elementos x que pertenecen al menos a un conjunto X en $\mathcal{S}$. De manera formal,

$$\cup\mathcal{S} = \{x \mid x \in X \text{ para alguna } X \in \mathcal{S}\}.$$

De igual manera, se define la intersección de una familia arbitraria de conjuntos, $\mathcal{S}$, como los elementos x que pertenecen a todo conjunto X en $\mathcal{S}$. De manera formal,

$$\cap\mathcal{S} = \{x \mid x \in X \text{ para toda } X \in \mathcal{S}\}.$$

Si

$$\mathcal{S} = \{A_1, A_2, \ldots, A_n\},$$

escribimos

$$\bigcup\mathcal{S} = \bigcup_{i=1}^{n} A_i, \qquad \bigcap\mathcal{S} = \bigcap_{i=1}^{n} A_i,$$

y si

$$\mathcal{S} = \{A_1, A_2, \ldots\},$$

escribimos

$$\bigcup\mathcal{S} = \bigcup_{i=1}^{\infty} A_i, \qquad \bigcap\mathcal{S} = \bigcap_{i=1}^{\infty} A_i.$$

Ejemplo 2.1.13 ▶

Para $i \geq 1$, defina

$$A_i = \{i, i+1, \ldots\} \qquad \text{y} \qquad \mathcal{S} = \{A_1, A_2, \ldots\}.$$

Entonces

$$\bigcup_{i=1}^{\infty} A_i = \bigcup\mathcal{S} = \{1, 2, \ldots\}, \qquad \bigcap_{i=1}^{\infty} A_i = \bigcap\mathcal{S} = \emptyset. \quad \blacktriangleleft$$

Una partición de un conjunto X divide a X en subconjuntos que no se traslapan. De manera más formal, se dice que una colección $\mathcal{S}$ de subconjuntos no vacíos de X es una **partición** del conjunto X si todo elemento de X pertenece exactamente a un miembro de $\mathcal{S}$. Observe que si $\mathcal{S}$ es una partición de X, $\mathcal{S}$ es disjunta por pares y $\cup\mathcal{S} = X$.

Ejemplo 2.1.14 ▶

Como cada elemento de

$$X = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

está en exactamente un miembro de

$$\mathcal{S} = \{\{1, 4, 5\}, \{2, 6\}, \{3\}, \{7, 8\}\},$$

S es una partición de X. $\blacktriangleleft$

Al principio de esta sección, se señaló que un conjunto es una colección no ordenada de elementos; es decir, un conjunto se determina por sus elementos y no por un orden particular en el que se listan los elementos. Sin embargo, algunas veces se desea tomar en cuenta el orden. Un **par ordenado** de elementos, escrito (a, b), se considera diferente del par ordenado (b, a), a menos que, por supuesto $a = b$. Dicho de otra manera, $(a, b) = (c, d)$ justo cuando $a = c$ y $b = d$. Si X y Y son conjuntos, sea $X \times Y$ el conjunto de todos los pares ordenados (x, y) donde $x \in X$ y $y \in Y$. $X \times Y$ se conoce como **producto cartesiano** de X y Y.

Ejemplo 2.1.15 ▶

Si $X = \{1, 2, 3\}$ y $Y = \{a, b\}$, entonces

$$X \times Y = \{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}$$

$$Y \times X = \{(a, 1), (b, 1), (a, 2), (b, 2), (a, 3), (b, 3)\}$$

$$X \times X = \{(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)\}$$

$$Y \times Y = \{(a, a), (a, b), (b, a), (b, b)\}.$$ ◀

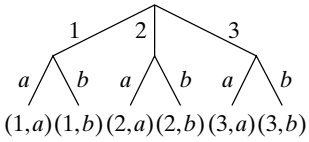


Figura 2.1.7 $|X \times Y| = |X| \cdot |Y|$, donde $X = \{1, 2, 3\}$ y $Y = \{a, b\}$. Existen 3 maneras de elegir el primer miembro del par ordenado (mostradas arriba en el diagrama) y, para cada elección, hay 2 maneras de seleccionar el segundo miembro del par ordenado (mostradas en la parte media del diagrama). Como se tienen 3 grupos de 2, hay $3 \cdot 2 = 6$ elementos en $X \times Y$ (etiquetados abajo en el diagrama).

El ejemplo 2.1.15 muestra que, en general, $X \times Y \neq Y \times X$.

Note que en el ejemplo 2.1.15, $|X \times Y| = |X| \cdot |Y|$ (ambos son iguales a 6). La razón es que hay 3 maneras de elegir un elemento de X para el primer miembro del par ordenado, hay 2 maneras de elegir un elemento de Y para el segundo miembro del par ordenado, y $3 \cdot 2 = 6$ (vea la figura 2.1.7). El argumento anterior se cumple para conjuntos finitos arbitrarios X y Y; siempre es verdad que $|X| \times |Y| = |X| \cdot |Y|$.

Ejemplo 2.1.16 ▶

En un restaurante se sirven cuatro entremeses,

$$r = \text{costillas}, \quad n = \text{nachos}, \quad s = \text{camarones}, \quad f = \text{queso fundido}$$

y tres entradas,

$$c = \text{pollo}, \quad b = \text{res}, \quad t = \text{trucha}.$$

Sea $A = \{r, n, s, f\}$ y $E = \{c, b, t\}$, el producto cartesiano $A \times E$ da 12 cenas posibles consistentes en un entremés y una entrada. ◀

Las listas ordenadas no están restringidas a dos elementos. Una ***n*-eada**, escrito $(a_1, a_2, \ldots, a_n)$ toma en cuenta el orden; es decir,

$$(a_1, a_2, \ldots, a_n) = (b_1, b_2, \ldots, b_n)$$

precisamente cuando

$$a_1 = b_1, a_2 = b_2, \ldots, a_n = b_n.$$

El producto cartesiano de los conjuntos $X_1, X_2, \ldots, X_n$ se define como el conjunto de todas las n-eadas $(x_1, x_2, \ldots, x_n)$ donde $x_i \in X_i$ para $i = 1, \ldots, n$; se denota por $X_1 \times X_2 \times \cdots \times X_n$.

Ejemplo 2.1.17 ▶

Si

$$X = \{1, 2\}, \quad Y = \{a, b\}, \quad Z = \{\alpha, \beta\},$$

entonces

$$X \times Y \times Z = \{(1, a, \alpha), (1, a, \beta), (1, b, \alpha), (1, b, \beta), (2, a, \alpha), (2, a, \beta), (2, b, \alpha), (2, b, \beta)\}.$$ ◀

Observe que en el ejemplo 2.1.17, $|X \times Y \times Z| = |X| \cdot |Y| \cdot |Z|$. En general,

$$|X_1 \times X_2 \times \cdots \times X_n| = |X_1| \cdot |X_2| \cdots |X_n|.$$

Esta última afirmación se puede probar por inducción sobre el número n de conjuntos (vea el ejercicio 95).

Ejemplo 2.1.18 ▶

Si A es el conjunto de entremeses, E es el conjunto de entradas y D el conjunto de postres, el producto cartesiano $A \times E \times D$ da todas las cenas posibles de un entremés, una entrada y un postre. ◀

Sugerencias para resolver problemas

Para probar que dos conjuntos A y B son iguales, escrito $A = B$, demuestre que para toda x, si $x \in A$, entonces $x \in B$, y si $x \in B$, entonces $x \in A$.

Para probar que A es un subconjunto de B, escrito $A \subseteq B$, demuestre que para toda x, si $x \in A$, entonces $x \in B$. Observe que si A es un subconjunto de B, es posible que $A = B$.

Para probar que A es un subconjunto propio de B, escrito $A \subset B$, demuestre que A es un subconjunto de B, como se describió en el párrafo anterior, y que $A \neq B$, es decir, que existe un elemento $x \in B$, pero $x \notin B$.

Para visualizar las relaciones entre conjuntos, use un diagrama de Venn. Un diagrama de Venn ayuda a determinar si una afirmación acerca de conjuntos en verdadera o falsa.

Un conjunto de elementos está determinado por sus miembros; el orden es irrelevante. Por otro lado, los pares y n-eadas ordenadas toman en cuenta el orden.

Sección de ejercicios de repaso

1. Defina un conjunto.

2. ¿Cuál es la notación de conjuntos?

3. Si X es un conjunto finito, ¿qué es $|X|$?

4. ¿Cómo se denota *x es un elemento del conjunto X*?

5. ¿Cómo se denota *x no es un elemento del conjunto X*?

6. ¿Cómo se denota el conjunto vacío?

7. Defina $X = Y$, donde X y Y son conjuntos.

8. Defina $X \subseteq Y$, donde X y Y son conjuntos.

9. Defina *X es un subconjunto propio de Y*.

10. ¿Cuál es el conjunto potencia de X? ¿Cómo se denota?

11. Si X tiene n elementos, ¿cuántos elementos tiene el conjunto potencia de X?

12. Defina *X unión Y*. ¿Cómo se denota la unión?

13. Si S es una familia de conjuntos, ¿cómo se define la unión de S? ¿Cómo se denota la unión?

14. Defina *X intersección Y*. ¿Cómo se denota la intersección de X y Y?

15. Si S es una familia de conjuntos, ¿cómo se define la intersección de S? ¿Cómo se denota la intersección?

16. Defina *X y Y son conjuntos disjuntos*.

17. ¿Qué es una familia de conjuntos disjuntos por pares?

18. Defina la *diferencia* de conjuntos X y Y. ¿Cómo se denota la diferencia?

19. ¿Qué es un conjunto universal?

20. ¿Qué es el complemento del conjunto X? ¿Cómo se denota?

21. qué es un diagrama de Venn?

22. Dibuje un diagrama de Venn de tres conjuntos e identifique el conjunto representado por cada región.

23. Enuncie las leyes asociativas para conjuntos.

24. Enuncie las leyes conmutativas para conjuntos.

25. Enuncie las leyes distributivas para conjuntos.

26. Enuncie las leyes de identidad para conjuntos.

27. Enuncie las leyes de complementos para conjuntos.

28. Enuncie las leyes de idempotencia para conjuntos.

29. Enuncie las leyes de acotación para conjuntos.

30. Enuncie las leyes de absorción para conjuntos.

31. Enuncie las leyes de involución para conjuntos.

32. Enuncie las leyes 0/1 para conjuntos.

33. Enuncie las leyes de De Morgan para conjuntos.

34. ¿Qué es una partición del conjunto X?

35. Defina el *producto cartesiano* de los conjuntos X y Y. ¿Cómo se denota este producto cartesiano?

36. Defina el *producto cartesiano* de los conjuntos $X_1, X_2, \ldots, X_n$. ¿Cómo se denota el producto cartesiano?

Ejercicios

En los ejercicios 1 al 16, establezca el universo como el conjunto $U = \{1, 2, 3, \ldots, 10\}$. *Sea* $A = \{1, 4, 7, 10\}$, $B = \{1, 2, 3, 4, 5\}$ *y* $C = \{2, 4, 6, 8\}$. *Liste los elementos de cada conjunto.*

1. $A \cup B$

2. $B \cap C$

3. $A - B$

4. $B - A$

5. $\overline{A}$

6. $U - C$

7. $\overline{U}$

8. $A \cup \emptyset$

9. $B \cap \emptyset$

10. $A \cup U$

11. $B \cap U$

12. $A \cap (B \cup C)$

13. $\overline{B} \cap (C - A)$

14. $(A \cap B) - C$

15. $\overline{A \cap B} \cup C$

16. $(A \cup B) - (C - B)$

En los ejercicios 17 al 24, dibuje un diagrama de Venn y sombree el conjunto indicado.

17. $A \cap \overline{B}$

18. $\overline{A} - B$

19. $B \cup (B - A)$

20. $(A \cup B) - B$

21. $B \cap \overline{(C \cup A)}$

22. $(\overline{A} \cup B) \cap (\overline{C} - A)$

23. $((C \cap A) - \overline{(B - A)}) \cap C$

24. $(B - \overline{C}) \cup ((B - \overline{A}) \cap (C \cup B))$

Los ejercicios 25 al 29 se refieren a un grupo de 191 estudiantes, de los cuales 10 toman francés, negocios y música; 36 toman francés y negocios; 20 están en francés y música; 18 en negocios y música; 65 en francés; 76 en negocios y 63 toman música.

25. ¿Cuántos toman francés y música pero no negocios?

26. ¿Cuántos toman negocios pero no francés ni música?

27. ¿Cuántos toman francés o negocios (o ambos)?

28. ¿Cuántos toman música o francés (o ambos) pero no negocios?

29. ¿Cuántos no toman ninguna de las tres materias?

30. Una encuesta sobre televisión de 151 personas encontró que 68 ven "La ley y el desorden"; 61 ven "Ala este"; 52 ven "Los tenores"; 16 ven tanto "La ley y el desorden" como "Ala este"; 25 ven "La ley y el desorden" y "Los tenores"; 19 ven "Ala este" y "Los tenores"; y 26 no ven ninguno de estos programas. ¿Cuántas personas ven los tres programas?

31. En un grupo de estudiantes, cada uno toma un curso de matemáticas o computación o ambos. Un quinto de los que toman matemáticas también toman computación y un octavo de los que toman computación también están en el curso de matemáticas. ¿Está más de un tercio de los estudiantes tomando el curso de matemáticas?

En los ejercicios 32 al 35, sea $X = \{1, 2\}$ y $Y = \{a, b, c\}$. Liste los elementos de cada conjunto.

32. $X \times Y$

33. $Y \times X$

34. $X \times X$

35. $Y \times Y$

En los ejercicios 36 al 39, sea $X = \{1, 2\}$, $Y = \{a\}$ y $Z = \{\alpha, \beta\}$. Liste los elementos de cada conjunto.

36. $X \times Y \times Z$

37. $X \times Y \times Y$

38. $X \times X \times X$

39. $Y \times X \times Y \times Z$

En los ejercicios 40 al 43, liste todas las particiones del conjunto.

40. $\{1\}$

41. $\{1, 2\}$

42. $\{a, b, c\}$

43. $\{a, b, c, d\}$

En los ejercicios 44 al 47, diga si es verdadero o falso.

44. $\{x\} \subseteq \{x\}$

45. $\{x\} \in \{x\}$

46. $\{x\} \in \{x, \{x\}\}$

47. $\{x\} \subseteq \{x, \{x\}\}$

En los ejercicios 48 al 52, determine si cada par de conjuntos es igual.

48. $\{1, 2, 3\}, \{1, 3, 2\}$

49. $\{1, 2, 2, 3\}, \{1, 2, 3\}$

50. $\{1, 1, 3\}, \{3, 3, 1\}$

51. $\{x \mid x^2 + x = 2\}, \{1, -2\}$

52. $\{x \mid x$ es un número real y $0 < x \leq 2\}, \{1, 2\}$

53. Liste los miembros de $\mathcal{P}(\{a, b\})$. ¿Cuáles son los subconjuntos propios de $\{a, b\}$?

54. Liste los miembros de $\mathcal{P}(\{a, b, c, d\})$. ¿Cuáles son los subconjuntos propios de $\{a, b, c, d\}$?

55. Si X tiene 10 elementos, ¿cuántos tiene $\mathcal{P}(X)$? ¿Cuántos subconjuntos propios tiene X?

56. Si X tiene n miembros, ¿cuántos subconjuntos propios tiene X?

57. Si X y Y son conjuntos no vacíos y $X \times Y = Y \times X$, ¿qué se concluye acerca de X y Y?

En cada uno de los ejercicios 58 al 70, si la afirmación es verdadera, pruébela; de otra manera, dé un contraejemplo. Los conjuntos X, Y y Z son subconjuntos de un conjunto universal. Suponga que el universo para los productos cartesianos es $U \times U$.

58. Para todos los conjuntos X y Y, x es un subconjunto de Y o Y es un subconjunto de X.

59. $X \cap (Y - Z) = (X \cap Y) - (X \cap Z)$ para todos los conjuntos X, Y y Z.

60. $(X - Y) \cap (Y - X) = \emptyset$ para todos los conjuntos X y Y.

61. $X - (Y \cup Z) = (X - Y) \cup Z$ para todos los conjuntos X, Y y Z.

62. $\overline{X - Y} = \overline{Y - X}$ para todos los conjuntos X y Y.

63. $\overline{X \cap Y} \subseteq X$ para todos los conjuntos X y Y.

64. $(X \cap Y) \cup (Y - X) = X$ para todos los conjuntos X y Y.

65. $X \times (Y \cup Z) = (X \times Y) \cup (X \times Z)$ para todos los conjuntos X, Y y Z.

66. $\overline{X \times Y} = \overline{X} \times \overline{Y}$ para todos los conjuntos X y Y.

67. $X \times (Y - Z) = (X \times Y) - (X \times Z)$ para todos los conjuntos X, Y y Z.

68. $X - (Y \times Z) = (X - Y) \times (X - Z)$ para todos los conjuntos X, Y y Z.

69. $X \cap (Y \times Z) = (X \cap Y) \times (X \cap Z)$ para todos los conjuntos X, Y y Z.

70. $X \times \emptyset = \emptyset$ para todo conjunto X.

En los ejercicios 71 al 74, ¿qué relación debe cumplirse entre los conjuntos A y B para que la condición enunciada sea verdadera?

71. $A \cap B = A$

72. $A \cup B = A$

73. $\overline{A} \cap U = \emptyset$

74. $\overline{A \cap B} = \overline{B}$

La diferencia simétrica de dos conjuntos A y B es el conjunto

$$A \triangle B = (A \cup B) - (A \cap B).$$

75. Si $A = \{1, 2, 3\}$ y $B = \{2, 3, 4, 5\}$, encuentre $A \triangle B$.

76. Describa la diferencia simétrica de los conjuntos A y B en palabras.

77. A partir de un universo U, describa $A \triangle A$, $A \triangle \overline{A}$, $U \triangle A$ y $\emptyset \triangle A$.

78. Pruebe o desapruebe: Si A, B y C son conjuntos que satisfacen $A \triangle C = B \triangle C$, entonces $A = B$.

79. Pruebe que

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

80. Encuentre una fórmula para $|A \cup B \cup C|$ similar a la fórmula del ejercicio 79. Demuestre que su fórmula se cumple para todos los conjuntos A, B y C.

81. Sea C un círculo y sea $\mathcal{D}$ el conjunto de todos los diámetros de C. ¿Qué es $\cap\mathcal{D}$? (Aquí "diámetros" significa un segmento de recta que pasa por el centro y tiene sus puntos terminales en la circunferencia del círculo).

82. Sea P el conjunto de enteros mayores que 1. Para $i \geq 2$, defina

$$X_i = \{ik \mid k \geq 2, k \in P\}.$$

Describa $P - \bigcup_{i=2}^{\infty} X_i$.

83. Pruebe las leyes asociativas para conjuntos [Teorema 2.1.12, inciso *a)*].

84. Pruebe las leyes conmutativas para conjuntos [teorema 2.1.12, inciso *b)*].

85. Pruebe la segunda ley distributiva para conjuntos [teorema 2.1.12, inciso *c)*].

86. Pruebe las leyes de identidad para conjuntos [teorema 2.1.12, inciso *d)*].

87. Pruebe las leyes de complementos para conjuntos [teorema 2.1.12, inciso *e)*].

88. Pruebe las leyes de idempotencia para conjuntos [teorema 2.1.12, inciso *f)*].

89. Pruebe las leyes de acotación para conjuntos [teorema 2.1.12, inciso *g)*].

90. Pruebe las leyes de absorción para conjuntos [teorema 2.1.12, inciso *h)*].

91. Pruebe las leyes de involución para conjuntos [teorema 2.1.12, inciso *i)*].

92. Pruebe las leyes 0/1 para conjuntos [teorema 2.1.12, inciso *j)*].

93. Pruebe las leyes de De Morgan para conjuntos [teorema 2.1.12, inciso *k)*].

94. Use la inducción para probar que si $X_1, \ldots, X_n$ y X son conjuntos, entonces

a) $X \cap (X_1 \cup X_2 \cup \cdots \cup X_n) = (X \cap X_1) \cup (X \cap X_2) \cup \cdots \cup (X \cap X_n)$.

b) $\overline{X_1 \cap X_2 \cap \cdots \cap X_n} = \overline{X_1} \cup \overline{X_2} \cup \cdots \cup \overline{X_n}$.

95. Use la inducción para probar que si $X_1, \ldots, X_n$ son conjuntos, entonces

$$|X_1 \times X_2 \times \cdots \times X_n| = |X_1| \cdot |X_2| \cdots |X_n|.$$

96. Pruebe que el número de subconjuntos S de $\{1, 2, \ldots, n\}$, con $|S|$ par es 2^{n-1}, $n \geq 1$.

2.2 ➜ Funciones

Si viajamos durante cierto tiempo a una velocidad constante, sabemos que

$$\text{distancia} = \text{velocidad} \times \text{tiempo}.$$

Entonces, si viajamos a 55 millas por hora durante t horas,

$$D = 55t, \tag{2.2.1}$$

donde t es el tiempo y D es la distancia viajada.

WWW

La ecuación (2.2.1) define una **función**. Una función asigna a cada miembro de un conjunto X exactamente un miembro de un conjunto Y. (Los conjuntos X y Y pueden o no ser el mismo.) La función definida por (2.2.1) asigna a cada número real no negativo t el valor $55t$. Por ejemplo, el número $t = 1$ se asigna al valor 55; el número $t = 3.45$ se asigna al valor 189.75; etcétera. Estas asignaciones se pueden representar como pares ordenados: (1, 55), (3.45, 189.75). Formalmente, se *define* una función como un tipo especial de conjunto de pares ordenados.

Definición 2.2.1 ▶

Sean X y Y dos conjuntos. Una *función* f de X a Y es un subconjunto del producto cartesiano $X \times Y$ que tiene la propiedad de que para cada $x \in X$, existe exactamente una $y \in Y$ con $(x, y) \in f$. En ocasiones denotamos una función f de X a Y como $f\colon X \to Y$.

El conjunto X se llama el *dominio de f*. El conjunto

$$\{y \mid (x, y) \in f\}$$

(que es un subconjunto de Y) se llama el *rango de f*. ◀

Ejemplo 2.2.2 ▶

El dominio de la función definida por (2.2.1) es el conjunto de todos los números reales no negativos. (Se supone que el tiempo está restringido a números reales no negativos). El rango también es igual al conjunto de todos los números reales no negativos. ◀

Ejemplo 2.2.3 ▶

El conjunto

$$f = \{(1, a), (2, b), (3, a)\}$$

es una función de $X = \{1, 2, 3\}$ a $Y = \{a, b, c\}$. A cada elemento de X se asigna un valor único en Y: 1 se asigna al valor único a; 2 se asigna al valor único b; y 3 se asigna al valor único a. Se puede describir la situación como se ve en la figura 2.2.1, donde una flecha de j a x significa que se asigna la letra x al entero j. Un dibujo como el de la figura 2.2.1 se llama **diagrama de flechas**. Para que un diagrama de flechas sea una función, la definición 2.2.1 requiere que haya justo una flecha desde cada elemento del dominio. Observe que la figura 2.2.1 tiene esta propiedad.

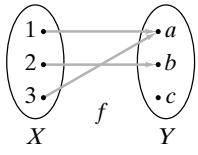


Figura 2.2.1 Diagrama de flechas de la función del ejemplo 2.2.3. Hay exactamente una flecha desde cada elemento en X.

La definición 2.2.1 permite volver a usar los elementos de Y. Para la función f, el elemento a en Y se usa dos veces. Más aún, la definición 2.2.1 *no* requiere que todos los

elementos en Y se usen. Ningún elemento de X se asignó al elemento c de Y. El dominio de f es X y el rango de f es $\{a, b\}$. ◀

Ejemplo 2.2.4 ▶

El conjunto

$$\{(1, a), (2, a), (3, b)\} \tag{2.2.2}$$

no es una función de $X = \{1, 2, 3, 4\}$ a $Y = \{a, b, c\}$ porque el elemento 4 en X no está asignado a un elemento en Y. También es claro, a partir del diagrama de flechas (figura 2.2.2), que este conjunto no es una función porque no hay flecha desde el 4. El conjunto (2.2.2) *es* una función de $X' = \{1, 2, 3\}$ a $Y = \{a, b, c\}$. ◀

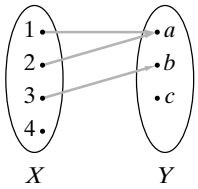


Figura 2.2.2 Diagrama de flechas del conjunto en el ejemplo 2.2.4, que no es una función porque no hay flecha desde el 4.

Ejemplo 2.2.5 ▶

El conjunto

$$\{(1, a), (2, b), (3, c), (1, b)\}$$

no es una función de $X = \{1, 2, 3\}$ a $Y = \{a, b, c\}$ porque 1 no está asignado a un elemento *único* en Y (1 está asignado a los valores a y b). También es claro, a partir del diagrama de flechas (figura 2.2.3), que este conjunto no es una función porque salen dos flechas de 1. ◀

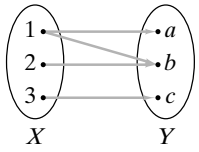


Figura 2.2.3 Diagrama de flechas del conjunto en el ejemplo 2.2.5, que no es una función porque hay dos flechas que salen de 1.

Dada una función f de X a Y, de acuerdo con la definición 2.2.1, para cada elemento x del dominio X, hay exactamente una $y \in Y$ con $(x, y) \in f$. Este valor único y se denota por $f(x)$. En otras palabras, $f(x) = y$ es otra manera de escribir $(x, y) \in f$.

Ejemplo 2.2.6 ▶

Para la función f del ejemplo 2.2.3, se escribe

$$f(1) = a, \qquad f(2) = b, \qquad f(3) = a. \quad ◀$$

El siguiente ejemplo muestra cómo a veces se usa la notación $f(x)$ para definir una función.

Ejemplo 2.2.7 ▶

Sea f una función definida por la regla

$$f(x) = x^2.$$

Por ejemplo,

$$f(2) = 4, \quad f(-3.5) = 12.25, \quad f(0) = 0.$$

Aunque con frecuencia encontramos funciones definidas de esta manera, la definición está incompleta, pues el dominio no está especificado. Si se dice que el dominio es el conjunto de todos los números reales, en la notación de pares ordenados se tendría

$$f = \{(x, x^2) \mid x \text{ es un número real}\}.$$

El rango de f es el conjunto de todos los números reales no negativos. ◀

Ejemplo 2.2.8 ▶

La mayoría de las calculadoras tienen una tecla $1/x$. Si se introduce un número y se oprime la tecla $1/x$, se despliega el recíproco del número introducido (o una aproximación de él). Esta función se define por la regla

$$R(x) = \frac{1}{x}.$$

El dominio es el conjunto de todos los números que se pueden introducir en la calculadora y cuyo recíproco se pueda calcular y desplegar en ella. El rango es el conjunto de todos los recíprocos que se pueden calcular y desplegar. Observe que por la naturaleza de la calculadora, el dominio y el rango son conjuntos finitos. ◀

Otra manera de visualizar una función es dibujar su gráfica. La **gráfica de una función *f*** cuyo dominio y rango son subconjuntos de los números reales se obtiene trazando puntos en el plano que corresponden a los elementos en *f*. El dominio está contenido en el eje horizontal y el rango en el eje vertical.

Ejemplo 2.2.9 ▶

La gráfica de la función $f(x) = x^2$ se ilustra en la figura 2.2.4. ◀

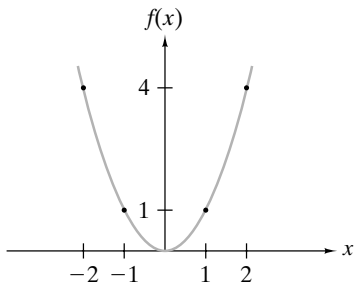


Figura 2.2.4 Gráfica de $f(x) = x^2$.

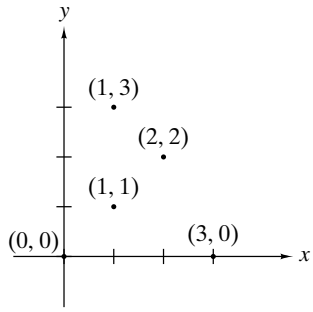


Figura 2.2.5 Conjunto que no es una función. La línea vertical $x = 1$ pasa por dos puntos del conjunto.

Se observa que el conjunto *S* de puntos en el plano define una función precisamente cuando cada línea vertical pasa a lo sumo por un punto de *S*. Si alguna recta vertical contiene dos o más puntos de un conjunto, el punto de dominio no asigna un punto del rango *único* y el conjunto no define una función (vea la figura 2.2.5).

Las funciones que implican el **operador módulo** desempeñan un papel importante en matemáticas y computación.

Definición 2.2.10 ▶

Si *x* es un entero y *y* es un entero positivo, se define *x* mod *y* como el residuo cuando *x* se divide entre *y*. ◀

Ejemplo 2.2.11 ▶

Se tiene

$$6 \bmod 2 = 0, \quad 5 \bmod 1 = 0, \quad 8 \bmod 12 = 8, \quad 199673 \bmod 2 = 1.$$ ◀

Ejemplo 2.2.12 ▶

¿Qué día de la semana será 365 después del miércoles?

Siete días después del miércoles es miércoles de nuevo; 14 días después del miércoles es miércoles otra vez; en general, si *n* es un entero positivo, 7*n* días después del miércoles es miércoles de nuevo. Entonces, necesitamos restar todos los 7 que se puedan de 365 y ver cuántos días quedan, esto es lo mismo que calcular 365 mod 7. Como 365 mod 7 = 1, 365 días después del miércoles será un día más tarde, es decir, jueves. Esto explica por qué, excepto en año bisiesto en que se agrega un día a febrero, el mismo mes y día en años consecutivos se mueve un día de la semana para adelante. ◀

Ejemplo 2.2.13 ▶

Número de libro estándar internacional

Un número de libro estándar internacional (ISBN, siglas en inglés para *International Standard Book Number*) es un código de 10 caracteres separados por guiones, como 0-8065-

0959-7. Un ISBN tiene cuatro partes: un código de grupo, un código de editor, un código que identifica al libro de manera única entre otros publicados por ese editor y un carácter de verificación para validar el ISBN.

Para el ISBN 0-8065-0959-7, el código de grupo es 0, que identifica al libro como uno de un país de habla inglesa. El código del editor 8065 identifica que el libro fue publicado por Citadel Press. El código 0959 identifica de manera única al libro entre los publicados por Citadel Press (en este caso, Brode: *Woody Allen: His Films and Career*). El carácter de verificación es s mod 11, donde s es la suma del primer dígito más dos veces el segundo dígito más tres veces el tercer dígito, . . . , más nueve veces el noveno dígito. Si este valor es 10, el carácter de verificación es X. Por ejemplo, la suma s para el ISBN 0-8065-0959-7 es

$$s = 0 + 2 \cdot 8 + 3 \cdot 0 + 4 \cdot 6 + 5 \cdot 5 + 6 \cdot 0 + 7 \cdot 9 + 8 \cdot 5 + 9 \cdot 9 = 249.$$

Entonces el carácter de verificación es 249 mod 11 = 7. ◀

Ejemplo 2.2.14 ▶

Función de dispersión (hashing)

Suponga que se tienen celdas en la memoria de una computadora indexadas de 0 a 10 (vea la figura 2.2.6). Deseamos guardar y recuperar enteros no negativos arbitrarios en estas celdas. Uno de los enfoques es usar una **función de dispersión**. Una función de dispersión toma un dato que debe guardarse (o recobrarse), calcula la primera opción para la ubicación del dato. Por ejemplo, para nuestro problema, guardar o recuperar el número n, podemos tomar como primera opción para la localización, n mod 11. Nuestra función de dispersión se convierte en

WWW

$$h(n) = n \bmod 11.$$

La figura 2.2.6 muestra el resultado de almacenar 15, 558, 32, 132, 102 y 5, en este orden, en celdas que originalmente estaban vacías.

Ahora suponga que queremos guardar 257. Como $h(257) = 4{,}257$ debe guardarse en el lugar 4; sin embargo, esta posición está ocupada. En tal caso, se dice que ocurre una **colisión**. De manera más precisa, una colisión ocurre para una función de dispersión H si $H(x) = H(y)$, pero $x \neq y$. Para manejar colisiones se requiere una **política para solución de colisión**. Una política sencilla es encontrar la celda superior inmediata disponible para contener ese dato (donde se supone que 0 sigue a 10, para este ejemplo). Si se usa esta política de solución de colisión, 257 se guardaría en el lugar 6 (vea la figura 2.2.6).

Si se quiere localizar un valor guardado n, se calcula $m = h(n)$ y se comienza a buscar en el lugar m. Si n no está en esa posición, se busca en la siguiente posición más alta (de nuevo 0 sigue a 10); si n no está en esta posición, se procede a la siguiente posición más alta y así sucesivamente. Si se encuentra una celda vacía o se regresa a la posición original, se concluye que n no está presente; de otra manera se obtiene la posición de n.

Si las colisiones son poco frecuentes y si cuando ocurre una se resuelve con rapidez, entonces la función de dispersión proporciona un método muy rápido para almacenar y recuperar datos. Como ejemplo, con frecuencia los datos personales se almacenan y recuperan mediante una función de dispersión sobre los números de identificación de los empleados.

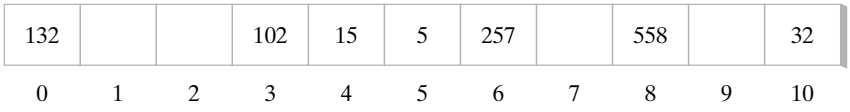


Figura 2.2.6 Celdas de la memoria de una computadora. ◀

Ejemplo 2.2.15 ▶

Números seudoaleatorios

Las computadoras muchas veces se usan para simular el comportamiento aleatorio. Un programa para un juego tal vez simule el lanzamiento de dados, un programa para el servicio a clientes quizá simule la llegada de los clientes a un banco. Este tipo de programas generan números que parecen aleatorios y se llaman **números seudoaleatorios**. Por ejemplo, el programa que lanza dados necesitaría pares de números seudoaleatorios, cada uno entre 1 y 6, para simular el resultado de los dados. Los números seudoaleatorios no son verdaderamente aleatorios; si se conoce el programa que genera los números, es posible predecir qué números se obtendrán.

El método que en general se usa para generar números seudoaleatorios se llama **método congruencial lineal**. Este método requiere cuatro enteros: el módulo m, el multiplicador a, el incremento c y una semilla s que satisfacen

$$2 \le a < m, \qquad 0 \le c < m, \qquad 0 \le s < m.$$

Después se hace $x_0 = s$. La sucesión seudoaleatoria generada, x_1, x_2, ..., está determinada por la fórmula

$$x_n = (ax_{n-1} + c) \bmod m.$$

La fórmula calcula el siguiente número seudoaleatorio usando su predecesor inmediato. Por ejemplo, si

$$m = 11, \quad a = 7, \quad c = 5, \quad s = 3,$$

entonces

$$x_1 = (ax_0 + c) \bmod m = (7 \cdot 3 + 5) \bmod 11 = 4$$

y

$$x_2 = (ax_1 + c) \bmod m = (7 \cdot 4 + 5) \bmod 11 = 0.$$

Cálculos similares muestran que la sucesión continúa:

$$x_3 = 5,\ x_4 = 7,\ x_5 = 10,\ x_6 = 9,\ x_7 = 2,\ x_8 = 8,\ x_9 = 6,\ x_{10} = 3.$$

Como $x_{10} = 3$, que es el valor de la semilla, ahora la sucesión se repite: 3, 4, 0, 5, 7, ...

Se ha invertido un gran esfuerzo en encontrar buenos valores para el método congruencial lineal. Las simulaciones como las que tienen que ver con el equipo aéreo y la investigación nuclear requieren "buenos" números aleatorios. En la práctica, se usan valores grandes para m y a. Los valores que se utilizan comúnmente son $m = 2^{31} - 1 = 2{,}147{,}483{,}647$, $a = 7^5 = 16{,}807$ y $c = 0$, que generan una sucesión de $2^{31} - 1$ enteros antes de repetir un valor. ◀

En la década de 1990, Daniel Corriveau de Quebec ganó tres juegos seguidos de un juego de lotería en computadora en Montreal, eligiendo cada vez 19 de 20 números correctamente. Las posibilidades en contra de esta hazaña son 6 mil millones a 1. Al principio, los oficiales suspicaces se rehusaron a pagarle. Aunque Corriveau atribuyó su éxito a la teoría del caos, lo que pasó en realidad fue que siempre que cortaban la energía eléctrica, el generador de números aleatorios iniciaba con la misma semilla, y generaba la misma sucesión de números aleatorios. El abatido casino pagó al final a Corriveau los $600,000 que le debían.

Ahora se definirá el **piso** y el **techo** de un número real.

Definición 2.2.16 ▶

El *piso* de x, denotado por $\lfloor x \rfloor$, es el mayor entero menor o igual que x. El *techo* de x, denotado por $\lceil x \rceil$, es el menor entero mayor o igual que x. ◀

Ejemplo 2.2.17 ▶

$$\lfloor 8.3 \rfloor = 8,\ \lceil 9.1 \rceil = 10,\ \lfloor -8.7 \rfloor = -9,\ \lceil -11.3 \rceil = -11,\ \lceil 6 \rceil = 6,\ \lceil -8 \rceil = -8$$

◀

El piso de x "redondea x hacia abajo", mientras que el techo de x "redondea x hacia arriba". Se usarán las funciones piso y techo a lo largo de este libro.

Ejemplo 2.2.18 ▶

La figura 2.2.7 muestra las gráficas de las funciones piso y techo. Un paréntesis cuadrado, [o], indica que el punto debe incluirse en la gráfica; un paréntesis, (o), indica que el punto debe excluirse de la gráfica.

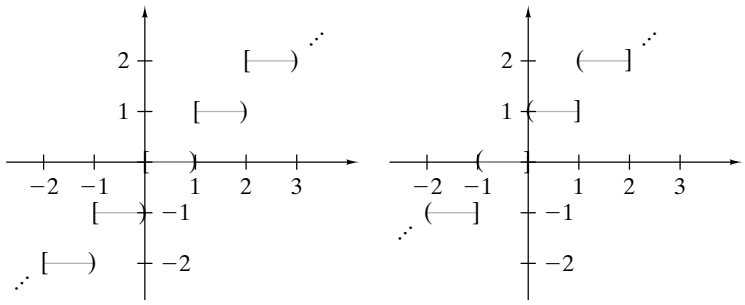


Figura 2.2.7 Gráficas de las funciones piso (gráfica izquierda) y techo (gráfica derecha). ◀

Ejemplo 2.2.19 ▶

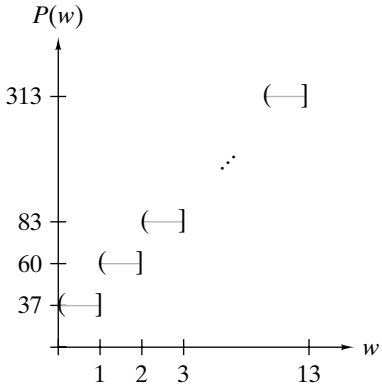


Figura 2.2.8 Gráfica de la función de importe postal $P(w) = 37 + 23\lceil w - 1\rceil$.

En 2005, el importe de un envío por correo en primera clase en Estados Unidos de hasta 13 onzas es de 37 centavos por la primera onza o fracción y 23 centavos por cada onza adicional o fracción. El importe $P(w)$ como función del peso w está determinado por la ecuación

$$P(w) = 37 + 23\lceil w - 1\rceil, \qquad 13 \geq w > 0.$$

La expresión $\lceil w - 1\rceil$ cuenta el número de onzas adicionales más allá de 1, donde una fracción cuenta como una onza adicional. Como ejemplos,

$$P(3.7) = 37 + 23\lceil 3.7 - 1\rceil = 37 + 23\lceil 2.7\rceil = 37 + 23 \cdot 3 = 106,$$
$$P(2) = 37 + 23\lceil 2 - 1\rceil = 37 + 23\lceil 1\rceil = 37 + 23 \cdot 1 = 60.$$

La gráfica de la función P se muestra en la figura 2.2.8. ◀

El teorema de cociente-residuo (Teorema 1.8.5) establece que si d y n son enteros, $d > 0$, existen enteros q (cociente) y r (residuo) que satisfacen

$$n = dq + r \qquad 0 \leq r < d.$$

Al dividir entre d se obtiene

$$\frac{n}{d} = q + \frac{r}{d}.$$

Como $0 \leq r/d < 1$,

$$\left\lfloor \frac{n}{d} \right\rfloor = \left\lfloor q + \frac{r}{d} \right\rfloor = q.$$

Así, se puede calcular el cociente q como $\lfloor n/d \rfloor$. Una vez calculado el cociente q, se calcula el residuo como

$$r = n - dq.$$

Antes se introdujo la notación n mod d para el residuo.

Ejemplo 2.2.20 ▶

Se tiene $36844/2427 = 15.18088\ldots$; entonces el cociente es

$$q = \lfloor 36844/2427 \rfloor = 15.$$

Por lo tanto, el residuo 36844 mod 2427 es

$$r = 36844 - 2427 \cdot 15 = 439.$$

Se tiene

$$n = dq + r \qquad \text{o} \qquad 36844 = 2427 \cdot 15 + 439.$$ ◀

Definición 2.2.21 ▶

Se dice que una función de X a Y es *uno a uno* (o *inyectiva*) si para cada $y \in Y$, existe a lo sumo una $x \in X$ con $f(x) = y$. ◀

Puesto que la cantidad de datos potenciales suele ser mucho más grande que la memoria disponible, es común que las funciones de dispersión no sean uno a uno (vea el ejemplo 2.2.14). En otras palabras, la mayoría de las funciones de dispersión producen colisiones.

Ejemplo 2.2.22 ▶ La función

$$f = \{(1, b), (3, a), (2, c)\}$$

de $X = \{1, 2, 3\}$ a $Y = \{a, b, c, d\}$ es uno a uno. ◀

Ejemplo 2.2.23 ▶ La función

$$f = \{(1, a), (2, b), (3, a)\}$$

no es uno a uno, ya que $f(1) = a = f(3)$. ◀

Ejemplo 2.2.24 ▶ Si X es el conjunto de personas que tiene número de seguro social y se asigna a cada persona $x \in X$ su número de seguro social $\text{SS}(x)$, se obtiene una función uno a uno puesto que diferentes personas siempre tendrán números diferentes asignados. Como esta correspondencia es uno a uno, el gobierno usa los números de seguro social como identificadores. ◀

Ejemplo 2.2.25 ▶ Si una función de X a Y es uno a uno, cada elemento de Y en el diagrama de flechas tendrá a lo más una flecha que llega a él (vea la figura 2.2.9). Si una función no es uno a uno, algún elemento de Y en el diagrama de flechas tendrá dos o más flechas que llegan (vea la figura 2.2.10).

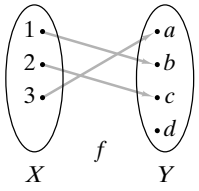


Figura 2.2.9 Función del ejemplo 2.2.22. Esta función es uno a uno porque cada elemento en Y tiene a lo más una flecha que llega a él. Esta función no es sobre Y porque no hay una flecha hacia d.

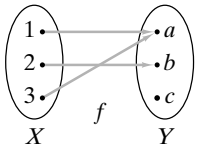


Figura 2.2.10 Una función que no es uno a uno. Esta función no es uno a uno porque a tiene dos flechas que llegan a ella. Esta función no es sobre Y porque no hay flecha que llegue a c. ◀

La condición indicada en la definición 2.2.21 para que una función f de X a Y sea uno a uno es equivalente a: para toda $x_1, x_2 \in X$, si $f(x_1) = f(x_2)$, entonces $x_1 = x_2$. En símbolos,

$$\forall x_1 \forall x_2((f(x_1) = f(x_2)) \rightarrow (x_1 = x_2)).$$

Esta forma de la definición con frecuencia se usa para probar que una función es uno a uno.

Ejemplo 2.2.26 ▶ Pruebe que la función

$$f(n) = 2n + 1$$

del conjunto de enteros positivos al conjunto de enteros positivos es uno a uno.

Debe demostrarse que para todos los enteros positivos n_1 y n_2, si $f(n_1) = f(n_2)$, entonces $n_1 = n_2$. Así, suponga que $f(n_1) = f(n_2)$. Usando la definición de f, esta ecuación

se traduce en

$$2n_1 + 1 = 2n_2 + 1.$$

Restando 1 en ambos lados de la ecuación y luego dividiéndolos entre 2 se obtiene

$$n_1 = n_2.$$

Por lo tanto, f es uno a uno. ◀

Una función f *no* es uno a uno si

$$\forall x_1 \forall x_2((f(x_1) = f(x_2)) \rightarrow (x_1 = x_2))$$

es falsa o, de manera equivalente, su negación es cierta. Usando las leyes generalizadas de De Morgan para lógica (Teorema 1.3.14) y el hecho de que

$$\neg(p \rightarrow q) \equiv p \wedge \neg q,$$

se encuentra que la negación es

$$\begin{aligned}\neg(\forall x_1 \forall x_2((f(x_1) = f(x_2)) \rightarrow (x_1 = x_2))) &\equiv \exists x_1 \neg(\forall x_2((f(x_1) = f(x_2)) \\ &\quad \rightarrow (x_1 = x_2))) \\ &\equiv \exists x_1 \exists x_2 \neg((f(x_1) = f(x_2)) \\ &\quad \rightarrow (x_1 = x_2)) \\ &\equiv \exists x_1 \exists x_2((f(x_1) = f(x_2)) \wedge \neg(x_1 = x_2)) \\ &\equiv \exists x_1 \exists x_2((f(x_1) = f(x_2)) \wedge (x_1 \neq x_2)).\end{aligned}$$

En palabras, una función *no* es uno a uno si existen x_1 y x_2 tales que $f(x_1) = f(x_2)$ y $x_1 \neq x_2$.

Ejemplo 2.2.27 ▶ Pruebe que la función

$$f(n) = 2^n - n^2$$

del conjunto de enteros positivos al conjunto de enteros positivos *no* es uno a uno.

Debemos encontrar enteros positivos n_1 y n_2, $n_1 \neq n_2$, tales que

$$f(n_1) = f(n_2).$$

Por prueba y error, se encuentra que

$$f(2) = f(4).$$

Por lo tanto, f no es uno a uno. ◀

Si el rango de una función f es Y, se dice que la función es **sobre** Y.

Definición 2.2.28 ▶ Si f es una función de X a Y y el rango de f es Y, se dice que f es *sobre* Y (o es una *función sobre* o *función suprayectiva*). ◀

Ejemplo 2.2.29 ▶ La función

$$f = \{(1, a), (2, c), (3, b)\}$$

de $X = \{1, 2, 3\}$ a $Y = \{a, b, c\}$ es uno a uno y sobre Y. ◀

Ejemplo 2.2.30 ▶

La función

$$f = \{(1, b), (3, a), (2, c)\}$$

no es sobre $Y = \{a, b, c, d\}$. Es sobre $\{a, b, c\}$. ◀

Ejemplo 2.2.31 ▶

Si una función de X a Y es sobre, cada elemento de Y en su diagrama de flechas tendrá al menos una flecha que llega a él (figura 2.2.11). Si una función de X a Y no es sobre, algún elemento de Y en el diagrama de flechas no tiene una flecha que apunta hacia él (figuras 2.2.9 y 2.2.10). ◀

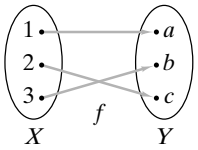


Figura 2.2.11 Función del ejemplo 2.2.29. Esta función es uno a uno porque cada elemento de Y tiene a lo sumo una flecha. La función también es sobre porque cada elemento de Y tiene al menos una flecha que apunta hacia él.

La condición enunciada en la definición 2.2.28 para que una función f de X a Y sea sobre Y es equivalente a: Para toda $y \in Y$, existe una $x \in X$ tal que $f(x) = y$. En símbolos,

$$\forall y \in Y \; \exists x \in X (f(x) = y).$$

Esta forma de definición con frecuencia se usa para probar que una función es sobre.

Ejemplo 2.2.32 ▶

Pruebe que la función

$$f(x) = \frac{1}{x^2}$$

del conjunto X de números reales distintos de cero al conjunto Y de números reales positivos es sobre Y.

Debemos probar que para toda $y \in Y$, existe una $x \in X$ tal que $f(x) = y$. Sustituyendo la fórmula de $f(x)$, esta ecuación se convierte en

$$\frac{1}{x^2} = y.$$

Al despejar x, se encuentra que

$$x = \pm \frac{1}{\sqrt{y}}.$$

Observe que $1/\sqrt{y}$ está definida porque y es un número real positivo. Si tomamos x como la raíz cuadrada positiva

$$x = \frac{1}{\sqrt{y}},$$

entonces $x \in X$. (También pudimos tomar $x = -1/\sqrt{y}$.) Entonces, para cada $y \in Y$, existe x, a saber $x = 1/\sqrt{y}$ tal que

$$f(x) = f(1/\sqrt{y}) = \frac{1}{(1/\sqrt{y})^2} = y.$$

Por lo tanto, f es sobre Y. ◀

Una función f de X a Y *no* es sobre Y si

$$\forall y \in Y \; \exists x \in X (f(x) = y)$$

es falsa o, de manera equivalente, su negación es verdadera. Usando las leyes generalizadas de De Morgan para lógica (Teorema 1.3.14) y el hecho de que

$$\neg(p \to q) \equiv p \wedge \neg q,$$

se encuentra que la negación es

$$\begin{aligned} \neg(\forall y \in Y \; \exists x \in X (f(x) = y)) &\equiv \exists y \in Y \; \neg(\exists x \in X (f(x) = y)) \\ &\equiv \exists y \in Y \; \forall x \in X \neg(f(x) = y) \\ &\equiv \exists y \in Y \; \forall x \in X (f(x) \neq y). \end{aligned}$$

En palabras, una función f de X a Y *no* es sobre Y si existe $y \in Y$ tal que para toda $x \in X$, $f(x) \neq y$.

Ejemplo 2.2.33 ▶ Pruebe que la función

$$f(n) = 2n - 1$$

del conjunto X de enteros positivos al conjunto Y de enteros positivos *no* es sobre Y.

Debe encontrarse un elemento $m \in Y$ tal que para toda $n \in X$, $f(n) \neq m$. Como $f(n)$ es un entero impar para toda n, se puede elegir como y cualquier entero positivo par, por ejemplo, $y = 2$. Entonces, $y \in Y$ y

$$f(n) \neq y$$

para toda $n \in X$. Por lo tanto, f no es sobre Y. ◀

Definición 2.2.34 ▶ Una función que es uno a uno y sobre se llama *biyección*. ◀

Ejemplo 2.2.35 ▶ La función f del ejemplo 2.2.29 es una biyección. ◀

Suponga que f es una función uno a uno, sobre de X a Y. Se puede demostrar (vea el ejercicio 77) que

$$\{(y, x) \mid (x, y) \in f\}$$

es una función uno a uno y sobre de Y a X. Esta nueva función, denotada por f^{-1}, se llama **inversa de *f***.

Ejemplo 2.2.36 ▶ Para la función

$$f = \{(1, a), (2, c), (3, b)\}$$

se tiene

$$f^{-1} = \{(a, 1), (c, 2), (b, 3)\}.$$ ◀

Ejemplo 2.2.37 ▶ A partir del diagrama de flechas para una función f uno a uno, sobre, de X a Y, es posible obtener el diagrama de flechas para f^{-1} simplemente invirtiendo la dirección de cada flecha (vea la figura 2.2.12, que es el diagrama de flechas para f^{-1}, donde f es la función de la figura 2.2.11).

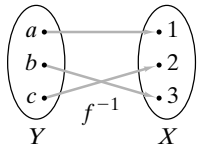


Figura 2.2.12 Inversa de la función en la figura 2.2.11. La inversa se obtiene invirtiendo todas la flechas de la figura 2.2.11.

◀

Ejemplo 2.2.38 ▶ La función

$$f(x) = 2^x$$

es una función uno a uno del conjunto $\mathbf{R}$ de todos los números reales sobre el conjunto $\mathbf{R}^+$ de todos los números reales positivos. Se derivará una fórmula para $f^{-1}(y)$.

Suponga que (y, x) está en f^{-1}; es decir,

$$f^{-1}(y) = \text{x} \tag{2.2.3}$$

Entonces $(x, y) \in f$. Así,

$$y = 2^x.$$

Por la definición de logaritmo.

$$log_2 y = x. \tag{2.2.4}$$

Al combinar (2.2.3) y (2.2.4) se tiene

$$f^{-1}(y) = x = log_2 y.$$

Esto es, para cada $y \in \mathbf{R}^+$, $f^{-1}(y)$ es el logaritmo base 2 de y. La situación se resume diciendo que el inverso de la función exponencial es la función logaritmo. ◀

Sea g una función de X a Y y sea f una función de Y a Z. A partir de $x \in X$, se aplica g para determinar un elemento único $y = g(x) \in Y$. Después se aplica f para determinar un elemento único $z = f(y) = f(g(x)) \in Z$. Esta acción compuesta se llama **composición**.

Definición 2.2.39 ▶ Sea g una función de X a Y y sea f una función de Y a Z. La *composición de f con g*, denotada por $f \circ g$, es la función

$$(f \circ g)(x) = f(g(x))$$

de X a Z. ◀

Ejemplo 2.2.40 ▶ A partir de

$$g = \{(1, a), (2, a), (3, c)\}$$

una función de $X = \{1, 2, 3\}$ a $Y = \{a, b, c\}$, y

$$f = \{(a, y), (b, x), (c, z)\},$$

una función de Y a $Z = \{x, y, z\}$, la función composición de X a Z es la función

$$f \circ g = \{(1, y), (2, y), (3, z)\}.$$ ◀

Ejemplo 2.2.41 ▶ Si se tiene el diagrama de flechas para una función g de X a Y y el diagrama de flechas para una función f de Y a Z, es posible obtener el diagrama de flechas para la composición $f \circ g$ con sólo "seguir las flechas" (vea la figura 2.2.13).

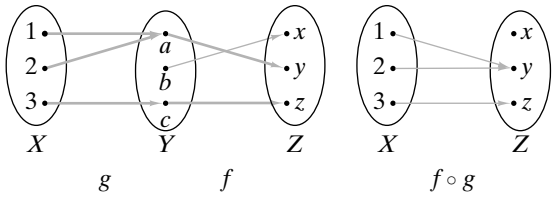


Figura 2.2.13 Composición de las funciones del ejemplo 2.2.40. La composición se obtiene dibujando una flecha de x en X a z en Z siempre que haya flechas de x a alguna y en Y y de y a z.

◀

Ejemplo 2.2.42 ▶ Si $f(x) = \log_3 x$ y $g(x) = x^4$,

$$f(g(x)) = \log_3(x^4), \qquad g(f(x)) = (\log_3 x)^4.$$ ◀

Ejemplo 2.2.43 ▶ En ocasiones, la composición permite descomponer funciones complejas en funciones más sencillas. Por ejemplo, la función

$$f(x) = \sqrt{\operatorname{sen} 2x}$$

se puede descomponer en las funciones

$$g(x) = \sqrt{x}, \qquad h(x) = \operatorname{sen} x, \quad w(x) = 2x.$$

Podemos escribir

$$f(x) = g(h(w(x))).$$

La técnica de descomposición es importante en el cálculo diferencial, ya que existen reglas para diferenciar funciones sencillas como g, h y w, además de reglas acerca de cómo diferenciar la composición de funciones. Al combinar estas reglas, es factible diferenciar funciones más complicadas. ◀

Un **operador binario** sobre un conjunto X asocia un elemento en X con cada par ordenado de elementos en X.

Definición 2.2.44 ▶ Una función de $X \times X$ a X se llama un *operador binario* sobre X. ◀

Ejemplo 2.2.45 ▶ Sea $X = \{1, 2, \ldots\}$. Si se define

$$f(x, y) = x + y,$$

donde $x, y \in X$, entonces f es un operador binario sobre X. ◀

Un **operador unitario** sobre un conjunto X asocia con cada elemento singular de X un elemento en X.

Definición 2.2.46 ▶ Una función de X a X se llama *operador unitario* sobre X. ◀

Ejemplo 2.2.47 ▶ Sea U un conjunto universal. Si se define

$$f(X) = \overline{X},$$

donde $X \in \mathrm{P}(U)$, entonces f es un operador unitario sobre $\mathrm{P}(U)$. ◀

Sugerencias para resolver problemas

Es claro que la clave para resolver problemas que implican funciones es comprender la definición de función. Una función f de X a Y se puede interpretar de muchas maneras. Formalmente, f es un subconjunto de $X \times Y$ que tiene la propiedad de que para toda $x \in X$, existe una $y \in Y$ única tal que $(x, y) \in X \times Y$. De manera informal, se puede pensar en f como en un *mapeo* de elementos de X a Y. El diagrama de flechas resalta esta visión de una función. Para que un diagrama de flechas sea una función, debe haber exactamente una flecha de cada elemento de X a algún elemento de Y.

Una función es un concepto muy general. *Cualquier* subconjunto de $X \times Y$ que tiene la propiedad de que para toda $x \in X$ existe una $y \in Y$ única, tal que $(x, y) \in X \times Y$ es una función. Una función se define listando sus miembros; por ejemplo,

$$\{(a, 1), (b, 3), (c, 2), (d, 1)\}$$

es una función de $\{a, b, c, d\}$ a $\{1, 2, 3\}$. Aquí, parece que no existe una fórmula para la pertenencia; la definición sólo indica qué pares forman la función.

Por otro lado, una función se puede definir por una fórmula. Por ejemplo,

$$\{(n, n + 2) \mid n \text{ es un entero positivo}\}$$

define una función de un conjunto de enteros positivos a un conjunto de enteros positivos. La "fórmula" para el mapeo es "suma 2".

La notación $f(x)$ se utiliza para indicar qué elemento en el rango se asocia con un elemento x en el dominio o para definir una función. Por ejemplo, para la función

$$f = \{(a, 1), (b, 3), (c, 2), (d, 1)\},$$

se puede escribir $f(a) = 1$, $f(b) = 3$, etcétera. Suponiendo que el dominio de definición está constituido por los enteros positivos, la ecuación

$$g(n) = n + 2$$

define la función

$$\{(n, n + 2) \mid n \text{ es un entero positivo}\}$$

del conjunto de enteros positivos al conjunto de enteros positivos.

Para probar que una función f de X a Y es uno a uno, demuestre que para toda $x_1, x_2 \in X$, si $f(x_1) = f(x_2)$, entonces $x_1 = x_2$.

Para probar que una función f de X a Y es sobre, demuestre que para toda $y \in Y$, existe una $x \in X$ tal que $f(x) = y$.

Sección de ejercicios de repaso

1. ¿Qué es una función de X a Y?

2. Explique cómo usar un diagrama de flechas para describir una función.

3. ¿Qué es la gráfica de una función?

4. A partir de un conjunto de puntos en el plano, ¿cómo puede decirse si es una función?

5. ¿Cuál es el valor de x mod y?

6. ¿Qué es una función de dispersión?

7. ¿Qué es una colisión para una función de dispersión?

8. ¿Qué es una política de solución de una colisión?

9. ¿Qué son números seudoaleatorios?

10. Explique cómo funciona el generador congruencial lineal de números aleatorios y dé un ejemplo de un generador congruencial lineal de números aleatorios.

11. ¿Qué es el piso de x? ¿Cómo se denota el piso de x?

12. ¿Qué es el techo de x? ¿Cómo se denota el techo de x?

13. Defina una *función uno a uno*. Dé un ejemplo de una función uno a uno. Explique cómo usar un diagrama de flechas para determinar si una función es uno a uno.

14. Defina una *función sobre*. Dé un ejemplo de una función sobre. Explique cómo usar un diagrama de flechas para determinar si una función es sobre.

15. ¿Qué es una biyección? Dé un ejemplo de una biyección. Explique cómo usar un diagrama de flechas para determinar si una función es una biyección.

16. Defina *función inversa*. Dé un ejemplo de una función y su inversa. A partir del diagrama de flechas de una función, ¿cómo se puede encontrar el diagrama de flechas de la función inversa?

17. Defina *composición de funciones*. ¿Cómo se denota la composición de f y g? Dé un ejemplo de funciones f y g y su composición. A partir de los diagramas de flechas de dos funciones, ¿cómo se puede encontrar el diagrama de flechas de la composición de las funciones?

18. ¿Qué es un operador binario? Proporcione un ejemplo de operador binario.

19. ¿Qué es un operador unitario? Proporcione un ejemplo de un operador unitario.

Ejercicios

Determine si cada conjunto en los ejercicios 1 al 5 es una función de $X = \{1, 2, 3, 4\}$ a $Y = \{a, b, c, d\}$. Si es una función, encuentre su dominio y rango, dibuje el diagrama de flechas y determine si es uno a uno, sobre o ambas. Si es uno a uno y sobre, dé la descripción de la función inversa como un conjunto de pares ordenados. Dibuje el diagrama de flechas y dé el dominio y el rango de la función inversa.

1. $\{(1, a), (2, a), (3, c), (4, b)\}$

2. $\{(1, c), (2, a), (3, b), (4, c), (2, d)\}$

3. $\{(1, c), (2, d), (3, a), (4, b)\}$

4. $\{(1, d), (2, d), (4, a)\}$

5. $\{(1, b), (2, b), (3, b), (4, b)\}$

Dibuje las gráficas de las funciones en los ejercicios 6 al 9. El dominio de cada función es el conjunto de números reales.

6. $f(x) = \lceil x \rceil - \lfloor x \rfloor$

7. $f(x) = x - \lfloor x \rfloor$

8. $f(x) = \lceil x^2 \rceil$

9. $f(x) = \lfloor x^2 - x \rfloor$

Determine si cada función en los ejercicios 10 al 15 es uno a uno. El dominio de cada función es el conjunto de números reales. Si la función no es uno a uno, demuéstrelo. Además, determine si f es sobre el conjunto de números reales. Si f no es sobre, demuéstrelo.

10. $f(x) = 6x - 9$

11. $f(x) = 3x^2 - 3x + 1$

12. $f(x) = \operatorname{sen} x$

13. $f(x) = 2x^3 - 4$

14. $f(x) = 3^x - 2$

15. $f(x) = \dfrac{x}{1 + x^2}$

16. Dé un ejemplo de una función diferente a las presentadas en el libro que sea uno a uno, pero no sobre, y pruebe que su función tiene las propiedades requeridas.

17. Dé un ejemplo de una función diferente a las presentadas en el libro que sea sobre, pero no uno a uno, y pruebe que su función tiene las propiedades requeridas.

18. Dé un ejemplo de una función diferente a las presentadas en el libro que no sea uno a uno ni sobre, y pruebe que su función tiene las propiedades requeridas.

Cada función en los ejercicios 19 al 24 es uno a uno en el dominio especificado X. Si se hace Y = rango de f, se obtiene una biyección de X a Y. Encuentre la función inversa.

19. $f(x) = 4x + 2$, $X =$ conjunto de números reales

20. $f(x) = 3^x$, $X =$ conjunto de números reales

21. $f(x) = 3 \log_2 x$, $X =$ conjunto de números reales positivos

22. $f(x) = 3 + \dfrac{1}{x}$, $X =$ conjunto de números reales distintos de cero

23. $f(x) = 4x^3 - 5$, $X =$ conjunto de números reales

24. $f(x) = 6 + 2^{7x-1}$, $X =$ conjunto de números reales

25. A partir de

$$g = \{(1, b), (2, c), (3, a)\},$$

una función de $X = \{1, 2, 3\}$ a $Y = \{a, b, c, d\}$ y

$$f = \{(a, x), (b, x), (c, z), (d, w)\},$$

una función de Y a $Z = \{\text{w, x, y, z}\}$, escriba $f \circ g$ como un conjunto de pares ordenados y dibuje su diagrama de flechas.

26. Sean f y g funciones de los enteros positivos a los enteros positivos definidos por las ecuaciones

$$f(n) = 2n + 1, \qquad g(n) = 3n - 1.$$

Encuentre las composiciones $f \circ f$, $g \circ g$, $f \circ g$ y $g \circ f$.

27. Sean f y g funciones de los enteros positivos a los enteros positivos definidos por las ecuaciones

$$f(n) = n^2, \qquad g(n) = 2^n.$$

Encuentre las composiciones $f \circ f$, $g \circ g$, $f \circ g$ y $g \circ f$.

28. Sean f y g funciones de los números reales positivos a los números reales positivos definidos por las ecuaciones

$$f(x) = \lfloor 2x \rfloor, \qquad g(x) = x^2.$$

Encuentre las composiciones $f \circ f$, $g \circ g$, $f \circ g$ y $g \circ f$.

En los ejercicios 29 al 34, descomponga la función en funciones más sencillas como en el ejemplo 2.2.43.

29. $f(x) = \log_2(x^2 + 2)$

30. $f(x) = \dfrac{1}{2x^2}$

31. $f(x) = \operatorname{sen} 2x$

32. $f(x) = 2 \operatorname{sen} x$

33. $f(x) = (3 + \operatorname{sen} x)^4$

34. $f(x) = \dfrac{1}{(\cos 6x)^3}$

35. A partir de

$$f = \{(x, x^2) \mid x \in X\},$$

una función de $X = \{-5, -4, \ldots, 4, 5\}$ al conjunto de enteros, escriba f como un conjunto de pares ordenados y dibuje el diagrama de flechas de f. ¿Es f uno a uno o sobre?

36. ¿Cuántas funciones hay de $\{1, 2\}$ a $\{a, b\}$? ¿Cuál es uno a uno? ¿Cuál es sobre?

37. A partir de

$$f = \{(a, b), (b, a), (c, b)\}$$

una función de $X = \{\text{a, b, c}\}$ a X:

a) Escriba $f \circ f$ y $f \circ f \circ f$ como conjuntos de pares ordenados.

b) Defina

$$f^n = f \circ f \circ \cdots \circ f$$

como la composición de n-veces f consigo misma. Escriba f^9 y f^{623} como conjuntos de pares ordenados.

38. Sea f la función de $X = \{0, 1, 2, 3, 4\}$ a X definida por

$$f(x) = 4x \bmod 5.$$

Escriba f como un conjunto de pares ordenados y dibuje el diagrama de flechas de f. ¿Es f uno a uno? ¿Es f sobre?

39. Sea f la función de $X = \{0, 1, 2, 3, 4, 5\}$ a X definida por

$$f(x) = 4x \bmod 6.$$

Escriba f como un conjunto de pares ordenados y dibuje el diagrama de flechas de f. ¿Es f uno a uno? ¿Es f sobre?

40. Compruebe el dígito de verificación del ISBN de este libro.

41. Los códigos de producto universales (UPC, siglas en inglés para *universal product code*) son los códigos de barras que identifican los productos para que la caja pueda registrar el precio de manera automática. Un UPC es un código de 12 dígitos donde el primero caracteriza el tipo de producto (0 identifica un artículo de abarrotes común, 2 es un artículo vendido por peso, 3 es un artículo médico, 4 es un artículo especial, 5 es un cupón y 6 y 7 son artículos no vendidos en tiendas al menudeo). Los 5 dígitos siguientes identifican el fabricante, los 5 dígitos siguientes identifican el producto y el último es un dígito de verificación o control. (Todos los códigos UPC tienen un dígito de verificación. Siempre está presente en el código de barras, pero puede no aparecer en la versión impresa). Por ejemplo, el UPC para un paquete de 10 tostadas Ortega es 0-54400-00800-5. El primer 0 dice que se trata de un artículo de abarrotes. Los 5 dígitos que siguen, 54400 identifican al fabricante Nabisco Foods, y los 5 dígitos siguientes identifican el producto como un paquete de 10 tostadas Ortega.

El dígito de verificación se calcula como sigue. Primero se calcula s, donde s es 3 veces la suma de cada número con posición impar más la suma de cada número con posición par, excepto el dígito de verificación. Este último es el número c, entre 0 y 9 que satisface $(c + S) \bmod 10 = 0$. Para el código del paquete de tostadas se tendría

$$s = 3(0 + 4 + 0 + 0 + 8 + 0) + 5 + 4 + 0 + 0 + 0 = 45.$$

Como $(5 + 45) \bmod 10 = 0$, el dígito de control es 5.

Encuentre el dígito de control para el UPC cuyos 11 primeros dígitos son 3-41280-21414.

Para cada función de dispersión en los ejercicios 42 al 45, muestre cómo se insertarían los datos en el orden indicado para las celdas inicialmente vacías. Use la política de solución de colisión del ejemplo 2.2.14.

42. $h(x) = x \bmod 11$; celdas indexadas 0 a 10; datos: 53, 13, 281, 743, 377, 20, 10, 796.

43. $h(x) = x \bmod 17$; celdas indexadas 0 a 16; datos: 714, 631, 26, 373, 775, 906, 509, 2032, 42, 4, 136, 1028.

44. $h(x) = x^2 \bmod 11$; celdas y datos como en el ejercicio 42.

45. $h(x) = (x^2 + x) \bmod 17$; celdas y datos como en el ejercicio 43.

46. Suponga que se almacenan y recuperan datos como se describe en el ejemplo 2.2.14. ¿Surgirá algún problema si se eliminan datos? Explique su respuesta.

47. Suponga que se almacenan datos como se describe en el ejemplo 2.2.14 y que nunca se almacenan más de 10 datos. ¿Surgirá algún problema al recuperar los datos si se detiene la búsqueda al encontrar una celda vacía? Explique su respuesta.

48. Suponga que se almacenan datos como se describe en el ejemplo 2.2.14 y se recuperan como se describe en el ejercicio 47. ¿Surgirá algún problema si se eliminan datos? Explique su respuesta.

Sea g una función de X a Y y sea f una función de Y a Z. Para cada afirmación en los ejercicios 49 al 56, si es verdadera, demuéstrela; de otra manera, proporcione un contraejemplo.

49. Si g es uno a uno, entonces $f \circ g$ es uno a uno.

50. Si f es sobre, entonces $f \circ g$ es sobre.

51. Si g es sobre, entonces $f \circ g$ es sobre.

52. Si f y g son sobre, entonces $f \circ g$ es sobre.

53. Si f y g son uno a uno y sobre, entonces $f \circ g$ son uno a uno y sobre.

54. Si $f \circ g$ es uno a uno, entonces f es uno a uno.

55. Si $f \circ g$ es uno a uno, entonces g es uno a uno.

56. Si $f \circ g$ es sobre, entonces f es sobre.

Si f es una función de X a Y y $A \subseteq X$ y $B \subseteq Y$, se define

$$f(A) = \{f(x) \mid x \in A\}, \qquad f^{-1}(B) = \{x \in X \mid f(x) \in B\}.$$

Llamamos a $f^{-1}(B)$ la imagen inversa de B bajo f.

57. Sea

$$g = \{(1, a), (2, c), (3, c)\}$$

una función de $X = \{1, 2, 3\}$ a $Y = \{a, b, c, d\}$. Sea $S = \{1\}$, $T = \{1,3\}$, $U = \{a\}$ y $V = \{a, c\}$. Encuentre $g(S)$, $g(T)$, $g^{-1}(U)$ y $g^{-1}(V)$.

★**58.** Sea f una función de X a Y. Pruebe que f es uno a uno si y sólo si

$$f(A \cap B) = f(A) \cap f(B)$$

para todos los subconjuntos A y B de X. [Cuando S es un conjunto, se define $f(S) = \{f(x) \mid x \in S\}$].

★**59.** Sea f una función de X a Y. Pruebe que f es uno a uno si y sólo si siempre que g es una función uno a uno de cualquier conjunto A a X, $f \circ g$ es uno a uno.

★**60.** Sea f una función de X a Y. Demuestre que f es sobre Y si y sólo si siempre que g es una función de Y sobre cualquier conjunto Z, $g \circ f$ es sobre Z.

61. Sea f una función de X sobre Y. Sea

$$\mathcal{S} = \{f^{-1}(\{y\}) \mid y \in Y\}.$$

Demuestre que S es una partición de X.

Los ejercicios 62 al 68 usan las siguientes definiciones. Sea U un conjunto universal y sea $X \subseteq U$. Defina

$$C_X(x) = \begin{cases} 1 & si\ x \in X \\ 0 & si\ x \notin X. \end{cases}$$

C_x se llama función característica de X (en U). (El siguiente rincón de solución de problemas le ayudará a comprender los siguientes ejercicios).

62. Pruebe que $C_{X \cap Y}(x) = C_X(x)C_Y(x)$ para toda $x \in U$.

63. Pruebe que $C_{X \cup Y}(x) = C_X(x) + C_Y(x) - C_X(x)C_Y(x)$ para toda $x \in U$.

64. Pruebe que $C_{\overline{X}}(x) = 1 - C_X(x)$ para toda $x \in U$.

65. Pruebe que $C_{X-Y}(x) = C_X(x)[1 - C_Y(x)]$ para toda $x \in U$.

66. Pruebe que si $X \subseteq Y$, entonces $C_X(x) \leq C_Y(x)$ para toda $x \in U$.

67. Encuentre una fórmula para $C_{X \triangle Y}$. ($X \triangle Y$ es la diferencia simétrica de X y Y. La definición se da en el ejercicio 75, sección 2.1.)

68. Pruebe que la función f de $\mathcal{P}(U)$ al conjunto de funciones características en U definido por

$$f(X) = C_X$$

es uno a uno y sobre.

69. Sean X y Y conjuntos. Pruebe que existe una función uno a uno de X a Y si y sólo si existe una función de Y a X.

Un operador binario f en un conjunto X es conmutativo si $f(x, y) = f(y, x)$ para toda $x, y \in X$. En los ejercicios 70 al 74, establezca si la función f es un operador binario en el conjunto X. Si f no es un operador binario, diga por qué. Indique si se cumple que cada operador binario es conmutativo.

70. $f(x, y) = x + y, \quad X = \{1, 2, \ldots\}$

71. $f(x, y) = x - y, \quad X = \{1, 2, \ldots\}$

72. $f(x, y) = x \cup y, \quad X = \mathcal{P}(\{1, 2, 3, 4\})$

73. $f(x, y) = x/y, \quad X = \{0, 1, 2, \ldots\}$

74. $f(x, y) = x^2 + y^2 - xy, \quad X = \{1, 2, \ldots\}$

En los ejercicios 75 y 76, dé un ejemplo de un operador unitario [diferente de $f(x) = x$, para toda x] sobre el conjunto dado.

75. $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$

76. El conjunto de todos los subconjuntos finitos de $\{1, 2, 3, \ldots\}$

77. Pruebe que si f es una función uno a uno y sobre de X a Y, entonces

$$\{(y, x) \mid (x, y) \in f\}$$

es una función uno a uno y sobre de Y a X.

En los ejercicios 78 al 80, si la afirmación es verdadera para todos los números reales, pruébela; de otra manera dé un contraejemplo.

78. $\lceil x + 3 \rceil = \lceil x \rceil + 3$

79. $\lceil x + y \rceil = \lceil x \rceil + \lceil y \rceil$

80. $\lfloor x + y \rfloor = \lfloor x \rfloor + \lceil y \rceil$

81. Pruebe que si n es un entero impar,

$$\left\lfloor \frac{n^2}{4} \right\rfloor = \left(\frac{n-1}{2} \right) \left(\frac{n+1}{2} \right).$$

82. Pruebe que si n es un entero impar,

$$\left\lceil \frac{n^2}{4} \right\rceil = \frac{n^2+3}{4}.$$

83. Encuentre un valor de x para el cual $\lceil 2x \rceil = 2\lceil x \rceil - 1$..

84. Pruebe que $2\lceil x \rceil - 1 \leq \lceil 2x \rceil \leq 2\lceil x \rceil$ para todo número real x.

Los meses con viernes 13 en el año x se encuentran en el renglón

$$y = \left(x + \left\lfloor \frac{x-1}{4} \right\rfloor - \left\lfloor \frac{x-1}{100} \right\rfloor + \left\lfloor \frac{x-1}{400} \right\rfloor \right) \bmod 7$$

de la columna adecuada:

y	***Año no bisiesto***	***Año bisiesto***
0	Enero, Octubre	Enero, Abril, Julio
1	Abril, Julio	Septiembre, Diciembre
2	Septiembre, Diciembre	Junio
3	Junio	Marzo, Noviembre
4	Febrero, Marzo, Noviembre	Febrero, Agosto
5	Agosto	Mayo
6	Mayo	Octubre

85. Encuentre los meses con viernes 13 en 1945.

86. Encuentre los meses con viernes 13 en el presente año.

87. Encuentre los meses con viernes 13 en el año 2010.

Rincón de solución de problemas

Funciones

Problema

Sea U un conjunto universal y sea $X \subseteq U$. Defina

$$C_X(x) = \begin{array}{l} 1 \text{ si } x \in X \\ 0 \text{ si } x \notin X. \end{array}$$

[C_X se llama *función característica* de X (en U)]. Suponga que X y Y son subconjuntos arbitrarios del conjunto universal U. Demuestre que $C_{X\ Y}(x) = C_X(x) + C_Y(x)$ para toda $x \in U$ si y sólo si $X \cap Y = \emptyset$.

Cómo atacar el problema

Primero debemos aclarar lo que debe hacerse. Como la afirmación es de la forma p si y sólo si q, se tienen dos tareas: 1) probar que si p entonces q, 2) probar que si q entonces p. Es buena idea escribir justo lo que debe demostrarse.

Si $C_{X\cup Y}(x) = C_X(x) + C_Y(x)$ para toda $x \in U$,
entonces $X \cap Y = \emptyset$.

Si $X \cap Y = \emptyset$, entonces $C_{X\cup Y}(x) = C_X(x) + C_Y(x)$
para toda $x \in U$.

Considere la primera afirmación en la que se supone que $C_{X\cup Y}(x) = C_X(x) + C_Y(x)$ para toda $x \in U$ y demuestre que $X \cap Y = \emptyset$. ¿Cómo se demuestra que un conjunto, $X \cap Y$ en este caso, es el conjunto vacío? Se debe demostrar que $X \cap Y$ no tiene elementos. ¿Cómo se hace eso? Existen varias posibilidades, pero viene a la mente otra pregunta. ¿Qué pasa si $X \cap Y$ tiene un elemento? Esto sugiere que primero se puede probar la primera afirmación por contradicción o probar su contrapositiva. Si se hace

p: $C_{X\cup Y}(x) = C_X(x) + C_Y(x)$ para toda $x \in U$
q: $X \cap Y = \emptyset$,

la contrapositiva es $\neg q \rightarrow \neg p$. Ahora la negación de q es

$$\neg q\text{: } X \cap Y \neq \emptyset,$$

y usando la ley de De Morgan (a grandes rasgos, negar $\forall$ da como resultado $\exists$), la negación de p es

$$\neg p\text{: } C_{X\cup Y}(x) \neq C_X(x) + C_Y(x) \text{ para al menos una } x \in U.$$

Así, la contrapositiva es

Si $X \cap Y \neq \emptyset$, entonces $C_{X\cup Y}(x) \neq C_X(x) + C_Y(x)$
para al menos una $x \in U$.

Para la segunda afirmación, se supone que $X \cap Y = \emptyset$ y se prueba que $C_{X\cup Y}(x) = C_X(x) + C_Y(x)$ para toda $x \in U$. Presumiblemente, se puede usar la definición de C_X para calcular ambos lados de la ecuación para toda $x \in U$ y se verifica que los dos lados sean iguales. La definición de C_X sugiere que se use la prueba por casos: $x \in X \cup Y$ (cuando $C_{X\cup Y}(x) = 1$) y $x \notin X \cup Y$ (cuando $C_{X\cup Y}(x) = 0$).

Cómo encontrar una solución

Primero se considera probar la contrapositiva

Si $X \cap Y \neq \emptyset$, entonces $C_{X\cup Y}(x) \neq C_X(x) + C_Y(x)$
para al menos una $x \in U$.

Como se supone que $X \cap Y \neq \emptyset$, existe un elemento $x \in X \cap Y$. Ahora se comparan los valores de las expresiones $C_{X\cup Y}(x)$ y $C_X(x) + C_Y(x)$. Como $x \in X \cup Y$,

$$C_{X\cup Y}(x) = 1.$$

Como $x \in X \cap Y$, $x \in X$ y $x \in Y$. Por lo tanto,

$$C_X(x) + C_Y(x) = 1 + 1 = 2.$$

Se ha probado que

$C_{X\cup Y}(x) \neq C_X(x) + C_Y(x)$ para al menos una $x \in U$.

Ahora considere probar la segunda afirmación

Si $X \cap Y = \emptyset$, entonces $C_{X\cup Y}(x) = C_X(x) + C_Y(x)$ para toda $x \in U$.

Esta vez, se supone que $X \cap Y = \emptyset$. Se calculará cada lado de la ecuación

$$C_{X\cup Y}(x) = C_X(x) + C_Y(x)$$

para cada $x \in U$. Como se sugirió, se consideran los casos: $x \in X \cup Y$ y $x \notin X \cup Y$. Si $x \in X \cup Y$, entonces

$$C_{X\cup Y}(x) = 1\,.$$

Como $X \cap Y = \emptyset$, $x \in X$ o bien $x \in Y$ *pero no ambos*. Por lo tanto,

$$C_X(x) + C_Y(x) = 1 + 0 = 1 = C_{X\cup Y}(x)$$

o

$$C_X(x) + C_Y(x) = 0 + 1 = 1 = C_{X\cup Y}(x).$$

La ecuación

$$C_{X\cup Y}(x) = C_X(x) + C_Y(x)$$

es verdadera si $x \in X \cup Y$.

Si $x \notin X \cup Y$, entonces

$$C_{X\cup Y}(x) = 0.$$

Pero si $x \notin X \cup Y$, entonces $x \notin X$ y $x \notin Y$. Por lo tanto,

$$C_X(x) + C_Y(x) = 0 + 0 = 0 = C_{X\cup Y}(x).$$

La ecuación

$$C_{X\cup Y}(x) = C_X(x) + C_Y(x)$$

es verdadera si $x \notin X \cup Y$. Así,

$$C_{X\cup Y}(x) = C_X(x) + C_Y(x)$$

es verdadera para toda $x \in U$.

Solución formal

La prueba formal se escribe como sigue.

CASO →: Si $C_{X\cup Y}(x) = C_X(x) + C_Y(x)$ para toda $x \in U$, entonces $X \cap Y = \emptyset$.

Se demostrará la contrapositiva equivalente

Si $X \cap Y \neq \emptyset$, entonces $C_{X\cup Y}(x) \neq C_X(x) + C_Y(x)$ para al menos una $x \in U$.

Como $X \cap Y = \emptyset$, existe $x \in X \cap Y$. Puesto que $x \in X \cup Y$,

$$C_{X\cup Y}(x) = 1.$$

Como $x \in X \cap Y$, $x \in X$ y $x \in Y$. Por lo tanto,

$$C_X(x) + C_Y(x) = 1 + 1 = 2.$$

Así,

$$C_{X\cup Y}(x) \neq C_X(x) + C_Y(x).$$

CASO ←: Si $X \cap Y = \emptyset$, entonces $C_{X\cup Y}(x) = C_X(x) + C_Y(x)$ para toda $x \in U$.

Suponga que $x \in X \cup Y$. Entonces

$$C_{X\cup Y}(x) = 1.$$

Puesto que $X \cap Y = \emptyset$, $x \in X$ o $x \in Y$, pero no ambos. Por lo tanto,

$$C_X(x) + C_Y(x) = 1.$$

Así,

$$C_{X\cup Y}(x) = C_X(x) + C_Y(x).$$

Si $x \notin X \cup Y$, entonces

$$C_{X\cup Y}(x) = 0.$$

Si $x \notin X \cup Y$, entonces $x \notin X$ y $x \notin Y$. Por lo tanto,

$$C_X(x) + C_Y(x) = 0.$$

De nuevo,

$$C_{X\cup Y}(x) = C_X(x) + C_Y(x).$$

Así,

$$C_{X\cup Y}(x) = C_X(x) + C_Y(x)$$

para toda $x \in U$.

Resumen de las técnicas de solución del problema

Escriba con exactitud lo que debe probarse.

En lugar de demostrar $p \rightarrow q$ directamente, considere probar su contrapositiva $\neg q \rightarrow \neg p$ o probar por contradicción.

Para afirmaciones que incluyen una negación, las leyes de De Morgan resultan útiles.

Busque definiciones y teoremas relevantes para las expresiones mencionadas en las afirmaciones que deben demostrarse.

Una definición que incluye casos sugiere una prueba por casos.

2.3 ➜ Sucesiones y cadenas

La compañía Blue Taxi Inc. cobra \$1 por la primera milla y \$0.50 por cada milla adicional. La tabla siguiente muestra el costo de viajar de 1 a 10 millas. En general, el costo C_n de viajar n millas es 1.00 (el costo de viajar la primera milla) más 0.50 multiplicado por el número de millas adicionales $(n - 1)$. Es decir,

$$C_n = 1 + 0.5(n - 1).$$

Millas	Costo
1	\$1.00
2	1.50
3	2.00
4	2.50
5	3.00
6	3.50
7	4.00
8	4.50
9	5.00
10	5.50

Como ejemplo,

$$C_1 = 1 + 0.5(1 - 1) = 1 + 0.5 \cdot 0 = 1,$$
$$C_5 = 1 + 0.5(5 - 1) = 1 + 0.5 \cdot 4 = 1 + 2 = 3.$$

La lista de tarifas

$$C_1 = 1.00, \quad C_2 = 1.50, \quad C_3 = 2.00, \quad C_4 = 2.50, \quad C_5 = 3.00,$$
$$C_6 = 3.50, \quad C_7 = 4.00, \quad C_8 = 4.50, \quad C_9 = 5.00, \quad C_{10} = 5.50$$

proporciona un ejemplo de una **sucesión**, que es un tipo especial de función donde el dominio consiste en un conjunto de enteros consecutivos. Para la sucesión de tarifas, el dominio es el conjunto {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}. El n-ésimo término se denota por C_n, aunque para ser congruentes con la notación más general de funciones, podría escribirse (y a veces se escribe) $C(n)$. A n se le llama **índice** de la sucesión.

WWW

Una sucesión llamada, digamos, s se denota por s o $\{s_n\}$. Aquí s o $\{s_n\}$ denota a la sucesión completa

$$s_1, s_2, s_3, \ldots .$$

Se usa la notación s_n para denotar a un solo elemento de la sucesión s con el índice n.

Ejemplo 2.3.1 ▶

Considere la sucesión s

$$2, 4, 6, \ldots , \qquad 2n, \ldots .$$

El primer elemento de la sucesión es 2, el segundo es 4, etcétera. El elemento n de la sucesión es $2n$. Si el primer índice es 1, se tiene

$$s_1 = 2, \quad s_2 = 4, \quad s_3 = 6, \ldots , \quad s_n = 2n, \ldots . \qquad \blacktriangleleft$$

Ejemplo 2.3.2 ▶

Considere la sucesión t

$$a, a, b, a, b.$$

El primer elemento de la sucesión es a, el segundo elemento es a, etcétera. Si el primer índice es 1, se tiene

$$t_1 = a, \quad t_2 = a, \quad t_3 = b, \quad t_4 = a, \quad t_5 = b. \qquad \blacktriangleleft$$

Si el dominio de la sucesión es infinito (como en el ejemplo 2.3.1), se dice que la *sucesión es infinita*. Si el domino de la sucesión es finito (como en el ejemplo 2.3.2), se dice que la *sucesión es finita*. Cuando se desea establecer de manera explícita el índice inicial k de una sucesión infinita s, se escribe $\{s_n\}_{n=k}^{\infty}$. Por ejemplo, una sucesión infinita v cuyo índice inicial es 0 se denota por $\{v_n\}_{n=0}^{\infty}$. Una sucesión finita x indexada de i a j se denota por $\{x_n\}_{n=i}^{j}$. Por ejemplo, una sucesión t cuyo dominio es $\{-1, 0, 1, 2, 3\}$ se denota como $\{t_n\}_{n=-1}^{3}$.

Ejemplo 2.3.3 ▶

La sucesión $\{u_n\}$ definida por la regla

$$u_n = n^2 - 1, \qquad n \geq 0,$$

se denota por $\{u_n\}_{n=0}^{\infty}$. El nombre del índice se puede elegir de cualquier manera conveniente. Por ejemplo, la sucesión u también se puede denotar como $\{u_m\}_{m=0}^{\infty}$. La fórmula para el término que tiene el índice m es

$$u_m = m^2 - 1, \qquad m \geq 0.$$

Esta sucesión es infinita. $\blacktriangleleft$

Ejemplo 2.3.4 ▶

Defina una sucesión b mediante la regla b_n es la n-ésima letra en la palabra *digital*. Si el índice del primer término es 1, entonces $b_1 = d$, $b_2 = b_4 = i$ y $b_7 = l$. Ésta es una sucesión finita. Se denota por $\{b_k\}_{k=1}^{7}$. ◀

Ejemplo 2.3.5 ▶

Si x es la sucesión definida por

$$x_n = \frac{1}{2^n}, \qquad -1 \le n \le 4,$$

Los elementos de x son

$$2,\ 1,\ 1/2,\ 1/4,\ 1/8,\ 1/16.$$ ◀

Ejemplo 2.3.6 ▶

Defina una sucesión s como

$$s_n = 2^n + 4 \cdot 3^n, \qquad n \ge 0. \tag{2.3.1}$$

a) Encuentre s_0.
b) Encuentre s_1.
c) Encuentre una fórmula para s_i.
d) Encuentre una fórmula para s_{n-1}.
e) Encuentre una fórmula para s_{n-2}.
f) Demuestre que $\{s_n\}$ satisface

$$s_n = 5s_{n-1} - 6s_{n-2} \qquad \text{para toda } n \ge 2. \tag{2.3.2}$$

a) Sustituyendo n por 0 en la definición 2.3.1 se obtiene

$$s_0 = 2^0 + 4 \cdot 3^0 = 5.$$

b) Sustituyendo n por 1 en la definición 2.3.1 se obtiene

$$s_1 = 2^1 + 4 \cdot 3^1 = 14.$$

c) Sustituyendo n por i en la definición 2.3.1 se obtiene

$$s_i = 2^i + 4 \cdot 3^i.$$

d) Sustituyendo n por $n - 1$ en la definición 2.3.1 se obtiene

$$s_{n-1} = 2^{n-1} + 4 \cdot 3^{n-1}.$$

e) Sustituyendo n por $n - 2$ en la definición 2.3.1 se obtiene

$$s_{n-2} = 2^{n-2} + 4 \cdot 3^{n-2}.$$

f) Para probar la ecuación (2.3.2), se sustituyen s_{n-1} y s_{n-2} en el lado derecho de la ecuación (2.3.2) por las fórmulas en los incisos *d)* y *e)*. Después se usa álgebra para demostrar que el resultado es igual a s_n. Se obtiene

$$\begin{aligned} 5s_{n-1} - 6s_{n-2} &= 5(2^{n-1} + 4 \cdot 3^{n-1}) - 6(2^{n-2} + 4 \cdot 3^{n-2}) \\ &= (5 \cdot 2 - 6)2^{n-2} + (5 \cdot 4 \cdot 3 - 6 \cdot 4)3^{n-2} \\ &= 4 \cdot 2^{n-2} + 36 \cdot 3^{n-2} \\ &= 2^2 2^{n-2} + (4 \cdot 3^2)3^{n-2} \\ &= 2^n + 4 \cdot 3^n = s_n. \end{aligned}$$

La técnica mostrada en el ejemplo será útil al verificar la solución de las relaciones de recurrencia en el capítulo 7. ◀

Dos tipos importantes de sucesiones son las sucesiones crecientes y decrecientes, y sus relativas: las sucesiones no crecientes y no decrecientes. Una sucesión s es **creciente** si $s_n < s_{n+1}$ para toda n para la que n y $n+1$ están en el dominio de la sucesión. Una sucesión s es **decreciente** si $s_n > s_{n+1}$ para toda n para la que n y $n+1$ están en el dominio de la sucesión. Una sucesión s es **no decreciente** si $s_n \leq s_{n+1}$ para toda n para la que n y $n+1$ están en el dominio de la sucesión. (Una sucesión no decreciente es como una sucesión creciente excepto que "$<$" se sustituye por "$\leq$"). Una sucesión s es **no creciente** si $s_n \geq s_{n+1}$ para toda n para la que n y $n+1$ están en el dominio de la sucesión. (Una sucesión no creciente es como una sucesión decreciente excepto que "$>$" se sustituye por "$\geq$").

Ejemplo 2.3.7 ▶ La sucesión

$$2, 5, 13, 104, 300$$

es creciente y no decreciente. ◀

Ejemplo 2.3.8 ▶ La sucesión

$$a_i = \frac{1}{i}, \qquad i \geq 1,$$

es decreciente y no creciente. ◀

Ejemplo 2.3.9 ▶ La sucesión

$$100, 90, 90, 74, 74, 74, 30$$

es no creciente, pero *no* es decreciente. ◀

Ejemplo 2.3.10 ▶ La sucesión

$$100$$

es creciente, decreciente, no creciente y no decreciente, ya que no existe un valor de i para el cual i e $i+1$ sean índices. ◀

Una manera de formar una nueva sucesión a partir de otra es retener sólo ciertos términos de la sucesión original, manteniendo el orden de los términos. La sucesión obtenida se llama **subsucesión** de la sucesión original.

Definición 2.3.11 ▶ Sea $\{s_n\}$ una sucesión definida para $n = m, m+1, \ldots$, y sea $n_1, n_2, \ldots$ una sucesión creciente cuyos valores están en el conjunto $\{m, m+1, \ldots\}$. La sucesión $\{s_{n_k}\}$ recibe el nombre de *subsucesión* de $\{s_n\}$. ◀

Ejemplo 2.3.12 ▶ La sucesión

$$b, c \tag{2.3.3}$$

es una subsucesión de la sucesión

$$t_1 = a, \qquad t_2 = a, \qquad t_3 = b, \qquad t_4 = c, \qquad t_5 = q. \tag{2.3.4}$$

La subsucesión (2.3.3) se obtiene a partir de la sucesión (2.3.4) eligiendo el tercero y cuarto términos. La expresión n_k en la definición 2.3.11 indica qué términos de (2.3.4) elegir para obtener la subsucesión (2.3.3); así, $n_1 = 3$, $n_2 = 4$. La subsucesión (2.3.3) es

$$t_3,\ t_4 \quad \text{o} \quad t_{n_1},\ t_{n_2}.$$

Observe que la sucesión

$$c,\ b$$

no es una subsucesión de la sucesión (2.3.4) puesto que no se mantiene el orden de los términos de esta sucesión. ◀

Ejemplo 2.3.13 ▶ La sucesión

$$2, 4, 8, 16, \ldots, 2^k, \ldots \tag{2.3.5}$$

es una subsucesión de la sucesión

$$2, 4, 6, 8, 10, 12, 14, 16, \ldots, 2n, \ldots. \tag{2.3.6}$$

La subsucesión (2.3.5) se obtiene de la sucesión (2.3.6) eligiendo el primer término, el segundo, cuarto, octavo, etcétera; entonces el valor n_k de la definición 2.3.11 es $n_k = 2^{k-1}$. Si se define la sucesión (2.3.6) por $s_n = 2n$, la subsucesión (2.3.5) está definida por

$$s_{n_k} = s_{2^{k-1}} = 2 \cdot 2^{k-1} = 2^k. \quad ◀$$

Dos operaciones importantes en las sucesiones numéricas son sumar y multiplicar términos.

Definición 2.3.14 ▶ Si $\{a_i\}_{i=m}^n$ es una sucesión, se define

$$\sum_{i=m}^{n} a_i = a_m + a_{m+1} + \cdots + a_n, \qquad \prod_{i=m}^{n} a_i = a_m \cdot a_{m+1} \cdots a_n.$$

WWW El formalismo

$$\sum_{i=m}^{n} a_i \tag{2.3.7}$$

se llama *notación de suma* (o *sigma*) y

$$\prod_{i=m}^{n} a_i \tag{2.3.8}$$

se llama *notación de producto*.

En (2.3.7) o (2.3.8), i se llama *índice*, m se llama *límite inferior*, y n se llama *límite superior*. ◀

Ejemplo 2.3.15 ▶ Sea a una sucesión definida por $a_n = 2n$, $n \geq 1$. Entonces

$$\sum_{i=1}^{3} a_i = a_1 + a_2 + a_3 = 2 + 4 + 6 = 12,$$

$$\prod_{i=1}^{3} a_i = a_1 \cdot a_2 \cdot a_3 = 2 \cdot 4 \cdot 6 = 48.$$

◀

Ejemplo 2.3.16 ▶ La suma geométrica

$$a + ar + ar^2 + \cdots + ar^n$$

se puede rescribir en forma compacta usando la notación de suma,

$$\sum_{i=0}^{n} ar^i. \quad ◀$$

En ocasiones, es útil cambiar no sólo el nombre del índice, sino también sus límites. (El proceso es análogo a cambiar la variable de una integral en cálculo.)

Ejemplo 2.3.17 ▶

Cambio del índice y los límites en una suma

Rescriba la suma

$$\sum_{i=0}^{n} ir^{n-i},$$

sustituyendo el índice i por j, donde $i = j - 1$.

Como $i = j - 1$, el término ir^{n-i} se convierte en

$$(j-1)r^{n-(j-1)} = (j-1)r^{n-j+1}.$$

Como $j = i + 1$, cuando $i = 0$, $j = 1$. Entonces, el límite inferior para j es 1. De manera similar, cuando $i = n$, $j = n + 1$, y el límite superior para j es $n + 1$. Por lo tanto,

$$\sum_{i=0}^{n} ir^{n-i} = \sum_{j=1}^{n+1} (j-1)r^{n-j+1}. \quad \blacktriangleleft$$

Ejemplo 2.3.18 ▶

Sea a la sucesión definida por la regla $a_i = 2(-1)^i$, $i \geq 0$. Encuentre una fórmula para la sucesión s definida por

$$s_n = \sum_{i=0}^{n} a_i.$$

Se encuentra que

$$s_n = 2(-1)^0 + 2(-1)^1 + 2(-1)^2 + \cdots + 2(-1)^n$$

$$= 2 - 2 + 2 - \cdots \pm 2 = \begin{cases} 2 & \text{si } n \text{ es par} \\ 0 & \text{si } n \text{ es impar.} \end{cases} \quad \blacktriangleleft$$

Algunas veces, las notaciones de suma y producto se modifican para denotar sumas y productos indexados sobre conjuntos de enteros arbitrarios. De manera formal, si S es un conjunto finito de enteros y a es una sucesión,

$$\sum_{i \in S} a_i$$

denota la suma de los elementos $\{a_i \mid i \in S\}$. De manera similar,

$$\prod_{i \in S} a_i$$

denota el producto de los elementos $\{a_i \mid i \in S\}$.

Ejemplo 2.3.19 ▶

Si S denota el conjunto de números primos menores que 20,

$$\sum_{i \in S} \frac{1}{i} = \frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{11} + \frac{1}{13} + \frac{1}{17} + \frac{1}{19} = 1.455. \quad \blacktriangleleft$$

Una *cadena* es una sucesión finita de caracteres. En lenguajes de programación, las cadenas se emplean para denotar texto. Por ejemplo, en Java

```
''Vamos a leer Rolling Stone.''
```

denota la cadena que consiste en la sucesión de caracteres

Vamos a leer Rolling Stone.

(Las comillas marcan el inicio y fin de la cadena).

Dentro de una computadora, las *cadenas de bits* (cadenas de ceros y unos) representan datos e instrucciones a ejecutar. Como se verá en la sección 5.2, la cadena de bits 101111 representa el número 47.

Definición 2.3.20 ▶ Una *cadena sobre X*, donde *X* es un conjunto finito, es una sucesión finita de elementos de *X*. ◀

Ejemplo 2.3.21 ▶ Sea $X = \{a, b, c\}$. Si se hace

$$\beta_1 = b, \quad \beta_2 = a, \quad \beta_3 = a, \quad \beta_4 = c,$$

se obtiene una cadena sobre *X*. Esta cadena se escribe *baac*. ◀

Como una cadena es una sucesión, toma en cuenta el orden. Por ejemplo, la cadena *baac* es diferente de la cadena *acab*.

Las repeticiones en una cadena se especifican mediante superíndices. Por ejemplo, la cadena *bbaaac* se escribe b^2a^3c.

La cadena sin elementos se llama **cadena nula** y se denota por λ. Definimos X^* como el conjunto de todas las cadenas sobre *X*, incluyendo la cadena nula, y X^+ como el conjunto de todas la cadenas no nulas sobre *X*.

Ejemplo 2.3.22 ▶ Sea $X = \{a, b\}$. Algunos elementos en X^* son

$$\lambda, \quad a, \quad b, \quad abab, \quad b^{20}a^5ba.$$ ◀

La **longitud** de una cadena α es el número de elementos en α. La longitud de α se denota por $|\alpha|$.

Ejemplo 2.3.23 ▶ Si $\alpha = aabab$ y $\beta = a^3b^4a^{32}$, entonces,

$$|\alpha| = 5 \quad \text{y} \quad |\beta| = 39$$ ◀

Si α y β son dos cadenas, la cadena que consiste en α seguida de β, escrita $\alpha\beta$, se llama **concatenación** de α y β.

Ejemplo 2.3.24 ▶ Si $\gamma = aab$ y $\theta = cabd$, entonces

$$\gamma\theta = aabcabd, \quad \theta\gamma = cabdaab, \quad \gamma\lambda = \gamma = aab, \quad \lambda\gamma = \gamma = aab.$$ ◀

Ejemplo 2.3.25 ▶ Sea $X = \{a, b, c\}$. Si se define

$$f(\alpha, \beta) = \alpha\beta,$$

donde α y β son cadenas sobre *X*, entonces *f* es un operador binario sobre X^*. ◀

Una **subcadena** de una cadena α se obtiene eligiendo algunos o todos los elementos consecutivos de α. La definición formal es la siguiente.

Definición 2.3.26 ▶ Una cadena β es una subcadena de la cadena α si existen cadenas γ y δ con $\alpha = \gamma\beta\delta$. ◀

Ejemplo 2.3.27 ▶ La cadena $\beta = add$ es una subcadena de la cadena $\alpha = aaaddad$ puesto que, si se toma $\gamma = aa$ y $\delta = ad$, se tiene $\alpha = \gamma\beta\delta$. Observe que si β es una subcadena de α, γ es la parte de α que precede a β (en α), y δ es la parte de α que sigue a β (en α). ◀

Sugerencias para resolver problemas

Una sucesión es un tipo especial de función; el dominio es un conjunto de enteros consecutivos. Si $a_1, a_2, \ldots$ es una sucesión, los números 1, 2, . . . se llaman índices. El índice 1 iden-

tifica el primer elemento de la sucesión a_1; el índice 2 identifica el segundo elemento de la sucesión a_2; etcétera.

En este libro, "sucesión creciente" significa *estrictamente* creciente; es decir, la sucesión a es creciente si $a_n < a_{n+1}$ para toda n. Se requiere que a_n sea estrictamente menor que a_{n+1} para toda n. Permitir la igualdad lleva a lo que en este libro se llama "sucesión no decreciente". Esto es, la sucesión a es no decreciente si $a_n \leq a_{n+1}$ para toda n. Observaciones similares se aplican a las sucesiones decrecientes y no crecientes.

Sección de ejercicios de repaso

1. Defina *sucesión*.

2. ¿Qué es un índice en una sucesión?

3. Defina *sucesión creciente*.

4. Defina *sucesión decreciente*.

5. Defina *sucesión no creciente*.

6. Defina *sucesión no decreciente*.

7. Defina *subsucesión*.

8. ¿Qué es $\sum_{i=m}^{n} a_i$?

9. ¿Qué es $\prod_{i=m}^{n} a_i$?

10. Defina *cadena*.

11. Defina *cadena nula*.

12. Si X es un conjunto finito, ¿qué es X^*?

13. Si X es un conjunto finito, ¿qué es X^+?

14. Defina *longitud de una cadena*. ¿Cómo se denota la longitud de la cadena α?

15. Defina *concatenación de cadenas*. ¿Cómo se denota la concatenación de las cadenas α y β?

16. Defina *subcadena*.

Ejercicios

Responda a los ejercicios 1 al 3 para la sucesión s definida por

$$c, d, d, c, d, c.$$

1. Encuentre s_1.

2. Encuentre s_4.

3. Escriba s como una cadena.

Responda a los ejercicios 4 al 16 para la sucesión t definida por

$$t_n = 2n - 1, \quad n \geq 1.$$

4. Encuentre t_3.

5. Encuentre t_7.

6. Encuentre t_{100}.

7. Encuentre t_{2077}.

8. Encuentre $\sum_{i=1}^{3} t_i$.

9. Encuentre $\sum_{i=3}^{7} t_i$.

10. Encuentre $\prod_{i=1}^{3} t_i$.

11. Encuentre $\prod_{i=3}^{6} t_i$.

12. Encuentre una fórmula que represente esta sucesión como una sucesión cuyo índice inferior es 0.

13. ¿Es t creciente?

14. ¿Es t decreciente?

15. ¿Es t no creciente?

16. ¿Es t no decreciente?

Responda a los ejercicios 17 al 24 para la sucesión v definida por

$$v_n = n! + 2, \quad n \geq 1.$$

17. Encuentre v_3.

18. Encuentre v_4.

19. Encuentre $\sum_{i=1}^{4} v_i$.

20. Encuentre $\sum_{i=3}^{3} v_i$

21. ¿Es v creciente?

22. ¿Es v decreciente?

23. ¿Es v no creciente?

24. ¿Es v no decreciente?

Responda a los ejercicios 25 al 30 para la sucesión

$$q_1 = 8, \quad q_2 = 12, \quad q_3 = 12, \quad q_4 = 28, \quad q_5 = 33.$$

25. Encuentre $\sum_{i=2}^{4} q_i$.

26. Encuentre $\sum_{k=2}^{4} q_k$.

27. ¿Es q creciente?

28. ¿Es q decreciente?

29. ¿Es q no creciente?

30. ¿Es q no decreciente?

Responda a los ejercicios 31 al 34 para la sucesión

$$\tau_0 = 5, \quad \tau_2 = 5.$$

31. ¿Es τ creciente?

32. ¿Es τ decreciente?

33. ¿Es τ no creciente?

34. ¿Es τ no decreciente?

Responda a los ejercicios 35 al 38 para la sucesión

$$\Upsilon_2 = 5.$$

35. ¿Es Υ creciente?

36. ¿Es Υ decreciente?

37. ¿Es Υ no creciente?

38. ¿Es Υ no decreciente?

Responda a los ejercicios 39 al 50 para la sucesión definida por

$$a_n = n^2 - 3n + 3, \quad n \geq 1.$$

39. Encuentre $\sum_{i=1}^{4} a_i$.

40. Encuentre $\sum_{j=3}^{5} a_j$.

41. Encuentre $\sum_{i=4}^{4} a_i$. ***42.*** Encuentre $\sum_{k=1}^{6} a_k$.

43. Encuentre $\prod_{i=1}^{2} a_i$. ***44.*** Encuentre $\prod_{i=1}^{3} a_i$.

45. Encuentre $\prod_{n=2}^{3} a_n$. ***46.*** Encuentre $\prod_{x=3}^{4} a_x$.

47. ¿Es a creciente? ***48.*** ¿Es a decreciente?

49. ¿Es a no creciente? ***50.*** ¿Es a no decreciente?

Responda a los ejercicios 51 al 58 para la sucesión b definida por $b_n = n(-1)^n$, $n \geq 1$.

51. Encuentre $\sum_{i=1}^{4} b_i$. **52.** Encuentre $\sum_{i=1}^{10} b_i$.

53. Encuentre una fórmula para la sucesión c definida por

$$c_n = \sum_{i=1}^{n} b_i.$$

54. Encuentre una fórmula para la sucesión d definida por

$$d_n = \prod_{i=1}^{n} b_i.$$

55. ¿Es b creciente? **56.** ¿Es b decreciente?

57. ¿Es b no creciente? **58.** ¿Es b no decreciente?

Responda a los ejercicios 59 al 66 para la sucesión Ω *definida por* $\Omega_n = 3$ para toda n.

59. Encuentre $\sum_{i=1}^{3} \Omega_i$. **60.** Encuentre $\sum_{i=1}^{10} \Omega_i$.

61. Encuentre una fórmula para la sucesión c definida por

$$c_n = \sum_{i=1}^{n} \Omega_i.$$

62. Encuentre una fórmula para la sucesión d definida por

$$d_n = \prod_{i=1}^{n} \Omega_i.$$

63. ¿Es Ω creciente? **64.** ¿Es Ω decreciente?

65. ¿Es Ω no creciente? **66.** ¿Es Ω no decreciente?

Responda a los ejercicios 67 al 73 para la sucesión x definida por

$$x_1 = 2, \quad x_n = 3 + x_{n-1}, \qquad n \geq 2.$$

67. Encuentre $\sum_{i=1}^{3} x_i$. ***68.*** Encuentre $\sum_{i=1}^{10} x_i$.

69. Encuentre una fórmula para la sucesión c definida por

$$c_n = \sum_{i=1}^{n} x_i.$$

70. ¿Es x creciente? ***71.*** ¿Es x decreciente?

72. ¿Es x no creciente? ***73.*** ¿Es x no decreciente?

Responda a los ejercicios 74 al 81 para la sucesión w definida por

$$w_n = \frac{1}{n} - \frac{1}{n+1}, \qquad n \geq 1.$$

74. Encuentre $\sum_{i=1}^{3} w_i$. **75.** Encuentre $\sum_{i=1}^{10} w_i$.

76. Encuentre una fórmula para la sucesión c definida por

$$c_n = \sum_{i=1}^{n} w_i.$$

77. Encuentre una fórmula para la sucesión d definida por

$$d_n = \prod_{i=1}^{n} w_i.$$

78. ¿Es w creciente? **79.** ¿Es w decreciente?

80. ¿Es w no creciente? **81.** ¿Es w no decreciente?

82. Sea u la sucesión definida por

$$u_1 = 3, \quad u_n = 3 + u_{n-1}, \qquad n \geq 2.$$

Encuentre una fórmula para la sucesión d definida por

$$d_n = \prod_{i=1}^{n} u_i.$$

Los ejercicios 83 al 86 se refieren a la sucesión $\{s_n\}$ *definida por la regla*

$$s_n = 2n - 1, \quad n \geq 1.$$

83. Enumere los primeros siete términos de s.

Responda a los ejercicios 84 al 86 para la subsucesión de s obtenida tomando el primero, tercero, quinto, ... términos.

84. Liste los primeros siete términos de la subsucesión.

85. Encuentre una fórmula para la expresión n_k de la definición 2.3.11.

86. Encuentre una fórmula para el k-ésimo término de la subsucesión.

Los ejercicios 87 al 90 se refieren a la sucesión $\{t_n\}$ *definida por la regla*

$$t_n = 2^n, \qquad n \geq 1.$$

87. Liste los primeros siete términos de t.

Responda a los ejercicios 88 al 90 para la subsucesión de t obtenida al tomar el primero, segundo, cuarto, séptimo, undécimo, ... términos.

88. Liste los primeros siete términos de la subsucesión.

89. Encuentre una fórmula para la expresión n_k de la definición 2.3.11.

90. Encuentre una fórmula para el k-ésimo término de la subsucesión.

Responda a los ejercicios 91 al 94 usando las sucesiones y y z *definidas por*

$$y_n = 2^n - 1 \qquad z_n = n(n-1).$$

91. Encuentre $\left(\sum_{i=1}^{3} y_i\right)\left(\sum_{i=1}^{3} z_i\right)$. ***92.*** Encuentre $\left(\sum_{i=1}^{5} y_i\right)\left(\sum_{i=1}^{4} z_i\right)$.

93. Encuentre $\sum_{i=1}^{3} y_i z_i$. ***94.*** Encuentre $\left(\sum_{i=3}^{4} y_i\right)\left(\prod_{i=2}^{4} z_i\right)$.

Responda a los ejercicios 95 al 102 para la sucesión r definida por

$$r_n = 3 \cdot 2^n - 4 \cdot 5^n, \qquad n \geq 0.$$

95. Encuentre r_0. **96.** Encuentre r_1.

97. Encuentre r_2. **98.** Encuentre r_3.

99. Encuentre una fórmula para r_p.

100. Encuentre una fórmula para r_{n-1}.

101. Encuentre una fórmula para r_{n-2}.

102. Demuestre que $\{r_n\}$ satisface

$$r_n = 7r_{n-1} - 10r_{n-2}, \quad n \geq 2.$$

Responda a los ejercicios 103 al 110 para la sucesión z definida por

$$z_n = (2+n)3^n, \qquad n \geq 0.$$

103. Encuentre z_0.

104. Encuentre z_1.

105. Encuentre z_2.

106. Encuentre z_3.

107. Encuentre una fórmula para z_i.

108. Encuentre una fórmula para z_{n-1}.

109. Encuentre una fórmula para z_{n-2}.

110. Demuestre que $\{z_n\}$ satisface

$$z_n = 6z_{n-1} - 9z_{n-2}, \qquad n \geq 2.$$

111. Encuentre b_n, $n = 1, \ldots, 6$, donde

$$b_n = n + (n-1)(n-2)(n-3)(n-4)(n-5).$$

112. Rescriba la suma

$$\sum_{i=1}^{n} i^2 r^{n-i},$$

sustituyendo el índice i por k, donde $i = k+1$.

113. Rescriba la suma

$$\sum_{k=1}^{n} C_{k-1} C_{n-k},$$

sustituyendo el índice k por i donde $k = i+1$.

114. Sean a y b dos sucesiones y sea

$$s_k = \sum_{i=1}^{k} a_i.$$

Demuestre que

$$\sum_{k=1}^{n} a_k b_k = \sum_{k=1}^{n} s_k(b_k - b_{k+1}) + s_n b_{n+1}.$$

Esta ecuación, conocida como *fórmula de la suma por partes*, es el análogo discreto de la fórmula de integración por partes en cálculo.

115. En ocasiones, se generaliza la noción de sucesión según se define en esta sección al permitir índices más generales. Suponga que $\{a_{ij}\}$ es una sucesión indexada sobre los pares de enteros positivos. Demuestre que

$$\sum_{i=1}^{n} \left(\sum_{j=i}^{n} a_{ij} \right) = \sum_{j=1}^{n} \left(\sum_{i=1}^{j} a_{ij} \right).$$

116. Calcule la cantidad indicada usando las cadenas

$$\alpha = baab, \quad \beta = caaba, \quad \gamma = bbab.$$

a) $\alpha\beta$	*b)* $\beta\alpha$	*c)* $\alpha\alpha$
d) $\beta\beta$	*e)* $\lvert\alpha\beta\rvert$	*f)* $\lvert\beta\alpha\rvert$
g) $\lvert\alpha\alpha\rvert$	*h)* $\lvert\beta\beta\rvert$	*i)* $\alpha\lambda$
j) $\lambda\beta$	*k)* $\alpha\beta\gamma$	*l)* $\beta\beta\gamma\alpha$

117. Liste todas las cadenas sobre $X = \{0,1\}$ de longitud 2.

118. Liste todas las cadenas sobre $X = \{0,1\}$ de longitud 2 o menos.

119. Liste todas las cadenas sobre $X = \{0,1\}$ de longitud 3.

120. Liste todas las cadenas sobre $X = \{0,1\}$ de longitud 3 o menos.

121. Encuentre todas las subcadenas de la cadena *babc*.

122. Encuentre todas las subcadenas de la cadena aa*baabb*.

123. Use inducción para probar que

$$\sum \frac{1}{n_1 \cdot n_2 \cdots n_k} = n,$$

para toda $n \geq 1$, donde la suma se toma sobre todos los subconjuntos no vacíos $\{n_1, n_2, \ldots, n_k\}$ de $\{1, 2, \ldots, n\}$.

Sea L el conjunto de todas las cadenas, incluyendo la cadena nula, que se pueden construir con la aplicación repetida de las siguientes reglas:

- *Si $\alpha \in L$, entonces $a\alpha b \in L$ y $b\alpha a \in L$.*
- *Si $\alpha \in L$ y $\beta \in L$, entonces $\alpha\beta \in L$.*

Por ejemplo, ab está en L, ya que si se hace $\alpha = \lambda$, entonces $\alpha \in L$ y la primera regla establece que $ab = a\alpha b \in L$. De manera similar, $ba \in L$. Como otro ejemplo, aabb está en L porque si se toma $\alpha = ab$, entonces $\alpha \in L$; por la primera regla, $aabb = a\alpha b \in L$. Como ejemplo final, aabbba está en L, porque si se hace $\alpha = aabb$ y $\beta = ba$, entonces $\alpha \in L$ y $\beta \in L$; por la segunda regla, $aabbba = \alpha\beta \in L$.

124. Demuestre que *aaabbb* está en *L*.

125. Demuestre que *baabab* está en *L*.

126. Demuestre que *aab* no está en *L*.

127. Pruebe que si $\alpha \in L$, α tiene el mismo número de letras *a* y *b*.

★**128.** Pruebe que si α tiene el mismo número de letras *a* y *b*, entonces $\alpha \in L$.

Nota

La mayoría de las referencias generales sobre matemáticas discretas abordan los temas tratados en este capítulo. [Halmos; Lipschutz; y Stoll] son referencias recomendables para el lector que desee estudiar teoría de conjuntos y sus funciones con mayor detalle.

Repaso del capítulo

Sección 2.1

1. Conjunto: cualquier colección de objetos.
2. Notación de conjuntos: $\{x \mid x$ tiene la propiedad $P\}$
3. $|X|$: el número de elementos en el conjunto X
4. $x \in X$: x es un elemento del conjunto X
5. $x \notin X$: x no es un elemento del conjunto X

6. Conjunto vacío: $\emptyset$ o $\{\}$
7. $X = Y$, donde X y Y son conjuntos: X y Y tienen los mismos elementos
8. $X \subseteq Y$, X es un subconjunto de Y: todo elemento de X está también en Y
9. $X \subset Y$, X es un subconjunto propio de Y: $X \subseteq Y$ y $X \neq Y$
10. $\mathcal{P}(x)$, el conjunto potencia de X: conjunto de todos los subconjuntos de X
11. $|\mathcal{P}(X)| = 2^{|X|}$
12. $X \cup Y$, X unión Y: conjunto de elementos en X *o* Y o ambos
13. Unión de una familia S de conjuntos: $\cup S = \{x \mid x \in X \text{ para alguna } X \in S\}$
14. $X \cap Y$, X intersección Y: conjunto de elementos en X *y* Y
15. Intersección de una familia S de conjuntos: $\cap S = \{x \mid x \in X \text{ para toda } X \in S\}$
16. Conjuntos disjuntos X y Y: $X \cap Y = \emptyset$
17. Familia de conjuntos disjuntos por pares
18. $X - Y$, diferencia de X y Y, complemento relativo: conjunto de elementos en X pero no en Y
19. Conjunto universal, universo
20. $\overline{X}$, complemento de X: $U - X$, donde U es el conjunto universal
21. Diagrama de Venn
22. Propiedades de conjuntos (vea el Teorema 2.1.12)
23. Leyes de De Morgan para conjuntos: $\overline{(A \cup B)} = \overline{A} \cap \overline{B}$, $\overline{(A \cap B)} = \overline{A} \cup \overline{B}$
24. Partición de X: colección S de subconjuntos no vacíos de X tales que cada elemento en X pertenece exactamente a un miembro de S
25. Par ordenado: (x, y)
26. Producto cartesiano de X y Y: $X \times Y = \{(x, y) | x \in X, y \in Y\}$
27. Producto cartesiano de $X_1, X_2, \ldots, X_n$:

$$X_1 \times X_2 \times \cdots \times X_n = \{(a_1, a_2, \ldots, a_n) \mid a_i \in X_i\}$$

Sección 2.2

28. Función de X a Y, $f: X \to Y$: un subconjunto f de $X \times Y$ tal que para cada $x \in X$, existe exactamente una $y \in Y$ con $(x, y) \in f$
29. x mod y: residuo cuando x se divide entre y
30. Función de dispersión (*hashing*)
31. Colisión para una función de dispersión H: $H(x) = H(\text{y})$
32. Política de solución de colisión
33. Piso de x, $\lfloor \text{x} \rfloor$: mayor entero menor o igual que x
34. Techo de x, $\lceil \text{x} \rceil$: menor entero mayor o igual que x
35. Función uno a uno f: si $f(x) = f(x')$, entonces $x = x'$
36. Función sobre f de X a Y: rango de $f = Y$
37. Biyección: función uno a uno y sobre
38. Inversa f^{-1} de una función f uno a uno, sobre: $f: \{(y, x) \mid (x, y) \in f\}$
39. Composición de funciones: $f \circ g = f \circ g = \{(x, z) \mid (x, y) \in g \text{ y } (y, z) \in f\}$
40. Operador binario sobre X: función de $X \times X$ a X
41. Operador unitario sobre X: función de X a X

Sección 2.3

42. Sucesión: función cuyo dominio es un conjunto de enteros consecutivos
43. Índice: en la sucesión $\{s_n\}$, n es el índice
44. Sucesión creciente: $s_n < s_{n+1}$ para toda n
45. Sucesión decreciente: $s_n > s_{n+1}$ para toda n
46. Sucesión no creciente: $s_n \geq s_{n+1}$ para toda n
47. Sucesión no decreciente: $s_n \leq s_{n+1}$ para toda n
48. Subsucesión s_{n_k} de la sucesión $\{s_n\}$
49. Notación de suma o sigma: $\displaystyle\sum_{i=m}^{n} a_i = a_m + a_{m+1} + \cdots + a_n$
50. Notación de producto: $\displaystyle\prod_{i=m}^{n} a_i = a_m \cdot a_{m+1} \cdots a_n$
51. Suma geométrica: $\displaystyle\sum_{i=0}^{n} ar^i$

52. Cadena: sucesión finita

53. Cadena nula, λ: cadena sin elementos

54. X^*: conjunto de todas las cadenas sobre X, incluyendo la cadena nula

55. X^+: conjunto de todas las cadenas no nulas sobre X

56. Longitud de la cadena α, $|\alpha|$: número de elementos en α

57. Concatenación de cadenas α y β, $\alpha\beta$: α seguida de β

58. Subcadena de α: una cadena β para la cual existen cadenas γ y δ con $\alpha = \gamma\beta\delta$

Autoevaluación del capítulo

Sección 2.1

1. Si $A = \{1, 2, 3, 4, 5, 6, 7\}$, $B = \{x \mid x \text{ es un entero par}\}$, $C = \{2, 3, 4, 5, 6\}$, encuentre $(A \cap B) - C$.

2. Si X es un conjunto y $|X| = 8$, ¿cuántos miembros tiene $\mathcal{P}(X)$? ¿Cuántos sunconjuntos propios tiene X?

3. Si $A \cup B = B$, ¿qué relación debe cumplirse entre A y B?

4. Los conjuntos

$$\{3, 2, 2\} \qquad \{x \mid x \text{ es un entero y } 1 < x \leq 3\},$$

¿son iguales? Explique su respuesta.

Sección 2.2

5. Sea X el conjunto de cadenas sobre $\{a, b\}$ de longitud 4 y sea Y el conjunto de cadenas sobre $\{a, b\}$ de longitud 3. Defina la función f de X a Y mediante la regla

$$f(\alpha) = \text{cadena que consiste en los primeros tres caracteres de } \alpha.$$

¿Es f uno a uno? ¿Es f sobre?

6. Encuentre números reales x y y que satisfagan $\lfloor x \rfloor \lfloor y \rfloor = \lfloor xy \rfloor - 1$.

7. Dé ejemplos de funciones f y g tales que $f \circ g$ sea sobre, pero g no lo sea.

8. Para la función de dispersión

$$h(x) = x \bmod 13,$$

demuestre cómo los datos

$$784, 281, 1141, 18, 1, 329, 620, 43, 31, 684$$

se insertarían en el orden indicado en celdas inicialmente vacías indexadas de 0 a 12.

Sección 2.3

9. Para la sucesión a definida por $a_n = 2n + 2$, encuentre

a) a_6

b) $\displaystyle\sum_{i=1}^{3} a_i$

c) $\displaystyle\prod_{i=1}^{3} a_i$

d) una fórmula para una subsucesión de a obtenida al seleccionar cada tercer término de a comenzando con el primero.

10. Rescriba la suma

$$\sum_{i=1}^{n} (n - i)r^i$$

sustituyendo el índice i por k, donde $i = k + 2$.

11. Sea

$$b_n = \sum_{i=1}^{n} (i+1)^2 - i^2.$$

a) Encuentre b_5 y b_{10}.

b) Encuentre una fórmula para b_n.

c) ¿Es b creciente?

d) ¿Es b decreciente?

12. Sea $\alpha = ccddc$ y $\beta = c^3d^2$. Encuentre

a) $\alpha\beta$

b) $\beta\alpha$

c) $|\alpha|$

d) $|\alpha\alpha\beta\alpha|$

Ejercicios para computadora

En los ejercicios 1 al 6, suponga que un conjunto X de n elementos está representado como un arreglo A de tamaño al menos n + 1. Los elementos de X se listan de forma consecutiva en A comenzando con la primera posición y terminando con 0. Suponga además que ningún conjunto contiene al 0.

1. Escriba un programa para representar los conjuntos $X \cup Y$, $X \cap Y$, $X - Y$ y $X \triangle Y$, dados los arreglos que representan a X y Y. (La diferencia simétrica se denota por $\triangle$).

2. Escriba un programa para determinar si $X \subseteq Y$, dados los arreglos que representan a X y Y.

3. Escriba un programa para determinar si $X = Y$, dados los arreglos que representan a X y Y.

4. Suponga un universo representado como un arreglo, escriba un programa para representar el conjunto $\overline{X}$, dado el arreglo que representa a X.

5. Dado un elemento E en el arreglo de A que representa a X, escriba un programa que determine si $E \in X$.

6. Dado el arreglo que representa a X, escriba un programa que liste todos los subconjuntos de X.

7. Implemente un sistema de *hash* (utilizando la función de dispersión) para almacenar enteros en un arreglo.

8. Escriba un programa que determine si un ISBN (*International Standard Book Number*) es válido.

9. Escriba un programa que genere números seudoaleatorios.

En los ejercicios 10 al 15, suponga que una sucesión de $\{1, \ldots, n\}$ *a los números reales se representa como un arreglo A, indexado de 1 a n.*

10. Escriba un programa que pruebe si A es uno a uno.

11. Escriba un programa que pruebe si A es sobre en un conjunto dado.

12. Escriba un programa que pruebe si A es creciente.

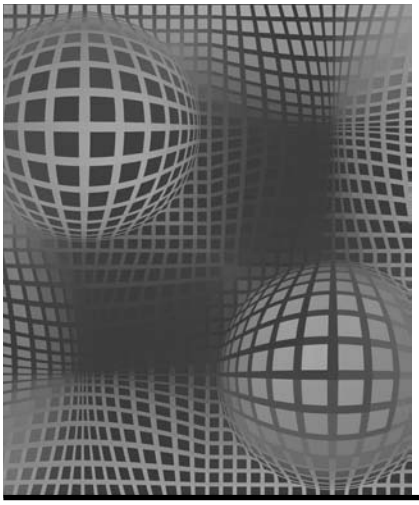
13. Escriba un programa que pruebe si A es decreciente.

14. Escriba un programa que pruebe si A es no creciente.

15. Escriba un programa que pruebe si A es no decreciente.

16. Escriba un programa para determinar si una sucesión es una subsucesión de otra secuencia.

17. Escriba un programa para determinar si una cadena es una subcadena de otra cadena.



Capítulo 3

RELACIONES

¿Una carcasa verde? ¡Por fin un color con el que me puedo relacionar!

DE *EL HOMBRE INCREÍBLE*

Las relaciones generalizan el concepto de funciones. La presencia del par ordenado (a, b) en una relación se interpreta como que existe una relación de a a b. El modelo de base de datos relacional que ayuda a los usuarios a tener acceso a la información de una base de datos (una colección de registros manejados por una computadora) se basa en el concepto de relación.

3.1 ➜ Relaciones

WWW

Se puede pensar en una **relación** de un conjunto a otro como en una tabla que lista los elementos del primer conjunto que se relacionan con los elementos del segundo conjunto. La tabla 3.1.1 muestra qué estudiantes están inscritos en cuáles cursos. Por ejemplo, Guillermo toma Computación y Arte, y María toma Matemáticas. En la terminología de las relaciones, se dice que Guillermo está relacionado con Computación y Arte, y que María está relacionada con Matemáticas.

Por supuesto, la tabla 3.1.1 en realidad es sólo un conjunto de pares ordenados. De manera abstracta, se *define* una relación como un conjunto de pares ordenados. En este contexto, se considera que el primer elemento del par ordenado está relacionado con el segundo elemento del par ordenado.

TABLA 3.1.1 ■ Relación de estudiantes con cursos

Estudiante	*Curso*
Guillermo	Computación
María	Matemáticas
Guillermo	Arte
Beatriz	Historia
Beatriz	Computación
David	Matemáticas

† Esta sección se puede omitir sin pérdida de continuidad.

Definición 3.1.1 ▶

Una relación (binaria) R de un conjunto X a un conjunto Y es un subconjunto del producto cartesiano $X \times Y$. Si $(x, y) \in R$, se escribe $x\ R\ y$, y se dice que *x está relacionada con y*. Si $X = Y$, R se llama *relación (binaria) sobre X*.

El conjunto

$$\{x \in X \mid (x, y) \in R \text{ para alguna } y \in Y\}$$

se llama *dominio* de R. El conjunto

$$\{y \in Y \mid (x, y) \in R \text{ para alguna } x \in X\}$$

se llama *rango* de R. ◀

Una función (vea la sección 2.2) es un tipo especial de relación. Una función f de X a Y es una relación de X a Y que tiene las propiedades:

a) El dominio de f es igual X.
b) Para cada $x \in X$, existe exactamente una $y \in Y$ tal que $(x, y) \in f$.

Ejemplo 3.1.2 ▶

Si

$$X = \{\text{Guillermo, María, Beatriz, David}\}$$

y

$$Y = \{\text{Computación, Matemáticas, Arte, Historia}\},$$

la relación R de la tabla 3.1.1 se puede escribir

$$R = \{(\text{Guillermo, Computación}), (\text{María, Matemáticas}),\ (\text{Guillermo, Arte}), (\text{Beatriz, Historia}),\ (\text{Beatriz, Computación}), (\text{David, Matemáticas})\}.$$

Como (Beatriz, Historia) $\in R$, se puede escribir Beatriz R Historia. El dominio (primera columna) de R es el conjunto X y el rango (segunda columna) de R es el conjunto Y. ◀

El ejemplo 3.1.2 muestra que es posible establecer una relación con sólo especificar qué pares ordenados pertenecen a la relación. El siguiente ejemplo indica que algunas veces es posible definir una relación dando la regla para pertenecer a la relación.

Ejemplo 3.1.3 ▶

Sea

$$X = \{2, 3, 4\} \quad \text{y} \quad Y = \{3, 4, 5, 6, 7\}.$$

Si se define una relación R de X a Y por

$$(x, y) \in R \quad \text{si } x \text{ divide a } y,$$

se obtiene

$$R = \{(2, 4), (2, 6), (3, 3), (3, 6), (4, 4)\}$$

Si se rescribe R como tabla, se obtiene

X	Y
2	4
2	6
3	3
3	6
4	4

El dominio de R es el conjunto $\{2, 3, 4\}$ y el rango de R es el conjunto $\{3, 4, 6\}$. ◀

Ejemplo 3.1.4 ▶

Sea R la relación sobre $X = \{1, 2, 3, 4\}$ definida por $(x, y) \in R$ si $x \leq y$, $x, y \in X$. Entonces

$$R = \{(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 3), (2, 4), (3, 3), (3, 4), (4, 4)\}.$$

El dominio y rango de R son ambos iguales a X. ◀

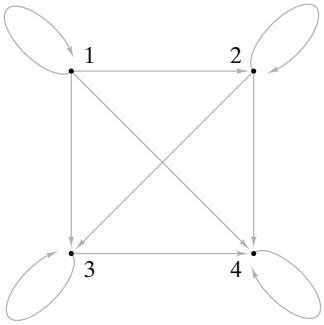


Figura 3.1.1 Digráfica de la relación del ejemplo 3.1.4.

Una manera informativa de visualizar una relación en un conjunto es dibujar su **digráfica.** (Las digráficas se estudiarán con más detalle en el capítulo 8. Por ahora, se mencionan las digráficas sólo en lo que se refiere a las relaciones). Para dibujar la digráfica de una relación en un conjunto X, primero dibujamos puntos o **vértices** para representar los elementos de X. En la figura 3.1.1 se dibujaron cuatro vértices para representar los elementos del conjunto X del ejemplo 3.1.4. Después, si el elemento (x, y) está en la relación, se dibuja una flecha (llamada **arista dirigida**) de x a y. En la figura 3.1.1, se dibujaron aristas dirigidas para representar los miembros de la relación R del ejemplo 3.1.4. Observe que un elemento de la forma (x, x) en una relación corresponde a una arista dirigida de x a x. Tales aristas se llaman **lazos**. Existe un lazo en todos los vértices de la figura 3.1.1.

Ejemplo 3.1.5 ▶

La relación R sobre $X = \{a, b, c, d\}$ dada por la digráfica de la figura 3.1.2 es

$$R = \{(a, a), (b, c), (c, b), (d, d)\}$$

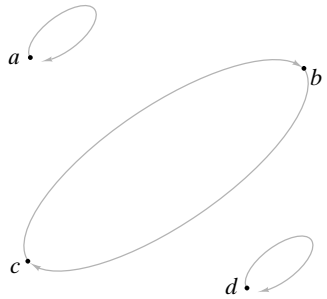


Figura 3.1.2 Digráfica de la relación del ejemplo 3.1.5.

◀

A continuación se definirán varias propiedades de las relaciones.

Definición 3.1.6 ▶

Una relación R en un conjunto X se llama *reflexiva* si $(x, x) \in R$ para toda $x \in X$. ◀

Ejemplo 3.1.7 ▶

La relación R sobre $X = \{1, 2, 3, 4\}$ definida por $(x, y) \in R$ si $x \leq y$, $x, y \in X$, es reflexiva porque cada elemento $x \in X$, $(x, x) \in R$; en particular, (1, 1), (2, 2), (3, 3) y (4, 4) están en R. La digráfica de una relación reflexiva tiene un lazo en cada vértice. Observe que la digráfica de esta relación (figura 3.1.1) tiene un lazo en cada vértice. ◀

Ejemplo 3.1.8 ▶

La relación

$$R = \{(a, a), (b, c), (c, b), (d, d)\}$$

sobre $X = \{a, b, c, d\}$ no es reflexiva. Por ejemplo, $b \in X$, pero $(b, b) \notin R$. El hecho de que esta relación no sea reflexiva también se observa en su digráfica (figura 3.1.2); el vértice b no tiene lazo. ◀

Definición 3.1.9 ▶

Una relación R sobre un conjunto X se llama *simétrica* si para toda $x, y \in X$, si $(x, y) \in R$, entonces $(y, x) \in R$. ◀

Ejemplo 3.1.10 ▶

La relación

$$R = \{(a, a), (b, c), (c, b), (d, d)\}$$

sobre $X = \{a, b, c, d\}$ es simétrica porque para toda x, y, si $(x, y) \in R$, entonces $(y, x) \in R$. Por ejemplo, (b, c) está en R y (c, b) también está en R. La digráfica de una relación simétrica tiene la propiedad de que siempre que existe una arista dirigida de v a w, también existe una arista dirigida de w a v. Note que la digráfica de la relación (figura 3.1.2) tiene la propiedad de que para toda arista dirigida de v a w, también existe la arista dirigida de w a v. ◀

Ejemplo 3.1.11 ▶

La relación R sobre $X = \{1, 2, 3, 4\}$ definida por $(x, y) \in R$ si $x \leq y$, $x, y \in X$, es no simétrica. Por ejemplo, $(2, 3) \in R$, pero $(3, 2) \notin R$. La digráfica de esta relación (figura 3.1.1) tiene una arista dirigida de 2 a 3, pero no de 3 a 2. ◀

Definición 3.1.12 ▶

Una relación R en un conjunto X se llama *antisimétrica* si para toda $x, y \in X$, si $(x, y) \in R$ y $x \neq y$, entonces $(y, x) \notin R$. ◀

Ejemplo 3.1.13 ▶

La relación R sobre $X = \{1, 2, 3, 4\}$ definida por $(x, y) \in R$ si $x \leq y$, $x, y \in X$, es antisimétrica porque para toda x, y, si $(x, y) \in R$ y $x \neq y$, entonces $(y, x) \notin R$. Por ejemplo, $(1, 2) \in R$, pero $(2, 1) \notin R$. La digráfica de una relación antisimétrica tiene la propiedad de que entre cualesquiera dos vértices existe a lo sumo una arista dirigida. Observe que la digráfica de esta relación (figura 3.1.1) tiene a lo sumo una arista dirigida entre cada par de vértices. ◀

Ejemplo 3.1.14 ▶

La relación

$$R = \{(a, a), (b, c), (c, b), (d, d)\}$$

sobre $X = \{a, b, c, d\}$ no es antisimétrica porque (b, c) y (c, b) están ambos en R. Observe que en la digráfica de esta relación (figura 3.1.2) hay dos aristas dirigidas entre b y c. ◀

Ejemplo 3.1.15 ▶

Si la relación *no* tiene miembros de la forma (x, y), $x \neq y$, entonces

$$\text{si } (x, y) \in R \text{ y } x \neq y, \text{ entonces } (y, x) \notin R$$

es vagamente cierto para toda $x, y \in X$ [porque $(x, y) \in R$ y $x \neq y$ es falsa para toda $x, y \in X$]. Por lo tanto, si una relación R *no* tiene miembros de la forma (x, y), $x \neq y$, entonces R es antisimétrica. Por ejemplo,

$$R = \{(a, a), (b, b), (c, c)\}$$

Figura 3.1.3 Digráfica de la relación del ejemplo 3.1.15.

sobre $X = \{a, b, c\}$ es antisimétrica. La digráfica de R mostrada en la figura 3.1.3 tiene a lo sumo una arista dirigida entre cada par de vértices. Note que R también es reflexiva y simétrica. Este ejemplo muestra que "antisimétrica" no es lo mismo que "no simétrica" porque esta relación, de hecho, es simétrica y antisimétrica. ◀

Definición 3.1.16 ▶

Una relación R en un conjunto X se llama *transitiva* si para toda $x, y, z \in X$, si (x, y) y $(y, z) \in R$, entonces $(x, z) \in R$. ◀

Ejemplo 3.1.17 ▶

La relación R sobre $X = \{1, 2, 3, 4\}$ definida por $(x, y) \in R$ si $x \leq y$, $x, y \in X$, es transitiva porque para todo x, y, z, si (x, y) y $(y, z) \in R$, entonces $(x, z) \in R$. Para verificar de manera formal que esta relación satisface la definición 3.1.16, se pueden listar todos los pares de la

forma (x, y) y (y, z) en R y comprobar que en cada caso $(x, z) \in R$.

Pares de la forma			*Pares de la forma*		
(x, y)	(y, z)	(x, z)	(x, y)	(y, z)	(x, z)
(1, 1)	(1, 1)	(1, 1)	(2, 2)	(2, 2)	(2, 2)
(1, 1)	(1, 2)	(1, 2)	(2, 2)	(2, 3)	(2,3)
(1, 1)	(1, 3)	(1, 3)	(2, 2)	(2, 4)	(2, 4)
(1, 1)	(1, 4)	(1, 4)	(2, 3)	(3, 3)	(2, 3)
(1, 2)	(2, 2)	(1, 2)	(2, 3)	(3, 4)	(2, 4)
(1, 2)	(2, 3)	(1, 3)	(2, 4)	(4, 4)	(2, 4)
(1, 2)	(2, 4)	(1, 4)	(3, 3)	(3, 3)	(3, 3)
(1, 3)	(3, 3)	(1, 3)	(3, 3)	(3, 4)	(3, 4)
(1, 3)	(3, 4)	(1, 4)	(3, 4)	(4, 4)	(3, 4)
(1, 4)	(4, 4)	(1, 4)	(4, 4)	(4, 4)	(4, 4)

En realidad, algunos elementos de la tabla anterior eran innecesarios. Si $x = y$ o $y = z$, no se necesita una verificación explícita de que la condición

$$\text{si } (x, y) \text{ y } (y, z) \in R, \text{ entonces } (x, z) \in R$$

se satisface, ya que será verdadera de modo automático. Suponga, por ejemplo, que $x = y$, y (x, y) y (y, z) están en R. Como $x = y$, $(x, z) = (y, z)$ está en R y la condición se cumple. Al eliminar los casos $x = y$ y $y = z$ sólo los siguientes deben comprobarse de manera explícita para verificar que la relación es transitiva:

Pares de la forma		
(x, y)	(y, z)	(x, z)
(1, 2)	(2, 3)	(1, 3)
(1, 2)	(2, 4)	(1, 4)
(1, 3)	(3, 4)	(1, 4)
(2, 3)	(3, 4)	(2, 4)

La digráfica de una relación transitiva tiene la propiedad de que siempre que haya aristas dirigidas de x a y y de y a z, también habrá una arista dirigida de x a z. Observe que la digráfica de esta relación (figura 3.1.1) tiene esta propiedad. ◀

Ejemplo 3.1.18 ▶

La relación

$$R = \{(a, a), (b, c), (c, b), (d, d)\}$$

sobre $X = \{a, b, c, d\}$ no es transitiva. Por ejemplo, (b, c) y (c, b) están en R, pero (b, b) no está en R. Observe que en la digráfica de esta relación (figura 3.1.2) hay aristas dirigidas de b a c y de c a b, pero no hay una arista dirigida de b a b. ◀

Las relaciones resultan útiles para ordenar los elementos de un conjunto. Por ejemplo, la relación R definida en el conjunto de enteros por

$$(x, y) \in R \quad \text{si } x \leq y$$

ordena los enteros. Advierta que la relación R es reflexiva, antisimétrica y transitiva. Este tipo de relación se llama **orden parcial**.

Definición 3.1.19 ▶

Una relación R en un conjunto X se llama *orden parcial* si R es reflexiva, antisimétrica y transitiva. ◀

Ejemplo 3.1.20 ▶

Como la relación R definida en los enteros positivos por

$$(x, y) \in R \qquad \text{si } x \text{ divide a } y$$

es reflexiva, antisimétrica y transitiva, R es un orden parcial. ◀

Si R es un orden parcial en un conjunto X, la notación $x \preceq y$ se usa algunas veces para indicar que $(x, y) \in R$. Esta notación sugiere que estamos interpretando la relación como una ordenación de los elementos de X.

Suponga que R es una relación de orden parcial en un conjunto X. Si $x, y \in X$ y ya sea $x \preceq y$ o $y \preceq x$, , se dice que x y y son **comparables.** Si $x, y \in X$ y $x \not\preceq y$ y $y \not\preceq x$, se dice que x y y son **incomparables.** Si todo par de elementos de X es comparable, se llama a R de **orden total.** La relación menor o igual que en los enteros positivos es de orden total, puesto que si x y y son enteros, $x \leq y$ o bien $y \leq x$. La razón para el término "orden parcial" es que, en general, algunos elementos de X pueden ser incomparables. La relación "divide" en los enteros positivos (vea el ejemplo 3.1.20) tiene elementos comparables e incomparables. Por ejemplo, 2 y 3 son incomparables (porque 2 no divide a 3 y 3 no divide a 2), pero 3 y 6 son comparables (ya que 3 divide a 6).

Una aplicación de orden parcial es la programación de tareas.

Ejemplo 3.1.21 ▶

Programación de tareas

Considere un conjunto T de tareas que deben realizarse para tomar fotos de interiores con flash con una cámara específica.

1. Retire la tapa del lente.
2. Enfoque la cámara.
3. Quite el seguro.
4. Encienda la unidad de flash.
5. Oprima el botón de disparo.

Algunas tareas deben realizarse antes que otras. Por ejemplo, la tarea 1 debe efectuarse antes que la tarea 2. Por otro lado, otras tareas se pueden realizar en cualquier orden, como por ejemplo, las tareas número 2 y 3.

La relación R definida en T por

$$i \, R \, j \qquad \text{si } i = j \text{ o la tarea } i \text{ debe hacerse antes que la } j$$

ordena las tareas. Se tiene

$$R = \{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (1, 2), (1, 5), (2, 5), (3, 5), (4, 5)\}.$$

Como R es reflexiva, antisimétrica y transitiva, es una de orden parcial. Una solución al problema de programar tareas con el fin de tomar una foto es una ordenación total de las tareas congruente con un orden parcial. De manera más precisa, se requiere un orden total de las tareas

$$t_1, \quad t_2, \quad t_3, \quad t_4, \quad t_5$$

tal que si $t_i \, R \, t_j$, entonces $i = j$ o t_i precede a t_j en la lista. Entre las soluciones se tiene

$$1, 2, 3, 4, 5$$

y

$$3, 4, 1, 2, 5.$$ ◀

Dada una relación R de X a Y, es posible definir una relación de Y a X invirtiendo el orden de cada par ordenado en R. La relación inversa generaliza la función inversa. La definición formal es la siguiente.

Definición 3.1.22 ▶

Sea R una relación de X a Y. La *inversa* de R, denotada por R^{-1}, es la relación de Y a X definida por

$$R^{-1} = \{(y, x) \mid (x, y) \in R\}.$$ ◀

Ejemplo 3.1.23 ▶

Si se define una relación R de $X = \{2, 3, 4\}$ a $Y = \{3, 4, 5, 6, 7\}$ por

$$(x, y) \in R \quad \text{si } x \text{ divide a } y,$$

se obtiene

$$R = \{(2, 4), (2, 6), (3, 3), (3, 6), (4, 4)\}.$$

El inverso de esta relación es

$$R^{-1} = \{(4, 2), (6, 2), (3, 3), (6, 3), (4, 4)\}.$$

En palabras, esta relación se describe como "es divisible entre". ◀

Si se tiene una relación R_1 de X a Y y una relación R_2 de Y a Z, se puede formar la composición de las relaciones aplicando primero la relación R_1 y después la relación R_2. La composición de las relaciones generaliza la composición de funciones. La definición formal es la siguiente.

Definición 3.1.24 ▶

Sea R_1 una relación de X a Y y R_2 una relación de Y a Z. La *composición* de R_1 y R_2, denotada por $R_2 \circ R_1$, es la relación de X a Z definida por

$$R_2 \circ R_1 = \{(x, z) \mid (x, y) \in R_1 \quad \text{y } (y, z) \in R_2 \quad \text{para alguna } y \in Y\}.$$ ◀

Ejemplo 3.1.25 ▶

La composición de las relaciones

$$R_1 = \{(1, 2), (1, 6), (2, 4), (3, 4), (3, 6), (3, 8)\}$$

y

$$R_2 = \{(2, u), (4, s), (4, t), (6, t), (8, u)\}$$

es

$$R_2 \circ R_1 = \{(1, u), (1, t), (2, s), (2, t), (3, s), (3, t), (3, u)\}.$$

Por ejemplo, $(1, u) \in R_2 \circ R_1$ porque $(1, 2) \in R_1$ y $(2, u) \in R_2$. ◀

Sugerencias para resolver problemas

Una relación R en un conjunto X es *reflexiva* si $(x, x) \in R$ para toda $x \in X$. En palabras, una relación es reflexiva si cada elemento en su dominio está relacionado consigo mismo. Para verificar si una relación es reflexiva, sólo se comprueba si (x, x) está presente en R para toda x. Dado un diagrama de flechas, la relación es reflexiva si tiene un lazo en cada vértice.

Una relación R en un conjunto X es *simétrica* si para toda $x, y \in X$, si $(x, y) \in R$, entonces $(y, x) \in R$. En palabras, una relación es simétrica si siempre que x está relacionada con y, ocurre que y está relacionada con x. Para verificar si una relación es simétrica, se ve si para cada miembro (x, y) en R, (y, x) también está presente. Dado un diagrama de flechas, la relación es simétrica si siempre que haya una arista dirigida de x a y, también hay una arista dirigida de y a x.

Una relación R en un conjunto X es *antisimétrica* si para toda $x, y \in X$, si $(x, y) \in R$ y $x \neq y$, entonces $(y, x) \notin R$. En palabras, una relación es antisimétrica si siempre que x

está relacionada con y y x y y son diferentes, entonces y *no* está relacionada con x. Para comprobar si una relación es antisimétrica, se verifica cada miembro (x, y), $x \neq y$, y se ve si (y, x) *no* está presente. Dado un diagrama de flechas, la relación es antisimétrica si cuando hay una arista dirigida de x a y , $x \neq y$, *no* hay una arista dirigida de y a x. Advierta que "no simétrica" no necesariamente es lo mismo que "antisimétrica".

Una relación R en un conjunto X es *transitiva* si para toda x , y, $z \in X$, si (x, y) y $(y, z) \in R$, entonces $(x, z) \in R$. En palabras, una relación es transitiva si siempre que x está relacionada con y y y está relacionada con z, entonces x está relacionada con z. Para comprobar si una relación es transitiva, se verifican todos los pares de la forma (x, y), (y, z) con $x \neq y$ y $y \neq z$, después se ve si (x, z) también está presente. Dado un diagrama de flechas, las relación es transitiva si cuando hay aristas dirigidas de x a y y de y a z, también hay una arista dirigida de x a z.

Un *orden parcial* está en una relación reflexiva, antisimétrica y transitiva.

La *inversa* R^{-1} de la relación R consiste en los elementos (y, x), donde $(x, y) \in R$. En palabras, x está relacionada con y en R si y sólo si y está relacionada con x en R^{-1}.

Si R_1 es una relación de X a Y y R_2 es una relación de Y a Z, la *composición* de R_1 y R_2, denotada por $R_2 \circ R_1$, es la relación de X a Z definida por

$$R_2 \circ R_1 = \{(x, z) \mid (x, y) \in R_1 \text{ y } (y, z) \in R_2 \text{ para alguna } y \in Y\}.$$

Para calcular la composición, se encuentran todos los pares de la forma $(x, y) \in R_1$ y $(y, z) \in R_2$; después se encuentran (x, z) en $R_2 \circ R_1$.

Sección de ejercicios de repaso

1. ¿Qué es una relación binaria de X a Y?
2. ¿Qué es el dominio de una relación binaria?
3. ¿Qué es el rango de una relación binaria?
4. ¿Qué es la digráfica de una relación binaria?
5. Defina *relación reflexiva*. Dé un ejemplo de una relación reflexiva. Dé un ejemplo de una relación que *no* sea reflexiva.
6. Defina *relación simétrica*. Dé un ejemplo de una relación simétrica. Dé un ejemplo de una relación *no* simétrica.
7. Defina *relación antisimétrica*. Dé un ejemplo de una relación antisimétrica. Dé un ejemplo de una relación que *no* sea antisimétrica.
8. Defina *relación transitiva*. Dé un ejemplo de una relación transitiva. Dé un ejemplo de una relación que *no* sea transitiva.
9. Defina *orden parcial* y dé un ejemplo de orden parcial.
10. Defina una *relación inversa* y dé un ejemplo de una relación inversa.
11. Defina *composición de relaciones* y dé un ejemplo de composición de relaciones.

Ejercicios

En los ejercicios 1 al 4, escriba la relación como un conjunto de pares ordenados.

1.

8840	Martillo
9921	Tenazas
452	Pintura
2207	Alfombra

2.

a	3
b	1
b	4
c	1

3.

Susana	Matemáticas
Ruth	Física
Samuel	Economía

4.

a	a
b	b

En los ejercicio 5 al 8, escriba la relación como tabla.

5. $R = \{(a, 6), (b, 2), (a, 1), (c, 1)\}$
6. $R = \{\{\text{Rogelio, Música}\}, (\text{Patricia, Historia}), (\text{Benjamín, Matemáticas}), (\text{Patricia, Ciencias Políticas})\}$
7. La relación R en $\{1, 2, 3, 4\}$ definida por $(x, y) \in R$ if $x^2 \geq y$
8. La relación R del conjunto X de planetas al conjunto Y de enteros definida por $(x, y) \in R$ si x está en la posición y respecto al sol (el más cercano al sol está en la posición 1, el segundo más cercano al sol está en la posición 2, y así sucesivamente).

En los ejercicios 9 al 12 dibuje la digráfica de la relación.

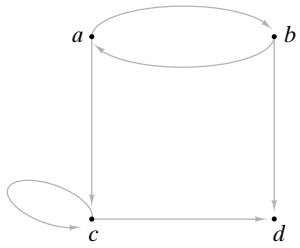
9. La relación del ejercicio 4 en $\{a, b, c\}$
10. La relación $R = \{(1, 2), (2, 1), (3, 3), (1, 1), (2, 2)\}$ sobre $X = \{1, 2, 3\}$

11. La relación $R = \{(1, 2), (2, 3), (3, 4), (4, 1)\}$ en $\{1, 2, 3, 4\}$

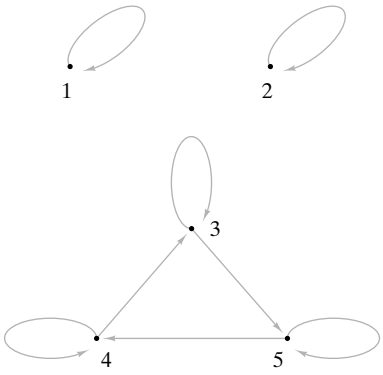
12. La relación del ejercicio 7

En los ejercicios 13 al 16, escriba la relación como un conjunto de pares ordenados.

13.



14.



15.

16.

17. Encuentre el dominio y el rango de cada relación en los ejercicios 1 al 16.

18. Encuentre la inversa (como conjunto de pares ordenados) de cada relación en los ejercicios 1 al 16.

Los ejercicios 19 al 24 se refieren a la relación R en el conjunto $\{1, 2, 3, 4, 5\}$ *definida por la regla* $(x, y) \in R$ *si 3 divide a* $x - y$.

19. Liste los elementos de R.

20. Liste los elementos de R^{-1}.

21. Encuentre el dominio de R.

22. Encuentre el rango de R.

23. Encuentre el dominio de R^{-1}.

24. Encuentre el rango de R^{-1}.

25. Repita los ejercicios 19 al 24 para la relación R en el conjunto $\{1, 2, 3, 4, 5\}$ definida por la regla $(x, y) \in R$ si $x + y \leq 6$.

26. Repita los ejercicios 19 al 24 para la relación R en el conjunto $\{1, 2, 3, 4, 5\}$ definida por la regla $(x, y) \in R$ si $x = y - 1$.

27. La relación del ejercicio 25, ¿es reflexiva, simétrica, antisimétrica, transitiva y/o de un orden parcial?

28. ¿La relación del ejercicio 26 es reflexiva, simétrica, antisimétrica, transitiva y/o de un orden parcial?

En los ejercicios 29 al 34, determine si cada relación definida en el conjunto de enteros positivos es reflexiva, simétrica, antisimétrica, transitiva y/o de un orden parcial.

29. $(x, y) \in R$ si $x = y^2$.

30. $(x, y) \in R$ si $x > y$.

31. $(x, y) \in R$ si $x \geq y$.

32. $(x, y) \in R$ si $x = y$.

33. $(x, y) \in R$ si 3 divides $x - y$.

34. $(x, y) \in R$ si 3 divides $x + 2y$.

35. Sea X un conjunto no vacío. Defina la relación en $\mathcal{P}(X)$, el conjunto potencia de X, como $(A, B) \in R$ si $A \subseteq B$. ¿Es ésta una relación reflexiva, simétrica, antisimétrica, transitiva y/o de un orden parcial?

36. Sea X el conjunto de todas las cadenas de 4 bits (por ejemplo, 0011, 0101, 1000). Defina una relación R sobre X como $s_1 \, R \, s_2$ si alguna subcadena s_1 de longitud 2 es igual a alguna subcadena s_2 de longitud 2. Ejemplo: 0111 R 1010 (porque ambas 0111 y 1010 contienen 01). 1110 $\not R$ 0001 (porque 1110 y 0001 no tienen una subcadena común de longitud 2). ¿Es ésta una relación reflexiva, simétrica, antisimétrica, transitiva y/o de un orden parcial?

37. Suponga que R_i es de orden parcial sobre X_i, $i = 1, 2$. Demuestre que R es de orden parcial en $X_1 \times X_2$ si se define

$$(x_1, x_2) \, R \, (x_1', x_2') \qquad \text{si } x_1 \, R_1 \, x_1' \text{ y } x_2 \, R_2 \, x_2'.$$

38. Sean R_1 y R_2 las relaciones en $\{1, 2, 3, 4\}$ dadas por

$$R_1 = \{(1, 1), (1, 2), (3, 4), (4, 2)\}$$
$$R_2 = \{(1, 1), (2, 1), (3, 1), (4, 4), (2, 2)\}.$$

Liste los elementos de $R_1 \circ R_2$ y $R_2 \circ R_1$.

Proporcione ejemplos de relaciones en $\{1, 2, 3, 4\}$ *que tengan las propiedades especificadas en los ejercicios 39 al 43.*

39. Reflexiva, simétrica, y no transitiva.

40. Reflexiva, no simétrica, y no transitiva.

41. Reflexiva, antisimétrica, y no transitiva.

42. No reflexiva, simétrica, no antisimétrica y transitiva.

43. No reflexiva, no simétrica, y transitiva.

Sean R y S relaciones sobre X. Determine si cada afirmación en los ejercicios 44 al 59 es verdadera o falsa. Si la afirmación es verdadera, demuéstrelo; de otra manera, dé un contraejemplo.

44. Si R y S son transitivas, entonces $R \cup S$ es transitiva.

45. Si R y S son transitivas, entonces $R \cap S$ es transitiva.

46. Si R y S son transitivas, entonces $R \circ S$ es transitiva.

47. Si R es transitiva, entonces R^{-1} es transitiva.

48. Si R y S son reflexivas, entonces $R \cup S$ es reflexiva.

49. Si R y S son reflexivas, entonces $R \cap S$ es reflexiva.

50. Si R y S son reflexivas, entonces $R \circ S$ es reflexiva.

51. Si R es reflexiva, entonces R^{-1} es reflexiva.

52. Si R y S son simétricas, entonces $R \cup S$ es simétrica.

53. Si R y S son simétricas, entonces $R \cap S$ es simétrica.

54. Si R y S son simétricas, entonces $R \circ S$ es simétrica.

55. Si R es simétrica, entonces R^{-1} es simétrica.

56. Si R y S son antisimétricas, entonces $R \cup S$ es antisimétrica.

57. Si R y S son antisimétricas, entonces $R \cap S$ es antisimétrica.

58. Si R y S son antisimétricas, entonces $R \circ S$ es antisimétrica.

59. Si R es antisimétrica, entonces R^{-1} es antisimétrica.

En los ejercicios 60 al 62, determine si cada relación R definida en la colección de todos los subconjuntos no vacíos de números reales es reflexiva, simétrica, antisimétrica, transitiva y/o de orden parcial.

60. $(A, B) \in R$ si para toda $\epsilon > 0$, existen $a \in A$ y $b \in B$ con $|a - b| < \epsilon$.

61. $(A, B) \in R$ si para toda $a \in A$ y $\epsilon > 0$, existe $b \in B$ con $|a - b| < \epsilon$.

62. $(A, B) \in R$ si para toda $a \in A$, b $\in B$ y $\epsilon > 0$, existen $a' \in A$ y $b' \in B$ con $|a - b'| <$? y $|a' - b| < \epsilon$.

63. ¿Qué está equivocado en el siguiente argumento, que se supone demuestra que cualquier relación R sobre X que es simétrica y transitiva es reflexiva?

Sea $x \in X$. Usando la simetría, se tiene que (x, y) y (y, x) están ambos en R. Como $(x, y), (y, x) \in R$, por la transitividad se tiene $(x, x) \in R$. Por lo tanto, R es reflexiva.

3.2 ➜ Relaciones de equivalencia

WWW

Suponga que se tiene un conjunto X de 10 pelotas, cada una de las cuales es roja, azul o verde (vea la figura 3.2.1). Si se dividen las pelotas en los conjuntos R, A y V de acuerdo con el color, la familia $\{R, A, V\}$ es una partición de X. (Recuerde que en la sección 2.1 se definió una partición de un conjunto X como la colección $\mathcal{S}$ de subconjuntos no vacíos de X tales que cada elemento en X pertenece exactamente a un miembro de $\mathcal{S}$).

Una partición es útil para definir una relación. Si $\mathcal{S}$ es una partición de X, se puede definir $x\ R\ y$ de modo que signifique que para algún conjunto $S \in \mathcal{S}$, tanto x como y pertenecen a S. Para el ejemplo de la figura 3.2.1, la relación obtenida se describe como "es del mismo color que". El siguiente teorema muestra que este tipo de relación siempre es reflexiva, simétrica y transitiva.

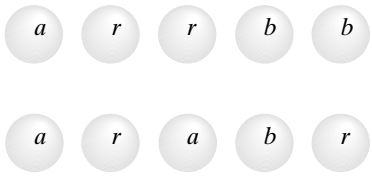


Figura 3.2.1 Conjunto de pelotas de colores.

Teorema 3.2.1

Sea $\mathcal{S}$ una partición de un conjunto X. Defina x R y de modo que signifique que para algún conjunto S en $\mathcal{S}$, tanto x como y pertenecen a S. Entonces R es reflexiva, simétrica y transitiva.

Demostración Sea $x \in X$. Por definición de partición, x pertenece a algún conjunto $S \in \mathcal{S}$. Entonces, $x\ R\ x$ y R es reflexiva.

Suponga que $x\ R\ y$. Entonces ambas x y y pertenecen a algún conjunto $S \in \mathcal{S}$. Como ambos y y x pertenecen a S, $y\ R\ x$ y R son simétricas.

Por último, suponga que $x\ R\ y$ y $y\ R\ z$. Entonces ambos x y y pertenecen a algún conjunto $S \in \mathcal{S}$ y ambos y y z pertenecen a algún conjunto $T \in \mathcal{S}$. Como y pertenece exactamente a un miembro de $\mathcal{S}$, debemos tener $S = T$. Por lo tanto, ambas x y z están en S y $x\ R\ z$. Se ha demostrado que R es transitiva.

Ejemplo 3.2.2 ▶

Considere la partición

$$\mathcal{S} = \{\{1, 3, 5\}, \{2, 6\}, \{4\}\}$$

de $X = \{1, 2, 3, 4, 5, 6\}$. La relación R sobre X dada por el teorema 3.2.1 contiene los pares ordenados (1, 1), (1, 3) y (1, 5) porque $\{1, 3, 5\}$ está en $\mathcal{S}$. La relación completa es

$$R = \{(1, 1), (1, 3), (1, 5), (3, 1), (3, 3), (3, 5), (5, 1), (5, 3), (5, 5), (2, 2), (2, 6), (6, 2), (6, 6), (4, 4)\}.$$

◀

Se tienen $\mathcal{S}$ y R como en el teorema 3.2.1. Si $S \in \mathcal{S}$, los miembros de S se pueden ver como equivalentes en el sentido de la relación R, que es la motivación para llamar **relaciones equivalentes** a las relaciones que son reflexivas, simétricas y transitivas. En el ejemplo 3.2.1, se tiene la relación "es del mismo color que"; entonces *equivalente* significa "es del mismo color que". Cada conjunto en la partición consiste en todas las pelotas de un color en particular.

Definición 3.2.3 ▶ Una relación que es reflexiva, simétrica y transitiva en un conjunto X se llama *relación de equivalencia sobre* X. ◀

Ejemplo 3.2.4 ▶ La relación R del ejemplo 3.2.2 es una relación equivalente en $\{1, 2, 3, 4, 5, 6\}$ por el Teorema 3.2.1. También se puede verificar directamente que R es reflexiva, simétrica y transitiva.

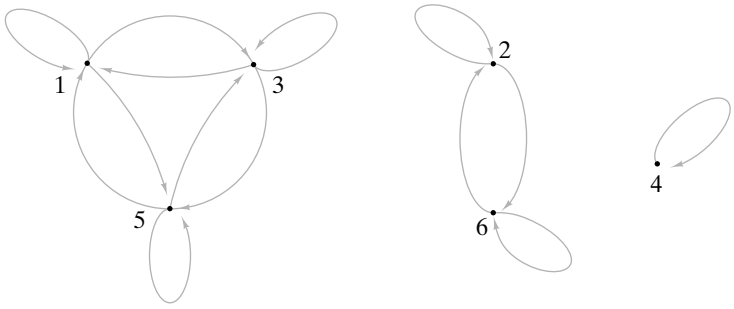


Figura 3.2.2 Digráfica de la relación del ejemplo 3.2.2.

La digráfica de la relación R del ejemplo 3.2.2 se ilustra en la figura 3.2.2. De nuevo, se ve que R es reflexiva (hay un lazo en cada vértice), simétrica (para toda arista dirigida de v a w, también existe una arista dirigida de w a v), y transitiva (si hay una arista dirigida de x a y y una arista dirigida de y a z, existe una arista dirigida de x a z). ◀

Ejemplo 3.2.5 ▶ Considere la relación

$$R = \{(1, 1), (1, 3), (1, 5), (2, 2), (2, 4), (3, 1), (3, 3), (3, 5), (4, 2), (4, 4), (5, 1), (5, 3), (5, 5)\}$$

en $\{1, 2, 3, 4, 5\}$. La relación es reflexiva porque $(1, 1), (2, 2), (3, 3), (4, 4), (5, 5) \in R$. La relación es simétrica porque siempre que (x, y) está en R, (y, x) también está en R. Por último, la relación es transitiva porque siempre que (x, y) y (y, z) están en R, (x, z) también está en R. Como R es reflexiva, simétrica y transitiva, R es una relación de equivalencia en $\{1, 2, 3, 4, 5\}$. ◀

Ejemplo 3.2.6 ▶ La relación R sobre $X = \{1, 2, 3, 4\}$ definida por $(x, y) \in R$ si $x \leq y$, con x y $y \in R$, no es una relación de equivalencia porque R no es simétrica. [Por ejemplo, $(2, 3) \in R$, pero $(3, 2) \notin R$]. La relación R es reflexiva y transitiva. ◀

Ejemplo 3.2.7 ▶ La relación

$$R = \{(a, a), (b, c), (c, b), (d, d)\}$$

sobre $X = \{a, b, c, d\}$ no es una relación de equivalencia porque R no es reflexiva ni transitiva. [No es reflexiva porque, por ejemplo, $(b, b) \notin R$. No es transitiva porque, por ejemplo, (b, c) y (c, b) están en R, pero (b, b) no está en R]. ◀

Dada una relación de equivalencia en un conjunto X, es posible hacer una partición de X agrupando miembros relacionados. Puede pensarse que los elementos relacionados entre sí son equivalentes. El siguiente teorema establece los detalles.

Teorema 3.2.8

Sea R una relación de equivalencia en un conjunto X. Para cada $a \in X$. sea

$$[a] = \{x \in X \mid x \, R \, a\}.$$

(En palabras, [a] es el conjunto de todos los elementos de X que están relacionados con a). Entonces

$$\mathcal{S} = \{[a] \mid a \in X\}$$

es una partición de X.

Demostración Debemos demostrar que todo elemento en X pertenece exactamente a un miembro de $\mathcal{S}$.

Sea $a \in X$. Como $a \, R \, a$, $a \in [a]$. Entonces todo elemento de X pertenece a *al menos un* miembro de $\mathcal{S}$. Falta demostrar que todo elemento de X pertenece a *exactamente* un miembro de $\mathcal{S}$; es decir,

$$\text{si } x \in X \text{ y } x \in [a] \cap [b], \text{ entonces } [a] = [b]. \qquad \textbf{(3.2.1)}$$

Primero probamos que para toda c, $d \in R$, si $c \, R \, d$, entonces $[c] = [d]$. Suponga que $c \, R \, d$. Sea $x \in [c]$. Entonces $x \, R \, c$. Como $c \, R \, d$ y R es transitiva, $x \, R \, d$. Por lo tanto, $x \in [d]$ y $[c] \subseteq [d]$. El argumento de que $[d] \subseteq [c]$ es el mismo que el que se acaba de dar, pero con los papeles de c y d intercambiados. Entonces $[c] = [d]$.

Ahora se prueba (3.2.1). Suponga que $x \in X$ y $x \in [a] \cap [b]$. Entonces $x \, R \, a$ y $x \, R \, b$. El resultado anterior prueba que $[x] = [a]$ y $[x] = [b]$. Por lo tanto, $[a] = [b]$.

Definición 3.2.9 ▶

Sea R una relación de equivalencia en un conjunto X. Los conjuntos $[a]$ definidos en el teorema 3.2.8 se llaman *clases de equivalencia de X dada por la relación R*. ◀

Ejemplo 3.2.10 ▶

En el ejemplo 3.2.4, se demostró que la relación

$$R = \{(1, 1), (1, 3), (1, 5), (3, 1), (3, 3), (3, 5), (5, 1), (5, 3), (5, 5), (2, 2), (2, 6), (6, 2), (6, 6), (4, 4)\}.$$

sobre $X = \{1, 2, 3, 4, 5, 6\}$ es una relación de equivalencia. La clase de equivalencia [1] que contiene a 1 consiste en todas las x tales que $(x, 1) \in R$. Por lo tanto,

$$[1] = \{1, 3, 5\}.$$

Las clases de equivalencia restantes se encuentran de manera similar:

$$[3] = [5] = \{1, 3, 5\}, \quad [2] = [6] = \{2, 6\}, \quad [4] = \{4\}.$$ ◀

Ejemplo 3.2.11 ▶

Las clases de equivalencia aparecen con bastante claridad en la digráfica de una relación de equivalencia. Las tres clases de la relación R del ejemplo 3.2.10 aparecen en la digráfica de R (mostrada en la figura 3.2.2) como las tres subgráficas con vértices $\{1, 3, 5\}$, $\{2, 6\}$ y $\{4\}$. Una subgráfica G que representa una clase de equivalencia es la subgráfica más grande de la digráfica original que tiene la propiedad de que para cualesquiera vértices v y w en G, hay una arista dirigida de v a w. Por ejemplo, si $v, w \in \{1, 3, 5\}$, se tiene una arista dirigida de v a w. Más aún, no pueden agregarse vértices adicionales a 1, 3, 5, por lo que el conjunto de vértices resultante tiene una arista dirigida entre cada par de vértices. ◀

Ejemplo 3.2.12 ▶

Existen dos clases de equivalencia para la relación de equivalencia

$$R = \{(1, 1), (1, 3), (1, 5), (2, 2), (2, 4), (3, 1), (3, 3), (3, 5), (4, 2), (4, 4), (5, 1), (5, 3), (5, 5)\}$$

en $\{1, 2, 3, 4, 5\}$ del ejemplo 3.2.5, a saber,

$$[1] = [3] = [5] = \{1, 3, 5\}, \quad [2] = [4] = \{2, 4\}.$$ ◀

Ejemplo 3.2.13 ▶

Es sencillo verificar que la relación

$$R = \{(a, a), (b, b), (c, c)\}$$

sobre $X = \{a, b, c\}$ es reflexiva, simétrica y transitiva. Así, R es una relación de equivalencia. Las clases de equivalencia son

$$[a] = \{a\}, \quad [b] = \{b\}, \quad [c] = \{c\}$$ ◀

Ejemplo 3.2.14 ▶

Sea $X = \{1, 2, \ldots, 10\}$. Definimos $x\ R\ y$ para indicar que 3 divide a $x - y$. Es sencillo verificar que la relación R es reflexiva, simétrica y transitiva. Así, R es una relación de equivalencia sobre X.

Se determinarán los miembros de las clases de equivalencia. La clase de equivalencia [1] consiste en todas las x con $x\ R\ 1$. Entonces

$$[1] = \{x \in X \mid \text{divide } x - 1\} = \{1, 4, 7, 10\}.$$

De manera similar,

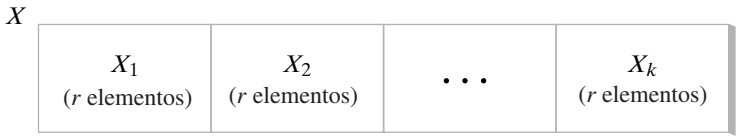
$$[2] = \{2, 5, 8\}, \quad [3] = \{3, 6, 9\}.$$

Estos tres conjuntos son una partición de X. Observe que

$$[1] = [4] = [7] = [10], \quad [2] = [5] = [8], \quad [3] = [6] = [9].$$

Para esta relación, *equivalencia* es "tiene el mismo residuo al dividir entre 3". ◀

Esta sección se cierra con la prueba de un resultado especial que se necesitará más adelante (vea las secciones 6.2 y 6.6). La prueba se ilustra en la figura 3.2.3.



$$|X| = rk$$

Figura 3.2.3 Prueba del Teorema 3.2.15.

Teorema 3.2.15

Sea R una relación de equivalencia en un conjunto finito X. Si cada clase de equivalencia tiene r elementos, existen $|X|/r$ clases de equivalencia.

Demostración Sean $X_1, X_2, \ldots, X_k$ las distintas clases de equivalencia. Como estos conjuntos hacen una partición de X,

$$|X| = |X_1| + |X_2| + \cdots + |X_k| = r + r + \cdots + r = kr$$

y se deriva la conclusión.

Sugerencias para resolver problemas

Una *relación de equivalencia* es una relación reflexiva, simétrica y transitiva. Para probar que una relación es de equivalencia, es necesario verificar que estas tres propiedades se cumplen (vea *sugerencias para resolver problemas* de la sección 3.1).

Una relación de equivalencia en un conjunto X crea una partición de X en subconjuntos ("Crear una partición" significa que cada x en X pertenece a exactamente uno de los subconjuntos de la partición.) Los subconjuntos que forman la partición se determinan de la siguiente manera. Elija $x_1 \in X$. Encuentre el conjunto, denotado por $[x_1]$, de todos los elementos relacionados con x_1. Elija otro elemento $x_2 \in X$ que no esté relacionado con x_1. Encuentre el conjunto $[x_2]$ de todos los elementos relacionados con x_2. Continúe de esta forma

hasta que todos los elementos de X estén asignados a un conjunto. Los conjuntos $[x_i]$ se llaman *clases de equivalencia*. La partición es

$$[x_1], [x_2], \ldots .$$

Los elementos de $[x_i]$ son *equivalentes* en el sentido de que todos están relacionados. Por ejemplo, la relación R, definida por $x\ R\ y$ si x y y son del mismo color, particiona el conjunto en subconjuntos donde cada subconjunto contiene los elementos que son todos del mismo color. Dentro de un subconjunto, los elementos son equivalentes en el sentido de que todos son de mismo color.

En la digráfica de una relación equivalente, una clase de equivalencia es la subgráfica más grande de la digráfica original que tiene la propiedad de que para cualesquiera vértices v y w en G, existe una arista dirigida de v a w.

Una partición de un conjunto da lugar a una relación de equivalencia. Si $X_1, \ldots, X_n$ es una partición del conjunto X y se define $x\ R\ y$ si para alguna i, x y y pertenecen ambos a X_i, entonces R es una relación de equivalencia sobre X. Las clases de equivalencia resultan ser $X_1, \ldots, X_n$. Así, "relación de equivalencia" y "partición de un conjunto" son diferentes puntos de vista de la misma situación. Una relación de equivalencia sobre X da lugar a una partición de X (a saber, las clases de equivalencia), y una partición de X da lugar a una relación de equivalencia (a saber, x está relacionada con y si x y y están en el mismo subconjunto de la partición). Este último hecho resulta útil para resolver ciertos problemas. Si le piden que encuentre una relación de equivalencia puede, ya sea encontrar directamente la relación de equivalencia, o bien construir una partición y después usar la relación de equivalencia asociada. De manera similar, si le piden que encuentre una partición, puede encontrar directamente la partición o construir una relación de equivalencia y después tomar las clases de equivalencia como la partición.

Sección de ejercicios de repaso

1. Defina *relación de equivalencia*. Dé un ejemplo de una relación de equivalencia. Dé un ejemplo de una relación que *no* sea una relación de equivalencia.
2. Defina *clase de equivalencia*. ¿Cómo se denota una clase de equivalencia? Dé un ejemplo de una clase de equivalencia para su relación de equivalencia del ejercicio 1.
3. Explique la relación entre partición de un conjunto y una relación de equivalencia.

Ejercicios

En los ejercicios 1 al 8, determine si la relación indicada es una relación de equivalencia en $\{1, 2, 3, 4, 5\}$. *Si la relación es una relación de equivalencia, liste las clases de equivalencia. (En los ejercicios 5 al 8,* $x, y \in \{1, 2, 3, 4, 5\}$.)

1. $\{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (1, 3), (3, 1)\}$
2. $\{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (1, 3), (3, 1), (3, 4), (4, 3)\}$
3. $\{(1, 1), (2, 2), (3, 3), (4, 4)\}$
4. $\{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (1, 5), (5, 1), (3, 5), (5, 3), (1, 3), (3, 1)\}$
5. $\{(x, y) \mid 1 \leq x \leq 5 \text{ y } 1 \leq y \leq 5\}$
6. $\{(x, y) \mid 4 \text{ divide a } x - y\}$
7. $\{(x, y) \mid 3 \text{ divide a } x + y\}$
8. $\{(x, y) \mid x \text{ divide a } 2 - y\}$

En los ejercicios 9 al 14, determine si la relación indicada es una relación de equivalencia en el conjunto de todas las personas.

9. $\{(x, y) \mid x \text{ y } y \text{ son de la misma altura}\}$
10. $\{(x, y) \mid x \text{ y } y \text{ en algún momento han vivido en el mismo país}\}$
11. $\{(x, y) \mid x \text{ y } y \text{ tienen el mismo nombre }\}$
12. $\{(x, y) \mid x \text{ es más alto que } y\}$
13. $\{(x, y) \mid x \text{ y } y \text{ tienen los mismos padres}\}$
14. $\{(x, y) \mid x \text{ y } y \text{ tienen el mismo color de pelo}\}$

En los ejercicios 15 al 20, liste los miembros de la relación de equivalencia en $\{1, 2, 3, 4\}$ *definida (como en el teorema 3.2.1) por la partición dada. Además, encuentre las clases de equivalencia* [1], [2], [3] y [4].

15. $\{\{1, 2\}, \{3, 4\}\}$
16. $\{\{1\}, \{2\}, \{3, 4\}\}$
17. $\{\{1\}, \{2\}, \{3\}, \{4\}\}$
18. $\{\{1, 2, 3\}, \{4\}\}$
19. $\{\{1, 2, 3, 4\}\}$
20. $\{\{1\}, \{2, 4\}, \{3\}\}$

En los ejercicios 21 al 23, sea $X = \{1, 2, 3, 4, 5\}$, $Y = \{3, 4\}$ y $C = \{1, 3\}$. *Defina la relación R en* $\mathcal{P}(X)$, *el conjunto de todos los subconjuntos de X, como*

$$A\ R\ B \text{ si } A \cup Y = B \cup Y.$$

21. Demuestre que R es una relación de equivalencia.
22. Liste los elementos de $[C]$, la clase de equivalencia que contiene a C.
23. ¿Cuántas clases de equivalencia diferentes hay?
24. Sea

$$X = \{\text{San Francisco, Pittsburg, Chicago, San Diego, Filadelfia, Los Ángeles}\}.$$

Defina una relación R sobre X como $x\ R\ y$ si x y y están en el mismo estado.

a) Demuestre que R es una relación de equivalencia.

b) Liste las clases de equivalencia de X.

25. Demuestre que si R es una relación de equivalencia sobre X, entonces

$$\text{dominio } R = \text{rango } R = X.$$

26. Si una relación de equivalencia tiene sólo una clase de equivalencia, ¿cómo debe verse la relación?

27. Si R es una relación de equivalencia en un conjunto X y $|X| = |R|$, ¿Cómo debe verse la relación?

28. Listando los pares ordenados, dé un ejemplo de una relación de equivalencia en $\{1, 2, 3, 4, 5, 6\}$ que tiene exactamente cuatro clases de equivalencia.

29. ¿Cuántas relaciones de equivalencia hay en el conjunto $\{1, 2, 3\}$?

30. Sea $X = \{1,2, \ldots, 10\}$. Defina una relación R sobre $X \times X$ como $(a, b)\ R\ (c, d)$ si $a + d = b + c$.

a) Demuestre que R es una relación de equivalencia sobre $X \times X$.

b) Liste un miembro de cada clase de equivalencia en $X \times X$.

31. Sea $X = \{1,2, \ldots, 10\}$. Defina una relación R sobre $X \times X$ como $(a, b)\ R\ (c, d)$ si $ad = bc$.

a) Demuestre que R es una relación de equivalencia sobre $X \times X$.

b) Liste un miembro de cada clase de equivalencia en $X \times X$.

c) Describa la relación R en términos familiares.

32. Sea R una relación reflexiva y transitiva sobre X. Demuestre que $R \cap R^{-1}$ es una relación de equivalencia sobre X.

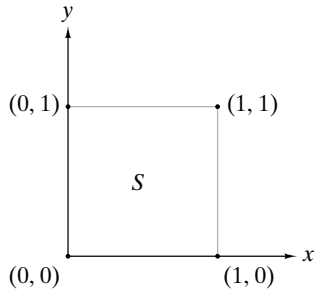
33. Sean R_1 y R_2 relaciones de equivalencia sobre X.

a) Demuestre que $R_1 \cap R_2$ es una relación de equivalencia sobre X.

b) Describa las clases de equivalencia de $R_1 \cap R_2$ en términos de las clases de equivalencia de R_1 y las clases de equivalencia de R_2.

34. Suponga que $\mathcal{S}$ es una colección de subconjuntos de un conjunto X y $X = \cup\, \mathcal{S}$. (No se supone que la familia $\mathcal{S}$ sea disjunta por pares). Defina $x\ R\ y$ de manera que para algún conjunto $S \in \mathcal{S}$, ambas x y y están en S. ¿Es R necesariamente reflexiva, simétrica y transitiva?

35. Sea S un cuadrado unitario que incluye el interior, como se muestra en la figura que sigue.



Defina la relación R en S por $(x, y)\ R\ (x', y')$ si $(x = x'$ y $y = y')$ o $(y = y'$ y $x = 0$ y $x' = 1)$ o $(y = y'$ y $x = 1$ y $x' = 0)$.

a) Demuestre que R es una relación de equivalencia en S.

b) Si los puntos en la misma clase de equivalencia se engomaran, ¿cómo describiría la figura que se forma?

36. Sea S un cuadrado unitario que incluye el interior (como en el ejercicio 35). Defina una relación R' en S por $(x, y)\ R'\ (x', y')$ si $(x = x'$ y $y = y')$ o $(y = y'$ y $x = 0$ y $x' = 1)$ o $(y = y'$ y $x = 1$ y $x' = 0)$ o $(x = x'$ y $y = 0$ y $y' = 1)$ o $(x = x'$ y $y = 1$ y $y' = 0)$. Sea

$$R = R' \cup \{((0, 0), (1, 1)), ((0, 1), (1, 0)), ((1, 0), (0, 1)), ((1, 1), (0, 0))\}.$$

a) Demuestre que R es una relación de equivalencia en S.

b) Si los puntos en la misma clase de equivalencia se engomaran, ¿como describiría la figura que se forma?

37. Sea f una función de X a Y. Defina una relación R sobre X por

$$x\ R\ y \quad \text{si } f(x) = f(y).$$

Demuestre que R es una relación de equivalencia sobre X.

38. Sea f una función característica en X. (La "función característica" se definió en el ejercicio 62, sección 2.2). Defina la relación R sobre X por $x\ R\ y$ si $f(x) = f(y)$. De acuerdo con el ejercicio anterior, R es una relación de equivalencia. ¿Cuáles son las clases de equivalencia?

39. Sea f una función de X a Y. Sea

$$\mathcal{S} = \{f^{-1}(\{y\}) \mid y \in Y\}.$$

[La definición de $f^{-1}(B)$, donde B es un conjunto, precede al ejercicio 57, sección 2.2]. Demuestre que $\mathcal{S}$ es una partición de X. Describa una relación de equivalencia que dé lugar a esta partición.

40. Sea R una relación de equivalencia en un conjunto A. Defina una función f de A al conjunto de clases de equivalencia de A mediante la regla

$$f(x) = [x].$$

¿Cuándo se tiene $f(x) = f(y)$?

41. Sea R una relación de equivalencia en un conjunto A. Suponga que g es una función de A a un conjunto X que tiene la propiedad de que si $x\ R\ y$, entonces $g(x) = g(y)$. Demuestre que

$$h([x]) = g(x)$$

define una función del conjunto de clases de equivalencia de A a X. [Lo que debe probarse es que h asigna de manera *única* un valor a $[x]$; es decir, si $[x] = [y]$, entonces $g(x) = g(y)$].

42. Sea X el conjunto de sucesiones con dominio finito. Defina una relación R sobre X como $s\ R\ t$ si $|\text{dominio } s| = |\text{dominio } t|$ y, si el dominio de s es $\{m, m + 1, \ldots, m + k\}$ y el dominio de t es $\{n, n + 1, \ldots, n + k\}$, $s_{m+i} = t_{n+i}$ para $i = 0, \ldots, k$.

a) Demuestre que R es una relación de equivalencia.

b) Explique en palabras qué significa que dos sucesiones en X sean equivalentes bajo la relación R.

c) Como una sucesión es una función, una sucesión es un conjunto de pares ordenados. Dos sucesiones son iguales si los dos conjuntos de pares ordenados son iguales. Compare las diferencias entre las dos sucesiones equivalentes en X y las dos sucesiones iguales en X.

Sea R una relación en un conjunto X. Defina

$$\rho(R) = R \cup \{(x, x) \mid x \in X\}$$
$$\sigma(R) = R \cup R^{-1}$$
$$R^n = R \circ R \circ R \circ \cdots \circ R \quad (n R\text{'s})$$
$$\tau(R) = \cup\{R^n \mid n = 1, 2, \ldots\}.$$

La relación $\tau(R)$ se llama cerradura transitiva *de R.*

43. Para las relaciones R_1 y R_2 del ejercicio 38, sección 3.1, encuentre $\rho(R_i)$, $\sigma(R_i)$, $\tau(R_i)$, y $\tau(\sigma(\rho(R_i)))$ para $i = 1, 2$.

44. Demuestre que $\rho(R)$ es reflexiva.

45. Demuestre que $\sigma(R)$ es simétrica.

46. Demuestre que $\tau(R)$ es transitiva.

★47. Demuestre que $\tau(\sigma(\rho(R)))$ es una relación de equivalencia que contiene a R.

★ **48.** Demuestre que $\tau(\sigma(\rho(R)))$ es la relación de equivalencia más pequeña sobre X que contiene a R; es decir, demuestre que si R' es una relación de equivalencia sobre X y $R' \supseteq R$, entonces $R' \supseteq \tau(\sigma(\rho(R)))$.

★ **49.** Demuestre que R es transitiva si y sólo si $\tau(R) = R$.

En los ejercicios 50 al 56, si la afirmación es verdadera para todas las relaciones R_1 y R_2 en un conjunto arbitrario X, demuéstrelo; de otra manera dé un contraejemplo.

50. $\rho(R_1 \cup R_2) = \rho(R_1) \cup \rho(R_2)$

51. $\sigma(R_1 \cap R_2) = \sigma(R_1) \cap \sigma(R_2)$

52. $\tau(R_1 \cup R_2) = \tau(R_1) \cup \tau(R_2)$

53. $\tau(R_1 \cap R_2) = \tau(R_1) \cap \tau(R_2)$

54. $\sigma(\tau(R_1)) = \tau(\sigma(R_1))$

55. $\sigma(\rho(R_1)) = \rho(\sigma(R_1))$

56. $\rho(\tau(R_1)) = \tau(\rho(R_1))$

Si X y Y son conjuntos, se define X como equivalente a Y si existe una función uno a uno, sobre de X a Y.

57. Demuestre que la equivalencia de conjuntos es una relación de equivalencia.

58. Si X y Y son conjuntos finitos y X es equivalente a Y, ¿qué indica acerca de X y Y?

59. Demuestre que los conjuntos $\{1,2, \ldots\}$ y $\{2,4, \ldots\}$ son equivalentes.

★***60.*** Demuestre que para cualquier conjunto X, X no es equivalente a $\mathcal{P}(X)$, el conjunto potencia de X.

Rincón de solución de problemas

Relaciones de equivalencia

Problema

Responda a las siguientes preguntas para la relación R definida sobre el conjunto de cadenas de 8 bits por $s_1 \, R \, s_2$, siempre que los primeros 4 bits de s_1 y s_2 coincidan.

a) Demuestre que R es una relación de equivalencia.

b) Liste un miembro de cada clase de equivalencia.

c) ¿Cuántas clases de equivalencia hay?

Cómo atacar el problema

Se comienza por observar algunas cadenas de 8 bits específicas que estén relacionadas según la relación R. Tomemos una cadena arbitraria 01111010 y encontremos cadenas relacionadas con ella. Una cadena s está relacionada con 01111010 si los primeros 4 bits de 01111010 y s coinciden. Esto significa que s debe comenzar con 0111 y los últimos 4 bits pueden ser cualesquiera. Un ejemplo es $s = 01111000$.

Si se listan todas las cadenas relacionadas con 01111010, al hacerlo debe tenerse cuidado de que 0111 vaya seguido de todas las cadenas posibles de 4 bits.

01110000,	01110001,	01110010,	01110011,
01110100,	01110101,	01110110,	01110111,
01111000,	01111001,	01111010,	01111011,
01111100,	01111101,	01111110,	01111111.

Suponiendo por el momento que R es una relación de equivalencia, la clase de equivalencia que contiene a 01111010, denotada por [01111010], consiste en todas las cadenas relacionadas con 01111010. Por lo tanto, lo que se acaba de calcular son los miembros de [01111010].

Observe que si se toma cualquier cadena en [01111010], por ejemplo, 01111100, y se calcula su clase de equivalencia [01111100], se obtendrá justo el mismo conjunto de cadenas, a saber, el conjunto de cadenas de 8 bits que comienzan con 0111.

Para obtener un ejemplo deferente, tendríamos que comenzar con una cadena cuyos 4 primeros bits fueran diferentes de 0111, digamos 1011. Como un ejemplo, las cadenas relacionadas con 10110100 son

10110000,	10110001,	10110010,	10110011,
10110100,	10110101,	10110110,	10110111,
10111000,	10111001,	10111010,	10111011,
10111100,	10111101,	10111110,	10111111.

Lo que se acaba de calcular son los miembros de [10110100]. Se ve que [01111010] y [10110100] no tienen miembros en común. Siempre es el caso cuando dos clases equivalentes son idénticas o no tienen miembros en común (vea el teorema 3.2.8).

Antes de leer más, calcule los miembros de algunas otras clases de equivalencia.

Cómo encontrar una solución

Para demostrar que R es una relación de equivalencia, debemos demostrar que R es reflexiva, simétrica y transitiva (vea la definición 3.2.3). Para cada propiedad, se tomará directamente la definición y se verificará que las condiciones especificadas en ella se cumplan.

Para que R sea reflexiva, debe tenerse $s \, R \, s$ para toda cadena s de ocho bits. Para que $s \, R \, s$ sea verdadera, los primeros 4 bits de s y s deben coincidir. No hay duda de que esto se cumple.

Para que R sea simétrica, para todas las cadenas de 8 bits s_1 y s_2, si $s_1 \, R \, s_2$, entonces $s_2 \, R \, s_1$. Usando la definición de R, esta condición se traduce en: si los primeros 4 bits de s_1 y s_2 coinciden, entonces los primeros 4 bits de s_2 y s_1 coinciden. Sin duda, éste es el caso.

Para que R sea transitiva, para todas las cadenas de 8 bits s_1, s_2 y s_3, si $s_1 \, R \, s_2$ y $s_2 \, R \, s_3$, entonces $s_1 \, R \, s_3$. De nuevo usando la definición de R, esta condición se traduce en: si los primeros 4 bits de s_1 y s_2 coinciden y los primeros 4 bits de s_2 y s_3 coinciden, entonces los primeros 4 bits de s_1 y s_3 coinciden. Esto también es cierto. Se ha probado que R es una relación de equivalencia.

En el análisis anterior, se encontró que cada cadena de 4 bits distinta determina una clase de equivalencia. Por ejemplo, la cadena 0111 determina la clase de equivalencia que

consiste en todas las cadenas de 8 bits que comienzan con 0111. Por lo tanto, el número de clases de equivalencia es igual al número de cadenas de 4 bits. Se puede sencillamente listarlas

0000, 0001, 0010, 0011,
0100, 0101, 0110, 0111,
1000, 1001, 1010, 1011,
1100, 1101, 1110, 1111

y después contarlas. Existen 16 clases de equivalencia.

Considere el problema de listar un miembro de cada clase de equivalencia. Las 16 cadenas de 4 bits en la lista anterior determinan 16 clases de equivalencia. La primera cadena 0000 define la clase de equivalencia que consiste en todas las cadenas de 8 bits que comienzan con 0000; la segunda cadena 0001 determina la clase de equivalencia que consiste en todas las cadenas de 8 bits que comienzan con 0001; y así sucesivamente. Entonces para dar una lista con un miembro de cada clase de equivalencia, sólo se necesita adjuntar alguna cadena de 4 bits a cada una de las cadenas en la lista anterior:

00000000, 00010000, 00100000, 00110000,
01000000, 01010000, 01100000, 01110000,
10000000, 10010000, 10100000, 10110000,
11000000, 11010000, 11100000, 11110000.

Solución formal

a) Se presentó una demostración formal de que R es una relación de equivalencia.

b)

00000000, 00010000, 00100000, 00110000,
01000000, 01010000, 01100000, 01110000,
10000000, 10010000, 10100000, 10110000,
11000000, 11010000, 11100000, 11110000

es una lista con un miembro de cada clase de equivalencia.

c) Existen 16 clases de equivalencia.

Resumen de las técnicas de solución de problemas

- Liste los elementos que están relacionados.
- Calcule algunas clases de equivalencia; es decir, liste *todos* los elementos relacionados con un elemento específico.
- Resulta útil resolver los incisos de un problema en un orden diferente que el indicado en el enunciado del problema. En nuestro ejemplo, fue útil ver algunos casos concretos para *suponer* que la relación era una relación de equivalencia antes de probar que de hecho lo era.
- Para demostrar que una relación específica R es una relación de equivalencia, vaya a las definiciones. Pruebe que R es reflexiva, simétrica y transitiva verificando directamente que R satisface la definición de reflexiva, simétrica y transitiva.
- Si el problema es contar el número de elementos que satisfacen alguna propiedad (en el problema se pide contar el número de clases de equivalencia) y el número es suficientemente pequeño, basta dar una lista de los elementos y contarlos.

Comentarios

En lenguajes de programación, sólo suelen ser significativos un número específico de caracteres de los nombres de las variables y los términos especiales (técnicamente, éstos se llaman *identificadores*). Por ejemplo, en el lenguaje de programación C, sólo los primeros 31 caracteres de los identificadores son significativos. Esto quiere decir que si dos identificadores comienzan con los mismos 31 caracteres, el sistema puede considerarlos idénticos.

Si se define una relación R en el conjunto de identificadores C como $s_1 \, R \, s_2$, siempre que los primeros 31 caracteres de s_1 y s_2 coincidan, entonces R es una relación de equivalencia. Una clase de equivalencia consiste en los identificadores que el sistema puede considerar idénticos.

3.3 ➜ Matrices de relaciones

WWW

Una matriz es una manera conveniente de representar una relación R de X a Y. Esta representación se puede usar en una computadora para analizar una relación. Se etiquetan los renglones con elementos de X (en algún orden arbitrario), y se etiquetan las columnas con elementos de Y (de nuevo, en algún orden arbitrario). Después, el elemento en el renglón x y la columna y se hace igual a 1 si $x \, R \, y$, y a 0 de otra manera. Esta matriz se llama **matriz de la relación *R*** (relativa al orden de X y Y).

Ejemplo 3.3.1 ▶ La matriz de la relación

$$R = \{(1, b), (1, d), (2, c), (3, c), (3, b), (4, a)\}$$

de $X = \{1, 2, 3, 4\}$ a $Y = \{a, b, c, d\}$ respecto a los órdenes 1, 2, 3, 4 y a, b, c, d es

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

◀

Ejemplo 3.3.2 ▶

La matriz de la relación R del ejemplo 3.3.1 relativa a los órdenes 2, 3, 4, 1 y d, b, a, c es

$$\begin{array}{c} \\ 2 \\ 3 \\ 4 \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} d & b & a & c \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \end{array}.$$

Es evidente que la matriz de una relación de X a Y es dependiente de los órdenes de X y Y. ◀

Ejemplo 3.3.3 ▶

La matriz de la relación R de $\{2, 3, 4\}$ a $\{5, 6, 7, 8\}$, relativa al orden 2, 3, 4 y 5, 6, 7, 8, definida por

$$x \, R \, y \qquad \text{si } x \text{ divide a } y$$

es

$$\begin{array}{c} \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 5 & 6 & 7 & 8 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}.$$

◀

Cuando se escribe la matriz de una relación R en un conjunto X (por ejemplo, de X a X), se usa el mismo orden para los renglones que se usa para las columnas.

Ejemplo 3.3.4 ▶

La matriz de la relación

$$R = \{(a, a), (b, b), (c, c), (d, d), (b, c), (c, b)\}$$

en $\{a, b, c, d\}$, relativa al orden a, b, c, d es

$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}.$$

◀

Observe que la matriz de una relación en un conjunto X es una matriz cuadrada.

Se puede determinar con facilidad si una relación R en una conjunto X es reflexiva al examinar la matriz A de R (relativa al orden). La relación R es reflexiva si y sólo si A tiene 1 en la diagonal principal. (La diagonal principal de una matriz cuadrada consiste en los elementos sobre la línea que va desde arriba a la izquierda hasta abajo a la derecha). La relación R es reflexiva si y sólo si $(x, x) \in R$ para toda $x \in X$. Pero esta última condición se cumple precisamente si hay unos en la diagonal principal. Advierta que la relación R del ejemplo 3.3.4 es reflexiva y que tiene unos en la diagonal principal de la matriz de R.

También es sencillo determinar si una relación R en un conjunto X es simétrica examinando la matriz A de R (relativa al orden). La relación R es simétrica si y sólo si para toda i y j, el elemento *ijésimo* de A es igual al elemento *jiésimo* de A. (De manera menos formal, R es simétrica si y sólo si A es simétrica respecto a la diagonal principal). La razón

es que R es simétrica si y sólo si siempre que (x, y) está en R, (y, x) también está en R. Pero esta última condición se cumple precisamente cuando A es simétrica respecto a la diagonal principal. Note que la relación R del ejemplo 3.3.4 es simétrica y que la matriz de R es simétrica respecto a la diagonal principal.

También se puede determinar rápidamente si una relación R es antisimétrica examinando la matriz de R (relativa al orden) (vea el ejercicio 11).

Se concluye mostrando cómo se relaciona la multiplicación de matrices con la composición de relaciones y cómo se puede usar la matriz de una relación para probar la transitividad.

Ejemplo 3.3.5 ▶

Sea R_1 la relación de $X = \{1, 2, 3\}$ a $Y = \{a, b\}$ definida por

$$R_1 = \{(1, a), (2, b), (3, a), (3, b)\},$$

y sea R_2 la relación de Y a $Z = \{x, y, z\}$ definida por

$$R_2 = \{(a, x), (a, y), (b, y), (b, z)\}.$$

La matriz de R_1 relativa al orden 1, 2, 3 y a, b es

$$A_1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cc} a & b \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \end{array},$$

y la matriz de R_2 relativa al orden a, b y x, y, z es

$$A_2 = \begin{array}{c} \\ a \\ b \end{array} \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \end{array}.$$

El producto de estas matrices es

$$A_1 A_2 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 2 & 1 \end{pmatrix}.$$

Se interpretará este producto.

El ikésimo elemento de A_1A_2 se calcula como

$$\begin{array}{c} \\ i \end{array} \begin{array}{c} \begin{array}{cc} a & b \end{array} \\ \begin{pmatrix} s & t \end{pmatrix} \end{array} \quad \begin{array}{c} k \\ \begin{pmatrix} u \\ v \end{pmatrix} \end{array} = su + tv.$$

Si este valor es diferente de cero, entonces su o tv son diferentes de cero. Suponga que $su \neq 0$. (El argumento es similar si $tv \neq 0$). Entonces $s \neq 0$ y $u \neq 0$. Esto significa que $(i, a) \in R_1$ y $(a, k) \in R_2$. Esto implica que $(i, k) \in R_2 \circ R_1$. Se ha demostrado que el elemento *ikésimo* en A_1A_2 es diferente de cero, entonces $(i, k) \in R_2 \circ R_1$. La recíproca también es cierta, como se verá.

Suponga que $(i, k) \in R_2 \circ R_1$. Entonces, o bien

1. $(i, a) \in R_1$ y $(a, k) \in R_2$

o

2. $(i, b) \in R_1$ y $(b, k) \in R_2$.

Si 1 se cumple, entonces $s = 1$ y $u = 1$, y así $su = 1$ y $su + tv$ es diferente de cero. De manera similar, si 2 se cumple, $tv = 1$ y de nuevo se tiene $su + tv$ diferente de cero. Esto demuestra que si $(i, k) \in R_2 \circ R_1$, entonces el elemento *ikésimo* en A_1A_2 es diferente de cero.

Se demostró que $(i, k) \in R_2 \circ R_1$ si y sólo si el elemento *ikésimo* de A_1A_2 es diferente de cero; entonces A_1A_2 es "casi" la matriz de la relación $R_2 \circ R_1$. Para obtener la matriz de la relación $R_2 \circ R_1$, sólo se necesita cambiar todos los elementos de A_1A_2 que son diferentes de cero a uno. Así, la matriz de la relación $R_2 \circ R_1$, relativa al orden elegido antes

1, 2, 3 y x, y, z es

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \end{array}.$$

◀

El argumento dado en el ejemplo 3.3.5 se cumple para cualesquiera relaciones. Este resultado se resume en el teorema 3.3.6.

Teorema 3.3.6

Sea R_1 una relación de X a Y y sea R_2 una relación de Y a Z. Seleccione el orden de X, Y y Z. Sea A_1 la matriz de la relación R_1 y sea A_2 la matriz de la relación R_2 respecto a los órdenes seleccionados. La matriz de la relación $R_2 \circ R_1$ respecto al orden seleccionado se obtiene sustituyendo por 1 *cada término diferente de cero en la matriz del producto A_1A_2.*

Demostración La prueba se describió antes del enunciado del teorema.

El teorema 3.3.6 aporta una prueba rápida para determinar si una relación es transitiva. Si A es la matriz de R (respecto a algún orden), se calcula A^2. Luego se comparan A y A^2. La relación R es transitiva si y sólo si siempre que el elemento i, j en A^2 es diferente de cero, el elemento i, j en A también es diferente de cero. La razón es que el elemento i, j en A^2 es diferente de cero si y sólo si hay elementos (i, k) y (k, j) en R (vea la demostración del teorema 3.3.6). Ahora R es transitiva si y sólo si siempre que (i, k) y (k, j) están en R, entonces (i, j) está en R. Pero (i, j) está en R si y sólo si el elemento i, j en A es diferente de cero. Por tanto, R es transitiva si y sólo si siempre que el elemento i, j en A^2 es diferente de cero, el elemento i, j en A también es diferente de cero.

Ejemplo 3.3.7 ▶

La matriz de la relación

$$R = \{(a, a), (b, b), (c, c), (d, d), (b, c), (c, b)\}$$

en $\{a, b, c, d\}$, respecto al orden a, b, c, d es

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Su cuadrado es

$$A^2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Se ve que siempre que el elemento i, j en A^2 es diferente de cero, el elemento i, j en A también es diferente de cero. Por lo tanto, R es transitiva. ◀

Ejemplo 3.3.8 ▶

La matriz de la relación

$$R = \{(a, a), (b, b), (c, c), (d, d), (a, c), (c, b)\}$$

en $\{a, b, c, d\}$, respecto al orden a, b, c, d es

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Su cuadrado es

$$A^2 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

El elemento del renglón 1, columna 2 de A^2 es diferente de cero, pero el elemento correspondiente en A es cero. Por lo tanto, R *no* es transitiva. ◀

Sugerencias para resolver problemas

La matriz de una relación R es otra manera de representar o especificar una relación de X a Y. El elemento en el renglón x y la columna y es 1 si $x\ R\ y$, o 0 si $x \not R\ y$.

Una relación es reflexiva si y sólo si la diagonal de su representación matricial contiene sólo unos.

Una relación es simétrica si y sólo si su matriz es simétrica (es decir, el elemento i, j siempre es igual al elemento j, i).

Sea R_1 una relación de X a Y y sea R_2 una relación de Y a Z. Sea A_1 la matriz de la relación R_1 y A_2 la matriz de la relación R_2. La matriz de la relación $R_2 \circ R_1$ se obtiene sustituyendo por 1 cada término diferente de cero en la matriz producto $A_1 A_2$.

Para probar si una relación es transitiva, sea A su matriz. Calcule A^2. La relación es transitiva si y sólo si siempre que el elemento i, j en A^2 es diferente de cero, el elemento i, j en A también es diferente de cero.

Sección de ejercicios de repaso

1. ¿Qué es la matriz de una relación?

2. A partir de la matriz de una relación, ¿cómo se puede determinar si la relación es reflexiva?

3. A partir de la matriz de una relación, ¿cómo se puede determinar si la relación es simétrica?

4. A partir de la matriz de una relación, ¿cómo se puede determinar si la relación es transitiva?

5. A partir de la matriz A_1 de la relación R_1 y la matriz A_2 de la relación R_2, explique cómo obtener la matriz de la relación $R_2 \circ R_1$.

Ejercicios

En los ejercicios 1 al 3, encuentre la matriz de la relación R de X a Y respecto al orden dado.

1. $R = \{(1, \delta), (2, \alpha), (2, \Sigma), (3, \beta), (3, \Sigma)\}$; orden de X: 1, 2, 3; orden de Y: α, β, Σ, δ

2. R como en el ejercicio 1; orden de X: 3, 2, 1; orden de Y: β,α, δ

3. $R = \{(x, a), (x, c), (y, a), (y, b), (z, d)\}$; orden de X: x, y, z; orden de Y: a, b, c, d

En los ejercicios 4 al 6, encuentre la matriz de la relación R sobre X relativa al orden dado.

4. $R = \{(1, 2), (2, 3), (3, 4), (4, 5)\}$; orden de X: 1, 2, 3, 4, 5

5. R como en el ejercicio 4; orden de X: 5, 3, 1, 2, 4

6. $R = \{(x, y) \mid x < y\}$; orden de X: 1, 2, 3, 4

7. Encuentre las matrices que representan las relaciones de los ejercicios 13 al 16 de la sección 3.1.

En los ejercicios 8 al 10, escriba la relación R dada por la matriz, como un conjunto de pares ordenados.

8.
$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array}\begin{array}{c} \begin{array}{cccc} w & x & y & z \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

9.
$$\begin{array}{c} \\ 1 \\ 2 \end{array}\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

10.
$$\begin{array}{c} \\ w \\ x \\ y \\ z \end{array}\begin{array}{c} \begin{array}{cccc} w & x & y & z \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

11. ¿Cómo se puede determinar con facilidad si una relación R es antisimétrica examinando la matriz de R (relativa a algún orden)?

12. Diga si la relación del ejercicio 10 es reflexiva, simétrica, transitiva, antisimétrica, de orden parcial y/o una relación de equivalencia.

13. A partir de la matriz de una relación R de X a Y, ¿cómo podemos encontrar la matriz de la relación inversa R^{-1}?

14. Encuentre la matriz de la inversa de cada una de las relaciones en los ejercicios 8 y 9.

15. Use la matriz de la relación para probar la transitividad (vea los ejemplos 3.3.7 y 3.3.8) para las relaciones de los ejercicios 4, 6 y 10.

En los ejercicios 16 al 18, encuentre

a) La matriz A_1 de la relación R_1 (relativa al orden dado)

b) La matriz A_2 de la relación R_2 (relativa al orden dado)

c) La matriz producto $A_1 A_2$.

d) Use los resultados del inciso c) para encontrar la relación $R_2 \circ R_1$

e) Use el resultado del inciso d) para encontrar la relación $R_2 \circ R_1$ (como un conjunto de pares ordenados).

16. $R_1 = \{(1, x), (1, y), (2, x), (3, x)\}$; $R_2 = \{(x, b), (y, b), (y, a), (y, c)\}$; bajo el orden: 1, 2, 3; x, y; a, b, c

17. $R_1 = \{(x, y) | x \text{ divide a } y\}$; R_1 es de X a Y; $R_2 = \{(y, z) | \ y > z\}$; R_2 es de Y a Z; orden de X y Y: 2, 3, 4, 5; orden de Z: 1, 2, 3, 4

18. $R_1 = \{(x, y) \mid x + y \leq 6\}$; R_1 es de X a Y; $R_2 = \{(y, z) \mid y = z + 1\}$; R_2 es de Y a Z; orden de X, Y y Z: 1, 2, 3, 4, 5

19. Dada la matriz de una relación de equivalencia R sobre X, ¿cómo se puede encontrar con facilidad la clase de equivalencia que contiene al elemento $x \in X$?

★ **20.** Sea R_1 una relación de X a Y y sea R_2 una relación de Y a Z. Elija el orden de X, Y y Z. Todas las matrices de relaciones son relativas a este orden. Sea A_1 la matriz de R_1 y sea A_2 la matriz de R_2. Demuestre que el elemento *ikésimo* de la matriz producto A_1A_2 es igual al número de elementos en el conjunto

$$\{m \mid (i, m) \in R_1 \quad \text{y} \quad (m, k) \in R_2\}.$$

21. Suponga que R_1 y R_2 son relaciones en un conjunto X, A_1 es la matriz de R_1 relativa a algún orden de X, y A_2 es la matriz de R_2 relativa a algún orden de X. Sea A la matriz cuyo elemento *ijésimo* es 1 si el elemento *ijésimo* de A_1 o de A_2 es 1. Pruebe que A es la matriz de $R_1 \cup R_2$.

22. Suponga que R_1 y R_2 son relaciones en un conjunto X, A_1 es la matriz de R_1 relativa a algún orden de X, y A_2 es la matriz de R_2 relativa a algún orden de X. Sea A la matriz cuyo elemento *ijésimo* es 1 si los elementos *ijésimo* de ambas A_1 y A_2 son 1. Pruebe que A es la matriz de $R_1 \cap R_2$.

23. Suponga que la matriz de la relación R_1 en $\{1, 2, 3\}$ es

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

relativa al orden 1, 2, 3, y que la matriz de la relación R_2 en $\{1, 2, 3\}$ es

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

relativa al orden 1, 2, 3. Con base en el ejercicio 21, encuentre la matriz de la relación $R_1 \cup R_2$ relativa al orden 1, 2, 3.

24. Con base en el ejercicio 22, encuentre la matriz de la relación $R_1 \cap R_2$ relativa al orden 1, 2, 3 para las relaciones del ejercicio 23.

25. ¿Cómo se puede determinar con facilidad si una relación R es una función examinando la matriz de R (relativa a algún orden)?

26. Sea A la matriz de una función f de X a Y (relativa a algún orden de X y Y). ¿Qué condiciones deben satisfacer A para que f sea sobre Y?

27. Sea A la matriz de una función f de X a Y (relativa a algún orden de X y Y). ¿Qué condiciones debe satisfacer A para que f sea uno a uno?

3.4 ➜ Bases de datos relacionales†

El prefijo "bi" en una relación binaria R se refiere al hecho de que R tiene dos columnas cuando R se presenta como una tabla. Con frecuencia es útil permitir que una tabla tenga un número arbitrario de columnas. Si la tabla tiene n columnas, la relación correspondiente sería una **relación *n*-aria**.

Ejemplo 3.4.1 ▶

La tabla 3.4.1 representa una relación 4-aria. Esta tabla expresa la relación entre número de identificación, nombre, posición y edad.

TABLA 3.4.1 ■ JUGADOR

Núm. identif.	*Nombre*	*Posición*	*Edad*
22012	Johnsonbaugh	c	22
93831	Glover	j	24
58199	Battey	p	18
84341	Cage	c	30
01180	Homer	1b	37
26710	Score	p	22
61049	Johnsonbaugh	j	30
39826	Singleton	2b	31

◀

También es posible expresar una relación n-aria como una colección de n-eadas.

† Esta sección se puede omitir sin pérdida de continuidad.

Ejemplo 3.4.2 ▶

La tabla 3.4.1 se puede expresar como el conjunto

{(22012, Johnsonbaugh, c, 22), (93831, Glover, j 24),
(58199, Battey, p, 18), (84341, Cage, c, 30),
(01180, Homer, 1b, 37), (26710, Score, p, 22),
(61049, Johnsonbaugh, j 30), (39826, Singleton, 2b, 31)}

de cuartetos. ◀

Una **base de datos** es una colección de registros que maneja una computadora. Por ejemplo, una base de datos de un línea aérea contiene registros de reservaciones de pasajeros, programación de vuelos, equipo, etcétera. Los sistemas de cómputo son capaces de almacenar grandes cantidades de información en bases de datos. Los datos están disponibles para diferentes aplicaciones. Los **sistemas de administración de bases de datos** son programas que ayudan a los usuarios a tener acceso a la información de las bases de datos. El modelo de **base de datos relacional**, inventado por E.F. Codd, se basa en el concepto de una relación *n*-aria. Se dará una breve introducción de algunas ideas fundamentales en la teoría de bases de datos relacionales. Para ver más detalles de las bases de datos relacionales se recomienda al lector consultar [Codd; Date; y Kroenke]. Comenzaremos por establecer parte de la terminología.

Las columnas de una relación *n*–aria se llaman **atributos (o campos)**. El dominio de un atributo es un conjunto al que pertenecen todos los elementos de ese atributo. Por ejemplo, en la tabla 3.4.1, el atributo Edad puede tomarse como el conjunto de todos los enteros positivos menores que 100. El atributo Nombre puede tomarse como todas las cadenas en el alfabeto con longitud 30 o menor.

Un solo atributo o combinación de atributos para una relación es una **llave** (en el área de computación es más común llamarle llave, aunque en algunos ambientes se le conoce como clave) si los valores de los atributos definen de manera única una *n*-eada. Por ejemplo, en la tabla 3.4.1, se puede tomar el atributo "número de identificación" como una llave. (Se supone que cada persona tiene un número de identificación único). El atributo Nombre no es una llave porque es posible que diferentes personas tengan el mismo nombre. Por la misma razón, no es conveniente tomar los atributos "posición" o "edad" como una llave. Una combinación del "nombre" y "posición" podrían usarse como llave para la tabla 3.4.1, ya que en nuestro ejemplo un jugador se define de manera única por un nombre y una posición.

Un sistema de administración de bases de datos responde a **consultas** (también llamadas *queries*). Una consulta es una petición de información de la base de datos. Por ejemplo, "encuentre todas las personas que juegan como jardinero" es una consulta significativa para la relación dada en la tabla 3.4.1. Se analizarán varias operaciones sobre las relaciones que se utilizan para responder a las consultas en el modelo de base de datos relacional.

Ejemplo 3.4.3 ▶

Selección

El **operador seleccionar** elige ciertas *n*–eadas de una relación. Las elecciones se hacen estableciendo condiciones sobre los atributos. Por ejemplo, para la relación JUGADOR dada en la tabla 3.4.1,

JUGADOR [Posición = c]

seleccionará los cuartetos

(22012, Johnsonbaugh, c, 22), (84341, Cage, c, 30) ◀

Ejemplo 3.4.4 ▶

Proyección

Mientras que el operador selección elige renglones de una relación, el **operador proyección** elige columnas. Además, elimina los duplicados. Por ejemplo, para la relación JUGADOR dada en la tabla 3.4.1,

JUGADOR[nombre, posición]

seleccionará las parejas

(Johnsonbaugh, c), (Glover, j), (Battey, p), (Cage, c),
(Homer, 1b), (Score, p), (Johnsonbaugh, j), (Singleton, 2b). ◀

Ejemplo 3.4.5 ▶

Unión

Los operadores selección y proyección manejan una sola relación; la **unión** maneja dos relaciones. La operación unión sobre las relaciones R_1 y R_2 comienza por examinar todos los pares de n-eadas, una de R_1 y otra de R_2. Si la condición de unión se satisface, las n-eadas se combinan para formar nuevas n-eadas. La condición de unión especifica una relación entre un atributo de R_1 y un atributo de R_2. Por ejemplo, se realizará una operación unión sobre las tablas 3.4.1 y 3.4.2. Como condición se tomará

Núm. identificación = NIP.

Tomamos un renglón de la tabla 3.4.1 y un renglón de la tabla 3.4.2 y si número de identificación = NIP, se combinan los renglones. Por ejemplo, el número de identificación 01180 en el quinto renglón (01180, Homer, 1b, 37) de la tabla 3.4.1 coincide con el NIP en el cuarto renglón (01180, Mutts) de la tabla 3.4.2. Estas n-eadas se combinan escribiendo primero la de la tabla 3.4.1, seguida de la n-eada de la tabla 3.4.2 y eliminando los elementos iguales en los atributos especificados para dar

(01180, Homer, 1b, 37, Mutts).

Esta operación se expresa como

JUGADOR [núm. identif. = NIP] ASIGNACIÓN.

La relación obtenida al ejecutar esta unión se muestra en la tabla 3.4.3.

TABLA 3.4.2 ■ ASIGNACIÓN

NIP	*Equipo*
39826	Blue Sox
26710	Mutts
58199	Jackalopes
01180	Mutts

TABLA 3.4.3 ■ JUGADOR [Núm. ident. = NIP] ASIGNACIÓN

Núm. ident.	*Nombre*	*Posición*	*Edad*	*Equipo*
58199	Battey	p	18	Jackalopes
01180	Homer	1b	37	Mutts
26710	Score	p	22	Mutts
39826	Singleton	2b	31	Blue Sox

Casi todas las consultas a una base de datos relacional requieren varias operaciones para proporcionar la respuesta.

Ejemplo 3.4.6 ▶

Describa las operaciones que proporcionan la respuesta a la consulta "encuentre los nombres de todas las personas que juegan para algún equipo".

Si primero se unen las relaciones dadas en las tablas 3.4.1 y 3.4.2 sujetas a la condición número de identificación = NIP, se obtendrá la tabla 3.4.3, que enumera a todas las personas que juegan para algún equipo, así como otros datos. Para obtener los nombres, sólo se necesita proyectar sobre el atributo nombre. Se obtiene la relación.

Nombre
Battey
Homer
Score
Singleton

Formalmente, estas operaciones se especificarían como

TEMP := JUGADOR [Núm. ident. = NIP] ASIGNACIÓN
TEMP [Nombre]

Ejemplo 3.4.7 ▶

Describa las operaciones que proporcionan la respuesta a la consulta "encuentre los nombres de todas las personas que juegan para Mutts".

Si primero se usa el operador selección para elegir renglones de la tabla 3.4.2 que se refieran a los jugadores de Mutts, se obtiene la relación

TEMP1

NIP	*Equipo*
26710	Mutts
01180	Mutts

Si ahora se unen la tabla 3.4.1 y la relación TEMP1 sujeta a Número de Identificación = NIP, se obtiene la relación

TEMP2

Núm. ident.	*Nombre*	*Posición*	*Edad*	*Equipo*
01180	Homer	1b	37	Mutts
26710	Score	p	22	Mutts

Si se proyecta la relación TEMP2 sobre el atributo Nombre, se obtiene la relación

Nombre
Homer
Score

Estas operaciones se especificarían formalmente como sigue:

TEMP1:= ASIGNACIÓN[Equipo = Mutts]
TEMP2:= JUGADOR[Núm. ident. = NIP]TEMP1
TEMP2[Nombre] ◀

Observe que las operaciones

TEMP1:= JUGADOR[Núm. ident. = NIP]ASIGNACIÓN
TEMP2:= TEMP1[Equipo = Mutts]
TEMP2[Nombre]

también darían respuesta a la consulta del ejemplo 3.4.7.

Sugerencias para resolver problemas

Una base de datos relacional representa datos como tablas (relaciones *n*-arias). La información de la base de datos se obtiene manejando las tablas. En esta sección, se analizaron las operaciones *selección* (consistente en elegir renglones especificados por una condición dada), *proyección* (consistente en elegir columnas especificadas por una condición dada) y *unión* (que implica combinar renglones de dos (o más) tablas según se especifica por una condición).

Sección de ejercicios de repaso

1. ¿Qué es una relación *n*-aria?

2. ¿Qué es un sistema de administración de bases de datos?

3. ¿Qué es una base de datos relacional?

4. ¿Qué es una llave?

5. ¿Qué es una consulta?

6. Explique cómo trabaja el operador selección y dé un ejemplo.

7. Explique cómo trabaja el operador proyección y dé un ejemplo.

8. Explique cómo trabaja el operador unión y dé un ejemplo.

Ejercicios

1. Exprese la relación de la tabla 3.4.4 como un conjunto de *n*-eadas.

TABLA 3.4.4 ■ EMPLEADO

Ident.	*Nombre*	*Administrador*
1089	Suzuki	Zamora
5620	Kaminski	Jones
9354	Jones	Yu
9551	Ryan	Washington
3600	Beaulieu	Yu
0285	Schmidt	Jones
6684	Manacotti	Jones

2. Exprese la relación de la tabla 3.4.5 como un conjunto de *n*-eadas.

TABLA 3.4.5 ■ DEPARTAMENTO

Dept.	*Administrador*
23	Jones
04	Yu
96	Zamora
66	Washington

3. Exprese la relación de la tabla 3.4.6 como un conjunto de *n*-eadas.

TABLA 3.4.6 ■ PROVEEDOR

Dept.	*Parte núm.*	*Cantidad*
04	335B2	220
23	2A	14
04	8C200	302
66	42C	3
04	900	7720
96	20A8	200
96	1199C	296
23	772	39

4. Exprese la relación de la tabla 3.4.7 como un conjunto de *n*-eadas.

TABLA 3.4.7 ■ COMPRADOR

Nombre	*Parte núm.*
United Supplies	2A
ABC Unlimited	8C200
United Supplies	1199C
JCN Electronics	2A
United Supplies	335B2
ABC Unlimited	772
Danny's	900
United Supplies	772
Underhanded Sales	20A8
Danny's	20A8
DePaul University	42C
ABC Unlimited	20A8

En los ejercicios 5 al 20, escriba una secuencia de operaciones que responda a una consulta. Además, proporcione una respuesta a la consulta. Utilice las tablas 3.4.4 a 3.4.7.

5. Encuentre los nombres de todos los empleados. (No incluya a los administradores).

6. Encuentre los nombres de todos los administradores.

7. Encuentre todos los números de partes.

8. Encuentre los nombres de todos los compradores.

9. Encuentre los nombres de todos los empleados administrados por Jones.

10. Encuentre todos los números de partes suministradas por el departamento 96.

11. Encuentre todos los compradores de la parte 20A8.

12. Encuentre todos los empleados del departamento 04.

13. Encuentre los números de partes de las que hay al menos 100 artículos disponibles.

14. Encuentre todos los números de departamentos que entregan partes a Danny′s.

15. Encuentre los números de partes y las cantidades compradas por United Supplies.

16. Encuentre todos los administradores de departamentos que producen partes para ABC Unlimited.

17. Encuentre los nombres de todos los empleados que trabajan en departamentos que suministran partes a JCN Electronics.

18. Encuentre todos los compradores de partes del departamento administrado por Jones.

19. Encuentre todos los compradores de partes producidas por el departamento para el que trabaja Suzuki.

20. Encuentre todos los números de partes y cantidades para el departamento de Zamora.

21. Establezca al menos tres relaciones *n*-arias con datos artificiales que puedan usarse en una base de datos médica. Ilustre cómo se usaría su base de datos proponiendo y respondiendo dos consultas. Además, escriba una secuencia de operaciones que sirva para responder la consulta.

22. Describa una operación *unión* sobre una base de datos relacional. Ilustre cómo funciona su operador respondiendo la siguiente consulta, usando las relaciones de las tablas 3.4.4 a 3.4.7: Encuentre los nombres de todos los empleados que trabajan en el departamento 23 o 96. Además, escriba una secuencia de operaciones que sirva para responder la consulta.

23. Describa una operación *intersección* en una base de datos relacional. Ilustre como trabajaría su operador respondiendo la siguiente consulta, usando las relaciones de las tablas 3.4.4 a 3.4.7: Encuentre los nombres de todos los compradores de las partes 2A y las partes 1199C. Además, escriba una secuencia de operaciones que sirva para responder la consulta.

24. Describa una operación *diferencia* en una base de datos relacional. Ilustre cómo trabajaría su operador respondiendo la siguiente consulta, usando las relaciones de las tablas 3.4.4 a 3.4.7: Encuentre los nombres de todos los empleados que no trabajan en el departamento 04. Además, escriba una secuencia de operaciones que sirva para responder la consulta.

Nota

Casi todas las referencias generales de matemáticas discretas manejan relaciones. [Codd; Date; Kroenke; y Ullman] son referencias recomendables para bases de datos y el modelo relacional en particular.

Repaso del capítulo

Sección 3.1

1. Relación binaria de X a Y: conjunto de pares ordenados (x, y), $x \in X$, $y \in Y$
2. Dominio de una relación binaria R: $\{x \mid (x, y) \in R\}$
3. Rango de una relación binaria R: $\{y \mid (x, y) \in R\}$
4. Digráfica de una relación binaria
5. Relación reflexiva R sobre X: $(x, x) \in R$ para toda $x \in X$
6. Relación simétrica R sobre X: para toda $x, y \in X$, si $(x, y) \in X$, entonces $(y, x) \in R$
7. Relación antisimétrica R sobre X: para toda $x, y \in X$, si $(x, y) \in R$ y $x \neq y$, entonces $(y, x) \notin R$
8. Relación transitiva R sobre X: para toda $x, y, z \in X$, si (x, y) y $(y, z) \in R$, entonces $(x, z) \in R$
9. Orden parcial: relación que es reflexiva, antisimétrica y transitiva
10. Relación inversa R^{-1}: $\{(y, x) \mid (x, y) \in R\}$
11. Composición de relaciones $R_2 \circ R_1$: $R_2 \circ R_1$: $\{(x, z) \mid (x, y) \in R_1$ y $(y, z) \in R_2$

Sección 3.2

12. Relación de equivalencia: relación que es reflexiva, simétrica y transitiva
13. Clase de equivalencia que contiene a a, dada por la relación de equivalencia R: $[a] = \{x \mid x\, R\, a\}$
14. Partición del conjunto mediante clases de equivalencia (Teorema 3.2.8)

Sección 3.3

15. Matriz de una relación
16. R es una relación reflexiva si y sólo si la diagonal principal de la matriz de R contiene unos.
17. R es una relación simétrica si y sólo si la matriz de R es simétrica respecto a la diagonal principal.
18. Si A_1 es la matriz de la relación R_1 y A_2 es la matriz de la relación R_2, la matriz de la relación $R_2 \circ R_1$ se obtiene sustituyendo cada término diferente de cero en la matriz del producto A_1A_2 por 1.
19. Si A es la matriz de la relación R, R es transitiva si y sólo si siempre que el elemento i, j en A^2 sea diferente de cero, el elemento i, j en A también es diferente de cero.

Sección 3.4

20. Relación n-aria: conjunto de n-eadas
21. Sistema de administración de bases de datos
22. Base de datos relacional
23. Llave
24. Consulta (*query*)
25. Selección
26. Proyección
27. Unión

Autoevaluación del capítulo

Sección 3.1

En los ejercicios 1 y 2, determine si la relación definida en el conjunto de enteros positivos es reflexiva, simétrica, antisimétrica, transitiva y/o de orden parcial.

1. $(x,y) \in R$ si 2 divide a $x + y$

2. $(x,y) \in R$ si 3 divide a $x + y$

3. Proporcione un ejemplo de una relación en $\{1, 2, 3, 4\}$ que sea reflexiva, no antisimétrica y no transitiva.

4. Suponga que R es una relación sobre X que es simétrica, y transitiva pero no reflexiva. Suponga también que $|X| \geq 2$. Defina una relación $\overline{R}$ sobre X por

$$\overline{R} = X \times X - R.$$

¿Cuál de las siguientes afirmaciones debe ser verdadera? Para cada afirmación falsa, proporcione un contraejemplo.

a) $\overline{R}$ es reflexiva.

b) $\overline{R}$ es simétrica.

c) $\overline{R}$ es no antisimétrica.

d) $\overline{R}$ es transitiva.

Sección 3.2

5. La relación

$$\{(1, 1), (1, 2), (2, 2), (4, 4), (2, 1), (3, 3)\}$$

¿es una relación de equivalencia en $\{1, 2, 3, 4\}$? Explique su respuesta.

6. Puesto que la relación

$$\{(1, 1), (2, 2), (3, 3), (4, 4), (1, 2), (2, 1), (3, 4), (4, 3)\}$$

es una relación de equivalencia en $\{1, 2, 3, 4\}$, encuentre [3], las clases de equivalencia que contienen a 3. ¿Cuántas clases de equivalencia (diferentes) hay?

7. Encuentre la relación de equivalencia (como un conjunto de pares ordenados) en $\{a, b, c, d, e\}$ cuyas clases de equivalencia son

$$\{a\}, \quad \{b, d, e\}, \quad \{c\}.$$

8. Sea R la relación definida en el conjunto de cadenas de 8 bits por $s_1 \, R \, s_2$ siempre que s_1 y s_2 tengan el mismo número de ceros.

a) Demuestre que R es una relación de equivalencia.

b) ¿Cuántas clases de equivalencia hay?

c) Liste un miembro de cada clase de equivalencia.

Sección 3.3

Los ejercicios 9 al 12 se refieren a las relaciones

$$R_1 = \{(1, x), (2, x), (2, y), (3, y)\}, \qquad R_2 = \{(x, a), (x, b), (y, a), (y, c)\}.$$

9. Encuentre la matriz A_1 de la relación R_1 relativa a los órdenes

$$1, 2, 3; x, y.$$

10. Encuentre la matriz A_2 de la relación R_2 relativa a los órdenes

$$x, y; a, b, c.$$

11. Encuentre la matriz producto $A_1 \, A_2$.

12. Use el resultado del ejercicio 11 para encontrar la matriz de la relación $R_1 \circ R_2$

Sección 3.4

En los ejercicios 13 al 16, escriba una secuencia de operaciones para responder a la consulta. Además, proporcione una respuesta a la consulta. Use las tablas 3.4.1 y 3.4.2.

13. Encuentre todos los equipos.

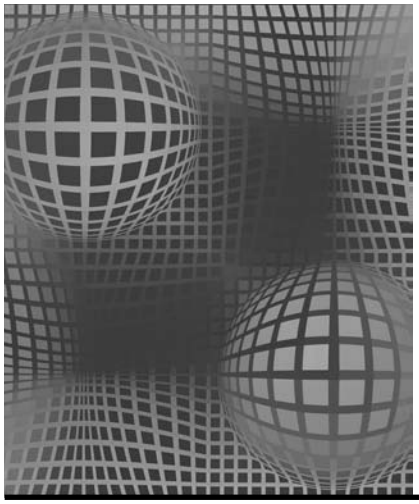
14. Encuentre todos los nombres y edades de los jugadores.

15. Encuentre los nombres de todos los equipos que tienen pitcher.

16. Encuentre los nombres de todos los equipos que tienen jugadores de 30 años o mayores.

Ejercicios para computadora

1. Escriba un programa que encuentre el dominio de una relación.
2. Escriba un programa que encuentre el rango de una relación.
3. Escriba un programa que determine si una relación es reflexiva.
4. Escriba un programa que determine si una relación es antisimétrica.
5. Escriba un programa que determine si una relación es transitiva.
6. Escriba un programa que determine la inversa de una relación.
7. Escriba un programa que encuentre la composición $R \circ S$ de las relaciones R y S.
8. Escriba un programa que verifique si una relación R es una relación de equivalencia. Si R es una relación de equivalencia, la salida del programa debe ser las clases de equivalencia de R.
9. Escriba un programa para determinar si una relación es una función del conjunto X al conjunto Y.
10. [*Proyecto*] Prepare un informe de una base de datos relacional comercial, como Oracle o Access.



Capítulo 4

ALGORITMOS

Es tan sencillo.
Paso 1: Encontramos la peor obra del mundo, un fracaso seguro.
Paso 2: Reúno un millón de dólares, hay muchas viejitas lindas en el mundo.
Paso 3: Tú te vas a trabajar en los libros. Una lista ficticia de avales, una para el gobierno, otra para nosotros. Tú puedes hacerlo Bloom, eres un mago.
Paso 4: Abrimos en Broadway y antes de que puedas decir algo.
Paso 5: Cerramos en Broadway.
Paso 6: Nos llevamos el millón y volamos a Río de Janeiro.

DE *THE PRODUCERS*

Un **algoritmo** es un método paso a paso para resolver algunos problemas. Los enfoques de este tipo para resolver problemas no son novedad; de hecho, la palabra "algoritmo" se deriva del nombre del matemático persa del siglo IX, al-Khowārizmī En la actualidad, "algoritmo" suele referirse a una solución ejecutable por una computadora. En este libro, la preocupación principal será que se pueda ejecutar en una computadora "tradicional", es decir, una computadora personal, con un solo procesador que ejecute instrucciones paso a paso.

Después de introducir los algoritmos y proporcionar varios ejemplos, nos dedicaremos al análisis de los algoritmos, es decir, al tiempo y espacio requeridos para ejecutarlos. Concluiremos con un examen de los algoritmos recursivos, los que se refieren o recurren a sí mismos.

4.1 ➜ Introducción

Por lo general, los algoritmos tienen las siguientes características:

- ***Entrada*** El algoritmo recibe datos de *entrada*.
- ***Salida*** El algoritmo produce una *salida*.
- ***Precisión*** Los pasos se establecen con precisión.
- ***Determinismo*** Los resultados intermedios de cada paso de ejecución son únicos y están determinados sólo por las entradas y los resultados de los pasos anteriores.

- ***Carácter finito*** El algoritmo termina; es decir, se detiene después de ejecutar un número finito de instrucciones.
- ***Corrección*** La salida producida por el algoritmo es correcta; es decir, el algoritmo resuelve el problema sin errores.
- ***Generalidad*** El algoritmo se aplica a un conjunto de entradas.

Como ejemplo, considere el siguiente algoritmo que encuentra el máximo de tres números *a*, *b* y *c*:

1. *grande* = *a*,
2. Si *b* > *grande*, entonces *grande* = *b*,
3. Si *c* > *grande*, entonces *grande* = *c*.

(Como se explica en el apéndice C, = es el operador asignación.)

La idea del algoritmo es inspeccionar los números uno por uno y copiar el valor más grande encontrado a la variable *grande*. En la conclusión del algoritmo, *grande* será igual al valor mayor de los tres números.

Ahora se muestra la manera en que el algoritmo anterior se ejecuta para algunos valores específicos de *a*, *b* y *c*. Esta simulación se llama **seguimiento o rastreo**. Primero suponga que

$$a = 1, \quad b = 5, \quad c = 3$$

En la línea 1, *grande* tiene el valor de *a* (1). En la línea 2, *b* > *grande* (5 > 1) es verdadera, de manera que *grande* es igual a *b* (5). En la línea 3, *c* > *grande* (3 > 5) es falsa, y no hay cambio. En este punto *grande* es 5, el valor mayor entre *a*, *b* y *c*.

Suponga que

$$a = 6, \quad b = 1, \quad c = 9$$

En la línea 1, *grande* tiene el valor de *a* (6). En la línea 2, *b* > *grande* (1 > 6) es falsa, y no hay cambio. En la línea 3, *c* > *grande* (9 > 6) es verdadera, de manera que *grande* es igual a 9. En este punto *grande* es 9, el valor mayor entre *a*, *b* y *c*.

Se verifica que el ejemplo de algoritmo tiene las propiedades establecidas al inicio de esta sección.

El algoritmo recibe los tres valores *a*, *b* y *c* como entrada y produce el valor *grande* como salida.

Los pasos del algoritmo se establecen con suficiente precisión de manera que éste puede escribirse en un lenguaje de programación y ser ejecutado por una computadora.

A partir de los valores de entrada, cada paso intermedio de un algoritmo produce un resultado único. Por ejemplo, dados los valores

$$a = 1, \quad b = 5, \quad c = 3,$$

en la línea 2, *grande* tiene el valor 5 sin importar quién ejecuta el algoritmo.

El algoritmo termina después de un número finito de pasos (tres pasos) y contesta correctamente la pregunta planteada (encontrar el mayor de tres valores de entrada).

El algoritmo es general; permite encontrar el valor más grande de *cualesquiera* tres números.

La descripción de qué es un algoritmo será suficiente para las necesidades de este libro. Sin embargo, debe observarse que es posible dar una definición matemática precisa de "algoritmo" (vea las notas del capítulo 12).

Aunque el lenguaje común en ocasiones es adecuado para especificar un algoritmo, la mayoría de los matemáticos y los especialistas en ciencias de la computación prefieren el **seudocódigo** por su precisión, estructura y universalidad. El seudocódigo recibe este nombre porque se parece a un código real en lenguaje de computadora, como C++ y Java. Existen muchas versiones de seudocódigo. A diferencia de los lenguajes para computadora que deben preocuparse por puntos y comas, mayúsculas y minúsculas, palabras reservadas y otros elementos, cualquier versión de seudocódigo es aceptable siempre y cuando sus instrucciones no sean ambiguas. Nuestro seudocódigo se describe con detalle en el apéndice C.

En cuanto al ejemplo de un algoritmo escrito en seudocódigo, se escribe el primer algoritmo de esta sección que encuentra el máximo de tres números.

Algoritmo 4.1.1

Encuentra el máximo de tres números

Este algoritmo encuentra el más grande entre los números a, b y c.

```
      Entrada: a, b, c
      Salida: grande (el mayor de a, b y c)
1.    máx3(a, b, c){
2.        grande = a
3.        if(b > grande) // si b es mayor que grande, actualiza grande
4.            grande = b
5.        if(c > grande) // si c es mayor que grande, actualiza grande
6.            grande = c
7.        return grande
8.    }
```

Nuestros algoritmos consisten en un título, una breve descripción del algoritmo, la entrada y la salida del algoritmo, y las funciones que contienen las instrucciones del algoritmo. El algoritmo 4.1.1 consta de una sola función. Para hacer más conveniente la referencia a las líneas individuales dentro de una función, en ocasiones se numeran algunas de ellas. La función del algoritmo 4.1.1 tiene ocho líneas numeradas.

Cuando se ejecuta la función del algoritmo 4.1.1, en la línea 2 se hace *grande* igual a a. En la línea 3 se comparan b y *grande*. Si b es mayor que *grande*, se ejecuta la línea 4

$$grande = b$$

pero si b no es mayor que *grande*, se salta a la línea 5. En la línea 5 se comparan c y *grande*. Si c es mayor que *grande*, se ejecuta la línea 6

$$grande = c$$

pero si c no es mayor que *grande*, se salta a la línea 7. Así, al llegar a la línea 7, el valor de *grande* será correctamente el mayor entre a, b y c.

En la línea 7 se regresa el valor de *grande*, que es igual al mayor de los números a, b y c, al que invoca la función y la termina. El algoritmo 4.1.1 ha encontrado correctamente el mayor de los tres números.

El método del algoritmo 4.1.1 resulta útil para encontrar el valor más grande en una sucesión.

Algoritmo 4.1.2

Encuentra el valor máximo en una sucesión

Este algoritmo encuentra el número más grande de $s_1, \ldots, s_n$.

```
Entrada: s, n
Salida: grande (el valor mayor en una sucesión s)
máx(s, n){
    grande = s_1
    for i = 2 to n
        if(s_i > grande)
            grande = s_i
    return grande
}
```

Se verifica que el algoritmo 4.1.2 es correcto probando que

$$grande \text{ es el valor mayor en la sucesión } s_1, \ldots, s_i$$

es un ciclo invariante, usando inducción sobre i.

Para el paso base ($i = 1$), se observa que antes de que inicie la ejecución del ciclo "for", se asigna a *grande* el valor s_1; entonces *grande* es sin duda el valor mayor de la subsucesión s_1.

Suponga que *grande* es el valor mayor en la subsucesión $s_1, \ldots, s_i$. Si $i < n$ es verdadera (de manera que el cuerpo del ciclo "for" se ejecuta de nuevo), i se convierte en $i + 1$. Suponga primero que $s_{i+1} > grande$. Entonces s_{i+1} es el valor mayor en la subsucesión $s_1, \ldots, s_i, s_{i+1}$. En este caso, el algoritmo asigna a *grande* el valor s_{i+1}. Ahora *grande* es igual al mayor valor en la subsucesión $s_1, \ldots, s_i, s_{i+1}$. Suponga ahora que $s_{i+1} \leq grande$. Entonces se concluye que *grande* es el valor mayor en la subsucesión $s_1, \ldots, s_i, s_{i+1}$. En este caso, el algoritmo *no* cambia el valor de *grande*; así, *grande* es el mayor valor en la subsucesión $s_1, \ldots, s_i, s_{i+1}$. Se ha probado el paso inductivo; por lo tanto,

grande es el mayor valor en la subsucesión $s_1, \ldots, s_i$

es un ciclo invariante.

El ciclo "for" termina cuando $i = n$. Como

grande es el mayor valor en la subsucesión $s_1, \ldots, s_i$

es un ciclo invariante, en este punto *grande* es el valor más grande en la subsucesión $s_1, \ldots, s_n$. Por lo tanto, el algoritmo 4.1.2 es correcto.

Sugerencias para resolver problemas

Para construir un algoritmo, con frecuencia es útil suponer que uno está a la mitad del algoritmo y parte del problema ya se resolvió. Por ejemplo, al encontrar el elemento más grande en una sucesión $s_1, \ldots, s_n$ (algoritmo 4.1.2), fue útil suponer que ya se había encontrado el elemento mayor *grande* en la subsucesión $s_1, \ldots, s_i$. Entonces, sólo hubo que ver el siguiente elemento s_{i+1} y si s_{i+1} era mayor que *grande*, simplemente se actualizaba *grande*. Si s_{i+1} no era mayor que *grande*, no se modificaba. Iterando este procedimiento se llega al algoritmo. Esta observación también llevó al ciclo invariante

grande es el valor mayor en la subsucesión $s_1, \ldots, s_i$,

que nos permitió *probar* que el algoritmo 4.1.2 era correcto.

Sección de ejercicios de repaso

1. ¿Qué es un algoritmo?

2. Describa las siguientes propiedades que por lo general tiene un algoritmo: entrada, salida, precisión, determinismo, carácter finito, corrección y generalidad.

3. ¿Qué es el seguimiento (o rastreo) de un algoritmo?

4. ¿Cuáles son las ventajas del seudocódigo sobre el texto común al escribir un algoritmo?

5. ¿Cómo se relacionan los algoritmos con las funciones del seudocódigo?

Ejercicios

1. Consulte en el directorio telefónico las instrucciones para llamar de larga distancia. ¿Qué propiedades de un algoritmo (entrada, salida, precisión, determinismo, carácter finito, corrección, generalidad) están presentes? ¿Qué propiedades faltan?

2. La conjetura de Goldbach establece que todo número par mayor que 2 es la suma de dos números primos. El siguiente es un algoritmo propuesto para verificar si la conjetura de Goldbach es verdadera:

1. Sea $n = 4$.
2. Si n no es la suma de dos primos, la salida es "no" y se detiene.
3. De otra manera, se aumenta n en 2 y se sigue al paso 2.
4. La salida es "sí" y se detiene.

¿Qué propiedades de un algoritmo (entrada, salida, precisión, determinismo, carácter finito, corrección, generalidad) tiene este algoritmo propuesto? ¿Depende alguna de ellas de la verdad de la conjetura de Goldbach (que todavía no resuelven los matemáticos)?

3. Escriba un algoritmo que encuentre el elemento menor entre a, b y c.

4. Escriba un algoritmo que encuentre el segundo elemento más pequeño entre a, b y c. Suponga que los valores de a, b y c son diferentes.

5. Escriba un algoritmo que regrese el valor más pequeño en la sucesión $s_1, \ldots, s_n$.

6. Escriba un algoritmo que regrese el valor más grande y el segundo elemento más grande en la sucesión $s_1, \ldots, s_n$. Suponga que $n > 1$ y que los valores de la sucesión son diferentes.

7. Escriba un algoritmo que regrese el valor más pequeño y el segundo elemento más pequeño en la sucesión $s_1, \ldots, s_n$. Suponga que $n > 1$ y que los valores de la sucesión son diferentes.

8. Escriba un algoritmo cuya salida sea el valor menor y mayor en la sucesión $s_1, \ldots, s_n$.

9. Escriba un algoritmo que regrese el índice de la primera ocurrencia del elemento más grande en la sucesión $s, \ldots, s_n$. *Ejemplo:* Si la sucesión es

6.2 8.9 4.2 8.9,

el algoritmo regresa el valor 2.

10. Escriba un algoritmo que regrese el índice de la última ocurrencia del elemento más grande en la sucesión $s_1, \ldots, s_n$. *Ejemplo:* Si la sucesión es

6.2 8.9 4.2 8.9,

el algoritmo regresa el valor 4.

11. Escriba un algoritmo que produzca la suma de la sucesión de números $s_1, \ldots, s_n$.

12. Escriba un algoritmo que regrese el índice del primer elemento que es menor que su predecesor en la sucesión $s_1, \ldots, s_n$. Si s está en orden no decreciente, el algoritmo regresa el valor 0. *Ejemplo:* Si la sucesión es

AMY BRUNO ELIE DAN ZEKE,

el algoritmo proporciona el valor 4.

13. Escriba un algoritmo que regrese el índice del primer elemento que es mayor que su predecesor en la sucesión $s_1, \ldots, s_n$. Si s está en orden no decreciente, el algoritmo regresa el valor 0. *Ejemplo:* Si la sucesión es

AMY BRUNO ELIE DAN ZEKE,

el algoritmo regresa el valor 2.

14. Escriba un algoritmo que invierta la sucesión $s_1, \ldots, s_n$. *Ejemplo:* Si la sucesión es

AMY BRUNO ELIE,

la sucesión invertida es

ELIE BRUNO AMY.

15. Escriba un método estándar para sumar dos enteros decimales positivos, que se enseña en primaria, como un algoritmo.

16. Escriba un algoritmo que recibe como entrada la matriz A de $n \times n$ y produce la transpuesta A^T.

17. Escriba un algoritmo que recibe como entrada la matriz de una relación R y prueba si R es reflexiva.

18. Escriba un algoritmo que recibe como entrada la matriz de una relación R y prueba si R es simétrica.

19. Escriba un algoritmo que recibe como entrada la matriz de una relación R y prueba si R es transitiva.

20. Escriba un algoritmo que recibe como entrada la matriz de una relación R y prueba si R es antisimétrica.

21. Escriba un algoritmo que recibe como entrada la matriz de una relación R y prueba si R es una función.

22. Escriba un algoritmo que recibe como entrada la matriz de una relación R y produce como salida la matriz de la relación inversa R^{-1}.

23. Escriba un algoritmo que recibe como entrada las matrices de las relaciones R_1 y R_2 y produce como salida la matriz de la composición $R_1 \circ R_2$.

24. Escriba un algoritmo cuya entrada sea una sucesión $s_1, \ldots, s_n$ y un valor x. (Suponga que todos los valores son números reales.) El algoritmo regresa verdadero si $s_i + s_j = x$, para alguna $i \neq j$, y falso de otra manera. *Ejemplo:* Si la sucesión de entrada es

2, 12, 6, 14

y $x = 26$, el algoritmo regresa verdadero porque $12 + 14 = 26$. Si la sucesión de entrada es

2, 12, 6, 14

y $x = 4$, el algoritmo regresa falso porque ningún par de términos *distintos* en la sucesión suma 4.

4.2 ➜ Ejemplos de algoritmos

Los algoritmos se concibieron para resolver muchos problemas. En esta sección se dan ejemplos de varios algoritmos útiles. En el resto del libro, se investigarán muchos otros algoritmos.

Búsqueda

Las computadoras destinan una gran cantidad de tiempo a buscar. Cuando un cajero busca un registro en un banco, un programa de computadora realiza la búsqueda. Tratar de encontrar una solución a un crucigrama o un movimiento óptimo en un juego se puede establecer como un problema de búsqueda. Usar una máquina de búsqueda en Internet es otro ejemplo de un problema de búsqueda. Encontrar un texto específico en un documento al correr un procesador de textos es otro ejemplo de un problema de búsqueda. Se analizará un algoritmo para resolver el problema de buscar un texto.

Suponga que se da el texto t (por ejemplo, un documento en un procesador de palabras) y queremos encontrar la primera ocurrencia del patrón p en t (por ejemplo, se desea encontrar la primera ocurrencia de la cadena $p =$ "Nueva Escocia" en t) o determinar que p no aparece en t. Se indexan los caracteres en t comenzando con 1. Un enfoque para buscar p es verificar si p ocurre en el índice 1 en t. Si así es, nos detenemos, puesto que ya se

encontró la primera ocurrencia de p en t. Si no, se verifica si p ocurre en el índice 2 en t. Si así es, nos detenemos, pues ya se encontró la primera ocurrencia de p en t. Si no, se verifica el índice 3 en t, y así sucesivamente.

El algoritmo de búsqueda de texto se establece como el algoritmo 4.2.1.

Algoritmo 4.2.1

Búsqueda de texto

Este algoritmo busca una ocurrencia del patrón p en el texto t. Regresa el índice más pequeño i tal que p ocurre en t comenzando en el índice i. Si p no ocurre en t, regresa 0.

```
Entrada: p (indexada de 1 a m), m, t (indexada de 1 a n), n
Salida:  i
busca_texto(p, m, t, n){
    for i = 1 to n − m + 1{
        j = 1
        // i es el índice en t del primer carácter de la
        // subcadena para comparar con p, y j es el índice
        // en p

        // el ciclo "while" compara t_i . . . t_{i+m−1} y p_1 . . . p_m
        while (t_{i+j−1} == p_j){
            j = j + 1
            if(j > m)
             return i
        }
    }
    return 0
}
```

La variable i marca el índice en t del primer carácter de la subcadena para compararlo con p. El algoritmo primero intenta $i = 1$, después $i = 2$, y así sucesivamente. El índice $n - m + 1$ es el último valor posible para i ya que, en este punto, la cadena $t_{n-m+1}t_{n-m+2}\cdots t_m$ tiene la longitud exacta m.

Después de establecer el valor i, el ciclo "while" compara $t_i \ldots t_{i+m-1}$ y $p_1 \ldots p_m$. Si los caracteres coinciden,

$$t_{i+j-1} == p_j$$

j se incrementa

$$j = j + 1$$

y se comparan los siguientes caracteres. Si j es $m + 1$, todos los m caracteres coinciden y hemos encontrado p en el índice i en t. En este caso, el algoritmo regresa i:

```
if (j > m)
    return i
```

Si el ciclo corrió hasta completarse, nunca se encontró una coincidencia, en cuyo caso el algoritmo regresa 0.

Ejemplo 4.2.2 ▶

La figura 4.2.1 muestra un rastreo del algoritmo 4.2.1 donde se busca el patrón "001" en el texto "010001". ◀

Orden

Ordenar (*sort*) una sucesión significa ponerla en algún orden específico. Si se tiene una sucesión de nombres, tal vez se desee ordenarla en orden no decreciente de acuerdo con el diccionario.

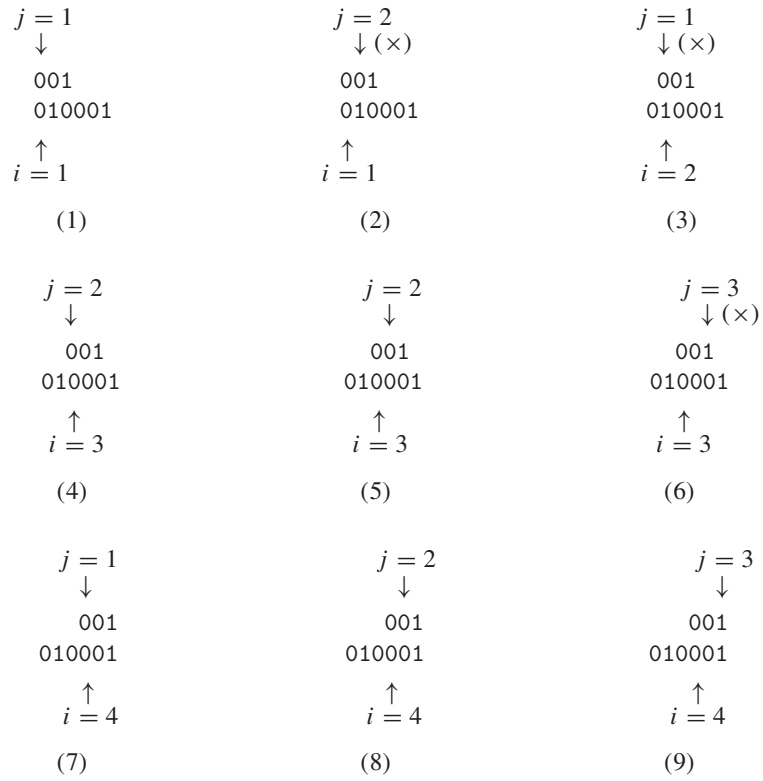


Figura 4.2.1 Búsqueda de "001" en "010001" usando el algoritmo 4.2.1. La cruz (×) en los pasos (2), (3) y (6) marcan una diferencia.

Por ejemplo, si la sucesión es

Jones, Johnson, Appel, Zamora, Chu,

después de ordenar la sucesión en orden no decreciente, se obtendría

Appel, Chu, Johnson, Jones, Zamora.

Una ventaja importante al usar una sucesión ordenada en lugar de una sin ordenar es que es mucho más sencillo encontrar un elemento específico. Imagine intentar encontrar el número del teléfono de un individuo en particular en el directorio de la ciudad de Nueva York ¡si los nombres no estuvieran en orden!

Se han diseñado muchos algoritmos para ordenar (vea [Knuth, 1998b]). Decidir qué algoritmo es preferible en una situación específica depende de factores como el tamaño de los datos y cómo estén representados. Analicemos la **inserción por orden (*insertion sort*)**, que es uno de los algoritmos más rápidos para ordenar sucesiones pequeñas (alrededor de 50 elementos).

Se supone que la entrada a inserción por orden es

$$s_1, \ldots, s_n$$

y la meta es colocar los datos en orden no decreciente. En la i-ésima iteración de la inserción por orden, la primera parte de la sucesión

$$s_1, \ldots, s_i$$

se habrá rearreglado de manera que esté ordenada. (En breve se explicará cómo se ordena $s_1, \ldots, s_i$). Después la inserción por orden *inserta* s_{i+1} en

$$s_1, \ldots, s_i$$

de manera que

$$s_1, \ldots, s_i, s_{i+1}$$

está ordenada.

Por ejemplo, suponga que $i = 4$ y $s_1, \ldots, s_4$ es

8	13	20	27

Si s_5 es 16, después de insertarlo, $s_1, \ldots, s_5$ se convierte en

8	13	16	20	27

Observe que 20 y 27, al ser *mayores que* 16, se mueven un índice a la derecha para que entre 16. Así, la parte de "inserción" del algoritmo es: Comenzando por la derecha de la sucesión ordenada, un elemento se mueve un índice a la derecha si es mayor que el elemento que se inserta. Esto se repite hasta llegar al primer índice o encontrar un elemento menor o igual que el elemento que se inserta.

Por ejemplo, para insertar 16 en

8	13	20	27

primero se comparan 16 y 27. Como 27 es mayor que 16, 27 se mueve un índice a la derecha:

8	13	20		27

Después se comparan 16 y 20. Como 20 es mayor que 16, 20 se mueve un índice a la derecha:

8	13		20	27

Ahora se comparan 16 y 13. Como 13 es menor o igual que 16, se inserta (es decir, se copia) 16 al tercer índice:

8	13	16	20	27

Con esto la subsucesión queda ordenada.

Una vez explicada la idea clave de la inserción por orden, completamos la explicación del algoritmo. La inserción por orden comienza por insertar s_2 en la subsucesión s_1. Observe que s_1 por sí misma está ordenada. Ahora s_1, s_2 está ordenada. Luego, la inserción por orden inserta s_3 en la susbsucesión ordenada s_1, s_2. Ahora s_1, s_2, s_3 está ordenada. Este procedimiento continúa hasta que la inserción por orden inserta s_n en la subsucesión ordenada $s_1, \ldots, s_{n-1}$. Ahora la sucesión completa $s_1, \ldots, s_n$ está ordenada. Se obtiene el siguiente algoritmo.

Algoritmo 4.2.3

Inserción por orden

Este algoritmo ordena la sucesión $s_1, \ldots, s_n$ en orden no decreciente.

```
Entrada: s, n
Salida:  s (ordenada)

inserción_por_orden(s, n){
    for i = 2 to n {
        val = s_i //guarda s_i para insertarla en el lugar correcto
        j = i − 1
        // si val < s_j, se mueve s_j a la derecha para dejar lugar a s_i
        while (j ≥ 1 ∧ val < s_j){
            s_{j+1} = s_j
            j = j − 1
        }
        s_{j+1} = val // inserta val
    }
}
```

Se deja la prueba de que el algoritmo 4.2.3 es correcto como ejercicio (vea el ejercicio 12).

Tiempo y espacio para los algoritmos

Es importante saber o poder estimar el tiempo (por ejemplo, el número de pasos) y el espacio (como número de variables, longitud de las sucesiones) requeridos por los algoritmos. Conocer el tiempo y el espacio que requiere un algoritmo permite comparar algoritmos que resuelven el mismo problema. Por ejemplo, si un algoritmo toma n pasos para resolver un problema y otro toma n^2 pasos para hacer lo mismo, sin duda será preferible el primero, suponiendo que los requerimientos de espacio son aceptables. En la sección 4.3 se dan las definiciones técnicas que permiten hacer afirmaciones rigurosas acerca del tiempo y espacio requeridos por los algoritmos.

El ciclo "for" en el algoritmo 4.2.3 siempre se ejecuta $n - 1$ veces, pero el número de veces que se ejecuta el ciclo "while" para un valor específico de i depende de los datos de entrada. Así, aun para un tamaño fijo n, el tiempo requerido por el algoritmo 4.2.3 depende de la entrada. Por ejemplo, si la sucesión de entrada ya está ordenada en orden no decreciente,

$$val < s_j$$

será falso siempre, y el cuerpo del ciclo while nunca se ejecutará. Esto se llama **tiempo del mejor caso (*best-case time*)**.

Por otro lado, si la sucesión está ordenada en orden *decreciente*,

$$val < s_j$$

será verdadera siempre, y el ciclo while se ejecutará el número máximo de veces. (El ciclo while se ejecutará $i - 1$ veces durante la i-ésima iteración del ciclo). Esto se llama **tiempo del peor caso (*worst-case time*)**.

Algoritmos aleatorizados

En ocasiones es necesario relajar los requerimientos para los algoritmos establecidos en la sección 4.1. Muchos de los algoritmos que se usan en la actualidad no son generales, determinísticos o ni siquiera finitos. Un sistema operativo (como Windows XP), por ejemplo, es mejor pensarlo como un programa que nunca termina en lugar de como un programa finito con entrada y salida. Los algoritmos escritos para más de un procesador, ya sea una máquina con multiprocesador o un entorno distribuido (como Internet), rara vez son determinísticos. Además, muchos problemas prácticos son demasiado difíciles para resolverlos

con eficiencia y comprometen la generalidad o la corrección necesaria. Como ilustración, se presenta un ejemplo que muestra la utilidad de permitir que un algoritmo tome decisiones aleatorias, violando el requerimiento de determinismo.

Un **algoritmo aleatorizado** *no* requiere que los resultados intermedios de cada paso de la ejecución estén definidos de manera única y dependan sólo de las entradas y los resultados de los pasos anteriores. Por definición, cuando un algoritmo aleatorizado se ejecuta, en algún punto hace una elección *aleatoria*. En la práctica, se usa un generador de números seudoaleatorios (vea el ejemplo 2.2.15).

Debe suponerse la existencia de una función

$$rand(i, j),$$

que regresa un entero aleatorio entre los enteros i y j, inclusive. Como ejemplo, se describe un algoritmo aleatorizado que desordena una sucesión de números. De manera más precisa, la entrada es la sucesión $a_1, \ldots, a_n$ y mueve los números a posiciones aleatorias. Los grandes torneos de *bridge* usan programas de computadora para barajar las cartas.

El algoritmo primero intercambia (es decir, cambia los valores de) a_1 y $a_{rand(1,n)}$. En este punto, el valor de a_1 puede ser igual a *cualquiera* de los valores originales de la sucesión. Después, el algoritmo intercambia a_2 y $a_{rand(2,n)}$. Ahora el valor de a_2 puede ser igual a cualquiera de los valores restantes de la sucesión. El algoritmo continúa hasta que intercambia a_{n-1} y $a_{rand(n-1,n)}$ y la sucesión completa está desordenada.

Algoritmo 4.2.4

Desordenar (shuffle)

Este algoritmo desordena los valores de una sucesión

$$a_1, \ldots, a_n.$$

```
    Entrada: a, n
Salida: a (desordenada)

desordena(a, n){
    for i = 1 to n − 1
        swap(a_i, a_rand(i,n))
}
```

Ejemplo 4.2.5 ▶

Suponga que la sucesión a

17	9	5	23	21

es la entrada que se va a *desordenar*. Primero se cambian a_i y a_j, donde $i = 1$ y $j = rand(1, 5)$. Si $j = 3$, después del intercambio (swap) se tiene

5	9	17	23	21
↑ i		↑ j		

Luego, $i = 2$. Si $j = rand(2, 5) = 5$, después del intercambio se tiene

5	21	17	23	9
	↑ i			↑ j

Ahora $i = 3$. Si $j = rand(3, 5) = 3$, la sucesión no cambia.
Por último, $i = 4$. Si $j = rand(4, 5) = 5$, después del intercambio se tiene

5	21	17	9	23
			↑ i	↑ j

Observe que la salida (es decir, la sucesión desordenada) depende de las elecciones aleatorias que hace el generador de números aleatorios. ◀

Sugerencias para resolver problemas

De nuevo, se hace hincapié en que, al construir un algoritmo, con frecuencia es útil suponer que uno está a la mitad del algoritmo y que esa parte del problema está resuelta. En la inserción por orden (algoritmo 4.2.3) fue útil *suponer* que las subsucesión $s_1, \ldots, s_i$ estaba ordenada. Entonces, todo lo que había que hacer era insertar el siguiente elemento s_{i+1} en el lugar adecuado. Al iterar dicho procedimiento se obtiene el algoritmo. Tales observaciones llevan a un ciclo invariante que sirve para probar que el algoritmo 4.2.3 es correcto (vea el ejercicio 12).

Sección de ejercicios de repaso

1. Proporcione ejemplos de problemas de búsqueda.

2. ¿Qué es búsqueda de texto?

3. Describa en palabras un algoritmo que resuelva el problema de búsqueda de texto.

4. ¿Qué significa ordenar una sucesión?

5. Dé un ejemplo que ilustre por qué es deseable ordenar una sucesión.

6. Describa la inserción por orden en palabras.

7. ¿Qué quiere decir tiempo y espacio requeridos por un algoritmo?

8. ¿Por qué es útil conocer o poder estimar el tiempo y el espacio requeridos por un algoritmo?

9. ¿Por qué a veces es necesario relajar los requerimientos establecidos en la sección 4.1 para un algoritmo?

10. ¿Qué es un algoritmo aleatorizado?

11. ¿Cuáles de los requerimientos de un algoritmo, según se estableció en la sección 4.1, viola un algoritmo aleatorizado?

12. Describa en palabras el algoritmo para desordenar.

13. Dé una aplicación del algoritmo para desordenar.

Ejercicios

1. Haga un seguimiento del algoritmo 4.2.1 para la entrada $t =$ "balalaika" y $p =$ "bala".

2. Haga un seguimiento del algoritmo 4.2.1 para la entrada $t =$ "balalaika" y $p =$ "lai".

3. Haga un seguimiento del algoritmo 4.2.1 para la entrada $t =$ "000000000" y $p =$ "001".

4. Haga el seguimiento del algoritmo 4.2.3 para la entrada

34 20 144 55.

5. Haga el seguimiento del algoritmo 4.2.3 para la entrada

34 20 19 5.

6. Haga el seguimiento del algoritmo 4.2.3 para la entrada

34 55 144 259.

7. Haga el seguimiento del algoritmo 4.2.3 para la entrada

34 34 34 34.

8. Haga el seguimiento del algoritmo 4.2.4 para la entrada

34 57 72 101 135.

Suponga que los valores de *rand* son

$rand(1, 5) = 5$, $rand(2, 5) = 4$,
$rand(3, 5) = 3$, $rand(4, 5) = 5$.

9. Haga el seguimiento del algoritmo 4.2.4 para la entrada

34 57 72 101 135.

Suponga que los valores de *rand* son

$rand(1, 5) = 2$, $rand(2, 5) = 5$,
$rand(3, 5) = 3$, $rand(4, 5) = 4$.

10. Haga el seguimiento del algoritmo 4.2.4 para la entrada

34 57 72 101 135.

Suponga que los valores de *rand* son

$rand(1, 5) = 5$, $rand(2, 5) = 5$,
$rand(3, 5) = 4$, $rand(4, 5) = 4$.

11. Pruebe que el algoritmo 4.2.1 es correcto.

12. Pruebe que el algoritmo 4.2.3 es correcto.

13. Escriba un algoritmo que regrese el índice de la primera ocurrencia del valor *clave* de la sucesión $s_1, \ldots, s_n$. Si la *clave* no está en la sucesión, el algoritmo regresa el valor 0. *Ejemplo:* Si la sucesión es

12 11 12 23

y la *clave* es 12, el algoritmo regresa el valor 1.

14. Escriba un algoritmo que regrese el índice de la última ocurrencia del valor *clave* en la sucesión $s_1, \ldots, s_n$. Si la *clave* no está en la sucesión, el algoritmo regresa el valor 0. *Ejemplo:* Si la sucesión es

12 11 12 23

y la *clave* es 12, el algoritmo regresa el valor 3.

15. Escriba un algoritmo cuya entrada es la sucesión $s_1, \ldots, s_n$ que está en orden no decreciente, y un valor x. (Suponga que todos los valores son números reales). El algoritmo inserta x en la sucesión de manera que la sucesión obtenida está en orden no decreciente. *Ejemplo:* Si la sucesión de entrada es

2 6 12 14

y $x = 5$, la sucesión resultante es

2 5 6 12 14.

16. Modifique el algoritmo 4.2.1 para que encuentre *todas* las ocurrencias de p en t.

17. Describa la entrada del *mejor caso* para el algoritmo 4.2.1.

18. Describa la entrada del *peor caso* para el algoritmo 4.2.1.

19. Modifique el algoritmo 4.2.3 de manera que ordene la sucesión $s_1, \ldots, s_n$ en orden *no creciente*.

20. El algoritmo de *selección por orden* acomoda la sucesión $s_1, \ldots, s_n$ en orden no decreciente, para ello encuentra primero el elemento más pequeño, por ejemplo s_i, y lo coloca en el primer lugar intercambiando s_1 y s_i. Después encuentra el algoritmo más pequeño en $s_2, \ldots, s_n$, de nuevo digamos s_i, y lo coloca en el segundo lugar intercambiando s_2 y s_i. Continúa hasta que la sucesión esté ordenada. Escriba selección por orden en seudocódigo.

21. Haga el seguimiento del algoritmo selección por orden (vea el ejercicio 20) para la entrada de los ejercicios 4 al 7.

22. Muestre que el tiempo para la selección por orden (vea el ejercicio 20) es el mismo que para todas las entradas de tamaño n.

4.3 ➜ Análisis de algoritmos

Un programa de computadora, aun cuando se derive de un algoritmo correcto, puede ser inútil para cierto tipo de entrada ya sea porque el tiempo necesario para correrlo o el espacio requerido para almacenar los datos, las variables del programa, etcétera, son demasiado grandes. El **análisis de un algoritmo** se refiere al proceso de derivar estimaciones del tiempo y el espacio necesarios para ejecutarlo. En esta sección se aborda el problema de estimar el tiempo requerido para ejecutar el algoritmo.

Suponga que se tiene un conjunto X de n elementos, algunos etiquetados con "rojo" y otros etiquetados con "negro", y se quiere encontrar el número de subconjuntos de X que contienen al menos un artículo rojo. Suponga que se construye un algoritmo que examina todos los subconjuntos de X y cuenta los que contienen al menos un elemento rojo y después se implementa este algoritmo como un programa de computadora. Como un conjunto que tiene n elementos tiene 2^n subconjuntos (vea el teorema 2.1.6), el programa requerirá al menos 2^n unidades de tiempo para la ejecución. No importa cuáles sean esas unidades de tiempo, 2^n crece con tanta rapidez cuando n se incrementa (vea la tabla 4.3.1) que, excepto por los valores pequeños de n, sería impráctico correr el programa.

Determinar los parámetros de desempeño de un programa de computadora es una tarea difícil y depende de un número de factores como la computadora que se usa, la mane-

TABLA 4.3.1 ■ Tiempo para ejecutar un algoritmo si un paso toma un microsegundo de ejecución. lg n denota $\log_2 n$ (el logaritmo de n base 2).

Número de pasos hasta terminar para una entrada de tamaño n	*Tiempo de ejecución si n =*				
	3	6	9	12	
1	10^{-6} seg	10^{-6} seg	10^{-6} seg	10^{-6} seg	
lg lg n	10^{-6} seg	10^{-6} seg	2×10^{-6} seg	2×10^{-6} seg	
lg n	2×10^{-6} seg	3×10^{-6} seg	3×10^{-6} seg	4×10^{-6} seg	
n	3×10^{-6} seg	6×10^{-6} seg	9×10^{-6} seg	10^{-5} seg	
n lg n	5×10^{-6} seg	2×10^{-5} seg	3×10^{-5} seg	4×10^{-5} seg	
n^2	9×10^{-6} seg	4×10^{-5} seg	8×10^{-5} seg	10^{-4} seg	
n^3	3×10^{-5} seg	2×10^{-4} seg	7×10^{-4} seg	2×10^{-3} seg	
2^n	8×10^{-6} seg	6×10^{-5} seg	5×10^{-4} seg	4×10^{-3} seg	
	50	100	1000	10^5	10^6
1	10^{-6} seg	10^{-6} seg	10^{-6} seg	10^{-6} seg	10^{-6} seg
lg lg n	2×10^{-6} seg	3×10^{-6} seg	3×10^{-6} seg	4×10^{-6} seg	4×10^{-6} seg
lg n	6×10^{-6} seg	7×10^{-6} seg	10^{-5} seg	2×10^{-5} seg	2×10^{-5} seg
n	5×10^{-5} seg	10^{-4} seg	10^{-3} seg	0.1 seg	1 seg
n lg n	3×10^{-4} seg	7×10^{-4} seg	10^{-2} seg	2 seg	20 seg
n^2	3×10^{-3} seg	0.01 seg	1 seg	3 hr	12 días
n^3	0.13 seg	1 seg	16.7 min	32 años	31,710 años
2^n	36 años	4×10^{16} años	3×10^{287} años	3×10^{30089} años	3×10^{301016} años

ra en que se representan los datos y cómo se traduce el programa en instrucciones de máquina. Aunque las estimaciones precisas del tiempo de ejecución de un programa deben tener en cuenta esos factores, es posible obtener información útil si se analiza el tiempo del algoritmo subyacente.

El tiempo necesario para ejecutar un algoritmo es una función de la entrada. Por lo general, es difícil obtener una fórmula explícita para esta función, y nos conformamos con menos. En lugar de manejar directamente los datos de entrada, se usan parámetros que caracterizan el *tamaño* de la entrada. Por ejemplo, si la entrada es un conjunto que contiene n elementos, se diría que el tamaño de la entrada es n. Tal vez se quiera conocer el tiempo mínimo necesario para ejecutar el algoritmo entre todas las entradas de tamaño n. Este tiempo se llama **tiempo del mejor caso** para entradas de tamaño n. También puede pedirse el tiempo máximo necesario para ejecutar el algoritmo entre todas las entradas de tamaño n. Este tiempo se llama **tiempo del peor caso** para entradas de tamaño n. Otro caso importante es el **tiempo del caso promedio**, es decir, el tiempo necesario para ejecutar el algoritmo para un conjunto finito de entradas todas de tamaño n.

Como la preocupación principal se refiere a la *estimación* del tiempo de un algoritmo en lugar del cálculo del tiempo exacto, siempre que se cuenten algunos pasos fundamentales, dominantes del algoritmo, se obtendrán medidas útiles del tiempo. Por ejemplo, si la actividad principal de un algoritmo es hacer comparaciones, como ocurre en una rutina para ordenar, se cuenta el número de comparaciones. Como otro ejemplo, si un algoritmo consiste en un solo ciclo que se ejecuta cuando mucho en C pasos, para alguna constante C, se cuenta el número de iteraciones del ciclo.

Ejemplo 4.3.1 ▶

Una definición razonable del tamaño de la entrada para algoritmo 4.1.2 que encuentra el valor mayor en una sucesión finita es el número de elementos en la sucesión de entrada. Una definición razonable del tiempo de ejecución es el número de iteraciones del ciclo "while". Con estas definiciones, los tiempos del *mejor caso*, el *peor caso* y el *caso promedio* para el algoritmo 4.1.2 para entradas de tamaño n son $n - 1$ cada uno, ya que el ciclo siempre se ejecuta $n - 1$ veces. ◀

Es común que nos interese menos el tiempo exacto, en el mejor y peor caso, requerido para ejecutar un algoritmo que la manera en que aumenta el tiempo del mejor y peor caso cuando se incrementa el tamaño de la entrada. Por ejemplo, suponga que en el peor caso el tiempo para ejecutar un algoritmo es

$$t(n) = 60n^2 + 5n + 1$$

para una entrada de tamaño n. Para n grande, el término $60n^2$ es aproximadamente igual que $t(n)$ (vea la tabla 4.3.2). En este sentido, $t(n)$ crece como lo hace $60n^2$.

Si $t(n)$ mide el tiempo en el peor caso para una entrada de tamaño n en segundos, entonces

$$T(n) = n^2 + \frac{5}{60}n + \frac{1}{60}$$

mide el tiempo en el peor caso para la entrada de tamaño n en minutos. Ahora, este cambio de unidades no afecta cómo aumenta el tiempo en el peor caso cuando crece el tamaño de la entrada, sino sólo las unidades en que se mide ese tiempo para una entrada de tamaño n. Entonces, cuando se describe el incremento en el tiempo del mejor o el peor caso conforme aumenta el tamaño de la entrada, no sólo se busca el término dominante [por ejemplo, $60n^2$ en la fórmula de $t(n)$], sino también se pueden ignorar los coeficientes constantes. Con esas suposiciones, $t(n)$ crece como lo hace n^2 cuando n se incrementa. Se dice que $t(n)$ es del **orden** n^2 y se escribe

$$t(n) = \Theta(n^2),$$

TABLA 4.3.2 ■ Comparación del crecimiento de $t(n)$ con $60n^2$.

n	$t(n) = 60n^2 + 5n + 1$	$60n^2$
10	6051	6000
100	600,501	600,000
1000	60,005,001	60,000,000
10,000	6,000,050,001	6,000,000,000

que se lee "$t(n)$ es theta de n^2". La idea básica es sustituir una expresión, como $t(n) = 60n^2 + 5n + 1$, con una expresión más sencilla, como n^2, que crece a la misma tasa que $t(n)$. Las definiciones formales se exponen a continuación.

Definición 4.3.2 ▶

Sean f y g dos funciones con dominio $\{1, 2, 3, \ldots\}$.

Se escribe

$$f(n) = O(g(n))$$

y se dice que $f(n)$ *es del orden a lo más* $g(n)$, o $f(n)$ *es O mayúscula de* $g(n)$ si existe una constante positiva C_1 tal que

$$|f(n)| \leq C_1|g(n)|$$

WWW para todos los enteros positivos n, excepto un número finito de ellos.

Se escribe

$$f(n) = \Omega(g(n))$$

y se dice que $f(n)$ *es del orden al menos* $g(n)$ o $f(n)$ *es omega de* $g(n)$ si existe una constante positiva C_2 tal que

$$|f(n)| \geq C_2\,|g(n)|$$

para todos los enteros positivos n, excepto un número finito de ellos.

Se escribe

$$f(n) = \Theta(g(n))$$

y se dice que $f(n)$ *es del orden* $g(n)$ o $f(n)$ *es theta de* $g(n)$ si $f(n) = O(g(n))$ y $f(n) = \Omega(g(n))$. ◀

La definición 4.3.2 se puede parafrasear como sigue: $f(n) = O(g(n))$ si, excepto por un factor constante y un número finito de excepciones, f está acotada por arriba por g. También se dice que g es una **cota superior asintótica** para f. De manera similar, $f(n) = \Omega(g(n))$ si, excepto por un factor constante y un número finito de excepciones, f está acotada por abajo por g. También se dice que g es una **cota inferior asintótica** para f. Además, $f(n) = \Theta(g(n))$ si, excepto por factores constantes y un número finito de excepciones, f está acotada arriba y abajo por g. También se dice que g es una **cota estrecha asintótica** de f.

Según la definición 4.3.2, si $f(n) = O(g(n))$, todo lo que se concluye es que, excepto por un factor constante y un número finito de excepciones, f está acotada *arriba* por g, de manera que g crece al menos tan rápido como f. Por ejemplo, si $f(n) = n$ y $g(n) = 2^n$, entonces $f(n) = O(g(n))$, pero g crece mucho más rápido que f. La afirmación $f(n) = O(g(n))$ no habla de una cota *inferior* para f. Por otro lado, si $f(n) = \Theta(g(n))$, se concluye que excepto por factores constantes y un número finito de excepciones, f está acotada *arriba* y *abajo* por g, por lo tanto, f y g crecen a la misma velocidad. Observe que $n = O(2^n)$, pero $n \neq \Theta(2^n)$.

Ejemplo 4.3.3 ▶

Dado que

$$60n^2 + 5n + 1 \leq 60n^2 + 5n^2 + n^2 = 66n^2 \qquad \text{para toda } n \geq 1,$$

se puede tomar $C_1 = 66$ en la definición 4.3.2 para obtener

$$60n^2 + 5n + 1 = O(n^2).$$

Como

$$60n^2 + 5n + 1 \geq 60n^2 \qquad \text{para toda } n \geq 1,$$

se puede tomar $C_2 = 60$ en la definición 4.3.2 para obtener

$$60n^2 + 5n + 1 = \Omega(n^2).$$

Como $60n^2 + 5n + 1 = O(n^2)$ y $60n^2 + 5n + 1 = \Omega(n^2)$,

$$60n^2 + 5n + 1 = \Theta(n^2). \quad \blacktriangleleft$$

El método del ejemplo 4.3.3 resulta útil para demostrar que un polinomio en n de grado k con coeficientes no negativos es $\Theta(n^k)$. [De hecho, *cualquier* polinomio en n de grado k es $\Theta(n^k)$, aun cuando algunos de sus coeficientes sean negativos. Para probar este resultado más general, debe modificarse el método del ejemplo 4.3.3].

Teorema 4.3.4

Sea

$$p(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$$

un polinomio en n de grado k, donde cada a_i es no negativa. Entonces

$$p(n) = \Theta(n^k).$$

Demostración Primero se demuestra que $p(n) = O(n^k)$. Sea

$$C_1 = a_k + a_{k-1} + \cdots + a_1 + a_0.$$

Entonces, para toda n,

$$\begin{aligned} p(n) &= a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0 \\ &\quad a_k n^k + a_{k-1} n^k + \cdots + a_1 n^k + a_0 n^k \\ &= (a_k + a_{k-1} + \cdots + a_1 + a_0) n^k = C_1 n^k. \end{aligned}$$

Por lo tanto, $p(n) = O(n^k)$.

Después, se demuestra que $p(n) = \Omega(n^k)$. Para toda n,

$$p(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0 \geq a_k n^k = C_2 n^k,$$

donde $C_2 = a_k$. Por lo tanto, $p(n) = \Omega(n^k)$.

Como $p(n) = O(n^k)$ y $p(n) = \Omega(n^k)$, $p(n) = \Theta(n^k)$.

Ejemplo 4.3.5 ▶

En este libro, $\lg n$ denota $\log_2 n$ (el logaritmo de n base 2). Como $\lg n < n$ para toda $n \geq 1$ (vea la figura 4.3.1),

$$2n + 3\lg n < 2n + 3n = 5n \qquad \text{para toda } n \geq 1.$$

Así,

$$2n + 3\lg n = O(n).$$

Además,

$$2n + 3\lg n \geq 2n \qquad \text{para toda } n \geq 1.$$

Entonces,

$$2n + 3\lg n = \Omega(n).$$

Por lo tanto,

$$2n + 3\lg n = \Theta(n).$$

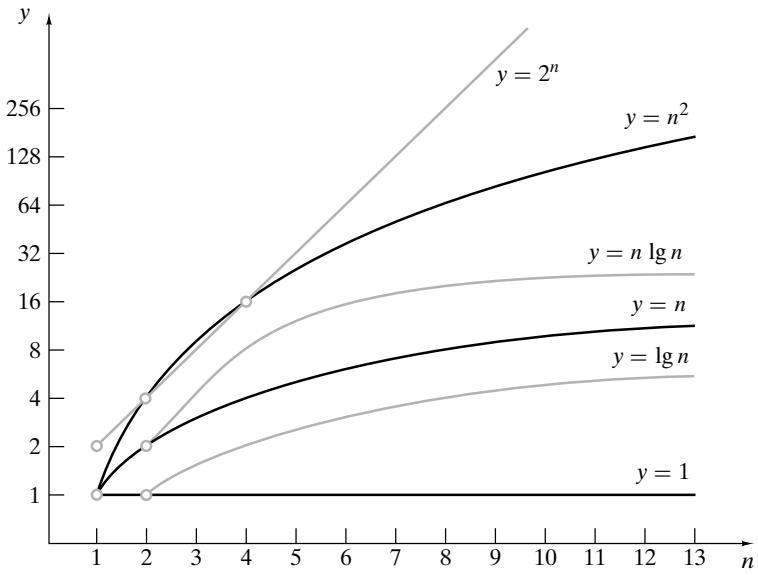


Figura 4.3.1 Crecimiento de algunas funciones comunes. ◀

Ejemplo 4.3.6 ▶

Si $a > 1$ y $b > 1$ (para asegurar que $\log_b a > 0$), por la fórmula del cambio de base para logaritmos [teorema B.37(e)],

$$\log_b n = \log_b a \log_a n \qquad \text{para toda } n \geq 1.$$

Por lo tanto,

$$\log_b n \leq C \log_a n \qquad \text{para toda } n \geq 1,$$

donde $C = \log_b a$. Así, $\log_b n = O(\log_a n)$.

Además,

$$\log_b n \geq C \log_a n \qquad \text{para toda } n \geq 1;$$

de manera que $\log_b n = \Omega(\log_a n)$. Como $\log_b n = O(\log_a n)$ y $\log_b n = \Omega(\log_a n)$, se concluye que $\log_b n = \Theta(\log_a n)$.

Como $\log_b n = \Theta(\log_a n)$, cuando se usa la notación asintótica no debemos preocuparnos por el número que se usa como la base de la función logaritmo (siempre que la base sea mayor que 1). Por esta razón, algunas veces se escribe simplemente log sin especificar la base. ◀

Ejemplo 4.3.7 ▶

Si se sustituye cada entero $1, 2, \ldots, n$ por n en la suma $1 + 2 + \ldots + n$, la suma no disminuye y se tiene

$$1 + 2 + \cdots + n \leq n + n + \cdots + n = n \cdot n = n^2 \qquad \text{para toda } n \geq 1 \qquad \textbf{(4.3.1)}$$

Entonces

$$1 + 2 + \cdots + n = O(n^2).$$

Para obtener una cota inferior, se puede imitar el argumento anterior y sustituir cada entero $1, 2, \ldots, n$ por 1 en la suma $1 + 2 + \ldots + n$ para obtener

$$1 + 2 + \cdots + n \geq 1 + 1 + \cdots + 1 = n \qquad \text{para toda } n \geq 1.$$

En este caso se concluye que

$$1 + 2 + \cdots + n = \Omega(n),$$

y mientras que la expresión anterior es verdadera, no se puede deducir una estimación Θ para $1 + 2 + \ldots + n$, ya que la cota superior n^2 y la cota inferior n no son iguales. Se requerirá más creatividad para derivar una cota inferior.

Una manera de obtener una cota inferior más precisa es usar un argumento como el del párrafo anterior, pero primero eliminar la primera mitad de los términos, para obtener

$$\begin{aligned} 1 + 2 + \cdots + n &\geq \lceil n/2 \rceil + \cdots + (n-1) + n \\ &\geq \lceil n/2 \rceil + \cdots + \lceil n/2 \rceil + \lceil n/2 \rceil \\ &= \lceil (n+1)/2 \rceil \lceil n/2 \rceil \geq (n/2)(n/2) = \frac{n^2}{4} \end{aligned} \qquad \textbf{(4.3.2)}$$

para toda $n \geq 1$. Ahora se puede concluir que

$$1 + 2 + \cdots + n = \Omega(n^2).$$

Por lo tanto,

$$1 + 2 + \cdots + n = \Theta(n^2). \quad \blacktriangleleft$$

Ejemplo 4.3.8 ▶

Si k es un entero positivo y, como en el ejemplo 4.3.7, se sustituye cada entero 1, 2, . . . , n por n, se tiene

$$1^k + 2^k + \cdots + n^k \leq n^k + n^k + \cdots + n^k = n \cdot n^k = n^{k+1}$$

para toda $n \geq 1$; entonces

$$1^k + 2^k + \cdots + n^k = O(n^{k+1}).$$

Se puede obtener también una cota inferior como en el ejemplo 4.3.7:

$$\begin{aligned} 1^k + 2^k + \cdots + n^k &\geq \lceil n/2 \rceil^k + \cdots + (n-1)^k + n^k \\ &\geq \lceil n/2 \rceil^k + \cdots + \lceil n/2 \rceil^k + \lceil n/2 \rceil^k \\ &= \lceil (n+1)/2 \rceil \lceil n/2 \rceil^k \geq (n/2)(n/2)^k = n^{k+1}/2^{k+1} \end{aligned}$$

para toda $n \geq 1$. Se concluye que

$$1^k + 2^k + \cdots + n^k = \Omega(n^{k+1}),$$

y, por lo tanto,

$$1^k + 2^k + \cdots + n^k = \Theta(n^{k+1}). \quad \blacktriangleleft$$

Observe la diferencia entre el polinomio

$$a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$$

en el teorema 4.3.4 y la expresión

$$1^k + 2^k + \cdots + n^k$$

en el ejemplo 4.3.8. Un polinomio tiene un número fijo de términos, mientras que el número de términos en la expresión del ejemplo 4.3.8 depende del valor de n. Más aún, el polinomio del teorema 4.3.4 es $\Theta(n^k)$, pero la expresión en el ejemplo 4.3.8 es $\Theta(n^{k+1})$.

El siguiente ejemplo da una notación theta para $\lg n!$.

Ejemplo 4.3.9 ▶

Usando un argumento similar al del ejemplo 4.3.7, se demuestra que

$$\lg n! = \Theta(n \lg n).$$

Por las propiedades de los logaritmos, se tiene

$$\lg n! = \lg n + \lg(n-1) + \cdots + \lg 2 + \lg 1$$

para toda $n \geq 1$. Como lg es una función creciente,

$$\lg n + \lg(n-1) + \cdots + \lg 2 + \lg 1 \leq \lg n + \lg n + \cdots + \lg n + \lg n = n \lg n$$

para toda $n \geq 1$. Se concluye que

$$\lg n! = O(n \lg n).$$

Para toda $n \geq 4$, se tiene

$$\begin{aligned}
\lg n + \lg(n-1) + \cdots + \lg 2 + \lg 1 &\geq \lg n + \lg(n-1) + \cdots + \lg\lceil n/2 \rceil \\
&\geq \lg\lceil n/2 \rceil + \cdots + \lg\lceil n/2 \rceil \\
&= \lceil (n+1)/2 \rceil \lg\lceil n/2 \rceil \\
&\geq (n/2) \lg(n/2) \\
&= (n/2)[\lg n - \lg 2] \\
&= (n/2)[(\lg n)/2 + ((\lg n)/2 - 1)] \\
&\geq (n/2)(\lg n)/2 \\
&= n \lg n/4
\end{aligned}$$

[ya que $(\lg n)/2 \geq 1$ para toda $n \geq 4$]. Por lo tanto,

$$\lg n! = \Omega(n \lg n).$$

Se deduce que

$$\lg n! = \Theta(n \lg n).$$ ◀

Ejemplo 4.3.10 ▶

Demuestre que si $f(n) = \Theta(g(n))$ y $g(n) = \Theta(h(n))$, entonces $f(n) = \Theta(h(n))$.

Como $f(n) = \Theta(g(n))$ <, existen constantes C_1 y C_2 tales que

$$C_1|g(n)| \leq |f(n)| \leq C_2|g(n)|$$

para todos los enteros positivos n excepto un número finito de ellos. Dado que $g(n) = \Theta(h(n))$, existen constantes C_3 y C_4 tales que

$$C_3|h(n)| \leq |g(n)| \leq C_4|h(n)|$$

para todos los enteros positivos n excepto un número finito de ellos. Por lo tanto,

$$C_1C_3|h(n)| \leq C_1|g(n)| \leq |f(n)| \leq C_2|g(n)| \leq C_2C_4|h(n)|$$

para todos los enteros positivos n. Se deduce que $f(n) = \Theta(h(n))$, ◀

Ahora se define a qué se refiere que en el peor caso, el mejor caso o el caso promedio el tiempo de un algoritmo sea del orden a lo más $g(n)$.

Definición 4.3.11 ▶

Si un algoritmo requiere $t(n)$ unidades de tiempo en el mejor caso para una entrada de tamaño n y

$$t(n) = O(g(n)),$$

se dice que el *tiempo en el mejor caso requerido por el algoritmo es de orden a lo más* $g(n)$ o que el *tiempo en el mejor caso requerido por el algoritmo* es $O(g(n))$.

Si un algoritmo requiere $t(n)$ unidades de tiempo para terminar en el peor caso para una entrada de tamaño n y

$$t(n) = O(g(n)),$$

se dice que el *tiempo en el peor caso requerido por el algoritmo es de orden a lo más* $g(n)$ o que el *tiempo en el peor caso requerido por el algoritmo* es $O(g(n))$.

Si un algoritmo requiere $t(n)$ unidades de tiempo para terminar en el caso promedio para una entrada de tamaño n y

$$t(n) = O(g(n)),$$

se dice que el *tiempo en el caso promedio requerido por el algoritmo es de orden a lo más* $g(n)$ o que el *tiempo en el caso promedio requerido por el algoritmo* es $O(g(n))$. ◀

Al reemplazar O por Ω y "a lo más" por "al menos" en la definición 4.3.11, se obtiene la definición de qué quiere decir que en el mejor caso, el peor caso o el caso promedio el tiempo de un algoritmo sea del orden de al menos $g(n)$. Si el tiempo requerido por el algoritmo en el mejor caso es $O(g(n))$ y $\Omega(g(n))$, se dice que el tiempo requerido por el algoritmo en el mejor caso es $\Theta(g(n))$. Una definición análoga se aplica al tiempo de un algoritmo en el peor caso y el caso promedio.

Ejemplo 4.3.12 ▶

Suponga que se sabe que un algoritmo toma

$$60n^2 + 5n + 1$$

unidades de tiempo para terminar en el peor caso para entradas de tamaño n. Se demostró en el ejemplo 4.3.3 que

$$60n^2 + 5n + 1 = \Theta(n^2).$$

Así, el tiempo en el peor caso requerido por este algoritmo es $\Theta(n^2)$. ◀

Ejemplo 4.3.13 ▶

Encuentre una notación theta en términos de n para el número de veces que la expresión $x = x + 1$ se ejecuta.

```
1. for i = 1 to n
2.     for j = 1 to i
3.         x = x + 1
```

Primero, i se hace igual a 1, al correr j de 1 a 1, la línea 3 se ejecuta una vez. Después, i se hace igual a 2 y cuando j corre de 1 a 2, la línea 3 se ejecuta 2 veces, y así sucesivamente. Entonces el número total de veces que se ejecuta la línea 3 (vea el ejemplo 4.3.7) es

$$1 + 2 + \cdots + n = \Theta(n^2).$$

Así, una notación theta para el número de veces que se ejecuta la expresión $x = x + 1$ es $\Theta(n^2)$. ◀

Ejemplo 4.3.14 ▶

Encuentre una notación theta en términos de n para el número de veces que se ejecuta la expresión $x = x + 1$:

```
1.  i = n
2.  while (i ≥ 1) {
3.      x = x + 1
4.      i = ⌊i/2⌋
5.  }
```

Primero, se examinan algunos casos específicos. A causa de la función piso, los cálculos se simplifican si n es una potencia de 2. Considere, por ejemplo, el caso $n = 8$. En la línea 1, i se hace igual a 8. En la línea 2, la condición $i \geq 1$ es verdadera. En la línea 3, se ejecuta la expresión $x = x + 1$ la primera vez. En la línea 4, i se hace igual a 4 y se regresa a la línea 2.

En la línea 2, la condición $i \geq 1$ de nuevo es verdadera. En la línea 3 se ejecuta la expresión $x = x + 1$ la segunda vez. En la línea 4, i se hace igual a 2 y se regresa a la línea 2.

En la línea 2, la condición $i \geq 1$ es verdadera de nuevo. En la línea 3 se ejecuta la expresión $x = x + 1$ la tercera vez. En la línea 4 i se hace igual a 1 y se regresa a la línea 2.

En la línea 2, la condición $i \geq 1$ es cierta otra vez. En la línea 3 se ejecuta la expresión $x = x + 1$ la cuarta vez. En la línea 4, i se hace igual a 0 y se regresa a la línea 2.

Esta vez en la línea 2, la condición $i \geq 1$ es falsa. La expresión $x = x + 1$ se ejecutó cuatro veces.

Ahora suponga que n es 16. En la línea 1, i es igual a 16. En la línea 2, la condición $i \geq 1$ es verdadera. En la línea 3, se ejecuta $x = x + 1$ la primera vez. En la línea 4, i queda igual a 8 y se regresa a la línea 2. Ahora la ejecución procede como antes; la expresión $x = x + 1$ se ejecuta cuatro veces más, para un total de cinco veces.

De manera similar, si $n = 32$, la expresión $x = x + 1$ se ejecuta un total de seis veces.

Comienza a surgir un patrón. Cada vez que se duplica el valor inicial, la expresión $x = x + 1$ se ejecuta una vez más. De manera más precisa, si $n = 2^k$, la expresión $x = x + 1$ se ejecuta $k + 1$ veces. Como k es el exponente de 2, $k = \lg n$. Entonces si $n = 2^k$, la expresión $x = x + 1$ se ejecuta $1 + \lg n$ veces.

Si n es un entero positivo arbitrario (no necesariamente una potencia de 2), está entre dos potencias de 2, es decir, para alguna $k \geq 1$,

$$2^{k-1} \leq n < 2^k.$$

Se usará inducción sobre k para demostrar que en este caso, la expresión $x = x + 1$ se ejecuta k veces.

Si $k = 1$, se tiene

$$1 = 2^{1-1} \leq n < 2^1 = 2.$$

Por lo tanto, n es 1. En este caso, la expresión $x = x + 1$ se ejecuta una vez. Con esto queda probado el paso base.

Ahora suponga que si n satisface

$$2^{k-1} \leq n < 2^k,$$

la expresión $x = x + 1$ se ejecuta k veces. Debemos demostrar que si n satisface

$$2^k \leq n < 2^{k+1},$$

la expresión $x = x + 1$ se ejecuta $k + 1$ veces.

Suponga que n satisface

$$2^k \leq n < 2^{k+1}.$$

En la línea 1, i se hace igual a n. En la línea 2, la condición $i \geq 1$ es verdadera. En la línea 3, se ejecuta $x = x + 1$ la primera vez. En la línea 4, i se hace igual a $\lfloor n/2 \rfloor$ y se regresa a la línea 2. Observe que

$$2^{k-1} \leq n/2 < 2^k.$$

Como 2^{k-1} es un entero, también debe tenerse

$$2^{k-1} \leq \lfloor n/2 \rfloor < 2^k.$$

Por la suposición de inducción, la expresión $x = x + 1$ se ejecuta k veces más, para llegar a un total de $k + 1$ veces. El paso inductivo queda completo; por lo tanto, si n satisface

$$2^{k-1} \leq n < 2^k,$$

la expresión $x = x + 1$ se ejecuta k veces.

Suponga que n satisface

$$2^{k-1} \leq n < 2^k.$$

Tomando logaritmos base 2, se tiene

$$k - 1 \leq \lg n < k.$$

Por lo tanto, k, el número de veces que la expresión $x = x + 1$ se ejecuta, satisface

$$\lg n < k \leq 1 + \lg n.$$

Como k es un entero, debe tenerse

$$k \leq 1 + \lfloor \lg n \rfloor.$$

Más aún,

$$\lfloor \lg n \rfloor < k.$$

De las dos últimas desigualdades se tiene que

$$k = 1 + \lfloor \lg n \rfloor.$$

Como

$$1 + \lfloor \lg n \rfloor = \Theta(\lg n),$$

una notación theta para el número de veces que la expresión $x = x + 1$ se ejecuta es $\Theta(\lg n)$.

◀

Muchos algoritmos se basan en la idea de dividir a la mitad repetidas veces. El ejemplo 4.3.14 muestra que para un tamaño n, dividir a la mitad repetidas veces toma el tiempo $\Theta(\lg n)$. Por supuesto, el algoritmo puede hacer el trabajo además de la división a la mitad que aumentará el tiempo global.

Ejemplo 4.3.15 ▶

Encuentre una notación theta en términos de n para el número de veces que se ejecuta la expresión $x = x + 1$.

```
1.  j = n
2.  while (j ≥ 1) {
3.      for i = 1 to j
4.          x = x + 1
5.      j = ⌊j/2⌋
6.  }
```

Sea $t(n)$ el número de veces que se ejecuta $x = x + 1$. La primera vez que se llega al cuerpo del ciclo "while", la expresión $x = x + 1$ se ejecuta n veces. Por lo tanto, $t(n) \geq n$ para toda $n \geq 1$ y $t(n) = \Omega(n)$.

Ahora se deriva una notación O mayúscula para $t(n)$. Después de hacer j igual a n, se llega al ciclo "while" por primera vez. La expresión $x = x + 1$ se ejecuta n veces. En la línea 5, j se sustituye por $\lfloor n/2 \rfloor$; así, $j \leq n/2$. Si $j \geq 1$, se ejecuta $x = x + 1$ a lo más $n/2$ veces más en la siguiente iteración del ciclo "while", y así sucesivamente. Si k denota el número de veces que se ejecuta el cuerpo del ciclo "while", el número de veces que se ejecuta $x = x + 1$ es a lo sumo

$$n + \frac{n}{2} + \frac{n}{4} + \cdots + \frac{n}{2^{k-1}}.$$

Esta suma geométrica (vea el ejemplo 1.7.4) es igual a

$$\frac{n\left(1 - \frac{1}{2^k}\right)}{1 - \frac{1}{2}}.$$

Ahora

$$t(n) \leq \frac{n\left(1 - \frac{1}{2^k}\right)}{1 - \frac{1}{2}} = 2n\left(1 - \frac{1}{2^k}\right) \leq 2n \quad \text{para toda } n \geq 1,$$

de modo que $t(n) = O(n)$. Entonces la notación theta para el número de veces que se ejecuta $x = x + 1$ es $\Theta(n)$. ◀

Ejemplo 4.3.16 ▶

Determine, en la notación theta, el número de veces en los tiempos del mejor caso, el peor caso y el caso promedio que se requieren para ejecutar el algoritmo 4.3.17, que se da enseguida. Suponga que la entrada es de tamaño n y que el tiempo de corrida del algoritmo es el número de comparaciones hechas en la línea 3. Además suponga que las $n + 1$ posibilidades de que la *clave* esté en una posición específica en la sucesión o no esté en la sucesión tienen la misma probabilidad.

El tiempo para el mejor caso se analiza como sigue. Si $s_1 = clave$, la línea 3 se ejecuta una vez. Entonces, el tiempo del mejor caso para el algoritmo 4.3.17 es

$$\Theta(1).$$

El tiempo del peor caso para el algoritmo 4.3.17 se analiza como sigue. Si la *clave* no está en la sucesión, la línea 3 se ejecuta n veces, de modo que el tiempo del peor caso de algoritmo 4.2.17 es

$$\Theta(n).$$

Por último, considere el tiempo del caso promedio para el algoritmo 4.3.17. Si la *clave* se encuentra en la posición *i-ésima*, la línea 3 se ejecuta i veces; si la *clave* no está en la sucesión, la línea 3 se ejecuta n veces. Entonces, el número promedio de veces que se ejecuta la línea 3 es

$$\frac{(1 + 2 + \cdots + n) + n}{n + 1}.$$

Ahora bien

$$\begin{aligned} \frac{(1 + 2 + \cdots + n) + n}{n + 1} &\leq \frac{n^2 + n}{n + 1} && \text{por (4.3.1)} \\ &= \frac{n(n + 1)}{n + 1} = n. \end{aligned}$$

Por lo tanto, el tiempo del caso promedio para el algoritmo 4.3.17 es

$$O\,(n)$$

Además,

$$\begin{aligned} \frac{(1 + 2 + \cdots + n) + n}{n + 1} &\geq \frac{n^2/4 + n}{n + 1} && \text{por (4.3.2)} \\ &\geq \frac{n^2/4 + n/4}{n + 1} = \frac{n}{4}. \end{aligned}$$

Por tanto, el tiempo del caso promedio para el algoritmo 4.3.17 es

$$\Omega(n).$$

Así que el tiempo del caso promedio para el algoritmo 4.3.17 es

$$\Theta\,(n).$$

Para este algoritmo, los tiempos del peor caso y el caso promedio son ambos $\Theta(n)$. ◀

Algoritmo 4.3.17

Búsqueda de una sucesión no ordenada

Dada la sucesión $s_1, \ldots, s_n$ y el valor *clave*, este algoritmo regresa el índice de *clave*. Si la *clave* no se encuentra, el algoritmo regresa 0.

Entrada: $s_1, s_2, \ldots, s_n$, n, y *clave* (el valor que se busca)
Salida: El índice de *clave*, o si la *clave* no se encuentra, 0.

```
1.  búsqueda_lineal(s, n, clave){
2.      for i = 1 to n
3.          if(clave == s_i)
4.              return i // búsqueda existosa
5.      return 0 // búsqueda no existosa
6.  }
```

TABLA 4.3.3 ■ Funciones de crecimiento comunes.

Forma theta	*Nombre*
$\Theta(1)$	Constante
$\Theta(\lg \lg n)$	Log log
$\Theta(\lg n)$	Log
$\Theta(n)$	Lineal
$\Theta(n \lg n)$	n log n
$\Theta(n^2)$	Cuadrática
$\Theta(n^3)$	Cúbica
$\Theta(n^k), k \geq 1$	Polinomial
$\Theta(c^n), c > 1$	Exponencial
$\Theta(n!)$	Factorial

Las constantes que se suprimen en la notación theta pueden ser muy importantes. Aun si para cualquier entrada de tamaño n, el algoritmo A requiere exactamente $C_1 n$ unidades de tiempo y el algoritmo B requiere justo $C_2 n^2$ unidades de tiempo, para ciertos tamaños de entradas el algoritmo B puede ser superior. Por ejemplo, suponga que para cualquier tamaño de entrada n, el algoritmo A requiere $300n$ unidades de tiempo y el algoritmo B requiere $5n^2$ unidades de tiempo. Para un tamaño de entrada $n = 5$, el algoritmo A requiere 1500 unidades de tiempo y el B 125 unidades, por lo que el algoritmo B es más rápido. Por supuesto, para entradas suficientemente grandes, el algoritmo A es mucho más rápido que el algoritmo B.

Ciertas funciones de crecimiento ocurren con tanta frecuencia que tienen nombres especiales, como se observa en la tabla 4.3.3. Las funciones en la tabla 4.3.3, con la excepción de $\Theta(n^k)$, se arreglan de manera que si $\Theta(f(n))$ está arriba de $\Theta(g(n))$, entonces $f(n) \leq g(n)$ para todos los enteros negativos excepto un número finito de ellos. Entonces, si los algoritmos A y B tienen tiempos de corrida $\Theta(f(n))$ y $\Theta(g(n))$, respectivamente, y $\Theta(f(n))$ está arriba de $\Theta(g(n))$ en la tabla 4.3.3, entonces el algoritmo A es más eficiente en tiempo que el algoritmo B para entradas suficientemente grandes.

Es importante desarrollar cierta sensibilidad para los tamaños relativos de las funciones en la tabla 4.3.3. En la figura 4.3.1 se graficaron algunas de estas funciones. Otra manera de desarrollar alguna apreciación para los tamaños relativos de las funciones $f(n)$ en la tabla 4.3.3 es determinar cuánto tiempo tomaría al algoritmo terminar, tiempo que es exactamente $f(n)$. Para este propósito, suponga que se tiene una computadora que puede ejecutar un paso en un microsegundo (10^{-6} segundos). La tabla 4.3.1 señala los tiempos de ejecución, bajo esta suposición, para varios tamaños de entradas. Observe que es práctico implementar un algoritmo que requiere 2^n pasos cuando la entrada es de tamaño n, sólo para tamaños muy pequeños de entradas. Los algoritmos que requieren n^2 o n^3 pasos también se vuelven poco prácticos para tamaños relativamente grandes de entradas. Observe además las mejoras drásticas que se obtienen al moverse de n^2 pasos a $n \lg n$ pasos.

Un problema cuyo algoritmo tiene un tiempo polinomial del peor caso se considera que posee un "buen" algoritmo; la interpretación es que ese problema tiene una solución eficiente. Estos problemas se llaman **factibles** o **tratables**. Por supuesto, si el tiempo del peor caso para resolver el problema es proporcional a un polinomio de alto grado, tomará un tiempo largo para resolverse. Por fortuna, en muchos casos importantes, la cota polinomial tiene un grado pequeño.

Se dice que un problema que no tiene un algoritmo con tiempo polinomial del peor caso es **intratable**. Se garantiza que cualquier algoritmo, si lo hay, que resuelve un problema intratable toma un largo tiempo de ejecución en el peor caso, aun para entradas de tamaño modesto.

Ciertos problemas son tan duros que ni siquiera tienen algoritmos. Se dice que un problema para el que no existe un algoritmo **no tiene solución**. Existe un considerable número de problemas que no tienen solución, y algunos de ellos son de gran importancia práctica. Uno de los primeros problemas que se probó que no tiene solución es el **problema de detención (*halting problem*)**: Dado un programa arbitrario y un conjunto de entradas, ¿se detendrá el programa en algún momento?

Un número grande de problemas solucionables tienen un estatus todavía indeterminado; se piensa que son intratables, pero no se ha demostrado que lo sean. (La mayor parte de ellos pertenecen a la clase de problemas de programación no lineal completos o NP-completos; vea más detalles en [Johnsonbaugh]). Un ejemplo de problema NP-completo es

> Dada una colección $\mathcal{C}$ de conjuntos finitos y un entero positivo $k \leq |\mathcal{C}|$, ¿contiene $\mathcal{C}$ al menos k conjuntos mutuamente ajenos?

Otros problemas NP-completos incluyen el del agente viajero y el problema del ciclo hamiltoniano (vea la sección 8.3).

Sugerencias para resolver problemas

Para derivar un notación O mayúscula para una expresión $f(n)$ directamente, debe encontrarse una constante C_1 y una expresión sencilla $g(n)$ (como n, $n \lg n$, n^2) tal que $\{f(n)| \leq C_1|g(n)|$ para toda n excepto un número finito. Recuerde que está intentando derivar una *desigualdad*, *no* una igualdad, de manera que es posible sustituir términos en $f(n)$ con otros si el resultado es *mayor* (vea el ejemplo 4.3.3).

Para derivar una notación omega para una expresión $f(n)$ directamente, debe encontrarse una constante C_2 y una expresión sencilla $g(n)$ tal que $|f(n)| \geq C_2|g(n)|$ para toda n excepto un número finito. De nuevo, está intentando derivar una *desigualdad* por lo que es posible sustituir términos en $f(n)$ con otros si el resultado es *menor* (otra vez vea el ejemplo 4.3.3).

Para derivar una notación theta, debe derivarse tanto la notación O mayúscula como la omega.

Otra manera de derivar estimaciones O mayúscula, omega y theta es usar resultados conocidos:

Expresión	*Nombre*	*Estimación*	*Referencia*
$a_k n^k + a_{k-1}n^{k-1} + \cdots + a_1 n + a_0$	Polinomio	$\Theta(n^k)$	Teorema 4.3.4
$1 + 2 + \cdots + n$	Suma aritmética (caso k=1 para siguiente elemento)	$\Theta(n^2)$	Ejemplo 4.3.7
$1^k + 2^k + \cdots + n^k$	Suma de potencias	$\Theta(n^{k+1})$	Ejemplo 4.3.8
$\lg n!$	log n factorial	$\Theta(n \lg n)$	Ejemplo 4.3.9

Para derivar una estimación asintótica para el tiempo de un algoritmo, cuente el número de pasos $t(n)$ requeridos por el algoritmo, y después obtenga una estimación para $t(n)$ como se describió. Es común que los algoritmos contengan ciclos; en tal caso, derivar $t(n)$ requiere contar el número de iteraciones de los ciclos.

Sección de ejercicios de repaso

1. ¿A qué se refiere el "análisis del algoritmo"?
2. ¿Qué es el tiempo del peor caso de un algoritmo?
3. ¿Qué es el tiempo del mejor caso de un algoritmo?
4. ¿Qué es el tiempo del caso promedio de un algoritmo?
5. Defina $f(n) = O(g(n))$. ¿Cómo se llama esta notación?
6. Dé una interpretación intuitiva de cómo se relacionan f y g si $f(n) = O(g(n))$.
7. Defina $f(n) = \Omega(g(n))$. ¿Cómo se llama esta notación?
8. Dé una interpretación intuitiva de cómo se relacionan f y g si $f(n) = \Omega(g(n))$.
9. Defina $f(n) = \Theta(g(n))$. ¿Cómo se llama esta notación?
10. Dé una interpretación intuitiva de cómo se relacionan f y g si $f(n) = \Theta(g(n))$.

Ejercicios

Seleccione una notación theta de la tabla 4.3.3 para cada expresión en los ejercicios 1 al 12.

1. $6n + 1$
2. $2n^2 + 1$
3. $6n^3 + 12n^2 + 1$
4. $3n^2 + 2n \lg n$
5. $2 \lg n + 4n + 3n \lg n$
6. $6n^6 + n + 4$
7. $2 + 4 + 6 + \cdots + 2n$
8. $(6n + 1)^2$
9. $(6n + 4)(1 + \lg n)$
10. $\dfrac{(n + 1)(n + 3)}{n + 2}$
11. $\dfrac{(n^2 + \lg n)(n + 1)}{n + n^2}$
12. $2 + 4 + 8 + 16 + \cdots + 2^n$

En los ejercicios 13 al 15, seleccione una notación theta para $f(n) + g(n)$.

13. $f(n) = \Theta(1), \quad g(n) = \Theta(n^2)$
14. $f(n) = 6n^3 + 2n^2 + 4, \quad g(n) = \Theta(n \lg n)$
15. $f(n) = \Theta(n^{3/2}), \quad g(n) = \Theta(n^{5/2})$

En los ejercicios 16 al 25, seleccione una notación theta entre

$$\Theta(1), \quad \Theta(\lg n), \quad \Theta(n), \quad \Theta(n \lg n),$$
$$\Theta(n^2), \quad \Theta(n^3), \quad \Theta(2^n), \quad \text{o} \quad \Theta(n!)$$

para el número de veces que se ejecuta la expresión $x = x + 1$.

16.
```
for i = 1 to 2n
    x = x + 1
```

17.
```
i = 1
while (i ≤ 2n) {
    x = x + 1
    i = i + 2
}
```

18.
```
for i = 1 to n
    for j = 1 to n
        x = x + 1
```

19.
```
for i = 1 to 2n
    for j = 1 to n
        x = x + 1
```

20.
```
for i = 1 to n
    for j = 1 to ⌊i/2⌋
        x = x + 1
```

21.
```
for i = 1 to n
    for j = 1 to n
        for k = 1 to n
            x = x + 1
```

22.
```
for i = 1 to n
    for j = 1 to n
        for k = 1 to i
            x = x + 1
```

23.
```
for i = 1 to n
    for j = 1 to i
        for k = 1 to j
            x = x + 1
```

24.
```
j = n
while (j ≥ 1) {
    for i = 1 to j
        x = x + 1
    j = ⌊j/3⌋
}
```

25.
```
i = n
while (i ≥ 1) {
    for j = 1 to n
        x = x + 1
    i = ⌊i/2⌋
}
```

26. Encuentre un notación theta para el número de veces que se ejecuta la expresión $x = x + 1$.

```
i = 2
while (i < n) {
   i = i^2
   x = x + 1
}
```

27. Sea $t(n)$ el número total de veces que se incrementa i y se disminuye j en el siguiente seudocódigo, donde $a_1, a_2, \ldots$ es una sucesión de números reales.

```
i = 1
j = n
while (i < j) {
   while (i < j ∧ a_i < 0)
      i = i + 1
   while (i < j ∧ a_j ≥ 0)
      j = j − 1
   if (i < j)
      swap(a_i, a_j)
}
```

Encuentre una notación theta para $t(n)$.

28. Encuentre una notación theta para el tiempo requerido en el peor caso por el siguiente algoritmo:

```
esclave(s, n, clave){
      for i = 1 to n − 1
         for j = i + 1 to n
            if (s_i + s_j == key)
               return 1
            else
               return 0
}
```

29. Además de encontrar una notación theta en los ejercicios 1 al 28, pruebe que ésta es correcta.

30. Encuentre el número exacto de comparaciones (líneas 10, 15, 17, 24 y 26) requeridas por el siguiente algoritmo cuando n es par y cuando n es impar. Encuentre la notación theta para este algoritmo.

Entrada: $s_1, s_2, \ldots, s_n, n$

Salida: *grande* (el elemento mayor en $s_1, s_2, \ldots, s_n$), *chico* (el elemento menor en $s_1, s_2, \ldots, s_n$)

```
1.  grande_chico(s, n, grande, chico){
2.     if(n == 1)
3.        grande = s_1
4.        chico = s_1
5.        return
6.        }
7.     m = 2⌊n/2⌋
8.     i = 1
9.     while (i ≤ m − 1) {
10.       if (s_i > s_{i+1})
11           swap(s_i, s_{i+1})
12.       i = i + 2
13.    }
14.    if (n > m) {
15.       if (s_{m−1} > s_n)
16.          swap(s_{m−1}, s_n)
17.       if (s_n > s_m)
18.          swap(s_m, s_n)
19.       }
20.    pequeño = s_1
21.    grande = s_2
22.    i = 3
23.    while (i ≤ m − 1)
24.       if (s_i < pequeño)
25.          pequeño = s_i
26.       if (s_{i+1} > grande)
27.          grande = s_{i+1}
28.       i = i + 2
29.    }
30. }
```

31. Este ejercicio muestra otra manera de adivinar una fórmula para $1 + 2 + \ldots + n$.

El ejemplo 4.3.7 sugiere que

$$1 + 2 + \cdots + n = An^2 + Bn + C \quad \text{para toda } n,$$

para algunas constantes A, B y C. Suponiendo que esto es cierto, sustituya $n = 1, 2, 3$ para obtener tres ecuaciones con tres incógnitas A, B y C. Ahora se despejan A, B y C. Ahora puede demostrarse la fórmula obtenida usando inducción matemática (vea la sección 1.7).

32. Suponga que $a > 1$ y que $f(n) = \Theta(\log_a n)$. Demuestre que $f(n) = \Theta(\lg n)$.

33. Demuestre que $n! = O(n^n)$.

34. Demuestre que $2^n = O(n!)$.

35. Usando un argumento parecido al de los ejemplos 4.3.7 al 4.3.9 o algún otro, demuestre que $\sum_{i=1}^{n} i \lg i = \Theta(n^2 \lg n)$

★***36.*** Demuestre que $n^{n+1} = O(2^{n^2})$

37. Demuestre que $\lg(n^k + c) = \Theta(\lg n)$ para toda $k > 0$ y $c > 0$ fijas.

38. Demuestre que si n es una potencia de 2, por ejemplo, $n = 2^k$, entonces

$$\sum_{i=0}^{k} \lg(n/2^i) = \Theta(\lg^2 n).$$

39. Suponga que $f(n) = O(g(n))$ y $f(n) \geq 0$ y $g(n) > 0$ para toda $n \geq 1$. Demuestre que para alguna constante C, $f(n) = Cg(n)$ para toda $n \geq 1$.

40. Establezca y pruebe un resultado para Ω similar al del ejercicio 39.

41. Establezca y pruebe un resultado para Θ similar al de los ejercicios 39 y 40.

Determine si cada expresión en los ejercicios 42 al 52 es verdadera o falsa. Si es falsa, dé un contraejemplo. Suponga que las funciones f, g y h toman sólo valores positivos.

42. $n^n = O(2^n)$

43. $2 + \text{sen } n = O(2 + \cos n)$

44. Si $f(n) = \Theta(h(n))$ y $g(n) = \Theta(h(n))$, entonces $f(n) + g(n) = \Theta(h(n))$.

45. Si $f(n) = \Theta(g(n))$ entonces $cf(n) = \Theta(g(n))$ para cualquier $c \neq 0$.

46. Si $f(n) = \Theta(g(n))$ entonces $2^{f(n)} = \Theta(2^{g(n)})$

47. Si $f(n) = \Theta(g(n))$ entonces $\lg f(n) = \Theta(\lg g(n))$. Suponga que $f(n) \geq 1$ y $g(n) \geq 1$ para toda $n = 1, 2, \ldots$.

48. Si $f(n) = O(g(n))$ entonces $g(n) = O(f(n))$

49. Si $f(n) = O(g(n))$, entonces $g(n) = \Omega(f(n))$.

50. Si $f(n) = \Theta(g(n))$, entonces $g(n) = \Theta(f(n))$

51. $f(n) + g(n) = \Theta(h(n))$, donde $h(n) = \text{máx } \{f(n), g(n)\}$

52. $f(n) + g(n) = \Theta(h(n))$, donde $h(n) = \text{mín } \{f(n), g(n)\}$

53. Escriba exactamente qué significa $f(n) \neq O(g(n))$.

54. ¿Qué está mal en el siguiente argumento que intenta demostrar que no se puede tener al mismo tiempo $f(n) \neq O(g(n))$ y $g(n) \neq O(f(n))$?

Si $f(n) \neq O(g(n))$, entonces para toda $C > 0$, $|f(n)| > C|g(n)|$. En particular, $|f(n)| > 2|g(n)|$. Si $g(n) \neq O(f(n))$, entonces para toda $C > 0$, $|g(n)| > C|f(n)|$. En particular, $|g(n)| > 2|f(n)|$. Pero ahora

$$|f(n)| > 2|g(n)| > 4|f(n)|.$$

Cancelando $|f(n)|$ se tiene $1 > 4$, que es una contradicción. Por lo tanto, no es posible tener $f(n) \neq O(g(n))$ y $g(n) \neq O(f(n))$ al mismo tiempo.

★ **55.** Encuentre las funciones f y g que satisfacen

$$f(n) \neq O(g(n)) \text{ y } g(n) \neq O(f(n)).$$

★ ***56.*** Proporcione un ejemplo de funciones positivas crecientes f y g definidas en los enteros positivos para las que

$$f(n) \neq O(g(n)) \text{ y } g(n) \neq O(f(n)).$$

★ **57.** Demuestre que $n^k = O(c^n)$ para toda $k = 1, 2, \ldots$ y $c > 1$.

58. Encuentre funciones f, g, h y t que satisfagan

$$f(n) = \Theta(g(n)), \qquad h(n) = \Theta(t(n)),$$
$$f(n) - h(n) \neq \Theta(g(n) - t(n)).$$

59. Suponga que el tiempo en el peor caso de un algoritmo es $\Theta(n)$. ¿Cuál es el error en el siguiente razonamiento? Como $2n = \Theta(n)$, el tiempo en el peor caso para correr el algoritmo con una entrada de tamaño $2n$ será aproximadamente el mismo que el tiempo en el peor caso para correr el algoritmo con una entrada de tamaño n.

60. ¿Define

$$f(n) \neq O(g(n))$$

una relación de equivalencia en el conjunto de funciones de valores reales en $\{1, 2, \ldots\}$?

61. ¿Define

$$f(n) \neq \Theta(g(n))$$

una relación de equivalencia en el conjunto de funciones de valores reales en $\{1, 2, \ldots\}$?

62. [Requiere integración]

a) Demuestre, consultando la figura, que

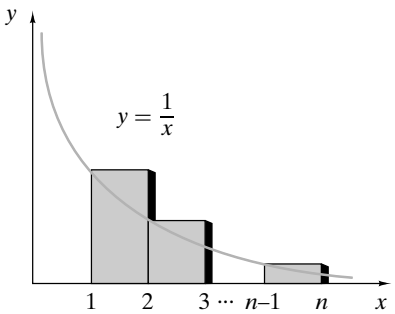
$$\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} < \log_e n.$$

b) Demuestre, consultando la figura, que

$$\log_e n < 1 + \frac{1}{2} + \cdots + \frac{1}{n-1}.$$

c) Use *a)* y *b)* para demostrar que

$$1 + \frac{1}{2} + \cdots + \frac{1}{n} = \Theta(\lg n).$$



63. [Requiere integración]. Use un argumento como el mostrado en el ejercicio 62 para demostrar que

$$\frac{n^{m+1}}{m+1} < 1^m + 2^m + \cdots + n^m < \frac{(n+1)^{m+1}}{m+1},$$

donde m es un entero positivo.

64. Usando la fórmula

$$\frac{b^{n+1} - a^{n+1}}{b-a} = \sum_{i=0}^{n} a^i b^{n-i} \qquad 0 \leq a < b$$

o de otra manera, demuestre que

$$\frac{b^{n+1} - a^{n+1}}{b-a} < (n+1)b^n \qquad 0 \leq a < b.$$

65. Sean $a = 1 + 1/(n+1)$ y $b = 1 + 1/n$ en la desigualdad del ejercicio 64 para demostrar que la sucesión $\{(1 + 1/n)^n\}$ es creciente.

66. Sean $a = 1$ y $b = 1 + 1/(2n)$ en la desigualdad del ejercicio 64 para demostrar que

$$\left(1 + \frac{1}{2n}\right)^n < 2$$

para toda $n \geq 1$. Use el ejercicio anterior para concluir que

$$\left(1 + \frac{1}{n}\right)^n < 4$$

para toda $n \geq 1$.

El método usado para probar los resultados de este ejercicio y su predecesor se debe aparentemente a Fort en 1862 (vea [Chrystal, vol. II, página 77]).

67. Usando los dos ejercicios anteriores o de otra manera, pruebe que

$$\frac{1}{n} \leq \lg(n+1) - \lg n < \frac{2}{n}$$

para toda $n \geq 1$.

68. Use el ejercicio anterior para probar que

$$\sum_{i=1}^{n} \frac{1}{i} = \Theta(\lg n).$$

(Compare con el ejercicio 62).

69. ¿Qué está mal en la siguiente "demostración" de que cualquier algoritmo tiene un tiempo de corrida $O(n)$?

Debemos demostrar que el tiempo requerido para una entrada de tamaño n es a lo más una constante multiplicada por n.

Paso base

Suponga que $n = 1$. Si el algoritmo toma C unidades de tiempo para una entrada de tamaño 1, tomará a lo más $C \cdot 1$ unidades de tiempo. Entonces la afirmación es verdadera para $n = 1$.

Paso inductivo

Suponga que el tiempo requerido para una entrada de tamaño n es a lo más $C'n$ y que el tiempo para procesar un elemento adicional es C''. Sea C el máximo entre C' y C''. Entonces el tiempo total requerido para una entrada de tamaño $n + 1$ es a lo más

$$C'n + C'' \leq Cn + C = C(n+1).$$

Esto verifica el paso inductivo.

Por inducción, para una entrada de tamaño n, el tiempo requerido es a lo más un tiempo constante n. Por lo tanto, el tiempo de corrida es $O(n)$.

En los ejercicios 70 al 75, determine si la afirmación es verdadera o falsa. Si es verdadera, demuéstrelo. Si es falsa, dé un contraejemplo. Suponga que f y g son funciones de valores reales definidas en el conjunto de enteros positivos y que $g(n) \neq 0$ para $n \geq 1$. Estos ejercicios requieren cálculo.

70. Si

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0,$$

entonces $f(n) \neq O(g(n))$.

71. Si

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0,$$

entonces $f(n) = \Theta(g(n))$.

72. Si

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c \neq 0,$$

entonces $f(n) = O(g(n))$.

73. Si

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c \neq 0,$$

entonces $f(n) = \Theta(g(n))$.

74. Si $f(n) = O(g(n))$, entonces

$$\lim_{n\to\infty} \frac{f(n)}{g(n)}$$

existe y es igual a algún número real.

75. Si $f(n) = \Theta(g(n))$, entonces

$$\lim_{n\to\infty} \frac{f(n)}{g(n)}$$

existe y es igual a algún número real.

★**76.** Use inducción para probar que

$$\lg n! \geq \frac{n}{2} \lg \frac{n}{2}.$$

77. [Requiere cálculo]. Sea $\ln x$ el logaritmo natural ($\log_e x$) de x. Use integración para obtener una estimación de la fórmula

$$n \ln n - n \leq \sum_{k=1}^{n} \ln k = \ln n!, \qquad n \geq 1.$$

78. Utilice el resultado del ejercicio 77 y la fórmula de cambio de base para logaritmos para obtener la fórmula

$$n \lg n - n \lg e \leq \lg n!, \qquad n \geq 1.$$

79. Deduzca

$$\lg n! \geq \frac{n}{2} \lg \frac{n}{2}$$

de la desigualdad del ejercicio 78.

Rincón de solución de problemas

Diseño y análisis de un algoritmo

Problema

Desarrolle y analice un algoritmo que regrese la suma máxima de valores consecutivos en la sucesión numérica

$$s_1, \ldots, s_n.$$

En notación matemática, el problema es encontrar la suma máxima de la forma $s_j + s_{j+1} + \cdots + s_i$. *Ejemplo:* Si la sucesión es

27 6 −50 21 −3 14 16 −8 42 33 −21 9,

el algoritmo regresa 115, la suma de

21 −3 14 16 −8 42 33.

Si todos los números en una sucesión son negativos, la suma máxima de valores consecutivos se define como 0. (La idea es que el máximo de 0 se logra al tomar una suma "vacía").

Cómo atacar el problema

Al desarrollar un algoritmo, una buena manera de comenzar es plantear la pregunta "¿cómo resolvería este problema a mano?" Al menos al principio, tome un enfoque directo. En este caso, tal vez sólo se enumeren las sumas de *todos* los valores consecutivos y se elija la más grande. Para la sucesión de ejemplo, las sumas son las siguientes:

	j											
i	1	2	3	4	5	6	7	8	9	10	11	12
1	27											
2	33	6										
3	−17	−44	−50									
4	4	−23	−29	21								
5	1	−26	−32	18	−3							
6	15	−12	−18	32	11	14						
7	31	4	−2	48	27	30	16					
8	23	−4	−10	40	19	22	8	−8				
9	65	38	32	82	61	64	50	34	42			
10	98	71	65	115	94	97	83	67	75	33		
11	77	50	44	94	73	76	62	46	54	12	−21	
12	86	59	53	103	82	85	71	55	63	21	−12	9

El elemento en la columna j, renglón i, es la suma

$$s_j + \ldots + s_i.$$

Por ejemplo, el elemento en la columna 4, renglón 7, es 48, la suma

$$s_4 + s_5 + s_6 + s_7 = 21 + -3 + 14 + 16 = 48.$$

Por inspección, se encuentra que 115 es la suma más grande.

Cómo encontrar una solución

Primero se escribe un seudocódigo para el algoritmo directo que calcula todas las sumas consecutivas y encuentra la mayor.

```
Entrada: s_1, . . . , s_n
Salida: máx
máx_sum1(s, n){
   //sum_ji es la suma s_j+. . .+s_i
   for i = 1 to n {
      for j = 1 to i − 1
            sum_ji = sum_j,i−1 + s_i
      sum_ii = s_i
   }
   //comparar toda sum_ji y encontrar el máximo
   máx = 0
   for i = 1 to n
      for j = 1 to i
            if(sum_ji > máx)
                  máx = sum_ji
   return máx
}
```

El primer ciclo anidado "for" calcula las sumas

$$sum_{ji} = s_j + \ldots + s_i.$$

El cálculo se basa en el hecho de que

$$sum_{ji} = s_j + \quad + s_i = s_j + \quad + s_{i\ \ 1} + s_i = sum_{j,i-1} + s_i.$$

El segundo "for" anidado recorre sum_{ji} y encuentra el valor más grande.

Como cada uno de los ciclos anidados toma un tiempo $\Theta(n^2)$, el tiempo de *máx_sum1* es $\Theta(n^2)$.

Se puede mejorar el tiempo real, pero no el tiempo asintótico del algoritmo, si se calcula el máximo dentro de los mismos ciclos "for" anidados donde se calcula sum_{ji}:

```
Entrada: s_1, . . . , s_n
Salida: máx
máx_sum2(s, n){
   //  sum_ji es la suma s_j+. . .+s_i.
   máx = 0
   for i = 1 to n {
      for j = 1 to i − 1{
            sum_ji = sum_j,i−1 + s_i
            if(sum_ji > máx)
                  máx = sum_ji
      }
      sum_ii = s_i
      if(sum_ii > máx)
            máx = sum_ii
   }
   return máx
}
```

Como el ciclo anidado toma un tiempo $\Theta(n^2)$, el tiempo de *máx_sum2* es $\Theta(n^2)$. Para reducir el tiempo asintótico, es necesario estudiar con cuidado el seudocódigo para ver si es posible mejorarlo.

Dos observaciones clave llevan a mejorar el tiempo. Primero, como se busca la suma *máxima*, no hay necesidad de registrar todas las sumas; se guardará sólo la suma máxima que termina en el índice i. Segundo, la línea

$$sum_{ji} = sum_{j,i-1} + s_i$$

muestra cómo se relaciona una suma consecutiva que termina en el índice $i - 1$ con una suma consecutiva que termina en el índice i. El máximo se calcula usando una fórmula similar. Si *sum* es la suma consecutiva máxima que termina en el índice $i - 1$, la suma consecutiva máxima que termina en el índice i se obtiene sumando s_i a *sum* siempre que $sum + s_i$ sea positiva. (Si alguna suma de términos consecutivos que termina en el índice i excede a $sum + s_i$, se podría eliminar s_i y obtener una suma de términos consecutivos que termina en $i - 1$ y que excede a *sum*, lo cual es imposible). Si $sum + s_i \leq 0$, la suma consecutiva máxima que termina en el índice i se obtiene con cero términos y tiene valor 0. Entonces se puede calcular la suma consecutiva máxima que termina en el índice i ejecutando

```
if(sum + s_i > 0)
   sum = sum + s_i
else
   sum = 0
```

Solución formal

```
   Entrada:   s_1, . . . , s_n
   Salida:    máx
máx_sum3(s, n){
   // máx es la suma máxima obtenida hasta ahora.
   // Después de la i-ésima iteración del ciclo "for"
   // sum es la suma consecutiva más grande
   // que termina en el índice i.
   máx = 0
   sum = 0
   for i = 1 to n {
      if (sum + s_i > 0)
         sum = sum + s_i
      else
         sum = 0
      if (sum > máx)
      máx = sum
   }
   return máx
}
```

Como este algoritmo tiene un solo ciclo "for" que corre de 1 a n, el tiempo de *máx_sum3* es $\Theta(n)$. El tiempo asintótico de este algoritmo no se puede mejorar más. Para encontrar la suma máxima de valores consecutivos, por lo menos debe verse cada elemento de la sucesión, lo que toma $\Theta(n)$.

Resumen de las técnicas de solución de problemas

- Al desarrollar un algoritmo, una buena manera de comenzar es hacer la pregunta "¿cómo resolvería esto a mano?"
- Al desarrollar un algoritmo, inicialmente tome un enfoque directo.
- Después de desarrollar un algoritmo, estudie con cuidado el seudocódigo para ver si es posible mejorarlo. Analice las partes que realizan los cálculos clave para tener un buen panorama de cómo mejorar la eficiencia del algoritmo.
- Igual que en la inducción matemática, amplíe la solución de un problema pequeño a un problema mayor. (En este problema, se extendió una suma que termina en el índice $i - 1$ a una suma que termina en el índice i).
- No repita cálculos. (En este problema, se extendió una suma que termina en el índice $i - 1$ a una suma que termina en el índice i agregando un término adicional en lugar de calcular la suma que termina en el índice i desde el principio. Este último método habría significado recalcular la suma que termina en el índice $i - 1$).

Comentarios

De acuerdo con [Bentley], el problema analizado en esta sección es la versión de una dimensión del problema original de dos dimensiones que maneja patrones coincidentes en imágenes digitales. El problema original era encontrar la suma máxima en una submatriz rectangular de una matriz $n \times n$ de números reales.

Ejercicios

1. Modifique *máx_sum3* de manera que calcule no sólo la suma máxima de valores consecutivos sino también los índices del primero y último términos de una subsucesión de suma máxima. Si no existe una subsucesión de suma máxima (lo que ocurriría, por ejemplo, si todos los valores de la sucesión fueran negativos), el algoritmo debe establecer el primero y último índices iguales a cero.

4.4 ➜ Algoritmos recursivos

WWW

Una **función recursiva** (seudocódigo) es una función que se invoca a sí misma. Un **algoritmo recursivo** es un algoritmo que contiene una función recursiva. La recursión es una forma poderosa, elegante y natural de resolver una clase amplia de problemas. Un problema de esta clase se resuelve mediante una técnica de *divide y vencerás* en la que el problema se descompone en problemas del mismo tipo que el problema original. Cada subproblema, a su vez, se descompone aún más hasta que el proceso produce subproblemas que se pueden resolver de manera directa. Por último, las soluciones de los subproblemas se combinan para obtener una solución del problema original.

Ejemplo 4.4.1 ▶

Recuerde que si $n \geq 1$, $n! = n(n-1) \cdots 2 \cdot 1$ y $0! = 1$. Observe que si $n \geq 2$, n factorial se puede escribir "en términos de sí mismo" ya que, si "despegamos" n, el producto que queda es simplemente $(n-1)!$; es decir.

$$n! = n(n-1)(n-2)\cdots 2 \cdot 1 = n \cdot (n-1)!.$$

Por ejemplo,

$$5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 5 \cdot 4!.$$

La ecuación

$$n! = n \cdot (n-1)!,$$

que es verdadera aun para $n = 1$, muestra cómo descomponer el problema original (calcular $n!$) en subproblemas cada vez más sencillos [calcular $(n-1)!$, calcular $(n-2)!$, . . .] hasta que el proceso llega al problema directo de calcular 0!. Las soluciones de estos problemas se pueden combinar, multiplicando, para resolver el problema original.

Por ejemplo, el problema de calcular 5! se reduce a calcular 4!; el problema de calcular 4! se reduce a calcular 3!, y así sucesivamente. La tabla 4.4.1 resume este proceso.

TABLA 4.4.1 ■ Descomposición del problema factorial.

Problema	*Problema simplificado*
5!	$5 \cdot 4!$
4!	$4 \cdot 3!$
3!	$3 \cdot 2!$
2!	$2 \cdot 1!$
1!	$1 \cdot 0!$
0!	Ninguno

TABLA 4.4.2 ■ Combinación de subproblemas en el problema factorial.

Problema	*Solución*
0!	1
1!	$1 \cdot 0! = 1$
2!	$2 \cdot 1! = 2$
3!	$3 \cdot 2! = 3 \cdot 2 = 6$
4!	$4 \cdot 3! = 4 \cdot 6 = 24$
5!	$5 \cdot 4! = 5 \cdot 24 = 120$

Una vez que el problema de calcular 5! se reduce a resolver subproblemas, la solución al problema más sencillo se utiliza para resolver el siguiente subproblema, etcétera, hasta que se resuelve el problema original. La tabla 4.4.2 ilustra cómo se combinan los subproblemas para calcular 5!. ◀

Ahora se escribirá un algoritmo recursivo que calcula factoriales. El algoritmo es una traducción directa de la ecuación

$$n! = n \cdot (n-1)!.$$

Algoritmo 4.4.2

Cálculo de *n* factorial

Este algoritmo recursivo calcula $n!$.

```
        Entrada: n, un entero mayor o igual que 0
        Salida: n!
1.  factorial(n)
2.      if (n == 0)
3.          return 1
4.      return n * factorial(n − 1)
5.  }
```

Se mostrará ahora cómo calcula $n!$ el algoritmo 4.4.2, para varios valores de n. Si $n = 0$, en la línea 3 la función regresa correctamente el valor 1.

Si $n = 1$, se procede a la línea 4 ya que $n \neq 0$. Se usa esta función para calcular 0!. Acabamos de ver que la función calcula 1 como el valor de 0!. En la línea 4, la función calcula correctamente el valor de 1!:

$$n \cdot (n-1)! = 1 \cdot 0! = 1 \cdot 1 = 1.$$

Si $n = 2$, se procede a la línea 4 ya que $n \neq 0$. Se emplea esta función para calcular 1!. Acabamos de ver que la función calcula 1 como el valor de 1!. En la línea 4, la función calcula correctamente el valor de 2!:

$$n \cdot (n-1)! = 2 \cdot 1! = 2 \cdot 1 = 2.$$

Si $n = 3$ se procede a la línea 4 ya que $n \neq 0$. Se utiliza esta función para calcular 2!. Acabamos de ver que la función calcula 2 como el valor de 2!. En la línea 4, la función calcula correctamente el valor de 3!:

$$n \cdot (n-1)! = 3 \cdot 2! = 3 \cdot 2 = 6.$$

Los argumentos anteriores se pueden generalizar usando inducción matemática para *probar* que el algoritmo 4.4.2 regresa el valor correcto de $n!$ para cualquier entero no negativo n.

Teorema 4.4.3

El algoritmo 4.4.2 regresa el valor de n!, $n \geq 0$.

Demostración

Paso base ($n = 0$)

Ya se ha observado que si $n = 0$, el algoritmo 4.4.2 regresa el valor correcto de 0! (1).

Paso inductivo

Suponga que el algoritmo 4.4.2 regresa el valor correcto de $(n-1)!$, $n > 0$. Ahora suponga que n es la entrada al algoritmo 4.4.2. Como $n \neq 0$, cuando se ejecuta la función en el algoritmo 4.4.2 se procede a la línea 4. Por la suposición inductiva, la función calcula el valor correcto de $(n-1)!$. En la línea 4, la función calcula el valor correcto de $(n-1)! \cdot n = n!$.

Por lo tanto, el algoritmo 4.4.2 regresa el valor correcto de $n!$ para todo entero $n \geq 0$.

Si se ejecuta en una computadora, en general el algoritmo 4.4.2 no será tan eficiente como la versión no recursiva a causa de todas las llamadas recurrentes generales.

Debe haber algunas situaciones en las que una función recursiva *no* se invoque a sí misma; de otra manera, se invocaría a sí misma sin detenerse. En el algoritmo 4.4.2, si $n = 0$, la función no se invoca a sí misma. Estos valores para los cuales una función recursiva no se invoca a sí misma se llaman *casos base*. Para resumir, toda función recursiva debe tener casos base.

Se mostró cómo utilizar la inducción matemática para probar que un algoritmo recursivo calcula el valor que asegura calcular. La relación entre la inducción matemática y los algoritmos recursivos es profunda. Con frecuencia, una prueba por inducción matemática se considera como un algoritmo para calcular un valor o llevar a cabo una construcción en particular. El paso base de una prueba por inducción matemática corresponde a los casos base de una función recursiva, y el paso inductivo de una prueba por inducción matemática corresponde al paso de una función recursiva donde la función se invoca a sí misma.

En el ejemplo 1.7.6 se demostró usando inducción matemática que, dado un tablero deficiente de $n \times n$ (un tablero que le falta un cuadro), donde n es una potencia de 2, se puede enlosar el tablero con trominos derechos (tres cuadros que forman una "L"; vea la figura 1.7.3). Ahora se traduce la prueba inductiva en un algoritmo recursivo para construir un enlosado de una tablero deficiente de $n \times n$ con trominos derechos, donde n es una potencia de 2.

Algoritmo 4.4.4

WWW

Enlosado de un tablero deficiente con trominos

Este algoritmo construye un enlosado con trominos derechos de un tablero deficiente de $n \times n$ donde n es una potencia de 2.

Entrada: n, una potencia de 2 (el tamaño del tablero); y la localización L del cuadro que falta

Salida: Enlosado de un tablero deficiente de $n \times n$

```
1.  enlosar(n, L){
2.     if(n == 2){
          // el tablero es un tromino derecho T
3.        enlosar con T
4.        return
5.     }
6.  dividir el tablero en cuatro tableros de (n/2) × (n/2)
7.  rotar el tablero para que el cuadro que falta esté en el cuadrante superior izquierdo
8.  colocar un tromino derecho en el centro // como en la figura 1.7.5
    // considere cada cuadro cubierto por el tromino central como
    // faltante, y denote los cuadros faltantes por m1, m2, m3, m4
9.  enlosar(n/2, m1)
10. enlosar(n/2, m2)
11. enlosar(n/2, m3)
12. enlosar(n/2, m4)
13. }
```

Usando el método de la prueba del teorema 4.4.3, se puede probar que el algoritmo 4.4.4 es correcto (ejercicio 4).

Se presenta un último ejemplo de un algoritmo recursivo.

Ejemplo 4.4.5 ▶

Un robot puede dar pasos de 1 o 2 metros. Se escribe un algoritmo para calcular el número de maneras que el robot puede caminar n metros. Como ejemplos:

Distancia	*Secuencia de pasos*	*Número de maneras de caminar*
1	1	1
2	1, 1 or 2	2
3	1, 1, 1 or 1, 2 or 2, 1	3
4	1, 1, 1, 1 or 1, 1, 2 or 1, 2, 1 or 2, 1, 1 or 2, 2	5

Sea $walk(n)$ el número de maneras en que el robot puede caminar n metros. Se ha observado que

$$walk(1) = 1, \quad walk(2) = 2.$$

Ahora suponga que $n > 2$. El robot comienza por dar un paso de 1 metro o un paso de 2 metros. Si el robot da primero un paso de 1 metro, queda una distancia de $n - 1$ metros; pero, por definición, el resto de la caminata se puede completar en $walk(n - 1)$ maneras. De manera similar, si el robot da primero un paso de 2 metros, queda una distancia de $n - 2$ metros y, en este caso, el resto de la caminata puede completarse en $walk(n - 2)$ maneras. Como la caminata debe comenzar con un paso de 1 metro o bien de 2 metros, todas las maneras de caminar n metros están tomadas en cuenta. Se obtiene la fórmula

$$walk(n) = walk(n - 1) + walk(n - 2).$$

Por ejemplo,

$$walk(4) = walk(3) + walk(2) = 3 + 2 = 5.$$

Se puede escribir un algoritmo para calcular $walk(n)$ traduciendo la ecuación

$$walk(n) = walk(n - 1) + walk(n - 2)$$

directamente en un algoritmo. Los casos base son $n = 1$ y $n = 2$. ◀

Algoritmo 4.4.6

Caminata de un robot

Este algoritmo calcula la función definida por

$$walk(n) = \begin{cases} 1, & n = 1 \\ 2, & n = 2 \\ walk(n-1) + walk(n-2) & n > 2. \end{cases}$$

```
Entrada: n
   Salida: walk(n)
   walk(n){
      if(n == 1 ∨ n == 2)
         return n
      return walk(n − 1) + walk(n − 2)
   }
```

Usando el método de la prueba del teorema 4.4.3, se puede probar que el algoritmo 4.4.6 es correcto (vea el ejercicio 7).

La sucesión

$$walk(1), walk(2), \ldots,$$

cuyos valores comienzan

$$1,\ 2,\ 3,\ 5,\ 8,\ 13, \ldots,$$

WWW está relacionada con la sucesión de Fibonacci. La **sucesión de Fibonacci** $\{f_n\}$ está definida por las ecuaciones

$$f_1 = 1$$
$$f_2 = 1$$
$$f_n = f_{n-1} + f_{n-2} \quad \text{para toda } n \geq 3.$$

La sucesión de Fibonacci comienza

$$1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13, \ldots,$$

Como

$$walk(1) = f_2,\ walk(2) = f_3,$$

y

$$walk(n) = walk(n-1) + walk(n-2),\ f_n = f_{n-1} + f_{n-2} \text{ para toda } n \geq 3,$$

Se sigue que

$$walk(n) = f_{n+1} \quad \text{para toda } n \geq 1.$$

(Este argumento se puede formalizar usando inducción matemática; vea el ejercicio 8).

La sucesión de Fibonacci recibe su nombre en honor a Leonardo Fibonacci (alrededor de 1170-1250), un comerciante y matemático italiano. La sucesión surgió por un enigma referente a conejos (vea los ejercicios 18 y 19). Después de regresar de Oriente en 1202, Fibonacci escribió su trabajo más famoso, *Liber Abaci* (disponible en una traducción al inglés de [Sigler]), que, además de contener lo que ahora se conoce como sucesión de Fibonacci, defendía el uso del sistema de números hindú-arábigos. Este libro fue una de las influencias principales para adoptar el sistema decimal en Europa Occidental. Fibonacci firmó muchos de sus trabajos como "Leonardo Bigollo". *Bigollo* se traduce como "viajero" o "cabeza dura". Existe evidencia de que Fibonacci disfrutaba que sus contemporáneos los consideraran cabeza dura por recomendar el uso de un nuevo sistema numérico.

La sucesión de Fibonacci surge en lugares inesperados. La figura 4.4.1 muestra una piña de pino con 13 espirales en sentido de la manecillas de reloj y 8 espirales en sentido contrario. Muchas plantas distribuyen sus semillas tan parejo como les es posible, con lo que maximizan el espacio para cada semilla. El patrón en el que el número de espirales es un nú-

Figura 4.4.1 Piña de pino. Tiene 13 espirales en sentido de las manecillas del reloj (marcadas con hilo blanco) y 8 espirales en sentido contrario (marcadas con hilo oscuro). [*Foto del autor; piña de pino cortesía de André Berthiaume y Sigrid (Anne) Settle*].

mero de Fibonacci proporciona la distribución más uniforme (vea [Naylor, Mitchison]). En la sección 5.3, la sucesión de Fibonacci aparece en el análisis del algoritmo euclideano.

Ejemplo 4.4.7 ▶ Utilice inducción matemática para demostrar que

$$\sum_{k=1}^{n} f_k = f_{n+2} - 1 \qquad \text{para toda } n \geq 1.$$

Para el paso base ($n = 1$), debemos demostrar que

$$\sum_{k=1}^{1} f_k = f_3 - 1.$$

Como $\sum_{k=1}^{1} f_k = f_1 = 1$ y $f_3 - 1 = 2 - 1 = 1$, la ecuación se verifica.

Para el paso inductivo, se supone el caso n

$$\sum_{k=1}^{n} f_k = f_{n+2} - 1$$

y se prueba el caso $n + 1$

$$\sum_{k=1}^{n+1} f_k = f_{n+3} - 1.$$

Ahora

$$\begin{aligned} \sum_{k=1}^{n+1} f_k &= \sum_{k=1}^{n} f_k + f_{n+1} \\ &= (f_{n+2} - 1) + f_{n+1} \qquad \text{por la suposición inductiva} \\ &= f_{n+1} + f_{n+2} - 1 \\ &= f_{n+3} - 1. \end{aligned}$$

La última igualdad es cierta por la definición de los números de Fibonacci:

$$f_n = f_{n-1} + f_{n-2} \quad \text{para toda } n \geq 3.$$

Puesto que se han verificado el paso base y el paso inductivo, la ecuación dada es verdadera para toda $n \geq 1$. ◀

Sugerencias para resolver problemas

Una función recursiva es una función que se invoca a sí misma. La clave para escribir una función recursiva es encontrar una instancia más pequeña del problema dentro del problema más grande. Por ejemplo, se puede calcular $n!$ de manera recursiva porque $n! = n \cdot (n - 1)!$ para toda $n \geq 1$. La situación es análoga al paso inductivo de la inducción matemática cuando debemos encontrar un caso más pequeño (como el caso n) dentro del caso grande (caso $n + 1$).

Como otro ejemplo, el enlosado del tablero deficiente de $n \times n$ con trominos cuando n es una potencia de 2 se puede hacer de manera recursiva porque es posible encontrar cuatro subtableros de $(n/2) \times (n/2)$ dentro del tablero original de $n \times n$. Observe la similitud del algoritmo para enlosar con el paso inductivo de la prueba de que todo tablero deficiente de $n \times n$ se puede enlosar con trominos cuando n es una potencia de 2.

Para probar una afirmación acerca de los números Fibonacci, use la ecuación

$$f_n = f_{n-1} + f_{n-2} \quad \text{para toda } n \geq 3.$$

Con frecuencia, la demostración usará inducción matemática *y* la ecuación anterior (vea el ejemplo 4.4.7).

Sección de ejercicios de repaso

1. ¿Qué es un algoritmo recursivo?
2. ¿Qué es una función recursiva?
3. Dé un ejemplo de una función recursiva.
4. Explique cómo funciona la técnica de *divide y vencerás*.
5. ¿Cuál es la base de una función recursiva?
6. ¿Por qué toda función recursiva debe tener un caso base?
7. ¿Cómo se define la sucesión de Fibonacci?
8. Dé los primeros cuatro valores de la sucesión de Fibonacci.

Ejercicios

1. Haga el seguimiento del algoritmo 4.4.2 para $n = 4$.
2. Haga el seguimiento del algoritmo 4.4.4 cuando $n = 4$ y el cuadro que falta está en la esquina superior izquierda del cuadrado.
3. Dé seguimiento al algoritmo 4.4.4 cuando $n = 8$ y el cuadrado que falta está a cuatro cuadros de la izquierda y a seis del borde superior.
4. Demuestre que el algoritmo 4.4.4 es correcto.
5. Dé seguimiento al algoritmo 4.4.6 para $n = 4$.
6. Haga el seguimiento del algoritmo 4.4.6 para $n = 5$.
7. Pruebe que el algoritmo 4.4.6 es correcto.
8. Pruebe que
$$walk(n) = f_{n+1} \quad \text{para toda } n \geq 1.$$
9. *a)* Use las fórmulas
$$s_1 = 1, \qquad s_n = s_{n-1} + n \qquad \text{para toda } n \geq 2,$$
para escribir un algoritmo recursivo que calcule
$$s_n = 1 + 2 + 3 + \ldots n.$$
b) Dé una prueba usando inducción matemática de que su algoritmo del inciso *a)* es correcto.
10. *a)* Use las fórmulas
$$s_1 = 2, \qquad s_n = s_{n-1} + 2n \qquad \text{para toda } n \geq 2,$$
para escribir un algoritmo recursivo que calcule
$$s_n = 2 + 4 + 6 + \ldots 2n.$$
b) Dé una prueba usando inducción matemática de que su algoritmo del inciso *a)* es correcto.
11. *a)* Un robot puede dar pasos de 1, 2 o 3 metros. Escriba un algoritmo recursivo que calcule el número de maneras en que el robot puede caminar n metros.
b) Dé una prueba usando inducción matemática de que su algoritmo del inciso *a)* es correcto.
12. Escriba un algoritmo recursivo para encontrar el mínimo de una sucesión finita de números. Dé una prueba con inducción matemática de que su algoritmo es correcto.
13. Escriba un algoritmo recursivo para encontrar el máximo de una sucesión finita de números. Dé una prueba usando inducción matemática de que su algoritmo es correcto.
14. Escriba un algoritmo recursivo que invierta una sucesión finita. Proporcione una prueba usando inducción matemática de que su algoritmo es correcto.
15. Escriba una algoritmo no recursivo para calcular $n!$.
★16. Un robot puede dar pasos de 1 o 2 metros. Escriba un algoritmo para numerar todas las maneras en que el robot puede caminar n metros.
★17. Un robot puede dar pasos de 1, 2 o 3 metros. Escriba un algoritmo para numerar todas las maneras en que un robot puede caminar n metros.

Los ejercicios 18 al 32 se refieren a la sucesión de Fibonacci $\{f_n\}$.

18. Suponga que al inicio del año hay un par de conejos y que cada mes cada par produce un nuevo par que se convierte en productivo después de un mes. Suponga, además, que no ocurren muertes. Sea a_n el número de pares de conejos al final del mes n. Demuestre que $a_1 = 1$, $a_2 = 2$ y $a_n - a_{n-1} = a_{n-2}$. Pruebe que $a_n = f_{n+1}$ para toda $n \geq 1$.
19. La pregunta original de Fibonacci era: En las condiciones del ejercicio 18, ¿cuántos pares de conejos hay después de un año? Responda a la pregunta de Fibonacci.
20. Demuestre que el número de maneras para enlosar un tablero de $2 \times n$ con piezas rectangulares de 1×2 es f_{n+1}, el n ésimo número de Fibonacci.
21. Use inducción matemática para demostrar que
$$f_n^2 = f_{n-1} f_{n+1} + (-1)^{n+1} \qquad \text{para toda } n \geq 2.$$
22. Demuestre que
$$f_{n+2}^2 - f_{n+1}^2 = f_n f_{n+3} \qquad \text{para toda } n \geq 1.$$
23. Demuestre que
$$f_n^2 = f_{n-2} f_{n+2} + (-1)^n \qquad \text{para toda } n \geq 3.$$
24. Use inducción matemática para demostrar que
$$\sum_{k=1}^{n} f_k^2 = f_n f_{n+1} \qquad \text{para toda } n \geq 1.$$
★25. Use inducción matemática para demostrar que
$$f_{2n} = f_{n+1}^2 - f_{n-1}^2 \quad \text{y} \quad f_{2n+1} = f_n^2 + f_{n+1}^2 \quad \text{para toda } n \geq 2.$$
26. Use inducción matemática para demostrar que para toda $n \geq 1$, f_n es par si y sólo si n es divisible entre 3.
27. Use inducción matemática para demostrar que para toda $n \geq 6$,
$$f_n > \left(\frac{3}{2}\right)^{n-1}.$$
28. Use inducción matemática para demostrar que para toda $n \geq 1$,
$$f_n \leq 2^{n-1}.$$
29. Use inducción matemática para demostrar que para toda $n \geq 1$,
$$\sum_{k=1}^{n} f_{2k-1} = f_{2n}, \qquad \sum_{k=1}^{n} f_{2k} = f_{2n+1} - 1.$$
★30. Use inducción matemática para demostrar que todo entero $n \geq 1$ se puede expresar como la suma de números de Fibonacci distintos, sin que haya dos consecutivos.
★31. Demuestre que la representación en el ejercicio 30 es única si no se permite f_1 como sumando.

32. Demuestre que para toda $n \geq 2$,

$$f_n = \frac{f_{n-1} + \sqrt{5f_{n-1}^2 + 4(-1)^{n+1}}}{2}.$$

Observe que esta fórmula da f_n en términos de un predecesor en lugar de dos predecesores como en la definición original.

33. [Requiere cálculo]. Suponga la fórmula para diferenciar productos:

$$\frac{d(fg)}{dx} = f\frac{dg}{dx} + g\frac{df}{dx}.$$

Use inducción matemática para probar que

$$\frac{dx^n}{dx} = nx^{n-1} \qquad \text{para } n = 1, 2, \ldots.$$

34. [Requiere cálculo]. Explique cómo la fórmula da un algoritmo recursivo para integrar $\log^n |x|$:

$$\int \log^n |x|\, dx = x \log^n |x| - n \int \log^{n-1} |x|\, dx.$$

Dé otros ejemplos de fórmula de integración recursivas.

Notas

La primera parte de [Knuth, 1977] introduce el concepto de un algoritmo y varios temas de matemáticas, incluyendo la inducción matemática. La segunda mitad se dedica a las estructuras de datos.

Casi todas las referencias generales de ciencias de la computación contienen algún análisis de algoritmos. Algunos libros específicos de algoritmos son [Aho; Baase; Brassard; Cormen; Johnsonbaugh; Knuth, 1997, 1998a, 1998b; Manber; Miller; Nievergelt; y Reingold]. [McNaughton] contiene un análisis bastante completo a nivel de introducción de qué es un algoritmo. El artículo de Knuth acerca de algoritmos ([Knuth, 1977]) y su artículo acerca del papel de los algoritmos en la ciencia de las matemáticas ([Knuth, 1985]) también se recomiendan. [Gardner, 1992] incluye un capítulo de la sucesión de Fibonacci.

Repaso del capítulo

Sección 4.1

1. Algoritmo
2. Propiedades de una algoritmo: entrada, salida, precisión, determinismo, carácter finito, corrección, generalidad
3. Seguimiento
4. Seudocódigo

Sección 4.2

5. Búsqueda
6. Búsqueda de texto
7. Algoritmo de búsqueda de texto
8. Ordenar
9. Inserción por orden
10. Tiempo y espacio para algoritmos
11. Tiempo en el mejor caso
12. Tiempo en el peor caso
13. Algoritmo aleatorizado
14. Algoritmo para desordenar

Sección 4.3

15. Análisis de algoritmos
16. Tiempo en el peor caso para un algoritmo
17. Tiempo en el mejor caso para un algoritmo
18. Tiempo en el caso promedio para un algoritmo
19. Notación de O mayúscula: $f(n) = O(g(n))$
20. Notación omega: $f(n) = \Omega(g(n))$
21. Notación theta: $f(n) = \Theta(g(n))$

Sección 4.4

22. Algoritmo recursivo
23. Función recursiva
24. Técnica de divide y vencerás

25. Casos base: situaciones en las que la función recursiva no se invoca a sí misma
26. Sucesión de Fibonacci $\{f_n\}$: $f_1 = 1$, $f_2 = 1$, $f_n = f_{n-1} + f_{n-2}$, $n \geq 3$

Autoevaluación del capítulo

Sección 4.1

1. Dé seguimiento al algoritmo 4.1.1 para los valores $a = 12$, $b = 3$ y $c = 0$.
2. Escriba un algoritmo que recibe como entrada números diferentes a, b y c y asigna los valores a, b y c a las variables x, y y z de manera que

$$x < y < z.$$

3. Escriba un algoritmo que regresa verdadero si los valores a, b y c son diferentes, y falso de otra manera.
4. ¿Cuáles de las propiedades de los algoritmos (entrada, salida, precisión, determinismo, carácter finito, corrección, generalidad) faltan en lo siguiente, si acaso faltan? Explique.

Entrada: S (un conjunto de enteros), m (un entero)
Salida: Todos los subconjuntos finitos de S que suman m

1. Liste todos los subconjuntos finitos de S y sus sumas.
2. Recorra la lista de subconjuntos en 1 y envíe a la salida todo aquel cuya suma sea m.

Sección 4.2

5. Dé seguimiento al algoritmo 4.2.1 para la entrada $t =$ "111011" y $p =$ "110".
6. Dé seguimiento al algoritmo 4.2.3 para la entrada

$$44 \quad 64 \quad 77 \quad 15 \quad 3.$$

7. Dé seguimiento al algoritmo 4.2.4 para la entrada

$$5 \quad 51 \quad 2 \quad 44 \quad 96.$$

Suponga que los valores *rand* son

$$rand(1, 5) = 1, \ rand(2, 5) = 3, \ rand(3, 5) = 5, \ rand(4, 5) = 5.$$

8. Escriba un algoritmo que recibe como entrada la sucesión

$$s_1, \ldots, s_n$$

dada en orden no decreciente e imprime todos los valores que aparecen más de una vez. *Ejemplo:* Si la sucesión es

$$1 \quad 1 \quad 1 \quad 5 \quad 8 \quad 8 \quad 9 \quad 12,$$

la salida es

$$1 \quad 8.$$

Sección 4.3

Seleccione una notación theta entre $\Theta(1)$, $\Theta(n)$, $\Theta(n^2)$, $\Theta(n^3)$, $\Theta(n^4)$, $\Theta(2^n)$ *o* $\Theta(n!)$ *para cada expresión en los ejercicios 9 y 10.*

9. $4n^3 + 2n - 5$
10. $1^3 + 2^3 + \cdots + n^3$
11. Seleccione una notación theta entre $\Theta(1)$, $\Theta(n)$, $\Theta(n^2)$, $\Theta(n^3)$, $\Theta(2^n)$ o $\Theta(n!)$ para el número de veces que se ejecuta la línea $x = x + 1$.

```
for i = 1 to n
    for j = 1 to n
        x = x + 1
```

12. Escriba un algoritmo que pruebe si dos matrices de $n \times n$ son iguales y encuentre una notación theta para el tiempo del peor caso.

Sección 4.4.

13. Dé seguimiento al algoritmo 4.4.4 (algoritmo de enlosar con trominos) cuando $n = 8$ y el cuadro que falta está a cuatro de la izquierda y a dos de arriba (borde superior).

Los ejercicios 14 al 16 se refieren a la sucesión de tribonacci $\{t_n\}$ definida por las ecuaciones

$$t_1 = t_2 = t_3 = 1, \qquad t_n = t_{n-1} + t_{n-2} + t_{n-3} \qquad \textit{para toda } n \geq 4.$$

14. Encuentre t_4 y t_5.

15. Escriba un algoritmo recursivo para calcular t_n, $n \geq 1$.

16. Dé una prueba usando inducción matemática de que su algoritmo para el ejercicio 15 es correcto.

Sección de ejercicios de repaso

1. Implemente el algoritmo 4.1.2 como un programa, encontrando el elemento mayor en la sucesión.

2. Implemente el algoritmo 4.2.1 como un programa de búsqueda de texto.

3. Implemente el algoritmo 4.2.3 como un programa de inserción por orden.

4. Implemente el algoritmo 4.2.4 como un programa para desordenar.

5. Corra desordenar (*shuffle*, algoritmo 4.2.4) muchas veces para la misma sucesión de entrada. ¿Cómo puede analizarse la salida para determinar si es verdaderamente "aleatorio"?

6. Implemente la selección por orden (ejercicio 20, sección 4.2) como un programa.

7. Compare los tiempos de corrida de inserción por orden (algoritmo 4.2.3) y selección por orden (ejercicio 20, sección 4.2) para varias entradas de tamaños diferentes. Incluya datos en orden no decreciente, en orden no creciente, datos que contienen muchos duplicados y datos en orden aleatorio.

8. Escriba un programa recursivo y otro no recursivo para calcular $n!$. Compare los tiempos requeridos por los programas.

9. Escriba un programa cuya entrada es un tablero de $2^n \times 2^n$ con un cuadro faltante y cuya salida es un enlosado del tablero con trominos.

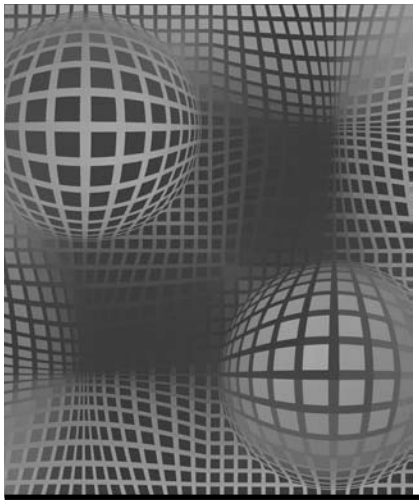
10. Escriba un programa que use una pantalla de gráficas para mostrar un enlosado con trominos de un tablero de $2^n \times 2^n$ con un cuadro faltante.

11. Escriba un programa que enlosa con trominos un tablero de $n \times n$ con un cuadro faltante, siempre que $n \neq 5$ y 3 no divida a n.

12. Escriba un programa recursivo y otro no recursivo para calcular la sucesión de Fibonacci. Compare los tiempos requeridos por los programas.

13. Un robot puede dar pasos de 1 o 2 metros. Escriba un programa para numerar todas las maneras en que el robot camina n metros.

14. Un robot puede dar pasos de 1, 2 o 3 metros. Escriba un programa para numerar todas las maneras en que el robot camina n metros.



Capítulo 5

INTRODUCCIÓN A LA TEORÍA DE NÚMEROS

Es una cámara digital. La tengo en 42 exposiciones, pero quiero ponerla en 47... porque es un número primo.

DE ALIAS

La **teoría de números** es una rama de las matemáticas que se ocupa de los números enteros. Por tradición, la teoría de números era una rama de las matemáticas *puras*, conocida por su naturaleza abstracta más que por sus aplicaciones. El gran matemático inglés G. H. Hardy (1877-1947) usó la teoría de números como ejemplo de una rama de las matemáticas bella pero impráctica. Sin embargo, a finales del siglo XX, la teoría de números ha adquirido una gran importancia en los sistemas criptográficos, esto es, sistemas que se usan para la seguridad en las comunicaciones.

En los capítulos anteriores, se usaron algunas definiciones básicas de la teoría de números como "divide" y "número primo". En la sección 5.1 se repasarán estas definiciones básicas y se ampliará el estudio a factorización única, máximo común divisor y mínimo común múltiplo.

En la sección 5.2 se analizarán las representaciones de los números enteros y algunos algoritmos para la aritmética entera.

El algoritmo euclidiano para calcular el máximo común divisor es el tema de la sección 5.3. Éste es sin duda uno de los algoritmos más antiguos. Euclides vivió aproximadamente en 295 A.C., y el algoritmo tal vez es anterior.

Como una aplicación de la teoría de números presentada en las secciones 5.1 a la 5.3, en la 5.4 se analizará el sistema RSA de seguridad en las comunicaciones.

5.1 ➜ Divisores

En esta sección se presentan las definiciones y terminología básicas. Primero se recordará la definición de "divide", y se introducirá cierta terminología relacionada.

Definición 5.1.1 ▶

Sean n y d enteros, $d \neq 0$. Se dice que *d divide a n* si existe un entero q que satisface $n = dq$; q se llama el *cociente* y d el *divisor* o *factor* de n. Si d divide a n, se escribe $d \mid n$. Si d no divide a n, se escribe $d \nmid n$. ◀

Ejemplo 5.1.2 ▶

Como $21 = 3 \cdot 7$, 3 divide a 21 y escribimos $3|21$. El cociente es 7; 3 recibe el nombre de divisor o factor de 21. ◀

Se observa que si n y d son enteros positivos y $d|n$, entonces $d \leq n$. (Si $d|n$, existe un entero q tal que $n = dq$. Como n y d son enteros positivos, $1 \leq q$. Por lo tanto, $d \leq dq = n$).

Ya sea que un entero $d > 0$ divida o no a un entero n, se obtiene un cociente único q y un residuo r como lo establece el teorema del cociente-residuo (Teorema 1.8.5): Existen enteros únicos q (cociente) y r (residuo) que satisfacen $n = dq + r$, $0 \leq r < d$. El residuo r es igual a cero si y sólo si d divide a n.

Algunas propiedades adicionales de los divisores se dan en el siguiente Teorema y serán útiles en el trabajo de este capítulo.

Teorema 5.1.3

Sean m, n y d enteros.

a) Si $d|m$ y $d|n$, entonces

$$d|(m + n).$$

b) Si $d|m$ y $d|n$, entonces

$$d|(m - n).$$

c) Si $d|m$, entonces $d|mn$.

Demostración *a)* Suponga que $d|m$ y $d|n$. Por la definición 5.1.1,

$$m = dq_1 \tag{5.1.1}$$

para algún entero q_1 y

$$n = dq_2 \tag{5.1.2}$$

para algún entero q_2. Si se suman las ecuaciones (5.1.1) y (5.1.2), se obtiene

$$m + n = dq_1 + dq2 = d(q_1 + q_2).$$

Por lo tanto, d divide a $m + n$ (con cociente $q_1 + q_2$). Esto prueba el inciso *a)*.

Las pruebas de los incisos *b)* y *c)* se dejan como ejercicios (vea los ejercicios 27 y 28).

Definición 5.1.4 ▶

Un entero mayor que 1 cuyos únicos divisores positivos son 1 y él mismo se llama *primo*. Un entero mayor que 1 que no es primo se llama *compuesto*. ◀

Ejemplo 5.1.5 ▶

El entero 23 es primo porque sus únicos divisores son 1 y él mismo. El entero 34 es compuesto porque es divisible entre 17, que no es 1 ni 34 ◀

Si un entero $n > 1$ es compuesto, entonces tiene un divisor positivo d diferente de 1 y él mismo. Como d es positivo y $d \neq 1$, $d > 1$. Como d es un divisor de n, $d \leq n$. Como $d \neq n$, $d < n$. Por lo tanto, para determinar si un entero positivo n es compuesto, es suficiente con probar si alguno de los enteros

$$2, 3, \ldots, n - 1$$

divide a n. Si algún entero en esta lista divide a n, entonces n es compuesto. Si no hay un entero en esta lista que divida a n, entonces n es primo. (En realidad, la lista se puede reducir de manera considerable; vea el teorema 5.1.7).

Ejemplo 5.1.6 ▶

Por inspección, se encuentra que ningún número de la lista

$$2, 3, 4, 5, \ldots, 41, 42$$

divide a 43; entonces 43 es primo.

Se verifica la lista

$$2, 3, 4, 5, \ldots, 449, 450$$

en busca de divisores potenciales de 451, se encuentra que 11 divide a 451 ($451 = 11 \cdot 41$); así, 451 es compuesto. ◀

En el ejemplo 5.1.6, para determinar si un entero positivo $n > 1$ es primo, se verificaron los divisores potenciales

$$2, 3, \ldots, n - 1.$$

En realidad es suficiente con verificar

$$2, 3, \ldots, \lfloor \sqrt{n} \rfloor.$$

Teorema 5.1.7

Un entero positivo n mayor que 1 es compuesto si y sólo si n tiene un divisor d que satisface $2 \leq d \leq \sqrt{n}$.

Demostración Debemos probar que

Si n es compuesto, entonces n tiene un divisor d que satisface $2 \leq d \leq \sqrt{n}$,

y

si n tiene un divisor d que satisface $2 \leq d \leq \sqrt{n}$, entonces n es compuesto.

Primero probamos

si n es compuesto, entonces n tiene un divisor d que satisface $2 \leq d \leq \sqrt{n}$.

Suponga que n es compuesto. El análisis que sigue al ejemplo 5.1.5 muestra que n tiene un divisor d' que satisface

$$2 \leq d' < n.$$

Se dará un argumento por casos. Si $d' \leq \sqrt{n}$, entonces n tiene un divisor d (a saber $d = d'$) que satisface $2 \leq d \leq \sqrt{n}$.

El otro caso es $d' > \sqrt{n}$. Como d' divide a n, por la definición 5.1.1 existe un entero q que satisface $n = d'q$. Entonces q también es un divisor de n. Se asegura que $q \leq \sqrt{n}$. Para demostrarlo se usa la prueba por contradicción. Suponga que $q \leq \sqrt{n}$. Multiplicando $d' > \sqrt{n}$ por $q > \sqrt{n}$, se obtiene

$$n = d'\, q > \sqrt{n}\,\sqrt{n} = n,$$

que es una contradicción. Entonces, $q \leq \sqrt{n}$. Por lo tanto, n tiene un divisor d (de hecho $d = q$) que satisface $2 \leq d \leq \sqrt{n}$.

Queda por probar

si n tiene un divisor d que satisface $2 \leq d \leq \sqrt{n}$, entonces n es compuesto.

Si n tiene un divisor d que satisface $2 \leq d \leq \sqrt{n}$, por la definición 5.1.4, n es compuesto. Esto completa la prueba.

Se usará el Teorema 5.1.7 para construir el siguiente algoritmo que prueba si un entero $n > 1$ es primo.

Algoritmo 5.1.8

Prueba para determinar si un entero es primo

Este algoritmo determina si el entero $n > 1$ es primo. Si n es primo, el algoritmo regresa 0. Si n es compuesto, el algoritmo regresa un divisor d que satisface $2 \leq d \leq \sqrt{n}$. Para probar si d divide a n, el algoritmo verifica si el residuo al dividir n entre d, n mod d, es cero.

Entrada: n

Salida: d

```
es_primo(n) {
    for d = 2 to ⌊√n⌋
        if(n mod d == 0)
            return d
    return 0
}
```

Ejemplo 5.1.9 ▶

Para determinar si 43 es primo, el algoritmo 5.1.8 verifica si alguno de los números

$$2, 3, 4, 5, 6 = \lfloor\sqrt{43}\rfloor$$

divide a 43. Como ninguno de ellos divide a 43, la condición

$$n \bmod d == 0$$

siempre es falsa. Por lo tanto, el algoritmo regresa 0 para indicar que 43 es primo.

Para determinar si 451 es primo, el algoritmo verifica si alguno de los números

$$2, 3, \ldots, 21 = \lfloor\sqrt{451}\rfloor$$

divide a 451. Para $d = 2, 3, \ldots, 10$, d no divide a 451 y la condición

$$n \bmod d == 0$$

es falsa. Sin embargo, cuando $d = 11$, d divide a 451 y la condición

$$n \bmod d == 0$$

es verdadera. Por lo tanto, el algoritmo regresa 11 para indicar que 451 es compuesto y 11 divide a 451. ◀

En el peor caso (cuando n es primo y el ciclo "for" corre hasta completarse), el algoritmo 5.1.8 toma un tiempo $\sqrt{n}$. Aunque el algoritmo 5.1.8 corra en tiempo polinomial en n (ya que $\sqrt{n} \leq n$), *no* corre en tiempo polinomial en el *tamaño* de la entrada (que es n). [Se puede representar n en un espacio mucho menor que $\Theta(n)$; vea el ejemplo 5.2.1]. Se dice que el algoritmo 5.1.8 no es un algoritmo con tiempo polinomial. No se sabe si existe un algoritmo con tiempo polinomial que pueda encontrar un factor de un entero dado; pero muchos científicos de la computación piensan que no existe tal algoritmo. Por otro lado, Manindra Agarwal y dos de sus alumnos, Nitin Saxena y Neeraj Kayal, descubrieron hace poco (2002) un algoritmo con tiempo polinomial que puede determinar si un entero es primo o no (vea [Agarwal]). La cuestión de la existencia de un algoritmo con tiempo polinomial que pueda factorizar un entero tiene un interés más que académico puesto que la seguridad de ciertos sistemas cifrados o encriptados se basa en la no existencia de un algoritmo de este tipo (vea la sección 5.4).

Observe que si un entero compuesto n es la entrada al algoritmo 5.1.8, el divisor que produce es primo; es decir, el algoritmo 5.1.8 regresa un factor primo de un entero compuesto. Para probar esto, se utiliza la prueba por contradicción. Si el algoritmo 5.1.8 regresa un divisor compuesto de n, por ejemplo a, entonces a tiene un divisor a' menor que a. Como a' también divide a n y $a' < a$, cuando el algoritmo 5.1.8 hace $d = a'$, regresará a', no a. Esta contradicción demuestra que si un entero compuesto n se alimenta al algoritmo 5.1.8, el divisor que se obtiene es primo.

Ejemplo 5.1.10 ▶

Si la entrada al algoritmo 5.1.8 es $n = 1274$, el algoritmo regresa el primo 2 porque 2 divide a 1274, en particular

$$1274 = 2 \cdot 637.$$

Si ahora la entrada es $n = 637$, el algoritmo 5.1.8 regresa el primo 7 porque 7 divide a 637, en particular

$$637 = 7 \cdot 91.$$

Si ahora se da $n = 91$ al algoritmo 5.1.8, regresará el primo 7 porque 7 divide 91, específicamente

$$91 = 7 \cdot 13.$$

Si ahora la entrada es $n = 13$, el algoritmo 5.1.8 regresa 0 porque 13 es primo.
Al combinar las ecuaciones anteriores se tiene 1247 como producto de primos

$$1274 = 2 \cdot 637 = 2 \cdot 7 \cdot 91 = 2 \cdot 7 \cdot 7 \cdot 13.$$

Se ha ilustrado cómo expresar cualquier entero mayor que 1 como un producto de primos. También es un hecho (aunque no se demostrará en este libro) que, excepto por el orden, los factores primos son únicos. Este resultado se conoce como **teorema fundamental de la aritmética** o **teorema de factorización única**. ◀

Teorema 5.1.11

Teorema fundamental de la aritmética

Cualquier entero mayor que 1 se puede expresar como un producto de primos. Más aún, si los primos se escriben en orden no decreciente, la factorización es única. En símbolos, si

$$n = p_1 p_2 \cdots p_i,$$

donde las p_k son primos y $p_1 \le p_2 \le \ldots \le p_i$, y

$$n = p'_1 p'_2 \cdots p'_j,$$

donde las p'_k son primos y $p'_1 \le p'_2 \le \ldots \le p'_i$, entonces $i = j$ y

$$p_k = p'_k \qquad \text{para toda } k = 1, \ldots, i.$$

Ahora se probará que el número de primos es infinito.

Teorema 5.1.12

El número de primos es infinito.

Demostración Será suficiente demostrar que si p es primo, existe un primo mayor que p. Con este fin, sean

$$p_1, p_2, \ldots, p_n$$

todos los primos diferentes menores o iguales que p. Considere el entero

$$m = p_1 p_2 \cdots p_n + 1.$$

Observe que cuando m se divide entre p_i, el residuo es 1:

$$m = p_i q + 1, \qquad q = p_1 p_2 \cdots p_{i-1} p_{i+1} \cdots p_n.$$

Por lo tanto, para toda $i = 1$ a n, p_i *no* divide a m. Sea p' un factor primo de m (m puede o no ser primo, vea el ejercicio 33). Entonces p' no es igual a ninguna p_i, $i = 1$ a n. Como $p_1, p_2, \ldots, p_n$ es una lista de todos los primos menores o iguales que p, debemos tener $p' > p$. Esto completa la prueba.

Ejemplo 5.1.13 ▶

Se ilustra cómo la demostración del Teorema 5.1.12 produce un primo mayor que 11. Se listan los primos menores o iguales que 11:

$$2, 3, 5, 7, 11,$$

Sea

$$m = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 + 1 = 2311.$$

Usando el algoritmo 5.1.8, se encuentra que 2311 es primo. Se ha encontrado un número primo, a saber 2311, mayor que 2, 3, 5, 7, 11. (Si hubiera resultado que 2311 no es primo, el algoritmo 5.1.8 habría encontrado un factor de 2311, que necesariamente sería mayor que cada uno de 2, 3, 5, 7, 11). ◀

WWW

El **máximo común divisor** de dos enteros m y n (que no son cero) es el entero positivo más grande que divide a los dos: m y n. Por ejemplo, el máximo común divisor de 4 y 6 es 2 y el máximo común divisor de 3 y 8 es 1. Se usa el concepto de máximo común divisor cuando se verifica si una fracción m/n, donde m y n son enteros, está simplificada. Si el máximo común divisor de m y n es 1, m/n está simplificada; de otra manera, es posible reducir m/n. Por ejemplo, $4/6$ no está reducida porque el máximo común divisor de 4 y 6 es 2, no 1. (Podemos dividir 4 y 6 entre 2). La fracción $3/8$ está simplificada porque el máximo común divisor de 3 y 8 es 1.

Definición 5.1.14 ▶

Sean m y n enteros diferentes de cero. Un *divisor común* de m y n es un entero que divide tanto a m como a n. El *máximo común divisor*, escrito

$$\text{mcd}(m, n),$$

es el divisor común de m y n más grande. ◀

Ejemplo 5.1.15 ▶

Los divisores positivos de 30 son

$$1, \quad 2, \quad 3, \quad 5, \quad 6, \quad 10, \quad 15, \quad 30.$$

y los divisores positivos de 105 son

$$1, \quad 3, \quad 5, \quad 7, \quad 15, \quad 21, \quad 35, \quad 105;$$

entonces los divisores comunes de 30 y 105 son

$$1, \quad 3, \quad 5, \quad 15.$$

Se concluye que el máximo común divisor de 30 y 105, mcd(30, 105), es 15. ◀

También se puede encontrar el máximo común divisor de dos enteros m y n observando con cuidado sus factorizaciones primas. Se ilustra esto con un ejemplo y después se explica la técnica con detalle.

Ejemplo 5.1.16 ▶

El máximo común divisor de 30 y 105 se encuentra observando sus factorizaciones primas

$$30 = 2 \cdot 3 \cdot 5 \qquad 105 = 3 \cdot 5 \cdot 7.$$

Observe que 3 es un divisor común de 30 y 105 ya que aparece en la factorización prima de ambos números. Por la misma razón, 5 también es un divisor común de 30 y 105. Además, $3 \cdot 5 = 15$ también es un divisor común de 30 y 105. Puesto que no hay un producto mayor de primos que sea común a los dos, 30 y 105, se concluye que 15 es el máximo común divisor de 30 y 105. ◀

El método del ejemplo 5.1.16 se establece como el Teorema 5.1.17.

Teorema 5.1.17

Sean m y n enteros, $m > 1$, $n > 1$, con factorizaciones primas

$$m = p_1^{a_1} p_2^{a_2} \cdots p_n^{a_n}$$

y

$$n = p_1^{b_1} p_2^{b_2} \cdots p_n^{b_n}.$$

(Si el primo p_i no es un factor de m, se hace $a_i = 0$. De manera similar, si el primo p_i no es un factor de n, se hace $b_i = 0$.) Entonces

$$\text{mcd}(m, n) = p_1^{\text{mín}(a_1,b_1)} p_2^{\text{mín}(a_2,b_2)} \cdots p_n^{\text{mín}(a_n,b_n)}.$$

Ejemplo 5.1.18 ▶

Usando la notación del teorema 5.1.17, se tiene

$$82320 = 2^4 \cdot 3^1 \cdot 5^1 \cdot 7^3 \cdot 11^0$$

y

$$950796 = 2^2 \cdot 3^2 \cdot 5^0 \cdot 7^4 \cdot 11^1.$$

Por el teorema 5.1.17,

$$\begin{aligned} \text{mcd}(82320, 950796) &= 2^{\text{mín}(4,\,2)} \cdot 3^{\text{mín}(1,\,2)} \cdot 5^{\text{mín}(1,\,0)} \cdot 7^{\text{mín}(3,\,4)} \cdot 11^{\text{mín}(0,1)} \\ &= 2^2 \cdot 3^1 \cdot 5^0 \cdot 7^3 \cdot 11^0 \\ &= 4116. \end{aligned}$$

◀

Ni el método de la "lista de todos los divisores" del ejemplo 5.1.15 ni el de los factores primos del ejemplo 5.1.18 es eficiente para encontrar el máximo común divisor. El problema es que ambos métodos requieren encontrar los factores primos de los números implicados y no se conoce un algoritmo eficiente para calcular estos factores primos. Sin embargo, en la sección 5.3 se presentará el algoritmo euclidiano, que proporciona una manera eficiente de calcular el máximo común divisor.

Un compañero del máximo común divisor es el mínimo común múltiplo.

Definición 5.1.19 ▶

Sean m y n enteros positivos. Un *múltiplo común* de m y n es un entero que es divisible tanto entre m como entre n. El *mínimo común múltiplo*, escrito

$$\text{mcm}(m, n),$$

es el múltiplo común positivo más pequeño de m y n. ◀

Ejemplo 5.1.20 ▶

El mínimo común múltiplo de 30 y 105, mcm(30, 105), es 210 porque 210 es divisible entre los dos (30 y 105) y, por inspección, ningún entero positivo menor que 210 es divisible por ambos, 30 y 105. ◀

Ejemplo 5.1.21 ▶

Podemos encontrar el mínimo común múltiplo de 30 y 105 observando sus factorizaciones primas

$$30 = 2 \cdot 3 \cdot 5 \qquad 105 = 3 \cdot 5 \cdot 7.$$

La factorización prima de mcm(30, 105) debe contener a 2, 3 y 5 como factores [para que 30 divida a mcm(30, 105)]. También debe contener a 3, 5 y 7 [para que 105 divida a mcm(30, 105)]. El número más pequeño con esta propiedad es

$$2 \cdot 3 \cdot 5 \cdot 7 = 210.$$

Por lo tanto, mcm(30, 105) = 210. ◀

Se establece el método del ejemplo 5.1.21 como el teorema 5.1.22.

Teorema 5.1.22

Sean m y n enteros, $m > 1$, $n > 1$, con factorizaciones primas

$$m = p_1^{a_1} p_2^{a_2} \cdots p_n^{a_n}$$

y

$$n = p_1^{b_1} p_2^{b_2} \cdots p_n^{b_n}.$$

(Si el primo p_i no es un factor de m, se deja $a_i = 0$. De manera similar, si el primo p_i no es un factor de n, se deja $b_i = 0$). Entonces

$$\text{mcm}(m, n) = p_1^{\text{máx}(a_1, b_1)} p_2^{\text{máx}(a_2, b_2)} \qquad p_n^{\text{máx}(a_n, b_n)}.$$

Ejemplo 5.1.23 ▶

Usando la notación del teorema 5.1.22, se tiene

$$82320 = 2^4 \cdot 3^1 \cdot 5^1 \cdot 7^3 \cdot 11^0$$

y

$$950796 = 2^2 \cdot 3^2 \cdot 5^0 \cdot 7^4 \cdot 11^1$$

Por el teorema 5.1.22,

$$\begin{aligned} \text{mcd}(82320, 950796) &= 2^{\text{máx}(4, 2)} \cdot 3^{\text{máx}(1, 2)} \cdot 5^{\text{máx}(1, 0)} \cdot 7^{\text{máx}(3, 4)} \cdot 11^{\text{máx}(0, 1)} \\ &= 2^4 \cdot 3^2 \cdot 5^1 \cdot 7^4 \cdot 11^1 \\ &= 19015920 \end{aligned}$$

◀

Ejemplo 5.1.24 ▶

En el ejemplo 5.1.15 se encontró que

$$\text{mcd}(30, 105) = 15,$$

y en el ejemplo 5.1.21 se encontró que

$$\text{mcm}(30, 105) = 210.$$

Observe que el producto de mcd y mcm es igual al producto del par de números; es decir,

$$\text{mcd}(30, 105) \cdot \text{mcm}(30, 105) = 15 \cdot 210 = 3150 = 30 \cdot 105.$$

Esta fórmula se cumple para cualquier par de números como se demostrará en el teorema 5.1.25. ◀

Teorema 5.1.25

Para cualesquiera enteros positivos m y n,

$$\text{mcd}(m, n) \cdot \text{mcm}(m, n) = mn.$$

Demostración Si $m = 1$, entonces $\text{mcd}(m, n) = 1$ y $\text{mcm}(m, n) = n$, así

$$\text{mcd}(m, n) \cdot \text{mcm}(m, n) = 1 \cdot n = mn.$$

De modo similar, si $n = 1$, entonces $\text{mcd}(m, n) = 1$ y $\text{mcm}(m, n) = m$, entonces

$$\text{mcd}(m, n) \cdot \text{mcm}(m, n) = 1 \cdot m = mn.$$

Por lo que podemos suponer que $m > 1$ y $n > 1$.

La prueba combina las fórmulas para el mcd (teorema 5.1.17) y el mcm (teorema 5.1.22) (que requieren que $m > 1$ y $n > 1$), con el hecho de que

$$\text{mín}(x, y) + \text{máx}(x, y) = x + y \quad \text{para toda } x \text{ y } y.$$

Esta última fórmula es verdadera porque uno de $\{\text{mín}(x, y), \text{máx}(x, y)\}$ es igual a x y el otro a y. Ahora se une todo esto para producir la demostración.

Se escriben las factorizaciones primas de m y n como

$$m = p_1^{a_1} p_2^{a_2} \cdots p_n^{a_n}$$

y

$$n = p_1^{b_1} p_2^{b_2} \cdots p_n^{b_n}.$$

(Si el primo p_i no es un factor de m, se hace $a_i = 0$. De la misma manera, si el primo p_i no es un factor de n, se hacer $b_i = 0$). Por el teorema 5.1.17,

$$\text{mcd}(m, n) = p_1^{\text{mín}(a_1,b_1)} \cdots p_n^{\text{mín}(a_n,b_n)},$$

y por el teorema 5.1.22,

$$\text{mcm}(m, n) = p_1^{\text{máx}(a_1,b_1)} \cdots p_n^{\text{máx}(a_n,b_n)}.$$

Por lo tanto,

$$\begin{aligned}\text{mcd}(m, n) \cdot \text{mcm}(m, n) &= [p_1^{\text{mín}(a_1,b_1)} \cdots p_n^{\text{mín}(a_n,b_n)}] \cdot \\ &\quad [p_1^{\text{máx}(a_1,b_1)} \cdots p_n^{\text{máx}(a_n,b_n)}] \\ &= p_1^{\text{mín}(a_1,b_1)+\text{máx}(a_1,b_1)} \cdots p_n^{\text{mín}(a_n,b_n)+\text{máx}(a_n,b_n)} \\ &= p_1^{a_1+b_1} \cdots p_n^{a_n+b_n} \\ &= [p_1^{a_1} \cdots p_n^{a_n}][p_1^{b_1} \cdots p_n^{b_n}] = mn.\end{aligned}$$

Si se tiene un algoritmo para calcular el máximo común divisor, es posible calcular el mínimo común múltiplo usando el teorema 5.1.25;

$$\text{mcm}(m, n) = \frac{mn}{\text{mcd}(m, n)}.$$

En particular, si se tiene un algoritmo *eficiente* para calcular el máximo común divisor, también se puede calcular de manera eficiente el mínimo común múltiplo.

Sugerencias para resolver problemas

La manera directa para determinar si un entero $n > 1$ es primo es probar si cualquiera de los números $2, 3, \ldots, \lfloor \sqrt{n} \rfloor$ divide a n. Mientras que esta técnica llega a ser muy tardada cuando n crece, es suficiente para valores de n relativamente pequeños. Esta técnica se puede iterar para encontrar la factorización prima de n, de nuevo para valores de n relativamente pequeños.

Se presentaron dos formas de encontrar el máximo común divisor de a y b. La primera fue hacer una lista de todos los divisores positivos de a y todos los divisores positivos de b y después, entre todos los divisores comunes, elegir el mayor. Esta técnica es tardada y se mostró, en principio, para ilustrar exactamente qué significan un divisor común y el máximo común divisor.

La segunda técnica consistió en comparar las factorizaciones primas de a y b y si p^i aparece en a y p^j aparece en b, se incluye $p^{\text{mín}(i,j)}$ en la factorización prima del máximo común divisor. Esta técnica trabaja bien si los números a y b son relativamente pequeños de manera que se puedan encontrar las factorizaciones primas de cada uno, o si estas factorizaciones primas están dadas. En la sección 5.3 se presenta el algoritmo euclidiano que encuentra de manera eficiente el máximo común divisor incluso para valores grandes de a y b.

Si se calcula el mcd(a, b), se puede calcular de inmediato el mínimo común múltiplo usando la fórmula

$$\text{mcm}(a, b) = \frac{ab}{\text{mcd}(a, b)}.$$

También es posible calcular el mínimo común múltiplo comparando las factorizaciones primas de a y b y si p^i aparece en a y p^j aparece en b, se incluye $p^{\text{máx}(i,j)}$ en la factorización prima del mínimo común múltiplo.

Sección de ejercicios de repaso

1. Defina *d divide a n*.
2. Defina *d es divisor de n*.
3. Defina *cociente*.
4. Defina *n es primo*.
5. Defina *n es compuesto*.
6. Explique por qué, al probar si un entero $n > 1$ es primo observando sus divisores, sólo se necesita verificar si algún número de 2 a $\lfloor \sqrt{n} \rfloor$ divide a n.
7. Explique por qué el algoritmo 5.1.8 no se considera un algoritmo de tiempo polinomial.
8. ¿Qué dice el teorema fundamental de la aritmética?
9. Demuestre que el número de primos es infinito.
10. ¿Qué es un divisor común?
11. ¿Qué es el máximo común divisor?
12. Explique cómo calcular el máximo común divisor de m y n, ambos diferentes de cero, a partir de su factorización prima.
13. ¿Qué es un múltiplo común?
14. ¿Qué es el mínimo común múltiplo?
15. Explique cómo calcular el mínimo común múltiplo de los enteros positivos m y n, a partir de sus factorizaciones primas.
16. ¿Cómo se relacionan el máximo común divisor y el mínimo común múltiplo?

Ejercicios

En los ejercicios 1 al 8, siga el algoritmo 5.1.8 para la entrada indicada.

1. $n = 9$
2. $n = 47$
3. $n = 209$
4. $n = 637$
5. $n = 1007$
6. $n = 4141$
7. $n = 3738$
8. $n = 1050703$
9. ¿Cuáles de los enteros en los ejercicios 1 al 8 son primos?
10. Encuentre la factorización prima de cada entero en los ejercicios 1 al 8.
11. Encuentre la factorización prima de 11!.

Encuentre el máximo común divisor de cada par de enteros de los ejercicios 12 al 24.

12. 0, 17
13. 5, 25
14. 60, 90
15. 110, 273
16. 220, 1400
17. 315, 825
18. 20, 40
19. 331, 993
20. 2091, 4807
21. $13, 13^2$
22. $15, 15^9$
23. $3^2 \cdot 7^3 \cdot 11 \cdot 2^3 \cdot 5 \cdot 7$
24. $3^2 \cdot 7^3 \cdot 11 \cdot 3^2 \cdot 7^3 \cdot 11$
25. Encuentre el mínimo común múltiplo de cada par de enteros de los ejercicios 13 al 24.
26. Para cada par de enteros de los ejercicios 13 al 24, verifique que $\text{mcd}(m, n) \cdot \text{mcm}(m, n) = mn$.
27. Sean m, n y d enteros. Demuestre que si $d \mid m$ y $d \mid n$, entonces $d \mid (m - n)$.
28. Sean m, n y d enteros. Demuestre que si $d \mid m$, entonces $d \mid mn$.
29. Sean m, n, d_1 y d_2 enteros. Demuestre que si $d_1 \mid m$ y $d_2 \mid n$, entonces $d_1 d_2 \mid mn$.
30. Sean n, c y d enteros. Demuestre que si $dc \mid nc$, entonces $d \mid n$.
31. Sean a, b y c enteros. Demuestre que si $a \mid b$ y $b \mid c$, entonces $a \mid c$.
32. Sugiera formas de hacer el algoritmo 5.1.8 más eficiente.
33. Dé un ejemplo de primos consecutivos $p_1 = 2, p_2, \ldots, p_n$ donde

$$p_1 p_2 \cdots p_n + 1$$

no es primo.

5.2 ➜ Representaciones de enteros y algoritmos enteros

WWW

Un **bit** es un dígito binario, es decir, un 0 o un 1. En una computadora digital, los datos y las instrucciones se codifican como bits. (El término *digital* se refiere al uso de los dígitos 0 y 1). La tecnología determina cómo se representan físicamente los bits dentro del sistema de la computadora. El hardware actual se apoya en el estado de un circuito electrónico para representar un bit. El circuito debe ser capaz de estar en dos estados: uno que representa a 1, y el otro a 0. En esta sección se estudia el **sistema de números binario**, que representa enteros que usan bits, y el **sistema numérico hexadecimal**, que representa enteros que usan 16 símbolos. El **sistema de números octal**, que representa enteros que usan 8 símbolos, se estudiará antes del ejercicio 42.

En el sistema numérico decimal, para representar los enteros se usan los 10 símbolos 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9. Al representar un entero, la posición del símbolo es significativa; leyendo desde la derecha, el primer símbolo representa al número de unidades, el siguiente símbolo el número de decenas, el siguiente el número de centenas, y así sucesivamente. Por ejemplo,

$$3854 = 3 \cdot 10^3 + 8 \cdot 10^2 + 5 \cdot 10^1 + 4 \cdot 10^0$$

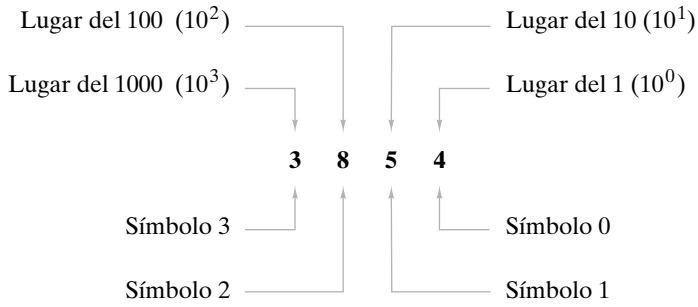


Figura 5.2.1 Sistema numérico decimal.

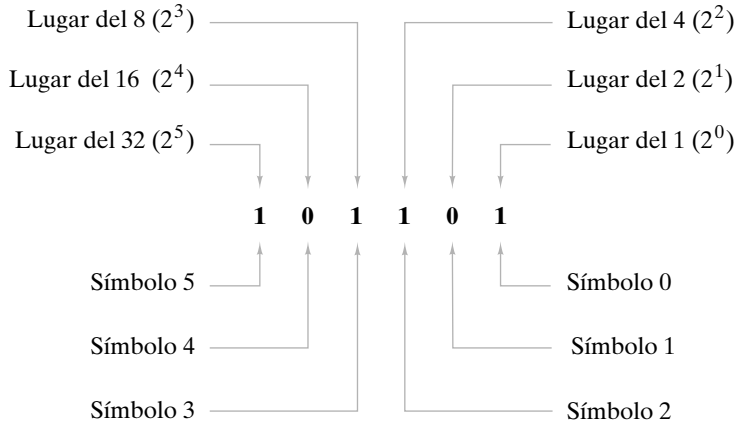


Figura 5.2.2 Sistema numérico binario.

(vea la figura 5.2.1). En general, el símbolo en la posición n (donde el símbolo en la extrema derecha está en la posición 0) representa el número de 10^n que hay. Como $10^0 = 1$, el símbolo en la posición 0 representa el número de 10^0 o decenas; como $10^1 = 10$, el símbolo en la posición 1 representa el número de 10^1 o decenas; como $10^2 = 100$, el símbolo en la posición 2 representa el número de 10^2 o centenas, y así sucesivamente. El valor en el que está basado el sistema (10 en este caso) recibe el nombre de **base** del sistema numérico.

En el sistema numérico binario (base 2), para representar enteros se necesitan sólo dos símbolos, 0 y 1. Para representar un entero, leyendo de derecha a izquierda, el primer símbolo representa el número de unos, el siguiente símbolo el número de números dos, el siguiente el número de cuatros, el siguiente el número de ochos, etcétera. Por ejemplo, en base 2,

$$101101 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

(vea la figura 5.2.2). En general, el símbolo en la posición n (con la posición 0 en la extrema derecha) representa el número de 2^n. Como $2^0 = 1$, el símbolo en la posición 0 es el número de 2^0 o unos; como $2^1 = 2$, el símbolo en la posición 1 representa el número de 2^1 o números 2; como $2^2 = 4$, el símbolo en la posición 2 es el número de 2^2 o cuatros; etcétera.

Ejemplo 5.2.1 ▶

Representación de enteros en la computadora

Los sistemas de cómputo representan los enteros en binario. Se calcula el número de bits necesarios para representar un entero positivo n. Observe que si la representación binaria del entero positivo n es

$$n = 1 \cdot 2^k + b_{k-1}2^{k-1} + \cdots + b_0 2^0,$$

entonces

$$2^k \leq n$$

y

$$\begin{aligned} n &= 1 \cdot 2^k + b_{k-1}2^{k-1} + \cdots + b_0 2^0 \\ &\leq 1 \cdot 2^k + 1 \cdot 2^{k-1} + \cdots + 1 \cdot 2^0 = 2^{k+1} - 1 < 2^{k+1}. \end{aligned}$$

(La última igualdad se deriva de la fórmula para la suma geométrica; vea el ejemplo 1.7.4). Como $2^k \leq n$, tomando logaritmos se obtiene

$$k \leq \lg n.$$

Como $n < 2^{k+1}$, de nuevo tomando logaritmos se obtiene

$$\lg n < k + 1.$$

Combinando estas desigualdades, se tiene

$$k + 1 \leq 1 + \lg n < k + 2. \qquad \textbf{(5.2.1)}$$

Por lo tanto, $k + 1 = \lfloor 1 + \lg n \rfloor$, que es el número de bits requerido para representar n.

El tiempo del peor caso para el algoritmo 5.1.8, que determina si el entero n es primo, es $\Theta(\sqrt{n})$. Por la ecuación (5.2.1), el tamaño s $(= k + 1)$ de la entrada n satisface $s \leq 1 + \lg n \leq 2 \lg$ para toda $n \geq 2$. Por lo tanto,

$$\lg n \geq s/2 \qquad \text{para toda } n \geq 2,$$

que es equivalente a

$$(1/2) \lg n \geq s/4 \quad \text{para toda } n \geq 2,$$

que a su vez es equivalente a

$$\lg n^{1/2} \geq s/4 \qquad \text{para toda } n \geq 2.$$

Elevando al cuadrado se tiene

$$\sqrt{n} \geq c^s,$$

para toda $n \geq 2$, donde $c = 2^{1/4}$. Por lo tanto, cuando n se da como entrada para el algoritmo 5.1.8, el tiempo del peor caso es al menos $C\sqrt{n}$ para alguna constante C, que es al menos Cc^s. Así, en el peor caso, el algoritmo 5.1.8 corre en *tiempo exponencial* respecto al tamaño de la entrada s. Se dice que el algoritmo 5.1.8 *no* es un algoritmo de tiempo polinomial. ◀

Si se desconoce qué sistema numérico se está usando, una representación resulta ambigua; por ejemplo, 101101 representa un número en decimal y otro muy diferente en binario. Con frecuencia, el contexto aclara qué sistema numérico se está usando; pero cuando se requiere que no exista duda, se da un subíndice al número para especificar la base: el subíndice 10 denota el sistema decimal y el subíndice 2 el binario. Por ejemplo, el número binario 101101 se escribe 101101_2.

Ejemplo 5.2.2 ▶

De binario a decimal

El número binario 101101_2 representa el número que tiene un 1, ningún 2, un 4, un 8, ningún 16 y un 32 (vea la figura 5.2.2). Esta representación se expresa como

$$101101_2 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0.$$

Calculando el lado derecho en decimal, se encuentra que

$$\begin{aligned} 101101_2 &= 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 \\ &= 32 + 8 + 4 + 1 = 45_{10}. \end{aligned}$$

◀

El método del ejemplo 5.2.2 se puede convertir en un algoritmo. Se generaliza al permitir una base arbitraria b.

Algoritmo 5.2.3

Conversión de un entero base *b* en decimal

Este algoritmo regresa el valor decimal del entero en base b $c_n c_{n-1} \cdots c_1 c_0$.

```
Entrada:  c, n, b
Salida:   val_dec
base_b_a_dec(c, n, b) {
    val_dec = 0
    potencia = 0
    for i = 0 to n {
        val_dec = val_dec + c_i*potencia
        potencia = potencia*b
    }
    return val_dec
}
```

El algoritmo 5.2.3 corre en tiempo $\Theta(n)$.

Ejemplo 5.2.4 ▶

Se muestra la manera en que el algoritmo 5.2.3 convierte el número binario 1101 en decimal. Se tiene $n = 3$, $b = 2$ y

$$c_3 = 1, \quad c_2 = 1, \qquad c_1 = 0, \quad c_0 = 1.$$

Primero *val_dec* se hace igual a 0 y *potencia* igual a 1. Después entramos al ciclo.

Como $i = 0$ y *potencia* $= 1$,

$$c_i * potencia = 1 * 1 = 1.$$

Entonces *val_dec* se convierte en 1. Al ejecutar

$$potencia = potencia * b$$

la *potencia* queda igual a 2. Se regresa al principio del ciclo "for".

Como $i = 1$ y *potencia* $= 2$,

$$c_i * potencia = 0 * 2 = 0.$$

Así, *val_dec* se queda en 1. Al ejecutar

$$potencia = potencia * b$$

se hace *potencia* igual a 4. Se regresa al principio del ciclo "for".

Como $i = 2$ y *potencia* $= 4$,

$$c_i * potencia = 1 * 4 = 4.$$

Entonces *val_dec* se convierte en 5. Al ejecutar

$$potencia = potencia * b$$

se establece *potencia* igual a 8. Se regresa al principio del ciclo "for".

Como $i = 3$ y *potencia* $= 8$,

$$c_i * potencia = 1 * 8 = 8.$$

Así, *val_dec* se convierte en 13. Al ejecutar

$$potencia = potencia * b$$

se hace *potencia* igual a 16. El ciclo "for" termina y el algoritmo regresa 13, el valor decimal del número binario 1101. ◀

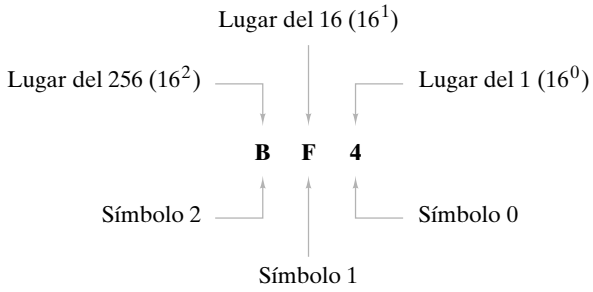


Figura 5.2.3 Sistema numérico hexadecimal.

Otras bases importantes para los sistemas numéricos en las ciencias de la computación son la base 8 u **octal** y la base 16 o **hexadecimal** (en ocasiones acortada a **hex**). Se analizará el sistema hexadecimal y se dejará el octal a los ejercicios (vea los ejercicios 42 al 47).

En el sistema numérico hexadecimal, para representar enteros se usan los símbolos 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F. Los símbolos comprendidos entre A y F se interpretan como los decimales 10 al 15. (En general, en el sistema numérico base N, se requieren N símbolos diferentes que son 0, 1, 2, . . . , $N - 1$). Al representar un entero, si se lee de derecha a izquierda, el primer símbolo representa el número de unos, el siguiente símbolo el número de 16, el siguiente el número de 16^2, y así sucesivamente. Por ejemplo, en base 16,

$$\text{B4F} = 11 \cdot 16^2 + 4 \cdot 16^1 + 15 \cdot 16^0$$

(vea la figura 5.2.3). En general, el símbolo en la posición n (donde la posición 0 es la extrema derecha) representa el número de números 16.

Ejemplo 5.2.5 ▶

De hexadecimal a decimal

Convierta el número hexadecimal B4F en decimal.

Se obtiene

$$\begin{aligned}\text{B4F}_{16} &= 11 \cdot 16^2 + 4 \cdot 16^1 + 15 \cdot 16^0 \\ &= 11 \cdot 256 + 4 \cdot 16 + 15 = 2816 + 64 + 15 = 2895_{10}.\end{aligned}$$

◀

El algoritmo 5.2.3 muestra cómo convertir un entero base b en decimal. Considere el problema inverso: convertir un número decimal a la base b. Suponga, por ejemplo, que se desea convertir el número decimal 91 en binario. Si se divide 91 entre 2, se obtiene

$$\begin{array}{r} 45 \\ 2\overline{)91} \\ 8 \\ 11 \\ 10 \\ \hline 1 \end{array}$$

Este cálculo muestra que

$$91 = 2 \cdot 45 + 1 \qquad \textbf{(5.2.2)}$$

Se comienza por expresar 91 en potencias de 2. Si después se divide 45 entre 2, se encuentra que

$$45 = 2 \cdot 22 + 1. \qquad \textbf{(5.2.3)}$$

Sustituyendo esta expresión para 45 en (5.2.2) se obtiene

$$\begin{aligned}91 &= 2 \cdot 45 + 1 \\ &= 2 \cdot (2 \cdot 22 + 1) + 1 \\ &= 2^2 \cdot 22 + 2 + 1.\end{aligned} \qquad \textbf{(5.2.4)}$$

Si ahora se divide 22 entre 2, se encuentra

$$22 = 2 \cdot 11$$

Al sustituir esta expresión para 22 en (5.2.4), se obtiene

$$\begin{aligned} 91 &= 2^2 \cdot 22 + 2 + 1 \\ &= 2^2 \cdot (2 \cdot 11) + 2 + 1 \\ &= 2^3 \cdot 11 + 2 + 1. \end{aligned} \tag{5.2.5}$$

Si luego se divide 11 entre 2, se encuentra

$$11 = 2 \cdot 5 + 1$$

Sustituyendo esta expresión para 11 en (5.2.5) se obtiene

$$91 = 2^4 \cdot 5 + 2^3 + 2 + 1. \tag{5.2.6}$$

Si ahora se divide 5 entre 2, se encuentra

$$5 = 2 \cdot 2 + 1.$$

Al sustituir esta expresión para 5 en (5.2.6) se obtiene

$$\begin{aligned} 91 &= 2^5 \cdot 2 + 2^4 + 2^3 + 2 + 1 \\ &= 2^6 + 2^4 + 2^3 + 2 + 1 \\ &= 1011011_2. \end{aligned}$$

El cálculo anterior muestra que los *residuos*, al dividir N entre 2 repetidas veces, dan los bits en la representación binaria de N. La primera división entre 2 en (5.2.2) da el bit de 1; la segunda división entre 2 en (5.2.3) da el bit de 2; etcétera. Se ilustrará otro ejemplo.

Ejemplo 5.2.6 ▶

Decimal a binario

Escriba el número decimal 130 en binario.

El cálculo muestra las divisiones sucesivas entre 2 con los residuos registrados a la derecha.

$2\overline{\vert 130}$	residuo $= 0$	bit de 1
$2\overline{\vert 65}$	residuo $= 1$	bit de 2
$2\overline{\vert 32}$	residuo $= 0$	bit de 4
$2\overline{\vert 16}$	residuo $= 0$	bit de 8
$2\overline{\vert 8}$	residuo $= 0$	bit de 16
$2\overline{\vert 4}$	residuo $= 0$	bit de 32
$2\overline{\vert 2}$	residuo $= 0$	bit de 64
$2\overline{\vert 1}$	residuo $= 1$	bit de 128
0		

El proceso se detiene cuando el cociente es 0. Recordando que el primer residuo da el número de unos, el segundo el número de números 2, etcétera, se obtiene

$$130_{10} = 10000010_2. \quad \blacktriangleleft$$

El método del ejemplo 5.2.6 se puede convertir en un algoritmo. Se generaliza permitiendo una base arbitraria b.

Algoritmo 5.2.7

Conversión de un entero decimal a la base b

Este algoritmo convierte el entero positivo m en un entero base b $c_n c_{n-1} \cdots c_1 c_0$. La variable n se usa como índice en la sucesión c. El valor de m mod b es el residuo cuando m se divide entre b. El valor de $\lfloor m/b \rfloor$ es el cociente cuando m se divide entre b.

```
Entrada: m, b
Salida: c, n
dec_a_base_b(m, b, c, n) {
    n = –1
    while(m > 0){
        n = n + 1
        c_n = m mod b
        m = ⌊m/b⌋
    }
}
```

Igual que un entero binario m tiene $\lfloor 1 + \lg m \rfloor$ bits, un entero base b tiene $\lfloor 1 + \log_b m \rfloor$ dígitos (vea el ejercicio 55). Entonces el algoritmo corre en el tiempo $\Theta(\log_b m)$.

Ejemplo 5.2.8 ▶

Se muestra cómo el algoritmo 5.2.7 convierte el número decimal $m = 11$ a binario.

El algoritmo primero establece n igual a –1. La primera vez que llegamos al ciclo "while", $m = 11$ y la condición $m > 0$ es verdadera; entonces se ejecuta el cuerpo del ciclo "while". La variable n se incrementa y ahora es 0. Como m mod $b = 11$ mod $2 = 1$, c_0 se hace 1. Como $\lfloor m/b \rfloor = \lfloor 11/2 \rfloor = 5$, m se hace 5. Se regresa al principio del ciclo "while".

Como $m = 5$, la condición $m > 0$ es verdadera; se ejecuta el cuerpo del ciclo. La variable n se incrementa y ahora es 1. Como m mod $b = 5$ mod $2 = 1$, c_1 se iguala a 1. Como $\lfloor m/b \rfloor = \lfloor 5/2 \rfloor = 2$, m se hace 2. Se regresa al principio del ciclo "while".

Como $m = 2$, la condición $m > 0$ es verdadera; se ejecuta el cuerpo del ciclo. La variable n se incrementa y ahora es 2. Como m mod $b = 2$ mod $2 = 0$, c_2 se iguala a 0. Como $\lfloor m/b \rfloor = \lfloor 2/2 \rfloor = 1$, m se hace 1. Se regresa al principio del ciclo "while".

Como $m = 1$, la condición $m > 0$ es verdadera; se ejecuta el cuerpo del ciclo. La variable n se incrementa y ahora es 3. Como m mod $b = 1$ mod $2 = 1$, c_3 se iguala a 1. Como $\lfloor m/b \rfloor = \lfloor 1/2 \rfloor = 0$, m se hace 0. Se regresa al principio del ciclo "while".

Como $m = 0$, el algoritmo termina. El valor 11 se ha convertido en el número binario

$$c_3 c_2 c_1 c_0 = 1011. \quad \blacktriangleleft$$

Ejemplo 5.2.9 ▶

Decimal a hexadecimal

Convierta el número decimal 20385 en hexadecimal.

El cálculo muestra las divisiones sucesivas entre 16 con los residuos registrados a la derecha.

$16\overline{)20385}$	residuo = 1	lugar de 1
$16\overline{)1274}$	residuo = 10	lugar de 16
$16\overline{)79}$	residuo = 15	lugar de 16^2
$16\overline{)4}$	residuo = 4	lugar de 16^3
0		

Nos detenemos cuando el cociente es 0. El primer residuo da el número de unos, el segundo el número de números 16, y así sucesivamente; entonces se obtiene

$$20385_{10} = 4\text{FA}1_{16}. \quad \blacktriangleleft$$

Después, la atención se centra en la suma de número en bases arbitrarias. El mismo método que se usa para sumar números decimales se emplea para sumar números binarios; sin embargo, debe sustituirse la tabla de suma decimal con la tabla de suma binaria.

+	0	1
0	0	1
1	1	10

(En decimal, $1 + 1 = 2$, y $2_{10} = 10_2$; entonces, en binario, $1 + 1 = 10$).

Ejemplo 5.2.10 ▶

Suma binaria

Sume los números binarios 10011011 y 1011011.

El problema se escribe como

```
  10011011
+  1011011
```

Igual que en la suma decimal, se comienza por la derecha, sumando 1 y 1. Esta suma es 10_2; entonces escribimos 0 y llevamos 1. En este punto el cálculo es

```
      1
  10011011
+  1011011
         0
```

Después sumamos 1 y 1 y 1, que es 11_2. Escribimos 1 y llevamos 1. En este punto el cálculo es

```
      1
  10011011
+  1011011
        10
```

Al continuar de esta manera, se obtiene

```
  10011011
+  1011011
  11110110
```

◀

Ejemplo 5.2.11 ▶

El problema de la suma de ejemplo 5.2.10, en decimal, es

```
  155
+  91
  246
```

◀

El método del ejemplo 5.2.10 se puede convertir en un algoritmo. Si los números que se van a sumar son $b_n b_{n-1} \cdots b_1 b_0$ y $b'_n b'_{n-1} \cdots b'_1 b'_0$, en la iteración $i > 0$ el algoritmo suma b_i, b'_i, y el bit que se acarrea de la iteración anterior. Al sumar tres bits, digamos B_1, B_2 y B_3, se obtiene un número binario de dos bits, digamos cb. Por ejemplo, si se calcula $1 + 0 + 1$, el resultado es 10_2; en nuestra notación, $c = 1$ y $b = 0$. Al revisar los diferentes casos, se verifica que podemos calcular la suma binaria $B_1 + B_2 + B_3$ calculando primero la suma en decimal y después recuperando c y b de las fórmulas

$$b = (B_1 + B_2 + B_3) \bmod 2, \qquad c = \lfloor (B_1 + B_2 + B_3)/2 \rfloor.$$

Algoritmo 5.2.12

Suma de números binarios

Este algoritmo suma los números binarios $b_n b_{n-1} \cdots b_1 b_0$ y $b'_n b'_{n-1} \cdots b'_1 b'_0$ y guarda la suma en $s_{n+1} s_n s_{n-1} \cdots s_1 s_0$.

Entrada: b, b', n

Salida: s

suma_binaria(b, b', n, s) {
  $lleva = 0$
  for $i = 0$ to n {
    $s_i = (b_i + b'_i + lleva) \bmod 2$
    $lleva = \lfloor b_i + b'_i + lleva)/2 \rfloor$
  }
  $s_{n+1} = lleva$
}

El algoritmo 5.2.12 corre en un tiempo $\Theta(n)$.

El siguiente ejemplo muestra que se pueden sumar números hexadecimales de la misma manera que se suman los números decimales o los binarios.

Ejemplo 5.2.13 ▶

Suma hexadecimal

Sume los números hexadecimales 84F y 42EA.

El problema se escribe como

```
   84F
+  42EA
```

Se comienza por la columna de la derecha sumando A y F. Como F es 15_{10} y A es 10_{10}, F + A = $15_{10} + 10_{10} = 25_{10} = 19_{16}$. Se escribe 9 y llevamos 1:

```
    1
   84F
+  42EA
    9
```

Después se suman 1, 4 y E, para obtener 13_{16}. Se escribe 3 y llevamos 1:

```
    1
   84F
+  42EA
   39
```

Si continuamos de esta manera, se obtiene

```
   84F
+  42EA
  4B39
```

◀

Ejemplo 5.2.14 ▶

El problema de la suma del ejemplo 5.2.13, en decimal, es

```
   2127
+ 17130
  19257
```

◀

Se pueden multiplicar números binarios modificando el algoritmo estándar para la multiplicación de números decimales (vea los ejercicios 64 y 65).

Se concluye con el análisis de un algoritmo especial, que se necesitará en la sección 5.4, para calcular potencias mod z. Primero se estudia un algoritmo para calcular una po-

tencia a^n (sin incluir mod z). La manera directa de calcular esta potencia es multiplicar repetidas veces por a

$$\underbrace{a \cdot a \cdots a}_{n\ a\text{'s}},$$

que usa $n - 1$ multiplicaciones. Se logran mejores resultados si se **eleva al cuadrado repetidas veces**.

Como ejemplo concreto, considere calcular a^{29}. Primero se calcula $a^2 = a \cdot a$, que requiere una multiplicación. Después se calcula $a^4 = a^2 \cdot a^2$, que usa una multiplicación más. Luego se calcula $a^8 = a^4 \cdot a^4$, que usa una multiplicación adicional y después se calcula $a^{16} = a^8 \cdot a^8$, que requiere una multiplicación más. Hasta aquí se han empleado sólo 4 multiplicaciones. Observe que la expansión de 29 en potencias de 2, que es la expansión binaria, es

$$29 = 1 + 4 + 8 + 16,$$

vemos que se puede calcular a^{29} como

$$a^{29} = a^1 \cdot a^4 \cdot a^8 \cdot a^{16},$$

que usa 3 multiplicaciones adicionales con un total de 7 multiplicaciones. La técnica directa utiliza 28 multiplicaciones.

En el ejemplo 5.2.6, se vio que los residuos, cuando n se divide repetidas veces entre 2, dan la expansión binaria de n. Si el residuo es 1, se incluye la potencia de 2 correspondiente; de otra manera, no se incluye. Se puede formalizar la técnica de elevar al cuadrado repetidas veces si se determina al mismo tiempo la expansión binaria del exponente.

Ejemplo 5.2.15 ▶

La figura 5.2.4 muestra cómo se calcula a^{29} elevando al cuadrado una y otra vez. Inicialmente, x se hace igual a a, y n al valor del exponente, en este caso 29. Después se calcula n mod 2. Como este valor es 1, se sabe que $1 = 2^0$ está incluido en la expansión binaria de 29. Por lo tanto, a^1 se incluye en el producto. Se da seguimiento al producto parcial en el *resultado*; así, el *resultado* se hace igual a a. Luego se calcula el cociente cuando 29 se divide entre 2. El cociente 14 se vuelve el nuevo valor de n. Después se repite este proceso.

x	Valor *de n*	n mod 2	*Resultado*	*Cociente cuando n actual se divide entre 2*
a	29	1	a	14
a^2	14	0	Sin cambio	7
a^4	7	1	$a \cdot a^4 = a^5$	3
a^8	3	1	$a^5 \cdot a^8 = a^{13}$	1
a^{16}	1	1	$a^{13} \cdot a^{16} = a^{29}$	0

Figura 5.2.4 Cálculo de a^{29} elevando al cuadrado repetidas veces.

Se eleva x al cuadrado para obtener a^2. Después se calcula n mod 2. Como este valor es 0, se sabe que $2 = 2^1$ no se incluye en la expansión binaria de 29. Entonces a^2 no está incluida en el producto, y el *resultado* no cambia. Después se calcula el cociente cuando 14 se divide entre 2. El cociente 7 se convierte en el nuevo valor de n. Luego se repite el proceso.

Se eleva x al cuadrado para obtener a^4. Después se calcula n mod 2. Como este valor es 1, se sabe que $4 = 2^2$ está incluido en la expansión binaria de 29. Entonces a^4 se incluye en el producto. El *resultado* se convierte en a^5. Después se calcula el cociente al dividir 7 entre 2. El cociente 3 es ahora el nuevo valor de n. El proceso continúa hasta que n es 0. ◀

Se establece el método de elevar al cuadrado repetidas veces como el algoritmo 5.2.16.

Algoritmo 5.2.16

Elevar a un exponente elevando al cuadrado varias veces

Este algoritmo calcula a^n elevando al cuadrado repetidas veces. El algoritmo se explica en el ejemplo 5.2.15.

```
Entrada:    a, n
Salida:     a^n

exp_via_cuadrado_repetido(a, n) {
resultado = 1
    x = a
    while (n > 0) {
        if (n mod 2 == 1)
            resultado = resultado * x
        x = x * x
        n = ⌊n/2⌋
    }
    return resultado
}
```

El número de veces que se ejecuta el ciclo "while" está determinado por n. La variable n se divide entre 2 repetidas veces

$$n = \lfloor n/2 \rfloor$$

y cuando n es cero, el ciclo termina. El ejemplo 4.3.14 muestra que toma un tiempo $\Theta(\lg n)$ reducir n a 0 repitiendo la división entre 2. En el cuerpo del ciclo "while" se realizan a lo sumo dos multiplicaciones. Entonces, el número de multiplicaciones es, cuando mucho, $\Theta(\lg n)$, que es una mejora respecto al algoritmo directo que requiere $\Theta(n)$ multiplicaciones. El cuello de botella en el algoritmo 5.2.16 es el tamaño de los números implicados. El valor que regresa a^n requiere $\lg a^n = n \lg a$ bits en su representación. Así, sólo copiar el valor final en el *resultado* toma por lo menos un tiempo $\Omega(n \lg a)$, que es exponencial respecto al tamaño n (vea el ejemplo 5.2.1).

En la sección 5.4, será necesario calcular $a^n \bmod z$ para valores grandes de a y n. En este caso, a^n será enorme; entonces es impráctico calcular a^n y después calcular el residuo cuando a^n se divide entre z. Hay forma de obtener mejores resultados. La idea clave es calcular el residuo después de cada multiplicación y así mantener los números relativamente pequeños. La justificación de esta técnica se expone en el siguiente teorema.

Teorema 5.2.17

Si a, b y z son enteros positivos,

$$ab \bmod z = [(a \bmod z)(b \bmod z)] \bmod z.$$

Demostración Sea $w = ab \bmod z$, $x = a \bmod z$ y $y = b \bmod z$. Como w es el residuo cuando ab se divide entre z, por el teorema del cociente-residuo, existe q_1 tal que

$$ab = q_1 z + w.$$

Entonces

$$w = ab - q_1 z.$$

De manera similar, existen q_2 y q_3 tales que

$$a = q_2 z + x, \quad b = q_3 z + y.$$

Ahora

$$\begin{aligned} w &= ab - q_1 z \\ &= (q_2 z + x)(q_3 z + y) - q_1 z \\ &= (q_2 q_3 z + q_2 y + q_3 x - q_1) z + xy \\ &= qz + xy, \end{aligned}$$

donde $q = q_2 q_3 z + q_2 y + q_3 x - q_1$. Por lo tanto,

$$xy = -qz + w;$$

es decir, w es el residuo cuando xy se divide entre z. Así, $w = xy \bmod z$, lo que se traduce en

$$ab \bmod z = [(a \bmod z)(b \bmod z)] \bmod z.$$

Ejemplo 5.2.18 ▶

Se muestra cómo calcular $572^{29} \bmod 713$ usando el algoritmo 5.2.16 y el Teorema 5.2.17. El número 572^{29} tiene 80 dígitos, de manera que el Teorema 5.2.17 sin duda simplifica los cálculos.

Para calcular a^{29}, se calculó sucesivamente

$$a, \qquad a^5 = a \cdot a^4, \qquad a^{13} = a^5 \cdot a^8, \qquad a^{29} = a^{13} \cdot a^{16}$$

(vea el ejemplo 5.2.15). Para calcular $a^{29} \bmod z$, se calcula de manera sucesiva

$$a \bmod z, \qquad a^5 \bmod z, \qquad a^{13} \bmod z, \qquad a^{29} \bmod z.$$

Cada multiplicación se realiza usando el Teorema 5.2.17. Se calcula a^2 con la fórmula

$$a^2 \bmod z = [(a \bmod z)(a \bmod z)] \bmod z.$$

Se calcula a^4 usando la fórmula

$$a^4 \bmod z = a^2 a^2 \bmod z = [(a^2 \bmod z)(a^2 \bmod z)] \bmod z,$$

etcétera.

Se calcula a^5 con la fórmula

$$a^5 \bmod z = a a^4 \bmod z = [(a \bmod z)(a^4 \bmod z)] \bmod z.$$

Se calcula a^{13} usando la fórmula

$$a^{13} \bmod z = a^5 a^8 \bmod z = [(a^5 \bmod z)(a^8 \bmod z)] \bmod z,$$

y así sucesivamente.

Lo siguiente muestra los cálculos de $572^{29} \bmod 713$:

$$\begin{aligned} 572^2 \bmod 713 &= (572 \bmod 713)(572 \bmod 713) \bmod 713 = 572^2 \bmod 713 = 630 \\ 572^4 \bmod 713 &= (572^2 \bmod 713)(572^2 \bmod 713) \bmod 713 = 630^2 \bmod 713 = 472 \\ 572^8 \bmod 713 &= (572^4 \bmod 713)(572^4 \bmod 713) \bmod 713 = 472^2 \bmod 713 = 328 \\ 572^{16} \bmod 713 &= (572^8 \bmod 713)(572^8 \bmod 713) \bmod 713 = 328^2 \bmod 713 = 634 \end{aligned}$$

$$\begin{aligned} 572^5 \bmod 713 &= (572 \bmod 713)(572^4 \bmod 713) \bmod 713 = 572 \cdot 472 \bmod 713 = 470 \\ 572^{13} \bmod 713 &= (572^5 \bmod 713)(572^8 \bmod 713) \bmod 713 = 470 \cdot 328 \bmod 713 = 152 \\ 572^{29} \bmod 713 &= (572^{13} \bmod 713)(572^{16} \bmod 713) \bmod 713 = 152 \cdot 634 \bmod 713 = 113. \end{aligned}$$

◀

La técnica demostrada en el ejemplo 5.2.18 se formaliza como el algoritmo 5.2.19.

Algoritmo 5.2.19

Elevar un exponente mod z elevando al cuadrado varias veces

Este algoritmo calcula $a^n \bmod z$ elevando al cuadrado una y otra vez. El algoritmo se explica en el ejemplo 5.2.18.

Entrada: a, n, z
Salida: $a^n \bmod z$

```
exp_mod_z_via_cuadrado_repetido(a, n, z) {
    resultado = 1
    x = a mod z
    while (n > 0) {
        if (n mod 2 == 1)
            resultado = (resultado * x) mod z
        x = (x * x) mod z
        n = ⌊n/2⌋
    }
    return resultado
}
```

La diferencia clave entre el algoritmo 5.2.16 y 5.2.19 es el tamaño de los números que se multiplican. En el algoritmo 5.2.19, los números multiplicados son los residuos después de dividir entre z, por lo que tienen magnitud menor que z. Si se modifica el método usual de multiplicar enteros base 10 por enteros base 2, se puede demostrar (vea al ejercicio 65) que el tiempo requerido para multiplicar a y b es $O(\lg a \lg b)$. Como el ciclo "while" en el algoritmo 5.2.19 ejecuta $\Theta(\lg n)$ veces, el tiempo total para dicho algoritmo es $O(\lg n \lg^2 z)$.

Sugerencias para resolver problemas

Para convertir el número base b

$$c_n b^n + c_{n-1} b^{n-1} + \cdots + c_1 b^1 + c_0 b^0$$

a decimal, se realizan las multiplicaciones y sumas en decimal indicadas.

Para convertir el número decimal n a la base b, se divide entre b, el cociente obtenido se divide entre b, el cociente obtenido se divide entre b, etcétera, hasta que el residuo sea cero. Los residuos dan la representación en base b de n. El primer residuo da el coeficiente de 1, el siguiente residuo da el coeficiente de b, y así sucesivamente.

Al multiplicar módulo z, calcule los residuos en cuanto pueda para minimizar los tamaños de los números implicados.

Sección de ejercicios de repaso

1. ¿Cuál es el valor del número decimal $d_n d_{n-1} \ldots d_1 d_0$? ? (Cada d_i es un entero entre 0 y 9).
2. ¿Cuál es el valor del número binario $b_n b_{n-1} \ldots b_1 b_0$? (Cada b_i es 0 o 1).
3. ¿Cuál es el valor del número hexadecimal $h_n h_{n-1} \ldots h_1 h_0$? (Cada h_i está entre 0 y 9 o entre A y F).
4. ¿Cuántos bits se requieren para representar el entero positivo n?
5. Explique cómo convertir de binario a decimal.
6. Explique cómo convertir de decimal a binario.
7. Explique cómo convertir de hexadecimal a decimal.
8. Explique cómo convertir de decimal a hexadecimal.
9. Explique cómo sumar números binarios.
10. Explique cómo sumar números hexadecimales.
11. Explique cómo calcular a^n elevando al cuadrado repetidas veces.
12. Explique cómo calcular $a^n \bmod z$ elevando al cuadrado repetidas veces.

Ejercicios

¿Cuántos bits se necesitan para representar cada entero en los ejercicios 1 al 7?

1. 60 ***2.*** 63 ***3.*** 64

4. 127 ***5.*** 128 ***6.*** 2^{1000}

7. 3^{1000}

En los ejercicios 8 al 13, exprese cada número binario en decimal.

8. 1001 ***9.*** 11011

10. 11011011 ***11.*** 100000

12. 11111111 **13.** 110111011011

En los ejercicios 14 al 19, exprese cada número decimal en binario.

14. 34 **15.** 61 **16.** 223

17. 400 **18.** 1024 **19.** 12,340

En los ejercicios 20 al 25, sume los números binarios.

20. 1001 + 1111

21. 11011 + 1101

22. 110110 + 101101

23. 101101 + 11011

24. 110110101 + 1101101

25. 1101 + 101100 + 11011011

En los ejercicios 26 al 31, exprese cada número hexadecimal en decimal.

26. 3A **27.** 1E9

28. 3E7C ***29.*** A03

30. 209D **31.** 4B07A

32. Exprese cada número binario en los ejercicios 8 al 13 en hexadecimal.

33. Exprese cada número decimal en los ejercicios 14 al 19 en hexadecimal.

34. Exprese cada número hexadecimal en los ejercicios 26, 27 y 29 en binario.

En los ejercicios 35 al 39, sume los números hexadecimales.

35. 4A+ B4 **36.** 195 + 76E

37. 49F7 + C66 ***38.*** 349CC + 922D

39. 82054 + AEFA3

40. ¿Representa 2010 un número binario?, ¿decimal?, ¿hexadecimal?

41. ¿Representa 1101010 un número binario?, ¿decimal?, ¿hexadecimal?

En el sistema numérico octal (base 8), para representar enteros se usan los símbolos 0, 1, 2, 3, 4, 5, 6 y 7. Al representar un entero, leyendo de derecha a izquierda, el primer símbolo representa el número de unos, el siguiente símbolo el número de ochos, el siguiente el número de ochos al cuadrado, etcétera. En general, el símbolo en la posición n (donde la extrema derecha es la posición 0) representa el número de números 8^n. En los ejercicios 42 al 47, exprese cada número octal en decimal.

42. 63 **43.** 7643

44. 7711 ***45.*** 10732

46. 1007 **47.** 537261

48. Exprese cada número binario en los ejercicios 8 al 13 en octal.

49. Exprese cada número decimal en los ejercicios 14 al 19 en octal.

50. Exprese cada número hexadecimal en los ejercicios 26 al 31 en octal.

51. Exprese cada número octal en los ejercicios 42 al 47 en hexadecimal.

52. ¿Representa 1101010 un número en octal?

53. ¿Representa 30470 un número binario?, ¿octal?, ¿decimal?, ¿hexadecimal?

54. ¿Representa 9450 un número binario?, ¿octal?, ¿decimal?, ¿hexadecimal?

55. Pruebe que un entero m en base b tiene $\lfloor 1 + \log_b m \rfloor$ dígitos.

En los ejercicios 56 al 58, siga el algoritmo 5.2.16 para el valor dado de n.

56. $n = 16$ **57.** $n = 15$ **58.** $n = 80$

En los ejercicios 59 al 61, siga el algoritmo 5.2.19 para los valores dados de a, n y z.

59. $a = 5, n = 10, z = 21$

60. $a = 143, n = 10, z = 230$

61. $a = 143, n = 100, z = 230$

62. Sea T_k la potencia más alta de 2 que divide a n. Demuestre que $T_{mn} = T_m + T_n$ para toda $m, n \geq 1$.

63. Sea S_n el número de unos en la representación binaria de n. Use inducción para probar que $T_{n!} = n - S_n$ para toda $n \geq 1$. (T_n se definió en el ejercicio anterior).

64. Modifique el método usual de multiplicación de enteros base 10 para la base 2, para producir un algoritmo que multiplique números binarios $b_m b_{m-1} \cdots b_1 b_0$ y $b'_n b'_{n-1} \cdots b'_1 b'_0$.

65. Demuestre que el tiempo requerido por el algoritmo del ejercicio 64 para multiplicar a y b es $O(\lg a \lg b)$.

5.3 ➜ El algoritmo euclidiano

En la sección 5.1 se estudiaron algunos métodos para calcular el máximo común divisor de dos enteros que resultaron ineficientes. El **algoritmo euclidiano** es un algoritmo antiguo, conocido y *eficiente* para encontrar el máximo común divisor de dos enteros.

El algoritmo euclidiano se basa en el hecho de que si $r = a \bmod b$, entonces

$$\text{mcd}(a, b) = \text{mcd}(b, r). \qquad \textbf{(5.3.1)}$$

Antes de probar (5.3.1), se ilustra cómo usa esta ecuación el algoritmo euclidiano para encontrar el máximo común divisor.

Ejemplo 5.3.1 ▶

Como 105 mod 30 = 15, por (5.3.1)

$$\text{mcd}(105, 30) = \text{mcd}(30,15).$$

Como 30 mod 15 = 0, por (5.3.1)

$$\text{mcd}(30, 15) = \text{mcd}(15, 0).$$

Por inspección, mcd(15, 0) = 15. Por lo tanto,

$$\text{mcd}(105, 30) = \text{mcd}(30, 15) = \text{mcd}(15, 0) = 15.$$ ◀

Ahora se demostrará la ecuación (5.3.1).

Teorema 5.3.2

Si a es un entero no negativo, b es un entero positivo y $r = a \bmod b$*, entonces*

$$\text{mcd}(a, b) = \text{mcd}(b, r).$$

Demostración Por el teorema del cociente-residuo, existen q y r que satisfacen

$$a = bq + r, \qquad \text{o} \leq r < b.$$

Se demuestra que el conjunto de divisores comunes de a y b es igual al conjunto de divisores comunes de b y r, lo que prueba el teorema.

Sea c un divisor común de a y b. Por el Teorema 5.1.3(c), $c \mid bq$. Como $c \mid a$ y $c \mid bq$, por el Teorema 5.1.3(b), $c \mid a - bq$ $(= r)$. Entonces c es un divisor común de b y r. Inversamente, si c es un divisor común de b y r, entonces, $c \mid bq$ y $c \mid bq + r$ $(= a)$ y c es un divisor común de a y b. Así, el conjunto de divisores comunes de a y b es igual al conjunto de divisores comunes de b y r. Por lo tanto,

$$\text{mcd}(a, b) = \text{mcd}(b, r).$$

Ahora se establecerá de manera formal el algoritmo euclidiano como el algoritmo 5.3.3.

Algoritmo 5.3.3

Algoritmo euclidiano

Este algoritmo encuentra el máximo común divisor de los enteros no negativos a y b, donde no son cero a y b.

```
        Entrada:    a y b (enteros no negativos, ambos diferentes de cero)
        Salida:     máximo común divisor de a y b
1.      mcd(a, b) {
2.          // sea a el mayor
3.          if (a < b)
4.              intercambia(a, b)
5.          while (b ¬=0) {
6.              r = a mod b
7.              a = b
8.              b = r
9.          }
10.         return a
11.     }
```

Se observa que el ciclo "while" en el algoritmo euclidiano (líneas 5 al 9) siempre termina al final del ciclo (líneas 7 y 8), los valores de a y b se actualizan con valores *más pequeños*. Como los enteros no negativos no pueden decrecer indefinidamente, en algún momento b se convierte en cero y el ciclo termina.

Sea $G = \text{mcd}(a, b)$, donde a y b son los valores de entrada al algoritmo 5.3.3. Se demuestra que el algoritmo 5.3.3 es correcto verificando que $G = \text{mcd}(a, b)$ es un invariante de ciclo, donde ahora a y b denotan la variables en el pseudocódigo.

Por definición, la invariante de ciclo es verdadera la primera vez que se llega a la línea 5. Suponga que $G = \text{mcd}(a, b)$ es verdadera antes de la siguiente iteración del ciclo y que $b \neq 0$. El Teorema 5.3.2 nos dice que después de ejecutar la línea 6,

$$\text{mcd}(a, b) = \text{mcd}(b, r).$$

En las líneas 7 y 8, a se convierte en b y b se convierte en r. Por lo tanto, $G = \text{mcd}(a, b)$ es verdadera para los nuevos valores de a y b. Se concluye que $G = \text{mcd}(a, b)$ es una invariante del ciclo. El ciclo "while" termina cuando b se hace 0. En este punto, la invariante del ciclo es $G = \text{mcd}(a, 0)$. El algoritmo regresa a [$= \text{mcd}(a, 0)$]. Entonces el valor que regresa el algoritmo es G, que por definición es el máximo común divisor de los valores de entrada. Por lo tanto, el algoritmo 5.3.3 es correcto.

El algoritmo 5.3.3 encuentra correctamente el máximo común divisor si se omiten las líneas 3 y 4 (vea el ejercicio 13). Estas líneas se incluyen porque simplifican el análisis del algoritmo 5.3.3 en el siguiente apartado.

Ejemplo 5.3.4 ▶

Se mostrará cómo el algoritmo 5.3.3 encuentra el mcd(504, 396).

Sean $a = 504$ y $b = 396$. Como $a > b$, pasamos a la línea 5. Como $b \neq 0$, se procede a la línea 6, donde se hace r igual a

$$a \bmod b = 504 \bmod 396 = 108.$$

Después pasamos a las líneas 7 y 8, donde a se hace igual a 396 y b igual a 108. Después regresamos a la línea 5.

Como $b \neq 0$, se procede a la línea 6, donde se hace r igual a

$$a \bmod b = 396 \bmod 108 = 72.$$

Después nos movemos a las líneas 7 y 8, donde a se hace igual a 108 y b igual a 72. Se regresa a la línea 5.

Como $b \neq 0$, se procede a la línea 6, donde r se iguala a

$$a \bmod b = 108 \bmod 72 = 36.$$

Ahora vamos a las líneas 7 y 8, donde a se hace igual a 72 y b a 36. Después regresamos a la línea 5.

Como $b \neq 0$, se procede a la línea 6, donde r se hace igual a

$$a \bmod b = 72 \bmod 36 = 0.$$

Pasamos de nuevo a las líneas 7 y 8, donde a es 36 y b es 0. Regresamos a la línea 5.

Ahora $b = 0$, por lo que vamos a la línea 10, donde la salida es a (36), el máximo común divisor de 396 y 504. ◀

Análisis del algoritmo euclidiano

Se analiza el desempeño del algoritmo 5.3.3 en el peor caso. Se define el tiempo requerido como el número de operaciones de módulo que se ejecutan en la línea 6. La tabla 5.3.1 lista el número de operaciones del módulo requeridas para algunos valores de entrada pequeños.

El peor caso para el algoritmo euclidiano ocurre cuando el número de operaciones de módulo es tan grande como sea posible. En referencia a la tabla 5.3.1, se puede determinar el par de entrada a, b, $a > b$, con a tan pequeña como sea posible, que requiere n operaciones de módulo para $n = 0, \ldots, 5$. Los resultados se presentan en la tabla 5.3.2.

Recuerde que la sucesión de Fibonacci $\{f_n\}$ (vea la sección 4.4) se define por las ecuaciones

$$f_1 = 1, \qquad f_2 = 1, \qquad f_n = f_{n-1} + f_{n-2}, \qquad n \geq 3.$$

La sucesión de Fibonacci comienza

$$1, \quad 1, \quad 2, \quad 3, \quad 5, \quad 8, \ldots.$$

Un patrón sorprendente se desarrolla en la tabla 5.3.2: la columna a es la sucesión de Fibonacci comenzando con f_2 y, excepto por el primer valor, la columna b también es la suce-

TABLA 5.3.1 ■ Número de operaciones de módulo requeridas por el algoritmo euclidiano para diferentes valores de entrada.

a \ *b*	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	—	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	1	1	2	1	2	1	2	1	2	1	2	1	2
3	0	1	2	1	2	3	1	2	3	1	2	3	1	2
4	0	1	1	2	1	2	2	3	1	2	2	3	1	2
5	0	1	2	3	2	1	2	3	4	3	1	2	3	4
6	0	1	1	1	2	2	1	2	2	2	3	3	1	2
7	0	1	2	2	3	3	2	1	2	3	3	4	4	3
8	0	1	1	3	1	4	2	2	1	2	2	4	2	5
9	0	1	2	1	2	3	2	3	2	1	2	3	2	3
10	0	1	1	2	2	1	3	3	2	2	1	2	2	3
11	0	1	2	3	3	2	3	4	4	3	2	1	2	3
12	0	1	1	1	1	3	1	4	2	2	2	2	1	2
13	0	1	2	2	2	4	2	3	5	3	3	3	2	1

TABLA 5.3.2 ■ Par más pequeño de entrada que requiere *n* operaciones de módulo en el algoritmo euclidiano.

a	*b*	*n* (= *número de operaciones de módulo*)
1	0	0
2	1	1
3	2	2
5	3	3
8	5	4
13	8	5

$34 = 91 \bmod 57$	(1 operación de módulo)
57, 34 requiere 4 operaciones de módulo	(con un total de 5)
$57 \geq f_6$ y $34 \geq f_5$	(por la suposición inductiva)
$\therefore\ 91 = 57 \cdot 1 + 34 \geq 57 + 34 \geq f_6 + f_5 = f_7$	

Figura 5.3.1 Prueba del Teorema 5.3.5. El par 91, 57, que requiere $n + 1 = 5$ operaciones de módulo, es la entrada al algoritmo euclidiano.

sión de Fibonacci ¡comenzando con f_2! Se llega a la conjetura de que si el par $a, b, a > b$, cuando se introduce al algoritmo euclidiano requiere $n \geq 1$ operaciones de módulo, entonces $a \geq f_{n+2}$ y $b \geq f_{n+1}$. Como evidencia adicional de la conjetura, si se calcula el par de entrada más pequeño que requiere 6 operaciones de módulo, se obtiene $a = 21$ y $b = 13$. El siguiente Teorema confirma que la conjetura es correcta. La prueba del Teorema se ilustra en la figura 5.3.1.

Teorema 5.3.5

Suponga que el par a, b, a > b, requiere n ≥ 1 operaciones de módulo cuando se introduce al algoritmo. Entonces $a \geq f_{n+2}$ y $b \geq f_{n+1}$, *donde* $\{f_n\}$ *denota la sucesión de Fibonacci.*

Demostración La prueba es por inducción sobre n.

Paso base ($n = 1$) Se ha observado que el teorema es cierto para $n = 1$.

Paso inductivo Suponga que el Teorema es cierto para $n \geq 1$. Debe demostrarse que el teorema es cierto para $n + 1$.

Suponga que el par $a, b, a > b$, requiere $n + 1$ operaciones de módulo cuando se introduce al algoritmo euclidiano. En la línea 6, se calcula $r = a \bmod b$. Entonces

$$a = bq + r, \qquad 0 \leq r < b. \tag{5.3.2}$$

Luego, el algoritmo repite esto usando los valores b y r, $b > r$. Estos valores requieren n operaciones de módulo adicionales. Por la suposición inductiva,

$$b \geq f_{n+2} \qquad \text{y} \qquad r \geq f_{n+1}. \tag{5.3.3}$$

Combinando (5.3.2) y (5.3.3), se obtiene

$$a = bq + r \geq b + r \geq f_{n+2} + f_{n+1} = f_{n+3}. \tag{5.3.4}$$

[La primera desigualdad en (5.3.4) se cumple porque $q > 0$; q no puede ser igual que 0 porque $a > b$.] Las desigualdades (5.3.3) y (5.3.4) dan

$$a \geq f_{n+3} \qquad \text{y} \qquad b \geq f_{n+2}$$

El paso inductivo está terminado y la prueba queda completa.

El teorema 5.3.5 resulta útil para analizar el desempeño del algoritmo euclidiano en el peor caso.

Teorema 5.3.6

Si los enteros en un intervalo de 0 *a* m, $m \geq 8$, *ambos diferentes de cero, se introducen al algoritmo euclidiano, entonces se requieren cuando mucho*

$$\log_{3/2} \frac{2m}{3}$$

operaciones de módulo.

Demostración Sea n el número máximo de operaciones de módulo requeridas por el algoritmo euclidiano para enteros en el intervalo de 0 a m, $m \geq 8$. Sea a, b un par de entrada en el intervalo de 0 a m que requiere n operaciones de módulo. La tabla 5.3.1 muestra que $n \geq 4$ y $a \neq b$. Se puede suponer que $a > b$. (Al intercambiar los valores de a y b no se altera el número de operaciones de módulo requeridas). Por el Teorema 5.3.5, $a \geq f_{n+2}$. Entonces

$$f_{n+2} \leq m.$$

Por el ejercicio 27, en la sección 4.4, como $n + 2 \geq 6$,

$$\left(\frac{3}{2}\right)^{n+1} < f_{n+2}.$$

Combinando estas dos desigualdades, se obtiene

$$\left(\frac{3}{2}\right)^{n+1} < m.$$

Tomando logaritmos con base 3/2, se obtiene

$$n + 1 < \log_{3/2} m.$$

Por lo tanto,

$$n < (\log_{3/2} m) - 1 = \log_{3/2} m - \log_{3/2} \frac{3}{2} = \log_{3/2} \frac{2m}{3}.$$

Como la función logaritmo crece con lentitud, el Teorema 5.3.6 nos dice que el algoritmo euclidiano es bastante eficiente, aun para valores grandes de los datos de entrada. Por ejemplo, dado que

$$\log_{3/2} \frac{2(1{,}000{,}000)}{3} = 33.07\ldots,$$

el algoritmo euclidiano requiere cuando mucho 33 operaciones de módulo para calcular el máximo común divisor de cualquier par de enteros, ambos diferentes de cero, en el intervalo de 0 a 1,000,000.

Un resultado especial

El siguiente resultado especial se usará para calcular el inverso del módulo de un entero (vea el siguiente apartado). Estos inversos se usan en los sistemas criptográficos RSA(sección 5.4). Sin embargo, este resultado especial también es útil de otras maneras (vea los ejercicios 24 y 26 y el rincón de solución de problemas que sigue).

Teorema 5.3.7

Si a y b son enteros no negativos, ambos diferentes de cero, existen enteros s y t tales que

$$\text{mcd}(a, b) = sa + tb.$$

El método del algoritmo euclidiano se puede usar para probar el teorema 5.3.7 y calcular s y t. Antes de demostrar el Teorema, se ilustrará la prueba con un ejemplo específico.

Ejemplo 5.3.8 ▶

Considere la forma en que el algoritmo euclidiano calcula el mcd(273, 110). Se comienza con $a = 273$ y $b = 110$. El algoritmo euclidiano calcula primero

$$r = 273 \bmod 110 = 53. \tag{5.3.5}$$

Después establece $a = 110$ y $b = 53$.

Luego el algoritmo calcula

$$r = 110 \bmod 53 = 4. \tag{5.3.6}$$

Después hace $a = 53$ y $b = 4$.

El algoritmo euclidiano calcula entonces

$$r = 53 \bmod 4 = 1 \tag{5.3.7}$$

Luego establece $a = 4$ y $b = 1$.

Ahora el algoritmo calcula

$$r = 4 \bmod 1 = 0.$$

Como $r = 0$, el algoritmo termina por encontrar el máximo común divisor de 273 y 110 como 1.

Para encontrar s y t, se trabaja hacia atrás, comenzando con la última ecuación [(5.3.7)] donde $r \neq 0$. La ecuación (5.3.7) se rescribe como

$$1 = 53 - 4 \cdot 13 \tag{5.3.8}$$

ya que el cociente cuando 53 se divide entre 4 es 13.

La ecuación (5.3.6) se rescribe como

$$4 = 110 - 53 \cdot 2.$$

Después se sustituye esta fórmula para 4 en la ecuación (5.3.8) para obtener

$$1 = 53 - 4 \cdot 13 = 53 - (110 - 53 \cdot 2)13 = 27 \cdot 53 - 13 \cdot 110. \tag{5.3.9}$$

La ecuación (5.3.5) se rescribe como

$$53 = 273 - 110 \cdot 2.$$

Se sustituye esta fórmula para 53 en la ecuación (5.3.9) y se obtiene

$$1 = 27 \cdot 53 - 13 \cdot 110 = 27(273 - 110 \cdot 2) - 13 \cdot 110 = 27 \cdot 273 - 67 \cdot 110.$$

Así, si $s = 27$ y $t = -67$, se obtiene

$$\text{mcd}(273, 110) = 1 = s \cdot 273 + t \cdot 110. \quad \blacktriangleleft$$

Demostración del Teorema 5.3.7 Dados $a > b \geq 0$, sea $r_0 = a$, $r_1 = b$ y r_i igual al valor de r después de la $(i-1)$ésima vez que se ejecuta el ciclo en el algoritmo 5.3.3 (por ejemplo, $r_2 = a \bmod b$). Suponga que r_n es el primer valor de r que es cero, de manera que $\text{mcd}(a, b) = r_{n-1}$.

En general,

$$r_i = r_{i+1}q_{i+2} + r_{i+2}. \tag{5.3.10}$$

Haciendo $i = n - 3$ en (5.3.10), se obtiene

$$r_{n-3} = r_{n-2}q_{n-1} + r_{n-1},$$

que se rescribe como

$$r_{n-1} = -q_{n-1}r_{n-2} + 1 \cdot r_{n-3}.$$

Se puede hacer $t_{n-3} = -q_{n-1}$ y $s_{n-3} = 1$ para obtener

$$r_{n-1} = t_{n-3}r_{n-2} + s_{n-3}r_{n-3}. \tag{5.3.11}$$

Haciendo $i = n - 4$ en (5.3.10) se obtiene

$$r_{n-4} = r_{n-3}q_{n-2} + r_{n-2}$$

o sea

$$r_{n-2} = -q_{n-2}r_{n-3} + r_{n-4}. \tag{5.3.12}$$

Al sustituir (5.3.12) en (5.3.11) se tiene

$$\begin{aligned} r_{n-1} &= t_{n-3}[-q_{n-2}r_{n-3} + r_{n-4}] + s_{n-3}r_{n-3} \\ &= [-t_{n-3}q_{n-2} + s_{n-3}]r_{n-3} + t_{n-3}r_{n-4}. \end{aligned}$$

Estableciendo $t_{n-4} = -t_{n-3}q_{n-2} + s_{n-3}$ y $s_{n-4} = t_{n-3}$, se obtiene

$$r_{n-1} = t_{n-4}r_{n-3} + s_{n-4}r_{n-4}.$$

Si se continúa de esta manera, al final se obtiene

$$\text{mcd}(r_0, r_1) = r_{n-1} = t_0r_1 + s_0r_0 = t_0\, b = s_0a.$$

Y si se hace $s = s_0$ y $t = t_0$, se llega a

$$\text{mcd}(r_0, r_1) = sa + tb.$$ ◀

Cálculo del inverso del módulo de un entero

Suponga que se tienen dos enteros $n > 0$ y $\phi > 1$ tal que $\text{mcd}(n, \phi) = 1$. Se mostrará cómo calcular de manera eficiente un entero s, $0 < s < \phi$ tal que $ns \bmod \phi = 1$. Llamamos a s el **inverso de *n* mod** ϕ. El sistema criptográfico RSA en la sección 5.4 requiere calcular con eficiencia este inverso.

Como el $\text{mcd}(n, \phi) = 1$, se usa el algoritmo euclidiano, como se explicó antes, para encontrar números s' y t' tales que $s'n + t'\phi = 1$. Entonces $ns' = -t'\phi + 1$ y, como $\phi > 1$, el residuo es 1. Entonces

$$ns' \bmod \phi = 1. \tag{5.3.13}$$

Advierta que s' es casi el valor deseado; el problema es que quizá s' no satisfaga $0 < s' < \phi$. Sin embargo, es posible convertir s' en el valor adecuado haciendo

$$s = s' \bmod \phi.$$

Ahora $0 \leq s < \phi$. De hecho, $s \neq 0$ puesto que si $s = 0$, entonces $\phi \mid s'$, lo que contradice (5.3.13). Como $s = s' \bmod \phi$, existe q tal que

$$s' = q\phi + s.$$

Combinando las ecuaciones anteriores, se tiene

$$ns = ns' - \phi nq = -t'\phi + 1 - \phi nq = \phi(-t' - nq) + 1.$$

Por lo tanto,

$$ns \bmod \phi = 1. \tag{5.3.14}$$

Ejemplo 5.3.9 ▶

Sean $n = 110$ y $\phi = 273$. En el ejemplo 5.3.8, se demostró que el mcd$(n, \phi) = 1$ y que

$$s'n + t'\phi = 1,$$

donde $s' = -67$ y $t' = 27$. Así,

$$110(-67) \bmod 273 = ns' \bmod \phi = 1.$$

Aquí $s = s' \bmod \phi = -67 \bmod 273 = 206$. Por lo tanto, el inverso de 110 módulo 273 es 206. ◀

Se concluye probando que el número s en la ecuación (5.3.14) es único. Suponga que

$$ns \bmod \phi = 1 = ns' \bmod \phi, \qquad 0 < s < \phi, \qquad 0 < s' < \phi.$$

Debe demostrarse que $s = s'$. Ahora

$$s' = (s' \bmod \phi)(ns \bmod \phi) = s'ns \bmod \phi = (s'n \bmod \phi)(s \bmod \phi) = s.$$

Por lo tanto, el número s en la ecuación (5.3.14) es único.

Sugerencias para resolver problemas

El algoritmo euclidiano para calcular el máximo común divisor de enteros no negativos a y b, ambos diferentes de cero, se basa en la ecuación

$$\text{mcd}(a, b) = \text{mcd}(b, r),$$

donde $r = a \bmod b$. Se sustituye el problema original, calcular el mcd(a, b), por el problema, calcular el mcd(b, r). Después se remplaza a por b y b por r, y se repite. A la larga, $r = 0$, de manera que la solución es mcd$(b, 0) = b$.

El algoritmo euclidiano es bastante eficiente. Si dos enteros en el intervalo de 0 a m, $m \geq 8$, ambos diferentes de cero, se dan como datos al algoritmo euclidiano, entonces se requieren cuando mucho

$$\log_{3/2} \frac{2m}{3}$$

operaciones de módulo.

Si a y b son enteros no negativos, ambos diferentes de cero, existen enteros s y t tales que

$$\text{mcd}(a, b) = sa + tb.$$

Para calcular s y t, se usa el algoritmo euclidiano. En un problema que implica el máximo común divisor, la ecuación anterior resultará útil. (Intente realizar los ejercicios 24 y 26).

Suponga que se tienen dos enteros $n > 0$ y $\phi > 1$ tales que mcd$(n, \phi) = 1$. Para calcular de manera eficiente un entero s, $0 < s < \phi$ tal que $ns \bmod \phi = 1$, primero se calculan s' y t' que satisfacen

$$\text{mcd}(n, \phi) = s'n + t'\phi$$

(vea el apartado "Cálculo del inverso del módulo de un entero"). Después se hace $s = s' \bmod \phi$.

Sección de ejercicios de repaso

1. Enuncie el algoritmo euclidiano.
2. ¿Qué teorema clave es la base para el algoritmo euclidiano?
3. Si el par $a, b, a > b$, requiere $n \geq 1$ operaciones de módulo cuando se alimenta al algoritmo euclidiano, ¿cómo se relacionan a y b con la sucesión de Fibonacci?
4. Dos enteros en el intervalo de 0 a m, $m \geq 8$, ambos diferentes de cero, se alimentan al algoritmo euclidiano. Dé una cota superior para el número de operaciones de módulo requeridas.
5. El Teorema 5.3.7 establece que existen enteros s y t tales que $\text{mcd}(a, b) = sa + tb$. Explique cómo utilizar el algoritmo euclidiano para calcular s y t.
6. Explique qué significa que s sea el inverso de n módulo ϕ.
7. Suponga que $\text{mcd}(n, \phi) = 1$. Explique cómo calcular el inverso de n módulo ϕ.

Ejercicios

Use el algoritmo euclidiano para encontrar el máximo común divisor de cada par de enteros en los ejercicios 1 al 10.

1. 60, 90
2. 110, 273
3. 220, 1400
4. 315, 825
5. 20, 40
6. 331, 993
7. 2091, 4807
8. 2475, 32670
9. 67942, 4209
10. 490256, 337
11. Para cada par de números a, b en los ejercicios 1 al 10, encuentre enteros s y t tales que $sa + tb = \text{mcd}(a, b)$.
12. Encuentre dos enteros a y b, cada uno menor que 100, que maximicen el número de iteraciones del ciclo "while" del algoritmo 5.3.3.
13. Demuestre que el algoritmo 5.3.3 encuentra correctamente el mcd(a, b) aun cuando se eliminen las líneas 3 y 4.
14. Escriba una versión recursiva del algoritmo euclidiano. Pruebe que su algoritmo es correcto.
15. Si a y b son enteros positivos, demuestre que $\text{mcd}(a, b) = \text{mcd}(a, a + b)$.
16. Demuestre que si $a > b \geq 0$, entonces

$$\text{mcd}(a, b) = \text{mcd}(a - b, b).$$

17. Con base en el ejercicio 16, escriba un algoritmo que calcule el máximo común divisor de dos enteros no negativos a y b, ambos diferentes de cero, que use la resta pero no las operaciones de módulo.
18. ¿Cuántas restas requiere el algoritmo del ejercicio 17 en el peor caso para números en el intervalo de 0 a m?
19. Amplíe las tablas 5.3.1 y 5.3.2 al intervalo de 0 a 21.
20. ¿Exactamente cuántas operaciones de módulo requiere el algoritmo euclidiano en el peor caso para números entre 0 y 1,000,000?
21. Pruebe que cuando el par f_{n+2}, f_{n+1} se introduce en el algoritmo euclidiano, $n \geq 1$, se requieren exactamente n operaciones de módulo.
22. Demuestre que para cualquier entero $k > 1$, el número de operaciones de módulo requeridas por el algoritmo euclidiano para calcular el mcd(a, b) es el mismo que el número de operaciones de módulo requeridas para calcular mcd(ka, kb).
23. Demuestre que el $\text{mcd}(f_n, f_{n+1}) = 1, n \geq 1$.
24. ★ Demuestre que si p es un número primo, a y b son enteros positivos y $p \mid ab$, entonces $p \mid a$ o $p \mid b$.
25. Dé un ejemplo de enteros positivos p, a y b donde $p \mid ab$, $p \nmid a$ y $p \nmid b$.
26. Sean m y n enteros positivos. Sea f la función de

$$X = \{0,1, \ldots, m - 1\}$$

en X definida por

$$f(x) = nx \bmod m.$$

Pruebe que f es uno a uno y sobre si y sólo si $\text{mcd}(m, n) = 1$.

Los ejercicios 27 al 31 muestran otra manera de probar que si a y b son enteros no negativos, ambos diferentes de cero, existen enteros s y t tales que

$$\text{mcd}(a, b) = sa + tb.$$

Sin embargo, a diferencia del algoritmo euclidiano, esta demostración no conduce a una técnica para calcular s y t.

27. Sea

$$X = \{sa + tb \mid sa + tb > 0 \text{ y } s \text{ y } t \text{ son enteros}\}.$$

Demuestre que X es no vacío.

28. Demuestre que X tiene un elemento menor. Denote por g el elemento menor.
29. Demuestre que si c es un divisor común de a y b, entonces c divide a g.
30. Demuestre que g es un divisor común de a y b. *Sugerencia:* Suponga que g no divide a a. Entonces $a = qg + r, 0 < r < g$. Obtenga una contradicción demostrando que $r \in X$.
31. Demuestre que g es el máximo común divisor de a y b.

En los ejercicios 32 al 38, demuestre que el $\text{mcd}(n, \phi) = 1$ *y encuentre el inverso s de n módulo ϕ que satisface* $0 < s < \phi$.

32. $n = 2, \phi = 3$
33. $n = 1, \phi = 47$
34. $n = 7, \phi = 20$
35. $n = 11, \phi = 47$
36. $n = 50, \phi = 231$
37. $n = 100, \phi = 231$
38. $n = 100, \phi = 243$
39. Demuestre que 6 no tiene inverso módulo 15. ¿Contradice esto el resultado que precede al ejemplo 5.3.9? Explique su respuesta.
40. Demuestre que $n > 0$ tiene un inverso módulo $\phi > 1$ si y sólo si $\text{mcd}(n, \phi) = 1$.

Rincón de solución de problemas

Composición del importe postal

Problema

Sean p y q enteros positivos que satisfacen mcd$(p, q) = 1$. Demuestre que existe n tal que para toda $k \geq n$, se puede lograr el importe postal de k centavos usando sólo timbres de p y q centavos.

Cómo atacar el problema

¿Suena familiar este tipo de problema? El ejemplo 1.8.1 usó inducción matemática para demostrar que el importe postal de cuatro centavos o más se puede componer usando sólo timbres de 2 y 5 centavos. Este resultado ilustra el problema para $p = 2$ y $q = 5$. En este caso, si se toma $n = 4$, para toda $k \geq 4$, el importe postal de k centavos se puede lograr usando sólo timbres de 2 y 5 centavos. En este momento, se recomienda revisar la prueba por inducción de dicho resultado.

La prueba por inducción del problema de $p = 2$ y $q = 5$ se resume como sigue. Primero se demuestran los casos base ($k = 4, 5$). En el paso inductivo, se supuso que el importe de $k - 2$ centavos se puede componer. Después se agregó un timbre de 2 centavos para lograr el importe de k centavos. La prueba inductiva se limitará a probar que si mcd$(p, q) = 1$ para p y q arbitrarios, existe n tal que para toda $k \geq n$, se puede lograr el importe postal de k centavos usando sólo timbres de p y q centavos.

Cómo encontrar una solución

Primero se maneja el caso trivial. Si cualquiera de los dos p o q es 1, se puede lograr un importe postal de k centavos para toda $k \geq 1$ usando k timbres de 1 centavo. Entonces, se supone que $p > 1$ y $q > 1$.

Primero se desarrollará cierta notación. Para un importe postal dado, n_p denota el número de timbres de p centavos usados y n_q el número de timbres de q centavos. Entonces el importe postal es

$$n_p p + n_q q.$$

¿Le recuerda algo esta expresión? El Teorema 5.3.7 establece que existen enteros s y t tales que

$$1 = \text{mcd}(p, q) = sp + tq. \tag{1}$$

Esta ecuación sugiere que se puede lograr el importe postal de 1 centavo usando s timbres de p centavos y t timbres de q centavos. El problema es que uno de los dos, s o t, debe ser negativo para que la suma $sp + tq$ sea 1 (ya que p y q son mayores que 1). De hecho, como p y q son ambos mayores que 1, uno de los dos, s o t, es positivo y el otro es negativo. Se supondrá que $s > 0$ y $t < 0$.

Se verá cómo debe funcionar el paso inductivo y después qué pasos base se necesitan. Imitando el caso específico presentado, sería bueno suponer que podemos formar un importe postal de $k - p$ centavos y después agregar un timbre de p centavos para tener el importe de k centavos. ¡No hay problema! Para que este paso inductivo funcione, los pasos base deben ser $n, n + 1, \ldots, n + p - 1$ para alguna n que se puede elegir.

Suponga que se puede tener un importe postal de n centavos:

$$n = n_p p + n_q q.$$

Mediante la ecuación (1), se puede componer un importe de $(n + 1)$ centavos.

$$n + 1 = (n_p p + n_q q) + (sp + tq) = (n_p + s)p + (n_q + t)q$$

usando $n_p + s$ timbres de p centavos y $n_q + t$ timbres de q centavos. Por supuesto, esta última afirmación es significativa sólo si $n_p + s \geq 0$ y $n_q + t \geq 0$. Sin embargo, $n_p + s \geq 0$ se cumple porque $n_p \geq 0$ y $s > 0$. Se puede *arreglar* que $n_q + t \geq 0$ se cumpla eligiendo $n_q \geq -t$.

De manera similar, es posible componer un importe de $(n + 2)$ centavos

$$\begin{aligned} n + 2 &= (n_p p + n_q q) + 2(sp + tq) \\ &= (n_p + 2s)p + (n_q + 2t)q \end{aligned}$$

usando $n_p + 2s$ timbres de p centavos y $n_q + 2t$ timbres de q centavos. Como antes, $n_p + 2s \geq 0$ se cumple porque $n_p \geq 0$ y $s > 0$. Se puede *arreglar* que $n_q + 2t \geq 0$ se cumpla eligiendo $n_q \geq -2t$. Observe que para esta elección de n_q, $n_q \geq -t$ también se cumple (de manera que *todavía* se puede tener un importe postal de $n + 1$ centavos también).

En general, se puede lograr un importe postal de $(n + i)$ centavos

$$\begin{aligned} n + i &= (n_p p + n_q q) + i(sp + tq) \\ &= (n_p + is)p + (n_q + it)q \end{aligned}$$

usando $n_p + is$ timbres de p centavos y $n_q + it$ timbres de q centavos. Como antes, $n_p + is \geq 0$ se cumple porque $n_p \geq 0$ y $s > 0$. Se puede *arreglar* que $n_q + it \geq 0$ se cumpla eligiendo $n_q \geq -it$. Observe que para esta elección de n_q, $n_q \geq -jt$ también se cumple para $j = 0, \ldots, i - 1$ (de manera que *todavía* se puede tener un importe de $n + j$ centavos también).

Se deduce que es posible formar un importe postal para $n, n + 1, \ldots, n + p - 1$ siempre que se elija $n_q = -(p - 1)t$. Cualquier valor para $n_p \geq 0$ servirá, así se toma $n_p = 0$. Esto hace $n = n_q q = -(p - 1)tq$.

Solución formal

Si p o q es igual a 1, se puede tomar $n = 1$; entonces suponga que $p > 1$ y $q > 1$. Por el Teorema 5.3.7, existen enteros s y t tales que $sp + tq = 1$. Como $p > 1$ y $q > 1$, $s \neq 0$ y $t \neq 0$. Más aún, ya sea s o t es negativo. Se puede suponer que $t < 0$. Entonces $s > 0$. Sea $n = -t(p - 1)q$. Se demostrará que se puede lograr un importe postal de $n, n + 1, \ldots, n + p - 1$ usando sólo timbres de p y q centavos.

Ahora

$$\begin{aligned} n + j &= -t(p - 1)q + j(sp + tq) \\ &= (js)p + (-t(p - 1) + jt)q. \end{aligned}$$

Si $0 \leq j \leq p - 1$, entonces

$$-t(p - 1) + jt \geq -t(p - 1) + t(p - 1) = 0.$$

Por lo tanto, se puede lograr un importe postal de $n + j$, $0 \leq j \leq p - 1$, usando js timbres de p centavos y $-t(p - 1) + jt$ timbres de q centavos.

Por último, se usa inducción para demostrar que se puede componer un importe postal de n centavos o más usando sólo timbres de p y q centavos. Los pasos base son $n, n+1, \ldots, n+p-1$. Suponga que $k \geq n+p$ y que se puede tener un importe de m que satisface $n \leq m < k$. En particular, se puede tener un importe de $k-p$. Se agrega un timbre de p centavos para tener el importe de k. Esto completa el paso inductivo.

Resumen de las técnicas de solución de problemas

- Se busca un problema similar. Ya habíamos estudiado un problema de importe postal en el ejemplo 1.8.1.
- Se intenta usar algunas ideas del problema similar. Para resolver este problema, se pudo modificar la prueba por inducción del ejemplo 1.8.1.
- Algunas veces la notación, las definiciones o el contexto sugiere algo útil. En este problema, la ecuación del importe postal $n_p p + n_q q$ tenía una forma similar a la del máximo común divisor $\text{mcd}(p, q) = sp + tq$. Combinar estas ecuaciones fue crucial para probar los casos base.
- No se preocupe al hacer una suposición. Si resulta que una suposición no se cumple, a veces se puede modificar de manera que la nueva suposición sea correcta. En este problema, se supuso que si se podía lograr el importe de n centavos usando n_p timbres de p centavos y n_q timbres de q centavos, se podía tener un importe de $(n+1)$ centavos usando $n_p + s$ timbres de p centavos y $n_q + t$ timbres de q centavos. Pudimos forzar que esta última afirmación fuera cierta eligiendo $n_q \geq -t$.

Ejercicios

1. Demuestre que si $\text{mcd}(p, q) > 1$, es falso que existe n tal que para toda $k \geq n$, se puede lograr el importe postal de k usando sólo timbres de p y q centavos.

5.4 ➜ El sistema criptográfico de llave pública RSA

WWW

La **criptología** es el estudio de los **sistemas** llamados **criptográficos** o **criptosistemas**, para las comunicaciones seguras. En un sistema criptográfico, el remitente transforma el mensaje antes de trasmitirlo, con la esperanza de que sólo los receptores autorizados puedan reconstruir el mensaje original (es decir, el mensaje antes de transformarlo). Se dice que el remitente envía un mensaje **cifrado** o **encriptado**, en tanto que el receptor **descifra** o **desencripta** el mensaje. Si el sistema criptográfico es seguro, las personas no autorizadas no podrán descubrir la técnica para encriptar, de manera que si leen el mensaje cifrado, no podrán descifrarlo. Los sistemas criptográficos son importantes para las grandes organizaciones (por ejemplo, el gobierno y la milicia), los negocios basados en Internet y los individuos. Por ejemplo, si se envía el número de una tarjeta de crédito por Internet, es importante que sólo el receptor al que se dirige pueda leerlo. En esta sección, se presentan algunos algoritmos que apoyan la seguridad de las comunicaciones.

En uno de los sistemas más antiguos y sencillos, tanto el remitente como el receptor tienen una clave que define un carácter sustituto para cada carácter potencial que se pueda enviar. Más aún, el remitente y el receptor no revelan la clave. Se dice que estas claves son *privadas*.

Ejemplo 5.4.1 ▶

Si una llave se define como

el carácter:	`ABCDEFGHIJKLMNOPQRSTUVWXYZ`
se sustituye por:	`EIJFUAXVHWP GSRKOBTQYDMLZNC`

El mensaje ENVÍA DINERO sería cifrado como `ARMWIEUWRATK`. El mensaje cifrado `UWRATKEARMWIUK` se descifraría como `DINERO ENVIADO`. ◀

Los sistemas sencillos como el del ejemplo 5.4.1 se pueden penetrar o averiguar con facilidad ya que ciertas letras (como E en inglés) y combinaciones de letras (como ER en inglés) aparecen con más frecuencia que otras. Otro problema con las llaves privadas en general es que deben enviarse por un medio seguro al remitente y el receptor antes de que se envíe el mensaje. El resto de esta sección se dedica al sistema **criptográfico RSA de llave pública**, llamado así en honor de sus inventores, Ronald L. Rivest, Adi Shamir y Leonard M. Adleman, que se piensa que es seguro. En el sistema RSA, cada participante hace pública una llave de cifrado y oculta la llave de descifrado. Para enviar un mensaje, basta con buscar la llave de cifrado en la tabla pública distribuida. El receptor descifra el mensaje usando la llave de descifrado oculta.

En el sistema RSA, los mensajes se representan como números. Por ejemplo, cada carácter se representa como un número. Si un espacio en blanco se representa como 1, A como 2, B como 3, etcétera. El mensaje `ENV A DINERO` se representaría como 6, 15, 23,

10, 2, 1, 5, 10, 15, 6, 19, 16. Si se desea, los enteros se pueden combinar en un solo entero

```
061523100201051015061916
```

(observe que se agregaron ceros a la izquierda para todos los números de un dígito).

A continuación se describe cómo trabaja un sistema RSA, se presenta un ejemplo concreto y se analiza por qué funciona. Cada receptor potencial elige dos primos p y q y calcula $z = pq$. Puesto que la seguridad del sistema RSA se basa en la incapacidad de otros de conocer el valor de z para descubrir los números p y q, es común que p y q se elijan de manera que cada uno tenga 100 o más dígitos. Después el receptor potencial calcula $\phi = (p - 1)(q - 1)$ y elige un entero n tal que el mcd$(n, \phi) = 1$. En la práctica, n suele ser primo. El par z, n se hace público. Por último, el receptor potencial calcula el número único s, $0 < s < \phi$, que satisface $ns \bmod \phi = 1$. (Una manera eficiente de calcular s se da en la sección 5.3). El número s se guarda en secreto y se usa para descifrar los mensajes.

Para enviar el entero a, $0 \leq a \leq z - 1$, al propietario de la llave pública z, n, el remitente calcula $c = a^n \bmod z$ y envía c. (El algoritmo 5.2.19 proporciona una manera eficiente de calcular $a^n \bmod z$). Para descifrar el mensaje, el receptor calcula $c^s \bmod z$, que se puede demostrar que es igual a a.

Ejemplo 5.4.2 ▶

Suponga que se elige $p = 23$, $q = 31$ y $n = 29$. Entonces $z = pq = 713$ y $\phi = (p - 1)(q - 1) = 660$. Ahora $s = 569$ ya que $ns \bmod \phi = 29 \cdot 569 \bmod 660 = 16501 \bmod 660 = 1$. El par z, $n = 713$, 29 se hace público.

Para trasmitir $a = 572$ al dueño de la llave pública 713, 29, el remitente calcula $c = a^n \bmod z = 579^{29} \bmod 713 = 113$. El receptor calcula $c^s \bmod z = 113^{569} \bmod 713 = 572$ para poder descifrar el mensaje. ◀

El resultado principal que hace que funcione el ciframiento y desciframiento es que

$$a^n \bmod z = a \qquad \text{para todo } 0 \leq a < z \text{ y } u \bmod \phi = 1$$

(vea la demostración en [Cormen: Teorema 31.36, p. 885]). Usando este resultado y el Teorema 5.2.17, se demuestra que el desciframiento produce el resultado correcto. Como $ns \bmod \phi = 1$,

$$c^s \bmod z = (a^n \bmod z)^s \bmod z = (a^n)^s \bmod z = a^{ns} \bmod z = a.$$

La seguridad del sistema RSA para cifrar se basa principalmente en el hecho de que, hasta ahora, no se cuenta con un algoritmo eficiente para factorizar enteros; es decir, no se conoce un algoritmo para factorizar enteros de d dígitos en tiempo polinomial, $O(d^k)$. Entonces, si se seleccionan primos p y q suficientemente grandes, es impráctico calcular la factorización $z = pq$. Si una persona que intercepta un mensaje pudiera encontrar la factorización, podría descifrar el mensaje igual que el receptor autorizado. Hasta ahora, no se conoce un método práctico para factorizar enteros con 200 dígitos o más, de manera que si p y q se eligen cada uno con 100 dígitos o más, pq tendrá alrededor de 200 dígitos o más, lo que parece lograr que el sistema RSA sea seguro.

La primera descripción del sistema criptográfico RSA se encuentra en la columna de Matin Gardner, en el número de febrero de 1977 de *Scientific American* (vea [Gardner, 1977]). Incluyó en esta columna un mensaje encriptado usando la clave z, n, donde z era el producto de primos con 64 y 65 dígitos, y $n = 9007$, además de una oferta de $100 a la primera persona que descifrara el código. Cuando se escribió el artículo, se estimaba que tomaría 40 mil billones de años factorizar z. De hecho, en abril de 1994, Arjen Lenstra, Paul Leyland, Michaek Graff y Derek Atkins, con la ayuda de 600 voluntarios de 25 países, usando más de 1600 computadoras, factorizaron z (vea [Taubes]). El trabajo se coordinó por Internet.

Otra manera posible de interceptar y descifrar un mensaje sería tomar la raíz n de $c \bmod z$, donde c es el valor cifrado. Como $c = a^n \bmod z$, la raíz n de $c \bmod z$ daría a, el valor descifrado. De nuevo, no se conoce un algoritmo con tiempo polinomial para calcular las raíces $n \bmod z$. También es concebible que un mensaje se pueda descifrar por algún me-

dio diferente a la factorización de enteros o la obtención de las raíces n mod z. Por ejemplo, a mediados de los 90, Paul Kocher propuso una manera de penetrar en el RSA con base en el tiempo que toma descifrar un mensaje (vea [English]). La idea es que llaves secretas diferentes requieren tiempos diferentes para descifrar los mensajes y, al usar esta información de tiempos, una persona no autorizada quizá sea capaz de descubrir la llave secreta y descifrar el mensaje. Para frustrar estos ataques, quienes están a cargo del RSA han tomado medidas para alterar el tiempo observado para descifrar los mensajes.

Sección de ejercicios de repaso

1. ¿A qué se refiere la "criptología"?

2. ¿Qué es un sistema criptográfico?

3. ¿Qué significa "encriptar o cifrar un mensaje"?

4. ¿Qué significa "desencriptar o descifrar un mensaje"?

5. En el sistema criptográfico RSA de llave pública, ¿cómo se cifra a y se envía al propietario de la llave pública z, n?

6. En el sistema criptográfico de llave pública RSA, ¿cómo se descifra c?

7. ¿En qué se basa la seguridad del sistema de ciframiento RSA?

Ejercicios

1. Cifre el mensaje `PLAZA SÉSAMO` usando la llave del ejemplo 5.4.1.

2. Descifre el mensaje `YKSIEYDQEMWYISWRIQ` usando la llave del ejemplo 5.4.1.

3. Cifre o encripte el mensaje `NO SOY UN BANDIDO` usando la llave del ejemplo 5.4.1.

4. Descifre el mensaje `FDWUIEAGEIVDI` usando la llave del ejemplo 5.4.1.

5. Cifre 333 usando la llave pública 713, 29 del ejemplo 5.4.2.

6. Descifre 411 usando $s = 569$ como en el ejemplo 5.4.2.

En los ejercicios 7 al 11, suponga que se eligen los primos $p = 17$, $q = 23$, y $n = 31$.

7. Calcule z.

8. Calcule ϕ.

9. Calcule s.

10. Cifre 101 usando la llave pública z, n.

11. Descifre 250.

En los ejercicios 12 al 16, suponga que se eligen los primos $p = 59$, $q = 101$ y $n = 41$.

12. Calcule z.

13. Calcule ϕ.

14. Calcule s.

15. Cifre 584 usando la llave pública z, n.

16. Descifre 250.

Notas

Una introducción accesible a la teoría elemental de números se encuentra en [Niven, 1980]. Un análisis amplio del máximo común divisor, incluyendo antecedentes históricos y otros temas elementales de la teoría de números, se encuentra en [Knuth, 1998a].

Puede encontrar detalles completos del sistema criptográfico RSA en [Cormen]. [Pfleeger] está dedicado a la seguridad en computación.

Repaso del capítulo

Sección 5.1.

1. d divide a n; $d \mid n$
2. d no divide a n: $d \nmid n$
3. d es un divisor o un factor de n
4. Primo
5. Compuesto
6. Teorema fundamental de la aritmética: cualquier entero mayor que 1 se puede escribir como un producto de primos
7. Divisor común
8. Máximo común divisor
9. Múltiplo común
10. Mínimo común múltiplo

Sección 5.2.

11. Bit
12. Sistema numérico decimal
13. Sistema numérico binario
14. Representación de enteros en la computadora: cuando se representa en binario, el entero positivo n requiere $\lfloor 1 + \lg n \rfloor$ bits
15. Sistema numérico hexadecimal
16. Base de un sistema numérico
17. Conversión de binario a decimal
18. Conversión de decimal a binario
19. Conversión de hexadecimal a decimal
20. Conversión de decimal a hexadecimal
21. Suma de números binarios
22. Suma de números hexadecimales
23. Cálculo de a^n elevando al cuadrado repetidas veces
24. $ab \bmod z = [(a \bmod z)(b \bmod z)] \bmod z$
25. Cálculo de $a^n \bmod z$ elevando al cuadrado repetidas veces

Sección 5.3

26. Algoritmo euclidiano
27. Si el par a, b, $a > b$, requiere $n > 1$ operaciones de módulo cuando se introduce al algoritmo euclidiano, entonces $a \geq f_{n+2}$ y $b \geq f_{n+1}$, donde $\{f_n\}$ denota la sucesión de Fibonacci.
28. Si enteros en el intervalo de 0 a m, $m \geq 8$, ambos diferentes de cero, se introducen al algoritmo euclidiano, entonces se requieren cuando mucho

$$\log_{3/2} \frac{2m}{3}$$

operaciones de módulo.
29. Si a y b son enteros no negativos, ambos diferentes de cero, existen enteros s y t tales que $\text{mcd}(a, b) = sa + tb$.
30. Cálculo de s y t tales que $\text{mcd}(a, b) = sa + tb$ usando el algoritmo euclidiano
31. Cálculo del módulo inverso de un entero

Sección 5.4

32. Criptología
33. Sistema criptográfico
34. Encriptar o cifrar un mensaje
35. Desencriptar o descifrar un mensaje
36. Sistema criptográfico RSA de llave pública: para cifrar a y enviarlo al propietario de la llave pública z, n, calcule $c = a^n \bmod z$ y envíe c. Para descifrar el mensaje, calcule $c^s \bmod z$, que se puede demostrar que es igual a a.
37. La seguridad del sistema de ciframiento RSA se basa en el hecho de que, por ahora, no se conoce un algoritmo eficiente para factorizar enteros.

Autoevaluación del capítulo

Sección 5.1

1. Siga el algoritmo 5.1.8 para el dato $n = 539$
2. Encuentre la factorización prima de 539.
3. Encuentre $\text{mcd}(2 \cdot 5^2 \cdot 7^2 \cdot 13^4, 7^4 \cdot 13^2 \cdot 17)$.
4. Encuentre $\text{mcm}(2 \cdot 5^2 \cdot 7^2 \cdot 13^4, 7^4 \cdot 13^2 \cdot 17)$.

Sección 5.2

5. Escriba el número binario 10010110 en decimal.
6. Escriba el número decimal 430 en binario y hexadecimal.
7. Siga el algoritmo 5.2.16 para el valor $n = 30$.
8. Siga el algoritmo 5.2.19 para los valores $a = 50$, $n = 30$, $z = 11$.

Sección 5.3

9. Use el algoritmo euclidiano para encontrar el máximo común divisor de los enteros 396 y 480.

10. Dado que $\log_{3/2} 100 = 11.357747$, proporcione una cota superior para el número de operaciones de módulo requeridas por el algoritmo euclidiano para enteros en el intervalo de 0 a 100,000,000.

11. Utilice el algoritmo euclidiano para encontrar enteros s y t que satisfacen $s \cdot 396 + t \cdot 480 = \text{mcd}(396, 480)$.

12. Demuestre que $\text{mcd}(196, 425) = 1$ y encuentre el inverso s de 196 módulo 425 que satisface $0 < s < 425$.

Sección 5.4

En los ejercicios 13 al 16, suponga que se eligen los primos $p = 13$, $q = 17$ y $n = 19$.

13. Calcule z y ϕ.

14. Calcule s.

15. Cifre 144 usando la llave pública z, n.

16. Descifre 28.

Ejercicios para computadora

1. Ponga en práctica, como programa, el algoritmo 5.1.8 probando si un entero positivo es primo.

2. Escriba un programa que haga conversiones entre decimal, hexadecimal y octal.

3. Escriba un programa que sume números binarios.

4. Escriba un programa que sume números hexadecimales.

5. Escriba un programa que sume números octales.

6. Ponga en práctica, como programa, el algoritmo 5.2.16 de elevar a un exponente elevando al cuadrado repetidas veces.

7. Ponga en práctica, como programa, el algoritmo 5.2.19 de elevar a un exponente mod z.

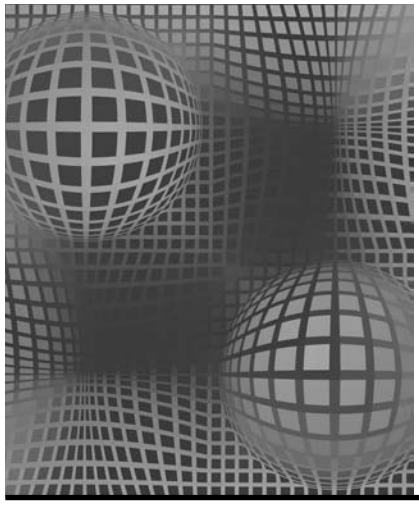
8. Escriba un programa recursivo y otro no recursivo para calcular el máximo común divisor. Compare los tiempos requeridos por los programas.

9. Escriba un programa que, dados los enteros no negativos a y b, ambos diferentes de cero, calcule los enteros s y t que satisfacen

$$\text{mcd}(a, b) = sa + tb.$$

10. Escriba un programa que, dados los enteros $n > 0$ y $\phi > 1$, $\text{mcd}(n, \phi) = 1$, calcule el inverso de n mod ϕ.

11. Ponga en práctica el sistema criptográfico de llave pública RSA.



Capítulo 6
MÉTODOS DE CONTEO Y EL PRINCIPIO DEL PALOMAR

Existe sólo un número dado de manos en una baraja.

DE SHANE

En muchos problemas discretos nos enfrentamos al problema de contar. Por ejemplo, en la sección 4.3 se vio que para estimar el tiempo de corrida de un algoritmo, era necesario contar el número de veces que se ejecutaban ciertos pasos o ciclos. Contar, también tiene un papel crucial en la teoría de probabilidad. En virtud de la importancia del conteo, se ha desarrollado una variedad de ayudas útiles, algunas bastante elaboradas. En este capítulo se desarrollan varias técnicas para contar. Dichas técnicas resultan útiles para derivar el teorema del binomio. El capítulo concluye con un análisis del principio del palomar, que con frecuencia permite probar la existencia de un objeto con ciertas propiedades.

6.1 ➜ Principios básicos

WWW

El menú de Comida Rápida de Kay se muestra en la figura 6.1.1. Como se observa, contiene dos entremeses, tres platos fuertes y cuatro bebidas. ¿Cuántas comidas diferentes están formadas por un plato fuerte y una bebida?

Si se listan todas las comidas posibles que consisten en un plato fuerte y una bebida,

HT, HM, HC, HR, CT, CM, CC, CR, FT, FM, FC, FR,

se ve que existen 12 comidas diferentes. (La comida que consiste en un plato fuerte cuya primera letra es X y una bebida cuya primera letra es Y se denota por XY. Por ejemplo, CR se refiere a una comida que consiste en carne asada y raspado de sabor). Observe que se tienen tres platos fuertes y cuatro bebidas, lo que da $12 = 3 \cdot 4$.

† Estas secciones se pueden omitir sin pérdida de continuidad.

ENTREMÉS

Nachos 2.15

Salami especial 1.90

PLATO FUERTE

Hamburguesa 3.25

Carne asada 3.65

Filete de pescado 3.15

BEBIDAS

Té 70

Malteada 85

Cola 75

Refresco de sabor 75

Figura 6.1.1 Comida Rápida de Kay.

Existen 24 comidas posibles de un entremés, un plato fuerte y una bebida.

NHT, NHM, NHC, NHR, NCT, NCM, NCC, NCR,
NFT, NFM, NFC, NFR, SHT, SHM, SHC, SHR,
SCT, SCM, SCC, SCR, SFT, SFM, SFC, SFR.

(La comida que consiste en un entremés cuya primera letra es *X*, un plato fuerte cuya primera letra es *Y* y una bebida cuya primera letra es *Z* se denota por *XYZ*). Observe que se ofrecen dos entremeses, tres platos fuertes y cuatro bebidas, y que $24 = 2 \cdot 3 \cdot 4$.

En cada uno de estos ejemplos se encuentra que el número total de comidas es igual al producto de los números de cada categoría. Estos ejemplos ilustran el **principio de la multiplicación**.

Principio de la multiplicación

Si una actividad se puede construir en t pasos sucesivos y el paso 1 *se puede hacer de* n_1 *maneras, el paso 2 se puede realizar de* n_2 *maneras, . . . , y el paso t de* n_t *maneras, entonces el número de actividades posibles diferentes es* $n_1 \cdot n_2 \cdot \cdot \cdot n_t$.

En el problema de contar el número de comidas que consisten en un plato fuerte y una bebida, el primer paso es "seleccionar el plato fuerte" y el segundo paso es "seleccionar la bebida". Así, $n_1 = 3$ y $n_2 = 4$ y, por el principio de la multiplicación, el número total de comidas es $3 \cdot 4 = 12$. La figura 6.1.2. muestra por qué se multiplica 3 por 4; se tienen tres grupos de cuatro objetos,

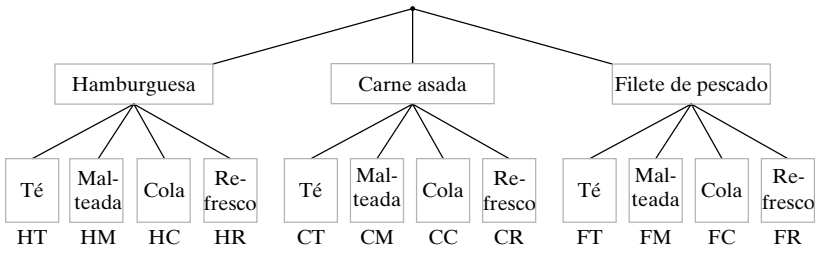


Figura 6.1.2 Ilustración del principio de la multiplicación.

En resumen, el principio de la multiplicación afirma que, cuando una actividad se construye en pasos sucesivos, se multiplican los números de maneras de realizar cada paso.

Ejemplo 6.1.1 ▶

¿Cuántas comidas de un plato fuerte y una bebida *opcional* están disponibles en Comida Rápida de Kay?

Una comida consistente en un plato fuerte y una bebida opcional se forma mediante un proceso de dos pasos. El primer paso es "seleccionar el plato fuerte" y el segundo es "seleccionar una bebida opcional". Existen $n_1 = 3$ maneras de seleccionar el plato fuerte (hamburguesa, carne asada, filete de pescado) y $n_2 = 5$ maneras de seleccionar la bebida opcional (té, malteada, cola, refresco, ninguno). Por el principio de la multiplicación, existen $3 \cdot 5 = 15$ comidas. Como confirmación, he aquí una lista de las 15 comidas (N = sin bebida):

HT, HM, HC, HR, HN, CT, CM, CC, CR, CN, FT, FM, FC, FR, FN. ◀

Ejemplo 6.1.2 ▶

Virus Melissa

A fines de los 90, un virus de computadora llamado Melissa causó estragos al acabar con los recursos del sistema. El virus se esparció por un correo electrónico que contenía un archivo adjunto de procesador de texto con un macro maligno. Cuando se abría el documento, el macro reenviaba el mensaje junto con el archivo del documento a las primeras 50 direcciones obtenidas de la libreta de direcciones del usuario. Cuando se recibían estas copias y se abrían, el macro de nuevo reenviaba el mensaje por correo electrónico y el documento de procesador de textos, y así sucesivamente. El virus causó problemas creando mensajes más rápido de lo que podían enviarse. Los mensajes listos para enviarse se almacenaban temporalmente en un disco. Si el disco se llenaba, el sistema podía caer en bloqueo permanente (congelarse o lo que se conoce habitualmente como "inhibirse") o incluso descomponerse.

Después de que el virus enviaba el correo a las primeras 50 direcciones, cada uno de esos receptores enviaba entonces el correo a 50 direcciones. Por el principio de la multiplicación, se tenían $50 \cdot 50 = 2500$ receptores más. Cada uno de ellos enviaba el correo a 50 direcciones. De nuevo, por el principio de la multiplicación, ahora había $50 \cdot 50 \cdot 50 = 125{,}000$ receptores adicionales. Después de más iteraciones, habría $50 \cdot 50 \cdot 50 \cdot 50 = 6{,}250{,}000$ receptores adicionales. Así, después de sólo cuatro iteraciones se habían enviado

$$6{,}250{,}000 + 125{,}000 + 2500 + 50 + 1 = 6{,}377{,}551$$

copias del mensaje. ◀

Ejemplo 6.1.3 ▶

a) ¿Cuántas cadenas de longitud 4 se pueden formar usando las letras *ABCDE* si no se aceptan repeticiones?
b) ¿Cuántas cadenas del inciso *a)* comienzan con la letra *B*?
c) ¿Cuántas cadenas del inciso *a)* no comienzan con la letra *B*?

a) Se usa el principio de la multiplicación. Una cadena de longitud 4 se construye en cuatro pasos sucesivos: se elige la primera letra; se elige la segunda; se elige la tercera; y se elige la cuarta letra. La primera letra se puede seleccionar de cinco maneras. Una vez elegida la primera letra, la segunda se puede elegir de cuatro maneras. Cuando se elige la segunda letra, la tercera se puede elegir de tres maneras. Una vez seleccionada la tercera letra, la cuarta se puede seleccionar de dos maneras. Por el principio de la multiplicación, existen

$$5 \cdot 4 \cdot 3 \cdot 2 = 120$$

cadenas.

b) Las cadenas que comienzan con la letra *B* se pueden construir en cuatro pasos sucesivos: se elige la primera, la segunda, la tercera y la cuarta letra. La primera letra (*B*) se puede escoger de una manera, la segunda letra de cuatro maneras, la

tercera de tres maneras y la cuarta de dos. Entonces, por el principio de la multiplicación, existen

$$1 \cdot 4 \cdot 3 \cdot 2 = 24$$

cadenas que comienzan con la letra B.

c) El inciso *a)* muestra que existen 120 cadenas de longitud 4 que se pueden formar usando las letras *ABCDE*, y el inciso *b)* prueba que 24 de ellas comienzan con la letra B. Se deduce que existen

$$120 - 24 = 96$$

cadenas que no comienzan con la letra B. ◀

Ejemplo 6.1.4 ▶

En una fotografía digital, deseamos codificar la cantidad de luz en cada punto como una cadena de ocho bits. ¿Cuántos valores son posibles en un punto?

Una codificación de ocho bits se puede construir en ocho pasos sucesivos: se selecciona el primer bit; el segundo bit; . . . ; el octavo bit. Como hay dos maneras de elegir cada bit, por el principio de la multiplicación, el número total de codificaciones de ocho bits es

$$2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 2^8 = 256. \quad ◀$$

Se dará una prueba usando el principio de la multiplicación de que un conjunto con n elementos tiene 2^n subconjuntos. Con anterioridad, se dio una prueba de este resultado usando inducción matemática (Teorema 2.1.6).

Ejemplo 6.1.5 ▶

Use el principio de la multiplicación para demostrar que un conjunto $\{x_1, \ldots, x_n\}$ de n elementos tiene 2^n subconjuntos.

Un subconjunto se construye en n pasos sucesivos: se elige o no se elige x_1; se elige o no x_2; . . . ; se elige o no x_n. Cada paso se puede realizar de dos maneras. Entonces, el número de subconjuntos posibles es

$$\underbrace{2 \cdot 2 \cdots 2}_{n \text{ factores}} = 2^n. \quad ◀$$

Ejemplo 6.1.6 ▶

Sea X un conjunto de n elementos. ¿Cuántos pares ordenados (A, B) satisfacen $A \subseteq B \subseteq X$?

Se tiene un par ordenado (A, B) que satisface $A \subseteq B \subseteq X$; se ve que cada elemento en X está exactamente en uno de A, $B - A$ o en $X - B$. Inversamente, si se asigna cada elemento de X a uno de los tres conjuntos A (y, por suposición, también a B y a X), $B - A$ (y, por suposición, también a X), o a $X - B$, se obtiene un par ordenado único (A, B) que satisface $A \subseteq B \subseteq X$. Entonces el número de pares ordenados (A, B) que satisfacen $A \subseteq B \subseteq X$ es igual al número de maneras para asignar a los elementos de X a los tres conjuntos A, $B - A$ y $X - B$. Se pueden hacer tales asignaciones mediante el siguiente proceso de n pasos: se asigna el primer elemento de X a uno de A, $B - A$, $X - B$; se asigna el segundo elemento de X a uno de A, $B - A$, $X - B$; . . . ; se asigna el n-ésimo elemento de X a uno de A, $B - A$, $X - B$. Como cada paso se puede hacer de tres maneras, el número de pares ordenados (A, B) que satisfacen $A \subseteq B \subseteq X$ es

$$\underbrace{3 \cdot 3 \cdots 3}_{n \text{ factores}} = 3^n. \quad ◀$$

Ahora se ilustrará el principio de la suma mediante un ejemplo y después se presentará el principio.

Ejemplo 6.1.7 ▶

¿Cuántas cadenas de ocho bits comienzan con 101 o con 111?

Una cadena de ocho bits que comienza con 101 se puede construir en cinco pasos sucesivos: se selecciona el cuarto bit; se selecciona el quinto bit; . . . ; se selecciona el octavo bit. Como cada uno de los cinco bits se puede seleccionar de dos maneras, por el principio de la multiplicación, existen

$$2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 2^5 = 32$$

cadenas de ocho bits que comienzan con 101. El mismo argumento se utiliza para mostrar que existen 32 cadenas de ocho bits que comienzan con 111. Como hay 32 cadenas de 8 bits que comienzan con 101 y 32 cadenas de 8 bits que comienzan con 111, existen $32 + 32 = 64$ cadenas de 8 bits que comienzan con 101 o 111. ◀

En el ejemplo 6.1.7 se sumaron los números de cadenas de 8 bits (32 y 32) de cada tipo para determinar el resultado final. El **principio de la suma** dice cuándo sumar para calcular el número total de posibilidades.

Principio de la suma

Suponga que $X_1, \ldots, X_t$ son conjuntos y que el i-ésimo conjunto X_i tiene n_i elementos. Si $\{X_1, \ldots, X_t\}$ es una familia de conjuntos ajenos por pares (es decir, si $i \neq j$, $X_i \cap X_j = \emptyset$}, el número de elementos posibles que se puede seleccionar de X_1 o X_2 o . . . o X_t es

$$n_1 + n_2 + \cdots + n_t.$$

(*De manera equivalente, la unión $X_1 \cup X_2 \cup \cdots \cup X_t$ contiene $n_1 + n_2 + \cdots + n_t$ elementos*).

En el ejemplo 6.1.7, X_1 podría denotar el conjunto de cadenas de 8 bits que comienzan con 101 y X_2 el conjunto de cadenas de 8 bits que comienzan con 111. Como X_1 es ajeno a X_2, de acuerdo con el principio de la suma, el número de cadenas de 8 bits de cualquier tipo, que es el número de elementos en $X_1 \cup X_2$, es $32 + 32 = 64$.

El principio de la suma se resume diciendo que se suma el número de elementos en cada subconjunto cuando es posible descomponer los elementos que se cuentan en subconjuntos ajenos.

Si se trata de contar objetos que se construyen en pasos sucesivos, se usa el principio de la multiplicación. Si se tienen conjuntos ajenos de objetos y se desea saber el número total de objetos, se usa el principio de la suma. Es importante reconocer cuándo aplicar cada principio. Esta habilidad se adquiere con la práctica y el análisis cuidadoso de cada problema.

Esta sección termina con ejemplos que ilustran ambos principios de conteo.

Ejemplo 6.1.8 ▶

¿De cuántas maneras se pueden seleccionar dos libros de temas diferentes entre cinco libros de computación distintos, tres libros de matemáticas diferentes y dos libros de arte distintos?

Mediante el principio de la multiplicación, se encuentra que podemos seleccionar dos libros, uno de computación y uno de matemáticas, de $5 \cdot 3 = 15$ maneras. De forma similar, podemos seleccionar dos libros, uno de computación y uno de arte, de $5 \cdot 2 = 10$ maneras, y podemos seleccionar dos libros, uno de matemáticas y uno de arte, de $3 \cdot 2 = 6$ maneras. Como estos conjuntos de selecciones son ajenas por pares, podemos usar el principio de la suma para concluir que existen

$$15 + 10 + 6 = 31$$

maneras de seleccionar dos libros con temas diferentes entre los libros de computación, matemáticas y arte. ◀

Ejemplo 6.1.9 ▶

Un comité de seis personas, compuesto por Alicia, Benjamín, Consuelo, Adolfo, Eduardo y Francisco, debe seleccionar un presidente, secretario y tesorero.

a) ¿De cuántas maneras pueden hacer esto?

b) ¿De cuántas maneras pueden hacerlo si Alicia o Benjamín debe ser el presidente?

c) ¿De cuántas maneras pueden hacerlo si Eduardo debe ocupar uno de los puestos?

d) ¿De cuántas maneras pueden hacerlo si tanto Adolfo como Francisco deben ocupar un puesto?

a) Se usa el principio de la multiplicación. Se selecciona a los directivos en tres pasos sucesivos: se elige el presidente, se elige el secretario, se elige el tesorero. El

presidente se puede elegir de seis maneras. Una vez elegido, el secretario se puede elegir de cinco maneras. Después de elegir al presidente y el secretario, el tesorero se puede seleccionar de cuatro maneras. Por lo tanto, el número total de posibilidades es

$$6 \cdot 5 \cdot 4 = 120.$$

b) Con un argumento como el del inciso *a)*, si Alicia es presidente, se tienen $5 \cdot 4 = 20$ maneras de seleccionar los puestos restantes. De igual manera, si Benjamín es presidente, existen 20 maneras de seleccionar los puestos restantes. Como estos casos son ajenos, por el principio de la suma, existen

$$20 + 20 = 40$$

posibilidades.

c) [Primera solución] Con un argumento como el del inciso *a)*, si Eduardo es presidente, se tienen 20 maneras de elegir los puestos restantes. De forma similar, si Eduardo es secretario, hay 20 posibilidades, y si Eduardo es tesorero, se tienen 20 posibilidades. Como estos tres casos son ajenos por pares, por el principio de la suma se tienen

$$20 + 20 + 20 = 60$$

posibilidades.

[Segunda solución] Considere que la actividad de asignar a Eduardo y otros dos para los puestos está compuesta por tres pasos sucesivos: asignar a Eduardo a un puesto, asignar el puesto más alto que queda, asignar el último puesto. Existen tres maneras de asignar a Eduardo a un puesto. Una vez asignado, existen cinco maneras de asignar el puesto más alto que queda. Una vez asignados estos dos puestos, hay cuatro maneras de asignar el último puesto. Por el principio de la multiplicación, existen

$$3 \cdot 5 \cdot 4 = 60$$

posibilidades.

d) Considere que la actividad de asignar a Adolfo, Francisco y otra persona a los puestos se compone de tres pasos sucesivos: asignar a Adolfo, a Francisco y el puesto que queda. Existen tres maneras de asignar a Adolfo. Una vez asignado, hay dos maneras de asignar a Francisco. Una vez asignados Adolfo y Francisco, hay cuatro maneras de asignar el último puesto. Por el principio de la multiplicación, existen

$$3 \cdot 2 \cdot 4 = 24$$

posibilidades. ◀

Sugerencias para resolver problemas

La clave para resolver problemas en esta sección es determinar cuándo usar el principio de la multiplicación y cuándo el principio de la suma. Utilice el principio de la multiplicación cuando tenga un proceso paso por paso para *construir una actividad*. Por ejemplo, para diseñar una comida del menú de Comida Rápida de Kay (figura 6.1.1) que consista en un entremés, un plato fuerte y una bebida, se requiere un proceso de tres pasos:

1. Elegir un entremés.
2. Elegir un plato fuerte.
3. Elegir una bebida.

El número de actividades diferentes posibles es el producto del número de maneras en que se puede realizar cada paso. En este caso, el entremés se puede seleccionar de 2 maneras, un plato fuerte de 3 maneras y una bebida de 4 maneras. Entonces, el número de comidas es $2 \cdot 3 \cdot 4 = 24$.

Utilice el principio de la suma cuando desee contar el número de elementos en un conjunto y pueda dividir el conjunto en subconjuntos que no se traslapan. Suponga, por ejemplo, que se desea contar el número total de artículos disponibles en Comida Rápida de Kay. Como hay 2 entremeses, 3 platos fuertes y 4 bebidas, y ningún artículo pertenece a dos categorías, el número total de artículos disponibles es

$$2 + 3 + 4 = 9.$$

Observe la diferencia entre los dos ejemplos. Para construir una comida compuesta de un entremés, un plato fuerte y una bebida en Comida Rápida de Kay se usa un proceso paso por paso. El tamaño del conjunto de comidas *no* se cuenta dividiendo el conjunto en subconjuntos que no se traslapen. Para contar el número de comidas se usa el principio de la multiplicación. Para contar el número de artículos disponibles en el restaurante de Kay, sólo se suma el número de artículos en cada categoría ya que las categorías son una división natural en subconjuntos que no se traslapan. *No* estamos contando los artículos individuales disponibles usando un proceso paso por paso. Para contar el número total de artículos disponibles, se usa el principio de la adición.

Sección de ejercicios de repaso

1. Enuncie el principio de la multiplicación y dé un ejemplo de su aplicación.

2. Enuncie el principio de la adición y dé un ejemplo de su aplicación.

Ejercicios

Encuentre el número de comidas en Comida Rápida de Kay (figura 6.1.1) que satisfacen las condiciones de los ejercicios 1 al *3*.

1. Un entremés y una bebida

2. Un entremés, un plato fuerte y una bebida opcional

3. Un entremés opcional, un plato fuerte y una bebida opcional

4. Un hombre tiene ocho camisas, cuatro pantalones y cinco pares de zapatos. ¿Cuántos atuendos diferentes son posibles?

5. Las opciones disponibles en un modelo específico de automóvil son cinco colores para el interior, seis colores de exterior, dos tipos de asientos, tres tipos de motor y tres tipos de radio. ¿De cuántas posibilidades diferentes dispone el cliente?

6. El sistema Braille para representar caracteres fue desarrollado a principios del siglo IX por Louis Braille. Los caracteres especiales para el invidente consisten en puntos en relieve. Las posiciones para los puntos se seleccionan en dos columnas verticales de tres puntos cada una. Debe haber al menos un punto en relieve. ¿Cuántos caracteres distintos de Braille puede haber?

7. Comente la siguiente nota del *New York Times*:

> Las camionetas de carga también llaman la atención por el aparente número infinito de maneras de personalizarlas; se necesitan las habilidades matemáticas de Will Hunting para obtener el total de configuraciones. Para comenzar, hay 32 combinaciones de cabinas (estándar, cabina club, cuadrángulo), cajas de carga (6.5 u 8 pies) y motores (3.9 litros V6, 5.2 litros V8, 5.9 litros V8, 5.9 litros V8, 5.9 litros turbo diesel alineado 6, 8 litros V10).

En los ejercicios 8 al 16, se lanzan dos dados, uno azul y otro rojo.

8. ¿Cuántos resultados posibles hay?

9. ¿Cuántos resultados suman 4?

10. ¿Cuantos resultados son dobles? (Un doble ocurre cuando los dos dados muestran el mismo número).

11. ¿Cuántos resultados suman 7 u 11?

12. ¿En cuántos resultados el dado azul muestra 2?

13. ¿En cuántos resultados exactamente un dado muestra 2?

14. ¿Cuántos resultados tienen al menos un dado que muestra 2?

15. ¿En cuántos resultados ningún dado muestra 2?

16. ¿Cuántos resultados dan una suma par?

En los ejercicios 17 al 19, suponga que existen 10 caminos de Oz a Media Tierra y 5 de Media Tierra a la Isla de la Fantasía.

17. ¿Cuántas rutas hay de Oz a la Isla de la Fantasía que pasan por Media Tierra?

18. ¿Cuántos viajes redondos de la forma Oz-Media Tierra-Isla de la Fantasía-Media Tierra-Oz hay?

19. ¿Cuántos viajes redondos de la forma Oz-Media Tierra-Isla de la Fantasía-Media Tierra-Oz hay donde en el viaje de regreso no se invierte la ruta original de Oz a la Isla de la Fantasía?

20. ¿Cuántas placas de automóvil se puede hacer que contengan tres letras seguidas de dos dígitos y si se permite que haya repeticiones? Y ¿si no hay repeticiones?

21. ¿Cuántas cadenas de 8 bits comienzan con 1100?

22. ¿Cuántas cadenas de 8 bits comienzan y terminan con 1?

23. ¿Cuántas cadenas de 8 bits tienen 1 en el segundo o el cuarto bit (o en ambos)?

24. ¿Cuántas cadenas de 8 bits tienen exactamente un 1?

25. ¿Cuántas cadenas de 8 bits tienen exactamente dos unos?

26. ¿Cuántas cadenas de 8 bits tienen al menos un 1?

27. ¿Cuántas cadenas de 8 bits se leen igual al derecho y al revés? (Un ejemplo de tal cadena es 01111110. Estas cadenas se llaman *palíndromos*).

En los ejercicios 28 al 33, un comité de seis personas compuesto por Alicia, Benjamín, Consuelo, Adolfo, Eduardo y Francisco debe elegir un presidente, secretario y tesorero.

28. ¿Cuántas selecciones excluyen a Consuelo?

29. ¿Cuántas selecciones existen en las que ni Benjamín ni Francisco tienen un puesto?

30. ¿Cuántas selecciones existen en las que tanto Benjamín como Francisco tienen un puesto?

31. ¿Cuántas selecciones hay con Adolfo en un puesto y Francisco no?

32. ¿Cuántas selecciones hay que tengan a Adolfo como presidente o que no incluyan a Adolfo?

33. ¿Cuántas selecciones hay donde Benjamín sea presidente o tesorero?

En los ejercicios 34 al 41, las letras ABCDE deben usarse para formar cadenas de longitud 3.

34. ¿Cuántas cadenas se pueden formar si se permiten repeticiones?

35. ¿Cuántas cadenas se pueden formar si no se permiten repeticiones?

36. ¿Cuántas cadenas comienzan con A, cuando hay repeticiones?

37. ¿Cuántas cadenas comienzan con A, si no hay repeticiones?

38. ¿Cuántas cadenas no contienen a la letra A cuando se permiten repeticiones?

39. ¿Cuántas cadenas no contienen a la letra A si no hay repeticiones?

40. ¿Cuántas cadenas contienen a la letra A, si se permiten repeticiones?

41. ¿Cuántas cadenas contienen a la letra A si no se permiten repeticiones?

Los ejercicios 42 al 52 se refieren a los enteros entre 5 y 200, inclusive.

42. ¿Cuántos números hay?

43. ¿Cuántos son pares?

44. ¿Cuántos son impares?

45. ¿Cuántos son divisibles entre 5?

46. ¿Cuántos son mayores que 72?

47. ¿Cuántos consisten en dígitos diferentes?

48. ¿Cuántos contienen el dígito 7?

49. ¿Cuántos no contienen el dígito 0?

50. ¿Cuántos son mayores que 101 y no contienen el dígito 6?

51. ¿Cuántos tienen dígitos en orden estrictamente creciente? (Por ejemplo, 13, 147, 8.)

52. ¿Cuántos son de la forma xyz, donde $0 \neq x < y$ y $y > z$?

53. *a)* ¿De cuántas maneras pueden ser diferentes los meses en que cumplen años cinco personas?

b) ¿Cuántas posibilidades hay para los meses de los cumpleaños de cinco personas?

c) ¿De cuántas maneras pueden por lo menos dos personas entre cinco tener su cumpleaños en el mismo mes?

Los ejercicios 54 al 58 se refieren a un conjunto de cinco libros de computación, tres de matemáticas y dos de arte, todos diferentes.

54. ¿De cuántas maneras pueden arreglarse estos libros en una repisa?

55. ¿De cuántas maneras pueden arreglarse éstos en una repisa si los cinco libros de computación van a la izquierda y los dos de arte a la derecha?

56. ¿De cuántas maneras se pueden arreglar estos libros en una repisa si los cinco de computación van a la izquierda?

57. ¿De cuántas maneras se pueden arreglar estos libros en una repisa si se agrupan todos los libros de la misma disciplina?

★ **58.** ¿De cuántas maneras se pueden arreglar estos libros en una repisa si los dos libros de arte no quedan juntos?

59. En algunas versiones de FORTRAN, un identificador consiste en una cadena de uno a seis caracteres alfanuméricos que comienza con una letra. (Un carácter alfanumérico es una letra de la A a la Z o un dígito del 0 al 9). ¿Cuántos identificadores válidos de FORTRAN existen?

60. Si X es un conjunto de n elementos y Y es un conjunto de m elementos, ¿cuántas funciones existen de X a Y?

★ ***61.*** Existen 10 copias de un libro y una copia de cada uno de otros 10 libros. ¿De cuántas maneras se pueden elegir 10 libros?

62. ¿Cuántos términos hay en la expansión de

$$(x + y)(a + b + c)(e + f + g)(h + i)?$$

★ **63.** ¿Cuántos subconjuntos de un conjunto de $(2n + 1)$ elementos tienen n elementos o menos?

64. ¿Cuántas relaciones antisimétricas existen en un conjunto de n elementos?

65. Si X y Y no son subconjuntos ajenos, no se puede sumar $|X|$ a $|Y|$ para calcular el número de elementos en $X \cup Y$. Pruebe que

$$|X \cup Y| = |X| + |Y| - |X \cap Y|$$

para conjuntos arbitrarios X y Y.

Use el resultado del ejercicio 65 para resolver los ejercicios 66 al 70.

66. ¿Cuántas cadenas de 8 bits comienzan con 100 o el cuarto bit es 1?

67. ¿Cuántas cadenas de 8 bits comienzan con 1 o terminan con 1?

En los ejercicios 68 y 69, un comité de seis personas constituido por Alicia, Benjamín, Consuelo, Adolfo, Eduardo y Francisco debe seleccionar un presidente, secretario y tesorero.

68. ¿Cuántas selecciones existen en las que Benjamín es presidente o Alicia es secretaria?

69. ¿Cuántas selecciones existen en las que Consuelo es presidente o Alicia tiene un puesto?

70. Se lanzan dos dados, uno azul y otro rojo. ¿Cuántos resultados tienen un 3 en el dado azul o una suma par?

71. ¿Cuántos operadores binarios hay en $\{1, 2, \ldots, n\}$?

72. ¿Cuántos operadores binarios conmutativos hay en $\{1, 2, \ldots, n\}$?

Rincón de solución de problemas

Conteo

Problema

Encuentre el número de triadas ordenadas de conjuntos X_1, X_2, X_3 que satisfacen

$$X_1 \cup X_2 \cup X_3 = \{1, 2, 3, 4, 5, 6, 7, 8\}$$
$$\text{y} \quad X_1 \cap X_2 \cap X_3 = \emptyset.$$

Por *triada ordenada*, se entiende que se toma en cuenta el orden de los conjuntos X_1, X_2, X_3. Por ejemplo, las triadas

$$\{1, 2, 3\}, \quad \{1, 4, 8\}, \quad \{2, 5, 6, 7\}$$

y

$$\{1, 4, 8\}, \quad \{1, 2, 3\}, \quad \{2, 5, 6, 7\}$$

se consideran diferentes.

Cómo atacar el problema

Sería bueno comenzar por numerar las triadas, pero hay tantas que sería difícil lograr un panorama general observando unas cuantas de ellas. Se simplificará el problema sustituyendo

$$\{1, 2, 3, 4, 5, 6, 7, 8\}$$

por $\{1\}$. ¿Qué puede ser más sencillo que $\{1\}$? (Bueno, quizá $\emptyset$, ¡pero es demasiado sencillo!). Ahora se pueden numerar todas la triadas ordenadas de conjuntos X_1, X_2, X_3 que satisfacen $X_1 \cup X_2 \cup X_3 = \{1\}$ y $X_1 \cap X_2 \cap X_3 = \emptyset$. Debemos poner un 1 en al menos uno de los conjuntos X_1, X_2, X_3 (para que la unión sea $\{1\}$), pero no debemos poner un 1 en los tres conjuntos X_1, X_2, X_3 (de otra manera, la intersección no sería vacía). Así, 1 estará en exactamente uno o dos de los conjuntos X_1, X_2, X_3. La lista completa de triadas ordenadas es la siguiente:

$$\begin{array}{llll} X_1 = \{1\}, & X_2 = \emptyset, & X_3 = \emptyset; \\ X_1 = \emptyset, & X_2 = \{1\}, & X_3 = \emptyset; \\ X_1 = \emptyset, & X_2 = \emptyset, & X_3 = \{1\}\ ; \\ X_1 = \{1\}, & X_2 = \{1\}, & X_3 = \emptyset; \\ X_1 = \{1\}, & X_2 = \emptyset, & X_3 = \{1\}\ ; \\ X_1 = \emptyset, & X_2 = \{1\}, & X_3 = \{1\}. \end{array}$$

Entonces existen seis triadas ordenadas de conjuntos X_1, X_2, X_3 que satisfacen

$$X_1 \cup X_2 \cup X_3 = \{1\} \qquad \text{y} \qquad X_1 \cap X_2 \cap X_3 = \emptyset.$$

Si subimos un nivel y numeramos todas las triadas ordenadas de conjuntos X_1, X_2, X_3 que satisfacen $X_1 \cup X_2 \cup X_3 = \{1, 2\}$ y $X_1 \cap X_2 \cap X_3 = \emptyset$. Como antes, debemos poner un 1 en al menos uno de los conjuntos X_1, X_2, X_3 (para que el 1 esté en la unión), pero no en los tres conjuntos X_1, X_2, X_3 (ya que la intersección no estaría vacía). Esta vez también debemos poner un 2 en al menos uno de los conjuntos X_1, X_2, X_3 (para que el 2 esté en la unión), pero no debemos poner un 2 en los tres conjuntos X_1, X_2, X_3 (ya que la intersección no estaría vacía). Entonces, cada uno de los elementos 1 y 2 estará justo en uno o dos de los conjuntos X_1, X_2, X_3. Se numeran los conjuntos de manera sistemática para reconocer algún patrón que aparezca. La lista completa de triadas ordenadas se muestra en la tabla que sigue. Por ejemplo, el elemento arriba a la izquierda, $X_1\ X_1$, especifica que hay un 1 en X_1 y un 2 en X_1; por lo tanto, este elemento da la triada ordenada

$$X_1 = \{1, 2\}, \quad X_2 = \emptyset, \quad X_3 = \emptyset.$$

Como se muestra, existen 36 triadas ordenadas de conjuntos X_1, X_2, X_3 que satisfacen

$$X_1 \cup X_2 \cup X_3 = \{1, 2\}$$
$$\text{y} \qquad X_1 \cap X_2 \cap X_3 = \emptyset.$$

Se ve que hay seis maneras de asignar el 1 a los conjuntos X_1, X_2, X_3, lo que explica seis líneas por bloque. De manera similar, existen seis maneras de asignar un 2 a los conjuntos X_1, X_2, X_3, lo que explica seis bloques.

Antes de seguir leyendo, ¿adivina cuántas triadas ordenadas de conjuntos X_1, X_2, X_3 satisfacen

$$X_1 \cup X_2 \cup X_3 = \{1, 2, 3\}$$
$$\text{y} \qquad X_1 \cap X_2 \cap X_3 = \emptyset.$$

1 está en	*2 está en*	*1 está en*	*2 está en*	*1 está en*	*2 está en*
X_1	X_1	X_1	X_2	X_1	X_3
X_2	X_1	X_2	X_2	X_2	X_3
X_3	X_1	X_3	X_2	X_3	X_3
$X_1,\ X_2$	X_1	$X_1,\ X_2$	X_2	$X_1,\ X_2$	X_3
$X_1,\ X_3$	X_1	$X_1,\ X_3$	X_2	$X_1,\ X_3$	X_3
$X_2,\ X_3$	X_1	$X_2,\ X_3$	X_2	$X_2,\ X_3$	X_3
X_1	$X_1,\ X_2$	X_1	$X_1,\ X_3$	X_1	$X_2,\ X_3$
X_2	$X_1,\ X_2$	X_2	$X_1,\ X_3$	X_2	$X_2,\ X_3$
X_2	$X_1,\ X_2$	X_3	$X_1,\ X_3$	X_3	$X_2,\ X_3$
$X_1,\ X_2$	$X_1,\ X_2$	$X_1,\ X_2$	$X_1,\ X_3$	$X_1,\ X_2$	$X_2,\ X_3$
$X_1,\ X_3$	$X_1,\ X_2$	$X_1,\ X_3$	$X_1,\ X_3$	$X_1,\ X_3$	$X_2,\ X_3$
$X_2,\ X_3$	$X_1,\ X_2$	$X_2,\ X_3$	$X_1,\ X_3$	$X_2,\ X_3$	$X_2,\ X_3$

Surge un patrón. Si $X = \{1,2, \ldots, n\}$, existen seis maneras de asignar cada uno de 1, 2, . . . , n a los conjuntos X_1, X_2, X_3. Por el principio de la multiplicación, el número de triadas ordenadas es 6^n.

Cómo encontrar otra solución

Se acaba de encontrar una solución al problema comenzando con un problema más sencillo y después descubriendo y justificando el patrón que surge.

Otro enfoque consiste en buscar un problema similar e imitar su solución. El problema del ejemplo 6.1.6 es similar al que se tiene, pues también es un problema de conteo que se refiere a conjuntos:

> Sea X un conjunto de n elementos. ¿Cuántos pares ordenados (A, B) satisfacen $A \subseteq B \subseteq X$?

(En este punto, sería una buena idea regresar y leer el ejemplo 6.1.6.) La solución a que se llega en el ejemplo 6.1.6 cuenta el número de maneras para asignar elementos de X a exactamente uno de los conjuntos A, $B - A$ o $X - B$.

Podemos resolver nuestro problema tomando un enfoque parecido. Cada elemento de X está exactamente en uno de

$$\overline{X_1} \cap X_2 \cap X_3, \quad X_1 \cap \overline{X_2} \cap X_3, \quad X_1 \cap X_2 \cap \overline{X_3},$$
$$\overline{X_1} \cap \overline{X_2} \cap X_3, \quad \overline{X_1} \cap X_2 \cap \overline{X_3}, \quad X_1 \cap \overline{X_2} \cap \overline{X_3}.$$

Como cada miembro de X se puede asignar a uno de estos conjuntos de seis maneras, por el principio de la multiplicación, el número de triadas ordenadas es 6^8.

Observe que aunque este enfoque para resolver el problema es diferente que el de la sección anterior, el argumento final es en esencia el mismo.

Solución formal

Cada elemento en X está exactamente en uno de

$$Y_1 = \overline{X_1} \cap X_2 \cap X_3, \quad Y_2 = X_1 \cap \overline{X_2} \cap X_3,$$
$$Y_3 = X_1 \cap X_2 \cap \overline{X_3}, \quad Y_4 = \overline{X_1} \cap \overline{X_2} \cap X_3,$$
$$Y_5 = \overline{X_1} \cap X_2 \cap \overline{X_3}, \quad Y_6 = X_1 \cap \overline{X_2} \cap \overline{X_3}.$$

Se puede construir una triada ordenada mediante el siguiente proceso de ocho pasos: se elige j, $1 \leq j \leq 6$ y se pone 1 en Y_j; se elige j, $1 \leq j \leq 6$ y se pone 2 en Y_j; . . . ; se elige j, $1 \leq j \leq 6$ y se pone 8 en Y_j. Por ejemplo, para construir una triada

$$\{1, 2, 3\}, \quad \{1, 4, 8\}, \quad \{2, 5, 6, 7\},$$

primero se elige $j = 3$ y se pone 1 en $Y_3 = X_1 \cap X_2 \cap \overline{X_3}$. Después se selecciona $j = 2$ y se pone 2 en $Y_2 = X_1 \cap X_2 \cap \overline{X_3}$. Las opciones restantes para j son $j = 6, 5, 4, 4, 4, 5$.

Cada selección para j se puede hacer de seis maneras. Por el principio de la multiplicación, el número de triadas ordenadas es

$$6 \cdot 6 \cdot 6 \cdot 6 \cdot 6 \cdot 6 \cdot 6 \cdot 6 = 6^8 = 1{,}679{,}616.$$

Resumen de la técnica de solución de problemas

- Sustituir el problema por un problema más sencillo. Una manera de hacerlo es reducir el tamaño del problema original.
- Numerar directamente los elementos que se van a contar.
- Numerar los elementos de manera sistemática para que surjan patrones.
- Buscar patrones.
- Buscar un problema similar e imitar su solución.

6.2 ➜ Permutaciones y combinaciones

WWW

Hay cuatro candidatos, Samuel, Ignacio, Héctor y Vilma, postulados para el mismo puesto. Para que la posición de los nombres en las boletas de votación no influya en los votantes, es necesario imprimir boletas con los nombres en todos los órdenes posibles. ¿Cuántas boletas diferentes habrá?

Se puede usar el principio de la multiplicación. Una boleta se elabora en cuatro pasos sucesivos: se selecciona el primer nombre de la lista; se selecciona el segundo nombre; se selecciona el tercero; y se selecciona el cuarto nombre de la lista. El primer nombre se puede elegir de cuatro maneras. Una vez elegido el primer nombre, el segundo se puede seleccionar de tres maneras. Cuando se tiene el segundo nombre, el tercero se puede elegir de dos maneras y el cuarto sólo de una manera. Por el principio de la multiplicación, el número de boletas diferentes es

$$4 \cdot 3 \cdot 2 \cdot 1 = 24.$$

Un ordenamiento de los objetos, como los nombres en las boletas, se llama **permutación**.

Definición 6.2.1 ▶

Una permutación de n elementos diferentes $x_1, \ldots, x_n$ es un ordenamiento de los n elementos $x_1, \ldots, x_n$. ◀

Ejemplo 6.2.2 ▶

Existen seis permutaciones de tres elementos. Si se denotan los elementos por A, B, C, las seis permutaciones son

$$ABC, \quad ACB, \quad BAC, \quad BCA, \quad CAB, \quad CBA. \qquad \blacktriangleleft$$

Se encontró que existen 24 maneras de ordenar cuatro candidatos en una boleta; así, hay 24 permutaciones de cuatro objetos. El método que se usó para contar el número de boletas diferentes con cuatro nombres se puede usar para derivar una fórmula para el número de permutaciones de n elementos.

La demostración del siguiente teorema para $n = 4$ se ilustra en la figura 6.2.1.

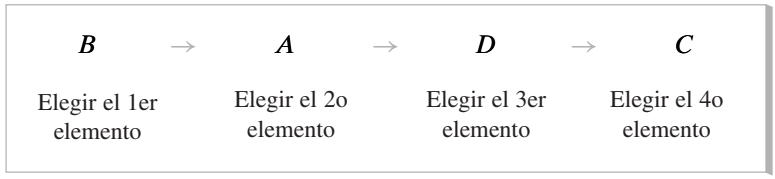


Figura 6.2.1 Prueba del teorema 6.2.3 para $n = 4$. Una permutación de *ABCD* se construye con la elección sucesiva del primer elemento, después el segundo elemento, luego el tercero y por último el cuarto.

Teorema 6.2.3

Existen n! permutaciones de n elementos.

Demostración Se usa el principio de la multiplicación. Una permutación de n elementos se construye en n pasos sucesivos: se elige el primer elemento; se elige el segundo elemento; . . . ; se elige el último elemento. El primer elemento se puede seleccionar de n maneras. Una vez elegido, el segundo elemento se puede seleccionar de $n - 1$ maneras. Una vez elegido, el tercer elemento se puede seleccionar de $n - 2$ maneras, y así sucesivamente. Por el principio de la multiplicación, existen

$$n(n-1)(n-2)\cdots 2 \cdot 1 = n!$$

permutaciones de n elementos.

Ejemplo 6.2.4 ▶

Existen

$$10! = 10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 3{,}628{,}800$$

permutaciones de 10 elementos. ◀

Ejemplo 6.2.5 ▶

¿Cuántas permutaciones de las letras *ABCDEF* contienen la subcadena *DEF*?

Para garantizar la presencia del patrón *DEF* en la subcadena, estas tres letras deben estar juntas en este orden. Las letras restantes, *A*, *B* y *C*, se pueden colocar de forma arbitraria. Podemos pensar en construir permutaciones de las letras *ABCDEF* que contengan el patrón *DEF* con la permutación de cuatro fichas: una con la etiqueta *DEF* y las otras con etiquetas *A*, *B* y *C* (figura 6.2.2). Por el Teorema 6.2.3, existen 4! permutaciones de cuatro objetos. Entonces, el número de permutaciones de las letras *ABCDEF* que contienen la subcadena *DEF* es

$$4! = 24. \qquad \blacktriangleleft$$

Figura 6.2.2 Cuatro fichas para permutar.

Ejemplo 6.2.6 ▶

¿Cuántas permutaciones de las letras *ABCDEF* contienen a las letras *DEF* juntas en cualquier orden?

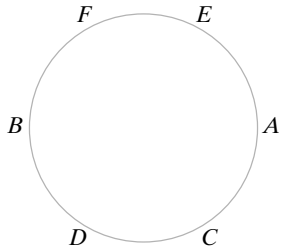
Este problema se resuelve mediante un procedimiento de dos pasos: se selecciona un orden de las letras *DEF*; se construye una permutación de *ABCDEF* que contenga el orden dado de las letras *DEF*. Por el Teorema 6.2.3, el primer paso se puede realizar de $3! = 6$ maneras y, de acuerdo con el ejemplo 6.2.5, el segundo paso se puede realizar de 24 maneras. Por el principio de la multiplicación, el número de permutaciones de las letras *ABCDEF*

que contienen las letras *DEF* juntas en cualquier orden es

$$6 \cdot 24 = 144.$$ ◀

Ejemplo 6.2.7 ▶

¿De cuántas maneras se pueden sentar seis personas alrededor de una mesa circular? Si un arreglo se obtiene de otro haciendo que todos se muevan n asientos en el sentido de las manecillas del reloj, los arreglos se consideran idénticos.



Denotemos a las personas como A, B, C, D, E y F. Como los arreglos obtenidos por rotación se consideran idénticos, es lo mismo si A se sienta en un lugar arbitrario. Para sentar a las otras cinco personas, se pueden ordenar y después sentarlas en ese orden en el sentido de las manecillas del reloj a partir de A. Por ejemplo la permutación *CDBFE* definiría el arreglo de la figura al margen. Como hay $5! = 120$ permutaciones de cinco elementos, hay 120 maneras de sentar a seis personas en una mesa circular.

El mismo argumento se puede usar para demostrar que hay $(n-1)!$ maneras de sentar a n personas alrededor de una mesa circular. ◀

Algunas veces se desea considerar un orden de r elementos seleccionados entre n elementos disponibles, es decir, de los n elementos tomados de r en r. Este ordenamiento se llama **permutación *r***.

Definición 6.2.8 ▶

Una permutación r de n elementos (distintos) $x_1, \ldots, x_n$ *es un ordenamiento de r elementos de* $[x_1, \ldots, x_n]$. El número de permutaciones r de un conjunto de n elementos diferentes se denota por $P(n, r)$. ◀

Ejemplo 6.2.9 ▶

Ejemplos de permutaciones 2 de a, b, c son

$$ab, \quad ba, \quad ca.$$ ◀

Si $r = n$ en la definición 6.2.8, se obtiene un ordenamiento de todos los n elementos. Entonces, una permutación n de n elementos es lo que antes se llamó simplemente permutación. El Teorema 6.2.3 dice que $P(n, n) = n!$. El número $P(n, r)$ de permutaciones r de un conjunto de n elementos cuando $r < n$ se obtiene de la demostración del Teorema 6.2.3. Esta demostración para $n = 6$ y $r = 3$ se ilustra en la figura 6.2.3.

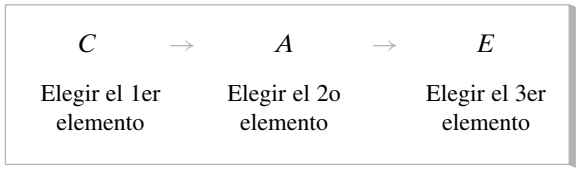


Figura 6.2.3 Prueba del Teorema 6.2.10 para $n = 6$ y $r = 3$. Una permutación r de *ABCDEF* se construye mediante la selección sucesiva del primero, segundo y tercer elementos.

Teorema 6.2.10

El número de permutaciones r de un conjunto de n objetos diferentes es

$$P(n, r) = n(n-1)(n-2)\cdots(n-r+1), \qquad r \le n.$$

Demostración Debe contarse el número de maneras de ordenar r elementos seleccionados de un conjunto de n elementos. El primer elemento se puede elegir de n maneras. Una vez que se elige el primer elemento, el segundo se puede seleccionar de $n-1$ maneras. Continuamos eligiendo elementos hasta que, habiendo elegido el elemento $r-1$, pasamos al elemento r que se puede seleccionar de $n-r+1$ maneras. Por el principio de la multiplicación, el número de permutaciones r de un conjunto de n objetos distintos es

$$n(n-1)(n-2)\cdots(n-r+1).$$

Ejemplo 6.2.11 ▶

De acuerdo con el teorema 6.2.10, el número de permutaciones 2 de $X = \{a, b, c\}$ es

$$P(3, 2) = 3 \cdot 2 = 6$$

Estas seis permutaciones son

$$ab,\ ac,\ ba,\ bc,\ ca,\ cb.$$ ◀

Ejemplo 6.2.12 ▶

¿De cuántas maneras se puede seleccionar el presidente, vicepresidente, secretario y tesorero de un grupo de 10 personas?

Es necesario contar el número de ordenamientos de cuatro personas seleccionadas de un grupo de 10, ya que cada arreglo elige (de manera única) un presidente (primera elección), un vicepresidente (segunda elección), un secretario (tercera elección) y un tesorero (cuarta elección). Por el Teorema 6.2.10, la solución es

$$P(10, 4) = 10 \cdot 9 \cdot 8 \cdot 7 = 5040.$$ ◀

Pudo haberse resuelto el ejemplo 6.2.12 usando directamente el principio de la multiplicación.

También, $P(n, r)$ se puede escribir en términos de factoriales:

$$\begin{aligned} P(n, r) &= n(n-1)\cdots(n-r+1) \\ &= \frac{n(n-1)\cdots(n-r+1)(n-r)\cdots 2 \cdot 1}{(n-r)\cdots 2 \cdot 1} = \frac{n!}{(n-r)!}. \end{aligned} \qquad \textbf{(6.2.1)}$$

Ejemplo 6.2.13 ▶

Usando (6.2.1), la solución del ejemplo 6.2.12 se rescribe como

$$P(10, 4) = \frac{10!}{(10-4)!} = \frac{10!}{6!}.$$ ◀

Ejemplo 6.2.14 ▶

¿De cuántas maneras pueden hacer cola siete marcianos y cinco venusinos si dos venusinos no se paran juntos?

Marcianos y venusinos se pueden formar siguiendo un proceso de dos pasos: formar marcianos; formar venusinos. Los marcianos se pueden formar de $7! = 5040$ maneras. Una vez formados los marcianos (por ejemplo en las posiciones M_1 a M_7), como los venusinos no pueden estar dos juntos, tendrán ocho posiciones posibles para formarse (espacios indicados por guión bajo);

$$\_\, M_1 \,\_\, M_2 \,\_\, M_3 \,\_\, M_4 \,\_\, M_5 \,\_\, M_6 \,\_\, M_7 \,\_\,.$$

Así, los venusinos pueden formarse de $P(8, 5) = 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 = 6720$ maneras. Por el principio de la multiplicación, el número de maneras en que siete marcianos y cinco venusinos puede hacer cola si dos venusinos no pueden estar juntos es

$$5040 \cdot 6720 = 33{,}868{,}800$$ ◀

Ahora se estudiarán las combinaciones. Una selección de objetos que no toma en cuenta el orden se llama **combinación.**

Definición 6.2.15 ▶

Dado un conjunto $X = \{x_1, \ldots, x_n\}$ que contiene n elementos (diferentes),

a) Una *combinación* r de X es una selección no ordenada de r elementos de X (es decir, un subconjunto de X de r elementos).

b) El número de combinaciones r de un conjunto de n elementos distintos se denota por $C(n, r)$ o $\binom{n}{r}$. ◀

Ejemplo 6.2.16 ▶

Un grupo de cinco estudiantes, María, Braulio, Rosa, Amanda y Néstor, ha decidido hablar con la directora del departamento de matemáticas para que ofrezcan más cursos de matemáticas discretas. La directora del departamento ha dicho que hablará con tres de los

estudiantes. ¿De cuántas maneras pueden estos cinco estudiantes elegir tres de ellos para hablar con la directora del departamento?

Al resolver este problema no debe tomarse en cuenta el orden. (Por ejemplo, no hay diferencia si la directora habla con María, Amanda y Néstor o con Néstor, María y Amanda). Con sólo listar las posibilidades, se ve que existen 10 maneras de elegir tres estudiantes de un grupo de cinco para hablar con la directora.

$$MBR, MBA, MRA, BRA, MBN, MRN, BRN, MAN, BAN, RAN.$$

En la terminología de la definición 6.2.15, el número de maneras en que los cinco estudiantes pueden elegir tres de ellos es $C(5, 3)$, el número de combinaciones de 3 de cinco elementos. Se ha encontrado que

$$C(5, 3) = 10. \quad \blacktriangleleft$$

Ahora se derivará la fórmula para $C(n, r)$ contando el número de permutaciones r de un conjunto de n elementos de dos maneras. La primera simplemente usa la fórmula $P(n, r)$. La segunda manera de contar el número de permutaciones de un conjunto de n elementos implica $C(n, r)$. Igualar los dos valores nos permitirá obtener una fórmula para $C(n, r)$.

Podemos construir permutaciones r de un conjunto X de n elementos en dos pasos sucesivos: primero, elegimos una combinación r de X (un subconjunto no ordenado de r elementos); segundo, se ordena. Por ejemplo, para construir una permutación de 2 de $\{a, b, c, d\}$, primero elegimos una combinación de 2 y después lo ordenamos. La figura 6.2.4 muestra cómo se obtienen todas las permutaciones 2 de $\{a, b, c, d\}$ de esta manera. El principio de multiplicación dice que el número de permutaciones r es el producto del número de combinaciones r por el número de ordenamientos de r elementos; es decir,

$$P(n, r) = C(n, r)r!.$$

Por lo tanto,

$$C(n, r) = \frac{P(n, r)}{r!}.$$

El siguiente teorema establece este resultado y da algunas maneras alternativas para escribir $C(n, r)$.

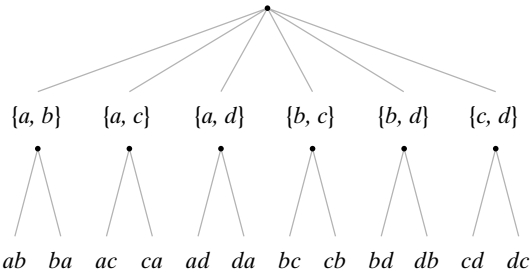


Figura 6.2.4 Permutaciones de 2 elementos de $\{a, b, c, d\}$.

Teorema 6.2.17

El número de combinaciones r de un conjunto de n objetos distintos es

$$C(n, r) = \frac{P(n, r)}{r!} = \frac{n(n-1)\cdots(n-r+1)}{r!} = \frac{n!}{(n-r)!\,r!}, \qquad r \le n.$$

Demostración La demostración de la primera ecuación está dada antes de enunciar el Teorema. Las otras formas de la ecuación se derivan del Teorema 6.2.10 y la ecuación (6.2.1).

Ejemplo 6.2.18 ▶

¿De cuántas maneras se puede seleccionar un comité de tres a partir de un grupo de 10 personas diferentes?

Como un comité es un grupo no ordenado de personas, la respuesta es

$$C(10, 3) = \frac{10 \cdot 9 \cdot 8}{3!} = 120. \quad \blacktriangleleft$$

Ejemplo 6.2.19 ▶ ¿De cuántas maneras se puede seleccionar un comité de dos mujeres y tres hombres de un grupo de cinco mujeres y seis hombres?

Igual que en el ejemplo 6.2.18, se encuentra que las dos mujeres se pueden elegir de

$$C(5, 2) = 10$$

maneras y que los tres hombres se pueden elegir de

$$C(6, 3) = 20$$

maneras. El comité se construye en dos pasos sucesivos: se elige a las mujeres; se elige a los hombres. Por el principio de la multiplicación, el número total de comités es

$$10 \cdot 20 = 200. \quad \blacktriangleleft$$

Ejemplo 6.2.20 ▶ ¿Cuántas cadenas de ocho bits contienen exactamente cuatro unos?

Una cadena de ocho bits que contiene cuatro unos se determina de manera única una vez que se establece qué bits son 1. Esto se puede hacer de

$$C(8, 4) = 70$$

maneras. ◀

Ejemplo 6.2.21 ▶ Una baraja común de 52 cartas consiste en cuatro palos

tréboles, diamantes, corazones y espadas

con 13 denominaciones cada uno

as, de 2 a 10, jack, reina y rey.

a) ¿Cuántas manos de póquer (sin ordenar) de cinco cartas, seleccionadas de una baraja común de 52 cartas, existen?
b) ¿Cuántas manos de póquer contienen cartas todas del mismo palo?
c) ¿Cuántas manos de póquer contienen tres cartas de una denominación y dos de una segunda denominación?

a) La respuesta está dada por la fórmula de las combinaciones

$$C(52, 5) = 2{,}598{,}960.$$

b) Una mano en la que todas las cartas son del mismo palo se puede construir en dos pasos sucesivos: seleccionamos un palo; seleccionamos cinco cartas del palo elegido. El primer paso se puede realizar de cuatro maneras y el segundo de $C(13, 5)$ maneras. Por el principio de la multiplicación, la respuesta es

$$4 \cdot C(13, 5) = 5148.$$

c) Una mano que contiene tres cartas de una denominación y dos de otra se puede construir en cuatro pasos sucesivos: seleccionamos la primera denominación; seleccionamos la segunda denominación; seleccionamos tres cartas de la primera denominación; seleccionamos dos cartas de la segunda denominación. La primera denominación se puede elegir de 13 maneras. Una vez elegida, se puede seleccionar la segunda denominación de 12 maneras. Se pueden elegir tres cartas de la primera denominación de $C(4, 3)$ maneras, y se pueden seleccionar dos cartas de la segunda denominación de $C(4, 2)$ maneras. Por el principio de la multiplicación,

la respuesta es

$$13 \cdot 12 \cdot C(4, 3) \cdot C(4, 2) = 3744.$$ ◀

Ejemplo 6.2.22 ▶

¿Cuántas rutas existen desde la esquina inferior izquierda de una cuadrícula $n \times n$ a la esquina superior derecha si los viajes están restringidos sólo a la derecha o hacia arriba? Una ruta se muestra en una cuadrícula de 4×4 en la figura 6.2.5 *a*).

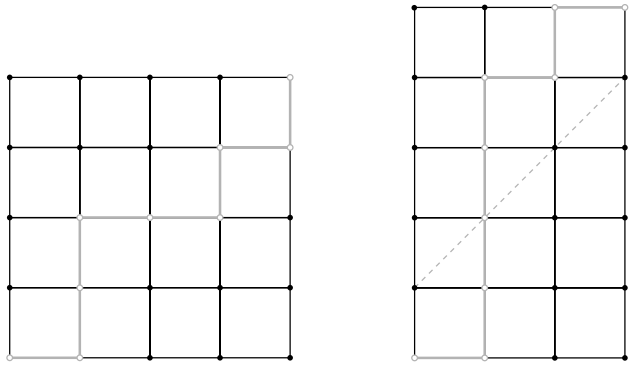


Figura 6.2.5 *a)* Rejilla de 4×4 con una ruta de la esquina inferior izquierda a la superior derecha. *b)* Ruta en *a)* transformada a una ruta en una rejilla de 5×3.

Cada ruta se puede describir por una cadena de n letras D (derecha) y A (arriba). Por ejemplo, la ruta mostrada en la figura 6.2.5 *a*), se describe como la cadena $DAADDADA$. Cualquier cadena se puede obtener seleccionando n posiciones de las D, sin importar el orden de selección, entre las $2n$ posiciones disponibles en la cadena y después llenando el resto de las posiciones con A. Así, existen $C(2n, n)$ rutas posibles. ◀

Ejemplo 6.2.23 ▶

¿Cuántas rutas hay de la esquina inferior izquierda de una rejilla cuadrada de $n \times n$ a la esquina superior derecha, si el viaje se restringe sólo hacia la derecha y arriba, y si se permite tocar pero no pasar hacia arriba de la diagonal que va de la esquina inferior izquierda a la superior derecha?

Un ruta que toca pero que no cruza la diagonal se llama una *ruta buena*, y la que cruza la diagonal se llama una *ruta mala*. El problema es contar el número de rutas buenas. Sea B_n el número de rutas buenas y M_n el número de rutas malas. En el ejemplo 6.2.22 se demostró que

$$B_n + M_n = C(2n, n);$$

entonces, basta con calcular el número de rutas malas.

Una ruta de la esquina inferior izquierda de una rejilla de $(n + 1) \times (n - 1)$ a la esquina superior derecha (sin restricciones) recibe el nombre de ruta de $(n + 1) \times (n - 1)$. La figura 6.2.5 *b*), muestra una ruta 5×3. Se demostrará que el número de rutas malas es igual al número de rutas de $(n + 1) \times (n - 1)$ describiendo una función uno a uno y sobre del conjunto de rutas malas al conjunto de rutas de $(n + 1) \times (n - 1)$.

Dada una ruta mala, se encuentra el primer movimiento (comenzando en la esquina inferior izquierda) que la lleva arriba de la diagonal. De ahí en adelante se sustituye cada movimiento a la derecha por uno hacia arriba y cada movimiento hacia arriba por uno a la derecha. Por ejemplo, la ruta de la figura 6.2.5 *a*), se transforma en la ruta que se muestra en la figura 6.2.5. *b*). Esta transformación también se puede lograr rotando la porción de la ruta que sigue al primer movimiento que cruza la diagonal, respecto a la línea punteada de la figura 6.2.5 *b*). Se ve que esta transformación sin duda asigna a cada ruta mala una ruta de $(n + 1) \times (n - 1)$.

Para demostrar que nuestra función es sobre, considere cualquier ruta de $(n + 1) \times (n - 1)$. Como esta ruta termina arriba de la diagonal, existe un primer movimiento en el que la cruza. Entonces se puede rotar el resto de la ruta respecto a la línea punteada de la

figura 6.2.5 *b*), para obtener una ruta mala. La imagen de esta ruta mala bajo nuestra función es la ruta de $(n+1) \times (n-1)$ con la que se comenzó. Por lo tanto, nuestra función es sobre. Nuestra función también es uno a uno, ya que podemos verificar que la función transforma rutas malas diferentes en rutas de $(n+1) \times (n-1)$ diferentes. Así, el número de rutas malas es igual al número de rutas de $(n+1) \times (n-1)$.

Un argumento como el del ejemplo 6.2.22 demuestra que el número de rutas de $(n+1) \times (n-1)$ es igual a $C(2n, n-1)$. Entonces, el número de rutas buenas es igual a

$$\begin{aligned} C(2n, n) - B_n &= C(2n, n) - C(2n, n-1) = \frac{(2n)!}{n!\,n!} - \frac{(2n)!}{(n-1)!\,(n+1)!} \\ &= \frac{(2n)!}{n!\,(n-1)!}\left(\frac{1}{n} - \frac{1}{n+1}\right) = \frac{(2n)!}{n!\,(n-1)!} \cdot \frac{1}{n(n+1)} \\ &= \frac{(2n)!}{(n+1)n!\,n!} = \frac{C(2n, n)}{n+1}. \end{aligned}$$

◀

WWW

Los números $C(2n, n)/(n+1)$ se llaman **números de Catalan** en honor al matemático belga Eugène-Charles Catalan (1814-1894), quien descubrió una derivación elemental de la fórmula de $C(2n, n)/(n+1)$. Catalan publicó numerosos artículos de análisis, combinatoria, álgebra, geometría, probabilidad y teoría de números. En 1844, llegó a la conjetura de que los únicos enteros positivos consecutivos que son potencias (esto es, i^j, donde $j \geq 2$) son 8 y 9. Más de 150 años después (en 2002) Preda Mihailescu demostró el resultado.

En este libro, los números de Catalan $C(2n, n)/(n+1)$ se denotan por C_n, $n \geq 1$, y C_0 se define como 1. Los primeros números de Catalan son

$$C_0 = 1, \quad C_1 = 1, \quad C_2 = 2, \quad C_3 = 5, \quad C_4 = 14, \quad C_5 = 42.$$

Igual que los números de Fibonacci, los números de Catalan tienen forma de aparecer en lugares inesperados (vea los ejercicios 71, 76 y 77 de esta sección, y 28 y 29 de la sección 7.1).

Esta sección concluye con otra demostración del teorema 6.2.17 que da una fórmula para el número de subconjuntos de r elementos de un conjunto de n elementos. La demostración se ilustra en la figura 6.2.6. Sea X un conjunto de n elementos. Se supone que funciona la fórmula $P(n, r) = n(n-1) \cdots (n-r+1)$ que cuenta el número de ordenamientos de subconjuntos de r elementos sacados de X. Para contar el número de subconjuntos de r elementos de X, *no* se desea tomar en cuenta el orden, se quiere considerar las permutaciones del mismo subconjunto *equivalente*. De manera formal, se define una relación R sobre un conjunto S de permutaciones r de X mediante la siguiente regla: $p_1 R p_2$ si p_1 y p_2 son permutaciones del mismo subconjunto de r elementos de X. De manera directa se verifica que R es una relación equivalente en S.

Si p es una permutación r de X, entonces p es una permutación de algún subconjunto X_r de r elementos de X; entonces, la clase de equivalencia que contiene a p consiste en todas las permutaciones de X_r. Se ve que cada clase de equivalencia tiene $r!$ elementos. Una clase de equivalencia se determina por el subconjunto de r elementos de X que se permuta para obtener a sus miembros. Por lo tanto, existen $C(n, r)$ clases de equivalencia. Como el subconjunto S tiene $P(n, r)$ elementos, por el teorema 3.2.15, $C(n, r) = P(n, r)/r!$.

Figura 6.2.6 Demostración alternativa del teorema 6.2.17 para $n = 4$ y $r = 2$. Cada cuadro contiene una clase de equivalencia para la relación R sobre el conjunto de permutaciones de 2 de $X = \{a, b, c, d\}$ definida por $p_1 R p_2$ si p_1 y p_2 son permutaciones del mismo subconjunto de 2 elementos de X. Existen $P(4, 2) = 12$ permutaciones de 2 elementos y 2 maneras de permutar cada permutación de 2. Como cada clase equivalente corresponde a un subconjunto de X, $12/2 = C(4, 2)$.

Sugerencias para resolver problemas

Los puntos importantes que deben recordarse en esta sección son que una permutación toma en cuenta el orden y una combinación *no* toma en cuenta el orden. Así, una clave para resolver problemas de conteo es determinar si se deben contar objetos ordenados o no ordenados. Por ejemplo, una fila de personas *distintas* se considera ordenada. Entonces, seis personas pueden hacer cola de 6! maneras; se usa la fórmula de la permutación. Un comité es un ejemplo típico de un grupo no ordenado. Por ejemplo, un comité de tres personas se puede seleccionar de un conjunto de 6 personas de $C(6, 3)$ maneras; aquí se usa la fórmula de la combinación.

Sección de ejercicios de repaso

1. ¿Qué es una permutación de $x_1, \ldots, x_n$?
2. ¿Cuántas permutaciones hay de un conjunto de n elementos? ¿Cómo se obtiene esta fórmula?
3. ¿Qué es una permutación r de $x_1, \ldots, x_n$?
4. ¿Cuántas permutaciones r hay de un conjunto de n elementos? ¿Cómo se obtiene esta fórmula?
5. ¿Cómo se denota el número de permutaciones r de un conjunto de n elementos?
6. ¿Qué es una combinación r de $\{x_1, \ldots, x_n\}$?
7. ¿Cuántas combinaciones r hay de un conjunto de n elementos? ¿Cómo se deriva esta fórmula?
8. ¿Cómo se denota el número de combinaciones r de un conjunto de n elementos?

Ejercicios

1. ¿Cuántas permutaciones hay de a, b, c, d?
2. Liste las permutaciones de a, b, c, d.
3. ¿Cuántas permutaciones de 3 hay de a, b, c, d?
4. Liste las permutaciones de 3 de a, b, c, d.
5. ¿Cuántas permutaciones hay de 11 objetos diferentes?
6. ¿Cuántas permutaciones de 5 hay de 11 objetos diferentes?
7. ¿De cuántas maneras se puede seleccionar el presidente, vicepresidente y secretario de un grupo de 11 personas?
8. ¿De cuántas maneras se puede seleccionar el presidente, vicepresidente, secretario y tesorero de un grupo de 12 personas?
9. ¿De cuántas maneras pueden terminar 12 caballos en el orden ganador, segundo lugar, tercer lugar?

En los ejercicios 10 al 18, determine cuántas cadenas se pueden formar ordenando las letras ABCDEF sujetas a las condiciones indicadas.

10. Contiene la subcadena *ACE*
11. Contiene las letras *ACE* juntas en cualquier orden
12. Contiene las subcadenas *DB* y *AE*
13. Contiene ya sea la subcadena *AE* o la subcadena *EA*
14. *A* aparece antes que *D*. *Ejemplos: BCAED*, *BCADE*
15. No contiene las subcadenas *AB*, *CD*
16. No contiene las subcadenas *AB*, *BE*
17. *A* aparece antes que *C* y *C* aparece antes que *E*
18. Contiene ya sea la subcadena *DB* o la subcadena *BE*
19. ¿De cuántas maneras pueden esperar en una fila 5 marcianos y 8 venusinos, si dos marcianos no pueden estar juntos?
20. ¿De cuántas maneras pueden esperar en una fila 5 marcianos, 10 mercurianos y 8 venusinos, si dos marcianos no pueden estar juntos?
21. ¿De cuántas maneras pueden esperar en una fila 5 marcianos y 5 venusinos?
22. ¿De cuántas maneras puede sentarse 5 marcianos y 5 venusinos en una mesa circular?
23. ¿De cuántas maneras pueden sentarse 5 marcianos y 5 venusinos en una mesa circular si dos marcianos no se pueden sentar juntos?
24. ¿De cuántas maneras pueden sentarse 5 marcianos y 8 venusinos en una mesa circular, si dos marcianos no pueden sentarse juntos?

En los ejercicios 25 al 27, sea $X = \{a, b, c, d\}$.

25. Calcule el número de combinaciones de 3 de X.
26. Liste las combinaciones de 3 de X.
27. Demuestre la relación entre las permutaciones de 3 y las combinaciones de 3 elementos de X con un dibujo como el de la figura 6.2.4.
28. ¿De cuántas maneras se puede seleccionar un comité de tres entre un grupo de 11 personas?
29. ¿De cuántas maneras se puede seleccionar un comité de cuatro entre un grupo de 12 personas?
30. En cierto momento del juego de lotería del estado de Illinois, se pidió a una persona que escogiera 6 números (en cualquier orden) entre 44 números. ¿De cuántas maneras puede hacerlo? El estado estaba considerando cambiar el juego de manera que se pidiera a una persona elegir 6 números entre 48. ¿De cuántas maneras podría hacerlo?

Los ejercicios 31 al 36 se refieren a un club cuyos miembros son 6 hombres y 7 mujeres.

31. ¿De cuántas maneras se puede elegir un comité de 5 personas?
32. ¿De cuántas maneras se puede elegir un comité de 3 hombres y 4 mujeres?

33. ¿De cuántas maneras se puede elegir un comité de 4 personas que tenga al menos una mujer?

34. ¿De cuántas maneras se puede seleccionar un comité de 4 personas que incluya al menos un hombre?

35. ¿De cuántas maneras se puede seleccionar un comité de 4 personas que incluya personas de uno y otro sexo?

36. ¿De cuántas maneras se puede elegir un comité de 4 personas de manera que Marta y Rodolfo no estén juntos?

37. ¿De cuántas maneras se puede elegir un comité de 4 republicanos, 3 demócratas y 2 independientes entre un grupo de 10 republicanos, 12 demócratas y 4 independientes?

38. ¿Cuántas cadenas de 8 bits contienen exactamente tres ceros?

39. ¿Cuántas cadenas de 8 bits contienen 3 ceros seguidos y 5 unos?

★***40.*** ¿Cuántas cadenas de 8 bits contienen al menos 2 ceros seguidos?

En los ejercicios 41 al 49, encuentre el número de manos de póquer de 5 cartas (sin ordenar), seleccionadas de una baraja común de 52 cartas, que tengan las propiedades indicadas.

41. Contienen 4 ases

42. Contienen 4 de un tipo, es decir, cuatro cartas de la misma denominación

43. Contienen sólo espadas

44. Contienen cartas de exactamente dos palos

45. Contienen cartas de todos los palos

46. De la forma A2345 del mismo palo

47. Son consecutivas y del mismo palo (suponga que el as tiene la denominación más baja).

48. Son consecutivas (suponga que el as tiene la denominación más baja).

49. Contiene 2 de una denominación, 2 de otra y 1 de una tercera denominación

50. Encuentre el número de manos de bridge de 13 cartas (sin ordenar) seleccionadas de una baraja común de 52 cartas.

51. ¿Cuántas manos de bridge son todas del mismo palo?

52. ¿Cuántas manos de bridge contienen exactamente 2 palos?

53. ¿Cuántas manos de bridge contienen los 4 ases?

54. ¿Cuántas manos de bridge contienen 5 espadas, 4 corazones, 3 tréboles y 1 diamante?

55. ¿Cuántas manos de bridge contienen 5 cartas de un palo, 4 de otro, 3 de otro y 1 de otro palo?

56. ¿Cuántas manos de bridge contienen 4 cartas de 3 palos y una carta del cuarto palo?

57. ¿Cuántas manos de bridge no tienen cartas con cara? (Una carta con cara es una de 10, J, Q, K, A.)

En los ejercicios 58 al 62, una moneda se lanza 10 veces.

58. ¿Cuántos resultados posibles hay? (Un *resultado* es una lista de 10 letras H y T que da el resultado de cada tirada. Por ejemplo, el resultado

H H T H T H H H T H

representa 10 tiradas, donde se obtiene cara las dos primeras veces, cruz la tercera, cara la cuarta, etcétera).

59. ¿Cuántos resultados tienen exactamente tres caras?

60. ¿Cuántos resultados tienen a lo sumo tres caras?

61. ¿Cuántos resultados tienen una cara en la quinta tirada?

62. ¿Cuántos resultados tienen el mismo número de caras y cruces?

Los ejercicios 63 al 66 se refieren a un cargamento de 50 microprocesadores, de los cuales 4 son defectuosos.

63. ¿De cuántas maneras se puede seleccionar un conjunto de cuatro microprocesadores?

64. ¿De cuántas maneras se puede seleccionar un conjunto de cuatro microprocesadores no defectuosos?

65. ¿De cuántas maneras se puede seleccionar un conjunto de cuatro microprocesadores que contenga exactamente dos defectuosos?

66. ¿De cuántas maneras se puede elegir un conjunto de cuatro microprocesadores que contenga al menos uno defectuoso?

★**67.** Demuestre que el número de cadenas de bits de longitud $n \geq 4$ que contienen exactamente dos ocurrencias de 10 es $C(n + 1, 5)$.

★**68** Demuestre que el número de cadenas de n bits que tienen exactamente k ceros sin que haya dos consecutivos es $C(n - k + 1, k)$.

★**69.** Demuestre que el producto de cualquier entero positivo y sus $k - 1$ sucesores es divisible entre $k!$.

70. Demuestre que existen $(2n - 1)(2n - 3) \cdot \cdots 3 \cdot 1$ maneras de elegir n pares entre $2n$ objetos distintos.

Los ejercicios 71 al 73 se refieren a una elección en la que dos candidatos, Ramírez y Uribe, pretenden el puesto de contralor. Después de tabular cada voto, Ramírez nunca estuvo atrás de Uribe. Este problema se conoce como el problema de la urna.

71. Suponga que cada candidato recibe exactamente r votos. Demuestre que el número de maneras en que pueden contarse los votos es C_r, el r-ésimo número de Catalan.

72. Suponga que Ramírez recibió justo r votos y Uribe recibió justo u votos, $r \geq u > 0$. Demuestre que el número de maneras en que pueden contarse los votos es $C(r + u, r) - C(r + u, r + 1)$.

73. Demuestre que si se recibieron exactamente n votos, el número de maneras de contar los votos es $C(n, \lceil n/2 \rceil)$.

74. Suponga que comenzamos en el origen del plano xy y damos n pasos unitarios (es decir, cada paso es de longitud uno), donde cada paso es vertical (arriba o abajo) u horizontal (derecha o izquierda). ¿Cuántas trayectorias de este tipo nunca pasan estrictamente abajo del eje x?

75. Suponga que comenzamos en el origen del plano xy y damos n pasos unitarios (es decir, cada paso es de longitud uno), donde cada paso es vertical (arriba o abajo) u horizontal (derecha o izquierda). ¿Cuántas trayectorias de este tipo se quedan en el primer cuadrante ($x \geq 0$, $y \geq 0$)?

76. Demuestre que el número de maneras en que $2n$ personas, sentadas alrededor de una mesa circular, pueden saludarse por pares sin que se crucen los brazos es C_n, el n-ésimo número de Catalan.

★**77.** Demuestre que la instrucción de imprimir en el seudocódigo

```
for i_1 = 1 to n
    for i_2 = 1 to mín(i_1, n − 1)
        for i_3 = 1 to mín(i_2, n − 2)
            . . .
            for i_{n−1} = 1 to mín(i_{n−2}, 2)
                for i_n = 1 to 1
                    println(i_1, i_2, . . . , i_n)
```

se ejecuta C_n veces, donde C_n es el n-ésimo número de Catalan.

78. Suponga que se tienen n objetos, r diferentes y $n - r$ idénticos. Dé otra derivación de la fórmula

$$P(n, r) = r!\, C(n, r)$$

contando el número de ordenamientos de los n objetos de dos maneras:

- Contando los ordenamientos al elegir primero las posiciones para los r objetos distintos.
- Contando los ordenamientos al elegir primero las posiciones para los $n - r$ objetos idénticos.

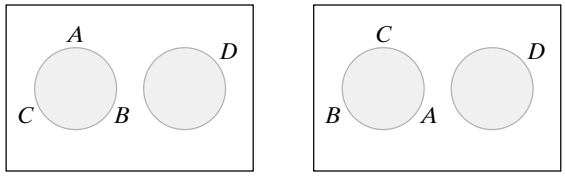
79. ¿Qué está mal en el siguiente argumento, que pretende demostrar que $4C(39, 13)$ manos de bridge contienen tres palos o menos?

Existen $C(39, 13)$ manos que contienen sólo tréboles, diamantes y espadas. De hecho, para cualesquiera tres palos, existen $C(39, 13)$ manos que contienen sólo esos tres palos. Como hay cuatro combinaciones de 3 de los palos, la respuesta es $4C(39, 13)$.

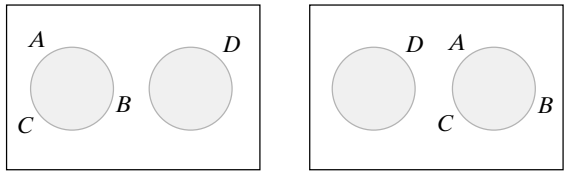
80. ¿Qué está mal en el siguiente argumento, que pretende demostrar que hay $13^4 \cdot 48$ manos de póquer (sin ordenar) de 5 cartas que contienen cartas de todos los palos?

Seleccione una carta de cada palo. Esto se puede hacer de $13 \cdot 13 \cdot 13 \cdot 13 = 13^4$ maneras. Como la quinta carta se puede elegir de 48 maneras, la respuesta es $13^4 \cdot 48$.

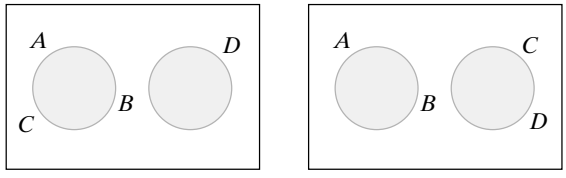
81. Sea $s_{n,k}$ el número de maneras en que n personas se pueden sentar alrededor de k mesas redondas, con al menos una persona en cada mesa. (Los números $s_{n,k}$ se llaman *números de Stirling del primer tipo*). El orden de las mesas *no* se toma en cuenta. El arreglo de lugares en una mesa *sí* se toma en cuenta excepto por las rotaciones. *Ejemplos:* El siguiente par *no* es diferente:



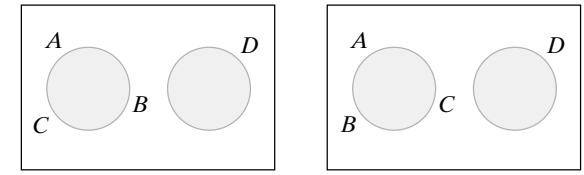
El siguiente par *no* es diferente:



El siguiente par *sí* es diferente:



El siguiente par *es* diferente:



a) Demuestre que $s_{n,k} = 0$ si $k > n$.

b) Demuestre que $s_{n,n} = 1$ para toda $n \geq 1$.

c) Demuestre que $s_{n,1} = (n - 1)!$ para toda $n \geq 1$.

d) Demuestre que $s_{n,n-1} = C(n, 2)$ para toda $n \geq 2$.

e) Demuestre que

$$s_{n,2} = (n - 1)! \left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n - 1}\right)$$

para toda $n \geq 2$.

f) Demuestre que

$$\sum_{k=1}^{n} s_{n,k} = n! \qquad \text{para toda n} \geq 1.$$

g) Encuentre una fórmula para $s_{n,n-2}$, $n \geq 3$ y pruébela.

82. Sea $S_{n,k}$ el número de maneras para hacer una partición de un conjunto de n elementos en exactamente k subconjuntos no vacíos. El orden de los subconjuntos no se toma en cuenta. (Los números $S_{n,k}$ se conocen como *números de Stirling del segundo tipo.*)

a) Demuestre que $S_{n,k} = 0$ si $k > n$.

b) Demuestre que $S_{n,n} = 1$ para toda $n \geq 1$.

c) Demuestre que $S_{n,1} = 1$ para toda $n \geq 1$.

d) Demuestre que $S_{3,2} = 3$.

e) Demuestre que $S_{4,2} = 7$.

f) Demuestre que $S_{4,3} = 6$.

g) Demuestre que $S_{n,2} = 2^{n-1} - 1$ para toda $n \geq 2$.

h) Demuestre que $S_{n,n-1} = C(n, 2)$ para toda $n \geq 2$.

i) Encuentre una fórmula para $S_{n,n-2}$, $n \geq 3$, y pruébela.

83. Demuestre que existen

$$\sum_{k=1}^{n} S_{n,k}$$

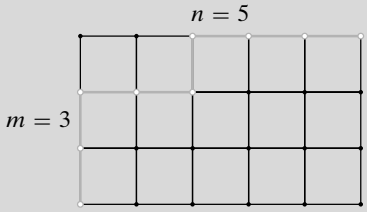
relaciones de equivalencia en un conjunto de n elementos. [Los números $S_{n,k}$ son los números de Stirling del segundo tipo (vea el ejercicio 82)].

Rincón de solución de problemas

Combinaciones

Problema

a) ¿Cuántas rutas hay de la esquina inferior izquierda a la esquina superior derecha de una rejilla de $m \times n$ en la que el movimiento se restringe a sólo la derecha o arriba? Por ejemplo, la siguiente figura es una rejilla de 3×5 y se muestra una ruta.



b) Divida las rutas en clases basándose en la primera vez que la ruta llega a la orilla superior para derivar la fórmula

$$\sum_{k=0}^{n} C(k+m-1, k) = C(m+n, m).$$

Cómo atacar el problema

En el ejemplo 6.2.22 se contaron las trayectorias de la esquina inferior izquierda a la esquina superior derecha de una rejilla de $n \times n$ en donde se restringió el movimiento sólo hacia la derecha o arriba. La solución a ese problema codificó cada ruta como una cadena de letras D (derecha) y A (arriba). Después, el problema se convirtió en uno de contar el número de estas cadenas. Cualquiera de estas cadenas se puede obtener seleccionando n posiciones para D, sin importar el orden de selección, entre las $2n$ posiciones disponibles en la cadena y después completando las posiciones restantes con letras A. Así, el número de cadenas y el número de rutas son iguales a $C(2n, n)$.

En el problema actual, se puede codificar cada ruta como una cadena de n letras D (derecha) y m letras A (arriba). Igual que en el problema anterior, debemos contar el número de estas cadenas. Cualquiera de ellas se puede obtener seleccionando n posiciones para letras D sin importar el orden de selección, entre las $n + m$ posiciones disponibles en la cadena y luego completando las posiciones restantes con letras A. Entonces, el número de cadenas y el número de rutas son iguales a $C(n + m, n)$. Ésta es la solución del inciso *a)*.

En el inciso *b)* se da una sugerencia importante: divida las rutas en clases con base en la primera vez que llegan a la orilla superior. Una ruta puede tocar por primera vez la orilla superior en cualquiera de $n + 1$ posiciones. En la figura anterior, la ruta mostrada llega primero a la orilla superior en la tercera posición desde la izquierda. Antes de seguir leyendo, piense por qué se dividen las rutas en clases.

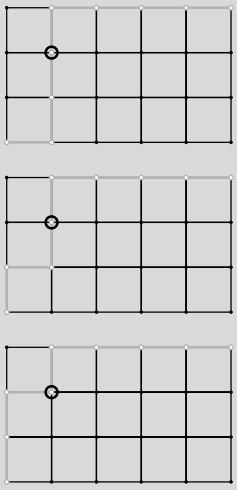
Observe que cuando se dividen las rutas en clases de acuerdo con la primera vez que llegan a la orilla superior:

- Las clases son *ajenas*. (Una ruta no puede llegar por primera vez a la orilla superior en dos posiciones o más). Observe también que todas las rutas se encuentran con la orilla superior en algún punto.
- Cada ruta pertenece a alguna clase.

En la terminología de la sección 2.1 (vea el ejemplo 2.1.14 y el análisis que le precede), las clases forman una *partición* de conjunto de rutas. Por esta razón, se aplica el principio de la suma, y la suma del número de rutas en cada clase es igual al número total de rutas. (Ninguna ruta se cuenta dos veces puesto que las clases no se traslapan, y cada ruta se cuenta una vez ya que cada una pertenece a alguna clase). Es evidente que la ecuación que se supone que debemos probar se obtiene al igualar la suma del número de rutas en cada clase con el número total de rutas.

Cómo encontrar la solución

Ya se resolvió el inciso *a)*. Para el inciso *b)*, observe la rejilla de 3×5. Hay exactamente una ruta que llega primero a la orilla superior en la primera posición de la izquierda. Existen 3 rutas que llegan por primera vez a la orilla superior en la segunda posición de la izquierda:



Observe que la única variación en las figuras anteriores ocurre entre el inicio y el punto marcado con un círculo. Dicho de otra manera, cuando una ruta llega al círculo, sólo hay una manera de terminar el viaje. Por lo tanto, basta contar el número de rutas de la esquina inferior izquierda a la esquina superior derecha en una rejilla de 2×1. Pero este problema se resolvió en el inciso *a)*. El número de rutas de la esquina inferior izquierda a la superior derecha en una rejilla de 2×1 es igual a $C(2 + 1, 1) = 3$. De manera similar, se encuentra que el número de rutas que llega primero a la orilla superior en la tercera posición de la izquierda es igual al número de rutas de la esquina inferior izquierda a la superior derecha

de una rejilla de 2×2, a saber, $C(2 + 2, 2) = 6$. Sumando se obtienen todas las rutas:

$$\begin{aligned} C(5 + 3, 5) = {} & C(0 + 2, 0) + C(1 + 2, 1) \\ & + C(2 + 2, 2) + C(3 + 2, 3) \\ & + C(4 + 2, 4) + C(5 + 2, 5). \end{aligned}$$

Si se sustituye cada término $C(k + 3 - 1, k)$ por su valor, se obtiene

$$56 = 1 + 3 + 6 + 10 + 15 + 21.$$

Usted debe verificar la fórmula anterior, encontrar las 6 rutas que llegan primero a la orilla superior en la tercera posición de la izquierda, y ver por qué el número de rutas de este tipo es igual al número de rutas de la esquina inferior izquierda a la superior derecha en una rejilla de 2×2.

Solución formal

a) Se puede codificar cada ruta como una cadena de n letras D (derecha) y m letras A (arriba). Cualquiera de estas cadenas se puede obtener seleccionando n posiciones para letras D, sin importar el orden de selección, entre las $n + m$ posiciones disponibles en la cadena y después completando las posiciones restantes con letras A. Entonces, el número de rutas es igual a $C(n + m, n)$.

b) Cada ruta se puede describir como una cadena que contiene n letras D y m letras A. La última A en tal cadena marca el punto en que la ruta llega por primera vez a la orilla superior. Se cuentan las cadenas dividiéndolas en clases que consisten en cadenas que terminan en A, AD, ADD, etcétera. Existen

$$C(n + m - 1, n)$$

cadenas que terminan en A, ya que debemos elegir n ranuras para entre las primeras $n + m - 1$ ranuras para las n letras A. Existen

$$C((n - 1) + m - 1, n - 1)$$

cadenas que terminan en AD, ya que debemos elegir $n - 1$ ranuras entre las primeras $(n - 1) + m - 1$ ranuras para las $n - 1$ letras D. En general, existen $C(k + m - 1, k)$ cadenas que terminan en AD^{n-k}. Como hay $C(m + n, m)$ cadenas en total, la fórmula se cumple.

Resumen de las técnicas de solución de problemas

- Busque problemas similares e imite la solución.
- Contar el número de miembros de un conjunto de dos maneras diferentes lleva a una ecuación. En particular, si $\{X_1, X_2, \ldots, X_n\}$ es una *partición de X*, se aplica el principio de la suma y

$$|X| = \sum_{i=1}^{n} |X_i|.$$

- Numere directamente algunos objetos que deben contarse.
- Busque patrones.

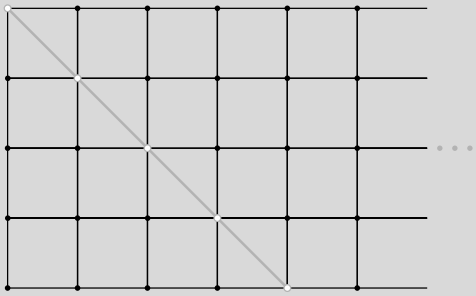
Comentarios

Es importante verificar que una supuesta partición sea en realidad una partición antes de usar el principio de la suma. Si X es el conjunto de cadenas de 5 bits y X_i es el conjunto de cadenas de 5 bits que contienen i ceros consecutivos, el principio de la suma *no* se aplica; los conjuntos X_i *no* son ajenos por pares. Por ejemplo, $00001 \in X_2 \cap X_3$. Como ejemplo de una partición de X, X_i podría ser el conjunto de cadenas de 5 bits que contienen exactamente i ceros.

Ejercicios

1. Divida las rutas en clases basadas en la primera vez que se encuentran una línea vertical y use el principio de la suma para derivar una fórmula como la que se probó en esta sección.

2. Divida las rutas en clases basadas en el punto en que la ruta cruza la línea inclinada que se muestra.



Use el principio de la suma para derivar una fórmula como la que se probó en esta sección.

6.3 ➜ Algoritmos para generar permutaciones y combinaciones

El grupo de rock "Unhinged Universe" ha grabado n videos cuyas duraciones son

$$t_1, t_2, \ldots, t_n$$

WWW segundos. Debe liberarse una cinta que pueda contener C segundos. Como ésta es la primera cinta grabada por "Unhinged Universe", el grupo quiere incluir todo el material posible. En-

tonces el problema es elegir un subconjunto $\{i_1, \ldots, i_k\}$ de $\{1, 2, \ldots, n\}$ tal que la suma

$$\sum_{j=1}^{k} t_{i_j} \qquad \textbf{(6.3.1)}$$

no exceda C y sea tan grande como sea posible. Un enfoque directo consiste en examinar todos los subconjuntos de $\{1, 2, \ldots, n\}$ y elegir un subconjunto de manera que la suma (6.3.1) no exceda C y sea lo más grande posible. Para llevar a la práctica este enfoque, se necesita un algoritmo que genere todas las combinaciones posibles de un conjunto de n elementos. En esta sección se desarrollan algoritmos para generar combinaciones y permutaciones.

Como existen 2^n subconjuntos de un conjunto de n elementos, el tiempo de corrida de un algoritmo que examina todos los subconjuntos es $\Omega(2^n)$. Como se vio en la sección 4.3, es poco práctico correr estos algoritmos excepto para valores pequeños de n. Desafortunadamente, existen problemas (un ejemplo de ellos es el problema de llenado de cinta de video que se describió) para los cuales no se conoce un método mucho mejor que el enfoque de "listar todos".

Nuestros algoritmos listan las permutaciones y combinaciones en **orden lexicográfico**. El orden lexicográfico generaliza el orden común del diccionario.

Se tienen dos palabras diferentes y, para determinar si una precede a la otra en el diccionario, se comparan las letras de las palabras. Existen dos posibilidades:

1. Las palabras tienen longitudes diferentes y cada letra en la palabra más corta es idéntica a la letra correspondiente en la palabra más larga.
2. Las palabras tienen la misma longitud o diferente y en alguna posición, las letras de las palabras difieren. **(6.3.2)**

Si se cumple el número 1, la palabra más corta precede a la más larga. (Por ejemplo, "can" precede a "canino" en el diccionario). Si se cumple el número 2, se localiza la posición p más a la izquierda en la que las letras difieren. El orden de las palabras se determina por el orden de las letras en la posición p. (Por ejemplo, "gladiador" precede a "gladiolo" en el diccionario. En la posición más a la izquierda en que las letras difieren, encontramos "a" en "gladiador" y "o" en "gladiolo"; "a" precede a "o" en el alfabeto).

El orden lexicográfico generaliza el orden común del diccionario sustituyendo el alfabeto por cualquier conjunto de símbolos donde se ha definido un orden. Se estudiarán las cadenas de enteros.

Definición 6.3.1 ▶

Sean $\alpha = s_1 s_2 \cdots s_p$ y $\beta = t_1 t_2 \cdots t_q$ cadenas en el conjunto $\{1, 2, \ldots, n\}$. Se dice que α es *menor que* β, *en el orden lexicográfico*, y se escribe $\alpha < \beta$ si ocurre

a) $p < q$ y $s_i = t_i$ para $i = 1, \ldots, p$,

o

b) para alguna i, $s_i \neq t_i$, y para la más pequeña de esas i, se tiene $s_i < t_i$. ◀

En la definición 6.3.1, el caso *a)* corresponde a la posibilidad 1 de (6.3.2) y el caso *b)* corresponde a la posibilidad 2 de (6.3.2).

Ejemplo 6.3.2 ▶

Sean $\alpha = 132$ y $\beta = 1324$ dos cadenas en el conjunto $\{1, 2, 3, 4\}$. En la notación de la definición 6.3.1, $p = 3$, $q = 4$, $s_1 = 1$, $s_2 = 3$, $s_3 = 2$, $t_1 = 1$, $t_2 = 3$, $t_3 = 2$, y $t_4 = 4$. Como $p = 3 < 4 = q$ y $s_i = t_i$ para $i = 1, 2, 3$, se cumple la condición *a)* de la definición 6.3.1. Por lo tanto, $\alpha < \beta$. ◀

Ejemplo 6.3.3 ▶

Sean $\alpha = 13246$ y $\beta = 1342$ dos cadenas en $\{1, 2, 3, 4, 5, 6\}$. En la notación de la definición 6.3.1, $p = 5$, $q = 4$, $s_1 = 1$, $s_2 = 3$, $s_3 = 2$, $s_4 = 4$, $s_5 = 6$, $t_1 = 1$, $t_2 = 3$, $t_3 = 4$, y $t_4 = 2$. La i más pequeña para la que $s_i \neq t_i$ es $i = 3$. Como $s_3 < t_3$, por la condición *b)* de la definición 6.3.1, $\alpha < \beta$. ◀

Ejemplo 6.3.4 ▶

Sean $\alpha = 1324$ y $\beta = 1342$ dos cadenas en $\{1, 2, 3, 4\}$. En la notación de la definición 6.3.1, $p = q = 4$, $s_1 = 1$, $s_2 = 3$, $s_3 = 2$, $s_4 = 4$, $t_1 = 1$, $t_2 = 3$, $t_3 = 4$, y $t_4 = 2$. La i más pequeña para la que $s_i \neq t_i$ es $i = 3$. Como $s_3 < t_3$, por la condición *b)* de la definición 6.3.1, $\alpha < \beta$. ◀

Ejemplo 6.3.5 ▶

Sean $\alpha = 13542$ y $\beta = 21354$ dos cadenas en $\{1, 2, 3, 4, 5\}$. En la notación de la definición 6.3.1, $s_1 = 1$, $s_2 = 3$, $s_3 = 5$, $s_4 = 4$, $s_5 = 2$, $t_1 = 2$, $t_2 = 1$, $t_3 = 3$, $t_4 = 5$, y $t_5 = 4$. La i más pequeña para la que $s_i \neq t_i$ es $i = 1$. Como $s_1 < t_1$, por la condición *b)* de la definición 6.3.1, $\alpha < \beta$. ◀

Para cadenas de la misma longitud en $\{1, 2, \ldots, 9\}$, el orden lexicográfico es el mismo que el orden numérico en los enteros positivos si se interpretan las cadenas como números decimales (vea los ejemplos 6.3.4 y 6.3.5). Para cadenas de diferente longitud, el orden lexicográfico es diferente del orden numérico (vea el ejemplo 6.3.3). En el resto de esta sección, la palabra *orden* se referirá al orden lexicográfico.

Primero se considera el problema de listar todas las combinaciones r de $\{1, 2, \ldots, n\}$. En nuestro algoritmo, se lista la combinación r $\{x_1, \ldots, x_r\}$ como la cadena $s_1 \cdots s_r$, donde $s_1 < s_2 < \cdots < s_r$ y $\{x_1, \ldots, x_n\} = \{s_1, \ldots, s_r\}$. Por ejemplo, la combinación de 3 $\{6, 2, 4\}$ quedará en la lista como 246.

Se presentarán las combinaciones r de $\{1, 2, \ldots, n\}$ en orden lexicográfico. Así, la primera cadena en la lista será $12 \cdots r$ y la última cadena listada será $(n - r + 1) \cdots n$.

Ejemplo 6.3.6 ▶

Considere el orden en el que se listan las combinaciones de 5 elementos de $\{1, 2, 3, 4, 5, 6, 7\}$. La primera cadena es 12345, que va seguida de 12346 y 12347. La siguiente cadena es 12356, seguida de 12357. La última cadena será 34567. ◀

Ejemplo 6.3.7 ▶

Encuentre la cadena que sigue a 13457 cuando se listan las combinaciones de 5 de $X = \{1, 2, 3, 4, 5, 6, 7\}$.

Ninguna cadena que comienza con 134 y represente una combinación de 5 de X excede a 13467. Así, la cadena que sigue a 13467 debe comenzar con 135. Como 13567 es la cadena más pequeña que comienza con 135 y representa una combinación de 5 elementos de X, la respuesta es 13567. ◀

Ejemplo 6.3.8 ▶

Encuentre la cadena que sigue a 2367 cuando se listan las combinaciones de 4 de $X = \{1, 2, 3, 4, 5, 6, 7\}$.

Ninguna cadena que comienza con 23 y represente una combinación de 4 de X excede a 2367. Entonces, la cadena que sigue a 2367 debe comenzar con 24. Como 2456 es la cadena más pequeña que comienza con 24 y representa una combinación de 4 elementos de X, la respuesta es 2456. ◀

Se está desarrollando un patrón. Dada una cadena $\alpha = s_1 \cdots s_r$, que representa la combinación r $\{s_1, \ldots, s_r\}$, para encontrar la siguiente cadena $\beta = t_1 \cdots t_r$, primero se encuentra el elemento más a la derecha s_m que no tiene su valor máximo. (s_r puede tener el valor máximo n, s_{n-1} puede tener el valor máximo $n - 1$, etcétera) Entonces,

$$t_i = s_i \quad \text{para } i = 1, \ldots, m - 1.$$

El elemento t_m es igual a $s_m + 1$. Para el resto de la cadena β se tiene

$$t_{m+1} \cdots t_r = (s_m + 2)(s_m + 3) \cdots.$$

El algoritmo es el siguiente.

Algoritmo 6.3.9

Generación de combinaciones

Este algoritmo lista todas las combinaciones r de $\{1, 2, \ldots, n\}$ en orden lexicográfico creciente.

Entrada: r, n
Salida: Todas las combinaciones de $\{1, 2, \ldots, n\}$ en orden lexicográfico creciente.

```
1.  combinación(r, n) {
2.      for i = 1 to r
3.          s_i = i
4.      println(s_1, . . . , s_r)//imprime la primera combinación r
5.      for i = 2 to C(n, r) {
6.          m = r
7.          val_máx = n
8.          while (s_m == val_máx) {
9.              //encuentra el elemento más a la derecha
                //que no tiene su valor máximo
10.             m = m − 1
11.             val_máx = val_máx − 1
12.         }
13.         //se incrementa el elemento más a la derecha
14.         s_m = s_m + 1
15.         //el resto de los elementos son sucesores de s_m
16.         for j = m + 1 to r
17.             s_j = s_{j−1} + 1
18.         println(s_1, . . . ,s_r)//imprime la i-ésima combinación
19.     }
20. }
```

Ejemplo 6.3.10 ▶

Se mostrará la manera en que el algoritmo 6.3.9 genera la combinación de 5 elementos de $\{1, 2, 3, 4, 5, 6, 7\}$ que sigue a 23467. Se supone que

$$s_1 = 2, \quad s_2 = 3, \quad s_3 = 4, \quad s_4 = 6, \quad s_5 = 7.$$

En la línea 13, se encuentra que s_3 es el elemento más a la derecha que no tiene su valor máximo. En la línea 14, s_3 se iguala a 5. En las líneas 16 y 17, s_4 se iguala a 6 y s_5 se hace igual a 7. En este punto

$$s_1 = 2, \quad s_2 = 3, \quad s_3 = 4, \quad s_4 = 6, \quad s_5 = 7.$$

Se ha generado la combinación de 5, 23567, que sigue a 23467. ◀

Ejemplo 6.3.11 ▶

Las combinaciones de 4 de $\{1, 2, 3, 4, 5, 6\}$ según las lista el algoritmo 6.3.9 son

1234, 1235, 1236, 1245, 1246, 1256, 1345, 1346,
1356, 1456, 2345, 2346, 2356, 2456, 3456. ◀

Igual que el algoritmo para generar las combinaciones r, el algoritmo para generar las permutaciones lista las permutaciones de $\{1, 2, \ldots, n\}$ en orden lexicográfico. (El ejercicio 16 pide un algoritmo que genere todas las permutaciones r de un conjunto de n elementos).

Ejemplo 6.3.12 ▶

Para construir la permutación de $\{1, 2, 3, 4, 5, 6\}$ que sigue a 163542, debemos conservar iguales el mayor número posible de dígitos de la izquierda.

¿Puede la permutación que sigue a la permutación dada tener la forma 1635_ _? Como la única permutación de la forma 1635_ _ distinta de la permutación dada es 163524, y 163524 es menor que 163542, la permutación que sigue a la permutación dada no es de la forma 1635_ _.

¿Puede la permutación que sigue a la permutación dada tener la forma 163_ _ _? Los últimos tres dígitos deben ser una permutación de {2, 4, 5}. Como 542 es la permutación más grande de {2, 4, 5}, cualquier permutación que comience con 163 es más pequeña que la permutación dada. Entonces, la permutación que sigue a la que se da no es de la forma 163_ _ _.

La razón para que la permutación que sigue a la permutación dada no pueda comenzar con 1635 o 163 es que, en cualquiera de los dos casos, el resto de los dígitos de la permutación dada (42 y 542, respectivamente) *decrece*. Por lo tanto, al trabajar desde la derecha debemos encontrar el primer dígito d cuyo vecino de la derecha r satisface $d < r$. En nuestro caso, el tercer dígito, 3, tiene esta propiedad. Entonces, la permutación que sigue a la permutación dada comenzará con 16.

El dígito que sigue a 16 debe exceder a 3. Como se desea la siguiente permutación más pequeña, el siguiente dígito es 4, el dígito más pequeño disponible. Entonces, la permutación deseada comienza con 164. Los dígitos restantes 235 deben estar en orden creciente para lograr el valor mínimo. Por lo tanto, la permutación que sigue a la permutación dada es 164235. ◀

Se puede ver que para generar todas la permutaciones de $\{1, 2, \ldots, n\}$, se comienza con la permutación $12 \cdots n$ y después se usa el método del ejemplo 6.3.12 repetidas veces para generar la siguiente permutación. El proceso termina cuando se genera la permutación $n(n - 1) \cdots 21$.

Ejemplo 6.3.13 ▶

Usando el método del ejemplo 6.3.12, se pueden listar las permutaciones de {1, 2, 3, 4} en orden lexicográfico como

1234, 1243, 1324, 1342, 1423, 1432, 2134, 2143,
2314, 2341, 2413, 2431, 3124, 3142, 3214, 3241,
3412, 3421, 4123, 4132, 4213, 4231, 4312, 4321. ◀

El algoritmo es el siguiente.

Algoritmo 6.3.14

Generación de permutaciones

Este algoritmo lista todas las permutaciones de $\{1, 2, \ldots, n\}$ en orden lexicográfico creciente.

Entrada: n
Salida: todas las permutaciones de $\{1, 2, \ldots, n\}$ en orden lexicográfico creciente.

```
1.  permutación(n) {
2.      for i = 1 to n
3.          s_i = i
4.      println(s_1, . . . , s_n) // imprime la primera permutación
5.      for i = 2 to n! {
6.          m = n − 1
7.          while (s_m > s_{m+1})
8.              // encuentra el primer decremento trabajando
                // desde la derecha
9.              m = m − 1
10.         k = n
11.         while (s_m > s_k)
12.             // encuentra el elemento s_k más a la derecha
                // con s_m < s_k
13.             k = k − 1
14.         swap(s_m, s_k)
15.         p = m + 1
16.         q = n
17.         while (p < q) {
18.             // cambia s_{m+1} y s_n, cambia s_{m+2} y s_{n-1}, etc.
19.             swap(s_p, s_q)
20.             p = p + 1
```

```
21.              q = q − 1
22.          }
23.          println(s_1, . . . , s_n)//imprime la i-ésima permutación
24.      }
25.  }
```

Ejemplo 6.3.15 ▶

Se mostrará la manera en que el algoritmo 6.3.14 genera la permutación que sigue a 163542. Suponga que

$$s_1 = 1, \quad s_2 = 6, \quad s_3 = 3, \quad s_4 = 5, \quad s_5 = 4, \quad s_6 = 2$$

y que estamos en la línea 6. El índice mayor m que satisface $s_m < s_{m+1}$ es 3. En las líneas 10 a la 13, se encuentra que el índice más grande k que satisface $s_k > s_m$ es 5. En la línea 14, se intercambian s_m y s_k. En este punto, se tiene $s = 164532$. En las líneas 15 a la 22, se invierte el orden de los elementos $s_4s_5s_6 = 532$. Se obtiene la permutación deseada, 164235. ◀

Sección de ejercicios de repaso

1. Defina *orden lexicográfico*.

2. Describa el algoritmo para generar combinaciones r.

3. Describa el algoritmo para generar permutaciones.

Ejercicios

En los ejercicios 1 al 3, encuentre la combinación r que generará el algoritmo 6.3.9 con n = 7 después de la combinación r dada.

1. 1356 **2.** 12367 **3.** 14567

En los ejercicios 4 al 6, encuentre la permutación que generará el algoritmo 6.3.14 después de la permutación dada.

4. 12354 **5.** 625431 **6.** 12876543

7. Para cada cadena en los ejercicios 1 al 3, explique (como en el ejemplo 6.3.10) exactamente la forma en que el algoritmo 6.3.9 genera la siguiente combinación r.

8. Para cada cadena en los ejercicios 4 al 6, explique (como en el ejemplo 6.3.15) exactamente la forma en que el algoritmo 6.3.14 genera la siguiente permutación.

9. Muestre la salida del algoritmo 6.3.9 cuando $n = 6$ y $r = 3$.

10. Muestre la salida del algoritmo 6.3.9 cuando $n = 6$ y $r = 2$.

11. Muestre la salida del algoritmo 6.3.9 cuando $n = 7$ y $r = 5$.

12. Muestre la salida del algoritmo 6.3.14 cuando $n = 2$.

13. Muestre la salida del algoritmo 6.3.14 cuando $n = 3$.

14. Modifique el algoritmo 6.3.9 de manera que la línea 5

```
5.        for i = 2 to C(n, r) {
```

se elimine. Base la condición de terminación en el hecho de que la última combinación r tiene todos los elementos s_i iguales a su valor máximo.

15. Modifique el algoritmo 6.3.14 de manera que la línea 5

```
5.        for i = 2 to n! {
```

se elimine. Base la condición de terminación en el hecho de que la última permutación tiene los elementos s_i en orden decreciente.

16. Escriba un algoritmo que genere todas las permutaciones r de un conjunto de n elementos.

17. Escriba un algoritmo cuya entrada es una combinación r de $\{1, 2, \ldots, n\}$. La salida es la siguiente combinación r (en orden lexicográfico). La primera combinación r sigue a la última combinación r.

18. Escriba un algoritmo cuya entrada es una permutación de $\{1, 2, \ldots, n\}$. La salida es la siguiente permutación (en orden lexicográfico). La primera permutación sigue a la última permutación.

19. Escriba una algoritmo cuya entrada es una combinación r $\{1, 2, \ldots, n\}$. La salida es la combinación r anterior (en orden lexicográfico). La última combinación r precede a la primera combinación r.

20. Escriba un algoritmo cuya entrada es una permutación de $\{1, 2, \ldots, n\}$. La salida es la permutación anterior (en orden lexicográfico). La última permutación precede a la primera.

★21. Escriba un algoritmo recursivo que genere todas las combinaciones r del conjunto $\{s_1, s_2, \ldots, s_n\}$. Divida el problema en dos subproblemas:

- Elabore una lista de las combinaciones r que contienen a s_1.
- Elabore una lista de las combinaciones r que no contienen a s_1.

22. Escriba un algoritmo recursivo que genere todas las permutaciones del conjunto $\{s_1, s_2, \ldots, s_n\}$. Divida el problema en n subproblemas:

- Elabore una lista de las permutaciones que comienzan con s_1.
- Elabore una lista de las permutaciones que comienzan con s_2.

  .
  .
  .

- Elabore una lista de las permutaciones que comienzan con s_n.

6.4 ➜ Introducción a la probabilidad discreta†

La **probabilidad** se desarrolló en el siglo XVII para analizar los juegos y, en su forma incipiente, implicaba el conteo. Por ejemplo, suponga que un dado de seis lados no cargado, con etiquetas 1, 2, 3, 4, 5, 6 en los lados, se lanza (vea la figura 6.4.1). "No cargado" significa que cada número tiene la misma oportunidad de aparecer cuando se lanza el dado. *WWW* Para calcular la posibilidad o *probabilidad* de que aparezca un número par, primero se *cuenta* de cuántas maneras puede salir un número par (tres: 2, 4, 6) y de cuántas maneras puede salir un número arbitrario (seis: 1, 2, 3, 4, 5, 6); entonces, la probabilidad es el cociente $3/6 = 1/2$. Después de introducir cierta terminología, se darán varios ejemplos de cálculo de probabilidades.

Figura 6.4.1 Lanzamiento de un dado no cargado. "No cargado" significa que es igualmente probable que salga cada número cuando se lanza el dado. [*Foto del autor. Mano cortesía de Ben Schneider*].

Un **experimento** es un proceso que lleva a un resultado. Un **evento** es un resultado o combinación de resultados de un experimento. El **espacio muestra** es el evento que consiste en todos los resultados posibles.

Ejemplo 6.4.1 ▶

Algunos ejemplos de experimentos son:

- Lanzar un dado de seis lados.
- Seleccionar al azar cinco microprocesadores en un lote de 1000.
- Seleccionar un recién nacido en el Hospital San Roque.

Algunos ejemplos de eventos que pueden ocurrir cuando se realizan los experimentos anteriores son:

- Obtener 4 al lanzar el dado de seis lados.
- No encontrar microprocesadores defectuosos en los cinco elegidos al azar del lote de 1000.
- Seleccionar una niña recién nacida en el Hospital San Roque.

Los espacios muestra para los experimentos anteriores son:

- Los números 1, 2, 3, 4, 5, 6, es decir, todos los resultados posibles al lanzar un dado.
- Todas las combinaciones de cinco microprocesadores seleccionados de un lote de 1000.
- Todos los recién nacidos en el Hospital San Roque. ◀

Si todos los resultados en un espacio muestra finito son igualmente probables, la probabilidad de un evento se define como el número de resultados en el evento dividido entre el número de resultados en el espacio muestra. En la siguiente sección, se relajará la suposición de que todos los resultados son igualmente probables.

† Esta sección se puede omitir sin pérdida de continuidad.

Definición 6.4.2 ▶ La *probabilidad* $P(E)$ de un evento E del espacio muestra finito S es

$$P(E) = \frac{|E|}{|S|}.^{\dagger}$$

◀

Ejemplo 6.4.3 ▶ Se lanzan dos dados no cargados. ¿Cuál es la probabilidad de que la suma de los números en los dados sea 10?

Como el primer dado puede mostrar cualquiera de seis números y el segundo dado puede mostrar cualquiera de seis números, por el principio de la multiplicación hay $6 \cdot 6 = 36$ sumas posibles; es decir, el tamaño del espacio muestra es 36. Existen tres maneras posibles de obtener la suma de 10, (4, 6), (5, 5), (6, 4); esto es, el tamaño del evento consistente en "obtener una suma de 10" es 3. [La notación (x, y) significa obtener x en el primer dado y y en el segundo]. Por lo tanto, la probabilidad es $3/36 = 1/12$. ◀

Ejemplo 6.4.4 ▶ Cinco microprocesadores se seleccionan al azar de un lote de 1000 entre los cuales 20 son defectuosos. Encuentre la probabilidad de no obtener microprocesadores defectuosos.

Existen $C(1000, 5)$ maneras de seleccionar 5 microprocesadores entre 1000. Existen $C(980, 5)$ maneras de seleccionar 5 microprocesadores sin defecto ya que hay $1000 - 20 = 980$ productos buenos. Por lo tanto, la probabilidad de obtener microprocesadores no defectuosos es

$$\frac{C(980, 5)}{C(1000, 5)} = \frac{980 \cdot 979 \cdot 978 \cdot 977 \cdot 976}{1000 \cdot 999 \cdot 998 \cdot 997 \cdot 996} = 0.903735781.$$

◀

Ejemplo 6.4.5 ▶ En el juego de lotería del estado de Illinois, para ganar el premio más alto, el jugador debe acertar en seis números distintos, en cualquier orden, entre los números 1 a 52 sacados aleatoriamente por un representante de la lotería. ¿Cuál es la probabilidad de elegir los números ganadores?

Seis números entre 52 se pueden elegir de $C(52, 6)$ maneras. Como hay una combinación ganadora, la probabilidad de elegir los números ganadores es

$$\frac{1}{C(52, 6)} = \frac{6!}{52 \cdot 51 \cdot 50 \cdot 49 \cdot 48 \cdot 47} = 0.000000049.$$

◀

Ejemplo 6.4.6 ▶ Una mano de bridge consiste en 13 cartas de una baraja común de 52 cartas. Encuentre la probabilidad de obtener una distribución 4−4−4−1, es decir, cuatro cartas en cada uno de tres palos diferentes y una carta de un cuarto palo.

Existen $C(52, 13)$ manos de bridge. El palo de una carta se puede elegir de 4 maneras y la carta en sí se puede seleccionar de 13 maneras. Una vez seleccionada esta carta, debemos elegir cuatro cartas de cada uno de los palos restantes, lo que se puede hacer de $C(13, 4)^3$ maneras. Así, existen $4 \cdot 13 \cdot C(13, 4)^3$ manos con la distribución 4−4−4−1. Por lo tanto, la probabilidad de obtener una distribución 4−4−4−1 es

$$\frac{4 \cdot 13 \cdot C(13, 4)^3}{C(52, 13)} = 0.03.$$

◀

Sección de ejercicios de repaso

1. ¿Qué es un experimento?
2. ¿Qué es un evento?
3. ¿Qué es un espacio muestra?
4. Si todos los resultados en un espacio muestra finito son igualmente probables, ¿cómo se define la probabilidad de un evento?

† Recuerde que $|X|$ es el número de elementos en un conjunto finito X.

Ejercicios

En los ejercicios 1 al 4, suponga que se lanzan una moneda y un dado.

1. Liste los miembros del espacio muestra.

2. Liste los miembros del evento "la moneda muestra cara y el dado muestra un número par".

3. Liste los miembros del evento "el dado muestra un número impar".

4. Liste los miembros del evento "la moneda muestra cara y el dado un número menor que 4".

En los ejercicios 5 al 7 se lanzan dos dados.

5. Liste los miembros del evento "la suma de los números en los dados es par".

6. Liste los miembros del evento "salen dobles" (esto es, sale el mismo número en los dos dados).

7. Liste los miembros del evento "sale 4 en al menos un dado".

8. Dé un ejemplo de un experimento diferente a los de esta sección.

9. Dé un ejemplo de un evento cuando se realiza el experimento del ejercicio 8.

10. ¿Cuál es el espacio muestra para el experimento del ejercicio 8?

11. Se lanza un dado no cargado. ¿Cuál es la probabilidad de obtener un 5?

12. Se lanza un dado no cargado. ¿Cuál es la probabilidad de obtener un número par?

13. Se lanza un dado no cargado. ¿Cuál es la probabilidad de no obtener un 5?

14. Se elige aleatoriamente una carta de una baraja común de 52 cartas. ¿Cuál es la probabilidad de que sea el as de espadas?

15. Se elige al azar una carta de una baraja común de 52 cartas. ¿Cuál es la probabilidad de que sea un jack?

16. Se elige al azar una carta de una baraja común de 52 cartas. ¿Cuál es la probabilidad de que sea un corazón?

17. Se lanzan dos dados no cargados. ¿Cuál es la probabilidad de que la suma de los números en los dados sea 9?

18. Se lanzan dos dados no cargados. ¿Cuál es la probabilidad de que la suma de los números en los dados sea impar?

19. Se lanzan dos dados no cargados. ¿Cuál es la probabilidad de obtener dobles?

20. Se eligen cuatro microprocesadores de un lote de 100 entre los que hay 10 defectuosos. Encuentre la probabilidad de no obtener defectuosos.

21. Se eligen cuatro microprocesadores de un lote de 100 entre los que hay 10 defectuosos. Encuentre la probabilidad de obtener exactamente un microprocesador defectuoso.

22. Se eligen cuatro microprocesadores de un lote de 100 entre los que hay 10 defectuosos. Encuentre la probabilidad de obtener cuando mucho un microprocesador defectuoso.

23. En el juego de California Daily 3, un jugador debe seleccionar tres números entre 0 y 9; se permiten repeticiones. Ganar una "jugada directa" requiere que los números coincidan en el orden exacto en que el representante de la lotería los saca al azar. ¿Cuál es la probabilidad de elegir los números ganadores?

24. En el juego de California Daily 3, un jugador debe elegir tres números entre 0 y 9. Ganar una "jugada de caja" requiere que los tres números coincidan en cualquier orden con los que saca aleatoriamente el representante de la lotería; se permiten repeticiones. ¿Cuál es la probabilidad de elegir los números ganadores, suponiendo que el jugador elige tres números distintos?

25. En el juego de lotería de Maryland, para ganar el premio más alto, el jugador debe elegir los mismos seis números distintos, en cualquier orden, que selecciona al azar el representante de la lotería entre los números 1 a 49. ¿Cuál es la probabilidad de elegir los números ganadores?

26. En el Big Game en varios estados, para ganar el premio más alto, el jugador debe acertar a los mismos cinco números distintos, en cualquier orden, elegidos entre los números 1 al 50, y a un número del Big Money Ball elegido entre 1 y 36, todos seleccionados al azar por un representante de la lotería. ¿Cuál es la probabilidad de elegir los números ganadores?

27. En el juego de Maryland Cash In Hand, para ganar el premio mayor, el jugador debe atinar a los mismos siete números distintos, en cualquier orden, seleccionados al azar entre 1 y 31 por el representante de la lotería. ¿Cuál es la probabilidad de elegir los números ganadores?

28. Encuentre la probabilidad de obtener una mano de bridge con una distribución 5−4−2−2, es decir, cinco cartas de un palo, cuatro de otro palo y dos cartas de cada uno de los otros dos palos.

29. Encuentre la probabilidad de obtener una mano de bridge que consiste sólo en cartas rojas, estos es, sin espadas ni tréboles.

Los ejercicios 30 al 33 se refieren a un estudiante mal preparado que presenta un examen de 10 preguntas para responderse como falso o verdadero, y adivina todas las respuestas.

30. ¿Cuál es la probabilidad de que el estudiante responda correctamente todas las preguntas?

31. ¿Cuál es la probabilidad de que el estudiante conteste incorrectamente todas las preguntas?

32. ¿Cuál es la probabilidad de que el estudiante responda bien exactamente una pregunta?

33. ¿Cuál es la probabilidad de que el estudiante responda bien exactamente cinco preguntas?

Los ejercicios 34 al 36 se refieren a una pequeña encuesta en la que se pide a 10 personas que elijan su refresco favorito entre Coca, Pepsi y RCola.

34. Si cada persona elige una refresco al azar, ¿cuál es la probabilidad de que ninguna de ellas elija Coca?

35. Si cada persona elige un refresco al azar, ¿cuál es la probabilidad de que al menos una persona no haya elegido Coca?

36. Si cada persona elige un refresco al azar, ¿cuál es la probabilidad de que todos elijan Coca?

37. Si se seleccionan aleatoriamente los registros de cinco estudiantes, ¿cuál es la probabilidad de que se hayan elegido de manera que el primer registro corresponda al menor promedio general, el segundo tenga el promedio general que sigue, etcétera?

Los ejercicios 38 al 40 se refieren a tres personas, cada una de las cuales elige aleatoriamente un casillero entre 12.

38. ¿Cuál es la probabilidad de que los tres casilleros seleccionados sean consecutivos?

39. ¿Cuál es la probabilidad de que ningún par de casilleros sea consecutivo?

40. ¿Cuál es la probabilidad de que al menos dos de los casilleros sean consecutivos?

Los ejercicios 41 al 44 se relacionan con una ruleta que tiene 38 números: 18 rojos, 18 negros, un 0, un 00 (0 y 00 no son rojos ni negros). Cuando la ruleta da vueltas, todos los números tienen la misma probabilidad de salir.

41. ¿Cuál es la probabilidad de que la ruleta se detenga en un número negro?

42. ¿Cuál es la probabilidad de que la ruleta se detenga en un número negro dos veces seguidas?

43. ¿Cuál es la probabilidad de que la ruleta se detenga en 0?

44. ¿Cuál es la probabilidad de que la ruleta se detenga en 0 o 00?

Los ejercicios 45 al 47 se refieren al problema de Monty Hall, en el que el concursante elige una de tres puertas; detrás de una hay un automóvil y detrás de las otras dos hay cabras. Una vez que el concursante elige una puerta, el conductor abre una de las otras dos que oculta una cabra. (Como hay dos cabras, el conductor siempre puede abrir una puerta que esconde una cabra sin importar cuál puerta haya elegido primero el concursante). El conductor da entonces al concursante la opción de abandonar la puerta elegida en favor de la otra cerrada que no seleccionó. Para cada estrategia, ¿cuál es la probabilidad de ganar el automóvil?

45. Quedarse con la puerta elegida al principio.

46. Tomar una decisión aleatoria acerca de quedarse con la puerta elegida o cambiar a la cerrada que no se eligió.

47. Cambiar a la puerta cerrada no elegida.

Los ejercicios 48 al 50 se relacionan con una variante del problema de Monty Hall, en el que el concursante elige una de cuatro puertas; atrás de una hay un automóvil y detrás de las otras tres hay cabras. Una vez que el concursante elige una puerta, el conductor abre una de la otras tres que oculta una cabra. Después da al concursante la opción de abandonar la puerta elegida en favor de una de las otras dos todavía cerradas. Para cada estrategia, ¿cuál es la probabilidad de ganar el automóvil?

48. Quedarse con la puerta elegida al principio.

49. Tomar una decisión aleatoria respecto a quedarse con la puerta elegida o cambiar a una de las otras dos puertas cerradas.

50. Cambiar a una de la puertas no elegidas, cerradas. La elección entre ellas se hace al azar.

51. En un examen de opción múltiple, una pregunta tiene tres opciones: A, B y C. Un estudiante elige A. El profesor establece que la opción C es incorrecta. ¿Cuál es la probabilidad de una respuesta correcta si se queda con la opción A? ¿Cuál es la probabilidad de una respuesta correcta si el estudiante cambia a B?

52. ¿Es correcto el siguiente razonamiento? Un inspector de salud del condado dijo al dueño de un restaurante que ofrece quiches de cuatro huevos que, como una investigación de la *Food and Drug Administration* (FDA) muestra que uno de cada cuatro huevos están contaminados con bacteria de salmonela, el restaurante debería usar sólo tres huevos en cada quiche.

53. Un juego de dos personas se juega lanzando una moneda no cargada hasta que aparece la sucesión HT (cara, cruz) o la sucesión TT (cruz, cruz). Si sale HT, el primer jugador gana; si sale TT, el segundo jugador gana. ¿Preferiría ser el primero o el segundo jugador? Explique por qué.

6.5 ➜ Teoría de probabilidad discreta†

En la sección 6.4, se supuso que todos los resultados eran igualmente probables; es decir, si hay n resultados posibles, la probabilidad de cada uno es $1/n$. En general, los resultados no son igualmente probables. Por ejemplo, un dado se "carga" de manera que sea más probable que aparezcan unos números que otros. Para manejar el caso de resultados que no son igualmente probables, se asigna una probabilidad $P(x)$ a cada resultado x. Los valores de $P(x)$ *no* necesitan ser iguales. P se llama *función de probabilidad*. En esta sección, se supondrá que todos los espacios muestra son finitos.

Definición 6.5.1 ▶

Una *función de probabilidad* P asigna a cada resultado x en un espacio muestra S un número $P(x)$ de tal manera que

$$0 \leq P(x) \leq 1, \quad \text{para toda } x \in S,$$

y

$$\sum_{x \in S} P(x) = 1. \qquad \blacktriangleleft$$

La primera condición garantiza que la probabilidad de un resultado sea no negativa y cuando mucho 1, y la segunda condición asegura que la suma de todas las probabilidad sea 1, es decir, que ocurrirá algún resultado cuando se realice el experimento.

Definición 6.5.2 ▶

Suponga que un dado está cargado de forma que los números 2 a 6 son igualmente probables, pero que 1 es tres veces más probable que cualquier otro número. Para modelar esta situación debe tenerse

$$P(2) = P(3) = P(4) = P(5) = P(6)$$

† Esta sección se puede omitir sin pérdida de continuidad.

y

$$P(1) = 3P(2).$$

Como

$$\begin{aligned} 1 &= P(1) + P(2) + P(3) + P(4) + P(5) + P(6) \\ &= 3P(2) + P(2) + P(2) + P(2) + P(2) + P(2) = 8P(2), \end{aligned}$$

debemos tener $P(2) = 1/8$. Por lo tanto,

$$P(2) = P(3) = P(4) = P(5) = P(6) = \frac{1}{8}$$

y

$$P(1) = 3P(2) = \frac{3}{8}.$$ ◀

La **probabilidad de un evento *E*** se define como la suma de las probabilidades de los resultados en E.

Definición 6.5.3 ▶

Sea E un evento. La *probabilidad de E*, $P(E)$, es

$$P(E) = \sum_{x \in E} P(x).$$

Ejemplo 6.5.4 ▶

Dadas las suposiciones del ejemplo 6.5.2, la probabilidad de obtener un número impar es

$$P(1) + P(3) + P(5) = \frac{3}{8} + \frac{1}{8} + \frac{1}{8} = \frac{5}{8}.$$

Por supuesto, con un dado no cargado (con probabilidades iguales), la probabilidad de obtener un número impar es $1/2$. ◀

Fórmulas

A continuación se desarrollan algunas fórmulas que resultan útiles para calcular probabilidades.

Teorema 6.5.5

Sea E un evento. La probabilidad de $\overline{E}$, el complemento de E, satisface

$$P(E) + P(\overline{E}) = 1.$$

Demostración Suponga que

$$E = \{x_1 \ldots, x_k\}$$

y

$$\overline{E} = \{x_{k+1}, \ldots, x_n\}.$$

Entonces

$$P(E) = \sum_{i=1}^{k} P(x_i) \qquad \text{y} \qquad P(\overline{E}) = \sum_{i=k+1}^{n} P(x_i).$$

Ahora

$$P(E) + P(\overline{E}) = \sum_{i=1}^{k} P(x_i) + \sum_{i=k+1}^{n} P(x_i)$$
$$= \sum_{i=1}^{n} P(x_i) = 1.$$

La última igualdad se deriva de la definición 6.5.1, que establece que la suma de las probabilidades de todos los resultados es igual a 1.

El Teorema 6.5.5 con frecuencia es útil cuando es más fácil calcular $\overline{E}$ que $P(E)$. Después de calcular $\overline{E}$, se puede obtener $P(E)$ restando $\overline{E}$ de 1.

Ejemplo 6.5.6 ▶

Se seleccionan aleatoriamente cinco microprocesadores de un lote de 1000 que contiene 20 defectuosos. En el ejemplo 6.4.4, se encontró que la probabilidad de no obtener microprocesadores defectuosos es 0.903735781. Por el Teorema 6.5.5, la probabilidad de obtener al menos uno defectuoso es

$$1 - 0.903735781 = 0.096264219.$$ ◀

Ejemplo 6.5.7 ▶

Problema del cumpleaños

WWW

Encuentre la probabilidad de que entre n personas, al menos dos hayan nacido el mismo mes y día (pero no necesariamente en el mismo año). Suponga que todos los meses y días son igualmente probables e ignore los cumpleaños del 29 de febrero.

Sea E el evento "al menos dos personas tienen el mismo cumpleaños". Entonces $\overline{E}$ es el evento "ningún par de personas tienen el mismo cumpleaños". Como se verá, es más sencillo calcular $P(\overline{E})$ que $P(E)$. Se utiliza el Teorema 6.5.5 para obtener la probabilidad deseada.

Como todos los meses y días tienen la misma probabilidad y se ignorarán los cumpleaños del 29 de febrero, el tamaño del espacio muestra es

$$365^n.$$

El cumpleaños de la primera persona puede ocurrir cualquiera de los 365 días. Si no hay dos personas que tengan el mismo cumpleaños, el cumpleaños de la segunda persona puede ocurrir cualquier día excepto el del cumpleaños de la primera persona. Por lo tanto, el cumpleaños de la segunda persona puede ocurrir en cualquiera de 364 días. De manera similar, el cumpleaños de la tercera persona puede ocurrir en cualquiera de 363 días. Se concluye que el tamaño del evento "ningún par de personas tienen el mismo cumpleaños" es

$$365 \cdot 364 \cdots (365 - n + 1).$$

Por el teorema 6.5.5, la probabilidad de que al menos dos personas tengan cumpleaños el mismo mes y día es

$$1 - \frac{365 \cdot 364 \cdots (365 - n + 1)}{365^n}.$$

Para $n = 22$, la probabilidad es 0.475695 y para $n = 23$, la probabilidad es 0.507297. Así, si $n \geq 23$, hay una probabilidad mayor que $1/2$ de que al menos dos personas tengan el mismo cumpleaños. Muchas personas pensarían que n tiene que ser *considerablemente* más grande que 23 para que la probabilidad sea mayor que $1/2$. ◀

Si E_1 y E_2 son eventos. el evento $E_1 \cup E_2$ representa el evento E_1 *o* E_2 (o ambos), y el evento $E_1 \cap E_2$ representa el evento E_1 *y* E_2.

Ejemplo 6.5.8 ▶

En un grupo de estudiantes, algunos toman arte y algunos computación. Se selecciona un estudiante de manera aleatoria. Sea A el evento "el estudiante toma arte" y sea C el evento "el

estudiante toma computación". Entonces $A \cup C$ es el evento el estudiante toma arte o computación o ambas", y $A \cap C$ es el evento "el estudiante toma arte y computación". ◀

El siguiente teorema da una fórmula para la probabilidad de la unión de dos eventos.

Teorema 6.5.9

Sean E_1 y E_2 dos eventos. Entonces

$$P(E_1 \cup E_2) = P(E_1) + P(E_2) - P(E_1 \cap E_2).$$

Demostración Sea

$$E_1 = \{x_1, \ldots, x_i\}$$
$$E_2 = \{y_1, \ldots, y_j\}$$
$$E_1 \cap E_2 = \{z_1, \ldots, z_k\},$$

y suponga que cada elemento de un conjunto se lista exactamente una vez por conjunto (vea la figura 6.5.1). Entonces en la lista

$$x_1, \ldots, x_i, y_1, \ldots, y_j,$$

$z_1, \ldots, z_k$ ocurre dos veces. Se concluye que

$$\begin{aligned} P(E_1 \cup E_2) &= \sum_{t=1}^{i} P(x_t) + \sum_{t=1}^{j} P(y_t) - \sum_{t=1}^{k} P(z_k) \\ &= P(E_1) + P(E_2) - P(E_1 \cap E_2). \end{aligned}$$

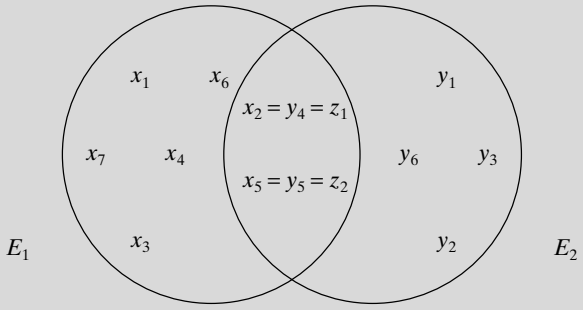


Figura 6.5.1 Los eventos E_1 y E_2. Las x denotan los elementos de E_1, las y denotan a los elementos de E_2, y las z denotan a los elementos de $E_1 \cap E_2$. Entonces, las z se cuentan dos veces: una vez entre las x y de nuevo entre las y.

Ejemplo 6.5.10 ▶

Se lanzan dos dados no cargados. ¿Cuál es la probabilidad de obtener dobles (el mismo número en los dos dados) o una suma de 6?

Sea E_1 el evento "salen dobles" y E_2 el evento "la suma es 6". Como los dobles se pueden obtener de seis maneras.

$$P(E_1) = \frac{6}{36} = \frac{1}{6}.$$

Como una suma de 6 se puede obtener de 5 maneras [(1, 5), (2, 4), (3, 3), (4, 2), (5, 1)],

$$P(E_2) = \frac{5}{36}.$$

El evento $E_1 \cap E_2$ es "salen dobles *y* la suma es 6". Como este último evento puede ocurrir sólo de una manera (con dos números 3),

$$P(E_1 \cap E_2) = \frac{1}{36}.$$

Por el Teorema 6.5.9, la probabilidad de obtener dobles o una suma de 6 es

$$P(E_1 \cup E_2) = P(E_1) + P(E_2) - P(E_1 \cap E_2) = \frac{1}{6} + \frac{5}{36} - \frac{1}{36} = \frac{5}{18}.$$ ◀

Los eventos E_1 y E_2 son **mutuamente excluyentes** si $E_1 \cap E_2 = \emptyset$. Se concluye, por el teorema 6.5.9, que si E_1 y E_2 son mutuamente excluyentes,

$$P(E_1 \cup E_2) = P(E_1) + P(E_2).$$

Corolario 6.5.11

Si E_1 y E_2 son eventos mutuamente excluyentes,

$$P(E_1 \cup E_2) = P(E_1) + P(E_2).$$

Demostración Como E_1 y E_2 son eventos mutuamente excluyentes, $E_1 \cap E_2 = \emptyset$. Por lo tanto, $P(E_1 \cap E_2) = 0$. El Teorema 6.5.9 ahora da

$$P(E_1 \cup E_2) = P(E_1) + P(E_2) - P(E_1 \cap E_2) = P(E_1) + P(E_2).$$

Ejemplo 6.5.12 ▶

Se lanzan dos dados no cargados. Encuentre la probabilidad de obtener dobles o una suma de 5.

Sea E_1 el evento "salen dobles" y sea E_2 el evento "la suma es 5". Note que E_1 y E_2 son mutuamente excluyentes: no pueden salir dobles y una suma de 5 al mismo tiempo. Como salen dobles de 6 maneras,

$$P(E_1) = \frac{6}{36} = \frac{1}{6}.$$

Como la suma de 5 se puede obtener de 4 maneras [(1, 4), (2, 3), (3, 2), (4, 1)],

$$P(E_2) = \frac{4}{36} = \frac{1}{9}.$$

Por el corolario 6.5.11,

$$P(E_1 \cup E_2) = P(E_1) + P(E_2) = \frac{1}{6} + \frac{1}{9} = \frac{5}{18}.$$ ◀

Probabilidad de condicional

Suponga que se lanzan dos dados no cargados. El espacio muestra consiste en los 36 resultados posibles; a cada uno se asigna un valor de 1/36 (figura 6.5.2). La probabilidad de obtener una suma de 10 es 1/12, la suma de los valores de los resultados que dan 10.

1, 1	2, 1	3, 1	4, 1	5, 1	6, 1
1, 2	2, 2	3, 2	4, 2	5, 2	6, 2
1, 3	2, 3	3, 3	4, 3	5, 3	6, 3
1, 4	2, 4	3, 4	4, 4	5, 4	6, 4
1, 5	2, 5	3, 5	4, 5	5, 5	6, 5
1, 6	2, 6	3, 6	4, 6	5, 6	6, 6

Figura 6.5.2 Lanzamiento de dos dados no cargados. Como a cada resultado se asigna el valor 1/36, la probabilidad de que la suma sea 10 es 1/12. Si al menos un dado muestra un 5, ocurre uno de los resultados sombreados. Los resultados sombreados se convierten en el nuevo espacio muestra y se asigna a cada resultado sombreado el valor 1/11. La probabilidad de que la suma sea 10 una vez que ocurre al menos un 5 es 1/11.

Se modificará un poco el ejemplo. Suponga que se lanzan dos dados y que nos dicen que al menos uno de ellos es 5. Ahora la probabilidad de que la suma sea 10 ya no es $1/12$ porque sabemos que ha ocurrido uno de los resultados sombreados en la figura 6.5.2. Como los 11 resultados sombreados son igualmente probables, la probabilidad de obtener una suma de 10 una vez que al menos un dado tiene 5 es $1/11$. Una probabilidad luego de que ocurrió un evento se llama **probabilidad condicional**.

Ahora se analizarán las probabilidades condicionales en general. Sea $P(E|F)$ la probabilidad de E dado F. En esta situación, F se convierte en el nuevo espacio muestra. Como los valores de los resultados en F originalmente sumaban $P(F)$, ahora se cambia el valor de cada resultado en F dividiéndolo entre $P(F)$ de manera que los valores reasignados sumen 1. Los resultados que satisfacen E dado que ocurrió F son precisamente los resultados en $E \cap F$. Si se suman los valores reasignados de los resultados en $E \cap F$, se obtiene el valor de $P(E|F)$:

$$\frac{P(E \cap F)}{P(F)}.$$

Este análisis motiva la siguiente definición.

Definición 6.5.13 ▶ Sean E y F dos eventos y suponga que $P(F) > 0$. La *probabilidad condicional de E dado F* es

$$P(E \mid F) = \frac{P(E \cap F)}{P(F)}.$$ ◀

Ejemplo 6.5.14 ▶ Se usa la definición 6.5.13 para calcular la probabilidad de que la suma sea 10, una vez que al menos un dado muestra 5, cuando se lanzan dos dados no cargados.

Sea E el evento "la suma es 10", y sea F el evento "al menos un dado muestra 5". El evento $E \cap F$ es "la suma es 10 *y* al menos un dado muestra 5". Como sólo un resultado pertenece a $E \cap F$,

$$P(E \cap F) = \frac{1}{36}.$$

Como 11 resultados pertenecen a F (vea la figura 6.5.2),

$$P(F) = \frac{11}{36}.$$

Por lo tanto,

$$P(E \mid F) = \frac{P(E \cap F)}{P(F)} = \frac{\frac{1}{36}}{\frac{11}{36}} = \frac{1}{11}.$$ ◀

Ejemplo 6.5.15 ▶ Los registros del clima muestran que la probabilidad de una presión barométrica alta es de 0.80, y que la probabilidad de lluvia y alta presión barométrica es de 0.10. Usando la definición 6.5.13, la probabilidad de lluvia dada una presión barométrica alta es

$$P(R \mid H) = \frac{P(R \cap H)}{P(H)} = \frac{0.10}{0.80} = 0.125,$$

donde R denota el evento "lluvia" y H denota el evento "presión barométrica alta". ◀

Eventos independientes

Si la probabilidad del evento E no depende del evento F en el sentido de que $P(E|F) = P(E)$, decimos que E y F son **eventos independientes**. Por la definición 6.5.13,

$$P(E \mid F) = \frac{P(E \cap F)}{P(F)}.$$

Entonces si E y F son eventos independientes,

$$P(E) = P(E \mid F) = \frac{P(E \cap F)}{P(F)}$$

o

$$P(E \cap F) = P(E)P(F).$$

Esta última ecuación se toma como la definición formal de eventos independientes.

Definición 6.5.16 ▶

Los eventos E y F son *independientes* si

$$P(E \cap F) = P(E)P(F).$$ ◀

Ejemplo 6.5.17 ▶

En forma intuitiva, si se lanza dos veces al aire una moneda no cargada, el resultado del segundo lanzamiento no depende del resultado del primero (después de todo, las monedas no tienen memoria). Por ejemplo, si H es el evento "cara en el primer lanzamiento" y T es el evento "cruz en el segundo lanzamiento", se espera que los eventos H y T sean independientes. Se usa la definición 6.5.16 para verificar que H y T de hecho son independientes.

El evento $H \cap T$ es el evento "cara en el primer lanzamiento y cruz en el segundo". Entonces $P(H \cap T) = 1/4$. Como $P(H) = 1/2 = P(T)$, se tiene

$$P(H \cap T) = \frac{1}{4} = \left(\frac{1}{2}\right)\left(\frac{1}{2}\right) = P(H)P(T).$$

Por lo tanto, los eventos H y T son independientes. ◀

Ejemplo 6.5.18 ▶

Jorge y Alicia presentan un examen final de matemáticas discretas. La probabilidad de que Jorge pase es de 0.70 y la probabilidad de que Alicia pase es de 0.95. Suponiendo que los eventos "Jorge pasa" y "Alicia pasa" son independientes, encuentre la probabilidad de que Jorge o Alicia, o ambos, pasen el examen final.

Suponga que J denota el evento "Jorge pasa el examen final" y A denota el evento "Alicia pasa el examen final". Calcule $P(J \cup A)$.

El Teorema 6.5.9 dice que

$$P(J \cup A) = P(J) + P(A) - P(J \cap A).$$

Puesto que se conocen $P(J)$ y $P(A)$, sólo se necesita calcular $P(J \cap A)$. Como los eventos J y A son *independientes*, la definición 6.5.16 dice que

$$P(J \cap A) = P(J)P(A) = (0.70)(0.95) = 0.665.$$

Por lo tanto,

$$P(J \cup A) = P(J) + P(A) - P(J \cap A) = 0.70 + 0.95 - 0.665 = 0.985.$$ ◀

Reconocimiento de patrones y el teorema de Bayes

El reconocimiento de patrones coloca elementos en *clases* basándose en las *características* de los elementos. Por ejemplo, el vino puede colocarse en las clases *de primera*, *vinos de mesa* y *de pobre calidad* según características como acidez y buqué. Una manera de hacer esta clasificación usa la teoría de probabilidad. A partir de un conjunto de características F, se calcula la probabilidad de una clase dado F para cada clase y se coloca el elemento en la clase más probable; es decir, la clase C elegida es aquella para la que $P(C \mid F)$ es la mayor.

Ejemplo 6.5.19 ▶

Sea R la clase *de primera*, T la clase *vinos de mesa* y S la clase *de pobre calidad*. Suponga que un vino específico tiene el conjunto de características F y

$$P(R \mid F) = 0.2, \qquad P(T \mid F) = 0.5, \qquad P(S \mid F) = 0.3.$$

Como la clase *vinos de mesa* tiene la mayor probabilidad, este vino se clasificaría como *vino de mesa*. ◀

El **Teorema de Bayes** es útil al calcular la probabilidad de una clase dado un conjunto de características.

Teorema 6.5.20

Teorema de Bayes

Suponga que las clases posibles son $C_1, \ldots, C_n$. Suponga además que las clases son mutuamente excluyentes por pares y que cada elemento que se va a clasificar pertenece a una de las clases. Para un conjunto de características F, se tiene

$$P(C_j \mid F) = \frac{P(F \mid C_j)P(C_j)}{\sum_{i=1}^{n} P(F \mid C_i)P(C_i)}.$$

Demostración Por la definición 6.5.13,

$$P(C_j \mid F) = \frac{P(C_j \cap F)}{P(F)},$$

y de nuevo por la definición 6.5.13,

$$P(F \mid C_j) = \frac{P(F \cap C_j)}{P(C_j)}.$$

Al combinar estas ecuaciones se obtiene

$$P(C_j \mid F) = \frac{P(C_j \cap F)}{P(F)} = \frac{P(F \mid C_j)P(C_j)}{P(F)}.$$

Para completar la prueba del Teorema de Bayes, es necesario demostrar que

$$P(F) = \sum_{i=1}^{n} P(F \mid C_i)P(C_i).$$

Puesto que cada elemento que se va a clasificar pertenece a una de las clases, se tiene

$$F = (F \cap C_1) \cup (F \cap C_2) \cup \cdots \cup (F \cap C_n).$$

Como los C_i son mutuamente excluyentes por pares, los $F \cap C_i$ también son mutuamente excluyentes. Por el corolario 6.5.11,

$$P(F) = P(F \cap C_1) + P(F \cap C_2) + \cdots + P(F \cap C_n).$$

De nuevo por la definición 6.5.13,

$$P(F \cap C_i) = P(F \mid C_i)P(C_i).$$

Por lo tanto,

$$P(F) = \sum_{i=1}^{n} P(F \mid C_i)P(C_i),$$

y esto completa la demostración.

Ejemplo 6.5.21 ▶

Telemercadeo

En la compañía de telemercadeo *SellPhone*, Dalia, Roberto y Leo hacen llamadas. La tabla siguiente muestra el porcentaje de llamadas que hace cada representante y el porcentaje de

personas que se molestan y cuelgan.

	Representante		
	Dalia	*Roberto*	*Leo*
Porcentaje de llamadas	40	25	35
Porcentaje de llamadas colgadas	20	55	30

Sea D el evento "Dalia llama", sea R el evento "Roberto llama", sea L el evento "Leo llama" y sea H el evento "colgaron". Encuentre $P(D)$, $P(R)$, $P(L)$, $P(H \mid D)$, $P(H \mid R)$, $P(H \mid L)$, $P(D \mid H)$, $P(R \mid H)$, $P(L \mid H)$, y $P(H)$.

Como Dalia hizo el 40% de las llamadas,

$$P(D) = 0.4.$$

En forma similar, de la tabla se obtiene

$$P(R) = 0.25, \qquad P(L) = 0.35.$$

Dado que Dalia hizo una llamada, la tabla muestra que el 20% de las personas cuelgan; por lo tanto,

$$P(H \mid D) = 0.2.$$

De manera similar,

$$P(H \mid R) = 0.55, \qquad P(H \mid L) = 0.3.$$

Para calcular $P(D \mid H)$, se usa el Teorema de Bayes:

$$\begin{aligned} P(D \mid H) &= \frac{P(H \mid D)P(D)}{P(H \mid D)P(D) + P(H \mid R)P(R) + P(H \mid L)P(L)} \\ &= \frac{(0.2)(0.4)}{(0.2)(0.4) + (0.55)(0.25) + (0.3)(0.35)} = 0.248. \end{aligned}$$

Un cálculo parecido usando el Teorema de Bayes da

$$P(R \mid H) = 0.426.$$

De nuevo usando el Teorema de Bayes o notando que

$$P(D \mid H) + P(R \mid H) + P(L \mid H) = 1,$$

obtenemos

$$P(L \mid H) = 0.326.$$

Por ultimo, la prueba del teorema de Bayes muestra que

$$\begin{aligned} P(H) &= P(H \mid D)P(D) + P(H \mid R)P(R) + P(H \mid L)P(L) \\ &= (0.2)(0.4) + (0.55)(0.25) + (0.3)(0.35) = 0.3225. \end{aligned}$$

◀

El último ejemplo de esta sección fue sugerido por Steve Jost.

Ejemplo 6.5.22 ▶

Detección del virus VIH

La prueba inmunoenzimática absorbente, conocida comúnmente como la prueba de ELISA, se usa para detectar anticuerpos en la sangre y puede indicar la presencia del virus VIH. Aproximadamente el 15% de los pacientes en una clínica tiene el VIH. Más aún, la prueba de ELISA sale positiva en cerca del 95% de quienes tienen VIH. Entre quienes no tienen el

virus VIH, alrededor del 2% de las pruebas de ELISA son positivas. Encuentre la probabilidad de que un paciente tenga el virus VIH si la prueba de ELISA es positiva.

Las clases son "tiene el virus VIH", que se denota por H, y "no tiene el virus VIH" $\overline{H}$. La característica "la prueba es positiva" se denota por *Pos*. Usando esta notación, la información dada se escribe

$$P(H) = 0.15, \quad P(\overline{H}) = 0.85, \quad P(Pos \mid H) = 0.95, \quad P(Pos \mid \overline{H}) = 0.02.$$

El teorema de Bayes da la probabilidad que se busca:

$$P(H \mid Pos) = \frac{P(Pos \mid H)P(H)}{P(Pos \mid H)P(H) + P(Pos \mid \overline{H})P(\overline{H})}$$
$$= \frac{(0.95)(0.15)}{(0.95)(0.15) + (0.02)(0.85)} = 0.893.$$

◀

Sección de ejercicios de repaso

1. ¿Qué es una función de probabilidad?
2. Si P es una función de probabilidad y todos los resultados son igualmente probables, ¿cuál es el valor de $P(x)$, donde x es un resultado?
3. ¿Cómo se define la probabilidad de un evento?
4. Si E es un evento, ¿cuál es la relación entre $P(E)$ y $\overline{E}$?
5. Si E_1 y E_2 son eventos, ¿qué representa el evento $E_1 \cup E_2$?
6. Si E_1 y E_2 son eventos, ¿qué representa el evento $E_1 \cap E_2$?
7. Si E_1 y E_2 son eventos, ¿cuál es la relación entre $P(E_1 \cup E_2)$, $P(E_1 \cap E_2)$, $P(E_1)$ y $P(E_2)$? Explique cómo se deriva la fórmula.
8. Explique qué significa que dos eventos sean mutuamente excluyentes.
9. Dé un ejemplo de eventos mutuamente excluyentes.
10. Si E_1 y E_2 son eventos mutuamente excluyentes, ¿cuál es la relación entre $P(E_1 \cup E_2)$, $P(E_1)$ y $P(E_2)$? Explique cómo se deriva la fórmula.
11. Proporcione una descripción informal, intuitiva del significado del evento *E dado F*.
12. ¿Cómo se denota el evento *E dado F*?
13. Dé una fórmula para la probabilidad de *E dado F*.
14. Explique qué significa que dos eventos sean "independientes".
15. Dé un ejemplo de eventos independientes.
16. ¿Qué es el reconocimiento de patrones?
17. Enuncie el Teorema de Bayes. Explique cómo se deriva la fórmula.

Ejercicios

Los ejercicios 1 al 3 se refieren al ejemplo 6.5.2 donde se carga un dado de manera que los números 2 al 6 tienen la misma probabilidad de salir, pero el 1 es tres veces más probable que cualquiera de los otros números.

1. Se lanza un dado. ¿Cuál es la probabilidad de obtener 5?
2. Se lanza un dado. ¿Cuál es la probabilidad de obtener un número par?
3. Se lanza un dado. ¿Cuál es la probabilidad de no obtener 5?

Los ejercicios 4 al 13 se refieren a un dado que se carga de manera que los números de 2, 4 y 6 tienen la misma probabilidad de salir; 1, 3 y 5 también son igualmente probables, pero el 1 es tres veces más probable que el 2.

4. Se lanza un dado. Asigne probabilidades a los resultados que modelen con exactitud las posibilidades de que salgan los diferentes números.
5. Se lanza un dado. ¿Cuál es la probabilidad de que salga 5?
6. Se lanza un dado. ¿Cuál es la probabilidad de obtener un número par?
7. Se lanza un dado. ¿Cuál es la probabilidad de que no salga 5?
8. Se lanzan dos dados. ¿Cuál es la probabilidad de obtener dobles?
9. Se lanzan dos dados. ¿Cuál es la probabilidad de que la suma sea 7?
10. Se lanzan dos dados. ¿Cuál es la probabilidad de que salgan dobles o que la suma sea 6?
11. Se lanzan dos dados. ¿Cuál es la probabilidad de que la suma sea 6 toda vez que al menos un dado muestra 2?
12. Se lanzan dos dados. ¿Cuál es la probabilidad de que la suma sea 6 o de que salgan dobles toda vez que al menos un dado muestra 2?
13. Se lanzan dos dados. ¿Cuál es la probabilidad de obtener una suma de 6 o una suma de 8 toda vez que al menos un dado muestra 2?

En los ejercicios 14 al 18, suponga que se lanzan una moneda y un dado. Sea E_1 el evento "la moneda muestra cruz", E_2 el evento "el dado muestra 3" y E_3 el evento "la moneda muestra cara y el dado un número impar".

14. Liste los elementos del evento E_1 o E_2.
15. Liste los elementos del evento E_2 o E_3.

16. ¿Son E_1 y E_2 mutuamente excluyentes?

17. ¿Son E_1 y E_3 mutuamente excluyentes?

18. ¿Son E_2 y E_3 mutuamente excluyentes?

19. Se seleccionan aleatoriamente seis microprocesadores de un lote de 100, entre los que 10 son defectuosos. Encuentre la probabilidad de obtener microprocesadores no defectuosos.

20. Se seleccionan aleatoriamente seis microprocesadores de un lote de 100, entre los que 10 son defectuosos. Encuentre la probabilidad de obtener al menos un microprocesador defectuoso.

21. Se seleccionan aleatoriamente seis microprocesadores de un lote de 100, entre los que 10 son defectuosos. Encuentre la probabilidad de obtener al menos tres microprocesadores defectuosos.

Los ejercicios 22 al 29 se refieren a una familia con cuatro hijos. Suponga que es igualmente probable que nazca un niño que una niña.

22. ¿Cuál es la probabilidad de que todas sean niñas?

23. ¿Cuál es la probabilidad de que haya exactamente dos niñas?

24. ¿Cuál es la probabilidad de al menos un niño y al menos una niña?

25. ¿Cuál es la probabilidad de que todos sean niñas dado que al menos uno es niña.

26. ¿Cuál es la probabilidad de exactamente dos niñas dado que hay al menos una niña?

27. ¿Cuál es la probabilidad de al menos un niño y al menos una niña dado que hay al menos una niña?

28. ¿Son independientes los eventos "hay niños de uno y otro sexo" y "hay a lo sumo un niño"?

29. ¿Son independientes los eventos "hay cuando mucho un niño" y "hay cuando mucho una niña"?

30. Una familia tiene n hijos. Suponga que es igualmente probable que nazca una niña o un niño. ¿Para qué valores de n son independientes los eventos "hay niños de uno y otro sexo" y "hay cuando mucho una niña"?

Los ejercicios 31 al 39 se refieren a lanzamiento repetido de una moneda no cargada.

31. Si se lanza la moneda 10 veces, ¿cuál es la probabilidad de que no salgan caras?

32. Si la moneda se lanza 10 veces, ¿cuál es la probabilidad de que salgan exactamente cinco caras?

33. Si la moneda se lanza 10 veces, ¿cuál es la probabilidad de "aproximadamente" cinco caras, es decir, exactamente 4, 5 o 6 caras?

34. Si la moneda se lanza 10 veces, ¿cuál es la probabilidad de al menos una cara?

35. Si la moneda se lanza 10 veces, ¿cuál es la probabilidad de cuando mucho cinco caras?

36. Si la moneda se lanza 10 veces, ¿cuál es la probabilidad de exactamente cinco caras dado que hay al menos una cara?

37. Si la moneda se lanza 10 veces, ¿cuál es la probabilidad de "aproximadamente" 5 caras, es decir, exactamente 4, 5 o 6 caras dado que salió al menos una cara?

38. Si la moneda se lanza 10 veces, ¿cuál es la probabilidad de al menos una cara dado que salió al menos una cara?

39. Si la moneda se lanza 10 veces, ¿cuál es la probabilidad de cuando mucho cinco caras dado que hay al menos una cruz?

40. Suponga que se selecciona un luchador profesional al azar entre 90 luchadores, de los cuales 35 pesan más de 350 libras, 20 son rudos, y 15 pesan más de 350 libras y son rudos. ¿Cuál es la probabilidad de que el luchador seleccionado pese más de 350 libras y sea rudo?

41. Suponga que la probabilidad de que una persona tenga dolor de cabeza es de 0.01, que la probabilidad de que una persona tenga fiebre dado que tiene dolor de cabeza es de 0.4, y que la probabilidad de que una persona tenga fiebre es de 0.02. Encuentre la probabilidad de que una persona tenga dolor de cabeza dado que tiene fiebre.

Los ejercicios 42 al 45 se refieren a una compañía que compra computadoras a tres vendedores y da seguimiento al número de máquinas defectuosas. La siguiente tabla muestra los resultados.

	Vendedor		
	Acme	*DotCom*	*Nuclear*
Porcentaje comprado	55	10	35
Porcentaje defectuoso	1	3	3

Sea A el evento "la computadora se compró a Acme", sea D el evento "la computadora se compró a DotCom", sea N el evento "la computadora se compró a Nuclear" y sea B el evento "la computadora estaba defectuosa".

42. Encuentre $P(A)$, $P(D)$ y $P(N)$.

43. Encuentre $P(B \mid A)$, $P(B \mid D)$, y $P(B \mid N)$.

44. Encuentre $P(A \mid B)$, $P(D \mid B)$, y $P(N \mid B)$.

45. Encuentre $P(B)$.

46. En el ejemplo 6.5.22, ¿qué tan pequeña deberá ser $P(H)$ para que la conclusión sea "no hay VIH" aun cuando el resultado de la prueba sea positivo?

47. Demuestre que para cualesquiera eventos E_1 y E_2.

$$P(E_1 \cap E_2) \geq P(E_1) + P(E_2) - 1.$$

48. Use inducción matemática para demostrar que si $E_1, E_2, \ldots, E_n$ son eventos, entonces

$$P(E_1 \cup E_2 \cup \cdots \cup E_n) \leq \sum_{i=1}^{n} P(E_i).$$

49. Si E y F son eventos independientes, ¿son $\overline{E}$ y $\overline{F}$ independientes?

50. Si E y F son eventos independientes, ¿son E y $\overline{F}$ independientes?

51. ¿Es correcto el siguiente razonamiento? Explique su respuesta.

Una persona preocupada por la posibilidad de una bomba en un avión estima que la probabilidad de tal evento es de 0.000001. No satisfecha con las posibilidades, la persona calcula la probabilidad de *dos* bombas en un avión como

$$0.000001^2 = 0.000000000001.$$

Satisfecha ahora con las posibilidades, la persona lleva una bomba siempre que viaja por avión de manera que la probabilidad de que alguien más lleve una bomba (y haya dos bombas en el avión) sea de 0.000000000001, suficientemente pequeña para estar a salvo.

52. Un atleta entusiasta decide competir en la carrera de Maratón del Este-Sureste. El atleta se retirará si termina la maratón o después de tres intentos. La probabilidad de terminar la maratón en un intento es de 1/3. Analice el siguiente argumento que, suponiendo independencia, demuestra que el atleta casi tiene la certidumbre (pero no por completo) de que terminará la maratón.

Como la probabilidad de cada intento es de $1/3 = 0.3333$, después de tres intentos, la probabilidad de completar la maratón es de 0.9999, que significa que el atleta está casi seguro, pero no por completo, de que terminará la maratón.

6.6 ➜ Permutaciones y combinaciones generalizadas

WWW

En la sección 6.2, se estudiaron los ordenamientos y las selecciones sin permitir repeticiones. En esta sección se consideran los ordenamientos de sucesiones que contienen repeticiones y selecciones no ordenadas en las que se permiten repeticiones.

Ejemplo 6.6.1 ▶

¿Cuántas cadenas se pueden formar usando las siguientes letras?

M I S S I S S I P P I

Por la duplicación de letras, las respuesta no es 11! sino un número menor que 11!. Considere el problema de llenar 11 espacios,

— — — — — — — — — — —,

con las letras dadas. Existen $C(11, 2)$ maneras de elegir posiciones para dos letras *P*. Una vez seleccionadas esas posiciones, existen $C(9, 4)$ maneras de elegir posiciones para cuatro letras *S*. Una vez seleccionadas esas posiciones, existen $C(5, 4)$ maneras de elegir posiciones para las cuatro letras *I*. Después de hacer estas selecciones, queda una posición por llenar por la *M*. Mediante el principio de la multiplicación, el número de maneras de ordenar las letras es

$$C(11, 2)C(9, 4)C(5, 4) = \frac{11!}{2!\,9!}\frac{9!}{4!\,5!}\frac{5!}{4!\,1!} = \frac{11!}{2!\,4!\,4!\,1!} = 34{,}650. \quad \blacktriangleleft$$

La solución al ejemplo 6.61 adopta una forma agradable. El número 11 que aparece en el numerador es el número total de letras. Los valores en el denominador dan el número de duplicados de cada letra. El método se puede usar para establecer un fórmula general.

Teorema 6.6.2

Suponga que una sucesión S de n artículos tiene n_1 objetos idénticos del tipo 1, n_2 objetos idénticos del tipo 2, . . . , n_t objetos idénticos del tipo t. Entonces, el número de ordenamientos de S es

$$\frac{n!}{n_1!\,n_2!\cdots n_t!}.$$

Demostración Se asignan posiciones a cada uno de los *n* artículos para crear un ordenamiento de *S*. Se pueden asignar posiciones a los n_1 objetos del tipo 1 de $C(n, n_1)$ maneras. Después de hacer estas asignaciones, se pueden asignar posiciones a los n_2 artículos del tipo 2 de $C(n - n_1, n_2)$ maneras, y así sucesivamente. Por el principio de la multiplicación, el número de ordenamientos es

$$\begin{aligned} &C(n, n_1)C(n - n_1, n_2)C(n - n_1 - n_2, n_3)\cdots C(n - n_1 - \cdots - n_{t-1}, n_t) \\ &= \frac{n!}{n_1!\,(n - n_1)!}\frac{(n - n_1)!}{n_2!\,(n - n_1 - n_2)!}\cdots\frac{(n - n_1 - \cdots - n_{t-1})!}{n_t!\,0!} \\ &= \frac{n!}{n_1!\,n_2!\cdots n_t!}. \end{aligned}$$

Ejemplo 6.6.3 ▶

¿De cuántas maneras pueden dividirse 8 libros diferentes entre 3 estudiantes si Brenda obtiene 4 libros, y Samuel y Mariana 2 cada uno?

Coloque los libros en algún orden fijo. Ahora considere los ordenamientos de 4 libros de *B*, 2 libros de *S* y dos libros de *M*. Un ejemplo es

B B B S M B M S.

Cada ordenamiento determina una distribución de los libros. Para el ordenamiento anterior, Brenda obtiene los libros 1, 2, 3 y 6, Samuel obtiene los libros 4 y 8, y Mariana los libros

5 y 7. Así, el número de maneras para ordenar *BBBBSSMM* es el número de maneras para distribuir los libros. Por el Teorema 6.6.2, este número es

$$\frac{8!}{4!\,2!\,2!} = 420.$$

◀

Se puede desarrollar otra prueba del Teorema 6.6.2 usando relaciones. Suponga que una sucesión S de n artículos tienen n_i objetos idénticos del tipo i para $i = 1, \ldots, t$. Sea X el conjunto de n elementos obtenidos de S al considerar los n_i objetos diferentes del tipo i *distintos* para $i = 1, \ldots, t$. Por ejemplo, si S es la sucesión de letras

M I S S I S S I P P I,

X sería el conjunto

$$\{M, I_1, S_1, S_2, I_2, S_3, S_4, I_3, P_1, P_2, I_4\}.$$

Se define una relación R en el conjunto de todas las permutaciones de X por la regla $p_1 R p_2$ si p_2 se obtiene de p_1 permutando el orden de los objetos tipo 1 (pero sin cambiar su posición) y/o permutando el orden de los objetos tipo 2 (pero sin cambiar su posición)... y/o permutando el orden de los objetos tipo t (pero sin cambiar su posición); por ejemplo,

$$(I_1 S_1 S_2 I_2 S_3 S_4 I_3 P_1 P_2 I_4 M)\ R\ (I_2 S_3 S_2 I_1 S_4 S_1 I_3 P_1 P_2 I_4 M).$$

Se verifica de manera directa que R es una relación de equivalencia en el conjunto de todas las permutaciones de X.

La clase de equivalencia que contiene la permutación p consiste en todas las permutaciones de X que son idénticas si consideramos los objetos tipo i idénticos para $i = 1, \ldots, t$. Entonces cada clase de equivalencia tiene $n_1!n_2! \cdots n_t!$ elementos. Como una clase de equivalencia está determinada por un ordenamiento de S, el número de ordenamientos de S es igual al número de clases de equivalencia. Existen $n!$ permutaciones de X y, por el Teorema 3.2.15, el número de ordenamientos de S es

$$\frac{n!}{n_1!\,n_2! \cdots n_t!}.$$

A continuación se estudiará el problema de contar selecciones no ordenadas cuando se permiten repeticiones.

Ejemplo 6.6.4 ▶

Considere 3 libros: de computación, física e historia. Suponga que la biblioteca tienen al menos 6 copias de cada uno. ¿De cuántas maneras se pueden seleccionar 6 libros?

El problema es elegir, sin importar el orden, selecciones de 6 elementos del conjunto {computación, física, historia}, con repeticiones permitidas. Una selección se determina de manera única mediante el número seleccionado de cada tipo de libro. Una selección en particular se denota como

Computación	Física	Historia
x x x	| x x |	x

Se ha designado la selección que consiste en 3 libros de computación, 2 de física y 1 de historia. Otro ejemplo de una selección es

Computación	Física	Historia
	| x x x x |	x x

que denota la selección que consiste en cero libros de computación, 4 de física y 2 de historia. Se observa que cada ordenamiento de seis x y dos | denota una selección. Entonces el problema es contar el número de este tipo de ordenamientos. Pero esto es justo el número de maneras

$$C(8, 2) = 28$$

de elegir dos posiciones para las | de ocho posiciones posibles. Así, existen 28 maneras de seleccionar seis libros. ◀

El método usado en el ejemplo 6.6.4 se puede usar para derivar un resultado general.

Teorema 6.6.5

Si X es un conjunto que contiene t elementos, el número de selecciones no ordenadas de k elementos de X, con repeticiones, es

$$C(k + t - 1, t - 1) = C(k + t - 1, k).$$

Demostración Sea $X = \{a_1, \ldots, a_t\}$. Considere los $k + t - 1$ espacios

$$\_ \; \_ \; \_ \cdots \_ \; \_$$

y $k + t - 1$ símbolos que consisten en k símbolos $\times$ y $t - 1$ símbolos $|$. Cada colocación de estos símbolos en los espacios determina una selección. El número n_1 de $\times$ hasta encontrar la primera $|$ representa la selección de n_1 a_1; el número n_2 de $\times$ entre la primera y la segunda $|$ representa la selección de n_2 a_2; y así sucesivamente. Como hay $C(k + t - 1, t - 1)$ maneras de seleccionar las posiciones para las $|$, también hay $C(k + t - 1, t - 1)$ selecciones. Esto es igual a $C(k + t - 1, k)$, el número de maneras de seleccionar las posiciones para las $\times$; entonces existen

$$C(k + t - 1, t - 1) = C(k + t - 1, k)$$

selecciones no ordenadas de k elementos de X, con repeticiones.

Ejemplo 6.6.6 ▶

Suponga que existen tres pilas de pelotas rojas, azules y verdes, y que cada pila contiene al menos 8 pelotas.

a) ¿De cuántas maneras se pueden seleccionar 8 pelotas?

b) ¿De cuántas maneras se pueden seleccionar 8 pelotas si debe tenerse al menos una pelota de cada color?

Por el Teorema 6.6.5, el número de maneras para seleccionar 8 pelotas es

$$C(8 + 3 - 1, 3 - 1) = C(10, 2) = 45.$$

También se puede usar el Teorema 6.6.5 para resolver el inciso *b)* si primero se selecciona una pelota de cada color. Para completar la selección, deben elegirse 5 pelotas adicionales. Esto se puede hacer de

$$C(5 + 3 - 1, 3 - 1) = C(7, 2) = 21$$

maneras. ◀

Ejemplo 6.6.7 ▶

¿De cuántas maneras pueden distribuirse 12 libros idénticos de matemáticas entre los estudiantes Ana, Beatriz, Carmen y Daniel?

Se puede usar el Teorema 6.6.5 para resolver este problema si se considera como un problema de etiquetar cada libro con el nombre del estudiante que lo recibe. Esto es lo mismo que seleccionar 12 artículos (los nombres de los estudiantes) del conjunto {Ana, Beatriz, Carmen, Daniel}, con repeticiones permitidas. Por el Teorema 6.6.5, el número de maneras de hacer esto es

$$C(12 + 4 - 1, 4 - 1) = C(15, 3) = 455.$$ ◀

Ejemplo 6.6.8 ▶

a) ¿Cuántas soluciones en enteros no negativos hay para la ecuación

$$x_1 + x_2 + x_3 + x_4 = 29? \tag{6.6.1}$$

b) ¿Cuántas soluciones enteras hay para (6.6.1) que satisfacen $x_1 > 0$, $x_2 > 1$, $x_3 > 2$, $x_4 \geq 0$?

a) Cada solución de (6.6.1) es equivalente a seleccionar 29 elementos, x_i del tipo i,

$i = 1, 2, 3, 4$. Según el Teorema 6.6.5, el número de selecciones es

$$C(29 + 4 - 1, 4 - 1) = C(32, 3) = 4960.$$

b) Cada solución de (6.6.1) que satisface las condiciones dadas es equivalente a seleccionar 29 elementos, x_i del tipo i, $i = 1, 2, 3, 4$, donde, además, debemos tener al menos un elemento tipo 1, al menos dos elementos tipo 2 y al menos 3 elementos tipo 3. Primero se selecciona un elemento tipo 1, dos elementos tipo 2 y tres elementos tipo 3. Después, se eligen 23 elementos adicionales. Por el Teorema 6.6.5, esto se puede hacer de

$$C(23 + 4 - 1, 4 - 1) = C(26, 3) = 2600$$

maneras. ◀

Ejemplo 6.6.9 ▶

¿Cuántas veces se ejecuta la instrucción de imprimir?

for $i_1 = 1$ to n
  for $i_2 = 1$ to i_1
    for $i_3 = 1$ to i_2
    ⋱
      for $i_k = 1$ to i_{k-1}
        println$(i_1, i_2, \ldots, i_k)$

Observe que cada línea de salida consiste en k enteros

$$i_1 i_2 \cdots i_k, \tag{6.6.2}$$

donde

$$n \geq i_1 \geq i_2 \geq \cdots \geq i_k \geq 1, \tag{6.6.3}$$

y ocurre cada sucesión (6.6.2) que satisface (6.6.3). Entonces el problema es contar el número de maneras de elegir k enteros, con repeticiones permitidas, del conjunto $\{1, 2, \ldots, n\}$. [Cualquiera de estas selecciones se puede ordenar para producir (6.6.3).] Por el Teorema 6.6.5, el número total de selecciones posibles es

$$C(k + n - 1, k).$$ ◀

Sugerencias para resolver problemas

Las fórmulas de la sección 6.6 generalizan las fórmulas de la sección 6.2 al permitir repeticiones. Una *permutación* es un ordenamiento de $s_1, \ldots, s_n$, donde las s_i son *distintas*. Existen $n!$ permutaciones. Ahora suponga que se tienen n artículos que contienen *duplicados*, en particular, n_i objetos idénticos de tipo i, para $i = 1, \ldots, t$. Entonces el número de ordenamientos es

$$\frac{n!}{n_1! n_2! \cdots n_t!}.$$

Para determinar si una de estas fórmulas es relevante para un problema específico, primero asegúrese de que el problema pide *ordenamientos*. Si los artículos que se van a ordenar son *distintos*, hay que usar la fórmula para permutaciones. Por otro lado, si hay *duplicados* entre los artículos que se van a ordenar, es conveniente utilizar la fórmula

$$\frac{n!}{n_1! n_2! \cdots n_t!}.$$

Una *combinación r* es una selección no ordenada de r elementos tomados de n elementos, *sin permitir repeticiones*. Existen $C(n, r)$ combinaciones r. Ahora suponga que se desea contar las selecciones no ordenadas de k elementos entre t elementos, *con repeticiones permitidas*. El número de estas selecciones es

$$C(k + t - 1, t - 1).$$

Para determinar si una de estas fórmulas es relevante para un problema determinado, primero asegúrese de que el problema pide selecciones *no ordenadas*. Si los artículos deben elegirse *sin repetición*, se utiliza la fórmula de combinación. Por otro lado, si los artículos deben elegirse *con repetición*, se emplea la fórmula

$$C(k + t - 1, t - 1) .$$

La tabla siguiente resume las fórmulas:

	Sin repetición	*Con repetición*
Selecciones ordenadas	$n!$	$n!/(n_1! \cdots n_t!)$
Selecciones no ordenadas	$C(n, r)$	$C(k + t - 1, t - 1)$

Sección de ejercicios de repaso

1. ¿Cuántos ordenamientos existen de n elementos de t tipos con n_i objetos idénticos del tipo i? ¿Cómo se deriva esta fórmula?

2. ¿Cuántas selecciones no ordenadas de k elementos hay, tomadas de un conjunto de t elementos, con repeticiones? ¿Cómo se deriva esta fórmula?

Ejercicios

En los ejercicios 1 al 3, determine el número de cadenas que se pueden formar al ordenar las letras indicadas.

1. *GUIDE* **2.** *SCHOOL*

3. *SALESPERSONS*

4. ¿Cuántas cadenas se pueden formar ordenando las letras *SALESPERSONS* si las cuatro *S* deben ser consecutivas?

5. ¿Cuántas cadenas se pueden formar ordenando las letras *SALESPERSONS* si dos *S* no pueden estar juntas?

6. ¿Cuántas cadenas se pueden formar ordenando las letras *SCHOOL* si se usan algunas o todas las letras?

Los ejercicios 7 al 9 se refieren a las selecciones entre las historietas cómicas Acción, Superman, Capitán Marvel, Archie, X-Man y Nancy.

7. ¿Cuántas maneras hay para seleccionar 6 historietas?

8. ¿Cuántas maneras hay para seleccionar 10 historietas?

9. ¿Cuántas maneras hay para seleccionar 10 historietas si elegimos al menos una de cada título?

10. ¿Cuántas rutas hay en el sistema de coordenadas xyz normal desde el origen al punto (i, j, k), donde i, j y k son enteros positivos, si estamos limitados a pasos unitarios en la dirección positiva de x, en la dirección positiva de y y en la dirección positiva de z?

11. Un examen tiene 12 problemas. ¿De cuántas maneras se pueden asignar puntos (enteros) a los problemas si el total es 100 y cada problema vale por lo menos 5 puntos?

12. Un coleccionista de bicicletas tiene 100 de ellas. ¿De cuántas maneras es posible guardar las bicicletas en cuatro almacenes si las bicicletas y los almacenes se consideran diferentes?

13. Un coleccionista de bicicletas tiene 100 de ellas. ¿De cuántas maneras se pueden almacenar las bicicletas en cuatro almacenes si las bicicletas son indistinguibles, pero los almacenes se consideran diferentes?

14. ¿De cuántas maneras se pueden dividir 10 libros diferentes entre 3 estudiantes si el primer estudiante obtiene 5 libros, el segundo 3 y el tercero 2 libros?

Los ejercicios 15 al 21 se refieren a pilas idénticas de pelotas rojas, azules y verdes, donde cada pila contiene por lo menos 10 pelotas.

15. ¿De cuántas maneras se pueden seleccionar 10 pelotas?

16. ¿De cuántas maneras se pueden seleccionar 10 pelotas si debe elegirse al menos una pelota roja?

17. ¿De cuántas maneras se pueden seleccionar 10 pelotas si debe haber al menos una roja, al menos 2 azules y al menos 3 verdes?

18. ¿De cuántas maneras se pueden seleccionar 10 pelotas si debe haber exactamente una pelota roja?

19. ¿De cuántas maneras se pueden seleccionar 10 pelotas si deben elegirse exactamente una pelota roja y al menos una azul?

20. ¿De cuántas maneras se pueden seleccionar 10 pelotas si debe haber cuando mucho una roja?

21. ¿De cuántas maneras se pueden seleccionar 10 pelotas si las pelotas rojas deben ser el doble que las verdes?

En los ejercicios 22 al 29, encuentre el número de soluciones enteras de

$$x_1 + x_2 + x_3 = 15$$

sujeto a las condiciones indicadas.

22. $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$

23. $x_1 \geq 1, x_2 \geq 1, x_3 \geq 1$

24. $x_1 = 1, x_2 \geq 0, x_3 \geq 0$

25. $x_1 \geq 0, x_2 > 0, x_3 = 1$

26. $0 \leq x_1 \leq 6, x_2 \geq 0, x_3 \geq 0$

★ **27.** $0 \leq x_1 < 6, 1 \leq x_2 < 9, x_3 \geq 0$

★ ***28.*** Encuentre el número de soluciones enteras de

$$x_1 + x_2 + x_3 + x_4 = 12$$

que satisfacen $0 \leq x_1 \leq 4$, $0 \leq x_2 \leq 5$, $0 \leq x_3 \leq 8$, y $0 \leq x_4 \leq 9$.

29. Demuestre que el número de soluciones en enteros no negativos de la desigualdad

$$x_1 + x_2 + \cdots + x_n \leq M,$$

donde M es un entero no negativo, es $C(M + n, n)$.

30. ¿Cuántos enteros entre 1 y 1,000,000 tienen la suma de dígitos igual a 15?

★ **31.** ¿Cuántos enteros entre 1 y 1,000,000 tienen la suma de dígitos igual a 20?

32. ¿Cuántas maneras de repartir en el bridge hay? (Repartir es lo mismo que hacer una partición de la baraja de 52 cartas en 4 manos, cada una con 13 cartas).

33. ¿De cuántas maneras pueden elegirse tres equipos que contienen 4, 2 y 2 personas, entre un grupo de 8 personas?

34. Una ficha de *dominó* es un rectángulo dividido en dos cuadros, con cada cuadro numerado de 0, 1, . . . , 6, con repeticiones. ¿Cuántas fichas diferentes de dominó hay?

Los ejercicios 35 al 40 se refieren a una bolsa que contiene 20 pelotas: 6 rojas, 6 verdes y 8 moradas.

35. ¿De cuántas maneras se pueden seleccionar 5 pelotas si todas se consideran diferentes?

36. ¿De cuántas maneras se pueden seleccionar 5 pelotas si las pelotas del mismo color se consideran idénticas?

37. ¿De cuántas maneras se pueden sacar 2 pelotas rojas, 3 verdes y 2 moradas, si todas las pelotas se consideran diferentes?

38. Se sacan 5 pelotas y se remplazan. Después se sacan otras 5 pelotas. ¿De cuántas maneras puede hacerse esto si las pelotas se consideran diferentes?

39. Se sacan 5 pelotas sin remplazarlas. Después se sacan otras 5 pelotas. ¿De cuántas maneras puede hacerse esto si las pelotas se consideran diferentes?

40. Se sacan 5 pelotas y al menos una es roja, después se remplazan. Luego se sacan 5 pelotas y cuando mucho una es verde. ¿De cuántas maneras puede hacerse esto si las pelotas se consideran diferentes?

41. ¿De cuántas maneras se pueden distribuir 15 libros de matemáticas idénticos entre 6 estudiantes?

42. ¿De cuántas maneras se pueden distribuir 15 libros de computación idénticos y 10 libros de psicología idénticos entre 5 estudiantes?

43. ¿De cuántas maneras se pueden colocar 10 pelotas idénticas en 12 cajas, si cada caja puede contener una pelota?

44. ¿De cuántas maneras se pueden colocar 10 pelotas idénticas en 12 cajas, si cada caja puede contener 10 pelotas?

45. Demuestre que $(kn)!$ es divisible entre $(n!)^k$.

46. Considere

```
for i1 = 1 to n
    for i2 = 1 to i1
        println(i1, i2)
```

y el ejemplo 6.6.9 para deducir

$$1 + 2 + \cdots + n = \frac{n(n+1)}{2}.$$

★ ***47.*** Use el ejemplo 6.6.9 para probar la fórmula

$$C(k-1, k-1) + C(k, k-1) + \cdots + C(n+k-2, k-1) = C(k+n-1, k).$$

48. Escriba un algoritmo que liste todas las soluciones en enteros no negativos de

$$x_1 + x_2 + x_3 = n.$$

49. ¿Qué está equivocado en el siguiente argumento, que pretende contar el número de particiones de un conjunto de 10 elementos en 8 subconjuntos (no vacíos)?

Liste los elementos con espacios entre ellos:

$$x_1 \_ x_2 \_ x_3 \_ x_4 \_ x_5 \_ x_6 \_ x_7 \_ x_8 \_ x_9 \_ x_{10}.$$

Cada vez que se llenan 7 de los 9 espacios con 7 barras verticales, se obtiene una partición de $\{x_1, \ldots, x_{10}\}$ en 8 subconjuntos. Por ejemplo, la partición $\{x_1\}, \{x_2\}, \{x_3, x_4\}\ \{x_5\}, \{x_6\}, \{x_7, x_8\}\ \{x_9\}, \{x_{10}\}$ se representaría como

$$x_1 \mid x_2 \mid x_3\, x_4 \mid x_5 \mid x_6 \mid x_7\, x_8 \mid x_9 \mid x_{10}.$$

Entonces, la solución al problema es $C(9, 7)$.

Los ejercicios 50 y 51 se refieren a 10 discos compactos idénticos que se dan al azar a María, Iván y Juan.

50. ¿Cuál es la probabilidad de que cada persona reciba al menos dos discos compactos?

51. ¿Cuál es la probabilidad de que Iván reciba exactamente 3 discos compactos?

6.7 ➜ Coeficientes binomiales e identidades combinatorias

A primera vista, la expresión $(a + b)^n$ no tiene mucho que ver con combinaciones; pero como se verá en esta sección, es posible obtener una fórmula para la expansión de $(a + b)^n$ usando la fórmula para el número de combinaciones r de n objetos. Con frecuencia, una expresión algebraica se relaciona con algún proceso de conteo. Varias técnicas de conteo avanzadas usan este tipo de métodos (vea [Riordan; y Tucker]).

WWW

El **teorema binomial** proporciona una fórmula para los coeficientes en la expansión de $(a + b)^n$. Como

$$(a+b)^n = \underbrace{(a+b)(a+b)\cdots(a+b)}_{n \text{ factores}}, \tag{6.7.1}$$

la expresión es el resultado de seleccionar a o b en cada uno de los n factores, multiplicando

TABLA 6.7.1 ■ Cálculo de $(a + b)^3$.

Selección en el primer factor $(a + b)$	*Selección en el segundo factor* $(a + b)$	*Selección en el tercer factor* $(a + b)$	*Producto de selecciones*
a	a	a	$aaa = a^3$
a	a	b	$aab = a^2b$
a	b	a	$aba = a^2b$
a	b	b	$abb = ab^2$
b	a	a	$baa = a^2b$
b	a	b	$bab = ab^2$
b	b	a	$bba = ab^2$
b	b	b	$bbb = b^3$

las selecciones y después sumando todos los productos obtenidos. Por ejemplo, en la expansión de $(a + b)^3$, se elige ya sea a o b en el primer factor $(a + b)$; ya sea a o b en el segundo factor $(a + b)$; y ya sea a o b en el tercer factor $(a + b)$; se multiplican las selecciones y luego se suman los productos obtenidos. Si se elige a en todos los factores y se multiplica, el resultado es el término aaa. Si se elige a en el primer factor, b en el segundo y a en el tercero y se multiplica, se obtiene el término aba. La tabla 6.7.1 muestra todas la posibilidades. Si se suman los productos de todas las selecciones, se obtiene

$$\begin{aligned}(a + b)^3 &= (a + b)(a + b)(a + b)\\ &= aaa + aab + aba + abb + baa + bab + bba + bbb\\ &= a^3 + a^2b + a^2b + ab^2 + a^2b + ab^2 + ab^2 + b^3\\ &= a^3 + 3a^2b + 3ab^2 + b^3.\end{aligned}$$

En (6.7.1), un término de la forma $a^{n-k}b^k$ surge al elegir b en k factores y a en los otros $n - k$ factores. Pero esto se puede hacer de $C(n, k)$ maneras, ya que $C(n, k)$ cuenta el número de maneras de seleccionar k objetos entre n objetos. Entonces $a^{n-k}b^k$ aparece $C(n, k)$ veces. Se concluye que

$$\begin{aligned}(a + b)^n = {} & C(n, 0)a^nb^0 + C(n, 1)a^{n-1}b^1 + C(n, 2)a^{n-2}b^2\\ & + \cdots + C(n, n - 1)a^1b^{n-1} + C(n, n)a^0b^n.\end{aligned} \tag{6.7.2}$$

Este resultado se conoce como el *teorema binomial*.

Teorema 6.7.1

Teorema binomial

Si a y b son números reales y n es un entero positivo, entonces

$$(a + b)^n = \sum_{k=0}^{n} C(n, k)a^{n-k}b^k.$$

Demostración La demostración precede al enunciado del teorema.

El teorema binomial también se demuestra usando inducción sobre n (vea el ejercicios 16).

Los números $C(n, r)$ se conocen como **coeficientes binomiales** porque aparecen en la expansión (6.7.2) del binomio $a + b$ elevado a una potencia.

Ejemplo 6.7.2 ▶

Tomando $n = 3$ en el Teorema 6.7.1, se obtiene

$$\begin{aligned}(a + b)^3 &= C(3, 0)a^3b^0 + C(3, 1)a^2b^1 + C(3, 2)a^1b^2 + C(3, 3)a^0b^3\\ &= a^3 + 3a^2b + 3ab^2 + b^3.\end{aligned}$$

◀

Ejemplo 6.7.3 ▶

Obtenga la expansión de $(3x - 2y)^4$ usando el teorema del binomio.

Si se toma $a = 3x$, $b = -2y$ y $n = 4$ en el Teorema 6.7.1, se obtiene

$$\begin{aligned}(3x-2y)^4 &= (a+b)^4\\ &= C(4,0)a^4b^0 + C(4,1)a^3b^1 + C(4,2)a^2b^2\\ &\quad + C(4,3)a^1b^3 + C(4,4)a^0b^4\\ &= C(4,0)(3x)^4(-2y)^0 + C(4,1)(3x)^3(-2y)^1\\ &\quad + C(4,2)(3x)^2(-2y)^2 + C(4,3)(3x)^1(-2y)^3\\ &\quad + C(4,4)(3x)^0(-2y)^4\\ &= 3^4x^4 + 4\cdot 3^3x^3(-2y) + 6\cdot 3^2x^2(-2)^2y^2\\ &\quad + 4(3x)(-2)^3y^3 + (-2)^4y^4\\ &= 81x^4 - 216x^3y + 216x^2y^2 - 96xy^3 + 16y^4.\end{aligned}$$

◀

Ejemplo 6.7.4 ▶

Encuentre el coeficiente de a^5b^4 en la expansión de $(a + b)^9$.

El término que implica a a^5b^4 surge en el teorema del binomio al tomar $n = 9$ y $k = 4$:

$$C(n,k)a^{n-k}b^k = C(9,4)a^5b^4 = 126a^5b^4.$$

Entonces, el coeficiente de a^5b^4 es 126. ◀

Ejemplo 6.7.5 ▶

Encuentre el coeficiente de $x^2y^3z^4$ en la expansión de $(x + y + z)^9$.

Como

$$(x+y+z)^9 = (x+y+z)(x+y+z)\cdots(x+y+z) \quad \text{(nueve términos)},$$

se obtiene $x^2y^3z^4$ cada vez que se multiplican las x seleccionadas en 2 de los 9 términos, las y seleccionadas en 3 de los 9 términos y las z seleccionadas en 4 de los 9 términos. Se pueden elegir dos términos para las x de $C(9, 2)$ maneras. Una vez hecha esta selección, se pueden elegir tres términos para las y de $C(7,3)$ maneras. Esto deja los cuatro términos restantes para las z. Entonces, el coeficiente de $x^2y^3z^4$ en la expansión de $(x + y + z)^9$ es

$$C(9,2)C(7,3) = \frac{9!}{2!\,7!}\frac{7!}{3!\,4!} = \frac{9!}{2!\,3!\,4!} = 1260.$$

◀

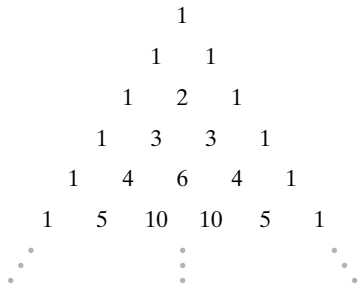


Figura 6.7.1 Triángulo de Pascal.

WWW

Es posible escribir los coeficientes binomiales en un esquema triangular llamado **triángulo de Pascal** (vea la figura 6.7.1). El contorno está formado por unos, y cualquier valor interior es la suma de los números arriba de él. Esta relación se establece formalmente en el siguiente teorema. La demostración es un argumento combinatorio. Una identidad que se obtiene de un proceso de conteo se llama **identidad combinatoria** y el argumento que lleva a su formulación se llama **argumento combinatorio**.

Teorema 6.7.6

$$C(n+1,k) = C(n,k-1) + C(n,k)$$

para $1 \le k \le n$.

Demostración Sea X un conjunto de n elementos. Se elige $a \notin X$. Entonces $C(n + 1, k)$ es el número de subconjuntos de k elementos de $Y - X \cup \{a\}$. Ahora los subconjuntos de k elementos de Y se pueden dividir en dos clases ajenas:

1. Subconjuntos de Y que no contienen a a.
2. Subconjuntos de Y que contienen a a.

Los subconjuntos de la clase 1 son sólo subconjuntos de k elementos de X y existen $C(n, k)$ de ellos. Cada subconjunto de la clase 2 consiste en un subconjunto de $(k-1)$ elementos de X junto con a y existen $C(n, k-1)$ de ellos. Por lo tanto,

$$C(n+1, k) = C(n, k-1) + C(n, k)$$

El Teorema 6.7.6 también se demuestra mediante el Teorema 6.2.17 (ejercicios 17 de esta sección).

La sección finaliza mostrando cómo emplear el teorema binomial (Teorema 6.7.1) y el Teorema 6.7.6 para derivar otras identidades combinatorias.

Ejemplo 6.7.7 ▶

Use el teorema binomial para derivar la ecuación

$$\sum_{k=0}^{n} C(n, k) = 2^n. \tag{6.7.3}$$

La suma es la misma que la suma en el teorema binomial,

$$\sum_{k=0}^{n} C(n, k) a^{n-k} b^k,$$

excepto que falta la expresión $a^{n-k}b^k$. Una manera de "eliminar" esta expresión es hacer $a = b = 1$, en cuyo caso el teorema binomial se convierte en

$$2^n = (1+1)^n = \sum_{k=0}^{n} C(n, k) 1^{n-k} 1^k = \sum_{k=0}^{n} C(n, k). \quad \blacktriangleleft$$

También es posible probar la ecuación (6.7.3) mediante un argumento combinatorio. Dado un conjunto X de n elementos, $C(n, k)$ cuenta el número de subconjuntos de k elementos. Así, el lado derecho de la ecuación (6.7.3) cuenta el número de subconjuntos de X. Pero el número de subconjuntos de X es 2^n; se ha probado de nuevo (6.7.3).

Ejemplo 6.7.8 ▶

Use el Teorema 6.7.6 para demostrar que

$$\sum_{i=k}^{n} C(i, k) = C(n+1, k+1). \tag{6.7.4}$$

Se usa el Teorema 6.7.6 en la forma

$$C(i, k) = C(i+1, k+1) - C(i, k+1)$$

para obtener

$$\begin{aligned} &C(k, k) + C(k+1, k) + C(k+2, k) + \cdots + C(n, k) \\ &\quad = 1 + C(k+2, k+1) - C(k+1, k+1) + C(k+3, k+1) \\ &\qquad - C(k+2, k+1) + \cdots + C(n+1, k+1) - C(n, k+1) \\ &\quad = C(n+1, k+1). \end{aligned}$$

$\blacktriangleleft$

El ejercicio 47 de la sección 6.6 presenta otra manera de demostrar la ecuación (6.7.4).

Ejemplo 6.7.9 ▶

Use la ecuación (6.7.4) para encontrar la suma

$$1 + 2 + \ldots + n.$$

Se escribe

$$\begin{aligned} 1 + 2 + \cdots + n &= C(1, 1) + C(2, 1) + \cdots + C(n, 1) \\ &= C(n + 1, 2) \qquad \text{la ecuación (6.7.4)} \\ &= \frac{(n + 1)n}{2}. \end{aligned}$$

◀

Sección de ejercicios de repaso

1. Enuncie el teorema binomial.
2. Explique cómo se deriva el teorema binomial.
3. ¿Qué es el triángulo de Pascal?
4. Establezca las fórmula que se utilizan para generar el triángulo de Pascal.

Ejercicios

1. Expanda $(x + 4)^4$ usando el teorema binomial
2. Expanda $(2c - 3d)^5$ usando el teorema binomial.

En los ejercicios 3 al 9, encuentre el coeficiente del término cuando la expresión se expande.

3. $x^4y^7; (x + y)^{11}$
4. $s^6t^6; (2s - t)^{12}$
5. $x^2y^3z^5; (x + y + z)^{10}$
6. $w^2x^3y^2z^5; (2w + x + 3y + z)^{12}$
7. $a^2x^3; (a + x + c)^2(a + x + d)^3$
8. $a^2x^3; (a + ax + x)(a + x)^4$
9. $a^3x^4; (a + \sqrt{ax} + x)^2(a + x)^5$

En los ejercicios 10 al 12, encuentre el número de términos de la expansión de cada expresión.

10. $(x + y + z)^{10}$
11. $(w + x + y + z)^{12}$
★12. $(x + y + z)^{10}(w + x + y + z)^2$
13. Encuentre el siguiente renglón del triángulo de Pascal a partir del renglón

$$1 \quad 7 \quad 21 \quad 35 \quad 35 \quad 21 \quad 7 \quad 1.$$

14. *a)* Demuestre que $C(n, k) < C(n, k + 1)$ si y sólo si $k < (n - 1)/2$.
    b) Use el inciso *a)* para deducir que el máximo de $C(n, k)$ para $k = 0, 1, \ldots, n$ es $C(n, \lfloor n/2 \rfloor)$.
15. Use el teorema binomial para demostrar que

$$0 = \sum_{k=0}^{n} (-1)^k C(n, k).$$

16. Use inducción sobre n para probar el teorema binomial.
17. Pruebe el teorema binomial 6.7.6 usando el teorema 6.2.17.
18. Dé un argumento combinatorio para demostrar que

$$C(n, k) = C(n, n - k).$$

★19. Demuestre la ecuación (6.7.4) mediante un argumento combinatorio.
20. Encuentre la suma

$$1 \cdot 2 + 2 \cdot 3 + \ldots + (n - 1)n.$$

★21. Use la ecuación (6.7.4) para derivar una fórmula para

$$1^2 + 2^2 + \cdots + n^2.$$

22. Use el teorema binomial para demostrar que

$$\sum_{k=0}^{n} 2^k C(n, k) = 3^n.$$

23. Suponga que n es par. Pruebe que

$$\sum_{k=0}^{n/2} C(n, 2k) = 2^{n-1} = \sum_{k=1}^{n/2} C(n, 2k - 1).$$

24. Pruebe

$$(a + b + c)^n = \sum_{0 \le i+j \le n} \frac{n!}{i!\, j!\, (n - i - j)!} a^i b^j c^{n-i-j}.$$

25. Use el ejercicio 24 para escribir la expansión de $(x + y + z)^3$.
26. Pruebe

$$3^n = \sum_{0 \le i+j \le n} \frac{n!}{i!\, j!\, (n - i - j)!}.$$

★27. Dé un argumento combinatorio para probar que

$$\sum_{k=0}^{n} C(n, k)^2 = C(2n, n).$$

28. Pruebe

$$n(1 + x)^{n-1} = \sum_{k=1}^{n} C(n, k) k x^{k-1}.$$

29. Use el resultado del ejercicio 28 para demostrar que

$$n2^{n-1} = \sum_{k=1}^{n} kC(n, k). \qquad \textbf{(6.7.5)}$$

★30. Pruebe la ecuación (6.7.5) por inducción.
31. Una *sucesión de suavizado* $b_0, \ldots, b_{k-1}$ es una sucesión (finita) que satisface $b_i \ge 0$ para $i = 0, \ldots, k - 1$ y $\sum_{i=0}^{k-1} b_i = 1$. Un *suavizado de la sucesión (infinita)* $a_1, a_2, \ldots$ *por la sucesión de suavizado* $b_0, \ldots, b_{k-1}$ es la sucesión $\{a_j'\}$ definida por

$$a_j' = \sum_{i=0}^{k-1} a_{i+j} b_i.$$

La idea es que al promediar se suaviza el ruido en los datos.

El *suavizador binomial de tamaño k* es la sucesión

$$\frac{B_0}{2^n}, \ldots, \frac{B_{k-1}}{2^n},$$

donde $B_0, \ldots, B_{k-1}$ es el renglón n del triángulo de Pascal (el renglón 0 es el renglón superior).

Sea c_0, c_1 la sucesión de suavizado definida por $c_0 = c_1 = 1/2$. Demuestre que si c suaviza a una sucesión a, c suaviza a la sucesión que se obtiene, y así sucesivamente k veces; entonces la sucesión que resulta se obtiene mediante un suavizado de a por el suavizador binomial de tamaño $k + 1$.

32. En el ejemplo 6.1.6 se demostró que existen 3^n pares ordenados (A, B) que satisfacen $A \subseteq B \subseteq X$, donde X es un conjunto de n elementos. Derive este resultado considerando los casos $|A| = 0$, $|A| = 1, \ldots, |A| = n$, y después usando el teorema binomial.

33. Demuestre que

$$\sum_{k=m}^{n} C(k, m)H_k = C(n + 1, m + 1)\left(H_{n+1} - \frac{1}{m + 1}\right)$$

para toda $n \geq m$, donde H_k, el k-ésimo número armónico, está definido como

$$H_k = \sum_{i=1}^{k} \frac{1}{i}.$$

6.8 ➜ El principio del palomar

WWW

El **principio del palomar** (también conocido como el *principio de la pichonera, principio de la cajonera de Dirichlet* o el *principio de la caja de zapatos*) suele ser útil al responder la pregunta: ¿Hay un elemento que tiene una propiedad dada? Cuando se aplica con éxito el principio del palomar, sólo indica que existe el objeto; el principio no dice cómo encontrar el objeto ni cuántos objetos hay.

La primera versión del principio del palomar que se estudiará asegura que si n palomas vuelan y entran a k palomares y $k < n$, algunos palomares contendrán al menos dos palomas (figura 6.8.1). La razón por la que esta afirmación es cierta se aprecia mediante un argumento por contradicción. Si la conclusión es falsa, cada palomar contiene cuando mucho una paloma y, en este caso, se puede rendir cuenta de cuando mucho k palomas. Como hay n palomas y $n > k$, se tiene una contradicción.

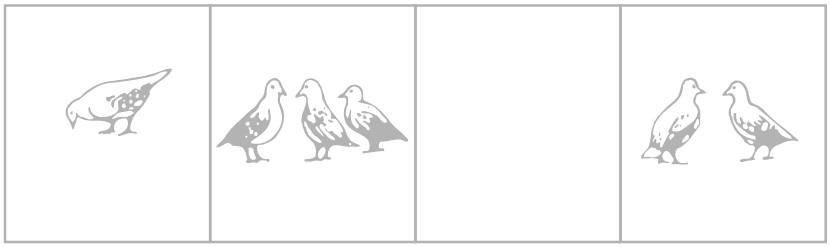


Figura 6.8.1 $n = 6$ palomas en $k = 4$ palomares. Algún palomar contiene al menos dos palomas.

Principio del palomar (primera forma)

Si n palomas vuelan a los palomares y $k < n$, algunos palomares contienen al menos dos palomas.

Se observa que el principio del palomar no establece cómo localizar el hoyo que contiene dos palomas o más. Sólo asegura la *existencia* de un hoyo con dos palomas o más.

Para aplicar el principio del palomar, debemos decidir qué objetos tendrán el papel de palomas y qué objetos tendrán el papel de palomares. El primer ejemplo ilustra una posibilidad.

Ejemplo 6.8.1 ▶

Diez personas tienen nombres de pila Alicia, Bernardo y Carlos y apellidos López, Maza y Noriega. Demuestre que al menos dos personas tienen el mismo nombre y apellido.

Existen nueve nombres posibles para las 10 personas. Si se piensa en las personas como palomas y en los nombres como los palomares, se puede considerar que la asignación de nombres a personas es lo mismo que asignar palomares a las palomas. Por el principio del palomar, algún nombre (palomar) se asigna al menos a dos personas (palomas). ◀

Se enunciará de otra manera el principio del palomar.

Principio del palomar (segunda forma)

Si f es una función de un conjunto finito X a un conjunto finito Y y $|X| > |Y|$*, entonces* $f(x_1) = f(x_2)$ para alguna $x_1, x_2 \in X$, $x_1 \neq x_2$.

La segunda forma del principio del palomar se reduce a la primera forma si X se define como el conjunto de palomas y Y el conjunto de palomares. Se asigna la paloma x al hoyo $f(x)$. Por la primera forma del principio del palomar, al menos dos palomas, $x_1, x_2 \in X$, se asignan al mismo palomar; es decir, $f(x_1) = f(x_2)$ para alguna $x_1, x_2 \in X$, $x_1 \neq x_2$.

Los siguientes ejemplos ilustran la aplicación de la segunda forma del principio del palomar.

Ejemplo 6.8.2 ▶

Si 20 procesadores están interconectados, demuestre que al menos dos de ellos tienen conexión directa al mismo número de procesadores.

Denote los procesadores como 1, 2, . . . , 20. Sea a_i el número de procesadores a los que el procesador i está conectado directamente. Debe demostrarse que $a_i = a_j$ para alguna $i \neq j$. El dominio de la función a es $X = \{1,2, \ldots, 20\}$ y el recorrido o imagen Y es un subconjunto de $\{0,1, \ldots, 19\}$. Por desgracia, $|X| = \{0,1, \ldots, 19\}$ y no es posible usar de inmediato la segunda forma del principio del palomar.

Se examinará la situación con más detalle. Observe que no se puede tener $a_i = 0$, para alguna i, y $a_j = 19$ para alguna j, porque se tendría un procesador (el i-ésimo procesador) sin conectar a otro procesador y, al mismo tiempo, otro procesador (el j-ésimo) conectado a todos los otros procesadores (incluso el i-ésimo). Entonces, el recorrido Y es un subconjunto ya sea de $\{0,1, \ldots, 18\}$ o bien de $\{1,2, \ldots, 19\}$. En cualquier caso, $|Y| < 20 = |X|$. Por la segunda forma del principio del palomar, $a_i = a_j$ para alguna $i \neq j$, como se desea. ◀

Ejemplo 6.8.3 ▶

Demuestre que si se seleccionan 151 cursos diferentes de computación numerados entre 1 y 300 inclusive, al menos dos tienen números consecutivos.

Sean los números seleccionados

$$c_1, \quad c_2, \quad \ldots, \quad c_{151}. \tag{6.8.1}$$

Los 302 números que consisten en los números (6.8.1) junto con

$$c_1 + 1, \quad c_2 + 1, \quad \ldots, \quad c_{151} + 1 \tag{6.8.2}$$

tienen valores que van del 1 al 301. Por la segunda forma del principio del palomar, al menos dos de estos valores coinciden. Los números (6.8.1) son todos diferentes y, por lo mismo, los números (6.8.2) también son distintos. Entonces debe ocurrir que uno en (6.8.1) y uno en (6.8.2) son iguales. Así, se tiene

$$c_i = c_j + 1$$

y el curso c_i sigue al curso c_j. ◀

Ejemplo 6.8.4 ▶

Un inventario consiste en una lista de 80 artículos, cada uno marcado "disponible" o "no disponible". Hay 45 artículos disponibles. Demuestre que hay al menos dos artículos disponibles en la lista que están separados por exactamente 9 artículos. (Por ejemplo, los artículos disponibles en las posiciones 13 y 22 o en las posiciones 69 y 78 satisfacen esta condición).

Sea a_i la posición del i-ésimo artículo disponible. Debe probarse que $a_i - a_j = 9$ para alguna i y j. Considere los números

$$a_1, \quad a_2, \quad \ldots, \quad a_{45} \tag{6.8.3}$$

y

$$a_1 + 9, \quad a_2 + 9, \quad \ldots, \quad a_{45} + 9. \tag{6.8.4}$$

Los 90 números en (6.8.3) y (6.8.4) tienen valores posibles sólo del 1 al 89. Por la segunda forma del principio del palomar, dos de los números deben coincidir. No es posible tener dos de (6.8.3) o dos de (6.8.4) idénticos; entonces, algún número en (6.8.3) es igual a algún número en (6.8.4). Por lo tanto, $a_i - a_j = 9$ para alguna i y j, como se quería. ◀

Ahora se establece el principio del palomar en otra forma más.

Principio del palomar (tercera forma)

Sea f una función de un conjunto finito X a un conjunto finito Y. Suponga que $|X| = n$ y $|Y| = m$. Sea $k = \lfloor n/m \rfloor$. Entonces hay al menos k valores $a_1, \ldots, a_k \in X$ tal que

$$f(a_1) = f(a_2) = \cdots = f(a_k).$$

Para probar la tercer forma del principio del palomar, se da un argumento por contradicción. Sea $Y = \{y_1, \ldots, y_m\}$. Suponga que la conclusión es falsa. Entonces hay cuando mucho $k - 1$ valores de $x \in X$ con $f(x) = y_1$; hay cuando mucho $k - 1$ valores de $x \in X$ con $f(x) = y_2$; . . . ; hay cuando mucho $k - 1$ valores x con $f(x) = y_m$. Entonces hay cuando mucho $m(k - 1)$ miembros en el dominio de f. Pero

$$m(k-1) < m\frac{n}{m} = n,$$

que es una contradicción. Por lo tanto, hay al menos k valores, $a_1, \ldots, a_k \in X$, tales que

$$f(a_1) = f(a_2) = \cdots = f(a_k).$$

El último ejemplo ilustra el uso de la tercera forma del principio del palomar.

Ejemplo 6.8.5 ▶

Una característica útil de las fotografías en blanco y negro es el brillo promedio de la foto. Digamos que dos fotos son similares si su brillo promedio difiere en no más de un valor fijo. Demuestre que entre seis fotografías, hay tres que son mutuamente similares o bien tres que son mutuamente no similares.

Denote las fotografías por $P_1, P_2, \ldots, P_6$. Cada uno de los cinco pares

$$(P_1, P_2), \quad (P_1, P_3), \quad (P_1, P_4), \quad (P_1, P_5), \quad (P_1, P_6),$$

tiene el valor "similar" o "no similar". Por la tercera forma del principio del palomar, hay al menos $\lceil 5/2 \rceil = 3$ pares con el mismo valor; es decir, hay tres pares

$$(P_1, P_i), \quad (P_1, P_j), \quad (P_1, P_k)$$

todos similares o no similares. Suponga que cada par es similar. (El caso en que cada par es no similar se ve en el ejercicio 8). Si cualquier par

$$(P_i, P_j), \quad (P_i, P_k), \quad (P_j, P_k) \qquad \textbf{(6.7.5)}$$

es similar, entonces estas dos fotografías junto con P_1 son mutuamente similares y se tienen tres fotografías mutuamente similares. De otra manera, cada uno de lo pares (6.8.5) es no similar y se tienen tres pares de fotografías mutuamente no similares. ◀

Sección de ejercicios de repaso

1. Enuncie las tres formas del principio del palomar.

2. Dé un ejemplo del uso de cada forma del principio del palomar.

Ejercicios

1. Trece personas tienen nombres de pila Dora, Evita y Fernando, y apellidos Olmos, Pérez, Quintana y Rodríguez. Demuestre que al menos dos personas tienen el mismo nombre y apellido.

2. Dieciocho personas tienen nombres de pila Alfredo, Benjamín y César y apellidos Domínguez y Enríquez. Demuestre que al menos tres personas tienen el mismo nombre y apellido.

3. La profesora Eugenia recibe su salario todos los viernes. Demuestre que en algunos meses le pagan tres veces.

4. ¿Es posible interconectar cinco procesadores de manera que exactamente dos de ellos tengan conexión directa a un número idéntico de procesadores? Explique su respuesta.

5. Un inventario consiste en una lista de 115 artículos, cada uno marcado como "disponible" o "no disponible". Hay 60 artículos disponibles. Demuestre que hay al menos 2 artículos disponibles en la lista que están separados por exactamente 4 artículos.

6. Un inventario consiste en una lista de 100 artículos, cada uno marcado como "disponible" o "no disponible". Hay 55 artículos disponibles. Demuestre que hay al menos dos artículos disponibles en la lista que están separados por exactamente 9 artículos.

★ ***7.*** Un inventario consiste en una lista de 80 artículos, cada uno marcado como "disponible" o "no disponible". Hay 50 artículos disponibles. Demuestre que hay al menos 2 artículos no disponibles en la lista que están separados por 3 o por 6 artículos.

8. Complete el ejemplo 6.8.5 demostrando que si los pares (P_1, P_i), (P_1, P_j), (P_1, P_k) son no similares, hay tres fotografías que son mutuamente similares o mutuamente no similares.

9. ¿Se puede afirmar necesariamente la conclusión del ejemplo 6.8.5 si hay menos de 6 fotografías? Explique por qué.

10. ¿Se puede afirmar necesariamente la conclusión del ejemplo 6.8.5 si hay más de 6 fotografías? Explique por qué.

En los ejercicios 11 al 14 dé un argumento que muestre que si X es cualquier subconjunto de $(n + 2)$ elementos del conjunto $\{1, 2, \ldots, 2n + 1\}$ y m es el elemento más grande en X, existen i y j distintos en X con $m = i + j$.

Para cada elemento $k \in X - \{m\}$, sea

$$a_k = \begin{cases} k & \text{if } k \le \dfrac{m}{2} \\ m - k & \text{if } k > \dfrac{m}{2}. \end{cases}$$

11. ¿Cuántos elementos hay en el dominio de a?

12. Demuestre que el recorrido de a está contenido en $\{1, 2, \ldots, n\}$.

13. Explique por qué los ejercicios 11 y 12 implican que $a_i = a_j$ para alguna $i \neq j$.

14. Explique por qué el ejercicio 13 implica que existen i y j distintos en X con $m = i + j$.

15. Dé un ejemplo de un subconjunto X de $(n + 1)$ elementos del conjunto $\{1, 2, \ldots, 2n + 1\}$ que tenga la propiedad: no hay dos elementos distintos $i, j \in X$ para los que $i + j \in X$.

En los ejercicios 16 al 19 dé un argumento que pruebe el siguiente resultado:

Una sucesión $a_1, a_2, \ldots, a_{n^2+1}$ de $n^2 + 1$ números diferentes contiene ya sea una subsucesión creciente de longitud $n + 1$ o bien una subsucesión decreciente de longitud $n + 1$.

Suponga, a manera de contradicción, que toda subsucesión creciente o decreciente tiene longitud n o menor. Sea b_i la longitud de la subsucesión creciente más larga que comienza en a_i, y sea c_i la longitud de la subsucesión decreciente más larga que comienza en a_i.

16. Demuestre que los pares ordenados (b_i, c_i), $i = 1, \ldots, n^2 + 1$, son distintos.

17. ¿Cuántos pares ordenados (b_i, c_i) hay?

18. Explique por qué $1 \le b_i \le n$ y $1 \le c_i \le n$.

19. ¿Cuál es la contradicción?

En los ejercicios 20 al 23 dé un argumento que demuestre que en un grupo de 10 personas hay al menos dos tales que la diferencia o la suma de sus edades es divisible entre 16. Suponga que las edades están dadas en números enteros.

Sean $a_1, \ldots, a_{10}$ las edades. Sea $r_i = a_i \bmod 16$ y sea

$$s_i = \begin{cases} r_i & \text{if } r_i \le 8 \\ 16 - r_i & \text{if } r_i > 8. \end{cases}$$

20. Demuestre que $s_1, \ldots, s_{10}$ tienen valores que van de 0 a 8.

21. Explique por qué $s_j = s_k$ para alguna $j \neq k$.

22. Suponga que $s_j = s_k$ para alguna $j \neq k$. Explique por qué si $s_j = r_j$ y $s_k = r_k$ o $s_j = 16 - r_j$ y $s_k = 16 - r_k$, entonces 16 divide a $a_j - a_k$.

23. Demuestre que si las condiciones en el ejercicio 22 no se cumplen, entonces 16 divide a $a_j + a_k$.

24. Demuestre que en la expansión decimal del cociente de dos enteros, en algún momento un bloque de dígitos se repite. *Ejemplos:*

$$\frac{1}{6} = 0.1\underline{6}66\ldots, \qquad \frac{217}{660} = 0.32\underline{878}787\ldots.$$

★**25.** Doce jugadores de básquetbol, cuyos uniformes están numerados del 1 al 12, están colocados alrededor del cuadro central de la cancha con un arreglo arbitrario. Demuestre que para algunos tres jugadores consecutivos, la suma de sus números es al menos 20.

★**26.** Para la situación del ejercicio 25, encuentre y pruebe una estimación de qué tan grande debe ser la suma de los números para algunos cuatro jugadores consecutivos.

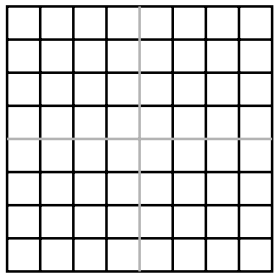
★**27.** Sea f una función uno a uno de $X = \{1, 2, \ldots, n\}$ sobre X. Sea $f^k = f \circ f \circ \cdots \circ f$ la composición de f, k veces consigo misma. Demuestre que existen enteros positivos diferentes i y j tales que $f^i(x) = f^j(x)$ para toda $x \in X$. Demuestre que para algún entero positivo k, $f^k(x) = x$ para toda $x \in X$.

★***28.*** Un rectángulo de 3×7 se divide en 21 cuadrados cada uno coloreado rojo o negro. Pruebe que el tablero contiene un rectángulo no trivial (no $1 \times k$ o $k \times 1$) cuyos cuatro cuadrados en las esquinas son todos negros o todos rojos.

★**29.** Pruebe que si p unos y q ceros se colocan alrededor de un círculo de manera arbitraria, donde p, q y k son enteros positivos que satisfacen $p \ge kq$, el arreglo debe contener al menos k unos consecutivos.

★**30.** Escriba un algoritmo que, dada una sucesión a, encuentre la longitud de la subsucesión creciente de a más larga.

31. Una rejilla de $2k \times 2k$ se divide en $4k^2$ cuadrados y cuatro subrejillas de $k \times k$. La figura siguiente muestra la rejilla para $k = 4$:



Demuestre que es imposible marcar k cuadrados en la subrejilla de $k \times k$ superior izquierda, y k cuadrados en la subrejilla de $k \times k$ inferior derecha, de manera que no haya dos cuadros marcados en el mismo renglón, columna o diagonal de la rejilla de $2k \times 2k$.

Ésta es una variante del *problema de las n reinas* que se analiza con detalle en la sección 9.3.

Notas

Un libro elemental referente a métodos de conteo es [Niven, 1965]. Las referencias sobre combinatoria son [Brualdi; Even, 1973; Liu, 1968; Riordan; y Roberts]. [Vilenkin] contiene muchos ejemplos resueltos de combinatoria. Las referencias generales de matemáticas discretas [Liu, 1985; y Tucker] dedican varias secciones a los temas del capítulo 6. [Even, 1973; Hu, y Reingold] estudian los algoritmos de combinatoria. Las referencias de probabilidad son [Billingsley; Ghahramani; Kelly; Ross; y Rozanov]. [Fukunaga; Gose; y Nader son libros acerca de reconocimiento de patrones.

Repaso del capítulo

Sección 6.1

1. Principio de la multiplicación
2. Principio de la suma

Sección 6.2

3. Permutaciones de $x_1, \ldots, x_n$; ordenamiento de $x_1, \ldots, x_n$
4. $n!$ = número de permutaciones de un conjunto de n elementos
5. Permutaciones r de $x_1, \ldots, x_n$; ordenamiento de r elementos de $x_1, \ldots, x_n$
6. $P(n, r)$: número de permutaciones r de un conjunto de n elementos;

$$P(n, r) = n(n-1)\cdots(n-r+1)$$

7. Combinación r de $\{x_1, \ldots, x_n\}$: subconjunto (no ordenado) de $\{x_1, \ldots, x_n\}$ que contiene r elementos
8. $C(n, r)$: número de combinaciones r de un conjunto de n elementos; $C(n, r) = P(n, r)/r! = n!/[(n-r)!\,r!]$

Sección 6.3

9. Orden lexicográfico
10. Algoritmo para generar combinaciones r: algoritmo 6.3.9
11. Algoritmo para generar permutaciones: algoritmo 6.3.14

Sección 6.4

12. Experimento
13. Evento
14. Espacio muestra
15. Probabilidad de un evento cuando todos los resultados son igualmente probables

Sección 6.5

16. Función de probabilidad
17. Probabilidad de un evento
18. Si E es un evento, $P(E) + P(\overline{E}) = 1$
19. Si E_1 y E_2 sin eventos $P(E_1 \cup E_2) = P(E_1) + P(E_2) - P(E_1 \cap E_2)$
20. Los eventos E_1 y E_2 son mutuamente excluyentes si $E_1 \cap E_2 = \emptyset$
21. Si los eventos E_1 y E_2 son mutuamente excluyentes, $P(E_1 \cup E_2) = P(E_1) + P(E_2)$
22. Si E y F son eventos y $P(F) > 0$, la probabilidad condicional de E dado F es $P(E \mid F) = P(E \cap F)/P(F)$
23. Los eventos E y F son independientes si $P(E \cap F) = P(E)P(F)$
24. Teorema de Bayes: Si las clases posibles son $C_1, \ldots, C_n$, cada par de estas clases es mutuamente excluyente y cada elemento que se va a clasificar pertenece a una de estas clases, para un conjunto de características F se tiene

$$P(C_j \mid F) = \frac{P(F \mid C_j)}{\sum_{i=1}^{n} P(F \mid C_i)P(C_i)}.$$

Sección 6.6

25. Número de ordenamientos de n artículos de t tipos con n_1 objetos idénticos de tipo $i = n!/[n_1! \cdots n_t!]$

26. Número de selecciones no ordenadas de k elementos de un conjunto de t elementos, con repeticiones permitidas $= C(k + t - 1, k)$

Sección 6.7

27. Teorema binomial: $(a + b)^n = \sum_{k=0}^{n} C(n, k)a^{n-k}b^k$

28. Triángulo de Pascal: $C(n + 1, k) = C(n, k - 1) + C(n, k)$

Sección 6.8

29. Principio del palomar (tres formas)

Autoevaluación del capítulo

Sección 6.1

1. ¿Cuántas cadenas de 8 bits comienzan con 0 y terminan con 101?

2. ¿De cuántas maneras se pueden seleccionar 3 libros cada uno de un tema diferente, entre un conjunto de 6 libros de historia, 9 libros clásicos, 7 libros de leyes y 4 libros de educación, todos distintos?

3. ¿Cuántas funciones hay de un conjunto de n elementos sobre $\{0, 1\}$?

4. Un comité de 7 personas compuesto por Gregorio, Humberto, Isaac, Jazmín, Karen, Laura y Manuel debe seleccionar presidente, vicepresidente, coordinador de eventos sociales, secretario y tesorero. ¿De cuántas maneras pueden asignarse los puestos si Gregorio es secretario o no le asignan un puesto?

Sección 6.2

5. ¿Cuántas combinaciones de 3 hay de 6 objetos?

6. ¿Cuántas cadenas se pueden formar ordenando las letras *ABCDEF* si *A* aparece antes de *C* y *E* aparece antes de *C*?

7. ¿Cuántas manos de 6 cartas seleccionadas de una baraja común de 52 cartas contienen 3 cartas de un palo y 3 cartas de otro palo?

8. Un envío de 100 discos compactos contiene 5 defectuosos. De cuántas maneras se puede seleccionar un conjunto de 4 discos compactos que contenga más discos defectuosos que buenos?

Sección 6.3

9. Encuentre las combinaciones de 5 que genera el algoritmo 6.3.9 después de 12467 si $n = 7$.

10. Encuentre las combinaciones de 6 que genera el algoritmo 6.3.9 después de 145678 si $n = 8$.

11. Encuentre la permutación que genera el algoritmo 6.3.14 después de 6427135.

12. Encuentre la permutación que genera el algoritmo 6.3.14 después de 625431.

Sección 6.4

13. Se selecciona una carta al azar de una baraja común de 52 cartas. ¿Cuál es la probabilidad de que sea un corazón?

14. Se lanzan dos dados no cargados. ¿Cuál es la probabilidad de que la suma de los números que aparecen sea 8?

15. En el juego "Cash In Hand" de Maryland, el concursante elige 7 números distintos entre 1 y 31. Él gana una cantidad modesta ($40) si exactamente 5 números, en cualquier orden, coinciden con 5 de los 7 elegidos aleatoriamente por un representante de la lotería. ¿Cuál es la probabilidad de ganar $40?

16. Encuentre la probabilidad de obtener una mano de bridge con distribución 6−5−2−0, es decir, 6 cartas de un palo, 5 de otro palo, 2 de otro y ninguna carta del cuarto palo.

Sección 6.5

17. Se carga una moneda de manera que es cinco veces más probable que salga cara que cruz. Asigne probabilidades a los eventos que modelen con exactitud las posibilidades de que ocurran.

18. Una familia tiene 3 hijos. Suponga que es igualmente probable que nazca una niña o un niño. ¿Son independientes los eventos "hay niños de uno y otro sexo" y "cuando mucho hay una niña"? Explique su respuesta.

19. Jorge y Alicia presentan un examen final de C++. La probabilidad de que Jorge pase es de 0.75 y la probabilidad de que Alicia pase es de 0.80. Suponga que los eventos "Jorge pasa el examen final" y "Alicia pasa el examen final" son independientes. Encuentre la probabilidad de que Jorge no pase. Encuentre la probabilidad de que ambos pasen. Encuentre la probabilidad de que ambos reprueben. Encuentre la probabilidad de que al menos uno pase.

20. Tulio, Roberto y José escriben programas para establecer horarios para las tareas de fabricación de perros de juguete. La siguiente tabla muestra el porcentaje de código escrito por cada uno y el porcentaje de fallas de cada persona.

	Codificador		
	Tulio	*Roberto*	*José*
Porcentaje de código	30	45	25
Porcentaje de fallas	3	2	5

Dado que se encontró una falla, encuentre la probabilidad de que estuviera en el código de José.

Sección 6.6

21. ¿Cuántas cadenas se pueden formar ordenando las letras ILLINOIS?

22. ¿Cuántas cadenas se pueden formar ordenando las letras ILLINOIS si la *I* aparece antes que una *L*?

23. ¿De cuántas maneras pueden dividirse 12 libros diferentes entre cuatro estudiantes si cada estudiante obtiene tres libros?

24. ¿Cuántas soluciones enteras de

$$x_1 + x_2 + x_3 + x_4 = 17$$

satisfacen $x_1 \geq 0, x_2 \geq 1, x_3 \geq 2, x_4 \geq 3$?

Sección 6.7

25. Expanda la expresión $(s - r)^4$ usando el teorema binomial.

26. Encuentre el coeficiente de x^3yz^4 en la expansión de $(2x + y + z)^8$.

27. Utilice el teorema binomial para probar que

$$\sum_{k=0}^{n} 2^{n-k}(-1)^k C(n, k) = 1.$$

28. Rote el triángulo de Pascal al contrario de las manecillas del reloj de manera que en el primer renglón haya unos. Explique por qué el segundo renglón lista los enteros positivos en orden 1, 2, ...

Sección 6.8

29. Demuestre que todo conjunto de 15 calcetines elegidos entre 14 pares de calcetines contiene al menos un par.

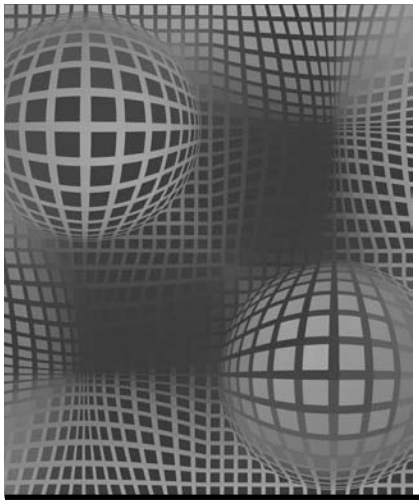
30. Diecinueve personas tienen nombres de pila Zeke, Wally y Linda; segundos nombres Leo y David, y apellidos Yu, Zamora y Smith. Demuestre que al menos dos personas tienen el mismo primer nombre, segundo nombre y apellido.

31. Un inventario consiste en una lista de 200 artículos, cada uno marcado como "disponible" o "no disponible". Existen 110 artículos disponibles. Demuestre que hay al menos dos artículos disponibles en la lista que están separados exactamente por 19 artículos.

32. Sea $P = \{p_1, p_2, p_3, p_4, p_5\}$ un conjunto de cinco puntos (diferentes) en el plano euclidiano normal, cada uno de los cuales tiene coordenadas enteras. Demuestre que algún par tiene un punto medio con coordenadas enteras.

Ejercicios para computadora

1. Escriba un programa que genere todas las combinaciones r de los elementos $\{1, \ldots, n\}$.
2. Escriba un programa que genere todas las permutaciones de los elementos $\{1, \ldots, n\}$.
3. Escriba un programa que genere todas las permutaciones r de los elementos $\{1, \ldots, n\}$.
4. [*Proyecto*] Entregue un trabajo acerca de algoritmos diferentes de los presentados en este capítulo para generar combinaciones y permutaciones. Desarrolle algunos de estos algoritmos como programas.
5. Escriba un programa que liste todas las permutaciones de *ABCDEF* en las que *A* aparece antes que *D*.
6. Escriba un programa que liste todas las permutaciones de *ABCDEF* en las que *C* y *E* estén juntas en cualquier orden.
7. Escriba un programa que liste todas las maneras en que m marcianos y n venusinos pueden esperar en una fila si no debe haber dos venusinos juntos.
8. Escriba un programa para calcular los números de Catalan.
9. Escriba un programa que genere el triángulo de Pascal hasta el nivel n, para n arbitraria.
10. Escriba un programa que encuentre una subsucesión creciente o decreciente de longitud $n + 1$ de una sucesión de $n^2 + 1$ números distintos.



Capítulo 7

RELACIONES DE RECURRENCIA

¿Quieres decirme ahora?
¿Decirte qué?
¿Qué tratas de investigar? ¿Sabes? Es curioso, tratas de investigar qué es lo que quiere tu papá que investigue, y yo trato de investigar por qué quieres investigar. Podría seguir para siempre, ¿no es así?

DE *THE BIG SLEEP*

Este capítulo ofrece una introducción a las relaciones de recurrencia. Estas relaciones son útiles en ciertos problemas de conteo. Una relación de recurrencia relaciona el n-ésimo elemento de una sucesión con sus predecesores. Como las relaciones de recurrencia tienen una relación cercana con los algoritmos recursivos, las relaciones de recurrencia surgen de manera natural en el análisis de éstos.

7.1 ➜ Introducción

Considere las siguientes instrucciones para generar una sucesión:

1. Iniciar con 5.
2. Dado cualquier término, sume 3 para obtener el siguiente.

WWW

Si se listan los términos de la sucesión, se obtiene

$$5,\ 8,\ 11,\ 14,\ 17,\ \ldots. \tag{7.1.1}$$

El primer término es 5 por la instrucción 1. El segundo término es 8 porque la instrucción 2 dice que se sume 3 a 5 para obtener el siguiente término. El tercer término es 11 porque la instrucción 2 dice que se sume 3 a 8 para obtener el siguiente término. Si se siguen las instrucciones 1 y 2, se puede calcular cualquier término de la sucesión. Las instrucciones 1 y 2 no dan una fórmula explícita para el n-ésimo término de la sucesión en el sentido de proporcionar una fórmula en la que se pueda "sustituir n" para obtener el valor del n-ésimo término, sino que al calcular término por término en algún momento se podrá obtener cualquier término de la sucesión.

Si la sucesión (7.1.1) se denota por $a_1, a_2, \ldots$, se puede enunciar de nuevo la instrucción 1 como

$$a_1 = 5 \tag{7.1.2}$$

y la instrucción 2 se puede establecer como

$$a_n = a_{n-1} + 3, \quad n \geq 2. \tag{7.1.3}$$

Haciendo $n = 2$ en (7.1.3), se obtiene

$$a_2 = a_1 + 3.$$

Por (7.1.2), $a_1 = 5$; entonces

$$a_2 = a_1 + 3 = 5 + 3 = 8.$$

Haciendo $n = 3$ en (7.1.3), se obtiene

$$a_3 = a_2 + 3.$$

Como $a_2 = 8$,

$$a_3 = a_2 + 3 = 8 + 3 = 11.$$

Usando (7.1.2) y (7.1.3), es posible calcular cualquier término en la sucesión justo como se hizo al seguir las instrucciones 1 y 2. Se observa que (7.1.2) y (7.1.3) son equivalentes a las instrucciones 1 y 2.

La ecuación (7.1.3) proporciona un ejemplo de **relación de recurrencia.** Ésta define una sucesión expresando el n-ésimo valor en términos de ciertos predecesores. En (7.1.3) el n-ésimo valor está dado en términos del valor precedente inmediato. Para que una relación de recurrencia como la (7.1.3) defina una sucesión, debe proporcionarse un valor o valores de "inicio", como (7.1.2). Estos valores de inicio se llaman **condiciones iniciales**. Las definiciones formales son las siguientes.

Definición 7.1.1 ▶

Una *relación de recurrencia* para la sucesión $a_0, a_1, \ldots$ es una ecuación que relaciona a_n con ciertos predecesores $a_0, a_1, \ldots, a_{n-1}$.

Las *condiciones iniciales* para una sucesión $a_0, a_1, \ldots$ son valores dados en forma explícita para un número finito de términos de la sucesión. ◀

Se ha visto que es posible definir una sucesión mediante una relación de recurrencia con ciertas condiciones iniciales. Ahora se darán varios ejemplos de relaciones de recurrencia.

Ejemplo 7.1.2 ▶

La sucesión de Fibonacci (vea el análisis que sigue al algoritmo 4.4.6) se define por la relación de recurrencia

$$f_n = f_{n-1} + f_{n-2}, \qquad n \geq 3,$$

y las condiciones iniciales

$$f_1 = 1, \qquad f_2 = 1.$$ ◀

Ejemplo 7.1.3 ▶

Una persona invierte \$1000 al 12% de interés anual compuesto. Si A_n representa la cantidad al final de n años, encuentre una relación de recurrencia y las condiciones iniciales que definen la sucesión $\{A_n\}$.

WWW

Al final de $n - 1$ años, la cantidad es A_{n-1}. Después de un año más, se tendrá la cantidad A_{n-1} más el interés. Entonces

$$A_n = A_{n-1} + (0.12)A_{n-1} = (1.12)A_{n-1}, \qquad n \geq 1. \tag{7.1.4}$$

Para aplicar esta relación de recurrencia para $n = 1$, es necesario conocer el valor de A_0. Como A_0 es la cantidad inicial, se tiene la condición inicial

$$A_0 = 1000. \tag{7.1.5}$$

◀

La condición inicial (7.1.5) y la relación de recurrencia (7.1.4) permiten calcular el valor de A_n para cualquier n. Por ejemplo,

$$\begin{aligned} A_3 &= (1.12)A_2 = (1.12)(1.12)A_1 \\ &= (1.12)(1.12)(1.12)A_0 = (1.12)^3(1000) = 1404.93. \end{aligned} \tag{7.1.6}$$

Así, al final del tercer año, la cantidad es \$1404.93.

El cálculo (7.1.6) se puede realizar para un valor arbitrario de n para obtener

$$\begin{aligned} A_n &= (1.12)A_{n-1} \\ &\vdots \\ &= (1.12)^n(1000). \end{aligned}$$

Se observa que en ocasiones es posible derivar una fórmula explícita a partir de una relación de recurrencia y las condiciones iniciales. Encontrar fórmulas explícitas a partir de relaciones de recurrencia es el tema de la sección 7.2.

Aunque es fácil obtener una fórmula explícita a partir de la relación de recurrencia y la condición inicial para la sucesión del ejemplo 7.1.3, no resulta tan evidente cómo obtener una fórmula explícita para la sucesión de Fibonacci. En la sección 7.2 se expone un método que lleva a esta fórmula.

Las relaciones de recurrencia, los algoritmos recursivos y la inducción matemática tienen una conexión estrecha. En los tres, se supone que se conocen casos anteriores del caso actual. Una relación de recurrencia usa valores anteriores en secuencia para calcular el valor actual. Un algoritmo recursivo usa casos más pequeños de la entrada actual para procesar esta entrada. El paso inductivo en una demostración por inducción matemática supone la verdad de casos anteriores del enunciado para probar la verdad del enunciado actual.

Una relación de recurrencia que define una sucesión se puede convertir directamente en un algoritmo para calcular la sucesión. Por ejemplo, el algoritmo 7.1.4, derivado de la relación de recurrencia (7.1.4) y la condición inicial (7.1.5), calcula la sucesión del ejemplo 7.1.3.

Algoritmo 7.1.4

Cálculo del interés compuesto

Este algoritmo recursivo calcula la cantidad de dinero al final de n años suponiendo una cantidad inicial de \$1000 y una tasa anual de interés del 12% compuesto.

Entrada: n, el número de años
Salida: la cantidad de dinero al final de n años

```
1.  interes_compuesto(n) {
2.      if (n == 0)
3.          return 1000
4.      return 1.12*interes_compuesto(n − 1)
5.  }
```

El algoritmo 7.1.4 es una traducción directa de las ecuaciones (7.1.4) y (7.1.5), que definen la sucesión $A_0, A_1, \ldots$. Las líneas 2 y 3 corresponden a la condición inicial (7.1.5), y la línea 4 corresponde a la relación de recurrencia (7.1.4).

Ejemplo 7.1.5 ▶

Sea S_n el número de subconjuntos de un conjunto de n elementos. Como al pasar de un conjunto de $(n-1)$ elementos a un conjunto de n elementos se duplica el número de subconjuntos (vea el teorema 2.1.6), se obtiene la relación de recurrencia

$$S_n = 2S_{n-1}.$$

La condición inicial es

$$S_0 = 1.$$

◀

Una de las razones principales para usar relaciones de recurrencia es que algunas veces es más sencillo determinar el n-ésimo término de una sucesión en términos de sus predecesores que encontrar una fórmula explícita para el n-ésimo término en términos de n. Los siguientes ejemplos intentan ilustrar esta tesis.

Ejemplo 7.1.6 ▶

Sea S_n el número de cadenas de n bits que no contienen el patrón 111. Desarrolle la relación de recurrencia para $S_1, S_2, \ldots$ y las condiciones iniciales que definen la sucesión S.

Se contará el número de cadenas de n bits que no contienen el patrón 111

a) que comienzan con 0;

b) que comienzan con 10;

c) que comienzan con 11.

Como el conjunto de cadenas de los tipos *a)*, *b)* y *c)* son ajenos, por el principio de la suma, S_n será igual a la suma de los números de cadenas de los tipos *a)*, *b)* y *c)*. Suponga que una cadena de n bits comienza con 0 y no contiene el patrón 111. Entonces la cadena de $(n-1)$ bits que sigue al 0 inicial no contiene el patrón 111. Como cualquier cadena de $(n-1)$ bits que no contenga 111 puede seguir al 0 inicial, hay S_{n-1} cadenas del tipo *a)*. Si una cadena de n bits comienza con 10 y no contiene el patrón 111, entonces la cadena de $(n-2)$ bits que sigue al 10 inicial no puede contener el patrón 111; por lo tanto, hay S_{n-2} cadenas del tipo *b)*. Si una cadena de n bits comienza con 11 y no contiene el patrón 111, entonces el tercer bit debe ser 0. La cadena de $(n-3)$ bits que sigue al 110 inicial no puede contener el patrón 111; por lo tanto, hay S_{n-3} cadenas del tipo *c)*. Entonces

$$S_n = S_{n-1} + S_{n-2} + S_{n-3}, \qquad n \geq 4.$$

Por inspección, se encuentran las condiciones iniciales

$$S_1 = 2, \qquad S_2 = 4, \qquad S_3 = 7.$$ ◀

Ejemplo 7.1.7 ▶

Recuerde (vea el ejemplo 6.2.23) que el número de Catalan C_n es igual al número de rutas de la esquina inferior izquierda de una rejilla cuadrada de $n \times n$ a la esquina superior derecha si estamos restringidos a viajar sólo a la derecha o hacia arriba y si se permite tocar pero no pasar arriba de la diagonal entre la esquina inferior izquierda y la superior derecha. Tal ruta recibe el nombre de *ruta buena*. Se da una relación de recurrencia para los números de Catalan.

Las rutas buenas se dividen en clases con base en la primera vez que tocan la diagonal después de salir de la esquina inferior derecha. Por ejemplo, la ruta en la figura 7.1.1 toca la diagonal primero en el punto (3, 3). Las rutas que tocan la diagonal primero en el punto (k, k) se consideran construidas por un proceso de dos pasos: primero, se construye la parte de (0, 0) a (k, k). Segundo, se construye la parte de (k, k) a (n, n). Una buena ruta siempre sale de (0, 0) moviéndose hacia la derecha a (1, 0) y siempre llega a (k, k) moviéndose hacia arriba desde $(k, k-1)$. Los movimientos de (1, 0) a $(k, k-1)$ dan una ruta

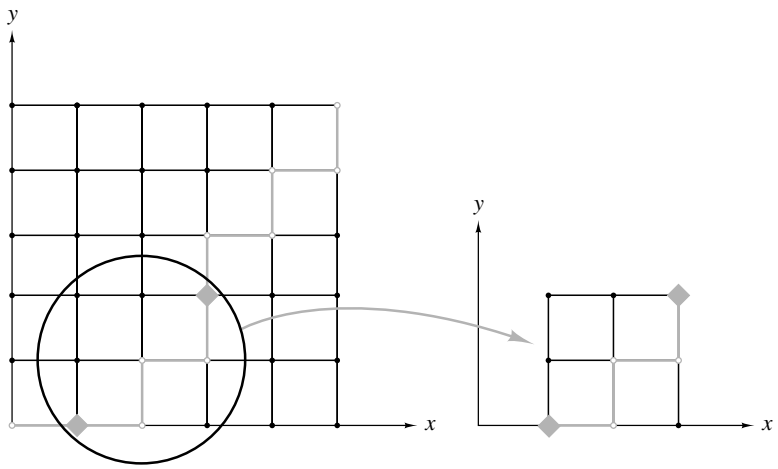


Figura 7.1.1 Descomposición de una ruta buena.

buena en la rejilla de $(k-1) \times (k-1)$ con esquina en $(1, 0)$, $(1, k-1)$, $(k, k-1)$ y $(k, 0)$. [En la figura 7.1.1, se marcaron los puntos $(1, 0)$ y $(k, k-1)$, $k = 3$, con rombos, y se aisló la subrejilla de $(k-1) \times (k-1)$]. Así, hay C_{k-1} rutas de $(0, 0)$ a (k, k) que tocan primero a la diagonal en (k, k). La parte de (k, k) a (n, n) es una buena ruta en la rejilla de $(n-k) \times (n-k)$ con esquinas en (k, k), (k, n), (n, n) y (n, k) (vea la figura 7.1.1). Hay C_{n-1} rutas de este tipo. Por el principio de la multiplicación, hay $C_{k-1}C_{n-k}$ rutas buenas en una rejilla de $n \times n$ que tocan primero la diagonal en (k, k). Las rutas buenas que tocan por primera vez la diagonal en (k, k) son diferentes de las que tocan la diagonal por primera vez en (k', k'), $k \neq k'$. Entonces se utiliza el principio de la suma a fin de obtener una relación de recurrencia para el número total de rutas buenas en una rejilla de $n \times n$:

$$C_n = \sum_{k=1}^{n} C_{k-1}C_{n-k}. \qquad \blacktriangleleft$$

Ejemplo 7.1.8 ▶

Torre de Hanoi

La Torre de Hanoi es un juego que consiste en tres estacas montadas en una tabla y n discos de varios tamaños con agujeros en sus centros (vea la figura 7.1.2). Se supone que si un disco está en una estaca, sólo un disco de diámetro más pequeño se puede colocar encima de él. Si se tienen todos los discos apilados en una estaca como en la figura 7.1.2, el problema es transferir los discos a otra estaca moviendo un disco a la vez.

WWW

Se dará una solución y después se encontrará una relación de recurrencia y una condición inicial para la sucesión $c_1, c_2, \ldots$, donde c_n denota el número de movimientos que requiere nuestra solución para resolver el juego de n discos. Después se mostrará que nuestra solución es óptima; es decir, se demostrará que ninguna otra solución usa menos movimientos.

Se dará un algoritmo recursivo. Si hay sólo un disco, simplemente se mueve a la estaca deseada. Si se tienen $n > 1$ discos en la estaca 1 como en la figura 7.1.2, se comienza por invocar de manera recursiva el algoritmo para mover los $n-1$ discos superiores a la estaca 2 (vea la figura 7.1.3). Durante estos movimientos, el disco hasta abajo de la estaca 1 se queda fijo. Después, se mueve el disco que queda en la estaca 1 a la estaca 3. Por último, de nuevo se invoca el algoritmo de manera recursiva para mover los $n-1$ discos de la estaca 2 a la estaca 3. Se han movido con éxito n discos de la estaca 1 a la estaca 3.

Si $n > 1$, el problema de los $(n-1)$ discos se resuelve dos veces y se mueve un disco de manera explícita. Por lo tanto,

$$c_n = 2c_{n-1} + 1, \qquad n > 1.$$

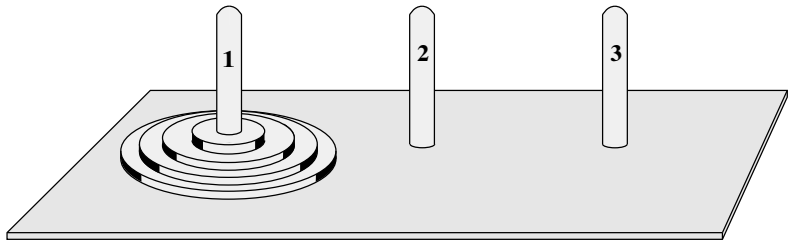


Figura 7.1.2 Torre de Hanoi.

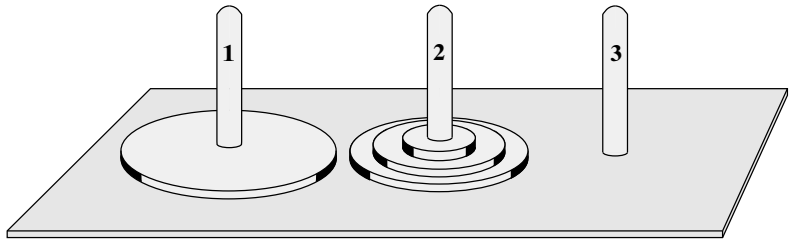


Figura 7.1.3 Resultado después de mover en forma recursiva los $n-1$ discos superiores de la estaca 1 a la estaca 2 en la Torre de Hanoi.

La condición inicial es

$$c_1 = 1.$$

En la sección 7.2, se demostrará que $c_n = 2^n - 1$.

Ahora se probará que nuestra solución es óptima. Sea d_n el número de movimientos que requiere una solución óptima. Se usa inducción matemática para demostrar que

$$c_n = d_n, \qquad n \geq 1. \tag{7.1.7}$$

Paso base ($n = 1$)

Por inspección,

$$c_1 = 1 = d_1;$$

entonces (7.1.7) es verdadera cuando $n = 1$.

Paso inductivo

Suponga que (7.1.7) es verdadera para $n - 1$. Considere el punto en una solución óptima del problema de n discos cuando el disco más grande se mueve por primera vez. El disco más grande debe estar solo en una estaca (de manera que pueda sacarse de la estaca) y otra estaca debe estar vacía (de manera que pueda recibir el disco más grande). Entonces los $n - 1$ discos más pequeños deben estar apilados en la tercera estaca (vea la figura 7.1.3). En otra palabras, debe haberse resuelto el problema de $(n - 1)$ discos que requiere al menos d_{n-1} movimientos. El disco más grande se mueve, y esto requiere un movimiento adicional. Por último, en algún punto se movieron los $n - 1$ discos encima del disco más grande, lo que requirió al menos d_{n-1} movimientos adicionales. Se concluye que

$$d_n \geq 2d_{n-1} + 1.$$

Por la suposición inductiva, $c_{n-1} = d_{n-1}$. Entonces

$$d_n \geq 2d_{n-1} + 1 = 2c_{n-1} + 1 = c_n. \tag{7.1.8}$$

La última igualdad se deriva de la relación de recurrencia para la sucesión $c_1, c_2, \ldots$. Por definición, ninguna solución requerirá menos movimientos que una solución óptima, de manera que

$$c_n \geq d_n. \tag{7.1.9}$$

Las desigualdades (7.1.8) y (7.1.9) se combinan para dar

$$c_n = d_n.$$

El paso inductivo queda completo. Por lo tanto, nuestra solución es óptima. ◀

El juego de la Torre de Hanoi fue inventado por el matemático francés Édouard Lucas a finales del siglo XIX. (Lucas fue la primera persona en llamar a la sucesión 1, 1, 2, 3, 5, . . . la sucesión de Fibonacci). También se creó el siguiente mito que acompañaba al juego (para ayudar a venderlo, según se piensa). Se decía que el juego provenía de una torre mítica de oro que consistía en 64 discos. Los monjes debían transferir los 64 discos según las reglas establecidas. Se decía que antes de que los monjes terminaran de mover la torre, ésta se colapsaría y vendría el fin del mundo con el estruendo de un relámpago. Como se requieren al menos $2^{64} - 1 = 18{,}446{,}744{,}073{,}709{,}551{,}615$ movimientos para resolver el juego de la Torre de Hanoi con 64 discos, se puede tener la certeza de que algo hubiera ocurrido a la torre antes de moverla por completo.

Ejemplo 7.1.9 ▶

La tela de araña en economía

Se supondrá un modelo económico en el que la oferta y la demanda están dadas por ecuaciones lineales (vea la figura 7.1.4). En particular, la demanda está dada por la ecuación

$$p = a - bq,$$

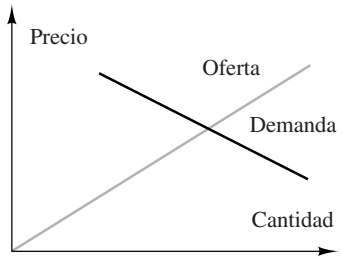


Figura 7.1.4 Un modelo económico.

donde p es el precio, q es la cantidad, y a y b son parámetros positivos. La idea es que al aumentar el precio, los consumidores demandan menos producto. La oferta está dada por la ecuación

$$p = kq,$$

donde p es el precio, q es la cantidad, y k es un parámetro positivo. La idea es que conforme aumenta el precio, el fabricante estará dispuesto a proporcionar cantidades mayores.

Se supondrá también que hay un tiempo de retraso mientras el proveedor reacciona a los cambios. (Por ejemplo, toma tiempo fabricar bienes y toma tiempo que la cosecha crezca). Los intervalos de tiempo discretos se denotan por $n = 0, 1, \ldots$. Se supone que la demanda está dada por la ecuación

$$p_n = a - bq_n;$$

es decir, en el tiempo n, la cantidad q_n de producto se venderá al precio p_n. Se supone que la oferta está dada por la ecuación

$$p_n = kq_{n+1}; \tag{7.1.10}$$

esto es, se requiere una unidad de tiempo para que la manufactura se ajuste a la cantidad q_{n+1}, en el tiempo $n + 1$, al precio p_n, del tiempo anterior n.

Si se despeja q_{n+1} de la ecuación (7.1.10) y se sustituye en la ecuación de la demanda para el tiempo $n + 1$,

$$p_{n+1} = a - bq_{n+1},$$

se obtiene la relación de recurrencia

$$p_{n+1} = a - \frac{b}{k}p_n$$

para el precio. Esta relación de recurrencia se resolverá en la sección 7.2.

Los cambios de precio en el tiempo se observan en una gráfica. Si el precio inicial es p_0, el fabricante estará dispuesto a proporcionar la cantidad q_1, en el tiempo $n = 1$. Se localiza esta cantidad moviéndose horizontalmente a la curva de oferta (vea la figura 7.1.5). Sin embargo, las fuerzas del mercado bajan el precio a p_1, como se ve al moverse verticalmente a la curva de demanda. En el precio p_1, el fabricante estará dispuesto a proporcionar la cantidad q_2 en el tiempo $n = 2$, como se ve al moverse horizontalmente a la curva de oferta. Ahora las fuerzas de mercado suben el precio a p_2, como lo muestra el movimiento vertical a la curva de demanda. Si continúa este proceso, se obtiene la tela de araña mostrada en la figura 7.1.5.

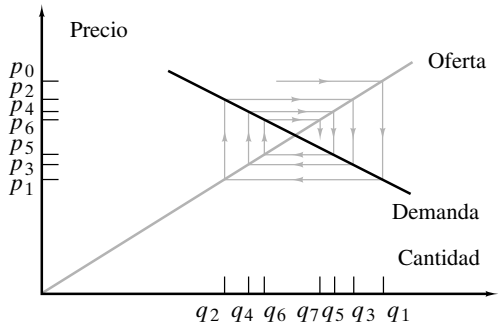


Figura 7.1.5 Tela de araña con un precio estabilizador.

Para las funciones de oferta y demanda de la figura 7.1.5, el precio se acerca al dado por la intersección de las curvas de oferta y demanda. Sin embargo, no siempre ocurre esto. Por ejemplo, en la figura 7.1.6, el precio fluctúa entre p_0 y p_1, mientras que en la figura 7.1.7, los cambios de precio son cada vez más pronunciados. El comportamiento está determinado por las pendientes de las líneas de oferta y demanda. Para producir el comportamiento fluctuante

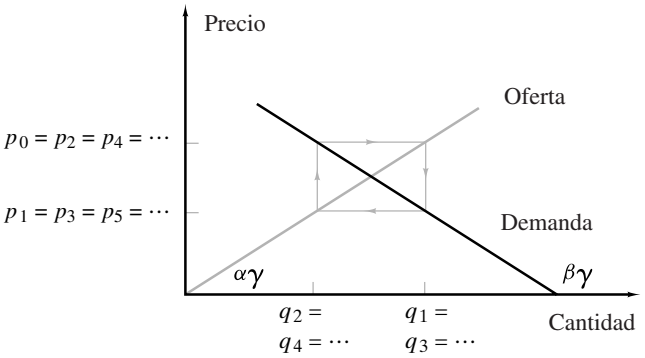


Figura 7.1.6 Tela de araña con fluctuaciones de precio.

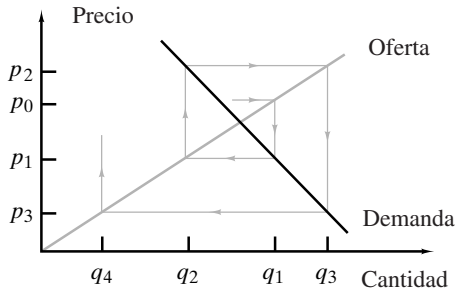


Figura 7.1.7 Tela de araña con cambios de precios crecientes.

de la figura 7.1.6, los ángulos α y β deben sumar 180°. Las pendientes de las curvas de oferta y demanda son tan α y tan β, respectivamente; entonces en la figura 7.1.6, se tiene.

$$k = \tan\alpha = -\tan\beta = b.$$

Se ha demostrado que el precio fluctúa entre dos valores cuando $k = b$. Un análisis similar muestra que el precio tiende al valor dado por la intersección de las curvas de oferta y demanda (figura 7.1.5) cuando $b < k$; el caso de los cambios de precio crecientes (figura 7.1.7) ocurre cuando $b > k$ (vea los ejercicios 35 y 36). En la sección 7.2 se analiza el comportamiento del precio en el tiempo mediante el análisis de una fórmula explícita para el precio p_n. ◀

Es posible extender la definición de relación de recurrencia para incluir funciones indexadas sobre n-eadas de enteros positivos. El último ejemplo es de esta forma.

Ejemplo 7.1.10 ▶

Función de Ackermann

Las funciones de Ackermann se definen por las relaciones de recurrencia

$$A(m, 0) = A(m-1, 1), \qquad m = 1, 2, \ldots, \tag{7.1.11}$$

$$A(m, n) = A(m-1, A(m, n-1)), \qquad m = 1, 2, \ldots, \quad n = 1, 2, \ldots, \tag{7.1.12}$$

WWW y las condiciones iniciales

$$A(0, n) = n + 1, \qquad n = 0, 1, \ldots. \tag{7.1.13}$$

La función de Ackermann es de importancia teórica por su rápida tasa de crecimiento. Las funciones relacionadas con la función de Ackermann aparecen en la complejidad del tiempo de ciertos algoritmos como el tiempo para ejecutar algoritmos de unir/encontrar (vea [Tarjan, pp. 22-29]).

El cálculo

$$\begin{aligned} A(1, 1) &= A(0, A(1, 0)) && \text{by (7.1.12)} \\ &= A(0, A(0, 1)) && \text{by (7.1.11)} \\ &= A(0, 2) && \text{by (7.1.13)} \\ &= 3 && \text{by (7.1.13)} \end{aligned}$$

ilustra el uso de las ecuaciones (7.1.11) a (7.13). ◀

Sugerencias para resolver problemas

Para establecer una relación de recurrencia, primero se *define*, digamos A_n, como la cantidad deseada. Por ejemplo, sea A_n la cantidad de dinero al final del n años si se invierten \$1000 al 12% de interés anual compuesto. Después, igual que en la demostración por inducción, *se buscan casos más pequeños dentro del caso n*. Todavía en el ejemplo del interés compuesto, el caso $n - 1$ está "dentro" del caso n en el sentido de que la cantidad de

dinero al final de n años es la cantidad al final de $n-1$ años más el interés. Se obtiene la relación de recurrencia

$$A_n = A_{n-1} + (0.12)A_{n-1}.$$

Sección de ejercicios de repaso

1. ¿Qué es una relación de recurrencia?
2. ¿Qué es una condición inicial?
3. ¿Qué es el interés compuesto y cómo se describe mediante una relación de recurrencia?
4. ¿En qué consiste el juego de la Torre de Hanoi?
5. Dé una solución al juego de la Torre de Hanoi.
6. Describa la tela de araña de la economía.
7. Defina la *función de Ackermann*.

Ejercicios

En los ejercicios 1 al 3, encuentre una relación de recurrencia y condiciones iniciales que generen una sucesión que comience con los términos dados.

1. 3, 7, 11, 15, . . .
2. 3, 6, 9, 15, 24, 39, . . .
3. 1, 1, 2, 4, 16, 128, 4096, . . .

En los ejercicios 4 al 8, suponga que una persona invierte \$2000 al 14% de interés anual compuesto. Sea A_n la cantidad al final de n años.

4. Encuentre una relación de recurrencia para la sucesión $A_0, A_1, \ldots$.
5. Encuentre una condición inicial para la sucesión $A_0, A_1, \ldots$.
6. Encuentre A_1, A_2 y A_3.
7. Encuentre una fórmula explícita para A_n.
8. ¿Cuánto tiempo tomará que una persona duplique su inversión inicial?

Si una persona invierte en una anualidad con amparo tributario, el dinero invertido, igual que el interés ganado, no está sujeto a gravamen sino hasta que se retira de la cuenta. En los ejercicios 9 al 12, suponga que una persona invierte \$2000 cada año en una anualidad con amparo tributario a un interés del 10% anual compuesto. Sea A_n la cantidad de dinero al final de n años.

9. Encuentre una relación de recurrencia para la sucesión $A_0, A_1, \ldots$.
10. Encuentre una condición inicial para la sucesión $A_0, A_1, \ldots$.
11. Encuentre A_1, A_2 y A_3.
12. Encuentre una fórmula explícita para A_n.

En los ejercicios 13 al 17, suponga que una persona invierte \$3000 al 12% de interés compuesto cada trimestre. Sea A_n la cantidad al final de n años.

13. Encuentre una relación de recurrencia para la sucesión $A_0, A_1, \ldots$.
14. Encuentre una condición inicial para la sucesión $A_0, A_1, \ldots$.
15. Encuentre A_1, A_2 y A_3.
16. Encuentre una fórmula explícita para A_n.
17. ¿Cuánto tiempo tomará que una persona duplique la inversión inicial?
18. Sea S_n el número de cadenas de n bits que no contienen el patrón 000. Encuentre una relación de recurrencia y condiciones iniciales para la sucesión $\{S_n\}$.

Los ejercicios 19 al 21 se refieren a la sucesión S donde S_n denota el número de cadenas de n bits que no contienen el patrón 000.

19. Encuentre una relación de recurrencia y condiciones iniciales para la sucesión $\{S_n\}$.
20. Demuestre que $S_n = f_{n+2}$, $n = 1, 2, \ldots$, donde f denota la sucesión de Fibonacci.
21. Considerando el número de cadenas de n bits con exactamente i ceros y el ejercicio 20, demuestre que

$$f_{n+2} = \sum_{i=0}^{\lfloor (n+1)/2 \rfloor} C(n+1-i, i), \qquad n = 1, 2, \ldots,$$

donde f denota la sucesión de Fibonacci.

Los ejercicios 22 al 24 se refieren a la sucesión $S_1, S_2, \ldots$, donde S_n denota el número de cadenas de n bits que no contienen el patrón 010.

22. Calcule S_1, S_2, S_3 y S_4.
23. Considerando el número de cadenas de n bits que no contienen el patrón 010 y que no comienzan con 0 (es decir, comienzan con 1); que comienzan con un 0 (es decir, comienzan con 01); que comienzan con dos ceros, etcétera, derive la relación de recurrencia

$$S_n = S_{n-1} + S_{n-3} + S_{n-4} + S_{n-5} + \cdots + S_1 + 3. \qquad \textbf{(7.1.14)}$$

24. Sustituyendo n por $n-1$ en (7.1.14), escriba una fórmula para S_{n-1}. Reste la fórmula para S_{n-1} de la fórmula para S_n y utilice el resultado para derivar la relación de recurrencia

$$S_n = 2S_{n-1} - S_{n-2} + S_{n-3}.$$

En los ejercicios 25 al 30, $C_0, C_1, C_2, \ldots$ denota la sucesión de números de Catalan.

25. Dado que $C_0 = C_1 = 1$ y $C_2 = 2$, calcule C_3, C_4 y C_5 usando la relación de recurrencia del ejemplo 7.1.7.
26. Demuestre que los números de Catalan están dados por la relación de recurrencia

$$(n+2)C_{n+1} = (4n+2)C_n, \qquad n \geq 0,$$

y la condición inicial $C_0 = 1$.

27. Pruebe que

$$C_n \geq \frac{4^{n-1}}{n^2} \qquad \text{para toda } n \geq 1.$$

28. Derive una relación de recurrencia y una condición inicial para el número de maneras de colocar paréntesis en el producto

$$a_1 * a_2 * \cdots * a_n, \qquad n \geq 2.$$

Ejemplos: Existe una manera de colocar paréntesis en $a_1 * a_2$, a saber,

$(a_1 * a_2)$. Existen dos maneras de colocar paréntesis en $a_1 * a_2 * a_3$, a saber, $((a_1 * a_2) * a_3)$ y $(a_1 * (a_2 * a_3))$. Deduzca que el número de maneras de colocar paréntesis en el producto de n elementos es C_{n-1}, $n \geq 2$.

★ ***29.*** Derive una relación de recurrencia y una condición inicial para el número de maneras de dividir un polígono convexo de $(n + 2)$ lados, $n \geq 1$, en triángulos dibujando $n - 1$ líneas a través de los vértices que no se cruzan en el interior del polígono. (Un polígono es *convexo* si cualquier línea que une dos puntos del polígono está completamente contenida en el polígono). Por ejemplo, existen cinco maneras de dividir un pentágono convexo en triángulos dibujando dos líneas que no se cruzan a través de los vértices.

Deduzca que el número de maneras de dividir un polígono convexo de $(n + 2)$ lados en triángulos dibujando $n - 1$ líneas que no se cruzan a través de los vértices es C_n, $n \geq 1$.

30. Dividiendo las rutas de la esquina inferior izquierda a la esquina superior derecha en una rejilla de $(n + 1)\times(n + 1)$, donde el movimiento está restringido sólo a la derecha o arriba, en clases basadas en la primera vez que la ruta llega a la diagonal que va de la esquina inferior izquierda a la superior derecha, después de salir de la esquina inferior izquierda, derive la relación de recurrencia

$$C_n = \frac{1}{2}C(2(n+1), n+1) - \sum_{k=0}^{n-1} C_k C(2(n-k), n-k).$$

En los ejercicios 31 y 32, sea S_n el número de rutas desde la esquina inferior izquierda de una rejilla de $n \times n$ a la esquina superior derecha, donde el movimiento está restringido a la derecha, arriba o en diagonal al noreste [es decir, de (i, j) a $(i + 1, j + 1)$] y donde se permite tocar pero no cruzar la diagonal de la esquina inferior izquierda a la superior derecha. Los números S_0, S_1, . . . *se llaman números de Schröeder.*

31. Demuestre que $S_0 = 1$, $S_1 = 2$, $S_2 = 6$ y $S_3 = 22$.

32. Derive una relación de recurrencia para la sucesión de Schröeder.

33. Escriba las soluciones explícitas para el juego de la Torre de Hanoi para $n = 3, 4$.

34. ¿A qué valores tienden el precio y la cantidad en el ejemplo 7.1.9 cuando $b < k$?

35. Demuestre que cuando $b < k$ en el ejemplo 7.1.9, el precio tiende al dado por la intersección de las curvas de oferta y demanda.

36. Demuestre que cuando $b > k$ en el ejemplo 7.1.9, las diferencias entre precios sucesivos aumentan.

Los ejercicios 37 al 43 se refieren a la función de Ackermann A(m, n).

37. Calcule $A(2, 2)$ y $A(2, 3)$.

38. Utilice inducción para demostrar que

$$A(1, n) = n + 2, \qquad n = 0, 1, \ldots.$$

39. Utilice inducción para demostrar que

$$A(2, n) = 3 + 2n, \qquad n = 0, 1 \ldots.$$

40. Suponga una fórmula para $A(3, n)$y pruébela por inducción.

★ **41.** Pruebe que $A(m, n) > n$ para toda $m \geq 0$, $n \geq 0$ por inducción sobre m. El paso inductivo usará inducción sobre n.

42. Usando el ejercicio 41 o de otra manera, pruebe que $A(m, n) > 1$ para toda $m \geq 1$, $n \geq 0$.

43. Usando el ejercicio 41 o de otra manera, pruebe que $A(m, n) < A(m, n + 1)$ para toda $m \geq 0$, $n \geq 0$.

Lo que hemos llamado la función de Ackermann, en realidad se deriva de la función de Ackermann original definida por

$$\begin{aligned}
AO(0, y, z) &= z + 1, && y, z \geq 0,\\
AO(1, y, z) &= y + z, && y, z \geq 0,\\
AO(2, y, z) &= yz, && y, z \geq 0,\\
AO(x + 3, y, 0) &= 1, && x, y \geq 0,\\
AO(x + 3, y, z + 1) &= && \\
&\quad AO(x + 2, y, AO(x + 3, y, z)), && x, y, z \geq 0.
\end{aligned}$$

Los ejercicios 44 al 47 se refieren a la función AO y a la función de Ackermann A.

44. Demuestre que $A(x, y) = AO(x, 2, y + 3) - 3$ para $y \geq 0$ y $x = 0, 1, 2$.

45. Demuestre que $AO(x, 2, 1) = 2$ para $x \geq 2$.

46. Demuestre que $AO(x, 2, 2) = 4$ para $x \geq 2$.

★ ***47.*** Demuestre que $A(x, y) = AO(x, 2, y + 3) - 3$ para $x, y \geq 0$.

48. Una red consiste en n nodos. Cada nodo tiene instalaciones de comunicaciones y almacenamiento local. Periódicamente, todos los archivos deben compartirse. Un *enlace* consiste en dos nodos que comparten archivos. En particular, cando los nodos A y B se enlazan, A transmite todos sus archivos a B y B transmite todos sus archivos a A. sólo existe un enlace a la vez, y después de establecer un enlace y compartir archivos, el enlace se elimina. Sea a_n el número mínimo de enlaces requerido por n nodos para que todos los archivos estén disponibles para todos lo nodos.

a) Demuestre que $a_2 = 1$, $a_3 \leq 3$, $a_4 \leq 4$.

b) Demuestre que $a_n \leq a_{n-1} + 2$, $n \geq 3$.

49. Si P_n denota el número de permutaciones de n objetos diferentes, encuentre una relación de recurrencia y una condición inicial para la sucesión $P_1, P_2, \ldots$.

50. Suponga que tiene n dólares y que cada día compra ya sea jugo de naranja (\$1), leche (\$2) o cerveza (\$2). Si R_n es el número de maneras de gastar todo el dinero, demuestre que

$$R_n = R_{n-1} + 2R_{n-2}.$$

Tome en cuenta el orden. Por ejemplo, hay 11 maneras de gastar \$4: *LC*, *CL*, *NNL*, *NNC*, *NLN*, *NCN*, *LNN*, *CNN*, *NNNN*, *LL*, *CC*.

51. Suponga que tiene n dólares y que cada día compra ya sea cinta adhesiva (\$1), papel (\$1), plumas (\$2), lápices (\$2) o clips (\$3). Si R_n es el número de maneras de gastar todo el dinero, derive una relación de recurrencia para la sucesión $R_1, R_2, \ldots$.

52. Sea R_n el número de regiones en las que se divide el plano por n líneas. Suponga que cada par de líneas se unen en un punto, pero que nunca se unen tres líneas en un punto. Derive una relación de recurrencia para la sucesión $R_1, R_2, \ldots$.

Los ejercicios 53 y 54 se refieren a la sucesión S_n definida por

$$S_1 = 0, \quad S_2 = 1, \quad S_n = \frac{S_{n-1} + S_{n-2}}{2}, \quad n = 3, 4, \ldots.$$

53. Calcule S_3 y S_4.

★ **54.** Suponga una fórmula para S_n y use inducción matemática para demostrar que es correcta.

★ ***55.*** Sea F_n el número de funciones f de $X = \{1, \ldots, n\}$ en X que tiene la propiedad de que si i está en el recorrido de f, entonces 1, 2, . . . ,

$i - 1$ también están en el recorrido de f. (Haga $F_0 = 1$.) Demuestre que la sucesión $F_0, F_1, \ldots$ satisface la relación de recurrencia

$$F_n = \sum_{j=0}^{n-1} C(n, j) F_j.$$

56. Si α es una cadena de bits, sea $C(\alpha)$ el número máximo de ceros consecutivos en α. [*Ejemplos:* C(10010) = 2, $C(00110001) = 3$]. Sea S_n el número cadenas α de n bits con $C(\alpha) \leq 2$. Desarrolle una relación de recurrencia para $S_1, S_2, \ldots$.

57. La sucesión $g_1, g_2, \ldots$ se define por la relación de recurrencia

$$g_n = g_{n-1} + g_{n-2} + 1, \qquad n \geq 3,$$

y las condiciones iniciales

$$g_1 = 1, \qquad g_2 = 3.$$

Utilizando inducción matemática o de otra manera, demuestre que

$$g_n = 2 f_{n+1} - 1, \qquad n \geq 1,$$

donde $f_1, f_2, \ldots$ es la sucesión de Fibonacci.

58. Considere la fórmula

$$u_n = \begin{cases} u_{3n+1} & \text{si } n \text{ es impar y mayor que 1} \\ u_{n/2} & \text{si } n \text{ es par y mayor que 1} \end{cases}$$

y la condición inicial $u_1 = 1$. Explique por qué la fórmula *no* es una relación de recurrencia. Una conjetura desde hace mucho tiempo es que para todo entero positivo n, u_n está definida y es igual a 1. Calcule u_n para $n = 2, \ldots, 7$.

59. Defina la sucesión $t_1, t_2, \ldots$ por la relación de recurrencia

$$t_n = t_{n-1} t_{n-2}, \qquad n \geq 3,$$

y condiciones iniciales

$$t_1 = 1, \qquad t_2 = 2.$$

¿Qué está mal en la siguiente "demostración" de que $t_n = 1$ para toda $n \geq 1$?

Paso base Para $n = 1$, se tiene $t_1 = 1$; entonces, el paso base se verifica.

Paso inductivo Suponga que $t_k = 1$ para $k < n$. Debemos probar que $t_n = 1$. Ahora

$$\begin{aligned} t_n &= t_{n-1} t_{n-2} \\ &= 1 \cdot 1 \quad \text{por la suposición inductiva} \\ &= 1. \end{aligned}$$

El paso inductivo queda completo.

60. Derive una relación de recurrencia para $C(n, k)$, el número de subconjuntos de k elementos de un conjunto de n elementos. En particular, escriba $C(n + 1, k)$ en términos de $C(n, i)$ para la i apropiada.

61. Derive una relación de recurrencia para $S(k, n)$, el número de maneras de elegir k elementos, con repeticiones permitidas, de n tipos disponibles. En particular, escriba $S(k, n)$ en términos de $S(k - 1, i)$ para la i apropiada.

62. Sea $S(n, k)$ el número de funciones de $\{1, \ldots, n\}$ sobre $\{1, \ldots, k\}$. Demuestre que $S(n, k)$ satisface la relación de recurrencia

$$S(n, k) = k^n - \sum_{i=1}^{k-1} C(k, i) S(n, i).$$

63. La *sucesión de Lucas* $L_1, L_2, \ldots$ (llamada así en honor de Édouard Lucas, inventor del juego de la Torre de Hanoi) se define por la relación de recurrencia

$$L_n = L_{n-1} + L_{n-2}, \qquad n \geq 3,$$

y las condiciones iniciales

$$L_1 = 1, \qquad L_2 = 3.$$

a) Encuentre los valores de L_3, L_4 y L_5.

b) Demuestre que

$$L_{n+2} = f_{n+1} + f_{n+3}, \qquad n \geq 1,$$

donde $f_1, f_2, \ldots$ denotan la sucesión de Fibonacci.

64. Establezca la relación de recurrencia

$$s_{n+1,k} = s_{n,k-1} + n s_{n,k}$$

para los números de Stirling del primer tipo (vea el ejercicio 81, sección 6.2).

65. Establezca la relación de recurrencia

$$S_{n+1,k} = S_{n,k-1} + k S_{n,k}$$

para los números de Stirling del segundo tipo (vea ejercicio 82, sección 6.2).

★66. Demuestre que

$$S_{n,k} = \frac{1}{k!} \sum_{i=0}^{k} (-1)^i (k - i)^n C(k, i),$$

donde $S_{n,k}$ denota un número de Stirling del segundo tipo (vea ejercicio 82, sección 6.2).

67. Suponga que una persona invierte una suma de dinero a r % de interés anual compuesto. Explique la regla general: Para estimar el tiempo para duplicar la inversión, divida 70 entre r.

68. Derive una relación de recurrencia para el número de multiplicaciones necesarias para evaluar un determinante de $n \times n$ por el método de cofactores.

Una permutación de sube/baja es una permutación p de 1, 2, . . . , n *que satisface*

$$p(i) < p(i + 1) \qquad \text{para } i = 1, 3, 5, \ldots$$

y

$$p(i) > p(i + 1) \qquad \text{para } i = 2, 4, 6, \ldots.$$

Por ejemplo, hay cinco permutaciones sube/baja de 1, 2, 3, 4:

1, 3, 2, 4; 1, 4, 2, 3; 2, 3, 1, 4;
2, 4, 1, 3; 3, 4, 1, 2.

Sea E_n el número de permutaciones sube/baja de 1, 2, . . . , n. *(Defina $= E_0 = 1$). Los números $E_0, E_1, E_2, \ldots$ se llaman números de Euler.*

69. Liste todas las permutaciones sube/baja de 1, 2, 3. ¿Cuál es el valor de E_3?

70. Liste todas las permutaciones sube/baja de 1, 2, 3, 4, 5. ¿Cuál es el valor de E_5?

71. Demuestre que en una permutación sube/baja de 1, 2, . . . , n, n debe ocurrir en la posición $2i$, para alguna i.

★72. Use el ejercicio 71 para derivar la relación de recurrencia

$$E_n = \sum_{j=1}^{\lfloor n/2 \rfloor} C(n - 1, 2j - 1) E_{2j-1} E_{n-2j}.$$

★**73.** Considerando dónde debe ocurrir 1 en una permutación sube/baja, derive la relación de recurrencia

$$E_n = \sum_{j=0}^{\lfloor (n-1)/2 \rfloor} C(n-1, 2j) E_{2j} E_{n-2j-1}.$$

★ **74.** Pruebe que

$$E_n = \frac{1}{2} \sum_{j=1}^{n-1} C(n-1, j) E_j E_{n-j-1}.$$

7.2 ➜ Solución de relaciones de recurrencia

WWW

Resolver una relación de recurrencia que implica una sucesión $a_0, a_1, \ldots$ significa encontrar una fórmula explícita para el término general a_n. En esta sección se estudian dos métodos para resolver relaciones de recurrencia: el de **iteraciones** y un método especial que se aplica a **relaciones de recurrencia homogéneas lineales con coeficientes constantes**. Métodos más poderosos, como los que utilizan las funciones generadoras, se encuentran en [Brualdi].

Para resolver una relación de recurrencia que implica la sucesión $a_0, a_1, \ldots$ por iteración, se usa la relación de recurrencia para escribir el n-ésimo término a_n en términos de algunos de sus predecesores $a_{n-1}, \ldots, a_0$. Después se usa la relación de recurrencia de manera sucesiva para sustituir cada uno de $a_{n-1}, \ldots$ por algunos de sus predecesores. El proceso continúa hasta obtener una fórmula explícita. El método iterativo se usó para resolver la relación de recurrencia del ejemplo 7.1.3.

Ejemplo 7.2.1 ▶

Es posible resolver la relación de recurrencia

$$a_n = a_{n-1} + 3, \tag{7.2.1}$$

sujeta a la condición inicial

$$a_1 = 2,$$

mediante iteraciones. Al sustituir n por $n-1$ en la ecuación (7.2.1), se obtiene

$$a_{n-1} = a_{n-2} + 3.$$

Si se sustituye esta expresión de a_{n-1} en la ecuación (7.2.1), se obtiene

$$\begin{aligned} a_n &= \boxed{a_{n-1}} \quad + 3 \\ &\downarrow \\ &= \boxed{a_{n-2} + 3} \quad + 3 \\ &= a_{n-2} + 2 \cdot 3. \end{aligned} \tag{7.2.2}$$

Sustituyendo n por $n-2$ en la ecuación (7.2.1), se obtiene

$$a_{n-2} = a_{n-3} + 3.$$

Si se sustituye esta expresión de a_{n-2} en (7.2.2), se llega a

$$\begin{aligned} a_n &= \boxed{a_{n-2}} \quad + 2 \cdot 3 \\ &\downarrow \\ &= \boxed{a_{n-3} + 3} \quad + 2 \cdot 3 \\ &= a_{n-3} + 3 \cdot 3. \end{aligned}$$

En general, se tiene

$$a_n = a_{n-k} + k \cdot 3.$$

Si se hace $k = n-1$ en esta última expresión, se tiene

$$a_n = a_1 + (n-1) \cdot 3.$$

Como $a_1 = 2$, se obtiene la fórmula explícita

$$a_n = 2 + 3(n - 1)$$

para la sucesión a. ◀

Ejemplo 7.2.2 ▶

Se puede resolver la relación de recurrencia

$$S_n = 2S_{n-1}$$

del ejemplo 7.1.5, sujeta a la condición inicial

$$S_0 = 1,$$

mediante iteraciones:

$$S_n = 2S_{n-1} = 2(2S_{n-2}) = \cdots = 2^n S_0 = 2^n.$$ ◀

Ejemplo 7.2.3 ▶

Crecimiento de población

Suponga que la población de venados en el condado Rustic es 1000 en el tiempo $n = 0$ y que el aumento entre el tiempo $n - 1$ y el tiempo n es un 10% del tamaño en el tiempo $n - 1$. Escriba una relación de recurrencia y una condición inicial que defina la población de venados en el tiempo n y después resuelva la relación de recurrencia.

Sea d_n la población de venados en el tiempo n. Se tiene la condición inicial

$$d_0 = 1000.$$

El incremento del tiempo $n - 1$ al tiempo n es $d_n - d_{n-1}$. Como este incremento es el 10% del tamaño en el tiempo $n - 1$, se obtiene la relación de recurrencia

$$d_n - d_{n-1} = 0.1d_{n-1},$$

que se escribe como

$$d_n = 1.1d_{n-1}.$$

Es posible resolver la relación de recurrencia mediante iteraciones:

$$\begin{aligned} d_n &= 1.1d_{n-1} = 1.1(1.1d_{n-2}) = (1.1)^2(d_{n-2}) \\ &= \cdots = (1.1)^n d_0 = (1.1)^n 1000. \end{aligned}$$

La suposiciones implican un crecimiento de población exponencial. ◀

Ejemplo 7.2.4 ▶

Encuentre una fórmula explícita para c_n, el número mínimo de movimientos en que se puede resolver el juego de la Torre de Hanoi con n discos (vea el ejemplo 7.1.8).

En el ejemplo 7.1.8 se obtuvo la relación de recurrencia

$$c_n = 2c_{n-1} + 1 \qquad \textbf{(7.2.3)}$$

y la condición inicial

$$c_1 = 1.$$

Aplicando el método iterativo a la ecuación (7.2.3), se obtiene

$$\begin{aligned} c_n &= 2c_{n-1} + 1 \\ &= 2(2c_{n-2} + 1) + 1 \\ &= 2^2 c_{n-2} + 2 + 1 \\ &= 2^2(2c_{n-3} + 1) + 2 + 1 \\ &= 2^3 c_{n-3} + 2^2 + 2 + 1 \end{aligned}$$

$$\begin{aligned}
&\vdots \\
&= 2^{n-1}c_1 + 2^{n-2} + 2^{n-3} + \cdots + 2 + 1 \\
&= 2^{n-1} + 2^{n-2} + 2^{n-3} + \cdots + 2 + 1 \\
&= 2^n - 1.
\end{aligned}$$

El último paso se obtiene de la fórmula para la suma geométrica (vea el ejemplo 1.7.4). ◀

Ejemplo 7.2.5 ▶

Se trata de resolver la relación de recurrencia

$$p_n = a - \frac{b}{k} p_{n-1}$$

para el precio p_n en el modelo económico del ejemplo 7.1.9 con iteraciones. Para simplificar la notación, se hace $s = -b/k$.

$$\begin{aligned}
p_n &= a + sp_{n-1} \\
&= a + s(a + sp_{n-2}) \\
&= a + as + s^2 p_{n-2} \\
&= a + as + s^2(a + sp_{n-3}) \\
&= a + as + as^2 + s^3 p_{n-3} \\
&\vdots \\
&= a + as + as^2 + \cdots + as^{n-1} + s^n p_0 \\
&= \frac{a - as^n}{1 - s} + s^n p_0 \\
&= s^n \left(\frac{-a}{1 - s} + p_0 \right) + \frac{a}{1 - s} \\
&= \left(-\frac{b}{k} \right)^n \left(\frac{-ak}{k + b} + p_0 \right) + \frac{ak}{k + b}. \qquad \textbf{(7.2.4)}
\end{aligned}$$

Se observa que si $b/k < 1$, el término

$$\left(-\frac{b}{k} \right)^n \left(\frac{-ak}{k + b} + p_0 \right)$$

se vuelve pequeño conforme n crece de manera que el precio tiende a estabilizarse aproximadamente en $ak/(k + b)$. Si $b/k = 1$, (7.2.4) muestra que p_n oscila entre p_0 y p_1. Si $b/k > 1$, (7.2.4) muestra que la diferencia entre precios sucesivos crece. Antes se observaron estas propiedades en una gráfica (vea el ejemplo 7.1.9). ◀

Se analizará una clase especial de relaciones de recurrencia.

Definición 7.2.6 ▶

Una *relación de recurrencia homogénea lineal de orden k con coeficientes constantes* es una relación de recurrencia de la forma

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}, \quad c_k \neq 0. \qquad \textbf{(7.2.5)}$$

◀

Observe que una relación de recurrencia homogénea lineal de orden k con coeficientes constantes (7.2.5) junto con k condiciones iniciales

$$a_0 = C_0, \quad a_1 = C_1, \ldots, \quad a_{k-1} = C_{k-1},$$

define de manera única una sucesión $a_0, a_1, \ldots$.

Ejemplo 7.2.7 ▶ Las relaciones de recurrencia

$$S_n = 2S_{n-1} \tag{7.2.6}$$

del ejemplo 7.2.2 y

$$f_n = f_{n-1} + f_{n-2}, \tag{7.2.7}$$

que define la sucesión de Fibonacci, son ambas relaciones de recurrencia homogéneas lineales con coeficientes constantes. La relación de recurrencia (7.2.6) es de orden 1 y la (7.2.7) es de orden 2. ◀

Ejemplo 7.2.8 ▶ La relación de recurrencia

$$a_n = 3a_{n-1}a_{n-2} \tag{7.2.8}$$

no es una relación de recurrencia homogénea lineal con coeficientes constantes. En una relación de recurrencia de este tipo, cada término es de la forma ca_k. Los términos como $a_{n-1}a_{n-2}$ no se permiten. Se dice que las relaciones de recurrencia como la (7.2.8) son *no lineales*. ◀

Ejemplo 7.2.9 ▶ La relación de recurrencia

$$a_n - a_{n-1} = 2n$$

no es una relación de recurrencia homogénea lineal con coeficientes constantes porque la expresión del lado derecho de la ecuación no es cero. (Se dice que este tipo de ecuaciones son *no homogéneas*. Las relaciones de recurrencia no homogéneas lineales con coeficientes constantes se analizan en los ejercicios 40 al 46). ◀

Ejemplo 7.2.10 ▶ La relación de recurrencia

$$a_n = 3na_{n-1}$$

no es una relación de recurrencia homogénea lineal con coeficientes constantes porque el coeficiente $3n$ no es constante. Se trata de una relación de recurrencia homogénea lineal con coeficientes no constantes. ◀

Se ilustrará el método general para resolver relaciones de recurrencia homogéneas lineales con coeficientes constantes encontrando una fórmula explícita para la sucesión definida por la relación de recurrencia

$$a_n = 5a_{n-1} - 6a_{n-2} \tag{7.2.9}$$

y condiciones iniciales

$$a_0 = 7, \qquad a_1 = 16. \tag{7.2.10}$$

Con frecuencia en matemáticas, al tratar de resolver un caso más difícil de algunos problemas, se comienza con una expresión que resuelve una versión más sencilla. Para la relación de recurrencia de primer orden (7.2.6), se encontró en el ejemplo 7.2.2 que la solución era de la forma

$$S_n = t^n;$$

entonces, para el primer intento de encontrar una solución de la relación de recurrencia de segundo orden (7.2.9), se investigará una solución de la forma $V_n = t^n$.

Si $V_n = t^n$ debe resolver (7.2.9), debe tenerse

$$V_n = 5V_{n-1} - 6V_{n-2}$$

o

$$t^n = 5t^{n-1} - 6t^{n-2}$$

o

$$t^n - 5t^{n-1} + 6t^{n-2} = 0.$$

Dividiendo entre t^{n-2}, se obtiene la ecuación equivalente

$$t^2 - 5t + 6 = 0. \tag{7.2.11}$$

Al resolver (7.2.11), se encuentran las soluciones

$$t = 2, \quad t = 3.$$

Hasta ahora, se tienen dos soluciones S y T de (7.2.9), dadas por

$$S_n = 2^n, \qquad T_n = 3^n. \tag{7.2.12}$$

Se puede verificar (vea el teorema 7.2.11) que si S y T son soluciones de (7.2.9), entonces $bS + dT$, donde b y d son números cualesquiera, también es una solución de (7.2.9). En nuestro caso, si se define una sucesión U mediante la ecuación

$$\begin{aligned} U_n &= bS_n + dT_n \\ &= b2^n + d3^n, \end{aligned}$$

U es una solución de (7.2.9).

Para satisfacer las condiciones iniciales (7.2.10), debe tenerse

$$7 = U_0 = b2^0 + d3^0 = b + d, \qquad 16 = U_1 = b2^1 + d3^1 = 2b + 3d.$$

Al despejar b y d de estas ecuaciones, se obtiene

$$b = 5, \qquad d = 2.$$

Por lo tanto, la sucesión U definida por

$$U_n = 5 \cdot 2^n + 2 \cdot 3^n$$

satisface la relación de recurrencia (7.2.9) y las condiciones iniciales (7.2.10). Se concluye que

$$a_n = U_n = 5 \cdot 2^n + 2 \cdot 3^n, \quad \text{para } n = 0, 1, \ldots.$$

Ahora se resumirán y justificarán las técnicas usadas para resolver la relación de recurrencia anterior.

Teorema 7.2.11

Sea

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} \tag{7.2.13}$$

una relación de recurrencia homogénea lineal de segundo orden con coeficientes constantes.

Si S y T son soluciones de (7.2.13), *entonces $U = bS + dT$ también es una solución de (7.2.13).*

Si r es una raíz de

$$t^2 - c_1 t - c_2 = 0, \tag{7.2.14}$$

entonces la sucesión r^n, $n = 0, 1, \ldots$, *es una solución de* (7.2.13).

Si a es la sucesión definida por (7.2.13),

$$a_0 = C_0, \qquad a_1 = C_1, \tag{7.2.15}$$

y r_1 *y* r_2 *son raíces de* (7.2.14) *con* $r_1 \neq r_2$, *entonces existen constantes b y d tales que*

$$a_n = br_1^n + dr_2^n, \qquad n = 0, 1, \ldots.$$

Demostración Como S y T son soluciones de (7.2.13),

$$S_n = c_1 S_{n-1} + c_2 S_{n-2}, \qquad T_n = c_1 T_{n-1} + c_2 T_{n-2}.$$

Si se multiplica la primera ecuación por b y la segunda por d y se suman, se obtiene

$$\begin{aligned} U_n = bS_n + dT_n &= c_1(bS_{n-1} + dT_{n-1}) + c_2(bS_{n-2} + dT_{n-2}) \\ &= c_1 U_{n-1} + c_2 U_{n-2}. \end{aligned}$$

Por lo tanto, U es una solución de (7.2.13).

Como r es una raíz de (7.2.14),

$$r^2 = c_1 r + c_2.$$

Ahora

$$c_1 r^{n-1} + c_2 r^{n-2} = r^{n-2}(c_1 r + c_2) = r^{n-2} r^2 = r^n;$$

así, la sucesión r^n, $n = 0, 1, \ldots$, es una solución de (7.2.13).

Si se establece $U_n = br_1^n + dr_2^n$, entonces U es una solución de (7.2.13). Para cumplir con las condiciones iniciales (7.2.15), debe tenerse

$$U_0 = b + d = C_0, \qquad U_1 = br_1 + dr_2 = C_1.$$

Si se multiplica la primera ecuación por r_1 y se resta, se obtiene

$$d(r_1 - r_2) = r_1 C_0 - C_1.$$

Como $r_1 - r_2 \neq 0$, se puede despejar d. De manera similar, se despeja b. Con estas opciones para b y d, se tiene

$$U_0 = C_0, \qquad U_1 = C_1.$$

Sea a la sucesión definida por (7.2.13) y (7.2.15). Como U satisface (7.2.13) y (7.2.15), se concluye que $U_n = a_n$, $n = 0, 1, \ldots$.

Ejemplo 7.2.12 ▶

Más crecimiento de población

Suponga que la población de venados del condado Rustic es 200 en el tiempo $n = 0$ y 220 en el tiempo $n = 1$, y que el incremento del tiempo $n - 1$ al tiempo n es dos veces el incremento del tiempo $n - 2$ al tiempo $n - 1$. Escriba la relación de recurrencia y una condición inicial que define la población de venados en el tiempo n y después resuelva la relación de recurrencia.

Sea d_n la población de venados en el tiempo n. Se tienen las condiciones iniciales

$$d_1 = 200, \qquad d_2 = 220.$$

El incremento del tiempo $n - 1$ al tiempo n es $d_n - d_{n-1}$, y el incremento del tiempo $n - 2$

al tiempo $n-1$ es $d_{n-1}-d_{n-2}$. Entonces se obtiene la relación de recurrencia

$$d_n - d_{n-1} = 2\left(d_{n-1} - d_{n-2}\right),$$

que se rescribe como

$$d_n = 3d_{n-1} - 2d_{n-2}.$$

Para resolver esta relación de recurrencia, primero se resuelve la ecuación cuadrática

$$t^2 - 3t + 2 = 0$$

para obtener las raíces 1 y 2. La sucesión d es de la forma

$$d_n = b \cdot 1^n + c \cdot 2^n = b + c2^n.$$

Para cumplir las condiciones iniciales, se requiere

$$200 = d_0 = b + c, \qquad 220 = d_1 = b + 2c.$$

Al despejar b y c, se encuentra que $b = 180$ y $c = 20$. Entonces d_n está dada por

$$d_n = 180 + 20 \cdot 2^n.$$

Como en el ejemplo 7.2.3, el crecimiento es exponencial. ◀

Ejemplo 7.2.13 ▶

Encuentre una fórmula explícita para la sucesión de Fibonacci.

La sucesión de Fibonacci está definida por la relación de recurrencia homogénea lineal de segundo orden

$$f_n - f_{n-1} - f_{n-2} = 0, \qquad n \geq 3,$$

y las condiciones iniciales

$$f_1 = 1, \qquad f_2 = 1.$$

Comenzamos por usar la fórmula cuadrática para resolver

$$t^2 - t - 1 = 0.$$

Las soluciones son

$$t = \frac{1 \pm \sqrt{5}}{2}.$$

Entonces, la solución es de la forma

$$f_n = b\left(\frac{1+\sqrt{5}}{2}\right)^n + d\left(\frac{1-\sqrt{5}}{2}\right)^n.$$

Para satisfacer las condiciones iniciales, debe tenerse

$$b\left(\frac{1+\sqrt{5}}{2}\right) + d\left(\frac{1-\sqrt{5}}{2}\right) = 1$$

$$b\left(\frac{1+\sqrt{5}}{2}\right)^2 + d\left(\frac{1-\sqrt{5}}{2}\right)^2 = 1.$$

Despejando b y d de estas ecuaciones se obtiene

$$b = \frac{1}{\sqrt{5}}, \qquad d = -\frac{1}{\sqrt{5}}.$$

Por lo tanto, una fórmula explícita para la sucesión de Fibonacci es

$$f_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n.$$

De manera sorprendente, aunque f_n es un entero, la fórmula anterior incluye el número irracional $\sqrt{5}$ ◀

El Teorema 7.2.11 establece que cualquier solución de (7.2.13) puede estar dada en términos de dos soluciones básicas r_1^n y r_2^n. Sin embargo, en caso de que la ecuación (7.2.14) tenga dos raíces iguales r, se obtiene sólo una solución básica r^n. El siguiente teorema muestra que en este caso, nr^n proporciona la otra solución básica.

Teorema 7.2.14

Sea

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} \tag{7.2.16}$$

una relación de recurrencia homogénea lineal de segundo orden con coeficientes constantes.

Sea a una sucesión que satisface (7.2.16) *y*

$$a_0 = C_0, \qquad a_1 = C_1.$$

Si ambas raíces de

$$t^2 - c_1 t - c_2 = 0 \tag{7.2.17}$$

son iguales a r, entonces existen constantes b y d tales que

$$a_n = br^n + dnr^n, \qquad n = 0, 1, \ldots.$$

Demostración La prueba del teorema 7.2.11 demuestra que la sucesión r^n, $n = 0, 1, \ldots$, es una solución de (7.2.16). Se demuestra que la sucesión nr^n, $n = 0, 1, \ldots$, también es una solución de (7.2.16).

Como r es la única solución de (7.2.17), debe tenerse

$$t^2 - c_1 t - c_2 = (t - r)^2.$$

Entonces,

$$c_1 = 2r, \qquad c_2 = -r^2.$$

Ahora

$$\begin{aligned} c_1\left[(n-1)r^{n-1}\right] + c_2\left[(n-2)r^{n-2}\right] &= 2r(n-1)r^{n-1} - r^2(n-2)r^{n-2} \\ &= r^n\left[2(n-1) - (n-2)\right] = nr^n. \end{aligned}$$

Por lo tanto, la sucesión nr^n, $n = 0, 1, \ldots$, es una solución de (7.2.16).

Por el Teorema 7.2.11, la sucesión U definida por $U_n = br^n + dnr^n$ es una solución de (7.2.16).

La demostración de que existen constantes b y d tales que $U_0 = C_0$ y $U_1 = C_1$ es similar al argumento dado en el Teorema 7.2.11 y se deja como ejercicio (ejercicio 48). Se concluye que $U_n = a_n$, $n = 0, 1, \ldots$.

Ejemplo 7.2.15 ▶

Resuelva la relación de recurrencia

$$d_n = 4(d_{n-1} - d_{n-2}) \tag{7.2.18}$$

sujeta a las condiciones iniciales

$$d_0 = 1 = d_1.$$

Según el teorema 7.2.11, $S_n = r^n$ es una solución de (7.2.18), donde r es una solución de

$$t^2 - 4t + 4 = 0. \tag{7.2.19}$$

Entonces, se obtiene la solución

$$S_n = 2^n$$

de (7.2.18). Como 2 es la única solución de (7.2.19), por el Teorema 7.2.14,

$$T_n = n2^n$$

también es una solución de (7.2.18). Así, la solución general de (7.2.18) es de la forma

$$U = aS + bT.$$

Debe tenerse

$$U_0 = 1 = U_1.$$

Estas últimas ecuaciones se convierten en

$$aS_0 + bT_0 = a + 0b = 1, \qquad aS_1 + bT_1 = 2a + 2b = 1.$$

Despejando a y b, se obtiene

$$a = 1, \qquad b = -\frac{1}{2}.$$

Por lo tanto, la solución de (7.2.18) es

$$d_n = 2^n - n2^{n-1}.$$ ◀

Para la relación de recurrencia homogénea lineal general de orden k con coeficientes constantes (7.2.5), si r es una raíz de

$$t^k - c_1 t^{k-1} - c_2 t^{k-2} - \cdots - c_k = 0$$

de multiplicidad m, es posible demostrar que

$$r^n, \quad nr^n, \quad \ldots, \quad n^{m-1}r^n$$

son soluciones de (7.2.5). Este hecho se puede usar, justo como en los ejemplos anteriores para relaciones de recurrencia de orden 2, para resolver una relación de recurrencia homogénea lineal de orden k con coeficientes constantes. Consulte el enunciado preciso y la demostración del resultado general en [Brualdi].

Sugerencias para resolver problemas

Para resolver una relación de recurrencia en la que a_n está definida en términos de su predecesor inmediato a_{n-1}

$$a_n = \ldots a_{n-1} \ldots,$$

use *iteraciones*. Comience con la ecuación original

$$a_n = \ldots a_{n-1} \ldots.$$

Sustituya a_{n-1}, lo que lleva a una expresión con a_{n-2}, para obtener

$$a_n = \ldots a_{n-2} \ldots .$$

Continúe hasta obtener a_k (por ejemplo, a_0), cuyo valor está dado de manera explícita como la condición inicial

$$a_n = \ldots a_k \ldots .$$

Sustituya el valor de a_k y habrá resuelto la relación de recurrencia.

Para resolver la relación de recurrencia

$$a_n = c_1 a_{n-1} + c_2 a_{n-2},$$

primero resuelva la ecuación

$$t^2 - c_1 t - c_2 = 0.$$

Si las raíces son r_1 y r_2, $r_1 \neq r_2$, la solución es

$$a_n = b r_1^n + d r_2^n$$

para algunas constantes b y d. Para determinar estas constantes, use las condiciones iniciales, por ejemplo $a_0 = C_0$, $a_1 = C_1$. Establezca $n = 0$ y después $n = 1$ en

$$a_n = b r_1^n + d r_2^n$$

para obtener

$$C_0 = a_0 = b + d$$
$$C_1 = a_1 = b r_1 + d r_2.$$

Ahora resuelva

$$C_0 = b + d$$
$$C_1 = b r_1 + d r_2$$

para que b y d den la solución de la relación de recurrencia.

Si las raíces de

$$t^2 - c_1 t - c_2 = 0$$

son r_1 y r_2, $r_1 = r_2$, las solución es

$$a_n = b r^n + d n r^n$$

para algunas constantes b y d, donde $r = r_1 = r_2$. Para determinar estas constantes, use las condiciones iniciales, digamos $a_0 = C_0$, $a_1 = C_1$. Establezca $n = 0$ y después $n = 1$ en

$$a_n = b r^n + d n r^n$$

para obtener

$$C_0 = a_0 = b$$
$$C_1 = a_1 = b r + d r.$$

Ahora resuelva

$$C_0 = b$$
$$C_1 = b r + d r$$

para b y d a fin de obtener la solución de la relación de recurrencia.

Sección de ejercicios de repaso

1. Explique cómo resolver una relación de recurrencia por iteraciones.
2. ¿Qué es una relación de recurrencia homogénea lineal de n-ésimo orden con coeficientes constantes?
3. Dé un ejemplo de una relación de recurrencia homogénea lineal de segundo orden con coeficientes constantes.
4. Explique cómo resolver una relación de recurrencia homogénea lineal de segundo orden con coeficientes constantes.

Ejercicios

Diga si cada relación de recurrencia en los ejercicios 1 al 10 es una relación de recurrencia homogénea lineal con coeficientes constantes. Dé el orden de cada relación de recurrencia.

1. $a_n = -3a_{n-1}$
2. $a_n = 2na_{n-1}$
3. $a_n = 2na_{n-2} - a_{n-1}$
4. $a_n = a_{n-1} + n$
5. $a_n = 7a_{n-2} - 6a_{n-3}$
6. $a_n = a_{n-1} + 1 + 2^{n-1}$
7. $a_n = (\lg 2n)a_{n-1} - [\lg(n-1)]a_{n-2}$
8. $a_n = 6a_{n-1} - 9a_{n-2}$
9. $a_n = -a_{n-1} - a_{n-2}$
10. $a_n = -a_{n-1} + 5a_{n-2} - 3a_{n-3}$

En los ejercicios 11 al 26, resuelva la relación de recurrencia indicada para las condiciones iniciales que se señalan.

11. Ejercicio 1; $a_0 = 2$
12. Ejercicio 2; $a_0 = 1$
13. Ejercicio 4; $a_0 = 0$
14. Resuelva la relación de recurrencia $a_n = 2^n a_{n-1}$, $n > 0$, con la condición inicial $a_0 = 1$.
15. $a_n = 6a_{n-1} - 8a_{n-2}$; $a_0 = 1$, $a_1 = 0$
16. $a_n = 7a_{n-1} - 10a_{n-2}$; $a_0 = 5$, $a_1 = 16$
17. $a_n = 2a_{n-1} + 8a_{n-2}$; $a_0 = 4$, $a_1 = 10$
18. $2a_n = 7a_{n-1} - 3a_{n-2}$; $a_0 = a_1 = 1$
19. Ejercicio 6; $a_0 = 0$
20. Ejercicio 8; $a_0 = a_1 = 1$
21. $a_n = -8a_{n-1} - 16a_{n-2}$; $a_0 = 2$, $a_1 = -20$
22. $9a_n = 6a_{n-1} - a_{n-2}$; $a_0 = 6$, $a_1 = 5$
23. La sucesión de Lucas

$$L_n = L_{n-1} + L_{n-2}, \quad n \geq 3; \qquad L_1 = 1, \quad L_2 = 3$$

24. Ejercicio 50, sección 7.1
25. Ejercicio 52, sección 7.1
26. La relación de recurrencia que precede al ejercicio 53, sección 7.1
27. La población de *Utopía* aumenta en un 5% por año. En 2000 la población era 10,000. ¿Cuál era la población en 1970?
28. Suponga que la población de venados en el condado Rustic es 0 en el tiempo $n = 0$. Suponga que en el tiempo n se introducen $100n$ venados en sus bosques y que la población aumenta un 20% cada año. Escriba una relación de recurrencia y una condición inicial que defina la población de venados en el tiempo n y después resuelva la relación de recurrencia. La fórmula siguiente resultará útil:

$$\sum_{i=1}^{n-1} ix^{i-1} = \frac{(n-1)x^n - nx^{n-1} + 1}{(x-1)^2}.$$

Los ejercicios 29 al 33 se refieren a Tito y Samuel, que lanzan monedas no cargadas. Si las monedas son ambas cara o ambas cruz, Tito gana. Si una moneda tiene cara y la otra cruz, Samuel gana. Tito comienza con T monedas y Samuel comienza con S monedas.

29. Sea p_n la probabilidad de que Tito gane todas las monedas de Samuel si Tito comienza con n monedas. Escriba la relación de recurrencia para p_n.
30. ¿Cuál es el valor de p_0?
31. ¿Cuál es el valor de p_{S+T}?
32. Resuelva la relación de recurrencia del ejercicio 29.
33. Encuentre la probabilidad de que Tito gane todas las monedas de Samuel.

Algunas veces, una relación de recurrencia que no es una ecuación lineal homogénea con coeficientes constantes se puede transformar en una ecuación de este tipo. En los ejercicios 34 y 35, haga las sustituciones dadas y resuelva la relación de recurrencia que resulta, después encuentre la solución de la relación de recurrencia original.

34. Resuelva la relación de recurrencia

$$\sqrt{a_n} = \sqrt{a_{n-1}} + 2\sqrt{a_{n-2}}$$

con condiciones iniciales $a_0 = a_1 = 1$ haciendo la sustitución $b_n = \sqrt{a_n}$.

35. Resuelva la relación de recurrencia

$$a_n = \sqrt{\frac{a_{n-2}}{a_{n-1}}}$$

con condiciones iniciales $a_0 = 8$, $a_1 = 1/(2\sqrt{2})$ tomando logaritmos en ambos lados y haciendo la sustitución $b_n = \lg a_n$.

En los ejercicios 36 al 38, resuelva la relación de recurrencia para las condiciones iniciales señaladas.

★36. $a_n = -2na_{n-1} + 3n(n-1)a_{n-2}$; $a_0 = 1$, $a_1 = 2$

★37. $c_n = 2 + \sum_{i=1}^{n-1} c_i$, $n \geq 2$; $c_1 = 1$

38. $A(n, m) = 1 + A(n-1, m-1) + A(n-1, m)$, $n - 1 \geq m \geq 1$, $n \geq 2$; $A(n, 0) = A(n, n) = 1$, $n \geq 0$

39. Demuestre que

$$f_{n+1} \geq \left(\frac{1+\sqrt{5}}{2}\right)^{n-1}, \qquad n \geq 1,$$

donde f denota la sucesión de Fibonacci.

40. La ecuación

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + f(n) \qquad \textbf{(7.2.20)}$$

se llama **relación de recurrencia lineal no homogénea de segundo orden con coeficientes constantes**.

Sea $g(n)$ una solución de (7.2.20). Demuestre que cualquier solución U de (7.2.20) es de la forma

$$U_n = V_n + g(n), \qquad \textbf{(7.2.21)}$$

donde V es una solución de la ecuación homogénea (7.2.13).

Si $f(n) = C$ en (7.2.20), se puede demostrar que $g(n) = C'$ en (7.2.21) si 1 no es una raíz de

$$t^2 - c_1 t - c_2 = 0, \qquad \textbf{(7.2.22)}$$

$g(n) = C'n$ si 1 es una raíz de (7.2.22) de multiplicidad uno, y $g(n) = C'n^2$ si 1 es una raíz de (7.2.22) de multiplicidad dos. De manera similar, si $f(n) = Cn$, se puede demostrar que $g(n) = C'_1 n + C'_0$ si 1 no es una raíz de (7.2.22), $g(n) = C'_1 n^2 + C'_0 n$ si 1 es una raíz de multiplicidad uno, y $g(n) = C'_1 n^3 + C'_0 n^2$ si 1 es una raíz de multiplicidad dos. Si $f(n) = Cn^2$, se puede demostrar que $g(n) = C'_2 n^2 + C'_1 n + C'_0$ si 1 no es una raíz de (7.2.22), $g(n) = C'_2 n^3 + C'_1 n^2 + C'_0 n$ si 1 es una raíz de multiplicidad uno y $g(n) = C'_2 n^4 + C'_1 n^3 + C'_0 n^2$ si 1 es una raíz de multiplicidad dos. Si $f(n) = C^n$, se puede demostrar que $g(n) = C'C^n$ si C no es una raíz de (7.2.22), $g(n) = C'nC^n$ si C es una raíz de multiplicidad uno, y $g(n) = C'n^2C^n$ si C es una raíz de multiplicidad dos. Las constantes se pueden determinar sustituyendo $g(n)$ en la relación de recurrencia e igualando los coeficientes de los dos lados de la ecuación resultante. Como ejemplos, los términos constantes en los dos lados de la ecuación deben ser iguales, y el coeficiente de n en el lado izquierdo de la ecuación debe ser igual al coeficiente de n en el lado derecho de la ecuación. A partir de estos hechos y del ejercicio 40, encuentre las soluciones generales de las relaciones de recurrencia de los ejercicios 41 al 46.

41. $a_n = 6a_{n-1} - 8a_{n-2} + 3$

42. $a_n = 7a_{n-1} - 10a_{n-2} + 16n$

43. $a_n = 2a_{n-1} + 8a_{n-2} + 81n^2$

44. $2a_n = 7a_{n-1} - 3a_{n-2} + 2^n$

45. $a_n = -8a_{n-1} - 16a_{n-2} + 3n$

46. $9a_n = 6a_{n-1} - a_{n-2} + 5n^2$

47. La ecuación

$$a_n = f(n)a_{n-1} + g(n)a_{n-2} \qquad \textbf{(7.2.23)}$$

se llama **relación de recurrencia lineal homogénea de segundo orden**. Los coeficientes $f(n)$ y $g(n)$ no son necesariamente constantes. Demuestre que si S y T son soluciones de (7.2.23), entonces $bS + dT$ también es una solución de (7.2.23).

48. Suponga que las dos raíces de

$$t^2 - c_1 t - c^2 = 0$$

son iguales a r, y suponga que a_n satisface

$$a_n = c_1 a_{n-1} + c_2 a_{n-2}, \qquad a_0 = C_0, \qquad a_1 = C_1.$$

Demuestre que existen constantes b y d tales que

$$a_n = br^n + dnr^n, \qquad n = 0, 1, \ldots,$$

para completar así la prueba del Teorema 7.2.14.

49. Sea a_n el número mínimo de enlaces requeridos para resolver el problema de comunicación de n nodos (vea el ejercicio 48, sección 7.1). Use iteraciones para demostrar que $a_n \leq 2n - 4$, $n \geq 4$.

El juego de la Torre de Hanoi con cuatro estacas y n discos tiene las mismas reglas que el de tres estacas; la única diferencia es que se tiene una estaca adicional. Los ejercicios 50 al 53 se refieren al siguiente algoritmo para resolver el juego de la Torre de Hanoi con cuatro estacas y n discos.

Suponga que las estacas están numeradas 1, 2, 3, 4 y que el problema es mover los discos, que inicialmente están apilados en la estaca 1, a la estaca 4. Si $n = 1$, se mueve el disco a la estaca 4 y se detiene. Si $n > 1$, sea k_n el entero más grande que satisface

$$\sum_{i=1}^{k_n} i \leq n.$$

Se fijan k_n discos hasta abajo de la estaca 1. Se invoca este algoritmo en forma recursiva para mover los $n - k_n$ discos hasta arriba de la estaca 1 a la estaca 2. Durante esta parte del algoritmo, los k_n discos de abajo en la estaca 1 permanecen fijos. Después se mueven los k_n discos de la estaca 1 a la estaca 4 invocando el algoritmo óptimo de tres estacas (vea el ejemplo 7.1.8) y usando sólo las estacas 1, 3 y 4. Por último, de nuevo en forma recursiva, se invoca este algoritmo para mover los $n - k_n$ discos de la estaca 2 a la 4. Durante esta parte del algoritmo, los k_n discos en la estaca 4 están fijos. Sea $T(n)$ el número de movimientos requeridos por este algoritmo.

Este algoritmo, aunque no se sabe que sea óptimo, usa tan pocos movimientos como cualquier otro algoritmo que se haya propuesto para el problema de cuatro estacas.

50. Derive la relación de recurrencia

$$T(n) = 2T(n - k_n) + 2^{k_n} - 1.$$

51. Calcule $T(n)$ para $n = 1, \ldots, 10$. Compare estos valores con el número óptimo de movimientos para resolver el problema de tres estacas.

★52. Sea

$$r_n = n - \frac{k_n(k_n + 1)}{2}.$$

Usando inducción o de otra manera, pruebe que

$$T(n) = (k_n + r_n - 1)2^{k_n} + 1.$$

★53. Demuestre que $T(n) = O(4^{\sqrt{n}})$.

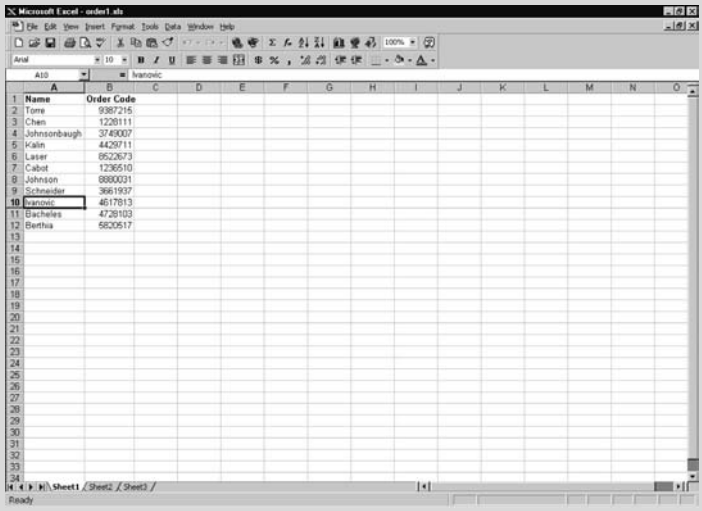
★54. Dé un algoritmo óptimo para resolver la variante de la Torre de Hanoi en donde se permite una pila de discos, excepto por la pila inicial y final, siempre que el disco más grande esté hasta abajo (los discos arriba del disco más grande en la pila pueden tener el orden que sea). El problema es transferir los discos a otra estaca moviendo un disco a la vez comenzando con todos los discos apilados en una estaca en orden del más grande (abajo) al más pequeño como en el juego original. La posición final será la misma que en el juego original: los discos se apilan en orden del más grande (abajo) al más pequeño. Pruebe que su algoritmo es óptimo. Este problema se debe a John McCarthy.

Rincón de solución de problemas

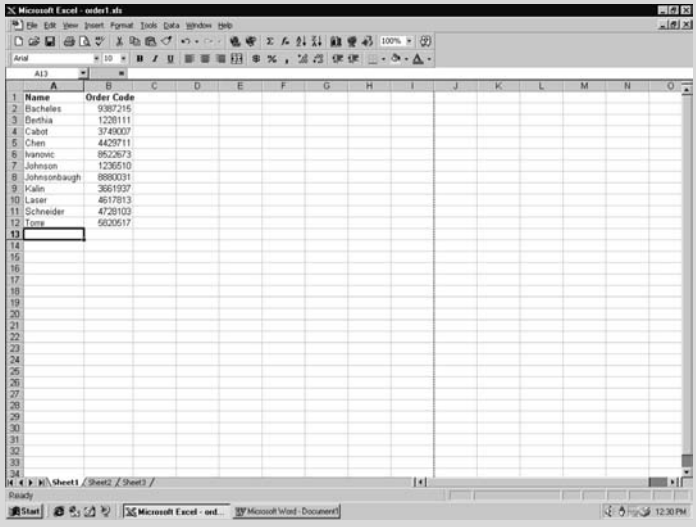
Relaciones de recurrencia

Problema

a) Varias personas ordenan discos compactos. Sus nombres, junto con los códigos de su orden, se introducen en una hoja de cálculo:



Los registros deben organizarse en orden alfabético, pero ocurre un error cuando se ordenan los *nombres* y los códigos de las órdenes no se cambian.



Como resultado, nadie recibe los discos compactos correctos. Sea D_n el número de maneras que n personas pueden todas recibir las órdenes equivocadas. Demuestre que la sucesión $D_1, D_2, \ldots$ satisface la relación de recurrencia

$$D_n = (n-1)(D_{n-1} + D_{n-2}).$$

b) Resuelva la relación de recurrencia del inciso *a)* haciendo la sustitución $C_n = D_n - nD_{n-1}$.

Cómo atacar el problema

Antes de atacar el problema, considere lo que se requiere para el inciso *a)*. Para probar la relación de recurrencia, debemos reducir el problema de n personas a los problemas de $(n-1)$ y $(n-2)$ personas (ya que la fórmula de D_n está dada en términos de D_{n-1} y D_{n-2}). Entonces, si se observan sistemáticamente algunos ejemplos, debe notarse la forma en que el caso de n personas se relaciona con los casos de $n-1$ y $n-2$ personas. La situación es similar a la de la inducción matemática y los algoritmos recursivos en donde un caso dado de un problema se relaciona con casos más pequeños del mismo problema.

Se darán algunos ejemplos. El caso más pequeño es $n = 1$. Una sola persona debe obtener el orden correcto, de manera que $D_1 = 0$. Para $n = 2$, hay una manera de que todos obtengan la orden equivocada: la persona 1 obtiene la orden 2 y la persona 2, la orden 1. Así, $D_2 = 1$. Antes de continuar se desarrollará cierta notación para la distribución de órdenes. La notación elegida con cuidado ayudará a resolver el problema.

Se escribe

$$c_1, c_2, \ldots, c_n$$

para indicar que la persona 1 recibió la orden c_1, la persona 2 obtuvo la orden c_2, y así sucesivamente. La manera en que dos personas reciben las órdenes equivocadas se denota por 2,1.

Si $n = 3$, la persona 1 obtiene la orden 2 o la 3, por lo que las posibilidades son 2,?,? y 3,?,?. Se buscarán los números que faltan. Suponga que la persona 1 recibe la orden 2. La persona 2 no puede recibir la orden 1 (porque la persona 3 obtendría la orden correcta); entonces, la persona 2 recibe la orden 3. Esto deja la orden 1 para la persona 3. Entonces, si la persona 1 obtiene la orden 2, la única posibilidad es

$$2, 3, 1.$$

Suponga que la persona 1 recibe la orden 3. La persona 3 no puede recibir la orden 1 (porque la persona 2 obtendría la orden correcta); entonces la persona 3 recibe la orden 2. Esto deja la orden 1 para la persona 2. Entonces, si la persona 1 recibe la orden 3, la única posibilidad es

$$3, 1, 2.$$

Así, $D_3 = 2$.

Se verifica que la relación de recurrencia se cumple para $n = 3$:

$$D_3 = 2 = 2(1 + 0) = (3 - 1)(D_2 + D_1).$$

Si $n = 4$, la persona 1 obtiene la orden 2, 3 o 4, y las posibilidades son 2,?,?,?; 3,?,?,? y 4,?,?,?. (Si hay n personas, la persona 1 recibe la orden 2 o 3 o . . . o $n - 1$. Estas $n - 1$ posibilidades justifican el factor $n - 1$ en la relación de recurrencia). Se buscarán los números que faltan. Suponga que la persona 1 obtiene la orden 2. Si la persona 2 recibe la orden 1, entonces la 3 recibe la orden 4 y la persona 4 recibe la orden 3, lo que da 2,1,3,4. Si la persona 2 no recibe la orden 1, las posibilidades son 2,3,4,1 y 2,4,1,3. Entonces, si la persona 1 recibe la orden 2, hay tres posibilidades. De manera similar, si la persona 1 recibe la orden 3, hay tres posibilidades y si le llega la orden 4 hay tres posibilidades. Hay que listar las posibilidades para confirmar esta última afirmación. Entonces $D_4 = 9$.

Se verifica que la relación de recurrencia también se cumple para $n = 4$:

$$D_4 = 9 = 3(2 + 1) = (4 - 1)(D_3 + D_2).$$

Antes de seguir leyendo, trabaje el caso $n = 5$. Liste sólo las posibilidades cuando la persona 1 recibe la orden 2. (Existen demasiadas posibilidades para listarlas en su totalidad.) También verifique la relación de recurrencia para $n = 5$.

Observe que si la persona 1 recibe la orden 2 y la persona 2 recibe la orden 1, el número de maneras para que las otras personas reciban las órdenes equivocadas es D_{n-2} (las $n - 2$ personas restantes deben todas tener las órdenes equivocadas). Esto justifica la presencia de D_{n-2} en la relación de recurrencia. Se tendrá una solución siempre que el término D_{n-1} aparezca en el caso que queda: la persona 1 recibe la orden 2, pero la persona 2 no recibe la orden 1.

Cómo encontrar una solución

Suponga que hay n personas. Se resumirá el argumento desarrollado mediante los ejemplos. La persona 1 recibe la orden 2 o 3, . . . , o n; por lo que hay $n - 1$ maneras posibles de que la persona 1 reciba la orden incorrecta. Suponga que la persona 1 recibe la orden 2. Hay dos posibilidades: la persona 2 recibe la orden 1 y la persona 2 no recibe la orden 1. Si la persona 2 obtiene la orden 1, el número de maneras para que las otras personas tengan las órdenes equivocadas es D_{n-2}. El caso que queda es que la persona 2 no recibe la orden 1.

Se escribe con cuidado lo que se tiene que contar. Las personas 2, 3, . . . , n tienen entre ellas las órdenes 1, 3, 4, . . . , n (la orden 2 no está porque la tiene la persona 1). Se quiere encontrar el número de maneras de que cada persona 2, 3, . . . , n obtenga la orden incorrecta *y* la persona 2 no obtenga la orden 1. Éste casi es el problema de que $n - 1$ personas reciban todas órdenes equivocadas. Se convierte en este problema si se dice a las personas 2, 3, . . . , n que la orden 1 es la orden 2. Ahora la persona 2 no obtendrá la orden 1 porque piensa que es la orden 2. Como hay $n - 1$ personas, hay D_{n-1} maneras de que las personas 2, 3, . . . , n reciban todas órdenes equivocadas *y* la persona 2 no reciba la orden 1. Se concluye que hay $D_{n-1} + D_{n-2}$ maneras de que la persona 1 reciba la orden 2 y las otras personas reciban órdenes equivocadas. Como hay $n - 1$ maneras posibles de que la persona 1 obtenga la orden equivocada, el resultado es la relación de recurrencia.

La relación de recurrencia define D_n en términos de D_{n-1} y D_{n-2} por lo que no se puede resolver usando iteraciones. Además, la relación de recurrencia no tiene coeficientes constantes (aunque es lineal) de manera que no se puede resolver usando el teorema 7.2.11 o el 7.2.14. Esto explica la necesidad de hacer la sustitución en el inciso *b)*. Es evidente que después de hacer la sustitución, la relación de recurrencia para C_n se resuelve usando los métodos de la sección 7.2.

Si se expande

$$D_n = (n - 1)(D_{n-1} + D_{n-2}),$$

se obtiene

$$D_n = nD_{n-1} - D_{n-1} + (n - 1)D_{n-2}.$$

Si después se mueve nD_{n-1} al lado izquierdo de la ecuación (para obtener una expresión igual a C_n), se obtiene

$$D_n - nD_{n-1} = -D_{n-1} + (n - 1)D_{n-2}.$$

Ahora el lado izquierdo de la ecuación es igual a C_n y el lado derecho es igual a $-C_{n-1}$. Entonces se obtiene la relación de recurrencia

$$C_n = -C_{n-1}.$$

Esta ecuación se resuelve con iteraciones.

Solución formal

a) Suponga que n personas tienen las órdenes incorrectas. Se considera la orden que tiene una persona p. Suponga que p tiene la orden de q. Se consideran dos casos: q tiene la orden de p y q no tiene la orden de p.

Hay D_{n-2} distribuciones en las que q tiene la orden de p ya que las $n-2$ órdenes restantes están en posesión de las $n-2$ personas restantes, pero cada una con la orden equivocada.

Se demostrará que hay D_{n-1} distribuciones en las que q no tiene la orden de p. Note que el conjunto S de órdenes que poseen las $n-1$ personas (excluyendo a p) incluye todas menos la orden de q (que tiene p). Temporalmente asigne la propiedad de la orden de p a q. Entonces cualquier distribución de S entre las $n-1$ personas en la que nadie tiene su propia orden lleva a una distribución en la que q no tiene la orden que en realidad es de p. Como hay D_{n-1} distribuciones de éstas, existen D_{n-1} distribuciones donde q no tiene la orden de p.

Se concluye que hay $D_{n-1} + D_{n-2}$ distribuciones donde p tiene la orden de q. Como p puede tener cualquiera de las $n-1$ órdenes, la relación de recurrencia se comprueba.

b) Al efectuar la sustitución dada, se obtiene

$$C_n = -C_{n-1}.$$

Usando iteraciones, se obtiene

$$\begin{aligned} C_n &= (-1)^1 C_{n-1} \\ &= (-1)^2 C_{n-2} = \cdots \\ &= (-1)^{n-2} C_2 \\ &= (-1)^n C_2 \\ &= (-1)^n (D_2 - 2D_1) = (-1)^n. \end{aligned}$$

Por lo tanto,

$$D_n - nD_{n-1} = (-1)^n.$$

Al resolver esta última relación de recurrencia mediante iteraciones, se tiene

$$\begin{aligned} D_n &= (-1)^n + nD_{n-1} \\ &= (-1)^n + n[(-1)^{n-1} + (n-1)D_{n-2}] \\ &= (-1)^n + n(-1)^{n-1} \\ &\quad + n(n-1)[(-1)^{n-2} + (n-2)D_{n-3}] \\ &\vdots \\ &= (-1)^n + n(-1)^{n-1} + n(n-1)(-1)^{n-2} + \cdots \\ &\quad - [n(n-1)\cdots 4] + [n(n-1)\cdots 3]. \end{aligned}$$

Resumen de las técnicas de solución de problemas

- Cuando los ejemplos se vuelven poco concisos, desarrolle una notación para describirlos con claridad. Si se elige con cuidado, la notación será de gran ayuda en la solución del problema.
- Cuando estudie los ejemplos, intente encontrar una relación entre el problema y casos más pequeños del mismo problema.
- Con frecuencia ayuda escribir con detalle lo que se quiere contar.
- Algunas veces es posible convertir una relación de recurrencia que no es lineal homogénea con coeficientes constantes en una ecuación con tales características. Una relación de recurrencia así se resuelve por el método de la sección 7.2.

Comentario

El nombre técnico de una permutación en la que ningún elemento está en su posición original es **perturbación**.

Probabilidad

Se puede calcular la probabilidad de que nadie reciba la orden correcta si se supone que todas las permutaciones son igualmente probables. En este caso, como hay $n!$ permutaciones, la probabilidad de que nadie reciba la orden correcta es

$$\begin{aligned} \frac{D_n}{n!} &= \frac{1}{n!}\{(-1)^n + n(-1)^{n-1} \\ &\quad + n(n-1)(-1)^{n-2} + \cdots \\ &\quad - [n(n-1)\cdots 4] + [n(n-1)\cdots 3]\} \\ &= \frac{(-1)^n}{n!} + \frac{(-1)^{n-1}}{(n-1)!} + \frac{(-1)^{n-2}}{(n-2)!} + \cdots \\ &\quad - \frac{1}{3!} + \frac{1}{2!}. \end{aligned}$$

Cuando n aumenta, los términos adicionales son tan pequeños que la probabilidad cambia muy poco. En otras palabras, para n suficientemente grande, la probabilidad de que nadie reciba la orden correcta es, en esencia, ¡insensible al número de órdenes!

El cálculo nos dice que

$$e^x + \sum_{i=0}^{\infty} \frac{x^i}{i!}.$$

En particular,

$$e^{-1} = \sum_{i=0}^{\infty} \frac{(-1)^i}{i!} = \frac{1}{2!} - \frac{1}{3!} + \frac{1}{4!} \cdots.$$

Entonces, para n grande, la probabilidad de que nadie reciba la orden correcta es aproximadamente $e^{-1} = 0.368$.

7.3 ➜ Aplicaciones al análisis de algoritmos

En esta sección se usan las relaciones de recurrencia para analizar el tiempo que requieren los algoritmos. La técnica consiste en desarrollar una relación de recurrencia y condiciones iniciales que definan una sucesión $a_1, a_2, \ldots$, donde a_n es el tiempo (en el mejor caso, caso promedio o peor caso) requerido por un algoritmo para ejecutar una entrada de tamaño n. Al resolver la relación de recurrencia, se determina el tiempo que requiere el algoritmo.

Nuestro primer algoritmo es una versión del algoritmo de orden de selección. Este algoritmo selecciona el artículo más grande y lo coloca al último, después repite de manera recursiva este proceso.

Algoritmo 7.3.1

WWW

Orden de selección

Este algoritmo ordena la sucesión

$$s_1, \quad s_2, \quad \ldots, \quad s_n$$

en orden no decreciente, eligiendo primero el artículo más grande y colocándolo al último y después ordenando de manera recursiva los elementos restantes.

Entrada: $s_1, s_2, \ldots, s_n$ y la longitud n de la sucesión
Salida: $s_1, s_2, \ldots, s_n$, arreglada en orden no decreciente

```
1.  orden_selección(s, n) {
2.      // caso base
3.      if (n == 1)
4.          return
5.      // encuentra el mayor
6.      índice_máx = 1 //suponga al inicio que s_1 es el mayor
7.      for i = 2 to n
8.          if (s_i > s_índice_máx)//se encontró el más grande, actualizar
9.              índice_máx = i
10.     // mover mayor al final
11.     cambiar (s_n, s_índice_máx)
12.     orden_selección(s, n − 1)
13. }
```

Como una medida del tiempo requerido por este algoritmo, se cuenta el número de comparaciones b_n en la línea 8 requeridas para ordenar n artículos. (Observe que los tiempos del mejor caso, el caso promedio y el peor caso son todos iguales para este algoritmo). De inmediato se obtiene la condición inicial

$$b_1 = 0.$$

Para obtener una relación de recurrencia para la sucesión $b_1, b_2, \ldots$, se simula la ejecución del algoritmo para un tamaño $n > 1$ arbitrario de entrada. Se cuenta al número de comparaciones en cada línea y después se suman estos números para obtener el número total de comparaciones b_n. En las líneas 1 al 7, hay cero comparaciones (del tipo que se está contando). En la línea 8, hay $n - 1$ comparaciones (ya que la línea 7 provoca que la línea 8 se ejecute n − 1 veces). Hay cero comparaciones en las líneas 9 a la 11. La llamada recursiva ocurre en la línea 12, donde se invoca este algoritmo con entrada de tamaño $n - 1$. Pero, por definición, ese algoritmo requiere b_{n-1} comparaciones para una entrada de tamaño $n - 1$. Entonces, hay b_{n-1} comparaciones en la línea 12. Por lo tanto, el número total de comparaciones es

$$b_n = n - 1 + b_{n-1},$$

que lleva a la relación de recurrencia deseada.

Nuestra relación de recurrencia se puede resolver con iteraciones:

$$\begin{aligned} b_n &= b_{n-1} + n - 1 \\ &= (b_{n-2} + n - 2) + (n - 1) \\ &= (b_{n-3} + n - 3) + (n - 2) + (n - 1) \\ &\vdots \\ &= b_1 + 1 + 2 + \cdots + (n - 2) + (n - 1) \\ &= 0 + 1 + 2 + \cdots + (n - 1) = \frac{(n-1)n}{2} = \Theta(n^2). \end{aligned}$$

Así, el tiempo requerido por el algoritmo 7.3.1 es $\Theta(n^2)$.

El siguiente algoritmo (el 7.3.2) es una **búsqueda binaria**. La búsqueda binaria busca un valor en una sucesión *ordenada* y regresa el índice del valor si lo encontró o 0 si no lo encontró. El algoritmo usa el enfoque de "divide y vencerás". La sucesión se divide en dos partes casi iguales (línea 4). Si el artículo está en el punto que divide (línea 5), el algoritmo termina. Si el artículo no se encuentra porque la sucesión está ordenada, una comparación adicional (línea 7) localizará la mitad de la sucesión en la que aparece al artículo, si está presente. Después se invoca de manera recursiva la búsqueda binaria (línea 11) para continuar la búsqueda.

Algoritmo 7.3.2

Búsqueda binaria

Este algoritmo busca un valor en una sucesión no decreciente y regresa el índice del valor si se encuentra o 0 si no se encuentra.

Entrada: Una sucesión $s_i, s_{i+1}, \ldots, s_j$, $i \geq 1$, arreglada en orden no decreciente, valor *clave*, i y j.

Salida: La salida es un índice k para el que $s_k = clave$, o si la *clave* no está en la sucesión, la salida es el valor 0.

```
1.  búsqueda_binaria(s, i, j, clave) {
2.      if(i > j) // no se encontró
3.          return 0
4.      k = ⌊(i + j)/2⌋
5.      if (clave == s_k) // se encontró
6.          return k
7.      if (clave < s_k) // busca en mitad izquierda
8.          j = k − 1
9.      else // busca en mitad derecha
10.         i = k + 1
11.     return búsqueda_binaria(s, i, j, clave)
12. }
```

Ejemplo 7.3.3 ▶

Se ilustra el algoritmo 7.3.2 para la entrada

$$s_1 = \text{'}B\text{'}, \qquad s_2 = \text{'}D\text{'}, \qquad s_3 = \text{'}F\text{'}, \qquad s_4 = \text{'}S\text{'},$$

y *clave* = 'S'. En la línea 2, como $i > j$ $(1 > 4)$ es falso, se procede a la línea 4 donde se establece $k = 2$. En la línea 5, como *clave*('S') no es igual a s_2 ('D'), se procede a la línea 7. En la línea 7, *clave* $< s_k$ ('S' < 'D') es falso, entonces en la línea 10 se establece $i = 3$. Después se invoca este algoritmo con $i = 3$, $j = 4$ para buscar *clave* en

$$s_3 = \text{'}F\text{'}, \qquad s_4 = \text{'}S\text{'}.$$

En la línea 2, como $i > j$ $(3 > 4)$ es falso, se precede a la línea 4, donde se establece $k = 3$. En la línea 5, como *clave* ('S') no es igual a s_3 ('F'), se procede a la línea 7. En la línea 7, *clave* $< s_k$ ('S' < 'F') es falso, de manera que en la línea 10, se establece $i = 4$. Después se invoca este algoritmo con $i = j = 4$ para buscar *clave* en

$$s_4 = \text{'}S\text{'}.$$

En la línea 2, como $i > j$ (4 > 4) es falso, se procede a la línea 4, donde se establece $k = 4$. En la línea 5, como *clave* ('*S*') es igual a s_4 ('*S*'), se regresa 4, el índice de *clave* en la sucesión *s*. ◀

Ahora se analizará el peor caso de la búsqueda binaria. Se define el tiempo requerido por la búsqueda binaria para el peor caso como el número de veces que se invoca el algoritmo en el peor caso para una sucesión que contiene *n* elementos. Sea a_n el tiempo para el peor caso.

Suponga que *n* es 1; es decir, suponga que la sucesión consiste en un elemento s_i, y que $i = j$. En el peor caso, no se encontrará el elemento en la línea 5, de manera que el algoritmo se invoca por segunda vez en la línea 11. Sin embargo, la segunda vez se tiene $i > j$ y el algoritmo se detiene sin éxito en la línea 3. Se ha demostrado que si *n* es 1, el algoritmo se invoca dos veces. Se obtiene la condición inicial

$$a_1 = 2. \tag{7.3.1}$$

Ahora suponga que $n > 1$. En este caso, $i < j$, y la condición en la línea 2 es falsa. En el peor caso, el elemento no se encontrará en la línea 5, y el algoritmo será invocado en la línea 11. Por definición, la línea 11 requerirá un total de a_m invocaciones, donde *m* es el tamaño de la sucesión que se alimenta a la línea 11. Como los tamaños de los lados izquierdo y derecho de la sucesión original son $\lfloor (n-1)/2 \rfloor$ y $\lfloor n/2 \rfloor$ y el peor caso ocurre con la sucesión más grande, el número total de invocaciones en la línea 11 será $a_{\lfloor n/2 \rfloor}$. La invocación original junto con las invocaciones en la línea 11 dan el total de invocaciones; así, se obtiene la relación de recurrencia

$$a_n = 1 + a_{\lfloor n/2 \rfloor}. \tag{7.3.2}$$

La relación de recurrencia (7.3.2) es típica de los algoritmos que resultan del enfoque "divide y vencerás". En general, estas relaciones de recurrencia no se resuelven con facilidad de manera explícita (sin embargo, vea el ejercicio 6). En su lugar, se estima el crecimiento de la sucesión usando la notación theta. Nuestro método para derivar una notación theta para sucesiones definidas por (7.3.1) y (7.3.2) ilustra un método general para manejar este tipo de relaciones de recurrencia. Primero se *resuelve explícitamente* (7.3.2) en caso de que *n* sea una potencia de 2. Si *n* no es una potencia de 2, *n* está entre dos potencias de 2, digamos 2^{k-1} y 2^k, y a_n está entre $a_{2^{k-1}}$ y a_{2^k}. Como se conoce las fórmulas explícitas para $a_{2^{k-1}}$ y a_{2^k}, se puede *estimar* a_n y con ello derivar una notación teta para a_n.

Primero se resuelve la relación de recurrencia (7.3.2) en caso de que *n* sea una potencia de 2. Si $n = 2^k$, (7.3.2) se convierte en

$$a_{2^k} = 1 + a_{2^{k-1}}, \qquad k = 1, 2, \ldots.$$

Si $b_k = a_{2^k}$, se obtiene la relación de recurrencia

$$b_k = 1 + b_{k-1}, \qquad k = 1, 2, \ldots, \tag{7.3.3}$$

y la condición inicial

$$b_0 = 2.$$

La relación de recurrencia (7.3.3) se resuelve por el método iterativo:

$$b_k = 1 + b_{k-1} = 2 + b_{k-2} = \cdots = k + b_0 = k + 2.$$

Entonces, si $n = 2^k$,

$$a_n = 2 + \lg n. \tag{7.3.4}$$

Un valor arbitrario de *n* cae entre dos potencias de 2, digamos,

$$2^{k-1} < n \leq 2^k. \tag{7.3.5}$$

Como la sucesión *a* es no decreciente (un hecho que se puede *probar* mediante inducción; vea el ejercicio 5),

$$a_{2^{k-1}} \leq a_n \leq a_{2^k}. \tag{7.3.6}$$

Observe que (7.3.5) da

$$k - 1 < \lg n \leq k. \tag{7.3.7}$$

A partir de (7.3.4), (7.3.6) y (7.3.7) se deduce que

$$\lg n < 1 + k = a_{2^{k-1}} \leq a_n \leq a_{2^k} = 2 + k < 3 + \lg n = O(\lg n).$$

Por lo tanto, $a_n = \Theta(\lg n)$, de manera que la búsqueda binaria es $\Theta(\lg n)$ en el peor caso. Este resultado es suficientemente importante para destacarlo como un teorema.

Teorema 7.3.4

El tiempo en el peor caso para una búsqueda binaria para una entrada de tamaño n es $\Theta(\lg n)$.

Demostración La demostración precede al enunciado del teorema.

En el último ejemplo se presentó y analizó otro algoritmo de ordenamiento conocido como **merge sort** (algoritmo 7.3.8). Se mostrará que el merge sort tiene un tiempo de corrida en el peor caso de $\Theta(n \lg n)$, de manera que para entradas grandes es más rápido que el de orden de selección (algoritmo 7.3.1), que tiene un tiempo de corrida en el peor caso de $\Theta(n^2)$. En la sección 9.7 se demostrará que el tiempo de *cualquier* algoritmo para ordenar que compara elementos y, según el resultado de la comparación, mueve elementos dentro del arreglo es $\Omega(n \lg n)$ en el peor caso; así que el algoritmo de merge sort resulta óptimo en esta clase de algoritmos de ordenamiento.

En el merge sort la sucesión que se va a ordenar,

$$s_i, \ldots, s_j,$$

se divide en dos sucesiones casi iguales,

$$s_i, \ldots, s_m, \qquad s_{m+1}, \ldots, s_j,$$

donde $m = \lfloor (i + j)/2 \rfloor$. Cada una de estas sucesiones se ordena de manera recursiva, y después se integran para producir un arreglo ordenado de la sucesión original. El proceso de integrar dos sucesiones ordenadas se llama **mezclado (merging)**.

Algoritmo 7.3.5

Fusión (mezcla) de dos sucesiones

Este algoritmo combina dos sucesiones no decrecientes en una sola sucesión no decreciente.

Entrada: Dos sucesiones no decrecientes: $s_i, \ldots, s_m$ y $s_{m+1}, \ldots, s_j$ y los índices i, m y j
Salida: La sucesión $c_i, \ldots, c_j$ que consiste en los elementos $s_i, \ldots, s_m$ y $s_{m+1}, \ldots, s_j$ mezclados en una sucesión no decreciente

```
1.  fusionar(s, i, m, j, c) {
2.      // p es la posición en la sucesión s_i, ..., s_m
3.      // q es la posición en la sucesión s_{m+1}, ..., s_j
4.      // r es la posición en la sucesión c_i, ..., c_j
5.      p = i
6.      q = m + 1
7.      r = i
8.      // se copia el menor entre s_p y s_q
9.      while (p ≤ m ∧ q ≤ j) {
10.         if (s_p < s_q){
11.             c_r = s_p
12.             p = p + 1
13.         }
14.         else {
15.             c_r = s_q
```

```
16.            q = q + 1
17.         }
18.         r = r + 1
19.      }
20.      // se copia el resto de la primera sucesión
21.      while (p ≤ m) {
22.  `      c_r = s_p
23.         p = p + 1
24.         r = r + 1
25.      }
26.      // se copia el resto de la sucesión
27.      while (q ≤ j) {
28.         c_r = s_q
29.         q = q + 1
30.         r = r + 1
31.      }
32.   }
```

Ejemplo 7.3.6 ▶

La figura 7.3.1 muestra la forma en que el algoritmo 7.3.5 mezcla las sucesiones

1, 3, 4 y 2, 4, 5, 6.

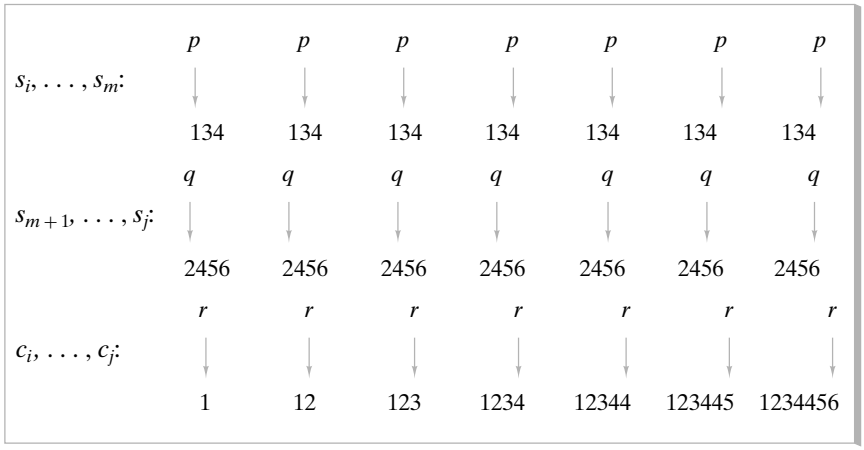


Figura 7.3.1 Mezcla de $s_i, \ldots, s_m$ y $s_{m+1}, \ldots, s_j$. El resultado es $c_i, \ldots, c_j$. ◀

El Teorema 7.3.7 muestra que, en el peor caso, se necesitan $n - 1$ comparaciones para fusionar dos sucesiones cuyas longitudes suman n.

Teorema 7.3.7

En el peor caso, el algoritmo 7.3.5 requiere $j - i$ comparaciones. En particular, en el peor caso, se necesitan $n - 1$ comparaciones para fusionar dos sucesiones cuyas longitudes suman n.

Demostración En el algoritmo 7.3.5, la comparación de los elementos de las sucesiones ocurre en el ciclo "while" en la línea 10. El ciclo se ejecuta mientras $p \leq m$ y $q \leq j$. Entonces, en el peor caso, el algoritmo 7.3.5 requiere $j - i$ comparaciones.

Ahora se usará el algoritmo 7.3.5 (mezcla) para construir el merge sort.

Algoritmo 7.3.8

WWW

Merge sort

Este algoritmo recursivo ordena una sucesión en orden no decreciente usando el algoritmo 7.3.5, que fusiona dos sucesiones no decrecientes.

Entrada: $s_i, \ldots, s_j$, i y j
Salida: $s_i, \ldots, s_j$ arreglada en orden no decreciente

```
1.  merge_sort(s, i, j) {
2.      // caso base: i == j
3.      if (i == j)
4.          return
5.      // divide la sucesión y ordena
6.      m = ⌊(i + j)/2⌋
7.      merge_sort(s, i, m)
8.      merge_sort(s, m + 1, j)
9.      // mezcla
10.     mezcla(s, i, m, j, c)
11.     // copiar c, la salida de mezcla, a s
12.     for k = i to j
13.         s_k = c_k
14. }
```

Ejemplo 7.3.9 ▶

La figura 7.3.2 muestra la forma en que el algoritmo 7.3.8 ordena la sucesión

12, 30, 21, 8, 6, 9, 1, 7.

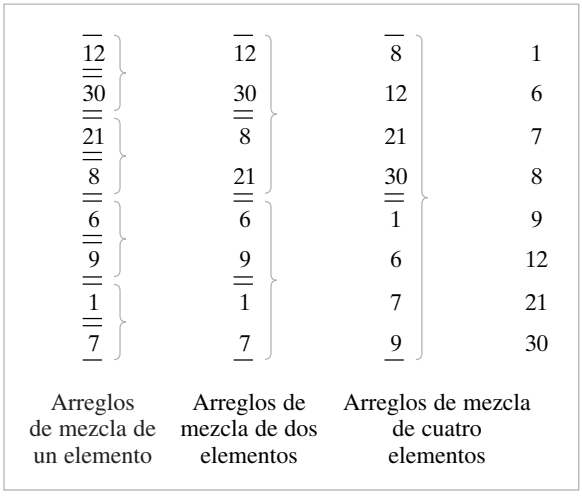


Figura 7.3.2 Ordenamiento por merge sort. ◀

Se concluye que el merge sort (algoritmo 7.3.8) es $\Theta(n \lg n)$ en el peor caso. El método de prueba es el mismo que se usó para demostrar que en la búsqueda binaria es $\Theta(\lg n)$ en el peor caso.

Teorema 7.3.10

El merge sort (algoritmo 7.3.8) es $\Theta(n \lg n)$ en el peor caso.

Demostración Sea a_n el número de comparaciones requeridas por el algoritmo 7.3.8 para ordenar n artículos en el peor caso. Entonces $a_1 = 0$. Si $n > 1$, a_n es cuando mucho la suma del número de comparaciones en el peor caso que resultan de las llamadas de

recurrencia en las líneas 7 y 8, y el número de comparaciones en el peor caso requeridas por la mezcla en la línea 10. Esto es,

$$a_n \leq a_{\lfloor n/2 \rfloor} + a_{\lfloor (n+1)/2 \rfloor} + n - 1.$$

De hecho, esta cota superior se logra (vea el ejercicio 12), de manera que

$$a_n = a_{\lfloor n/2 \rfloor} + a_{\lfloor (n+1)/2 \rfloor} + n - 1.$$

Primero se resuelve la relación de recurrencia anterior en caso de que n sea una potencia de 2, digamos $n = 2^k$. La ecuación se convierte en

$$a_{2^k} = 2a_{2^{k-1}} + 2^k - 1.$$

Esta última ecuación se resuelve usando iteraciones (vea la sección 7.2):

$$\begin{aligned}
a_{2^k} &= 2a_{2^{k-1}} + 2^k - 1 \\
&= 2[2a_{2^{k-2}} + 2^{k-1} - 1] + 2^k - 1 \\
&= 2^2 a_{2^{k-2}} + 2 \cdot 2^k - 1 - 2 \\
&= 2^2[2a_{2^{k-3}} + 2^{k-2} - 1] + 2 \cdot 2^k - 1 - 2 \\
&= 2^3 a_{2^{k-3}} + 3 \cdot 2^k - 1 - 2 - 2^2 \\
&\vdots \\
&= 2^k a_{2^0} + k \cdot 2^k - 1 - 2 - 2^2 - \cdots - 2^{k-1} \\
&= k \cdot 2^k - (2^k - 1) \\
&= (k-1)2^k + 1. \qquad \textbf{(7.3.8)}
\end{aligned}$$

Un valor arbitrario de n cae entre dos potencias de 2, como,

$$2^{k-1} < n \leq 2^k. \qquad \textbf{(7.3.9)}$$

Como la sucesión a es no decreciente (vea el ejercicio 15),

$$a_{2^{k-1}} \leq a_n \leq a_{2^k}. \qquad \textbf{(7.3.10)}$$

Observe que (7.3.9) da

$$k - 1 < \lg n \leq k. \qquad \textbf{(7.3.11)}$$

A partir de (7.3.8), (7.3.10) y (7.3.11), se deduce que

$$\begin{aligned}
\Omega(n \lg n) = (-2 + \lg n)\frac{n}{2} &< (k-2)2^{k-1} + 1 = a_{2^{k-1}} \\
&\leq a_n \leq a_{2^k} \leq k2^k + 1 \leq (1 + \lg n)2n + 1 = O(n \lg n).
\end{aligned}$$

Por lo tanto, $a_n = \Theta(n \lg n)$, por lo que el merge sort es $\Theta(n \lg n)$ en el peor caso.

Como se hizo notar, en la sección 9.7 se demostrará que una comparación basada en el algoritmo para ordenar es $\Omega(n \lg n)$ en el peor caso. Este resultado implica, en particular, que el merge sort es $\Omega(n \lg n)$ en el peor caso. Si ya se hubiera probado este resultado, para probar que el merge sort es $\Theta(n \lg n)$ en el peor caso, sería suficiente con probar que el merge sort es $O(n \lg n)$ en el peor caso.

Aun cuando el merge sort, algoritmo 7.3.8, es óptimo, tal vez no sea el algoritmo preferido para un problema de ordenamiento específico. Deben tomarse en cuenta factores como el tiempo en el caso promedio del algoritmo, el número de artículos a ordenar, la memoria disponible, las estructuras de datos que se usarán, si los artículos a ordenar están en memoria o residen en un dispositivo de almacenamiento periférico como disco o cintas, si los artículos a ordenar están "casi" ordenados y el hardware que se usará.

Sección de ejercicios de repaso

1. Explique cómo encontrar una relación de recurrencia que describa el tiempo que requiere un algoritmo recursivo.
2. ¿Cómo funciona el orden de selección?
3. ¿Cuál es el tiempo requerido por el orden de selección?
4. ¿Cómo funciona la búsqueda binaria? ¿Qué propiedades debe tener la entrada?
5. Escriba una relación de recurrencia que describa el tiempo requerido por la búsqueda binaria en el peor caso.
6. ¿Cuál es tiempo en el peor caso para la búsqueda binaria?
7. ¿Cómo funciona el algoritmo de fusión? ¿Qué propiedades debe tener la entrada?
8. ¿Cuál es el tiempo en el peor caso para el algoritmo de mezcla?
9. Explique cómo funciona el merge sort.
10. Escriba una relación de recurrencia que describa el tiempo requerido por el algoritmo de merge sort en el peor caso.
11. ¿Por qué es fácil despejar el tiempo en el peor caso del merge sort si el tamaño de la entrada es una potencia de 2?
12. Explique en palabras cómo se pueden obtener cotas sobre el tiempo en el peor caso del merge sort para una entrada de tamaño arbitrario, si se conoce el tiempo en el peor caso cuando el tamaño de la entrada es una potencia de 2.
13. ¿Cuál es el tiempo del merge sort en el peor caso?

Ejercicios

Los ejercicios 1 al 4 se refieren a la sucesión

$$s_1 = \text{'}C\text{'}, \quad s_2 = \text{'}G\text{'}, \quad s_3 = \text{'}J\text{'}, \quad s_4 = \text{'}M\text{'}, \quad s_5 = \text{'}X\text{'}.$$

1. Muestre cómo se ejecuta el algoritmo 7.3.2 en el caso de *clave* = 'G'.
2. Muestre cómo se ejecuta el algoritmo 7.3.2 en el caso de *clave* = 'P'.
3. Muestre cómo se ejecuta el algoritmo 7.3.2 en el caso de *clave* = 'C'.
4. Muestre cómo se ejecuta el algoritmo 7.3.2 en el caso de *clave* = 'Z'.
5. Sea a_n el tiempo en el peor caso de la búsqueda binaria (algoritmo 7.3.2). Pruebe que $a_n \leq a_{n+1}$ para $n \geq 1$.
6. ★ Pruebe que si a_n es el número de veces que se invoca el algoritmo de búsqueda binaria (algoritmo 7.3.2) en el peor caso para una sucesión que contiene n elementos, entonces para todo entero positivo n.

$$a_n = 2 + \lfloor \lg n \rfloor$$

7. El profesor T. R. S. Eighty propone la siguiente versión de la búsqueda binaria:

```
búsqueda_binaria2(s, i, j, clave) {
    if (i > j)
        return 0
    k = ⌊(i + j)/2⌋
    if (clave == s_k)
        return k
    k1 = búsqueda_binaria2(s, i, k − 1, clave)
    k2 = búsqueda_binaria2(s, k + 1, j, clave)
    return k1 + k2
}
```

   a) Demuestre que *búsqueda_binaria2* es correcto; es decir, si *clave* está presente, el algoritmo regresa su índice, pero si *clave* no está presente, regresa 0.

   b) Encuentre el tiempo de corrida en el peor caso de *búsqueda_binaria2*.

8. Suponga que el algoritmo A requiere $\lceil n \lg n \rceil$ comparaciones para ordenar n artículos y que el algoritmo B requiere $\lceil n^2/4 \rceil$ comparaciones para ordenar n artículos. ¿Para cuál n el algoritmo B es superior a A?
9. Muestre cómo el merge sort (algoritmo 7.3.8) ordena la sucesión 1, 9, 7, 3.
10. Muestre cómo el merge sort (algoritmo 7.3.8) ordena la sucesión 2, 3, 7, 2, 8, 9, 7, 5, 4.
11. Suponga que se tienen dos sucesiones, cada una de tamaño n, ordenadas en orden no decreciente.

    a) ¿En qué condiciones ocurre el máximo número de comparaciones en el algoritmo 7.3.5?

    b) ¿En qué condiciones ocurre el mínimo número de comparaciones en el algoritmo 7.3.5?

12. Defina a_n igual que en la demostración del teorema 7.3.10. Describa una entrada para la que

$$a_n = a_{\lfloor n/2 \rfloor} + a_{\lfloor (n+1)/2 \rfloor} + n - 1.$$

13. ¿Cuál es el número mínimo de comparaciones requeridas por el algoritmo 7.3.8 para ordenar una arreglo de tamaño 6?
14. ¿Cuál es el número máximo de comparaciones requeridas por el algoritmo 7.3.8 para ordenar un arreglo de tamaño 6?
15. Defina a_n igual que en la demostración del teorema 7.3.10. Demuestre que $a_n \leq a_{n+1}$ para toda $n \geq 1$.
16. Sea a_n el número de comparaciones requeridas por el merge sort en el peor caso. Demuestre que $a_n \leq 3n \lg n$ para $n = 1, 2, 3, \ldots$.
17. Demuestre que en el mejor caso, el merge sort requiere $\Theta(n \lg n)$ comparaciones.

Los ejercicios 18 al 22 se refieren al algoritmo 7.3.11.

Algoritmo 7.3.11

Cálculo de una exponencial

Este algoritmo calcula a^n de manera recursiva, donde a es un número real y n es un entero positivo

Entrada: a (número real), n (entero positivo)

Salida: a^n

```
1.  exp1(a, n) {
2.      if (n == 1)
3.          return a
4.      m = ⌊n/2⌋
5.      return exp1(a, m)*exp1(a, n − m)
6.  }
```

Sea b_n el número de multiplicaciones (línea 5) requeridas para calcular a^n.

18. Explique la manera en que el algoritmo 7.3.11 calcula a^n.

19. Encuentre la relación de recurrencia y las condiciones iniciales para la sucesión $\{b_n\}$.

20. Calcule b_2, b_3 y b_4.

21. Resuelva la relación de recurrencia del ejercicio 19 para el caso en el que n es una potencia de 2.

22. Pruebe que $b_n = n - 1$ para todo entero positivo n.

Los ejercicios 23 al 28 se refieren al algoritmo 7.3.12.

Algoritmo 7.3.12

Cálculo de una exponencial

Este algoritmo calcula a^n de forma recursiva, donde a es un número real y n es un entero positivo.

Entrada: a (número real), n (entero positivo)

Salida: a^n

```
1.  exp2(a, n) {
2.      if (n == 1)
3.          return a
4.      m = ⌊n/2⌋
5.      potencia = exp2(a, m)
6.      potencia = potencia * potencia
7.      if (n es par)
8.          return potencia
9.      else
10.         return potencia * a
11. }
```

Sea b_n el número de multiplicaciones (líneas 6 y 10) requeridas para calcular a^n.

23. Explique la forma en que el algoritmo 7.3.12 calcula a^n.

24. Demuestre que

$$b_n = \begin{cases} b_{(n-1)/2} + 2 & \text{si } n \text{ es impar} \\ b_{n/2} + 1 & \text{si } n \text{ es par.} \end{cases}$$

25. Encuentre b_1, b_2, b_3 y b_4.

26. Resuelva la relación de recurrencia del ejercicio 24 para el caso en el que n es una potencia de 2.

27. Muestre, con un ejemplo, que b es no decreciente.

★ **28.** Pruebe que $b_n = \Theta(\lg n)$.

Los ejercicios 29 al 34 se refieren al algoritmo 7.3.13.

Algoritmo 7.3.13

Encontrar los elementos más pequeño y más grande en una sucesión

Este algoritmo recursivo encuentra los elementos menor y mayor en una sucesión.

Entrada: $s_i, \ldots, s_j$, i y j

Salida: *mayor* (el elemento más grande en la sucesión), *menor* (el elemento más pequeño en la sucesión)

```
1.  mayor_menor (s, i, j, mayor, menor) {
2.      if (i == j) {
3.          mayor = s_i
4.          menor = s_i
5.          return
6.      }
7.      m = ⌊(i + j)/2⌋
8.      mayor_menor (s, i, m, mayor_izq, menor_izq)
9.      mayor_menor (s, m + 1, j, mayor_der, menor_der)
10.     if (mayor_izq > mayor_der)
11.         mayor = mayor_izq
12.     else
13.         mayor = mayor_der
14.     if (menor_izq > menor_der)
15.         menor = menor_der
16.     else
17.         menor = menor_izq
18. }
```

Sea b_n el número de comparaciones (líneas 10 y 14) requeridas para una entrada de tamaño n.

29. Explique la forma en que el algoritmo 7.3.13 encuentra los elementos mayor y menor.

30. Demuestre que $b_1 = 0$ y $b_2 = 2$.

31. Encuentre b_3.

32. Establezca la relación de recurrencia

$$b_n = b_{\lfloor n/2 \rfloor} + b_{\lfloor (n+1)/2 \rfloor} + 2 \qquad \textbf{(7.3.12)}$$

para $n > 1$.

33. Resuelva la relación de recurrencia (7.3.12) para el caso en el que n es una potencia de 2, para obtener

$$b_n = 2n - 2, \qquad n = 1, 2, 4, \ldots.$$

34. Use inducción matemática para demostrar que

$$b_n = 2n - 2$$

para todo entero positivo n.

Los ejercicios 35 al 38 se refieren al algoritmo 7.3.13, con los siguientes renglones insertados después de la línea 6.

```
6a.  if (j == i + 1) {
6b.      if (s_i > s_j) {
6c.          mayor = s_i
6d.          menor = s_j
6e.      }
6f.      else {
6g.          menor = s_i
6h.          mayor = s_j
6i.      }
6j.      return
6k.  }
```

Sea b_n el número de comparaciones (líneas 6b, 10 y 14) para una entrada de tamaño n.

35. Demuestre que $b_1 = 0$ y $b_2 = 1$.

36. Calcule b_3 y b_4.

37. Demuestre que la relación de recurrencia (7.3.12) se cumple para $n > 2$.

38. Resuelva la relación de recurrencia (7.3.12) para el caso en el que n es una potencia de 2 para obtener

$$b_n = \frac{3n}{2} - 2, \qquad n = 2, 4, 8, \ldots.$$

★***39.*** Modifique el algoritmo 7.3.13 insertando las líneas que preceden al ejercicio 35 después de la línea 6 y sustituyendo la línea 7 con lo siguiente:

```
7a.  if (j − i es impar ∧ (1 + j − i)/2 es impar)
7b.          m = ⌊(i + j)/2⌋ - 1
7c.  else
7d.          m = ⌊(i + j)/2⌋
```

Demuestre que en el peor caso, este algoritmo modificado requiere cuando mucho $\lceil (3n/2) - 2 \rceil$ comparaciones para encontrar los elementos mayor y menor en una arreglo de tamaño n.

Los ejercicios 40 al 44 se refieren al algoritmo 7.3.14.

Algoritmo 7.3.14

Orden de inserción (versión recursiva)

Este algoritmo ordena la sucesión

$$s_1, \quad s_2, \quad \ldots, \quad s_n$$

en orden no decreciente ordenando de manera recursiva los primeros $n - 1$ elementos y después insertando s_n en la posición correcta.

Entrada: $s_1, s_2, \ldots, s_n$ y la longitud n de la sucesión
Salida: $s_1, s_2, \ldots, s_n$ arreglados en orden no decreciente

```
1.  orden_inserción(s, n) {
2.      if (n == 1)
3.          return
4.      orden_inserción(s, n − 1)
5.      i = n − 1
6.      temp = s_n
7.      while (i ≥ 1 ∧ s_i > temp) {
8.          s_{i+1} = s_i
9.          i = i − 1
10.     }
11.     s_{i+1} = temp
12. }
```

Sea b_n el número de veces que se hace la comparación $s_i > temp$ en la línea 7 en el peor caso. Suponga que si $i < 1$, esa comparación no se hace.

40. Explique cómo ordena la sucesión el algoritmo 7.3.14.

41. ¿Qué entrada produce el comportamiento del peor caso para el algoritmo 7.3.14?

42. Encuentre b_1, b_2 y b_3.

43. Encuentre una relación de recurrencia para la sucesión $\{b_n\}$.

44. Resuelva la relación de recurrencia del ejercicio 43.

Los ejercicios 45 al 47 se refieren al algoritmo 7.3.15.

Algoritmo 7.3.15

Entrada: $s_1, \ldots, s_n, n$
Salida: $s_1, \ldots, s_n$

```
algorl(s, n) {
    i = n
    while(i ≥ 1) {
        s_i = s_i + 1
        i = ⌊i/2⌋
    }
    n = ⌊n/2⌋
    if(n ≥ 1)
        algorl(s, n)
}
```

Sea b_n el número de veces que se ejecuta la instrucción $s_i = s_i + 1$.

45. Encuentre una relación de recurrencia para la sucesión $\{b_n\}$ y calcule b_1, b_2 y b_3.

46. Resuelva la relación de recurrencia del ejercicio 45 para el caso en el que n es una potencia de 2.

47. Pruebe que $b_n = \Theta((\lg n)^2)$.

48. Encuentre una notación teta en términos de n para el número de veces que se llama a *algor2* cuando se invoca como *algor2*$(1, n)$.

```
algor2(i, j) {
    if (i == j)
        return
    k = ⌊(i + j)/2⌋
    algor2(i, k)
    algor2(k + 1, j)
}
```

Los ejercicios 49 al 53 se refieren al algoritmo 7.3.16.

Algoritmo 7.3.16

Entrada: Una sucesión $s_i, \ldots, s_j$ de ceros y unos
Salida: $s_i, \ldots, s_j$ donde todos los ceros preceden a todos los unos

```
ordenar(s, i, j) {
    if (i == j)
        return
    if (s_i == 1){
        cambiar(s_i, s_j)
        ordenar(s, i, j − 1)
    }
    else
        ordenar(s, i + 1, j)
}
```

49. Use inducción matemática sobre n, el número de artículos en la sucesión de entrada, para probar que *ordenar* de hecho produce como salida un versión rearreglada de la sucesión de entrada en la que todos los ceros preceden a todos los unos. (El paso base es $n = 1$).

Sea b_n el número de veces que se llama a ordenar cuando la sucesión de entrada contiene n elementos.

50. Encuentre b_1, b_2 y b_3.

51. Escriba una relación de recurrencia para b_n.

52. Despeje b_n de la relación de recurrencia del ejercicio 51.

53. En la notación teta, ¿cuál es el tiempo de corrida de *ordenar* como una función de n, el número de elementos en la sucesión de entrada?

54. Resuelva la relación de recurrencia

$$a_n = 3a_{\lfloor n/2 \rfloor} + n, \qquad n > 1,$$

para el caso en el que n es una potencia de 2. Suponga que $a_1 = 1$.

55. Demuestre que $a_n = \Theta(n^{\lg 3})$, donde a_n se define como en el ejercicio 54.

Los ejercicios 56 al 63 se refieren a un algoritmo que acepta como entrada la sucesión

$$s_i, \ldots, s_j.$$

Si $j > i$, los subproblemas

$$s_i, \ldots, s_{\lfloor (i+j)/2 \rfloor} \qquad \text{y} \qquad s_{\lfloor (i+j)/2+1 \rfloor}, \ldots, s_j$$

se resuelven de manera recursiva. Las soluciones a subproblemas de tamaño m y k pueden combinarse en el tiempo $c_{m,k}$ para resolver el problema original. Sea b_n el tiempo requerido por el algoritmo para una entrada de tamaño n.

56. Escriba una relación de recurrencia para b_n suponiendo que $c_{m,k} = 3$.

57. Escriba una relación de recurrencia para b_n suponiendo que $c_{m,k} = m + k$.

58. Resuelva la relación de recurrencia del ejercicio 56 para el caso en el que n es una potencia de 2, suponiendo que $b_1 = 0$.

59. Resuelva la relación de recurrencia del ejercicio 56 para el caso en el que n es una potencia de 2, suponiendo que $b_1 = 1$.

60. Resuelva la relación de recurrencia del ejercicio 57 para el caso en el que n es una potencia de 2, suponiendo que $b_1 = 0$.

61. Resuelva la relación de recurrencia del ejercicio 57 para el caso en el que n es una potencia de 2, suponiendo que $b_1 = 1$.

★ ***62.*** Suponga que si $m_1 \geq m_2$ y $k_1 \geq k_2$, entonces $c_{m_1,k_1} \geq c_{m_2,k_2}$. Demuestre que la sucesión $b_1, b_2, \ldots$ es no decreciente.

★ **63.** Suponiendo que $c_{m,k} = m + k$ y $b_1 = 0$, demuestre que $b_n \leq 4n \lg n$.

Los ejercicios 64 al 69 se refieren a la siguiente situación. Sea P_n un problema específico de tamaño n. Si P_n se divide en subproblemas de tamaños i y j, existe un algoritmo que combina las soluciones de estos subproblemas en una solución de P_n en un tiempo de cuando mucho $2 + \lg(ij)$. Suponga que ya se resolvió un problema de tamaño 1.

64. Escriba una relación de recurrencia para obtener P_n similar a la del algoritmo 7.3.8.

65. Sea a_n el tiempo en el peor caso para obtener P_n con el algoritmo del ejercicio 64. Demuestre que

$$a_n \leq a_{\lfloor n/2 \rfloor} + a_{\lfloor (n+1)/2 \rfloor} + 2 \lg n.$$

66. Sea b_n la relación de recurrencia obtenida en el ejercicio 65 sustituyendo "$\leq$" por "$=$". Suponga que $b_1 = a_1 = 0$. Demuestre que si n es una potencia de 2,

$$b_n = 4n - 2 \lg n - 4.$$

67. Demuestre que $a_n \leq b_n$ para $n = 1, 2, 3, \ldots$.

68. Demuestre que $b_n \leq b_{n+1}$ para $n = 1, 2, 3, \ldots$.

69. Demuestre que $a_n \leq 8n$ para $n = 1, 2, 3, \ldots$.

70. Suponga que $\{a_n\}$ es una sucesión no decreciente y que siempre que m divide a n,

$$a_n = a_{n/m} + d,$$

donde d es un número real positivo y m es un entero que satisface $m > 1$. Demuestre que $a_n = \Theta(\lg n)$.

★ **71.** Suponga que $\{a_n\}$ es una sucesión no decreciente y que siempre que m divide a n,

$$a_n = ca_{n/m} + d,$$

donde c y d son números reales que satisfacen $c > 1$ y $d > 0$, y m es un entero que satisface $m > 1$. Demuestre que $a_n = \Theta(n^{\log_m c})$.

72. [Proyecto] Investigue otros algoritmos para ordenar. Considere específicamente la complejidad, los estudios empíricos y las características especiales de los algoritmos (vea [Knuth, 1998b]).

Notas

Las relaciones de recurrencia se estudian con más detalle en [Liu, 1985; Roberts; y Tucker]. En [Johnsonbaugh] se presentan varias aplicaciones del análisis de algoritmos.

[Cull] proporciona algoritmos para resolver ciertos problemas de la Torre de Hanoi con complejidad mínima de espacio y tiempo. [Hinz] es un análisis exhaustivo de la Torre de Hanoi con 50 referencias.

La tela de araña en economía apareció primero en [Ezekiel].

Todos los libros de estructuras de datos y algoritmos tienen presentaciones amplias de búsqueda y ordenamiento (vea por ejemplo, [Brassard; Cormen; Johnsonbaugh; Knuth, 1998b; Kruse; y Nyhoff]).

Las relaciones de recurrencia también se conocen como ecuaciones diferenciadas. [Goldberg] contiene un análisis de las ecuaciones diferenciadas y sus aplicaciones.

Repaso del capítulo

Sección 7.1

1. Relación de recurrencia
2. Condición inicial
3. Interés compuesto
4. Torre de Hanoi
5. Tela de araña de la economía
6. Función de Ackermann

Sección 7.2

7. Solución de una relación de recurrencia por iteraciones
8. Relación de recurrencia homogénea lineal de orden n con coeficientes constantes y cómo resolver una relación de recurrencia de segundo orden
9. Crecimiento de población

Sección 7.3

10. Cómo encontrar una relación de recurrencia que describa el tiempo requerido por un algoritmo recursivo
11. Orden de selección

12. Búsqueda binaria
13. Mezcla de sucesiones
14. Merge sort

Autoevaluación del capítulo

Sección 7.1

1. Conteste los incisos *a)* a *c)* para la sucesión definida por las reglas:
1. El primer término es 3.
2. El n-ésimo término es n más el término anterior.

a) Escriba los primeros cuatro términos de la sucesión.
b) Encuentre una condición inicial para la sucesión.
c) Encuentre una relación de recurrencia para la sucesión.

2. Suponga que una persona invierte \$4000 al 17% de interés anual compuesto. Sea A_n la cantidad al final del año n. Encuentre una relación de recurrencia y una condición inicial para la sucesión $A_0, A_1, \ldots$.

3. Sea P_n el número de particiones de una conjunto de n elementos, Demuestre que $P_0, P_1, \ldots$ satisface la relación de recurrencia

$$P_n = \sum_{k=0}^{n-1} C(n-1, k)\, P_k.$$

4. Suponga que se tiene un tablero rectangular de $2 \times n$ dividido en $2n$ cuadros. Sea a_n el número de maneras de cubrir exactamente este tablero con fichas de dominó de 1×2. Demuestre que la sucesión $\{a_n\}$ satisface la relación de recurrencia

$$a_n = a_{n-1} + a_{n-2}.$$

Demuestre que $a_n = f_{n+1}$, donde $\{f_n\}$ es la sucesión de Fibonacci.

Sección 7.2

5. La relación de recurrencia

$$a_n = a_{n-1} + a_{n-3}$$

¿es una relación de recurrencia homogénea lineal con coeficientes constantes?

En los ejercicios 6 y 7, resuelva la relación de recurrencia sujeta a las condiciones iniciales.

6. $a_n = -4a_{n-1} - 4a_{n-2}; \quad a_0 = 2, \quad a_1 = 4$

7. $a_n = 3a_{n-1} + 10a_{n-2}; \quad a_0 = 4, \quad a_1 = 13$

8. Sea c_n el número de cadenas de $\{0, 1, 2\}$ de longitud n que contienen un número par de unos. Escriba una relación de recurrencia y una condición inicial que defina la sucesión $c_1, c_2, \ldots$. Resuelva la relación de recurrencia para obtener una fórmula explícita para c_n.

Sección 7.3

Los ejercicios 9 al 12 se refieren al siguiente algoritmo.

Algoritmo

Evaluación polinomial

Este algoritmo evalúa el polinomio

$$p(x) = \sum_{k=0}^{n} c_k x^{n-k}$$

en el punto t.

Entrada: La sucesión de coeficientes $c_0, c_1, \ldots, c_n$, el valor t y n
Salida: $p(t)$

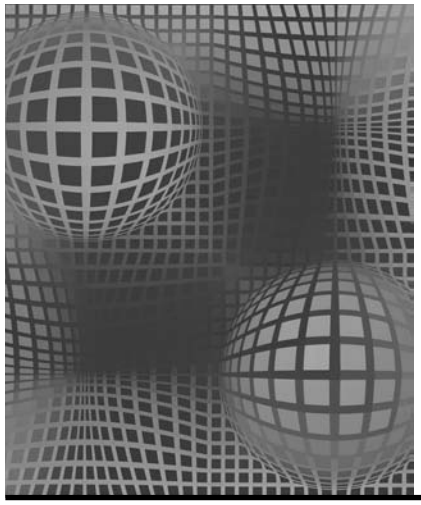
```
poli(c, n, t) {
    if (n == 0)
        return c_0
    return t * poli(c, n − 1, t) + c_n
}
```

Sea b_n el número de multiplicaciones requeridas para calcular $p(t)$.

9. Encuentre una relación de recurrencia y una condición inicial para la sucesión $\{b_n\}$.
10. Calcule b_1, b_2 y b_3.
11. Resuelva la relación de recurrencia del ejercicio 9.
12. Suponga que calculamos $p(t)$ mediante una técnica directa que requiere $n - k$ multiplicaciones para calcular $c_k t^{n-k}$. ¿Cuántas multiplicaciones se requerirán para calcular $p(t)$? ¿Preferiría este método al algoritmo anterior? Explique su respuesta.

Ejercicios para computadora

1. Escriba un programa que imprima la cantidad acumulada cada año si una persona invierte n dólares al $p\%$ de interés anual compuesto.
2. Escriba un programa que imprima la cantidad acumulada cada año si una persona invierte n dólares al $p\%$ de interés compuesto m veces al año.
3. Escriba un programa que resuelva el juego de la Torre de Hanoi con tres estacas.
4. Escriba un programa que resuelva el juego de la Torre de Hanoi con cuatro estacas en menos movimientos que la solución del juego para tres estacas.
5. Escriba un programa que despliegue la tela de araña de la economía.
6. Escriba un programa para calcular la función de Ackermann.
7. Escriba un programa que imprima una solución al programa de comunicación de n nodos (vea el ejercicio 48, sección 7.1).
8. Escriba un programa que imprima todas las maneras de gastar n dólares en las condiciones del ejercicio 50, sección 7.1).
9. Escriba un programa que imprima todas las cadenas de n bits que no contienen el patrón 010.
10. Escriba un programa que calcule la sucesión de Lucas (vea el ejercicio 63, sección 7.1).
11. Escriba un programa que liste todas las permutaciones sube/baja de longitud n (vea la definición antes del ejercicio 69, sección 7.1).
12. Implemente los algoritmos 7.3.1 (orden de selección) y 7.3.8 (merge sort) y otros algoritmos para ordenar como programas y compare los tiempos necesarios para ordenar n artículos.
13. Implemente la búsqueda binaria (algoritmo 7.3.2) como un programa. Mida el tiempo que emplea el programa para varias claves y varios valores de n. Compare estos resultados con la estimación teórica para el tiempo en el peor caso $\Theta(\lg n)$.
14. Escriba un programa para mezclar dos sucesiones.
15. Implemente un cálculo de a^n no recursivo que use la multiplicación repetida y los algoritmos 7.3.11 y 7.3.12 como programas de computadora, y compare los tiempos necesarios para ejecutar cada uno.
16. Implemente los métodos para calcular los elementos mayor y menor en un arreglo (vea los ejercicios 29 al 39, sección 7.3), y compare los tiempos necesarios para ejecutar cada uno.



Capítulo 8

TEORÍA DE GRÁFICAS

Bueno, salí de viaje, y fui al norte, a Providence.
Conocí al alcalde.
¡El alcalde de Providence!
Estaba sentado en el lobby del hotel.
¿Qué dijo?
Dijo "Buenos días", y yo dije: "Tiene una linda ciudad, alcalde". Y después tomó café conmigo. Luego fui a Waterbury, que es una ciudad agradable, una ciudad con un gran reloj, el famoso reloj de Waterbury. Vendí una buena factura ahí. Y después Boston; Boston es la cuna de la revolución. Una hermosa ciudad. Luego un par de pueblos en Massachussets, y seguí a Portland y Bangor y ¡de regreso a casa!

DE MUERTE DE UN VENDEDOR

Aunque la primera publicación de teoría de gráficas data de 1736 (vea el ejemplo 8.2.16) y varios resultados importantes de teoría de gráficas se obtuvieron en el siglo XIX, no fue sino hasta 1920 que surgió un interés sostenido, amplio e intenso en la teoría de gráficas. En realidad, el primer libro de texto de este tema ([König]) apareció en 1936. Sin duda, una de las razones del reciente interés en la teoría de gráficas es su aplicabilidad en muchos campos, incluyendo ciencias de la computación, química, investigación de operaciones, ingeniería eléctrica, lingüística y economía.

En este capítulo, primero se expone cierta terminología básica y ejemplos de gráficas. Después se analizan algunos conceptos importantes en la teoría de gráficas, que incluyen trayectorias y ciclos. Luego se presentan dos problemas clásicos, los ciclos hamiltonianos y el problema del agente viajero. Se presenta un algoritmo de la ruta más corta que encuentra con eficiencia la trayectoria más corta entre dos puntos dados. Después de exponer las maneras de representar las gráficas, se estudiará el tema de cuándo dos gráficas son en esencia la misma (es decir, cuándo dos gráficas son isomorfas) y cuándo una gráfica se puede dibujar en el plano sin que sus aristas se crucen. Se concluirá con la presentación de una solución basada en un modelo de gráfica para el juego de Locura instantánea.

8.1 ➜ Introducción

La figura 8.1.1 muestra el sistema de carreteras de Wyoming; cierta persona es responsable de inspeccionar este sistema. En particular, el inspector de carreteras debe recorrerlas y entregar in-

† Esta sección se puede omitir sin pérdida de continuidad.

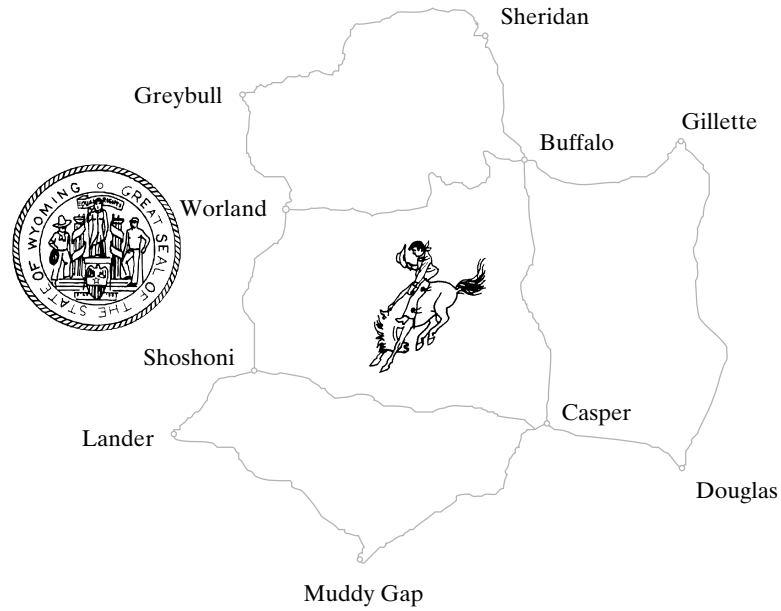


Figura 8.1.1 Parte del sistema de carreteras de Wyoming.

formes de las condiciones de los caminos, la visibilidad de las líneas pintadas, el estado de las señales, etcétera. Como el inspector vive en Greybull, la manera más económica de inspeccionar todos los caminos sería comenzar en Greybull, recorrer cada carretera exactamente una vez y regresar a Greybull. ¿Es esto posible? ¿Puede decidir antes de seguir leyendo?

WWW

El problema se puede modelar como una **gráfica**. De hecho, como las gráficas se dibujan con puntos y líneas, tienen la apariencia de mapas de carreteras. La figura 8.1.2, contiene el dibujo de una gráfica G que modela el mapa de la figura 8.1.1. Los puntos se llaman **vértices** y las líneas que conectan a los vértices se llaman **aristas**. (Más adelante en esta sección, se definirán todos los términos con cuidado). Cada vértice se etiquetó con las primeras tres letras de la ciudad correspondiente. Las aristas se etiquetaron $e_1, \ldots, e_{13}$. Cuando se dibuja una gráfica, la única información de importancia es qué vértices se conectan con qué aristas. Por esta razón, la gráfica de la figura 8.1.2 pudo haberse dibujado como la figura 8.1.3.

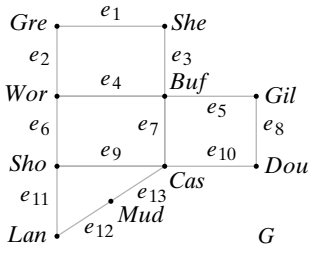


Figura 8.1.2 Modelo de gráficas para el sistema de carreteras de la figura 8.1.1.

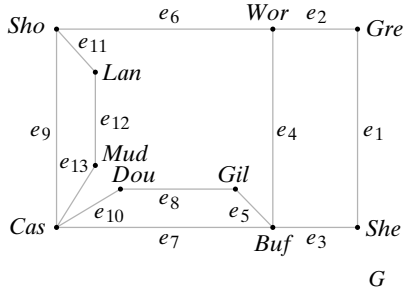


Figura 8.1.3 Modelo de gráficas alternativo, pero equivalente, del sistema de carreteras de la figura 8.1.1.

Si se inicia en el vértice v_0, se viaja por una arista al vértice v_1, por otra arista al vértice v_2, etcétera, y con el tiempo se llega al vértice v_n; este viaje completo recibe el nombre de **trayectoria** o **ruta** de v_0 a v_n. La trayectoria que comienza en *She*, va a *Buf* y termina en *Gil* corresponde a un viaje en el mapa de la figura 8.1.1 que comienza en Sheridan, va a Buffalo y termina en Gillette. El problema del inspector de carreteras se enuncia de otra manera para el modelo de gráficas G: ¿Existe una ruta del vértice *Gre* al vértice *Gre* que pase una vez por todas las aristas?

Es posible demostrar que el inspector de carreteras no puede comenzar en Greybull, viajar por cada camino justo una vez y regresar a Greybull. Para poner la respuesta en términos de gráficas, no existe una trayectoria del vértice *Gre* al vértice *Gre* en la figura 8.1.2 que recorra todas las aristas una vez. Para ver esto, suponga que existe tal trayectoria y considere el vértice *Wor*. Cada vez que se llega a *Wor* por alguna arista, se debe salir de *Wor* por una arista diferente. Más aún, cada arista que toca *Wor* se debe usar. Entonces las aristas en *Wor* ocurren en pares. Se concluye que un número par de aristas debe tocar *Wor*. Como tres aristas tocan a *Wor*, se tiene una contradicción. Por lo tanto, no existe una trayectoria del vértice *Gre* al vértice *Gre* en la figura 8.1.2 que recorra todas las aristas justo una vez. El argumento se aplica a una gráfica arbitraria G. Si G tiene una trayectoria del vértice v al vértice v que recorre todas las aristas exactamente una vez, un número par de aristas deben tocar cada vértice. Este problema se estudia con más detalle en la sección 8.2.

Por el momento, se dan algunas definiciones formales.

Definición 8.1.1 ▶

Una *gráfica* (o *gráfica no dirigida*) G consiste en un conjunto V de *vértices* (o *nodos*) y un conjunto E de *aristas* (o *arcos*) tal que cada arista $e \in E$ se asocia con un par no ordenado de vértices. Si existe una arista única e asociada con los vértices u y w, se escribe $e = (v, w)$ o $e = (w, v)$. En este contexto, (v, w) denota una arista entre v y w en una gráfica no dirigida y *no* es un par ordenado.

Una *gráfica dirigida* (o *digráfica*) G consiste en un conjunto V de *vértices* (o *nodos*) y un conjunto E de *aristas* (o *arcos*) tales que cada arista $e \in E$ está asociada con un par ordenado de vértices. Si hay una arista única e asociada con el par ordenado (v, w) de vértices, se escribe $e = (v, w)$, que denota una arista de v a w.

Se dice que una arista e en una gráfica (no dirigida o dirigida) que se asocia con el par de vértices v y w es *incidente sobre* v y w, y se dice que v y w son *incidentes sobre* e y son v*értices adyacentes*.

Si G es una gráfica (no dirigida o dirigida) con vértices V y aristas E, se escribe $G = (V, E)$.

A menos que se especifique lo contrario, se supone que los conjuntos E y V son finitos y que V es no vacío. ◀

Ejemplo 8.1.2 ▶

En la figura 8.1.2 la gráfica (no dirigida) G consiste en el conjunto de vértices

$$V = \{Gre, She, Wor, Buf, Gil, Sho, Cas, Dou, Lan, Mud\}$$

y el conjunto de aristas

$$E = \{e_1, e_2, \ldots, e_{13}\}$$

La arista e_1 se asocia con un par no ordenado de vértices $\{Gre, She\}$, y la arista e_{10} se asocia con el par no ordenado de vértices $\{Cas, Dou\}$. La arista e_1 se denota por (Gre, She) o (She, Gre), y la arista e_{10} se denota por (Cas, Dou) o (Dou, Cas). La arista e_4 es incidente sobre *Wor* y *Buf*, y los vértices *Wor* y *Buf* son adyacentes. ◀

Ejemplo 8.1.3 ▶

La figura 8.1.4 muestra una gráfica dirigida. Las aristas dirigidas se indican por flechas. La arista e_1 se asocia con el par ordenado de vértices (v_2, v_1) y la arista e_7 se asocia con

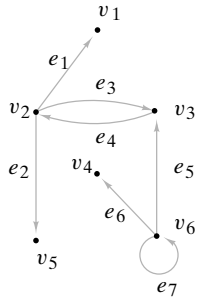


Figura 8.1.4 Una gráfica dirigida.

el par ordenado de vértices (v_6, v_6). La arista e_1 se denota (v_2, v_1), y la arista e_7 se denota (v_6, v_6). ◀

La definición 8.1.1 permite que diferentes aristas se asocien con el mismo par de vértices. Por ejemplo, en la figura 8.1.5, las aristas e_1 y e_2 se asocian ambas con el par de vértices $\{v_1, v_2\}$. Estas aristas se llaman **aristas paralelas**. Una arista incidente en un mismo vértice se llama **lazo**. Por ejemplo, en la figura 8.1.5, la arista $e_3 = (v_2, v_2)$ es un lazo. Un vértice como v_4 en la figura 8.1.5, que no incide en ninguna arista, se llama **vértice aislado**. Una gráfica sin lazos ni aristas paralelas se llama **gráfica simple**.

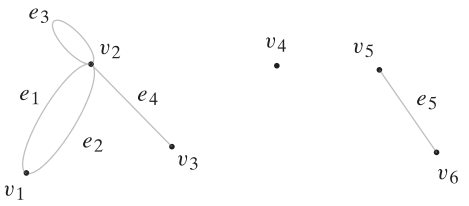


Figura 8.1.5 Una gráfica con aristas paralelas y un lazo.

Ejemplo 8.1.4 ▶ Como la gráfica de la figura 8.1.2 no tiene aristas paralelas ni lazos, es una gráfica simple. ◀

Algunos autores no permiten lazos o aristas paralelas cuando definen las gráficas. Se esperaría que si no se ha llegado a un acuerdo acerca de la definición de "gráfica", muchos de los términos en la teoría de gráficas tampoco tendrían definiciones estándar. Sin duda, esto es cierto. Al leer artículos y libros de gráficas, es necesario verificar las definiciones que se emplean.

Se dará un ejemplo que muestra cómo se utiliza un modelo de gráficas para analizar un problema de manufactura.

Ejemplo 8.1.5 ▶ Con frecuencia en la manufactura, es necesario hacer agujeros en hojas de metal (vea la figura 8.1.6). Luego se atornillan las componentes a estas hojas de metal. Los agujeros se perforan usando un taladro controlado por computadora. Para ahorrar tiempo y dinero, el taladro debe moverse tan rápido como sea posible. Se modelará la situación como una gráfica.

Los vértices de la gráfica corresponden a los agujeros (figura 8.1.7). Cada par de vértices se conecta por una arista. En cada arista se escribe el tiempo para mover el taladro entre los hoyos correspondientes. Una gráfica con números en las aristas (como en la figura 8.1.7) se llama **gráfica ponderada**. Si la arista e se etiqueta k, se dice que el **peso de la**

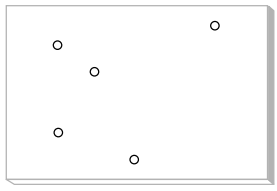


Figura 8.1.6 Hoja de metal con agujeros para tornillos.

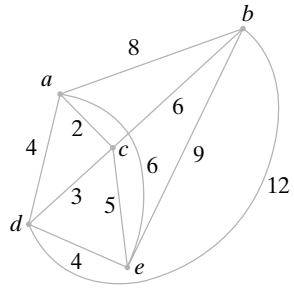


Figura 8.1.7 Modelo de gráficas de la hoja de metal de la figura 8.1.6. El peso de la arista es el tiempo para mover el taladro.

arista e es k. Por ejemplo, en la figura 8.1.7 el peso de la arista (c, e) es 5. En una gráfica ponderada, la **longitud de una ruta** es la suma de los pesos de las aristas en la ruta. Por ejemplo, en la figura 8.1.7 la longitud de la ruta que comienza en a, visita c y termina en b es 8. En este problema, la longitud de una trayectoria que comienza en el vértice v_1 y luego visita v_2, v_3, . . . , en este orden, y termina en v_n representa el tiempo que toma al taladro comenzar en el agujero h_1 y luego moverse a h_2, h_3, . . . , en ese orden y terminar en h_n, donde el agujero h_i corresponde el vértice v_i. Una ruta de longitud mínima que visita todos los vértices exactamente una vez representa la ruta óptima que debe seguir el taladro.

Suponga que en este problema se requiere que la trayectoria comience en el vértice a y termine en el vértice e. Se puede encontrar la ruta de longitud mínima numerando todas las rutas posibles de a a e que pasan por todos los vértices justo una vez y eligiendo la menor (vea la tabla 8.1.1). Se ve que la ruta que visita los vértices a, b, c, d, e, en ese orden, tiene longitud mínima. Por supuesto, un par diferente de vértices de inicio y terminación produciría una ruta aún más corta.

TABLA 8.1.1 ■ Trayectorias en la gráfica de la figura 8.1.7 de a a e que pasan por todos los vértices justo una vez, y sus longitudes.

Trayectoria	*Longitud*
a, b, c, d, e	21
a, b, d, c, e	28
a, c, b, d, e	24
a, c, d, b, e	26
a, d, b, c, e	27
a, d, c, b, e	22

◀

Numerar todas las trayectorias de un vértice v a un vértice w, como se hizo en el ejemplo 8.1.5, es una manera bastante tardada para encontrar la trayectoria de longitud mínima de v a w que visita todos los vértices una vez. Por desgracia, nadie conoce un método que sea mucho más práctico para gráficas arbitrarias. Este problema es una versión del **problema del agente viajero**. Se estudiará ese problema en la sección 8.3.

Ejemplo 8.1.6 ▶

Números de Bacon

WWW

El actor Kevin Bacon ha aparecido en numerosas películas que incluyen *Diner* y *Apollo 13*. Se dice que los actores que han aparecido en una película con Bacon tienen un *uno de número Bacon*. Por ejemplo, Ellen Barkin tiene un uno de número Bacon porque apareció con Bacon en *Diner*. Los actores que no hicieron una película con Bacon pero que trabajaron con un actor cuyo número Bacon es uno, se dice que tienen *dos de número de Bacon*. Los números de Bacon más altos se definen de manera similar. Por ejemplo, Bela Lugosi tiene un número de Bacon de tres. Lugosi actuó en *Black Friday* con Emmett Vogan, quien estuvo en *With a Song in My Heart* con Robert Wagner y Wagner trabajó en *Wild Things* con Bacon. Se desarrollará un modelo de gráficas para los números de Bacon.

Los vértices denotan actores; se coloca una arista entre dos actores diferentes si aparecieron juntos en al menos una película (vea la figura 8.1.8). En una gráfica no ponderada, la longitud de una ruta es el número de aristas en ella. Entonces el número de Bacon de un actor es la longitud de la ruta más corta desde el vértice correspondiente a ese actor hasta el vértice de Bacon. En la sección 8.4 se analiza el problema general de encontrar la ruta más corta en una gráfica. A diferencia de la situación del ejemplo 8.1.5, existen algoritmos eficientes para encontrar la ruta más corta.

Es interesante que la *mayoría* de los actores, incluso aquellos que murieron hace años, tienen números de Bacon de tres o menos. La página de Internet de este libro tiene un enlace a una página que verifica los números de Bacon de actores arbitrarios. Vea en el ejercicio 30 un modelo de gráficas similar.

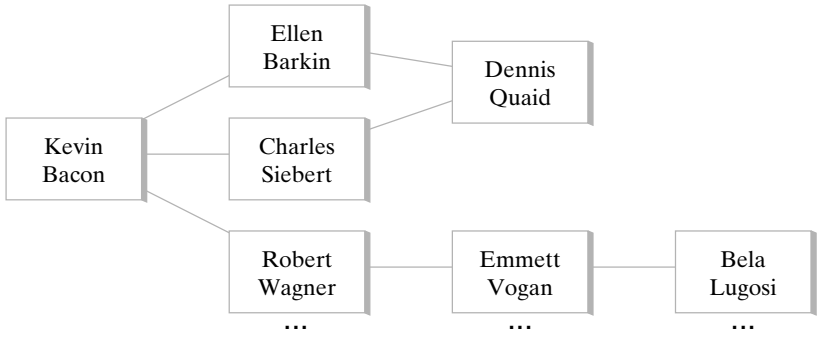


Figura 8.1.8 Parte de una gráfica que modela los números de Bacon. Los vértices denotan actores. Existe una arista entre dos actores si aparecieron juntos en al menos una película. Por ejemplo, hay una arista entre Ellen Barkin y Dennis Quaid porque ambos aparecieron en *The Big Easy*. El número de Bacon de un actor es la longitud de la ruta más corta entre el actor y Bacon. Por ejemplo, el número de Bacon de Bela Lugosi es tres porque la longitud de la ruta más corta entre Lugosi y Bacon es tres. ◀

Ejemplo 8.1.7 ▶

Gráficas de similitud

Este ejemplo trata el problema de agrupar objetos "similares" en clases basadas en las propiedades de los objetos. Por ejemplo, suponga que cierto número de personas implementan en C++ un algoritmo dado y que se quiere agrupar los programas "parecidos" en clases determinadas por ciertas propiedades de los programas (vea la tabla 8.1.2). Suponga que se seleccionan como propiedades

1. El número de líneas en el programa
2. El número de instrucciones para regresar (*return*) en el programa
3. El número de llamadas de funciones en el programa

TABLA 8.1.2 ■ Programas en C++ que implementan el mismo algoritmo

Programa	*Número de líneas de programa*	*Número de instrucciones "return"*	*Número de llamadas de funciones*
1	66	20	1
2	41	10	2
3	68	5	8
4	90	34	5
5	75	12	14

Una **gráfica de similitud** G se construye como sigue. Los vértices corresponden a los programas. Un vértice se denota por (p_1, p_2, p_3), donde p_i es el valor de la propiedad i. Se define una **función de disimilitud** s como sigue. Para cada par de vértices $v = (p_1, p_2, p_3)$ y $w = (q_1, q_2, q_3)$ se establece

$$s(v, w) = |p_1 - q_1| + |p_2 - q_2| + |p_3 - q_3|.$$

Si v_i es el vértice correspondiente al programa i, se obtiene

$$\begin{array}{llll} s(v_1, v_2) = 36, & s(v_1, v_3) = 24, & s(v_1, v_4) = 42, & s(v_1, v_5) = 30, \\ s(v_2, v_3) = 38, & s(v_2, v_4) = 76, & s(v_2, v_5) = 48, & s(v_3, v_4) = 54, \\ & s(v_3, v_5) = 20, & s(v_4, v_5) = 46. & \end{array}$$

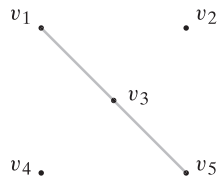


Figura 8.1.9 Una gráfica de similitud correspondiente a los programas de la tabla 8.1.2 con $S = 25$.

Si v y w son vértices correspondientes a dos programas, $s(v, w)$ es una medida de qué tan disímiles son los programas. Un valor grande de $s(v, w)$ indica disimilitud, mientras que un valor pequeño indica similitud.

Para un número fijo S, se inserta una arista entre dos vértices v y w si $s(v, w) < S$. (En general, habrá gráficas de similitud diferentes para valores distintos de S.) Se dice que v y w están **en la misma clase** si $v = w$ o si hay una trayectoria de v a w. En la figura 8.1.9

se muestra la gráfica correspondiente a los programas de la tabla 8.1.2 con $S = 25$. En ella, los programas se agrupan en tres clases: {1, 3, 5}, {2} y {4}. En un problema real, un valor adecuado para S podría seleccionarse por prueba y error o el valor de S podría elegirse de manera automática de acuerdo con algunos criterios predeterminados. ◀

Ejemplo 8.1.8 ▶

El cubo-*n* (hipercubo)

WWW

La computadora tradicional, con frecuencia llamada **computadora serial**, ejecuta una instrucción a la vez. Nuestra definición de "algoritmo" también supone que se ejecuta una instrucción a la vez. Estos algoritmos se llaman **algoritmos seriales**. Conforme los costos del hardware han disminuido, se ha vuelto factible construir **computadoras paralelas** con muchos procesadores que son capaces de ejecutar varias instrucciones a la vez. Las gráficas con frecuencia son modelos convenientes para describir estas máquinas. Los algoritmos asociados se conocen como **algoritmos paralelos**. Muchos problemas se resuelven con mayor rapidez usando computadoras paralelas en lugar de seriales. Se analizará un modelo de computación paralela conocido como el **cubo-*n*** o **hipercubo**.

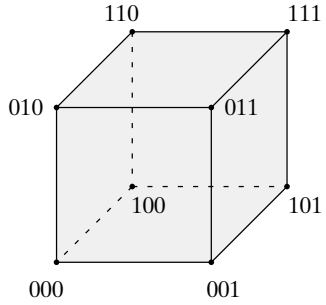


Figura 8.1.10 El cubo–3.

El cubo-n tiene 2^n procesadores, $n \geq 1$, que se representan por vértices (figura 8.1.10) etiquetados 0, 1, . . . , $2^n - 1$. Cada procesador tiene su propia memoria local. Una arista conecta a dos vértices si la representación binaria de sus etiquetas difiere exactamente en un bit. Durante una unidad de tiempo, todos los procesadores en el cubo-n pueden ejecutar una instrucción simultáneamente y después comunicarse con el procesador adyacente. Si un procesador necesita comunicarse con uno no adyacente, el primer procesador envía un mensaje que incluye la ruta al receptor, y el destino final del mismo. Puede tomar varias unidades de tiempo que un procesador se comunique con otro no adyacente.

También es posible describir el cubo-n de manera recursiva. El cubo-1 tiene dos procesadores, etiquetados 0 y 1, y una arista. Sean H_1 y H_2 dos cubos-$(n-1)$ cuyos vértices se etiquetan en binarios 0, . . . , $2^{n-1} - 1$ (vea la figura 8.1.11). Se coloca una arista entre cada par de vértices, una desde H_1 y otra desde H_2, siempre que los vértices tengan etiquetas idénticas. Se cambia la etiqueta L en cada vértice de H_1 a $0L$ y la etiqueta L en cada vértice de H_2 se cambia a $1L$. Se obtiene un cubo-n (ejercicio 39). Vea en los ejercicios 43 a 45 una manera alternativa de construir el cubo-n.

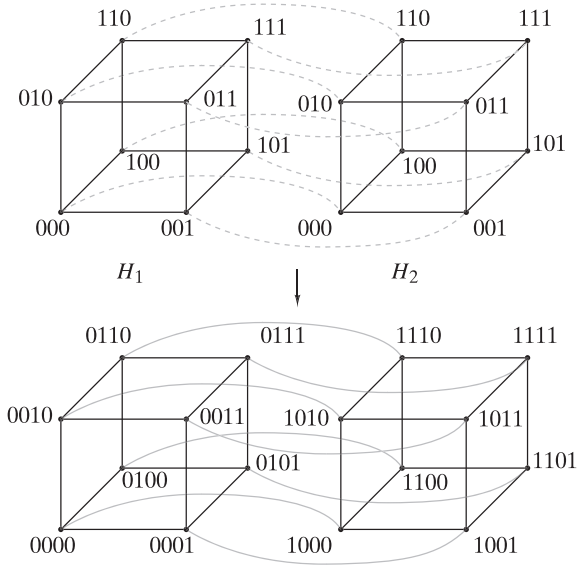


Figura 8.1.11 Combinación de dos cubos-3 para obtener un cubo-4.

El cubo-n es un modelo importante de computación porque se han construido varias máquinas de este tipo y están trabajando. Todavía más, algunos otros modelos de computación paralela se pueden simular con el hipercubo. Este último punto se considera con más detalle en los ejemplos 8.3.5 y 8.6.3. ◀

Se concluye esta sección de introducción con las definiciones de algunas gráficas especiales que aparecen con frecuencia en la teoría de gráficas.

Definición 8.1.9 ▶

La *gráfica completa sobre n vértices*, denotada por K_n, es la gráfica simple con n vértices en la que hay una arista entre cada par de vértices distintos. ◀

Ejemplo 8.1.10 ▶

La gráfica completa sobre cuatro vértices, K_4, se reproduce en la figura 8.1.12.

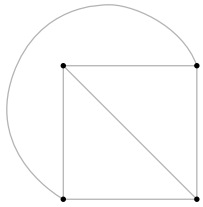


Figura 8.1.12 Gráfica completa K_4.

◀

Definición 8.1.11 ▶

Una gráfica $G = (V, E)$ es *bipartita* si existen subconjuntos V_1 y V_2 (cualquiera de los dos posiblemente vacío) de V tales que $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V$, y cada arista en E es incidente sobre un vértice en V_1 y un vértice en V_2. ◀

Ejemplo 8.1.12 ▶

La gráfica de la figura 8.1.13 es bipartita ya que si se deja

$$V_1 = \{v_1, v_2, v_3\} \qquad \text{y} \qquad V_2 = \{v_4, v_5\},$$

cada arista incide en un vértice en V_1 y en un vértice en V_2. ◀

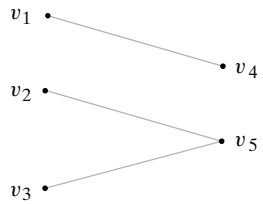


Figura 8.1.13 Gráfica bipartita.

Observe que la definición 8.1.11 establece que si e es una arista en una gráfica bipartita, entonces e incide en un vértice en V_1 y en un vértice en V_2. *No* establece que si v_1 es un vértice en V_1 y v_2 es un vértice en V_2, entonces hay una arista entre v_1 y v_2. Por ejemplo, la gráfica de la figura 8.1.13 es bipartita puesto que cada arista incide en un vértice en $V_1 = \{v_1, v_2, v_3\}$ y un vértices en $V_2 = \{v_4, v_5\}$. Sin embargo, no todas las aristas entre vértices de V_1 y V_2 están en la gráfica. Por ejemplo, la arista (v_1, v_5) no está.

Ejemplo 8.1.13 ▶

La gráfica de la figura 8.1.14 *no* es bipartita. A menudo es más fácil probar que una gráfica no es bipartita por contradicción.

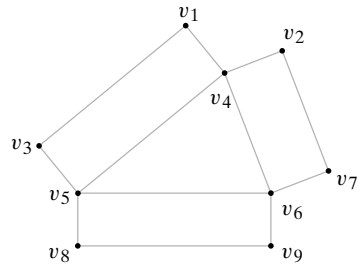


Figura 8.1.14 Gráfica que no es bipartita.

Suponga que la gráfica de la figura 8.1.14 es bipartita. Entonces se puede hacer una partición del conjunto de vértices en dos subconjuntos V_1 y V_2 tal que cada arista incida en un vértice en V_1 y un vértice en V_2. Ahora considere los vértices v_4, v_5 y v_6. Como v_4 y v_5 son adyacentes, uno está en V_1 y el otro en V_2. Se puede suponer que v_4 está en V_1 y v_5 en V_2. Como v_5 y v_6 son adyacentes y v_5 está en V_2, v_6 está en V_1. Como v_4 y v_6 son adyacentes y v_4 está en V_1, v_6 está en V_2. Pero resulta que v_6 está en V_1 y en V_2, lo cual es una contradicción ya que V_1 y V_2 son ajenos. Por lo tanto, la gráfica de la figura 8.1.14 no es bipartita. ◀

Ejemplo 8.1.14 ▶

La gráfica completa K_1 sobre un vértice es bipartita. Se puede establecer que V_1 es el conjunto que contiene el único vértice y V_2 es el conjunto vacío. Entonces cada arista (de hecho, ¡ninguna!) es incidente sobre un vértice en V_1 y un vértice en V_2. ◀

Definición 8.1.15 ▶

La *gráfica bipartita completa sobre m y n vértices*, denotada por $K_{m,n}$, es la gráfica simple donde el conjunto de vértices tiene una partición en V_1 con m vértices y V_2 con n vértices y donde el conjunto de aristas consiste en todas las aristas de la forma (v_1, v_2) con $v_1 \in V_1$ y $v_2 \in V_2$. ◀

Ejemplo 8.1.16 ▶

La gráfica bipartita completa sobre dos y cuatro vértices, $K_{2,4}$, se ilustra en la figura 8.1.15.

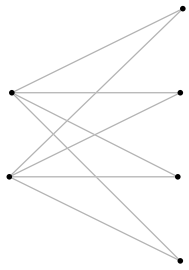


Figura 8.1.15
Gráfica bipartita completa $K_{2,4}$.

◀

Sugerencias para resolver problemas

Para modelar una situación dada como una gráfica, primero debe decidirse qué representan los vértices. Después una arista entre dos vértices representa algún tipo de relación. Por ejemplo, si varios equipos se enfrentan en juegos de fútbol, los vértices podrían ser los equipos. Después se podría poner una arista entre dos vértices (equipos) si los dos equipos representados por los vértices participan al menos en un juego. La gráfica mostraría entonces qué equipos se enfrentaron.

Para determinar si una gráfica es bipartita, intente separar los vértices en dos conjuntos ajenos V_1 y V_2 de manera que cada arista incida en un vértice de un conjunto y un vértice del otro conjunto. Si tiene éxito, la gráfica es bipartita y cuenta con los conjuntos V_1 y V_2. Si fracasa, la gráfica no es bipartita. Para tratar de separar los vértices en dos conjuntos ajenos, elija un vértice de inicio. Ponga $v \in V_1$. Coloque todos los vértices adyacentes a v en V_2. Elija un vértice $w \in V_2$. Coloque todos los vértices adyacentes a w en V_1. Elija un vértice $v' \in V_2$, $v' \neq v$. Ponga todos los vértices adyacentes a v' en V_2. Continúe de esta manera. Si puede colocar cada vértice ya sea en V_1 o en V_2, pero no en ambos, la gráfica es bipartita. Si en algún momento se ve forzado a colocar un vértice en ambos conjuntos V_1 y V_2, la gráfica no es bipartita.

Sección de ejercicios de repaso

1. Defina *gráfica no dirigida*.
2. Dé un ejemplo de algo en la vida real que se pueda modelar por medio de una gráfica no dirigida.
3. Defina *gráfica dirigida*.
4. Dé un ejemplo de algo en la vida real que se pueda modelar por medio de una gráfica dirigida.
5. ¿Qué significa para una arista ser *incidente sobre un vértice*?
6. ¿Qué significa para un vértice ser *incidente sobre una arista*?
7. ¿Qué significa para v y w ser *vértices adyacentes*?
8. ¿Qué son aristas paralelas?
9. ¿Qué es un lazo?
10. ¿Qué es un vértice aislado?
11. ¿Qué es una gráfica simple?
12. ¿Qué es una gráfica ponderada?
13. Dé un ejemplo de algo en la vida real que se pueda modelar por medio de una gráfica ponderada.
14. Defina la *longitud de una trayectoria* en una gráfica ponderada.
15. ¿Qué es una gráfica de similitud?

16. Defina un *cubo-n*.

17. ¿Qué es una computadora serial?

18. ¿Qué es un algoritmo serial?

19. ¿Qué es una computadora paralela?

20. ¿Qué es un algoritmo paralelo?

21. ¿Qué es una gráfica completa sobre n vértices? ¿Cómo se denota?

22. Defina *gráfica bipartita*.

23. ¿Qué es una gráfica bipartita completa sobre m y n vértices? ¿Cómo se denota?

Ejercicios

En un torneo, el Nieve venció a los Faisanes una vez, el Rascacielos venció al Tuna una vez, el Nieve venció al Rascacielos dos veces, los Faisanes vencieron al Tuna una vez y los Faisanes vencieron al Rascacielos una vez. En los ejercicios 1 al 4, use una gráfica para modelar el torneo. Los equipos son los vértices. Describa el tipo de gráfica usada (no dirigida, dirigida, simple).

1. Hay una arista entre los equipos si los equipos jugaron.

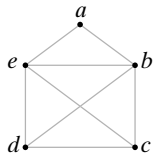
2. Hay una arista entre los equipos para cada juego jugado.

3. Hay una arista del equipo t_i al equipo t_j si t_i venció a t_j al menos una vez.

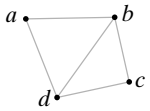
4. Hay una arista del equipo t_i al equipo t_j por cada victoria de t_i sobre t_j.

Explique por qué ninguna gráfica en los ejercicios 5 al 7 tiene una trayectoria del vértice a *al vértice* a *que pasa por cada arista justo una vez.*

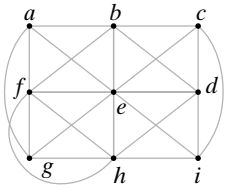
5.



6.

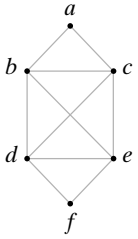


7.

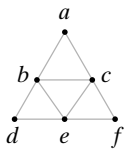


Pruebe que cada gráfica en los ejercicios 8 al 10 tiene una trayectoria del vértice a *al vértice* a *que pasa por cada arista justo una vez, encontrando la trayectoria por inspección.*

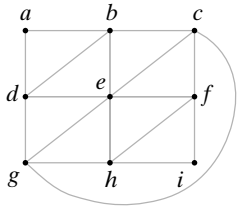
8.



9.

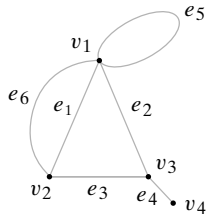


10.

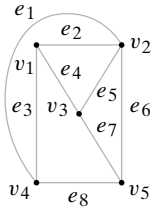


Para cada gráfica $G = (V, E)$ en los ejercicios 11 al 13, encuentre V, E, todas las aristas paralelas, lazos, vértices aislados, y diga si G es una gráfica simple. Además, diga sobre qué vértices incide la arista e_1.

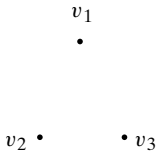
11.



12.



13.



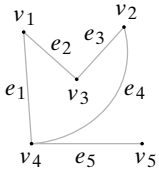
14. Dibuje K_3 y K_5.

15. Encuentre una fórmula para el número de aristas en K_n.

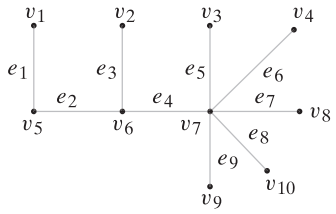
16. Dé un ejemplo de una gráfica bipartita diferente de los ejemplos de esta sección. Especifique los conjuntos ajenos de vértices.

Determine qué gráficas en los ejercicios 17 al 23 son bipartitas. Si la gráfica es bipartita, especifique los conjuntos ajenos de vértices.

17.



18.



19. Figura 8.1.2

20. Figura 8.1.5

21. Ejercicio 11

22. Ejercicio 12

23. Ejercicio 13

24. Dibuje $K_{2,3}$ y $K_{3,3}$.

25. Encuentre una fórmula para el número de aristas en $K_{m,n}$.

26. Muchos autores requieren que V_1 y V_2 sean no vacíos en la definición 8.1.11. Según estos autores, ¿cuáles gráficas en los ejemplos 8.1.12 a 8.1.14 son bipartitas?

En los ejercicios 27 al 29, encuentre una trayectoria de longitud mínima de v a w en la gráfica de la figura 8.1.7 que pase por cada vértice exactamente una vez.

27. $v = b, w = e$

28. $v = c, w = d$

29. $v = a, w = b$

30. Paul Erdős (1913−1996) fue uno de los matemáticos más prolíficos de todos los tiempos. Fue autor o coautor en cerca de 1500 artículos. Se dice que los matemáticos que trabajaron en un artículo con Erdős tienen un *número de Erdős de uno*. Los matemáticos que no son coautores con Erdős pero que publicaron con un matemático que tiene número de Erdős de uno, tienen un *número de Erdős de dos*. Los números de Erdős mayores se definen de manera similar. Por ejemplo, el autor de este libro tiene un número de Erdős de cinco. Johnsonbaugh es coautor de un artículo con Tadao Murata, quien es coautor con A. T. Amin; Amin es coautor con Peter J. Slater; Slater es coautor con Frank Harary; y Harary es coautor de un artículo con Erdős. Desarrolle un modelo de gráficas para los números de Erdős. En su modelo, ¿qué es un número de Erdős?

31. El modelo de gráficas para los números de Bacon (vea el ejemplo 8.1.6), ¿es una gráfica simple?

32. Dibuje la gráfica de similitud que se obtiene al hacer $S = 40$ en el ejemplo 8.1.7. ¿Cuántas clases hay?

33. Dibuje la gráfica de similitud que se obtiene al hacer $S = 50$ en el ejemplo 8.1.7. ¿Cuántas clases hay?

34. En general, ¿"es similar a" es una relación de equivalencia?

35. Sugiera propiedades adicionales para el ejemplo 8.1.7 que resulten útiles al comparar programas.

36. ¿Cómo se puede automatizar la selección de S para agrupar datos en clases usando un gráfica de similitud?

37. Dibuje un cubo-2.

38. Haga un dibujo como el de la figura 8.1.11 para mostrar cómo se construye un cubo-3 a partir de dos cubos-2.

39. Pruebe que la construcción recursiva en el ejemplo 8.1.8 de hecho lleva a un cubo-n.

40. ¿Cuántas aristas inciden en un vértice en un cubo-n?

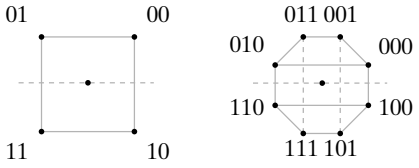
41. ¿Cuántas aristas hay en un cubo-n?

★42. ¿De cuántas maneras pueden etiquetarse los vértices de un cubo-n como $0, \ldots, 2^n - 1$, de forma que haya una arista entre dos vértices si y sólo si la representación binaria de sus etiquetas difiere exactamente en un bit.

[Bain] inventó un algoritmo para dibujar el cubo-n en el plano. En el algoritmo, todos los vértices están en el círculo de unidad en el plano xy. El ángulo de un punto es el ángulo desde el lado positivo del eje x en sentido contrario a las manecillas del reloj hasta el rayo que va del origen al punto. La entrada es n.

1. *Si $n = 0$, se coloca un vértice sin etiqueta en $(-1, 0)$ y se detiene.*
2. *Se invoca recursivamente este algoritmo con entrada $n - 1$.*
3. *Se mueve cada vértice para que su nuevo ángulo sea la mitad del actual, manteniendo las aristas conectadas.*
4. *Se refleja cada vértice y arista respecto al eje x.*
5. *Se conecta cada vértice arriba del eje x con su imagen de espejo abajo del eje x.*
6. *Se antepone 0 a las etiquetas de cada vértice arriba del eje x, y 1 a las etiquetas de los vértices abajo del eje x.*

Las siguientes figuras muestran la manera en que el algoritmo dibuja un cubo-2 y un cubo-3.

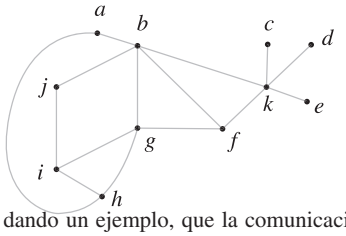


43. Muestre cómo el algoritmo construye el cubo-2 a partir del cubo-1.

44. Muestre cómo el algoritmo construye el cubo-3 a partir del cubo-2.

45. Muestre cómo el algoritmo construye el cubo-4 a partir del cubo-3.

Los ejercicios 46 al 48 se refieren a la siguiente gráfica. Los vértices representan oficinas. Una arista conecta dos oficinas si hay un enlace de comunicación entre las dos. Observe que cualquier oficina se puede comunicar con cualquier otra con un enlace de comunicación directo o haciendo que otros pasen el mensaje.

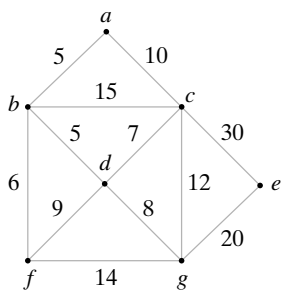


46. Muestre, dando un ejemplo, que la comunicación entre las oficinas es posible aun cuando se rompan algunos enlaces de comunicación.

47. ¿Cuál es el número máximo de enlaces de comunicación que se pueden romper teniendo todavía comunicación entre todas las oficinas?

48. Muestre una configuración en la que se rompió el número máximo de enlaces de comunicación y todavía es posible la comunicación entre todas las oficinas.

49. En la siguiente gráfica los vértices representan ciudades y los números sobre las aristas son los costos de construir las carreteras indicadas. Encuentre el sistema de carreteras menos costoso que conecte todas las ciudades.



En una gráfica de precedencias, los vértices modelan ciertas acciones. Por ejemplo, un vértice puede modelar una instrucción en un programa de computadora. Hay una arista del vértice v al vértice w si la acción modelada por v debe ocurrir antes que la acción modelada por w. Dibuje una gráfica de precedencias para cada programa de computadora en los ejercicios 50 al 52.

50.
$x = 1$
$y = 2$
$z = x + y$
$z = z + 1$

51.
$x = 1$
$y = 2$
$z = y + 2$
$w = x + 5$
$x = z + w$

52.
$x = 1$
$y = 2$
$z = 3$
$a = x + y$
$b = y + z$
$c = x + z$
$c = c + 1$
$x = a + b + c$

8.2 ➜ Trayectorias y ciclos

Si se piensa en los vértices de una gráfica como ciudades y las aristas como carreteras, una trayectoria (o ruta) corresponde a un viaje que comienza en alguna ciudad, pasa por varias ciudades y termina en alguna otra. Comenzaremos por dar una definición formal de trayectoria.

Definición 8.2.1 ▶

Sean v_0 y v_n vértices en una gráfica. Una *trayectoria* de v_0 a v_n de longitud n es una sucesión alternante de $n + 1$ vértices y n aristas que comienza en el vértice v_0 y termina en el vértice v_n,

$$(v_0, e_1, v_1, e_2, v_2, \ldots, v_{n-1}, e_n, v_n),$$

donde la arista e_i es incidente sobre los vértices v_{i-1} y v_i para $i = 1, \ldots, n$. ◀

El formalismo en la definición 8.2.1 significa: Comience en el vértice v_0; recorra la arista e_1 hasta v_1; siga por la arista e_2 hasta v_2, y así sucesivamente.

Ejemplo 8.2.2 ▶

En la gráfica de la figura 8.2.1,

$$(1, e_1, 2, e_2, 3, e_3, 4, e_4, 2) \qquad \textbf{(8.2.1)}$$

es una trayectoria de longitud 4 del vértice 1 al vértice 2.

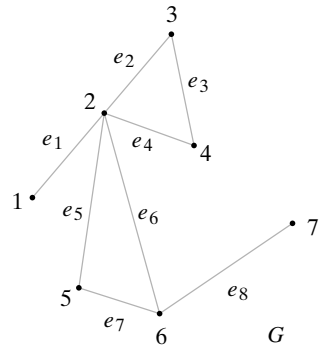


Figura 8.2.1 Grafica conexa con trayectorias $(1, e_1, 2, e_2, 3, e_3, 4, e_4, 2)$ de longitud 4 y (6) de longitud 0.

◀

Ejemplo 8.2.3 ▶ En la gráfica de la figura 8.2.1, la trayectoria (6) que consiste sólo en el vértice 6 es una trayectoria de longitud 0 del vértice 6 al vértice 6. ◀

En ausencia de aristas paralelas, al denotar una trayectoria se pueden suprimir las aristas. Por ejemplo, la trayectoria (8.2.1) también se puede escribir

$$(1, 2, 3, 4, 2).$$

Una **gráfica conexa** es una gráfica en la que se puede ir de cualquier vértice a cualquier otro vértice por una trayectoria. A continuación se da la definición formal.

Definición 8.2.4 ▶ Una gráfica G es *conexa* si dados cualesquiera dos vértices v y w en G, existe una trayectoria de v a w. ◀

Ejemplo 8.2.5 ▶ La gráfica G de la figura 8.2.1 es conexa ya que, dados cualesquiera dos vértices v y w en G, existe una trayectoria de v a w. ◀

Ejemplo 8.2.6 ▶ La gráfica G de la figura 8.2.2 no es conexa ya que, por ejemplo, no hay trayectoria del vértice v_2 al vértice v_5.

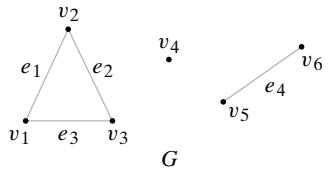


Figura 8.2.2 Gráfica no conexa. ◀

Ejemplo 8.2.7 ▶ Sea G una gráfica cuyo conjunto de vértices consiste en los 50 estados de Estados Unidos. Coloque una arista entre los estados v y w con una frontera común. Por ejemplo, hay una arista entre California y Oregon y entre Illinois y Missouri. No hay arista entre Georgia y Nueva York, como tampoco entre Utah y Nuevo México. (No cuenta si se tocan; los estados debe tener una frontera común). La gráfica G no es conexa porque no hay trayectoria de Hawaii a California (o de Hawaii a cualquier otro estado). ◀

Como se ve a partir de las figuras 8.2.1 y 8.2.2, una gráfica conexa es de una "pieza", mientras que una gráfica no conexa consiste en varias "piezas". Estas "piezas" son **subgráficas** de la gráfica original y se llaman **componentes**. Se darán las definiciones formales comenzando con la de subgráfica.

Una subgráfica G' de una gráfica G se obtiene seleccionando ciertas aristas y vértices de G sujetas a la restricción de que si se selecciona una arista e en G que incide en los vértices v y w, deben incluirse v y w en G'. La restricción asegura que G' sea de hecho una gráfica. La siguiente es la definición formal.

Definición 8.2.8 ▶ Sea $G = (V, E)$ una gráfica. (V', E') es una *subgráfica* de G si

a) $V' \subseteq V$ y $E' \subseteq E$.

b) Para toda arista $e' \in E'$, si e' incide en v' y w', entonces $v', w' \in V'$. ◀

Ejemplo 8.2.9 ▶ La gráfica $G' = (V', E')$ de la figura 8.2.3 es una subgráfica de la gráfica $G = (V, E)$ de la figura 8.2.4 puesto que $V' \subseteq V$ y $E' \subseteq E$.

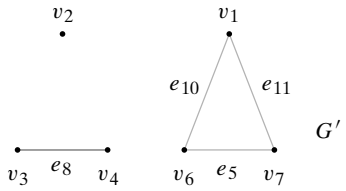


Figura 8.2.3 Una subgráfica de la gráfica de la figura 8.2.4.

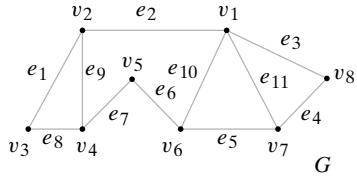


Figura 8.2.4 Gráfica, una de cuyas subgráficas aparece en la figura 8.2.3.

◀

Ejemplo 8.2.10 ▶

Encuentre todas las subgráficas de la gráfica G de la figura 8.2.5 que tengan al menos un vértice.

Si no se seleccionan aristas, se puede seleccionar uno o los dos vértices, lo que lleva a las subgráficas G_1, G_2 y G_3 mostradas en la figura 8.2.6. Si se selecciona la única arista disponible e_1, deben seleccionarse los dos vértices en los que e_1 es incidente. En este caso, se obtiene la subgráfica G_4 de la figura 8.2.6. Entonces G tiene las cuatro subgráficas mostradas en la figura 8.2.6.

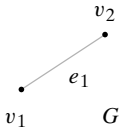


Figura 8.2.5 Gráfica para el ejemplo 8.2.10.

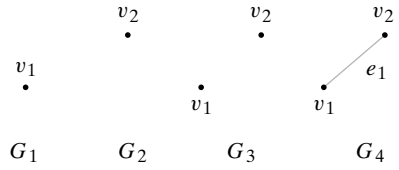


Figura 8.2.6 Las cuatro subgráficas de la gráfica de la figura 8.2.5.

◀

Ahora se definirá "componente".

Definición 8.2.11 ▶

Sea G una gráfica y sea v un vértice en G. La subgráfica G' de G que consiste en todas las aristas y vértices de G que están contenidos en alguna trayectoria que comienza en v se llama *componente* de G que contiene a v. ◀

Ejemplo 8.2.12 ▶

La gráfica G de la figura 8.2.1 tiene una componente, a saber, ella misma. Sin duda, una gráfica es conexa si y sólo si tiene exactamente una componente. ◀

Ejemplo 8.2.13 ▶

Sea G la gráfica de la figura 8.2.2. La componente de G que contiene a v_3 es la subgráfica

$$G_1 = (V_1, E_1), \qquad V_1 = \{v_1, v_2, v_3\}, \qquad E_1 = \{e_1, e_2, e_3\}.$$

La componente de G que contiene a v_4 es la subgráfica

$$G_2 = (V_2, E_2), \qquad V_2 = \{v_4\}, \qquad E_2 = \emptyset.$$

La componente de G que contiene a v_5 es la subgráfica

$$G_3 = (V_3, E_3), \qquad V_3 = \{v_5, v_6\}, \qquad E_3 = \{e_4\}.$$

◀

Otra caracterización de las componentes de una gráfica $G = (V, E)$ se obtiene al definir una relación R en el conjunto de vértices V mediante la regla

$$v_1 R v_2 \text{ si hay una trayectoria de } v_1 \text{ a } v_2.$$

Es posible demostrar (ejercicio 68) que R es una relación de equivalencia en V y que si $v \in V$, el conjunto de vértices en la componente que contiene a v es la clase de equivalencia

$$[v] = \{w \in V \mid wRv\}.$$

Observe que la definición de "trayectoria" permite repeticiones de vértices o aristas, o ambos. En la trayectoria (8.2.1), el vértice 2 aparece dos veces.

Las subclases de trayectorias se obtienen prohibiendo vértices o aristas duplicados o haciendo idénticos los vértices v_0 y v_n de la definición 8.2.1.

Definición 8.2.14 ▶

Sean v y w vértices en una gráfica G.

Una *trayectoria simple* de v a w es una ruta de v a w sin vértices repetidos.

Un *ciclo* (o *circuito*) es una trayectoria de longitud diferente de cero de v a v sin aristas repetidas.

Un *ciclo simple* es un ciclo de v a v en el que no hay vértices repetidos, excepto por el inicio y el fin que son iguales a v. ◀

Ejemplo 8.2.15 ▶

Para la gráfica de la figura 8.2.1, se tiene la siguiente información.

Trayectoria	*¿Trayectoria simple?*	*¿Ciclo?*	*¿Ciclo simple?*
(6, 5, 2, 4, 3, 2, 1)	No	No	No
(6, 5, 2, 4)	Sí	No	No
(2, 6, 5, 2, 4, 3, 2)	No	Sí	No
(5, 6, 2, 5)	No	Sí	Sí
(7)	Sí	No	No

◀

Ahora se examina de nuevo el problema introducido en la sección 8.1 de encontrar en una gráfica un ciclo que recorre cada arista exactamente una vez.

Ejemplo 8.2.16 ▶

Problema de los puentes de Königsberg

El primer artículo referente a la teoría de gráficas fue de Leonhard Euler en 1736. El artículo presentaba una teoría general que incluía una solución a lo que ahora se llama el problema de los puentes de Königsberg.

Dos islas en el río Pregel en Königsberg (ahora Kaliningrado en Rusia) estaban conectadas entre sí y con las orillas de río por puentes, como se aprecia en la figura 8.2.7. El problema es comenzar en cualquier lugar: A, B, C o D; cruzar cada puente exactamente una vez; luego regresar al lugar de inicio.

WWW

La configuración de los puentes se puede modelar como una gráfica, como se ve en la figura 8.2.8. Los vértices representan los lugares y las aristas representan los puentes. El problema de los puentes de Königsberg ahora se reduce a encontrar un ciclo en la gráfica de la figura 8.2.8 que incluya todas las aristas y todos los vértices. En honor a Euler, un ciclo en una gráfica G que incluye todas las aristas y todos los vértices de G se llama **ciclo de Euler**†. A partir del análisis en la sección 8.1, se ve que no hay un ciclo de Euler en la gráfica de la figura 8.2.8 porque hay un número impar de aristas incidentes en el vértice A. (De hecho, en la gráfica de la figura 8.2.8, cada vértice es incidente en un número impar de aristas).

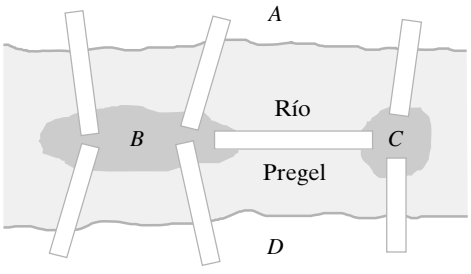


Figura 8.2.7 Los puentes de Königsberg.

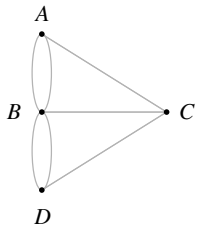


Figura 8.2.8 Gráfica que modela los puentes de Königsberg.

◀

† Por razones técnicas, si G consiste en un vértice v sin aristas, llamamos a la trayectoria (v) un ciclo de Euler para G.

La solución a la existencia de ciclos de Euler se establece mejor con la introducción del grado de un vértice. El **grado de un vértice** v, $\delta(v)$, es el número de aristas que inciden en v. (Por definición, cada ciclo sobre v contribuye con 2 al grado de v). En la sección 8.1 se encontró que si una gráfica G tiene un ciclo de Euler, entonces todo vértice en G tiene grado par. También es posible probar que G es conexa.

Teorema 8.2.17

Si una gráfica G tiene un ciclo de Euler, entonces G es conexa y todo vértice tiene grado par.

Demostración Suponga que G tiene un ciclo de Euler. Se argumentó en la sección 8.1 que todo vértice en G tiene grado par. Si v y w son vértices en G, la porción del ciclo de Euler que lleva de v a w sirve como trayectoria de v a w. Por lo tanto, G es conexa.

El inverso del Teorema 8.2.17 también es cierto. Se da una prueba por inducción matemática que ideó [Fowler].

Teorema 8.2.18

Si G es una gráfica conexa y cada vértice tiene grado par, entonces G tiene un ciclo de Euler.

Demostración La prueba se hace por inducción sobre el número n de aristas en G.

Paso básico ($n = 0$)

Como G es conexa, si G no tiene aristas, G consiste en un solo vértice. Un ciclo de Euler consiste en un solo vértice, sin aristas.

Paso inductivo

Suponga que G tiene n aristas, $n > 0$, y que cualquier gráfica conexa con k aristas, $k < n$, donde todos los vértices tienen grado par tiene un ciclo de Euler.

Es directo verificar que una gráfica conexa con uno o dos vértices, cada uno con grado par, tiene un ciclo de Euler (vea el ejercicio 69); entonces, se supone que la gráfica tiene al menos tres vértices.

Como G es conexa, hay vértices v_1, v_2 y v_3 en G con la arista e_1 incidente en v_1 y v_2 y la arista e_2 que incide en v_2 y v_3. Se eliminan las aristas e_1 y e_2 pero no los vértices, y se agrega la arista e incidente en v_1 y v_3 para obtener la gráfica G' [vea la figura 8.2.9*a)*]. Observe que cada componente de la gráfica G' tiene menos de n aristas y que en cada componente de la gráfica G', todos los vértices tienen grado par. Se demostrará que G' tiene ya sea una o dos componentes.

Sea v un vértice. Como G es conexa, existe una trayectoria P en G de v a v_1. Sea P' una porción de la trayectoria P que comienza en v cuyas aristas también están en G'. Ahora bien, P' termina en v_1, v_2 o v_3 porque la única manera de que P no sea una trayectoria en G' es que P contenga una de las aristas eliminadas e_1 o e_2. Si P' termina en v_1, entonces v está en la misma componente que v_1 en G'. Si P' termina en v_3 [vea figura 8.2.9*b)*], entonces v está en la misma componente que v_3 en G', que está en la misma componente que v_1 en G' (ya que la arista e en G' incide en v_1 y v_3). Si P' termina en v_2, entonces v_2 está en la misma componente que v. Por lo tanto, cualquier vértice en G' está en la misma componente que v_1 o que v_2. Entonces, G' tiene una o dos componentes.

Si G' tiene una componente, es decir, si G' es conexa, se aplica la hipótesis inductiva para concluir que G' tiene un ciclo de Euler C'. Este ciclo de Euler se puede modificar para producir un ciclo de Euler en G. Simplemente se sustituye la ocurrencia de la arista e en C' por las aristas e_1 y e_2.

Suponga que G' tiene dos componentes [vea la figura 8.2.9*c)*]. Por la hipótesis inductiva, la componente que contiene a v_1 tiene un ciclo de Euler C' y la componente que contiene a v_2 tiene un ciclo de Euler C'' que comienza y termina en v_2. Un ciclo de Euler en G se obtiene al modificar C' sustituyendo (v_1, v_3) en C' por (v_1, v_2) seguido de C'' seguido de (v_2, v_3) o sustituyendo (v_3, v_1) en C' por (v_3, v_2) seguido de C'' seguido de (v_2, v_1). El paso inductivo queda completo. G es un ciclo de Euler.

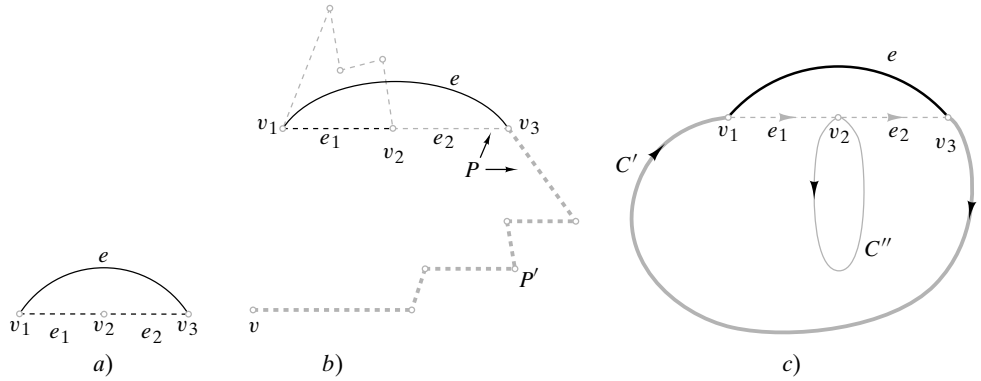


Figura 8.2.9 La prueba del Teorema 8.2.18. En *a)*, se eliminan las aristas e_1 y e_2 y se agrega la arista e. En *b)*, P (discontinua clara) es una trayectoria en G de v a v_1, y P' (discontinua gruesa) es la porción de P que inicia en v cuyas aristas también están en G'. Como se muestra, P' termina en v_3. Como la arista e está en G, hay una trayectoria en G' de v a v_1. Entonces v y v_1 están en la misma componente. En *c)*, C' es un ciclo de Euler para una componente, y C'' es un ciclo de Euler para la otra componente. Si se sustituye e en C' por e_1, C'', e_2, se obtiene un ciclo de Euler para G.

Si G es una gráfica conexa y todo vértice tiene grado par y G tiene sólo unas cuantas aristas, casi siempre se encuentra un ciclo de Euler por inspección.

Ejemplo 8.2.19 ▶

Sea G la gráfica de la figura 8.2.10. Use el Teorema 8.2.18 para verificar que G tiene un ciclo de Euler. Encuentre un ciclo de Euler para G.

Se observa que G es conexa y que

$$\delta(v_1) = \delta(v_2) = \delta(v_3) = \delta(v_5) = 4, \qquad \delta(v_4) = 6, \qquad \delta(v_6) = \delta(v_7) = 2.$$

Como el grado de cada vértice es par, por el Teorema 8.2.18, G tiene un ciclo de Euler. Por inspección, se encuentra el ciclo de Euler

$$(v_6, v_4, v_7, v_5, v_1, v_3, v_4, v_1, v_2, v_5, v_4, v_2, v_3, v_6). \qquad \blacktriangleleft$$

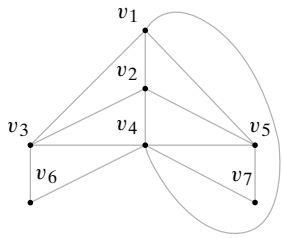


Figura 8.2.10 Gráfica para el ejemplo 8.2.19.

Ejemplo 8.2.20 ▶

Una ficha de dominó es un rectángulo dividido en dos cuadrados, cada uno con un número comprendido entre el 0 y el 6 (vea la figura 8.2.11). Los dos cuadrados en una misma ficha pueden tener el mismo número. Se demuestra que las diferentes fichas se pueden arreglar en un círculo de manera que las fichas que se tocan tengan cuadrados adyacentes con números idénticos.

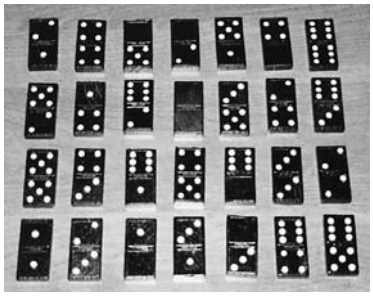


Figura 8.2.11 Dominó. [*Foto del autor*].

Se modela la situación como una gráfica G con siete vértices etiquetados 0,1, . . . , 6. Las aristas representan las fichas. Hay una arista entre cada par de vértices y hay un ciclo en cada vértice. Note que G es conexa. Ahora las fichas se pueden arreglar en un círculo de manera que las que se tocan tengan cuadrados adyacentes con números idénticos si y sólo

si G contiene un ciclo de Euler. Como el grado de cada vértice es 8 (recuerde que un ciclo contribuye con 2 al grado), cada vértice tiene grado par. Por el teorema 8.2.18, G tiene un ciclo de Euler. Por lo tanto, las fichas de dominó se pueden arreglar en círculo de manera que las que se tocan tengan cuadrados adyacentes con números idénticos. ◀

¿Qué se podría decir de una gráfica conexa en la que no todos los vértices tienen grado par? La primera observación (corolario 8.2.22) es que existe un número par de vértices con grado impar. Esto se deriva del hecho de que (Teorema 8.2.21) la suma de todos los grados en una gráfica es un número par.

Teorema 8.2.21

Si G es una gráfica con m aristas y vértices $\{v_1, v_2, \ldots, v_n\}$, entonces

$$\sum_{i=1}^{n} \delta(v_i) = 2m.$$

En particular, la suma de los grados de todos los vértices en una gráfica es par.

Demostración Cuando se suman los grados de todos los vértices, se cuenta cada arista (v_i, v_j) dos veces: una cuando se cuenta como (v_i, v_j) en los grados de v_i y de nuevo cuando se cuenta como (v_j, v_i) en los grados de v_j. Esto lleva a la conclusión.

Corolario 8.2.22

En una gráfica, existe un número par de vértices de grado impar.

Demostración Se dividen los vértices en dos grupos: los de grado par $x_1, \ldots, x_m$ y los de grado impar $y_1, \ldots, y_n$. Sea

$$S = \delta(x_1) + \delta(x_2) + \cdots + \delta(x_m), \qquad T = \delta(y_1) + \delta(y_2) + \cdots + \delta(y_n).$$

Por el Teorema 8.2.21, $S + T$ es par. Como S es la suma de números pares, S es par. Entonces T es par. Pero T es la suma de n números impares, y por lo tanto, n es par.

Suponga que una gráfica conexa G tiene exactamente dos vértices v y w de grado impar. Temporalmente se inserta una arista e de v a w. La gráfica G' que se obtiene es conexa y cada vértice tiene grado par. Por el Teorema 8.2.18, G' tiene un ciclo de Euler. Si se elimina e de este ciclo de Euler, se obtiene una trayectoria sin aristas repetidas de v a w que contiene todas las aristas y vértices de G. Se demostró que si una gráfica tiene exactamente dos vértices v y w de grado impar, existe una trayectoria con aristas repetidas que contiene todas las aristas y vértices de v a w. La demostración del inverso es similar.

Teorema 8.2.23

Una gráfica tiene una trayectoria sin aristas repetidas de v a w ($v \neq w$) que contiene todas las aristas y vértices si y sólo si es conexa y v y w son los únicos vértices que tienen grado impar.

Demostración Suponga que una gráfica tiene una trayectoria P sin aristas repetidas de v a w que contiene todas las aristas y vértices. Sin duda, la gráfica es conexa. Si se agrega una arista de v a w, la gráfica resultante tiene un ciclo de Euler, a saber, la trayectoria P junto con la arista agregada. Por el Teorema 8.2.17, cada vértice tiene grado par. Eliminar la arista agregada afecta sólo los grados de v y w, que se reducen en 1. Entonces, en la trayectoria original, v y w tienen grado impar y todos los otros vértices tienen grado par.

El inverso se analizó antes de enunciar el teorema.

En los ejercicios 42 y 44 se dan generalizaciones del Teorema 8.2.23.

Se concluye con la prueba de un resultado bastante especial que se usará en la sección 9.2.

Teorema 8.2.24

Si una gráfica G contiene un ciclo de v a v, G contiene un ciclo simple de v a v.

Demostración Sea

$$C = (v_0, e_1, v_1, \ldots, e_i, v_i, e_{i+1}, \ldots, e_j, v_j, e_{j+1}, v_{j+1}, \ldots, e_n, v_n)$$

un ciclo de v a v donde $v = v_0 = v_n$ (figura 8.2.12). Si C no es un ciclo simple, entonces $v_i = v_j$, para alguna $i < j < n$. Se puede sustituir C por el ciclo

$$C' = (v_0, e_1, v_1, \ldots, e_i, v_i, e_{j+1}, v_{j+1}, \ldots, e_n, v_n).$$

Si C' no es un ciclo simple de v a v, se repite el procedimiento anterior. En algún momento se obtiene un ciclo simple de v a v.

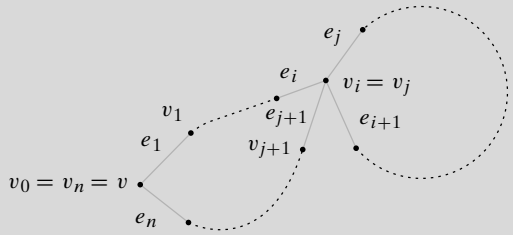


Figura 8.2.12 Ciclo que, o bien es simple, o se puede reducir a un ciclo simple.

Sección de ejercicios de repaso

1. ¿Qué es una trayectoria?
2. ¿Qué es una trayectoria simple?
3. Dé un ejemplo de una trayectoria que no sea simple.
4. ¿Qué es un ciclo?
5. ¿Qué es un ciclo simple?
6. Dé un ejemplo de un ciclo que no sea simple.
7. Defina *gráfica conexa*.
8. Dé un ejemplo de una gráfica conexa.
9. Dé un ejemplo de una gráfica no conexa.
10. ¿Qué es una subgráfica?
11. Dé un ejemplo de una subgráfica.
12. ¿Qué es una componente de una gráfica?
13. Dé un ejemplo de una componente de una gráfica.
14. Si una gráfica es conexa, ¿cuántas componentes debe tener?
15. Defina *grado de un vértice v*.
16. ¿Qué es un ciclo de Euler?
17. Establezca una condición necesaria y suficiente para que una gráfica tenga un ciclo de Euler.
18. Dé un ejemplo de una gráfica que tenga un ciclo de Euler. Especifique el ciclo de Euler.
19. Dé un ejemplo de una gráfica que *no* tenga un ciclo de Euler. Pruebe que no tiene un ciclo de Euler.
20. ¿Cuál es la relación entre la suma de los grados de los vértices en una gráfica y el número de aristas en la gráfica?
21. En una gráfica, ¿debe ser par el número de vértices de grado impar?
22. Establezca una condición necesaria y suficiente para que una gráfica tenga una trayectoria sin aristas repetidas de v a w $(v \neq w)$ que contenga todas las aristas y vértices.
23. Si una gráfica G contiene un ciclo de v a v, ¿debe G contener un ciclo simple de v a v?

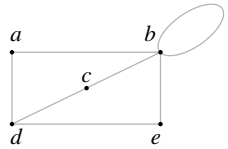
Ejercicios

En los ejercicios 1 al 9, diga si la trayectoria indicada en la gráfica es

a) una trayectoria simple,

b) un ciclo,

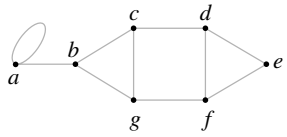
c) un ciclo simple.



1. (b, b)
2. (e, d, c, b)
3. (a, d, c, d, e)
4. (d, c, b, e, d)
5. $(b, c, d, a, b, e, d, c, b)$
6. (b, c, d, e, b, b)
7. (a, d, c, b, e)
8. (d)
9. (d, c, b)

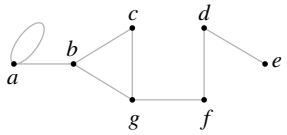
En los ejercicios 10 al 18, dibuje una gráfica que tenga las propiedades indicadas o explique por qué no existe esa gráfica.

10. Seis vértices cada uno de grado 3
11. Cinco vértices cada uno de grado 3
12. Cuatro vértices cada uno de grado 1
13. Seis vértices; cuatro aristas
14. Cuatro aristas; cuatro vértices de grados 1, 2, 3, 4
15. Cuatro vértices con grados 1, 2, 3, 4
16. Gráfica simple; seis vértices con grados 1, 2, 3, 4, 5, 5
17. Gráfica simple; cinco vértices con grados 2, 3, 3, 4, 4
18. Gráfica simple; cinco vértices con grados 2, 2, 4, 4, 4
19. Encuentre todas las trayectorias simples en la siguiente gráfica



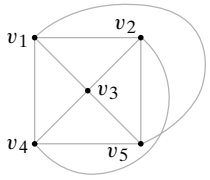
20. Encuentre todas las trayectorias simples de a a e en la gráfica del ejercicio 19.

21. Encuentre todas las subgráficas conexas de la siguiente gráfica, que contengan todos los vértices de la gráfica original y tengan tan pocas aristas como sea posible. ¿Cuáles son trayectorias simples? ¿Cuáles son ciclos? ¿Cuáles son ciclos simples?

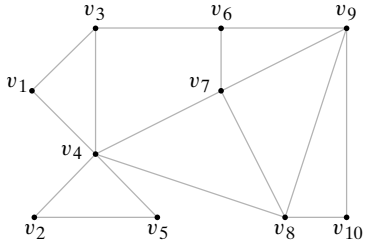


Encuentre el grado de cada vértice para las siguientes gráficas.

22.

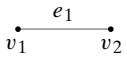


23.

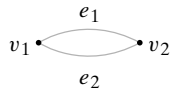


En los ejercicios 24 al 27, encuentre todas las subgráficas que tienen al menos un vértice de la gráfica dada.

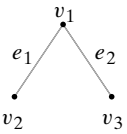
24.



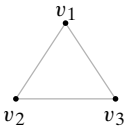
25.



26.



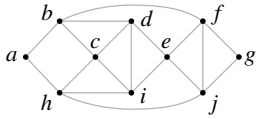
★ ***27.***



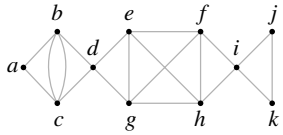
En los ejercicios 28 al 33, decida si las gráficas tienen un ciclo de Euler. Si lo tienen, muestre uno.

28. Ejercicio 21
29. Ejercicio 22
30. Ejercicio 23
31. Figura 8.2.4

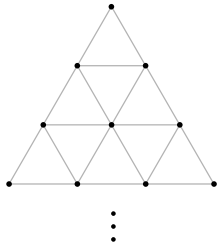
32.



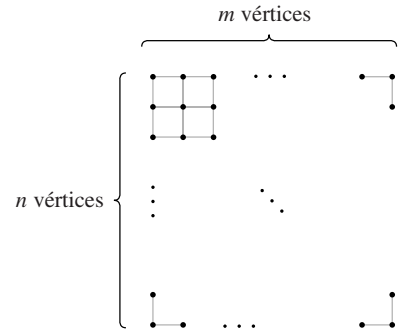
33.



34. La siguiente gráfica se continúa hasta una profundidad finita, arbitraria. ¿Contiene la gráfica un ciclo de Euler? Si la respuesta es afirmativa, describa uno.



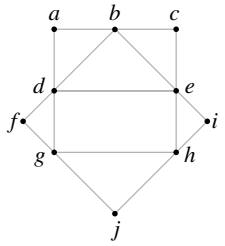
35. Una gráfica completa K_n, ¿cuándo contiene un ciclo de Euler?
36. Una gráfica bipartita $K_{m,n}$, ¿cuándo contiene un ciclo de Euler?
37. ¿Para qué valores de m y n una gráfica contiene un ciclo de Euler?



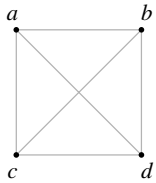
38. ¿Para qué valores de n, el cubo-n contiene un ciclo de Euler?

En los ejercicios 39 y 40, verifique que hay un número par de vértices de grado impar en la gráfica.

39.



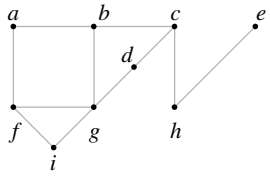
40.



41. Para la gráfica del ejercicio 39, encuentre una trayectoria sin aristas repetidas de d a e que contenga todas las aristas.

42. Sea G una gráfica conexa con cuatro vértices v_1, v_2, v_3 y v_4 de grado impar. Demuestre que existen trayectorias sin aristas repetidas de v_1 a v_2 y de v_3 a v_4 tales que cada arista en G está exactamente en una de las trayectorias.

43. Ilustre el ejercicio 42 usando la siguiente gráfica.



★

44. Establezca y pruebe una generalización del ejercicio 42 donde hay un número arbitrario de vértices de grado impar.

En los ejercicios 45 y 46, diga si cada afirmación es falsa o verdadera. Si es falsa dé un contraejemplo y si es verdadera, explique.

45. Sea G una gráfica y sean v y w vértices distintos. Si hay una trayectoria de v a w, existe una trayectoria simple de v a w.

46. Si una gráfica contiene un ciclo que incluye todas las aristas, se trata de un ciclo de Euler.

47. Sea G una gráfica conexa. Suponga que una arista e está en un ciclo. Demuestre que G con e eliminada sigue siendo conexa.

48. Dé un ejemplo de una gráfica conexa tal que la eliminación de cualquier arista produzca una gráfica no conexa. (Suponga que eliminar una arista no implica eliminar los vértices).

49. ¿Puede un caballo moverse en un tablero de ajedrez y regresar a su posición original haciendo cada movimiento exactamente una vez? (Un movimiento se considera hecho cuando se mueve en cualquiera de las dos direcciones).

50. Demuestre que si G' es una subgráfica conexa de una gráfica G, entonces G' está contenida en una componente.

51. Demuestre que si se hace una partición de un gráfica G en subgráficas conexas de manera que cada arista y cada vértice en G pertenezca a una de las subgráficas, las subgráficas son componentes.

52. Sea G una gráfica dirigida y sea G' la gráfica no dirigida que se obtiene de G si se ignora la dirección de las aristas. Suponga que G es conexa. Si v es un vértice en G, se dice que la *paridad de* v es *par* si el número de aristas de la forma (v, w) es par; la *paridad impar* se define de manera similar. Pruebe que si v y w son vértices en G que tienen paridad impar, es posible cambiar la orientación de ciertas aristas en G de manera que v y w tengan paridad par y la paridad de todos los otros vértices en G permanezca inalterable.

53. Demuestre que el número máximo de aristas en una gráfica simple, no conexa con n vértices es $(n-1)(n-2)/2$.

54. Demuestre que el número máximo de aristas en una gráfica bipartita simple con n vértices es $\lfloor n^2/4 \rfloor$.

Un vértice v en una gráfica conexa G es un punto de articulación si la eliminación de v y todas las aristas incidentes en v desconecta a G.

55. Dé un ejemplo de una gráfica con seis vértices que tenga exactamente dos puntos de articulación.

56. Dé un ejemplo de una gráfica con seis vértices que no tenga puntos de articulación.

57. Demuestre que un vértice v en una gráfica conexa G es un punto de articulación si y sólo si existen vértices w y x en G que tengan la propiedad de que toda trayectoria de w a x pasa por v.

Sea G una gráfica dirigida y sea v un vértice en G. El grado de entrada a v, $in(v)$, es el número de aristas de la forma (w, v). El grado de salida de v, $out(v)$, es el número de aristas de la forma (v, w). Un ciclo de Euler dirigido en G es una sucesión de aristas de la forma

$$(v_0, v_1), (v_1, v_2), \ldots, (v_{n-1}, v_n),$$

donde $v_0 = v_n$, cada arista en G ocurre exactamente una vez, y todos los vértices aparecen.

58. Demuestre que una gráfica dirigida G contiene un ciclo de Euler dirigido si y sólo si la gráfica no dirigida que se obtiene al ignorar la dirección de las aristas de G es conexa y además $in(v) = out(v)$ para todo vértice v en G.

★

★

Una sucesión de Bruijn para n (en ceros y unos) es una sucesión

$$a_1, \quad \ldots, \quad a_{2^n}$$

de 2^n *bits que tiene la propiedad de que si s es una cadena de bits de longitud n, para alguna m,*

$$s = a_m a_{m+1} \cdots a_{m+n-1}. \qquad \textbf{(8.2.2)}$$

En (8.2.2), se define $a_{2^n+i} = a_i$ para $i = 1, \ldots, 2^n - 1$.

59. Verifique que 00011101 es una sucesión de Bruijn para $n = 3$.

60. Sea G una gráfica dirigida con vértices correspondientes a todas las cadenas de bits de longitud $n-1$. Existe una arista dirigida del vértice $x_1 \cdots x_{n-1}$ a $x_2 \cdots x_n$. Demuestre que un ciclo de Euler dirigido corresponde a la sucesión de Bruijn.

61. Demuestre que existe una sucesión de Bruijn para cada $n = 1, 2, \ldots$.

62. Una *trayectoria cerrada* es una trayectoria de v a v. Demuestre que una gráfica conexa G es bipartita si y sólo si toda trayectoria cerrada en G tiene longitud par.

63. ¿Cuántas trayectorias de longitud $k \geq 1$ *hay en* K_n?

64. Demuestre que hay

$$\frac{n(n-1)[(n-1)^k - 1]}{n-2}$$

trayectorias cuyas longitudes están entre 1 y k, inclusive, en K_n, $n > 2$.

65. Sean v y w vértices distintos en K_n. Sea p_m el número de trayectorias de longitud m de v a w en K_n, $1 \leq m \leq n$.

a) Derive una relación de recurrencia para p_m.

b) Encuentre una fórmula explícita para p_m.

66. Sean v y w vértices distintos en K_n, $n \geq 2$. *Demuestre que el número de trayectorias simples de* v a w es

$$(n-2)!\sum_{k=0}^{n-2}\frac{1}{k!}.$$

★ **67.** [Requiere cálculo] Demuestre que hay $\lfloor n!e - 1 \rfloor$ trayectorias simples en K_n. ($e = 2.71828\ldots$ es la base del logaritmo natural).

68. Sea G una gráfica. Defina una relación R en el conjunto V de vértices de G como vRw si hay una trayectoria de v a w. Pruebe que R es una relación de equivalencia en V.

69. Pruebe que una gráfica conexa con uno o dos vértices, cada uno con grado par, tiene un ciclo de Euler.

Sea G una gráfica conexa. La distancia entre los vértices v y w en G, dist(v, w), es la longitud de una ruta más corta de v a w. El diámetro de G es

$$d(G) = máx\{dist(v, w) \mid v \text{ y } w \text{ son vértices en } G\}.$$

70. Encuentre el diámetro de la gráfica de la figura 8.2.10.

71. Encuentre el diámetro del cubo-n. En el contexto de computación en paralelo, ¿qué significa este valor?

72. Encuentre el diámetro de K_n, la gráfica completa de n vértices.

73. Demuestre que el número de trayectorias en la siguiente gráfica de v_1 a v_1 de longitud n es igual al $(n + 1)$-ésimo número de Fibonacci f_{n+1}.

74. Sea G una gráfica simple con n vértices cada uno con grado k y

$$k \geq \frac{n-3}{2} \qquad \text{si } n \bmod 4 = 1$$
$$k \geq \frac{n-1}{2} \qquad \text{si } n \bmod 4 \neq 1.$$

Demuestre que G es conexa.

Un ciclo en una gráfica dirigida simple [*es decir, una gráfica dirigida que tiene cuando mucho una arista de la forma* (v, w) *y ninguna arista de la forma* (v, v)] *es una sucesión de tres vértices o más*

$$(v_0, v_1, \ldots, v_n)$$

en la que (v_{i-1}, v_i) *es una arista para* $i = 1, \ldots, n$ *y* $v_0 = v_n$. *Una gráfica acíclica dirigida (gad) es una gráfica dirigida simple sin ciclos.*

75. Demuestre que una gad tiene el menos un vértice sin aristas que salen [es decir, hay al menos un vértice v tal que no tiene aristas de la forma (v, w)].

76. Demuestre que el número máximo de aristas en una gad de n vértices es $n(n-1)/2$.

77. Un *conjunto independiente* en una gráfica G es un subconjunto S de los vértices de G que tienen la propiedad de que hay dos vértices adyacentes en S. (Observe que $\emptyset$ es un conjunto independiente para cualquier gráfica). Pruebe el siguiente resultado que se debe a [Prodinger].

Sea P_n la gráfica que consiste en una trayectoria simple con n vértices. Pruebe que el número de conjuntos independientes en P_n es igual a f_{n+2}, $n = 1, 2, \ldots$, donde $\{f_n\}$ es la sucesión de Fibonacci.

78. Sea G una gráfica. Suponga que para cada par de vértices distintos v_1 y v_2 en G, existe un vértice único w en G tal que v_1 y w son adyacentes y v_2 y w son adyacentes.

a) Pruebe que si v y w son vértices no adyacentes en G, entonces $\delta(v) = \delta(w)$.

b) Pruebe que si existe un vértice con grado $k > 1$ y no hay vértices adyacentes a los otros vértices, entonces el grado de cada vértice es k.

Rincón de solución de problemas

Gráficas

Problemas

¿Será posible que en un departamento de 25 personas, abrumado por el desacuerdo, cada persona se lleve bien con exactamente cinco personas?

Cómo atacar el problema

¿Por dónde empezar? Como este problema está en el capítulo 8, que trata de gráficas, tal vez fuera una buena idea intentar modelarlo como una gráfica. Si el problema no estuviera asociado con una sección o capítulo específico del libro, quizá intentaríamos varios enfoques, uno de los cuales podría ser modelarlo como una gráfica. Muchos problemas discretos se resuelven usando gráficas. Esto no quiere decir que éste sea el único enfoque posible. Muchas veces al tomar enfoques diferentes, el mismo problema se puede resolver de varias maneras. (Un buen ejemplo se encuentra en [Wagon]).

Cómo encontrar una solución

Un aspecto fundamental al construir un modelo de gráficas es averiguar de qué gráfica se trata, es decir, cuáles son los vértices y cuáles las aristas. En este problema no hay muchas opciones; existen personas y desacuerdos. Se intentará haciendo que los vértices sean personas. Es muy común que en un modelo de gráficas las aristas indiquen una *relación* entre los vértices. En este caso la relación es "se lleva bien con", de manera que se pondrá una arista entre dos vértices (personas) si se llevan bien.

Ahora suponga que cada persona se lleva bien con exactamente otras cinco. Por ejemplo, en la figura que sigue, que muestra parte de la gráfica, Jeremías se lleva con Samanta, Alejandra, Laura, Berenice y Teodoro, y nadie más.

Se deduce que el grado de cada vértice es 5. Ahora se revisará la situación: se tienen 25 vértices y cada uno tiene grado 5. Antes de seguir leyendo, intente determinar si esto es posible.

El corolario 8.2.22 dice que existe un número par de vértices de grado impar. Se tiene una contradicción porque hay un número *impar* de vértices con grado impar. Por lo tanto, no es posible en un departamento con 25 personas abrumadas por el desacuerdo que cada una se lleve bien con exactamente cinco personas.

Solución formal

No. No es posible que en un departamento con 25 personas abrumadas por el desacuerdo, cada una se lleve bien con exactamente cinco personas. Suponga, a manera de contradicción, que sí es posible. Considere una gráfica donde los vértices son las personas y una arista conecta dos vértices (personas) si se llevan bien. Como cada vértice tiene grado impar, existe un número impar de vértices con grado impar, lo cual es una contradicción.

Resumen de las técnicas de solución de problemas

- Muchos problemas discretos se resuelven mediante modelos de gráficas.
- Para construir un modelo de gráficas, se determina qué representan los vértices y las aristas.
- Es muy común en un modelo de gráficas que las aristas indiquen una relación entre los vértices.

8.3 ➜ Ciclos hamiltonianos y el problema del agente viajero

Sir William Rowan Hamilton comercializó un juego a mediados del siglo XIX en la forma de un dodecaedro (vea la figura 8.3.1). Cada esquina tiene el nombre de una ciudad y el problema era comenzar en cualquier ciudad, viajar por las aristas, visitar cada ciudad justo una vez y regresar a la ciudad inicial. La gráfica de las aristas del dodecaedro se reproduce en la figura 8.3.2. El juego de Hamilton se resuelve si se encuentra un ciclo en la gráfica de la figura 8.3.2 que contenga cada vértice justo una vez (excepto por el vértice inicial y final, que aparece dos veces). Intente encontrar una solución antes de ver la que se da en la figura 8.3.3.

WWW

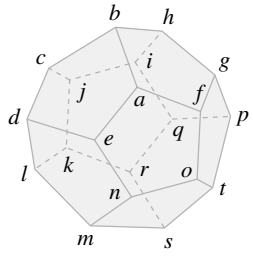


Figura 8.3.1 Juego de Hamilton.

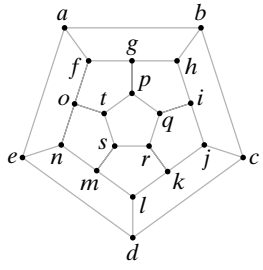


Figura 8.3.2 Gráfica del juego de Hamilton.

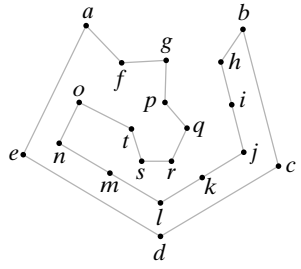


Figura 8.3.3 Recorrido de cada vértice una vez en la gráfica de la figura 8.3.2.

En honor a Hamilton, un ciclo en la gráfica G que contiene cada vértice en G justo una vez, excepto por el vértice inicial y final que aparece dos veces, recibe el nombre de **ciclo hamiltoniano**.

Hamilton (1805−1865) fue uno de los académicos más importantes de Irlanda. Fue profesor de astronomía en la Universidad de Dublín, donde publicó artículos de física y matemáticas. En esta última disciplina, Hamilton es famoso por inventar los cuaterniones, una generalización del sistema de números complejos. Los cuaterniones proporcionaron la inspiración para el desarrollo del álgebra moderna abstracta. En relación con esto, Hamilton introdujo el término *vector*.

Ejemplo 8.3.1 ▶

El ciclo (*a, b, c, d, e, f, g, a*) es un ciclo hamiltoniano para la gráfica de la figura 8.3.4. ◀

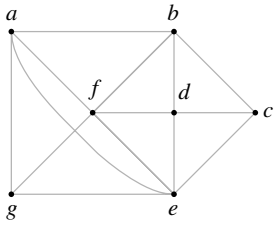


Figura 8.3.4 Gráfica con un ciclo hamiltoniano.

El problema de encontrar un ciclo hamiltoniano en una gráfica parece similar al de encontrar un ciclo de Euler en una gráfica. Un ciclo de Euler visita cada arista una vez, mientras que un ciclo de Hamilton visita cada vértice una vez; sin embargo, los problemas son en realidad bastante diferentes. Por ejemplo, la gráfica *G* de la figura 8.3.4 no tiene un ciclo de Euler ya que hay vértices de grado impar, pero el ejemplo 8.3.1 muestra que *G* tiene un ciclo hamiltoniano. Todavía más, a diferencia de la situación para los ciclos de Euler (vea los teoremas 8.2.17 y 8.2.18), no se conocen condiciones necesarias y suficientes que se verifiquen con facilidad para la existencia de un ciclo de Hamilton en una gráfica.

Los siguientes ejemplos muestran que algunas veces es posible afirmar que una gráfica no contiene un ciclo hamiltoniano.

Ejemplo 8.3.2 ▶

Muestre que la gráfica de la figura 8.3.5 no contiene un ciclo de Hamilton.

Como hay cinco vértices, un ciclo hamiltoniano debe tener cinco aristas. Suponga que se pueden eliminar aristas de la gráfica y dejar sólo el ciclo de Hamilton. Tendría que eliminarse una arista incidente en v_2 y una arista incidente en v_4, ya que cada vértice en un ciclo de Hamilton tiene grado 2. Pero esto deja sólo cuatro aristas que no son suficientes para un ciclo de Hamilton de longitud 5. Por lo tanto, la gráfica de la figura 8.3.5 no contiene un ciclo de Hamilton. ◀

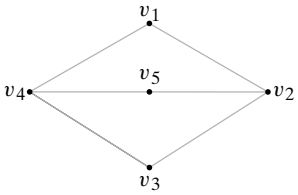


Figura 8.3.5 Una gráfica sin ciclo de Hamilton.

Debe tenerse cuidado de no contar una arista eliminada más de una vez cuando se usa un argumento como el del ejemplo 8.3.2 para probar que una gráfica no tiene un ciclo hamiltoniano. Observe en el ejemplo 8.3.2 (que se refiere a la figura 8.3.5) que si se elimina una arista incidente en v_2 y una arista incidente en v_4, estas aristas son diferentes. Por lo tanto, es correcto el razonamiento de que deben eliminarse dos aristas de la gráfica de la figura 8.3.5 para producir un ciclo hamiltoniano.

Como ejemplo de doble conteo, considere el siguiente argumento *incorrecto* que pretende demostrar que la gráfica de la figura 8.3.6 no tiene un ciclo hamiltoniano. Como hay cinco vértices, un ciclo de Hamilton debe tener cinco aristas. Suponga que se pueden eliminar aristas de la gráfica de manera que se obtenga un ciclo de Hamilton. Tendrían que eliminarse dos aristas incidentes en *c* y una arista incidente en cada vértice *a, b, d* y *e*. Esto deja dos aristas, que no son suficientes para un ciclo hamiltoniano. Por lo tanto, la gráfica de la figura 8.3.6 no contiene un ciclo hamiltoniano. El error en este argumento es que si se eliminan dos aristas incidentes en *c* (como debe hacerse), también se eliminan aristas incidentes en dos de *a, b, d* o *e*. No deben contarse otra vez las dos aristas eliminadas incidentes en los dos vértices. Observe que la gráfica de la figura 8.3.6 sí tiene un ciclo de Hamilton.

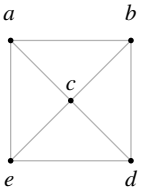


Figura 8.3.6 Gráfica *con* un ciclo hamiltoniano.

Ejemplo 8.3.3 ▶

Demuestre que la gráfica *G* de la figura 8.3.7 no contiene un ciclo hamiltoniano.

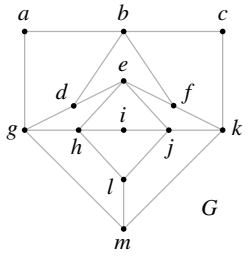


Figura 8.3.7 Gráfica sin ciclo de Hamilton.

Suponga que *G* tiene un ciclo de Hamilton *H*. Las aristas (a, b), (a, g), (b, c) y (c, k) deben estar en *H* puesto que cada vértice tiene grado 2. Entonces las aristas (b, d) y (b, f) no están en *H*. Por lo tanto, la aristas (g, d), (d, e), (e, f) y (f, k) están en *H*. Las aristas que, ahora se sabe

que están en *H,* forman un ciclo *C*. Al agregar una arista adicional a *C* se obtiene un vértice en *H* con grado mayor que 2. Esta contradicción muestra que *G* no tiene ciclos de Hamilton. ◀

El **problema del agente viajero** se relaciona con el problema de encontrar un ciclo hamiltoniano en una gráfica. (Se hizo una referencia breve a una variante del problema del agente viajero en la sección 8.1). El problema es: Dada una gráfica ponderada *G*, encuentre en *G* un ciclo de Hamilton con longitud mínima. Si se piensa en los vértices de una gráfica ponderada como ciudades y en los pesos de las aristas como distancias, el problema del agente viajero consiste en encontrar una ruta más corta en la que el agente viajero pueda visitar cada ciudad una vez, comenzando y terminando en la misma ciudad.

Ejemplo 8.3.4 ▶

El ciclo $C = (a, b, c, d, a)$ es un ciclo hamiltoniano para la gráfica *G* de la figura 8.3.8. Al sustituir cualquiera de las aristas en *C* por cualquiera de las aristas con etiqueta 11 aumentaría la longitud de *C*; entonces *C* es un ciclo hamiltoniano de longitud mínima para *G*. Así, *C* resuelve el problema del agente viajero para *G*.

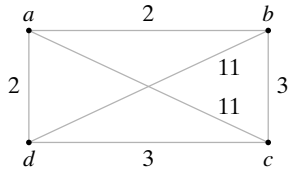


Figura 8.3.8 Gráfica para el problema del agente viajero.

◀

Aunque existen algoritmos (vea por ejemplo [Even, 1979]) para encontrar un ciclo de Euler, si existe, en un tiempo $\Theta(n)$ para una gráfica con *n* aristas, todos los algoritmos conocidos para encontrar ciclos de Hamilton requieren un tiempo ya sea exponencial o factorial en el peor caso. Por esta razón, los métodos que producen ciclos cercanos a la longitud mínima se usan con frecuencia en problemas que piden una solución al problema del agente viajero. La fama instantánea espera al descubridor de un algoritmo en tiempo polinomial para resolver el problema del ciclo de Hamilton (o del agente viajero) o de una demostración de que no existe un algoritmo en tiempo polinomial para estos problemas.

Se concluye esta sección con el estudio de ciclos hamiltonianos en el cubo-*n*.

Ejemplo 8.3.5 ▶

Códigos Gray y ciclos hamiltonianos en el cubo-*n*

Considere el **modelo de anillo** para la computación en paralelo que, al representarse por una gráfica, es un ciclo simple (figura 8.3.9). Los vértices representan procesadores. Una arista entre los procesadores *p* y *q* indica que *p* y *q* se pueden comunicar directamente entre sí. Se ve que cada procesador se puede comunicar directamente con exactamente otros dos procesadores. Los procesadores no adyacentes se comunican enviando mensajes.

WWW

El cubo-*n* (vea el ejemplo 8.1.7) es otro modelo para computación en paralelo. El cubo-*n* tiene un grado mayor de conectividad entre sus procesadores. Se considera el aspecto de cuándo un cubo-*n* puede simular un modelo de anillo con 2^n procesadores. En la terminología de gráficas, la pregunta es cuándo un cubo-*n* contiene un ciclo simple de 2^n vértices como una subgráfica o, dado que el cubo-*n* tiene 2^n procesadores, cuándo contiene un ciclo de Hamilton. [Se deja como ejercicio la pregunta de cuándo un cubo-*n* puede simular un modelo de anillo con un número arbitrario de procesadores(vea el ejercicio 18)].

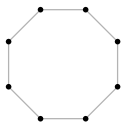


Figura 8.3.9 El modelo de anillo para la computación en paralelo.

Primero se observa que si el cubo-*n* contiene un ciclo de Hamilton, debe tenerse $n \geq 2$ puesto que el cubo-1 no tiene ciclos.

Recuerde (vea el ejemplo 8.1.7) que podemos etiquetar los vértices del cubo-n $0,1, \ldots, 2^n - 1$ de tal manera que una arista se conecte con dos vértices si y sólo si la representación binaria de sus etiquetas difiere en exactamente un bit. Entonces el cubo-*n* tiene

un ciclo hamiltoniano si y sólo si $n \geq 2$ *y existe una sucesión,*

$$s_1, \quad s_2, \quad \ldots, \quad s_{2^n} \tag{8.3.1}$$

donde cada s_i es una cadena de n bits que satisface:

- Toda cadena de n bits aparece en algún lugar de la sucesión.
- s_i y s_{i+1} difieren exactamente en un bit, $i = 1, \ldots, 2^n - 1$.
- s_{2^n} y s_1 difieren exactamente en un bit.

Una sucesión (8.3.1) se llama **código Gray**. Cuando $n \geq 2$, un código Gray (8.3.1) corresponde al ciclo hamiltoniano

$$s_1, \quad s_2, \quad \ldots, \quad s_{2^n}, \quad s_1$$

ya que cada vértice aparece y las aristas (s_i, s_{i+1}), $i = 1, \ldots, 2^n - 1$, y (s_{2^n}, s_1) son diferentes. Cuando $n = 1$, el código Gray 0, 1 corresponde a la trayectoria (0, 1, 0), que no es un ciclo porque la arista (0, 1) se repite.

El código Gray ha sido objeto de extensos estudios en otros contextos. Por ejemplo, se han usado códigos Gray al convertir información analógica en digital (vea [Deo]). Se mostrará cómo construir un código Gray para cada entero positivo n, lo que prueba que el cubo-n tiene un ciclo de Hamilton para todo entero positivo $n \geq 2$. ◀

Teorema 8.3.6

Sea G_1 la sucesión 0, 1. *Se define G_n en términos de G_{n-1} de acuerdo con las siguientes reglas:*

a) Sea G_{n-1}^R la sucesión G_{n-1} escrita al revés.
b) Sea G_{n-1}' la sucesión obtenida al colocar un 0 *como prefijo en cada miembro de G_{n-1}.*
c) Sea G_{n-1}'' la sucesión obtenida al colocar un 1 *como prefijo en cada miembro de G_{n-1}^R.*
d) Sea G_n la sucesión formada por G_{n-1}' seguida de G_{n-1}''.

Entonces G_n es un código Gray para cada entero positivo n.

Demostración Se prueba el teorema por inducción sobre n.

Paso base ($n = 1$)

Como la sucesión 0, 1 es un código Gray, el teorema se cumple cuando n es 1.

Paso inductivo

Suponga que G_{n-1} es un código Gray. Cada cadena en G_{n-1}' comienza con 0, de manera que cualquier diferencia entre cadenas consecutivas debe ser el resultado de bits distintos en las cadenas correspondientes en G_{n-1}. Pero como G_{n-1} es un código Gray, cada par consecutivo de cadenas en G_{n-1} difiere exactamente en un bit. Por lo tanto, cada par consecutivo de cadenas en G_{n-1}' difiere exactamente en un bit. De manera similar, cada par consecutivo de cadenas en G_{n-1}'' difiere exactamente en un bit.

Sea α la última cadena en G_{n-1}', y sea β la primera cadena en G_{n-1}''. Si se elimina el primer bit de α y el primer bit de β, las cadenas obtenidas son idénticas. Como el primer bit en α es 0 y el primer bit en β es 1, la última cadena en G_{n-1}' y la primera cadena en G_{n-1}'' difieren exactamente en un bit. De manera similar, la primera cadena en G_{n-1}' y la última cadena en G_{n-1}'' difieren exactamente en un bit. Por lo tanto, G_n es un código Gray.

Corolario 8.3.7

El cubo-n tiene un ciclo hamiltoniano para cada entero positivo $n \geq 2$.

Ejemplo 8.3.8 ▶ Se usa el Teorema 8.3.6 para construir el código Gray de G_3 comenzando con G_1.

G_1:	0	1						
G_1^R:	1	0						
G_1':	00	01						
G_1'':	11	10						
G_2:	00	01	11	10				
G_2^R:	10	11	01	00				
G_2':	000	001	011	010				
G_2'':	110	111	101	100				
G_3:	000	001	011	010	110	111	101	100

◀

Esta sección termina con el examen de un problema que data de hace unos 200 años.

Ejemplo 8.3.9 ▶

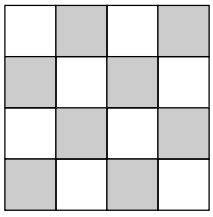
Recorrido del caballo

WWW

En ajedrez, el movimiento del caballo consiste en recorrer dos cuadros en sentido horizontal o vertical y un cuadro en dirección perpendicular. Por ejemplo, en la figura 8.3.10 un caballo en el cuadro marcado con K se puede mover a cualquiera de los cuadros marcados con X. El **recorrido del caballo de un tablero de $n \times n$** comienza en algún cuadro, visita cada cuadro exactamente una vez con movimientos permitidos, y regresa al cuadro inicial. El problema es determinar para qué valores de n existe el recorrido del caballo.

	X		X	
X				X
		K		
X				X
	X		X	

Figura 8.3.10
Movimientos permitidos para el caballo en ajedrez.



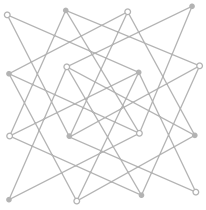


Figura 8.3.11
Tablero de ajedrez de 4×4 y la gráfica GK_4.

Se puede usar una gráfica para modelar este problema. Los cuadros del tablero, con color alternado en blanco y negro de la manera usual, son los vértices de la gráfica y hay una arista entre dos vértices si los cuadros correspondientes en el tablero representan un movimiento permitido para el caballo (figura 8.3.11). Se denota la gráfica por GK_n. Entonces hay un recorrido del caballo en el tablero de $n \times n$ si y sólo si GK_n tiene un ciclo de Hamilton.

Se demuestra que si GK_n tiene un ciclo de Hamilton, n es par. Para ver esto, observe que GK_n es bipartita. Se puede hacer una partición de los vértices en dos conjuntos: V_1 correspondiente a los cuadros blancos y V_2 correspondiente a los negros. Cada arista incide en un vértice de V_1 y uno de V_2. Como cualquier ciclo debe alternar entre un vértice en V_1 y uno en V_2, cualquier ciclo en GK_n debe tener longitud par. Pero como un ciclo hamiltoniano debe visitar cada vértice exactamente una vez, un ciclo hamiltoniano en GK_n debe tener longitud n^2. Entonces, n debe ser par.

En vista del resultado anterior, el tablero más pequeño posible que podría tener un recorrido del caballo es de 2×2, pero no tiene un recorrido porque es tan pequeño que el caballo no tendría movimientos permitidos. El siguiente tablero más pequeño es de 4×4, aunque se demostrará que tampoco tiene un recorrido del caballo.

Se da un argumento por contradicción para demostrar que GK_4 no tiene un ciclo de Hamilton. Suponga que GK_4 tiene un ciclo de Hamilton $C = (v_1, v_2, \ldots, v_{17})$. Se supone que v_1 corresponde al cuadro en la esquina superior izquierda. Los ocho cuadros en la fila superior e inferior del tablero se llaman *cuadros exteriores*, y los ocho cuadros de las filas centrales se llaman *cuadros interiores*. Observe que el caballo debe llegar a un cuadro exterior si está en uno interior y que de un cuadro exterior se mueve a uno interior. Entonces en el ciclo C, cada vértice que corresponde a un cuadro exterior debe ir precedido y seguido por un vértice que corresponde a un cuadro interior. Como hay el mismo número de cuadros exteriores e interiores, los vértices v_i donde i es impar corresponden a cuadros exteriores, y los vértices v_i donde i es par corresponden a cuadros interiores. Pero al observar los movimientos del caballo, se ve que los vértices v_i donde i es impar corresponden a cuadros blancos y los vértices v_i para i par corresponden a cuadros negros. Por lo tanto, los únicos cuadros exteriores que se visitan son blancos y los únicos cuadros interiores que se visitan son negros. Entonces C no es un ciclo de Hamilton. Esta contradicción completa la prueba de que GK_4 no tiene un ciclo de Hamilton. Este argumento lo expuso Louis Pósa cuando era adolescente.

La gráfica GK_6 tiene un ciclo de Hamilton. Este hecho se prueba simplemente exhibiendo uno (vea el ejercicio 21). Se puede probar, usando métodos elementales, que GK_n tie-

ne un ciclo de Hamilton para toda $n \geq 6$ (vea [Schwenk]). La demostración construye de manera explícita ciclos de Hamilton para ciertos tableros más pequeños y después pega los tableros pequeños para obtener los ciclos de Hamilton para tableros más grandes. ◀

Sugerencias para resolver problemas

Un *ciclo de Euler* comienza en un vértice, recorre cada *arista* exactamente una vez y regresa al vértice inicial. Los teoremas 8.2.17 y 8.2.18 permiten determinar con facilidad si una gráfica tiene un ciclo de Euler: una gráfica tiene un ciclo de Euler si y sólo si G es conexa y todo vértice tiene grado par.

Un *ciclo hamiltoniano* comienza en un vértice, visita cada *vértice* exactamente una vez (excepto por el inicial que se visita dos veces: al inicio y al final del ciclo de Hamilton) y regresa al vértice inicial. A diferencia de los teoremas 8.2.17 y 8.2.18, no se conocen condiciones necesarias y suficientes de verificación rápida para que una gráfica tenga un ciclo de Hamilton. Si una gráfica relativamente pequeña tiene un ciclo de Hamilton, la prueba y error descubrirán uno. Si una gráfica no tiene ciclos de Hamilton, algunas veces se puede usar el hecho de que un ciclo de Hamilton en una gráfica de n vértices tiene longitud n junto con la prueba por contradicción para probar que no tiene un ciclo hamiltoniano. La sección 8.3 contiene dos técnicas para pruebas por contradicción. En la primera, se supone que la gráfica tiene un ciclo de Hamilton. Ciertas aristas no pueden aparecer en el ciclo de Hamilton: si una gráfica tiene un vértice v de grado mayor que 2, sólo dos aristas incidentes en v pueden aparecer en el ciclo de Hamilton. Algunas veces se puede obtener una contradicción mostrando que con tantas aristas eliminadas en la gráfica no puede haber un ciclo de Hamilton (vea el ejercicio 8.3.2).

En la segunda técnica de prueba por contradicción mostrada en la sección 8.3, de nuevo se supone que la gráfica con n vértices tiene un ciclo de Hamilton. Después se argumenta que ciertas aristas *deben* estar en el ciclo hamiltoniano. Por ejemplo, si un vértice v tiene grado 2, ambas aristas incidentes en v deben estar en el ciclo. Algunas veces se puede obtener una contradicción mostrando que las aristas que deben estar en el ciclo de Hamilton forman un ciclo de longitud menor que n (vea el ejemplo 8.3.3).

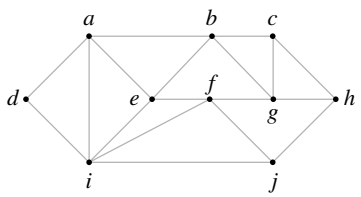
Sección de ejercicios de repaso

1. ¿Qué es un ciclo hamiltoniano?
2. Dé un ejemplo de una gráfica que tiene un ciclo de Hamilton y un ciclo de Euler. Pruebe que la gráfica tiene las propiedades especificadas.
3. Dé un ejemplo de una gráfica que tiene un ciclo de Hamilton pero no un ciclo de Euler. Pruebe que la gráfica tiene las propiedades especificadas.
4. Dé un ejemplo de una gráfica que no tiene un ciclo de Hamilton pero tiene un ciclo de Euler. Pruebe que la gráfica tiene las propiedades especificadas.
5. Dé un ejemplo de una gráfica que no tiene un ciclo hamiltoniano ni un ciclo de Euler. Pruebe que la gráfica tiene las propiedades especificadas.
6. ¿Cuál es el problema del agente viajero? ¿Cómo se relaciona con el problema del ciclo de Hamilton?
7. ¿Qué es el modelo de anillo para la computación en paralelo?
8. ¿Qué es el código Gray?
9. Explique cómo construir un código Gray.

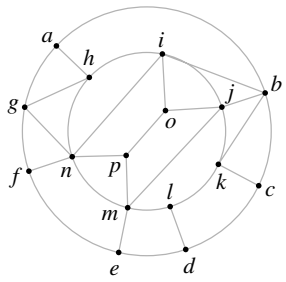
Ejercicios

Encuentre un ciclo hamiltoniano en cada gráfica.

1.

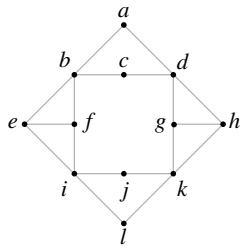


2.

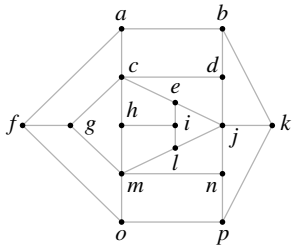


Demuestre que ninguna gráfica contiene un ciclo hamiltoniano.

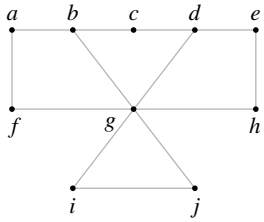
3.



4.

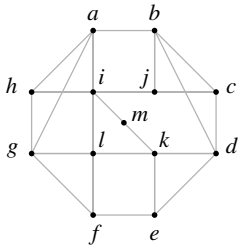


5.

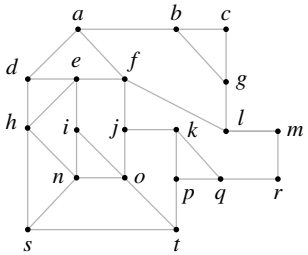


Determine si cada gráfica contiene un ciclo de Hamilton. Si así es, exhiba uno; de otra manera, dé un argumento para demostrar que no hay un ciclo de Hamilton.

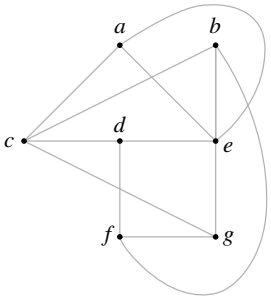
6.



7.



8.



9. Dé un ejemplo de una gráfica que tiene un ciclo de Euler pero no contenga un ciclo de Hamilton.

10. Dé un ejemplo de una gráfica que tiene un ciclo de Euler que también es un ciclo de Hamilton.

11. Dé un ejemplo de una gráfica que tiene un ciclo de Euler y un ciclo de Hamilton que no son idénticos.

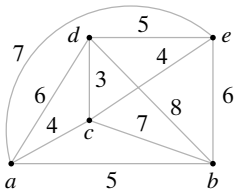
★***12.*** ¿Para qué valores de m y n la gráfica del ejercicio 37, sección 8.2, contiene un ciclo de Hamilton?

13. Modifique la gráfica del ejercicio 37, sección 8.2, insertando una arista entre el vértice en la fila i, columna 1, y el vértice en la fila i, columna m, para $i = 1, \ldots, n$. Demuestre que la gráfica obtenida siempre tiene un ciclo de Hamilton.

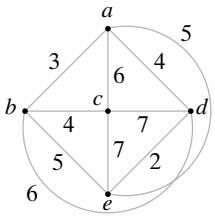
14. Demuestre que si $n \geq 3$, la gráfica completa sobre n vértices K_n contiene un ciclo hamiltoniano.

15. ¿Cuándo la gráfica completa bipartita $K_{m,n}$ contiene un ciclo hamiltoniano?

16. Demuestre que el ciclo (*e, b, a, c, d, e*) proporciona una solución al problema del agente viajero para la gráfica mostrada.



17. Resuelva el problema del agente viajero para la gráfica dada.



★***18.*** Sean m y n enteros que satisfacen $1 \leq m \leq 2^n$. Pruebe que el cubo-n tiene un ciclo simple de longitud m si y sólo si $m \geq 4$ *y* m es par.

19. Use el Teorema 8.3.6 para calcular el código Gray G_4.

20. Sea G una gráfica bipartita con conjuntos ajenos de vértices V_1 y V_2, como en la definición 8.1.11. Demuestre que si G tiene un ciclo de Hamilton, V_1 y V_2 tienen el mismo número de elementos.

21. Encuentre un ciclo de Hamilton en GK_6 (vea el ejemplo 8.3.9).

22. Describa un modelo de gráficas adecuado para resolver el siguiente problema: ¿Pueden arreglarse las permutaciones de $\{1, 2, \ldots, n\}$

en una sucesión de manera que las permutaciones adyacentes

$$p: \quad p_1, \ldots, p_n \qquad \text{y} \qquad q: \quad q_1, \ldots, q_n$$

satisfagan $p_i \neq q_i$ para $i = 1, \ldots, n$?

23. Resuelva el problema del ejercicio 22 para $n = 1, 2, 3, 4$. (La respuesta a la pregunta es "sí" para $n \geq 5$; vea [problema 1186] en las referencias).

24. Demuestre que las etiquetas consecutivas de los vértices en el círculo unitario de la descripción de Bain del cubo-n dan un código Gray (vea los ejercicios 43 a 45, sección 8.1).

Una trayectoria de Hamilton en una gráfica G es un trayectoria simple que contiene todos los vértices en G exactamente una vez. (Una trayectoria de Hamilton inicia y termina en vértices diferentes).

25. Si una gráfica tiene ciclo de Hamilton, ¿debe tener una trayectoria de Hamilton? Explique.

26. Si una gráfica tiene una trayectoria de Hamilton, ¿debe tener un ciclo de Hamilton? Explique.

27. ¿La gráfica de la figura 8.3.5 tiene una trayectoria de Hamilton?

28. ¿La gráfica de la figura 8.3.7 tiene una trayectoria de Hamilton?

29. ¿La gráfica del ejercicio 3 tiene una trayectoria de Hamilton?

30. ¿La gráfica del ejercicio 4 tiene una trayectoria de Hamilton?

31. ¿La gráfica del ejercicio 5 tiene una trayectoria de Hamilton?

32. ¿La gráfica del ejercicio 6 tiene una trayectoria de Hamilton?

33. ¿La gráfica del ejercicio 7 tiene una trayectoria de Hamilton?

34. ¿La gráfica del ejercicio 8 tiene una trayectoria de Hamilton?

35. ¿Para qué valores de m y n la gráfica del ejercicio 37, sección 8.2, tiene una trayectoria de Hamilton?

36. ¿Para qué valor de n la gráfica completa sobre n vértices tiene una trayectoria de Hamilton?

8.4 ➜ Un algoritmo de la ruta más corta

Recuerde (de la sección 8.1) que una gráfica ponderada es una gráfica en la que se asignan valores a las aristas y que la longitud de una trayectoria en una gráfica ponderada es la suma de los pesos de las aristas en la trayectoria. Sea $w(i, j)$ el peso de la arista (i, j). En las gráficas ponderadas con frecuencia se desea encontrar la **ruta más corta** (es decir, una trayectoria que tiene la longitud mínima) entre dos vértices dados. El algoritmo 8.4.1, ideado por E. W. Dijkstra, que resuelve con eficiencia este problema, es el tema de esta sección.

Edgar W. Dijkstra (1930-2002) nació en Holanda. Fue uno de los primeros en proponer la programación como una ciencia. Estaba tan dedicado a la programación que, cuando se casó en 1957, dejó asentado que su profesión era la de programador. Sin embargo, las autoridades holandesas dijeron que esa profesión no existía y tuvo que cambiar el dato a "físico teórico". Ganó el prestigioso premio Turing Award de la Association for Computing Machinery en 1972. Fue designado para presidir el Schlumberger Centennial en Ciencias de la Computación en la Universidad de Texas en Austin, en 1984, y se retiró como profesor emérito en 1999.

En esta sección, G denota una gráfica conexa ponderada. Se supone que los pesos son números positivos y que se quiere encontrar la ruta más corta del vértice a al vértice z. La suposición de que G es conexa se puede excluir (vea el ejercicio 9).

WWW El algoritmo de Dijkstra implica asignar etiquetas a los vértices. Sea $L(v)$ la etiqueta del vértice v. En cualquier punto, algunos vértices tienen etiquetas temporales y el resto son permanentes. Sea T el conjunto de vértices que tienen etiquetas temporales. Al ilustrar el algoritmo, se marcarán con un círculo los vértices que tiene etiquetas permanentes. Se demostrará después que si $L(v)$ es la etiqueta permanente del vértice v, entonces $L(v)$ es la longitud de una ruta más corta de a a v. Al inicio, todos los vértices tienen etiquetas temporales. Cada iteración del algoritmo cambia el estado de una etiqueta de temporal a permanente; entonces el algoritmo puede terminar cuando z recibe una etiqueta permanente. En este punto $L(v)$ da la longitud de la ruta más corta de a a z.

Algoritmo 8.4.1

Algoritmo de la ruta más corta de Dijkstra

Este algoritmo encuentra la longitud de una ruta más corta del vértice a al vértice z en una gráfica ponderada conexa. El peso de la arista (i, j) es $w(i, j) > 0$, y la etiqueta del vértice x es $L(x)$. Al terminar, $L(z)$ es la longitud de la ruta más corta de a a z.

Entrada: Una gráfica conexa ponderada en la que todos los pesos son positivos; vértices a a z
Salida: $L(z)$, la longitud de la ruta más corta de a a z

1. *dijkstra*(w, a, z, L) {
2. $L(a) = 0$

```
3.        para todos los vértices x ≠ a
4.            L(x) = ∞
5.        T = conjunto de todos los vértices
6.        // T es el conjunto de todos los vértices cuyas distancias más cortas desde a
7.        // no se han encontrado
8.        while(z ∈ T)  {
9.            seleccionar v ∈ T con L(v) mínimo
10.           T = T − {v}
11.           para cada x ∈ T adyacente a v
12.               L(x) = mín{L(x),L(v) + w(v, x)}
13.       }
14.   }
```

Ejemplo 8.4.2 ▶

Se mostrará la manera en que el algoritmo 8.4.1 encuentra la ruta más corta de a a z en la gráfica de la figura 8.4.1. (Los vértices en T no están marcados con círculos y tienen etiquetas temporales). La figura 8.4.2 muestra el resultado de ejecutar las líneas 2 a la 5. En la línea 8, z no tiene círculo. Se procede a la línea 9, donde se elige el vértice a, el vértice sin círculo con la etiqueta menor, y se marca con un círculo (figura 8.4.3). En las líneas 11 y 12 se actualiza cada uno de los vértices que no tienen círculo, b y f, adyacentes a a. Se obtienen las nuevas etiquetas

$$L(b) = \text{mín}\{\infty, 0 + 2\} = 2, \qquad L(f) = \text{mín}\{\infty, 0 + 1\} = 1$$

(vea la figura 8.4.3). En este punto, se regresa a la línea 8.

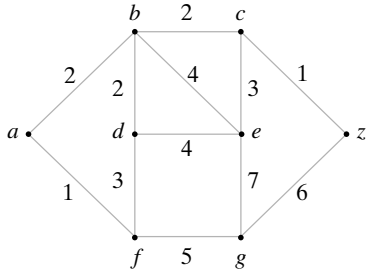


Figura 8.4.1 Gráfica para el ejemplo 8.4.2.

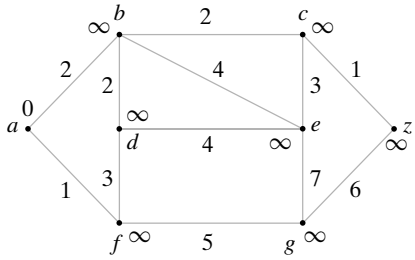


Figura 8.4.2 Inicialización del algoritmo de la ruta más corta de Dijkstra.

Como z no tiene círculo, se procede a la línea 9, donde se selecciona el vértice f, el vértice sin círculo con la etiqueta menor, y se marca con un círculo (figura 8.4.4). En las líneas 11 y 12 se actualizan las etiquetas de los vértices sin círculo, d y g, adyacentes a f. Se obtienen las etiquetas de la figura 8.4.4.

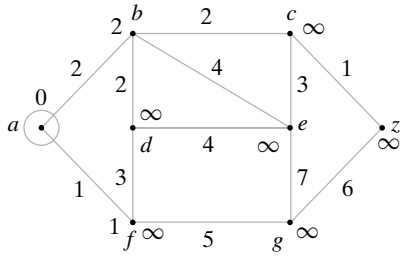


Figura 8.4.3 Primera iteración del algoritmo de la ruta más corta de Dijkstra.

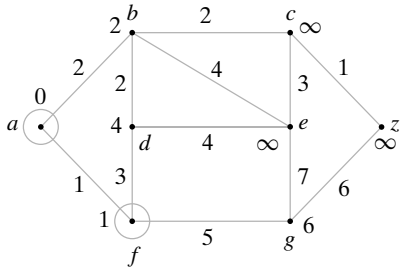


Figura 8.4.4 Segunda iteración del algoritmo de la ruta más corta de Dijkstra.

Debe verificarse que la siguiente iteración del algoritmo produce las etiquetas dadas en la figura 8.4.5 y que al terminar el algoritmo, z tiene la etiqueta 5, lo que indica que la longitud de la ruta más corta de a a z es 5. Una ruta más corta está dada por (a, b, c, z).

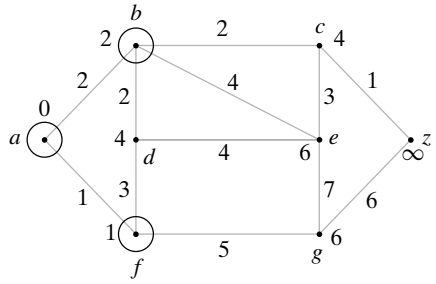


Figura 8.4.5 Tercera iteración del algoritmo de la ruta más corta de Dijkstra.

◀

Ahora se demostrará que el algoritmo 8.4.1 es correcto. La prueba se relaciona con el hecho de que el algoritmo de Dijkstra encuentra las longitudes de las rutas más cortas desde a en orden no decreciente.

Teorema 8.4.3

El algoritmo de la ruta más corta de Dijkstra (algoritmo 8.4.1) encuentra correctamente la longitud de una ruta más corta desde a hasta z.

Demostración Se usa inducción matemática sobre i para probar que la i-ésima vez que se lleva a la línea 9, $L(v)$ es la longitud de una ruta más corta de a a v. Cuando se prueba esto, se deduce que el algoritmo es correcto porque cuando se elige z en la línea 9, $L(z)$ dará la longitud de una ruta más corta de a a z.

Paso base ($i = 1$)

La primera vez que se llega a la línea 9, a causa de los pasos de inicialización (líneas 2 a la 4), $L(a)$ es cero y todos los otros valores de L son ∞. Entonces a se selecciona la primera vez que se llega la línea 9. Como $L(a)$ es cero, $L(a)$ es la longitud de una ruta más corta de a a a.

Paso inductivo

Suponga que para toda $k < i$, la k-ésima vez que se llega a la línea 9, $L(v)$ es la longitud de una ruta más corta de a a v.

Suponga que es la i-ésima vez que estamos en la línea 9 y que se elige v en T con valor mínimo de $L(v)$.

Primero se demuestra que si existe una trayectoria de a a un vértice w cuya longitud es menor que $L(v)$, entonces w no está en T (es decir, w ya se había seleccionado en la línea 9). Suponga a manera de contradicción que w está en T. Sea P una ruta más corta de a a w, sea x el vértice más cercano a a en P que está en T, y sea u el predecesor de x en P (vea la figura 8.4.6). Entonces u no está en T, de manera que u ya se había seleccionado en la línea 9 en una iteración anterior del ciclo "while". Por la suposición inductiva, $L(u)$ es la longitud de la ruta más corta de a a u. Ahora

$$L(x) \leq L(u) + w(u, x) \quad \leq \text{longitud de } P < L(v).$$

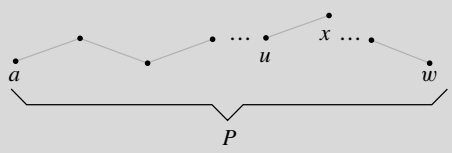


Figura 8.4.6 Prueba del Teorema 8.4.3. P es la ruta más corta de a a w, x es el vértice más cercano a a en P que está en T, y u es el predecesor de x en P.

Pero esta desigualdad muestra que v no es el vértice en T con $L(v)$ mínimo [$L(x)$ es menor]. Esta contradicción completa la prueba de que si hay una trayectoria de a al vértice w cuya longitud es menor que $L(v)$, entonces w no está en T.

El resultado anterior muestra, en particular, que si hubiera una trayectoria de a a v cuya longitud fuera menor que $L(v)$, v ya se habría seleccionado en la línea 9 y eliminado de T. Por lo tanto, cada ruta de a a v tiene longitud al menos de $L(v)$. Por construcción, existe una ruta de a a v de longitud $L(v)$, de manera que ésta es una ruta más corta de a a v. La prueba queda completa.

El algoritmo 8.4.1 encuentra la longitud de la ruta más corta de a a z. En la mayoría de las aplicaciones, también se desea identificar una ruta más corta. En pequeña modificación al algoritmo 8.4.1 muestra cómo encontrar una ruta más corta.

Ejemplo 8.4.4 ▶

Encuentre una ruta más corta de a a z y su longitud para la gráfica de la figura 8.4.7.

Se aplicará el algoritmo 8.4.1 con una ligera modificación. Además de marcar el vértice con un círculo, también se etiquetará con el nombre del vértice por el que se marca.

La figura 8.4.7 muestra el resultado de ejecutar las líneas 2 a la 4 del algoritmo 8.4.1. Primero, se pone un círculo en a (figura 8.4.8). Después, se etiquetan los vértices b y d adyacentes a a. El vértice b se etiqueta "a, 2" para indicar su valor y el hecho de que se etiqueta desde a. De manera similar, el vértice d se etiqueta "a, 1".

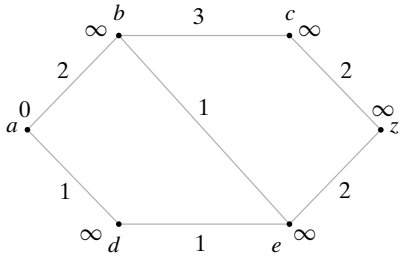


Figura 8.4.7 Inicialización del algoritmo de la ruta más corta de Dijkstra.

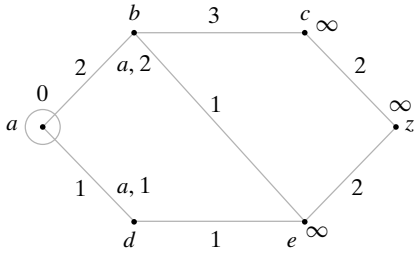


Figura 8.4.8 Primera iteración del algoritmo de la ruta más corta de Dijkstra.

Luego, se pone un círculo en el vértice d y se actualiza la etiqueta del vértice e adyacente a d (vea la figura 8.4.9). Después se pone un círculo en el vértice b y se actualizan las etiquetas de los vértices c y e (vea la figura 8.4.10). Se pone un círculo en el vértice e y se actualiza la etiqueta del vértice z (figura 8.4.11). En este punto, puede ponerse un círculo en z, y el algoritmo termina. La longitud de una ruta más corta de a a z es 4. Comenzando con z, nos movemos hacia atrás por las etiquetas para encontrar la ruta más corta.

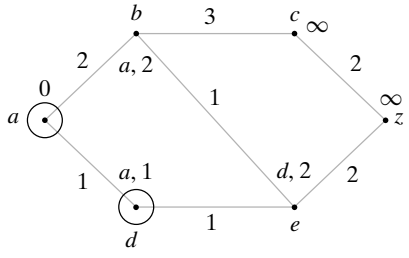


Figura 8.4.9 Segunda iteración del algoritmo de la ruta más corta de Dijkstra.

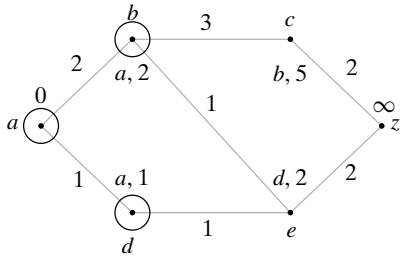


Figura 8.4.10 Tercera iteración del algoritmo de la ruta más corta de Dijkstra.

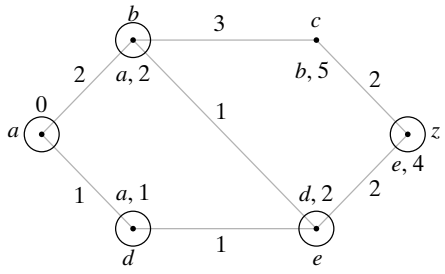


Figura 8.4.11 Conclusión del algoritmo de la ruta más corta de Dijkstra. ◀

El siguiente teorema muestra que el algoritmo de Dijkstra tiene un tiempo de $\Theta(n^2)$ en el peor caso.

Teorema 8.4.5

Para una entrada que consiste en una gráfica ponderada de n vértices, simple y conexa, el algoritmo de Dijkstra (algoritmo 8.4.1) tiene un tiempo de corrida en el peor caso de $\Theta(n^2)$.

Demostración Se considera el tiempo gastado en ciclos, que proporciona una cota superior en el tiempo total. La línea 4 se ejecuta $O(n)$ veces. Dentro del ciclo "while", la línea 9 toma un tiempo $O(n)$ [se podría encontrar $L(v)$ mínimo examinando todos los vértices en T]. El cuerpo del ciclo "for" (línea 12) toma un tiempo $O(n)$. Como las líneas 9 y 12 están anidadas en el ciclo "while" que toma un tiempo $O(n)$, el tiempo total para las líneas 9 y 12 es $O(n^2)$. Entonces el algoritmo de Dijkstra corre en un tiempo $O(n^2)$.

De hecho, para una selección adecuada de z, el tiempo es $\Omega(n^2)$ para K_n, la gráfica completa sobre n vértices, porque cada vértice es adyacente a los otros. Entonces el tiempo de corrida en el peor caso es $\Theta(n^2)$.

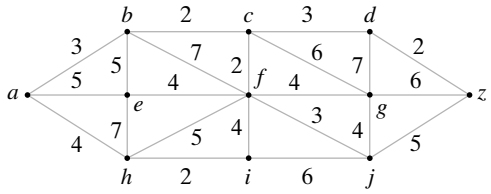
Cualquier algoritmo de la ruta más corta que recibe como entrada K_n, la gráfica completa sobre n vértices, debe examinar todas las aristas de K_n al menos una vez. Como K_n tiene $n(n-1)/2$ aristas (vea el ejercicio 15, sección 8.1), su tiempo de corrida en el peor caso debe ser al menos $n(n-1)/2 = \Omega(n^2)$. Se concluye a partir del Teorema 8.4.5 que el algoritmo 8.4.1 es óptimo.

Sección de ejercicios de repaso

1. Describa el algoritmo de una ruta más corta de Dijkstra.
2. Dé un ejemplo para mostrar la manera en que el algoritmo de Dijkstra encuentra una ruta más corta.
3. Pruebe que el algoritmo de Dijkstra encuentra correctamente una ruta más corta.

Ejercicios

En los ejercicios 1 al 5, encuentre la longitud de una ruta más corta entre cada par de vértices en la gráfica ponderada.



1. *a, f*
2. a, g
3. a, *z*
4. *b, j*
5. *h, d*
6. Escriba un algoritmo que encuentre la longitud de una ruta más corta entre dos vértices dados en una gráfica conexa ponderada y también encuentre una ruta más corta.
7. Escriba un algoritmo que encuentre las longitudes de las rutas más cortas de un vértice dado a todos los demás vértices en una gráfica G conexa ponderada.
★ 8. Escriba un algoritmo que encuentre las longitudes de las rutas más cortas entre todos los pares de vértices en una gráfica conexa ponderada simple que tiene n vértices con tiempo $O(n^3)$.
9. Modifique el algoritmo 8.4.1 para que acepte una gráfica ponderada que no necesariamente sea conexa. Al terminar, ¿qué es $L(z)$ si no hay trayectoria de a a z?
10. ¿Falso o verdadero? Cuando una gráfica conexa ponderada y los vértices a y z son la entrada al siguiente algoritmo, regresa la longitud de una ruta más corta de a a z. Si el algoritmo es correcto, pruébelo; de otra manera, dé un ejemplo de una gráfica conexa ponderada y vértices a y z para los que falla.

Algoritmo 8.4.6

```
algor(w, a, z) {
    longitud = 0
    v = a
    T = conjunto de todos los vértices
    while(v ¬ = z) {
        T = T − {v}
        seleccionar x ∈ T con w(v, x) mínimo
        longitud = longitud + w(v, x)
        v = x
    }
    return longitud
}
```

11. ¿Falso o verdadero? El algoritmo 8.4.1 encuentra la longitud de una ruta más corta en una gráfica conexa ponderada incluso si algunos pesos son negativos. Si es verdadero, pruébelo; de otra manera, proporcione un contraejemplo.

8.5 ➜ Representaciones de gráficas

En las secciones anteriores se representó una gráfica con un dibujo. En ocasiones, como por ejemplo al usar una computadora para analizar una gráfica, se necesita una representación más formal. El primer método de representación de una gráfica usa la **matriz de adyacencia**.

Ejemplo 8.5.1 ▶

Matriz de adyacencia

Considere la gráfica de la figura 8.5.1. Para obtener la matriz de adyacencia de esta gráfica, primero se selecciona un orden de los vértices, por ejemplo, *a, b, c, d, e*. Después se etiquetan los renglones y columnas de una matriz con los vértices ordenados. El elemento en esta matriz en el renglón i y la columna j, $i \neq j$, es el número de aristas incidentes en i y j. Si $i = j$, el elemento es dos veces el número de ciclos que inciden en i. La matriz de adyacencia para esta gráfica es

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 \\ 1 & 1 & 1 & 2 & 0 \end{pmatrix} \end{array}.$$

◀

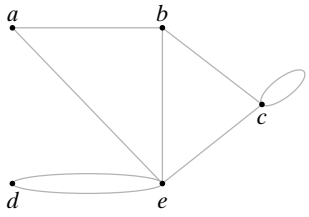


Figura 8.5.1 Gráfica para el ejemplo 8.5.1.

Observe que es posible obtener el grado de un vértice v en una gráfica G sumando el renglón v o la columna v en la matriz de adyacencia de G.

La matriz de adyacencia no es una manera muy eficiente para representar una gráfica. Como la matriz es simétrica respecto a la diagonal principal (los elementos en la línea de la esquina superior izquierda a la esquina inferior derecha), la información aparece dos veces, excepto la de la diagonal principal.

Ejemplo 8.5.2 ▶

La matriz de adyacencia de la gráfica simple de la figura 8.5.2 es

$$A = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} \end{array}.$$

◀

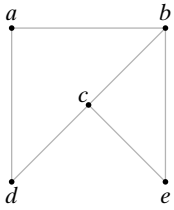


Figura 8.5.2 Gráfica para el ejemplo 8.5.2.

Se demostrará que si A es la matriz de adyacencia de una gráfica simple G, las potencias de A,

$$A, A^2, A^3, \ldots,$$

cuentan el número de diferentes longitudes. De manera más precisa, si los vértices de G se etiquetan 1, 2, . . . , el elemento ij de la matriz A^n es igual al número de trayectorias de i a j de longitud n. Por ejemplo, suponga que se eleva al cuadrado la matriz A del ejemplo 8.5.2

para obtener

$$A^2 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} = \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} \begin{matrix} \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{pmatrix} 2 & 0 & 2 & 0 & 1 \\ 0 & 3 & 1 & 2 & 1 \\ 2 & 1 & 3 & 0 & 1 \\ 0 & 2 & 0 & 2 & 1 \\ 1 & 1 & 1 & 1 & 2 \end{pmatrix} \end{matrix}.$$

Considere el elemento del renglón a y la columna c en A^2, obtenido al multiplicar los elementos del renglón a por los de la columna c de la matriz A y sumarlos

$$\begin{matrix} & & & b & & d & & \\ a & (0 & 1 & 0 & 1 & 0) & \begin{matrix} c \\ \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} \end{matrix} & \begin{matrix} \\ b \\ \\ d \\ \end{matrix} \end{matrix} = 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 1 = 2.$$

La única manera de que aparezca un producto diferente de cero en esta suma es que los dos elementos que se multiplican sean 1. Esto ocurre si hay un vértice v cuyo elemento en el renglón a es 1 y cuyo elemento en la columna c es 1. En otras palabras, debe haber aristas de la forma (a, v) y (v, c). Estas aristas forman la trayectoria (a, v, c) de longitud 2 de a a c y cada trayectoria aumenta la suma en 1. En este ejemplo, la suma es 2 porque hay dos trayectorias

$$(a, b, c), \quad (a, d, c)$$

de longitud 2 de a a c. En general, el elemento en el renglón x y la columna y de la matriz A^2 es el número de trayectorias de longitud 2 del vértice x al vértice y.

Los elementos en la diagonal principal de A^2 dan los grados de los vértices (cuando se trata de una gráfica simple). Considere, por ejemplo, el vértice c. El grado de c es 3 ya que c es incidente en las tres aristas (c, b), (c, b) y (c, e). Pero cada una de estas aristas se puede convertir en una trayectoria de longitud 2 de c a c:

$$(c, b, c), \quad (c, d, c), \quad (c, e, c).$$

De manera similar, una trayectoria de longitud 2 de c a c define una arista incidente en c. Entonces, el número de trayectorias de longitud 2 de c a c es 3, el grado de c.

Ahora se usará inducción para demostrar que los elementos de la n-ésima potencia de una matriz de adyacencia dan el número de trayectorias de longitud n.

Teorema 8.5.3

Si A es la matriz de adyacencia de una gráfica simple, el elemento ij de A^n es igual al número de trayectorias de longitud n del vértice i al vértice j, $n = 1, 2, \ldots$.

Demostración Se usará inducción sobre n.

Si $n = 1$, A^1 es simplemente A. El elemento ij es 1 si hay una arista de i a j, que es una trayectoria de longitud 1, y 0 de otra manera. Entonces, el teorema se cumple si $n = 1$. Esto verifica el paso base.

Suponga que el teorema es cierto para n. Ahora

$$A^{n+1} = A^n A$$

de manera que el elemento i en A^{n+1} se obtiene al multiplicar por pares los elementos en

el renglón i de A^n por los elementos en la columna k de A y sumarlos.

$$\text{renglón } i \text{ de } A^n \quad (s_1,\ s_2,\ \ldots,\ s_j,\ \ldots,\ s_m) \quad \begin{matrix}\\ \\ \\ \\ \\ \text{columna } k \text{ de } A\end{matrix} \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_j \\ \vdots \\ t_m \end{pmatrix}$$

$$= s_1t_1 + s_2t_2 + \cdots + s_jt_j + \cdots + s_mt_m$$

$$= \text{elemento } ik \text{ de } A^{n+1}.$$

Por inducción, s_j da el número de trayectorias de longitud n de i a j en la gráfica G. Ahora t_j es 0 o 1. Si t_j es 0, no hay arista de j a k, por lo que hay $s_jt_j = 0$ trayectorias de longitud $n + 1$ de i a k, donde la última arista es (j, k). Si t_j es 1, hay una arista del vértice j al vértice k (figura 5.8.3). Como hay s_j trayectorias de longitud n del vértice i al vértice j, hay $s_jt_j = s_j$ trayectorias de longitud $n + 1$ de i a k, donde la última arista es (j, k) (figura 8.5.3). Al sumar sobre j se cuentan todas las trayectorias de longitud $n + 1$ de i a k. Entonces, el elemento ik en A^{n+1} da el número de trayectorias de longitud $n + 1$ de i a k, y esto verifica el paso inductivo.

Por el principio de inducción matemática, el teorema queda establecido.

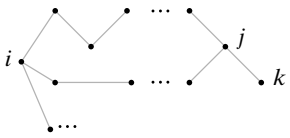


Figura 8.5.3 Prueba del Teorema 8.5.3. Una trayectoria de i a k de longitud $n + 1$ cuyo penúltimo vértice es j consiste en una trayectoria de longitud n de i a j seguida de la arista (j, k). Si hay s_j trayectorias de longitud n de i a j y t_j es 1 si la arista (i, k) existe y 0 de otra manera, la suma de s_jt_j sobre toda j da el número de trayectorias de longitud $n + 1$ de i a k.

Ejemplo 8.5.4 ▶

Después del ejemplo 8.5.2, se demostró que si A es la matriz de la gráfica de la figura 8.5.2, entonces

$$A^2 = \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} \begin{matrix} a & b & c & d & e \end{matrix} \begin{pmatrix} 2 & 0 & 2 & 0 & 1 \\ 0 & 3 & 1 & 2 & 1 \\ 2 & 1 & 3 & 0 & 1 \\ 0 & 2 & 0 & 2 & 1 \\ 1 & 1 & 1 & 1 & 2 \end{pmatrix}.$$

Al multiplicar,

$$A^4 = A^2A^2 = \begin{pmatrix} 2 & 0 & 2 & 0 & 1 \\ 0 & 3 & 1 & 2 & 1 \\ 2 & 1 & 3 & 0 & 1 \\ 0 & 2 & 0 & 2 & 1 \\ 1 & 1 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} 2 & 0 & 2 & 0 & 1 \\ 0 & 3 & 1 & 2 & 1 \\ 2 & 1 & 3 & 0 & 1 \\ 0 & 2 & 0 & 2 & 1 \\ 1 & 1 & 1 & 1 & 2 \end{pmatrix},$$

se encuentra que

$$A^4 = \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} \begin{matrix} a & b & c & d & e \end{matrix} \begin{pmatrix} 9 & 3 & 11 & 1 & 6 \\ 3 & 15 & 7 & 11 & 8 \\ 11 & 7 & 15 & 3 & 8 \\ 1 & 11 & 3 & 9 & 6 \\ 6 & 8 & 8 & 6 & 8 \end{pmatrix}.$$

El elemento en el renglón d y la columna e es 6, lo que significa que hay seis trayectorias de longitud 4 de d a e. Por inspección, se encuentra que son

$$(d, a, d, c, e), \quad (d, c, d, c, e), \quad (d, a, b, c, e),$$
$$(d, c, e, c, e), \quad (d, c, e, b, e), \quad (d, c, b, c, e).$$

◀

Otra representación matricial útil de una gráfica se conoce como **matriz de incidencia**.

Ejemplo 8.5.5 ▶

WWW

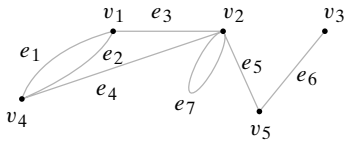


Figura 8.5.4 Gráfica para el ejemplo 8.5.5.

Matriz de incidencia

Para obtener la matriz de incidencia de la gráfica en la figura 8.5.4, se etiquetan los renglones con los vértices y las columnas con las aristas (en algún orden arbitrario). El elemento en el renglón v y la columna e es 1 si e es incidente en v, y 0 de otra manera. Entonces, la matriz de incidencia para la gráfica de la figura 8.5.4 es

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}.$$

Se entiende que una columna como e_7 representa un lazo. ◀

Observe que en una gráfica sin lazos cada columna tiene dos números 1 y que la suma de un renglón da el grado del vértice identificado con ese renglón.

Sección de ejercicios de repaso

1. ¿Qué es una matriz de adyacencia?

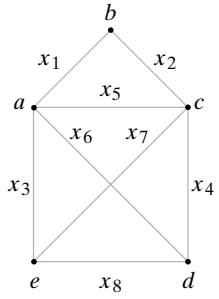
2. Si A es la matriz de adyacencia de una gráfica simple, ¿Cuáles son los valores de los elementos de A^n?

3. ¿Qué es una matriz de incidencia?

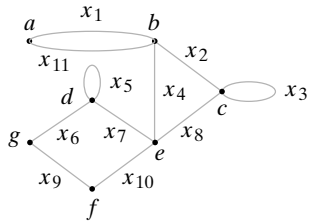
Ejercicios

En los ejercicios 1 al 6, escriba la matriz de adyacencia de cada gráfica.

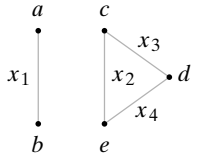
1.



2.



3.



4. La gráfica de la figura 8.2.2

5. La gráfica completa sobre cinco vértices K_5

6. La gráfica completa bipartita $K_{2,3}$

En los ejercicios 7 al 12, escriba la matriz de incidencia de cada gráfica.

7. La gráfica del ejercicio 1 ***8.*** La gráfica del ejercicio 2

9. La gráfica del ejercicio 3 ***10.*** La gráfica de la figura 8.2.1

11. La gráfica completa sobre cinco vértices K_5

12. La gráfica completa bipartita $K_{2,3}$

En los ejercicios 13 al 17, dibuje la gráfica representada por cada matriz de adyacencia.

13.
$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 2 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

14.
$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 2 \end{pmatrix} \end{array}$$

15.
$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 2 & 0 & 1 & 2 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

16.
$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 4 & 1 & 1 & 1 & 0 & 2 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 3 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 2 & 0 & 3 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

17. La matriz de 7×7 cuyo elemento ij es 1 si $i + 1$ divide a $j + 1$ o $j + 1$ divide a $i + 1$, $i \neq j$; cuyo elemento ij es 2 si $i = j$, y cuyo elemento ij es 0 en otros casos.

18. Escriba la matriz de adyacencia de las componentes de las gráficas dadas por las matrices de adyacencia de los ejercicios 13 al 17.

19. Calcule los cuadrados de las matrices de adyacencia de K_5 y las gráficas de los ejercicios 1 y 3.

20. Sea A la matriz de adyacencia para la gráfica del ejercicio 1. ¿Cuál es el elemento en el renglón a y la columna d de A^5?

21. Suponga que una gráfica tiene una matriz de adyacencia de la forma

$$A = \left(\begin{array}{c|c} & A' \\ \hline A'' & \end{array}\right),$$

donde todos los elementos de las submatrices A' y A'' son 0. ¿Cómo se ve la gráfica?

22. Repita el ejercicio 21 con "incidencia" en lugar de "adyacencia".

23. Sea A una matriz de adyacencia de una gráfica. ¿Por qué A^n es simétrica respecto a la diagonal principal para todo entero positivo n?

En los ejercicios 24 y 25, dibuje las gráficas representadas por las matrices de incidencia.

24.

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array}\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

25.

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array}\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

26. ¿Cómo debe verse una gráfica si algún renglón de su matriz de incidencia tiene sólo ceros?

27. Sea A la matriz de adyacencia de una gráfica G con n vértices. Sea

$$Y = A + A^2 + \cdots + A^{n-1}.$$

Si algún elemento fuera de la diagonal en la matriz Y es cero, ¿qué se puede decir de la gráfica G?

Los ejercicios 28 al 31 se refieren a la matriz de adyacencia A de K_5.

28. Sea n un entero positivo. Explique por qué todos los elementos de la diagonal de A^n son iguales y todos los elementos fuera de la diagonal de A^n son iguales.

Sea d_n el valor común de los elementos de la diagonal de A^n y sea a_n el valor común de los elementos fuera de la diagonal de A^n.

★**29.** Demuestre que

$$d_{n+1} = 4a_n; \quad a_{n+1} = d_n + 3a_n; \quad a_{n+1} = 3a_n + 4a_{n-1}.$$

★**30.** Demuestre que

$$a_n = \frac{1}{5}[4^n + (-1)^{n+1}].$$

31. Demuestre que

$$d_n = \frac{4}{5}[4^{n-1} + (-1)^n].$$

★**32.** Derive resultados similares a los de los ejercicios 29 al 31 para la matriz de adyacencia A de la gráfica K_m.

★**33.** Sea A la matriz de adyacencia de la gráfica $K_{m,n}$. Encuentre una fórmula para los elementos de A^j.

8.6 ➜ Isomorfismos de gráficas

Las siguientes instrucciones se dan a dos personas que no pueden ver el papel de la otra: "Dibuje y etiquete cinco vértices *a, b, c, d* y *e*. Conecte *a* con *b*, *b* con *c*, *c* con *d*, *d* con *e*, y *a* con *e*". Las gráficas producidas se aprecian en la figura 8.6.1. Sin duda estas figuras definen la misma gráfica aun cuando parezcan diferentes. Se dice que estas gráficas son **isomorfas**.

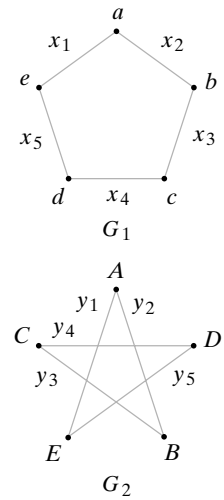


Figura 8.6.1 Gráficas isomorfas.

Definición 8.6.1 ▶

Las gráficas G_1 y G_2 son *isomorfas* si existe una función f uno a uno y sobre de los vértices de G_1 a los vértices de G_2 y una función g uno a uno y sobre de las aristas de G_1 a las aristas de G_2, de manera que una arista e es incidente en v y w en G_1 si y sólo si la arista

$g(e)$ es incidente en $f(v)$ y $f(w)$ en G_2. El par de funciones f y g reciben el nombre de *isomorfismo* de G_1 en G_2. ◀

Ejemplo 8.6.2 ▶

Un isomorfismo para las gráficas G_1 y G_2 de la figura 8.6.1 se define por

$$f(a) = A, \quad f(b) = B, \quad f(c) = C, \quad f(d) = D, \quad f(e) = E,$$
$$g(x_i) = y_i, \quad i = 1, \ldots, 5.$$

◀

Si se define una relación R en un conjunto de gráficas mediante la regla $G_1 R G_2$ si G_1 y G_2 son isomorfas, R es una relación de equivalencia. Cada clase de equivalencia consiste en un conjunto de gráficas isomorfas mutuamente excluyentes.

Ejemplo 8.6.3 ▶

Modelo de rejilla para computación en paralelo

Antes se estudió el problema de cuándo el cubo-n podía simular un modelo de anillo para computación en paralelo (vea el ejercicio 8.3.5). Ahora consideramos cuándo el cubo-n puede simular el **modelo de rejilla para computación en paralelo**.

El modelo de rejilla de dos dimensiones para computación en paralelo, cuando se describe como una gráfica, consiste en un arreglo rectangular de vértices conectados como se indica (figura 8.6.2). El problema "¿Cuándo puede un cubo-n simular una rejilla de dos dimensiones?" se enuncia de otra manera en la terminología de gráficas como "¿Cuándo un cubo-n contiene una subgráfica isomorfa a una rejilla de dos dimensiones?" Se mostrará que si M es una rejilla de p vértices por q vértices, donde $p \leq 2^i$ y $q \leq 2^j$, entonces el cubo-$(i + j)$ contiene una subgráfica isomorfa a M. (En la figura 8.6.2, se puede tomar $p = 6$, $q = 4$, $i = 3$ y $j = 2$. Entonces, el resultado indica que el cubo-5 contiene una subgráfica isomorfa a la gráfica de la figura 8.6.2).

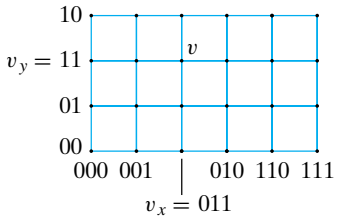


Figura 8.6.2 Modelo de rejilla para computación en paralelo.

Sea M una rejilla de p por q vértices, donde $p \leq 2^i$ y $q \leq 2^j$. Se considera que M es un arreglo rectangular en el espacio de 2 dimensiones ordinario con p vértices en dirección horizontal y q vértices en dirección vertical (figura 8.6.2). Como coordenadas para los vértices se usan elementos de los códigos Gray. (Los códigos Gray se explican en el ejemplo 8.3.5). Las coordenadas en dirección horizontal son los primeros p miembros de un código Gray de i bits y las coordenadas en dirección vertical son los primeros q miembros de un código Gray de j bits (vea la figura 8.6.2). Si un vértice v está en la rejilla, sean v_x la coordenada horizontal de v y v_y la coordenada vertical de v. Entonces se define una función f sobre los vértices de M por

$$f(v) = v_x v_y.$$

(La cadena $v_x v_y$ es la cadena v_x seguida de la cadena v_y). Observe que f es uno a uno.

Si (v, w) es una arista en M, las cadenas de bits $v_x v_y$ y $w_x w_y$ difieren exactamente en un bit. Entonces $(v_x v_y, w_x w_y)$ es una arista en el cubo-$(i + j)$. Se define la función g en las aristas de M por

$$g((v, w)) = (v_x v_y, w_x w_y).$$

Note que g es uno a uno. El par de funciones f, g es un isomorfismo de M sobre la subgráfica (V, E) del cubo-$(i + j)$, donde

$$V = \{f(v) \mid v \text{ es un vértice en } M\}, \qquad E = \{g(e) \mid e \text{ es una arista en } M\}.$$

Por lo tanto, si M es una rejilla de p por q vértices, donde $p \leq 2^i$ y $q \leq 2^j$, el cubo-$(i + j)$ contiene una subgráfica isomorfa a M.

El argumento dado se extiende a un número arbitrario de dimensiones (vea el ejercicio 11); es decir, si M es una rejilla de $p_1 \times p_2 \times \cdots \times p_k$, donde $p_i \leq 2^{n_i}$ para $i = 1, \ldots, k$, entonces el cubo-$(t_1 + t_2 + \cdots + t_k)$ contiene una subgráfica isomorfa a M. ◀

En general, la matriz de adyacencia de una gráfica cambia cuando se modifica el orden de sus vértices. También es posible demostrar que las gráficas G_1 y G_2 son isomórficas si y sólo si *para algún* orden de los vértices, sus matrices de adyacencia son iguales.

Teorema 8.6.4

Las gráficas G_1 y G_2 son isomorfas si y sólo si, para algún orden de sus vértices, sus matrices de adyacencia son iguales.

Demostración Suponga que G_1 y G_2 son isomorfas. Entonces existe una función f uno a uno y sobre, de los vértices de G_1 a los vértices de G_2, y una función g uno a uno y sobre de las aristas de G_1 a las aristas de G_2, de manera que una arista e es incidente en v y w si y sólo si la arista $g(e)$ incide en $f(v)$ y $f(w)$ en G_2.

Sea $v_1, \ldots, v_n$ un orden de los vértices de G_1. Sea A_1 la matriz de adyacencia de G_1 relativa al orden $v_1, \ldots, v_n$, y sea A_2 la matriz de adyacencia de G_2 relativa al orden $f(v_1), \ldots, f(v_n)$. Suponga que el elemento en el renglón i y la columna j, $i \neq j$, de A_1 es igual a k. Entonces existen k aristas, digamos $e_1, \ldots, e_k$, incidentes en v_i y v_j. Por lo tanto, hay exactamente k aristas $g(e_1), \ldots, g(e_k)$ incidentes en $f(v_i)$ y $f(v_j)$ en G_2. Entonces el elemento en el renglón i, columna j en A_2, que cuenta el número de aristas que inciden en $f(v_i)$ y $f(v_j)$ también es igual a k. Un argumento similar señala que los elementos en las diagonales de A_1 y A_2 son iguales. Por lo tanto, $A_1 = A_2$.

El inverso es similar y se deja como ejercicio (vea el ejercicio 25).

Corolario 8.6.5

Sean G_1 y G_2 gráficas simples. Las siguientes son equivalentes:

a) G_1 y G_2 son isomorfas.

b) Existe una función f uno a uno y sobre del conjunto de vértices de G_1 al conjunto de vértices de G_2 que satisface lo siguiente: los vértices v y w son adyacentes en G_1 si y sólo si los vértices $f(v)$ y $f(w)$ son adyacentes en G_2.

Demostración A partir de la definición 8.6.1 se concluye de inmediato que *a)* implica *b)*.

Se probará que *b)* implica *a)*. Suponga que existe una función f uno a uno y sobre del conjunto de vértices en G_1 al conjunto de vértices en G_2 que satisface lo siguiente: los vértices v y w son adyacentes en G_1 si y sólo si los vértices $f(v)$ y $f(w)$ son adyacentes en G_2.

Sea $v_1, \ldots, v_n$ un orden de los vértices de G_1. Sea A_1 la matriz de adyacencia de G_1 relativa al orden $v_1, \ldots, v_n$ y sea A_2 la matriz de adyacencia de G_2 relativa al orden $f(v_1), \ldots, f(v_n)$. Como G_1 y G_2 son gráficas simples, los elementos de las matrices de adyacencia son 1 (para indicar que los vértices son adyacentes) o 0 (para indicar que los vértices no son adyacentes). Como los vértices v y w son adyacentes en G_1 si y sólo si lo vértices $f(v)$ y $f(w)$ son adyacentes en G_2, se concluye que $A_1 = A_2$. Por el Teorema 8.6.4, G_1 y G_2 son isomorfas.

Ejemplo 8.6.6 ▶

La matriz de adyacencia de la gráfica G_1 en la figura 8.6.1 relativa al orden de los vértices *a, b, c, d, e,*

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array},$$

es igual a la matriz de adyacencia de la gráfica G_2 en la figura 8.6.1 relativa al orden de los vértices *A, B, C, D, E,*

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}.$$

De nuevo, se ve que G_1 y G_2 son isomorfas. ◀

Un problema interesante es determinar si dos gráficas son isomorfas. Aunque todos los algoritmos conocidos para probar un isomorfismo entre dos gráficas requieren tiempo exponencial o factorial en el peor caso, existen algoritmos que pueden determinar si un par de gráficas son isomorfas en tiempo lineal en el caso promedio (vea [Read] y [Babai]).

La siguiente es una forma de demostrar que dos gráficas simples G_1 y G_2 *no* son isomorfas. Encuentre una propiedad de G_1 que G_2 *no* tenga, pero que G_2 *tendría* si G_1 y G_2 fueran isomorfas. Esta propiedad se llama **invariante**. De forma más precisa, una propiedad P es una invariante si siempre que G_1 y G_2 sean gráficas isomorfas:

Si G_1 tiene la propiedad P, G_2 también tiene la propiedad P.

Por la definición 8.6.1, si las gráficas G_1 y G_2 son isomorfas, existen funciones uno a uno y sobre de las aristas (y vértices, respectivamente) de G_1 a las aristas (y vértices respectivamente) de G_2. Así, si G_1 y G_2 son isomorfas, G_1 y G_2 tienen el mismo número de aristas y el mismo número de vértices. Por lo tanto, si e y n son enteros no negativos, las propiedades "tiene e aristas" y "tiene n vértices" son invariantes.

Ejemplo 8.6.7 ▶

Las gráficas G_1 y G_2 en la figura 8.6.3 no son isomorfas, ya que G_1 tiene siete aristas y G_2 tiene seis, y "tiene siete aristas" es una invariante.

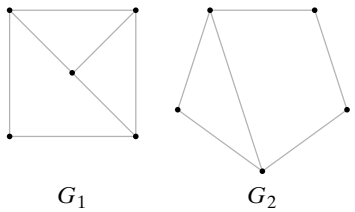


Figura 8.6.3 Gráficas no isomorfas. G_1 tiene siete aristas, G_2 tiene seis aristas.

◀

Ejemplo 8.6.8 ▶

Demuestre que si k es un entero positivo, "tiene un vértice de grado k" es una invariante.

Suponga que G_1 y G_2 son gráficas isomorfas y f (o g) es una función uno a uno y sobre de los vértices (o aristas) de G_1 sobre los vértices (o aristas) de G_2. Suponga que G_1 tiene un vértice v de grado k. Entonces hay k aristas $e_1, \ldots, e_k$ que inciden en v. Por la definición 8.6.1, $g(e_1), \ldots, g(e_k)$ son incidentes en $f(v)$. Como g es uno a uno, $\delta(f(v)) \geq k$.

Sea E una arista que incide en $f(v)$ en G_2. Como g es sobre, existe una arista e en G_1 con $g(e) = E$. Puesto que $g(e)$ incide en $f(v)$ en G_2, por la definición 8.6.1, e incide sobre v en G_1. Como $e_1, \ldots, e_k$ son las únicas aristas en G_1 que inciden en v, $e = e_i$ para alguna $i \in \{1, \ldots, k\}$. Ahora bien, $g(e_i) = g(e) = E$. Entonces $\delta(f(v)) \geq k$, de manera que G_2 tiene un vértice, a saber $f(v)$, de grado k. ◀

Ejemplo 8.6.9 ▶

Puesto que "tiene un vértice de grado 3" es una invariante, las gráficas G_1 y G_2 de la figura 8.6.4 no son isomorfas; G_1 tiene vértices (a y f) de grado 3, pero G_2 no tiene vértices de grado 3. Observe que G_1 y G_2 tiene el mismo número de vértices y aristas.

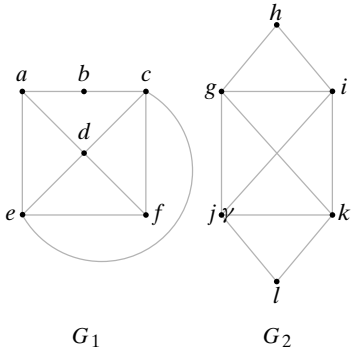


Figura 8.6.4 Gráficas no isomorfas. G_1 tiene vértices de grado 3, pero G_2 no tiene vértices de grado 3. ◀

Otra invariante frecuentemente útil es "tiene un ciclo simple de longitud k". Se deja la prueba de esta propiedad como una invariante para los ejercicios (ejercicio 12).

Ejemplo 8.6.10 ▶

Dado que "tienen un ciclo simple de longitud 3" es una invariante, las gráficas G_1 y G_2 de la figura 8.6.5 no son isomorfas; la gráfica G_2 tiene un ciclo simple de longitud 3, pero todos los ciclos simples en G_1 tienen longitud de al menos 4. Observe que G_1 y G_2 tienen el mismo número de aristas y vértices y que cada vértice en G_1 o G_2 tiene grado 4.

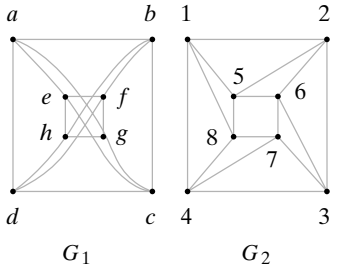


Figura 8.6.5 Gráficas no isomorfas. G_2 tiene un ciclo simple de longitud 3, pero G_1 no tiene ciclos simples de longitud 3. ◀

Sería sencillo probar si un par de gráficas son isomorfas si se pudiera encontrar un pequeño número de invariantes que fuera fácil verificar y que *sólo* las gráficas isomorfas compartieran. Desafortunadamente, nadie ha tenido éxito en encontrar tal conjunto de invariantes.

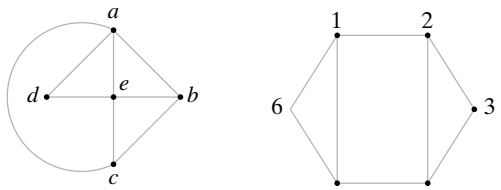
Sección de ejercicios de repaso

1. Establezca qué significa que dos gráficas sean isomorfas.
2. Dé un ejemplo de gráficas no idénticas isomorfas. Explique por qué son isomorfas.
3. Dé un ejemplo de dos gráficas que *no* sean isomorfas. Explique por qué no lo son.
4. ¿Qué es una invariante en una gráfica?
5. ¿Cuál es la relación de una "invariante" con un isomorfismo?
6. ¿Cómo se puede determinar si las gráficas son isomorfas a partir de sus matrices de adyacencia?
7. ¿Cuál es el modelo de rejilla para computación en paralelo?

Ejercicios

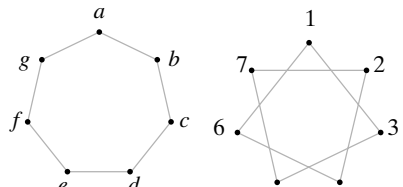
En los ejercicios 1 al 10, determine si las gráficas G_1 y G_2 son isomorfas. Si las gráficas son isomorfas, encuentre funciones f y g para la definición 8.6.1; de otra manera, dé una invariante que las gráficas no compartan.

1.



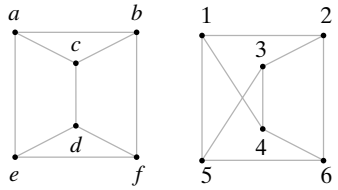
G_1 G_2

2.



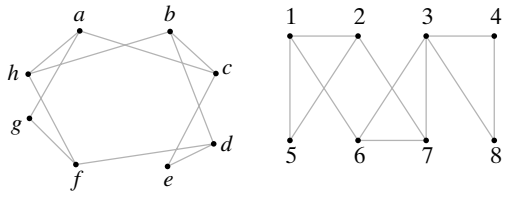
G_1 G_2

3.



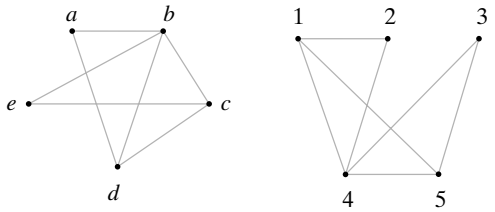
G_1 G_2

4.



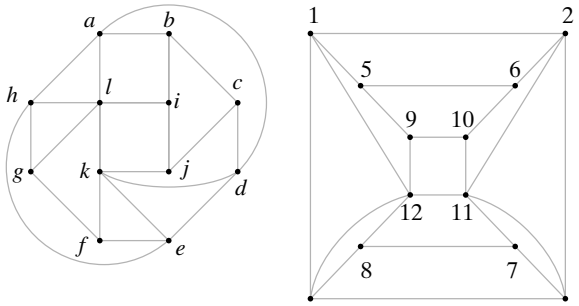
G_1 G_2

5.



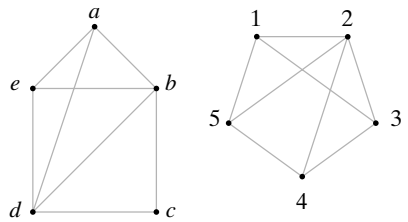
G_1 G_2

6.



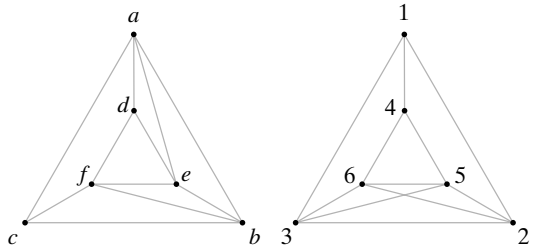
G_1 G_2

7.



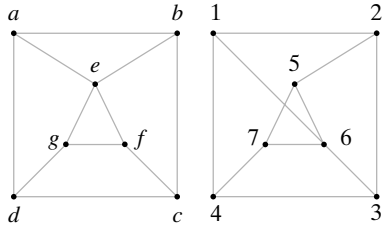
G_1 G_2

★ **8.**



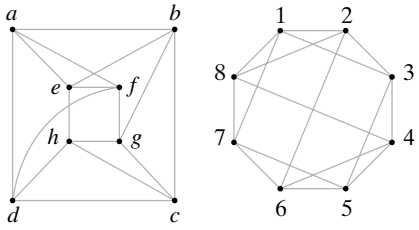
G_1 G_2

★ **9.**



G_1 G_2

★***10.***



G_1 G_2

11. Demuestre que si M es una rejilla de $p_1 \times p_2 \times \cdots \times p_k$, donde $p_i \leq 2^{t_i}$, para $i = 1, \ldots, k$, entonces el cubo-$(t_1 + t_2 + \cdots + t_k)$ contiene una subgráfica isomorfa a M.

En los ejercicios 12 al 16, muestre que la propiedad indicada es una invariante.

12. Tiene un ciclo simple de longitud k

13. Tiene n vértices de grado k

14. Es conexa

15. Tiene n ciclos simples de longitud k

16. Tiene una arista (v, w), donde $\delta(v) = i$ y $\delta(w) = j$

17. Encuentre una invariante que no esté dada en esta sección o en los ejercicios 12 al 16. Pruebe que su propiedad es invariante.

En los ejercicios 18 al 20, diga si cada propiedad es una invariante. Si es una invariante, pruebe que lo es; de otra manera, dé un contraejemplo.

18. Tiene un ciclo de Euler

19. Tiene un vértice dentro de algún ciclo simple

20. Es bipartita

21. Dibuje todas las gráficas simples no isomorfas de tres vértices.

22. Dibuje todas las gráficas simples no isomorfas de cuatro vértices.

23. Dibuje todas las gráficas no isomorfas, sin ciclos y conexas que tienen cinco vértices.

24. Dibuje todas las gráficas no isomorfas, sin ciclos y conexas que tienen seis vértices.

25. Demuestre que las gráficas G_1 y G_2 son isomorfas si sus vértices se puede ordenar de manera que sus matrices de adyacencia sean iguales.

El complemento de una gráfica simple G es la gráfica simple $\overline{G}$ con los mismos vértices que G. Una arista existe en $\overline{G}$ si y sólo si no existe en G.

26. Dibuje el complemento de la gráfica G_1 del ejercicio 1.

27. Dibuje el complemento de la gráfica G_2 del ejercicio 1.

★**28.** Demuestre que si G es una gráfica simple, G_1 o bien $\overline{G}$ es conexa.

29. Una gráfica simple es **autocomplementaria** si G y $\overline{G}$ son isomorfas.

a) Encuentre una gráfica autocomplementaria que tenga cinco vértices.

b) Encuentre otra gráfica autocomplementaria.

30. Sean G_1 y G_2 gráficas simples. Muestre que G_1 y G_2 son isomorfas si y sólo si $\overline{G}_1$ y $\overline{G}_2$ son isomorfas.

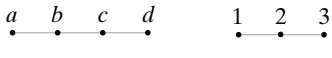
31. Dadas dos gráficas G_1 y G_2, suponga que existe una función f uno a uno y sobre de los vértices de G_1 a los vértices de G_2, y una función g uno a uno y sobre de las aristas de G_1 a las aristas de G_2, de manera que si una arista e incide en v y w en G_1, la arista $g(e)$ incide en $f(v)$ y $f(w)$ en G_2. ¿Son isomorfas G_1 y G_2?

Un homomorfismo de una gráfica G_1 a una gráfica G_2 es una función f del conjunto de vértices de G_1 al conjunto de vértices de G_2 con la propiedad de que si v y w son adyacentes en G_1, entonces $f(v)$ y $f(w)$ son adyacentes en G_2.

32. Suponga que G_1 y G_2 son gráficas simples. Demuestre que si f es un homomorfismo de G_1 a G_2 y f es uno a uno y sobre, G_1 y G_2 son isomorfas.

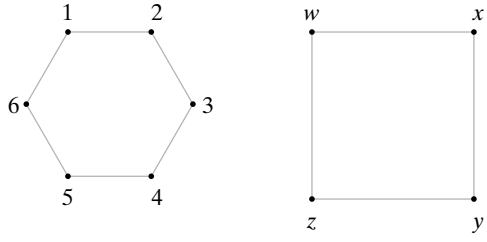
En los ejercicios 33 al 37, para cada par de gráficas, dé un ejemplo de un homomorfismo de G_1 a G_2.

33.



G_1 G_2

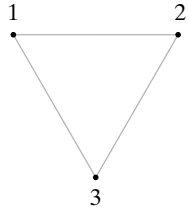
34.



G_1 G_2

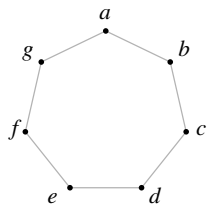
35. $G_1 = G_1$ del ejercicio 34; $G_2 = G_1$ del ejercicio 33

36. $G_1 = G_1$ del ejercicio 33

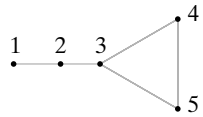


G_2

37.

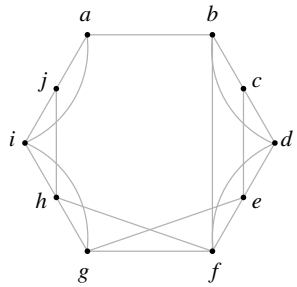


G_1



G_2

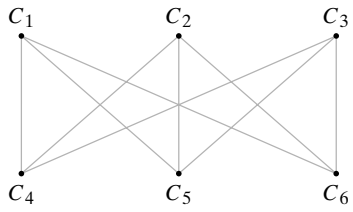
★ **38.** [Hell] Demuestre que el único homomorfismo de una gráfica a sí misma es la función identidad.



8.7 ➜ Gráficas planas

Tres ciudades, C_1, C_2 y C_3, deberán conectarse en forma directa mediante autopistas con cada una de otras tres ciudades, C_4, C_5 y C_6. ¿Puede diseñarse este sistema de carreteras de manera que las autopistas no se crucen? La figura 8.7.1 ilustra una sistema en el que las autopistas se cruzan. Si usted intenta dibujar un sistema en el que las carreteras no se crucen, pronto se convencerá de que no puede hacerlo. Más adelante en esta sección se explicará con cuidado por qué no se puede hacer.

WWW



$K_{3,3}$

Figura 8.7.1 Ciudades conectadas por autopistas.

Definición 8.7.1 ▶ Una gráfica es *plana* si se puede dibujar en el plano sin que sus aristas se crucen. ◀

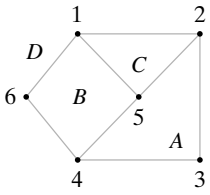


Figura 8.7.2 Gráfica plana conexa con $f = 4$ caras (A, B, C, D), $e = 8$ aristas y $v = 6$ vértices; $f = e - v + 2$.

Al diseñar circuitos impresos es deseable tener el menor número de cruces posible; así, el diseñador de circuitos impresos se enfrenta con el problema de gráficas planas.

Si una gráfica plana conexa se dibuja en el plano, éste se divide en regiones contiguas llamadas **caras**. Una cara se caracteriza por el ciclo que forma su frontera. Por ejemplo, en la gráfica de la figura 8.7.2, la cara A tiene como límite el ciclo (5, 2, 3, 4, 5) y el límite de la cara C es el ciclo (1, 2, 5, 1). La cara exterior D se considera limitada por el ciclo (1, 2, 3, 4, 6, 1). La gráfica de la figura 8.7.2 tiene $f = 4$ caras, $e = 8$ aristas y $v = 6$ vértices. Observe que f, e y v satisfacen la ecuación

$$f = e - v + 2. \tag{8.7.1}$$

En 1752, Euler probó que la ecuación (8.7.1) se cumple para cualquier gráfica conexa plana. Al final de esta sección se verá cómo probar (8.7.1), pero por ahora se mostrará cómo es que (8.7.1) se puede utilizar para demostrar que ciertas gráficas no son planas.

Ejemplo 8.7.2 ▶ Demuestre que la gráfica $K_{3,3}$ de la figura 8.7.1 no es plana.

Suponga que $K_{3,3}$ es plana. Como cada ciclo tiene al menos cuatro aristas, cada cara está limitada por al menos cuatro aristas. Entonces, el número de aristas que acotan las ca-

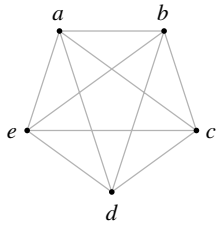


Figura 8.7.3 La gráfica no plana K_5.

ras es al menos $4f$. En una gráfica plana, cada arista pertenece al menos a dos ciclos frontera. Por lo tanto,

$$2e \geq 4f.$$

Usando (8.7.1), se encuentra que

$$2e \geq 4(e - v + 2). \tag{8.7.2}$$

Para la gráfica de la figura 8.7.1, $e = 9$ y $v = 6$, de manera que (8.7.2.) se convierte en

$$18 = 2 \cdot 9 \geq 4(9 - 6 + 2) = 20,$$

que es una contradicción. Por lo tanto, $K_{3,3}$ no es plana. ◀

Con un argumento similar (vea el ejercicio 15), es posible demostrar que la gráfica K_5 de la figura 8.7.3 no es plana.

Es obvio que si una gráfica contiene a $K_{3,3}$ o K_5 como subgráfica, no puede ser plana. El inverso es casi verdadero. Para establecer la situación de manera más precisa, deben introducirse algunos términos nuevos.

Definición 8.7.3 ▶

Si una gráfica G tiene un vértice v de grado 2 y aristas (v, v_1) y (v, v_2) con $v_1 \neq v_2$, se dice que las aristas (v, v_1) y (v, v_2) están en *serie*. La *reducción de una serie* consiste en eliminar el vértice v de la gráfica G y sustituir las aristas (v, v_1) y (v, v_2) por la arista (v_1, v_2). Se dice que la gráfica G' que resulta *se obtiene de G al reducir una serie*. Por convención, se dice que G se puede obtener a partir de sí misma mediante una reducción de serie. ◀

Ejemplo 8.7.4 ▶

En la gráfica G de la figura 8.7.4, las aristas (v, v_1) y (v, v_2) están en serie. La gráfica G' de la figura 8.7.4 se obtiene de G al reducir la serie.

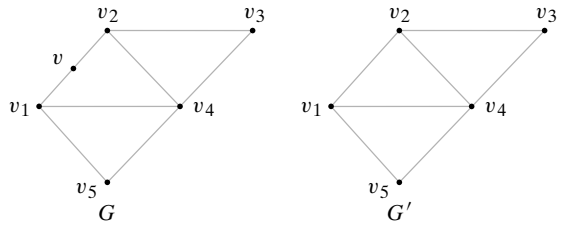


Figura 8.7.4 G' se obtiene de G al reducir una serie. ◀

Definición 8.7.5 ▶

Las gráficas G_1 y G_2 son un *homomorfismo* si G_1 y G_2 se pueden reducir a gráficas isomorfas mediante una secuencia de reducciones de serie. ◀

De acuerdo con las definiciones 8.7.3 y 8.7.5, cualquier gráfica es un homomorfismo de sí misma. Además, las gráficas G_1 y G_2 son un homomorfismo si G_1 se puede reducir a una gráfica isomorfa a G_2 o si G_2 se puede reducir a una gráfica isomorfa a G_1.

Ejemplo 8.7.6 ▶

Las gráficas G_1 y G_2 de la figura 8.7.5 son homomorfas ya que ambas se pueden reducir a la gráfica G' de la figura 8.7.5 por una secuencia de reducciones de series.

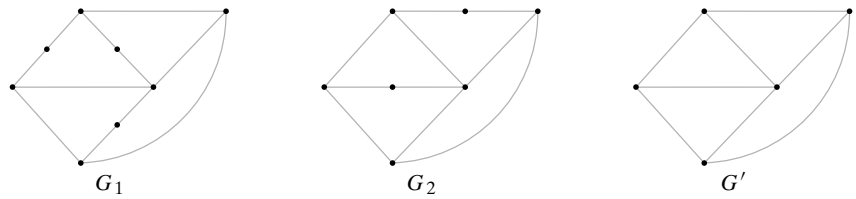


Figura 8.7.5 G_1 y G_2 son un homomorfismo; cada una se puede reducir a G'. ◀

Si se define una relación R en un conjunto de gráficas por la regla G_1RG_2 si G_1 y G_2 son un homomorfismo, R es una relación de equivalencia. Cada clase de equivalencia consiste en un conjunto de gráficas mutuamente homomorfas.

WWW

Ahora se establecerá una condición necesaria y suficiente para que una gráfica sea plana. Kuratowski fue el primero en establecer y probar el teorema en 1930. La demostración se encuentra en [Even, 1979].

Teorema 8.7.7

Teorema de Kuratowski

Una gráfica G es plana si y sólo si G no contiene una subgráfica homomorfa a K_5 o $K_{3,3}$.

Ejemplo 8.7.8 ▶

Demuestre que la gráfica G de la figura 8.7.6 no es plana mediante el teorema de Kuratowski.

Se intentará encontrar $K_{3,3}$ en la gráfica G de la figura 8.7.6. Primero observe que los vértices a, b, f y e tienen, cada uno, grado 4. En $K_{3,3}$ cada vértice tiene grado 3, de manera que se eliminan las aristas (a, b) y (f, e) para que todos los vértices tengan grado 3 (vea la figura 8.7.6). Se observa que si se eliminan una o más aristas, se obtendrán dos vértices de grado 2 y luego se pueden realizar dos reducciones de series. La gráfica resultante tendrá nueve aristas; como $K_{3,3}$ tiene nueve aristas, este enfoque parece prometedor. Por prueba y error, finalmente se ve que si se elimina la arista (g, h) y se realiza una reducción de serie, se obtiene una copia isomorfa de $K_{3,3}$ (vea la figura 8.7.7). Por lo tanto, la gráfica G de la figura 8.7.6 no es plana, puesto que contiene una subgráfica homomorfa a $K_{3,3}$.

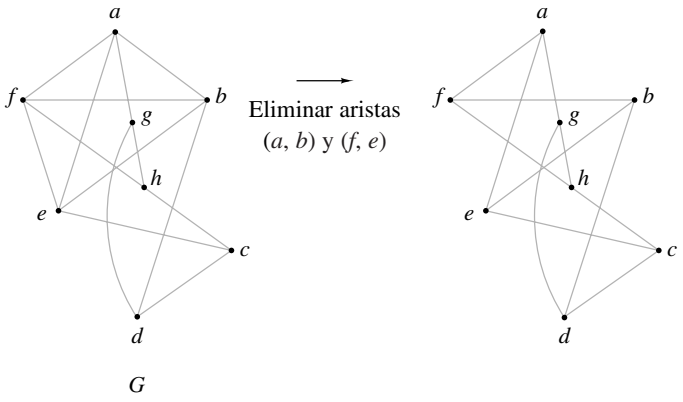


Figura 8.7.6 Eliminación de aristas para obtener una subgráfica.

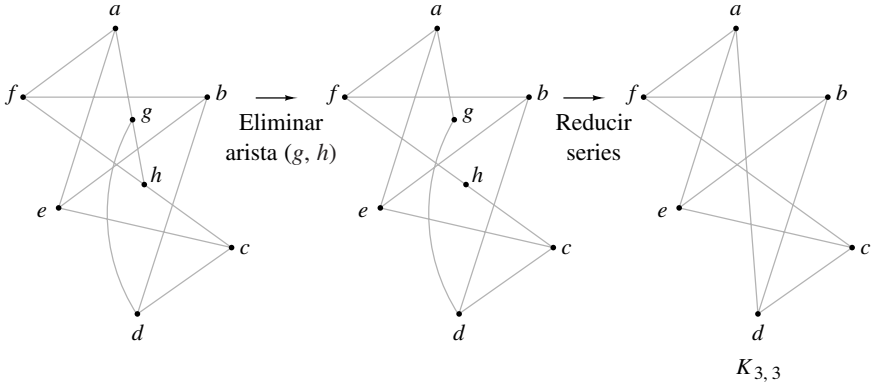


Figura 8.7.7 Eliminación de una arista para obtener una subgráfica, seguida de reducciones de series. ◀

Aunque el Teorema 8.7.7 da una caracterización elegante de las gráficas planas, no lleva a un algoritmo eficiente para reconocerlas. Sin embargo, se conocen algoritmos que pueden determinar si una gráfica que tiene n vértices es plana en un tiempo $O(n)$ (vea [Even, 1979]).

Esta sección concluye con la fórmula de Euler.

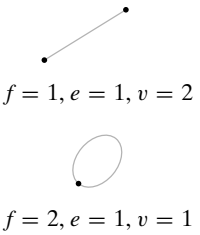


Figura 8.7.8 Paso base del Teorema 8.7.9.

Teorema 8.7.9

Fórmula de Euler para gráficas

Si G es una gráfica plana conexa con e aristas, v vértices y f caras, entonces

$$f = e - v + 2 \tag{8.7.3}$$

Demostración Se usará inducción sobre el número de aristas.

Suponga que $e = 1$. Entonces G es una de las dos gráficas mostradas en la figura 8.7.8. En cualquier caso, la fórmula se cumple. Esto verifica el paso base.

Suponga que la fórmula se cumple para gráficas planas conexas con n aristas. Sea G una gráfica con $n + 1$ aristas. Primero suponga que G no contiene ciclos. Elija un vértice v y trace una trayectoria que inicie en v. Como G no tiene ciclos, cada vez que se traza una arista, se llega a un nuevo vértice. En algún momento, se llegará al vértice a con grado 1, que no se puede dejar (vea la figura 8.7.9). Se elimina a y la arista x incidente en a de la gráfica G. La gráfica G' que resulta tiene n aristas; así, por la suposición inductiva, (8.7.3) se cumple para G'. Como G tiene una arista más que G', un vértice más que G' y el mismo número de caras, se concluye que (8.7.3) también se cumple para G.

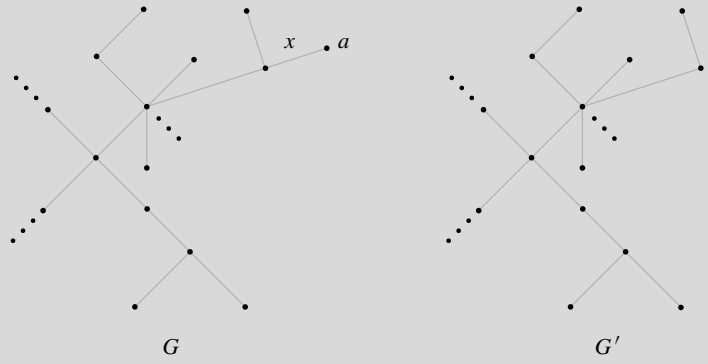


Figura 8.7.9 Prueba del Teorema 8.7.9 para el caso en que G no tiene ciclos. Se encuentra un vértice a con grado 1 y se elimina a y la arista x incidente en a.

Ahora suponga que G contiene un ciclo. Sea x una arista en un ciclo (figura 8.7.10). En este caso, x es parte de la frontera entre dos caras. Esta vez se elimina la arista x pero no los vértices, para obtener la gráfica G' (figura 8.7.10). De nuevo, G' tiene n

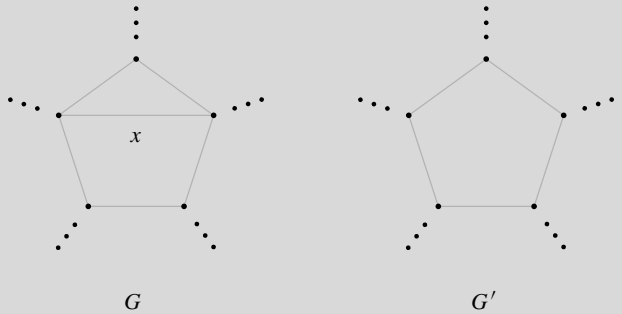


Figura 8.7.10 Prueba del Teorema 8.7.9 para el caso de que G tiene un ciclo. Se elimina la arista x en un ciclo.

aristas; entonces por la suposición inductiva, (8.7.3) se cumple para G'. Como G tiene una cara más que G', una arista más que G' y el mismo número de vértices que G', se concluye que (8.7.3) también se cumple para G.

Como esto verifica el paso inductivo, el principio de inducción matemática demuestra el teorema.

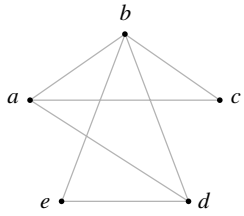
Sección de ejercicios de repaso

1. ¿Qué es una gráfica plana?
2. ¿Qué es una cara?
3. Establezca la ecuación de Euler para una gráfica plana conexa.
4. ¿Qué son aristas en serie?
5. ¿Qué es la reducción de serie?
6. Defina *gráficas homomorfas*.
7. Enuncie el teorema de Kuratowski.

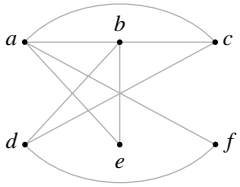
Ejercicios

En los ejercicios 1 al 3, demuestre que la gráfica es plana dibujándola de nuevo sin que se crucen las aristas.

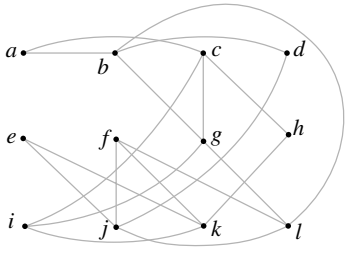
1.



2.

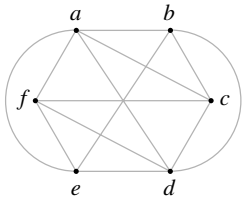


3.

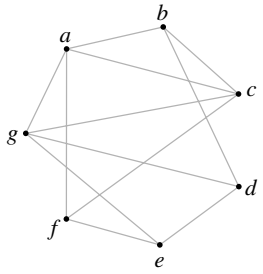


En los ejercicios 4 y 5, demuestre que la gráfica no es plana encontrando una subgráfica homomorfa a K_5 o $K_{3,3}$.

4.

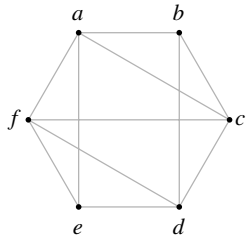


5.

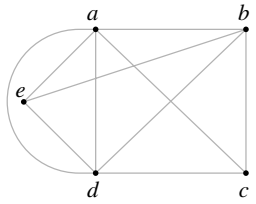


En los ejercicios 6 al 8, determine si la gráfica es plana. Si lo es, dibújela de nuevo sin que se crucen las aristas; de otra manera, encuentre una subgráfica homomorfa a K_5 o bien $K_{3,3}$.

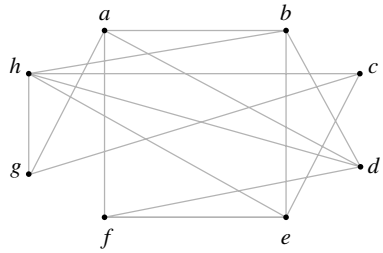
6.



7.



8.



9. Una gráfica plana conexa tiene 9 vértices que tienen grados 2, 2, 2, 3, 3, 3, 4, 4 y 5. ¿Cuántas aristas hay? ¿Cuántas caras tiene?

10. Demuestre que agregar o eliminar ciclos, aristas paralelas o aristas en serie no afecta el hecho de que una gráfica sea plana.

11. Demuestre que cualquier gráfica que tiene 4 vértices o menos es plana.

12. Demuestre que cualquier gráfica que tiene 5 vértices o menos y un vértice de grado 2 es plana.

13. Demuestre que en cualquier gráfica simple, conexa, plana, $e \leq 3v - 6$.

14. Dé un ejemplo de una gráfica simple, conexa y no plana para la que $e \geq 3v - 6$.

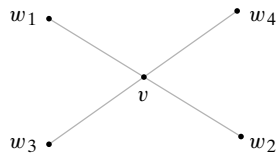
15. Use el ejercicio 13 para demostrar que K_5 no es plana.

★**16.** Demuestre que si una gráfica simple G tiene 11 vértices o más, entonces una de las dos, G o su complemento $\overline{G}$, no es plana.

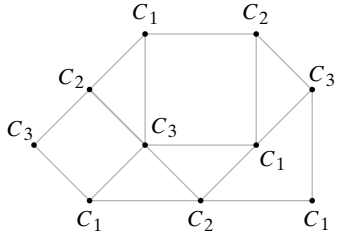
★**17.** Demuestre que si una gráfica plana tiene un ciclo de Euler, tiene un ciclo de Euler sin cruces. Una trayectoria P en una gráfica plana tiene un *cruce* si un vértice v aparece al menos dos veces en P y P se cruza a sí misma en v; es decir,

$$P = (\ldots, w_1, v, w_2, \ldots, w_3, v, w_4, \ldots),$$

donde los vértices se arreglan de manera que w_1, v, w_2 cruza a w_3, v, w_4 en v como en la figura siguiente.

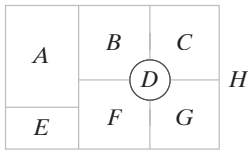


Un coloreado de una gráfica G con los colores C_1, C_2, . . . , C_n asigna a cada vértice un color C_i de manera que cada vértice tiene un color diferente del de los vértices adyacentes. Por ejemplo, la siguiente gráfica tiene tres colores. El resto de los ejercicios tienen que ver con colorear gráficas.



Un mapa plano es una gráfica plana donde las caras se interpretan como los países, las aristas son las fronteras entre esos países y los vértices representan las intersecciones de las fronteras. El problema de colorear un mapa plano G, de manera que los países con fronteras comunes no tengan el mismo color, se puede reducir al problema de colorear una gráfica construyendo primero la gráfica dual G′ de G de la siguiente manera. Los vértices de la gráfica dual G′ consisten en un punto en cada cara de G, incluso en la cara no acotada. Una arista en G′ conecta dos vértices si las caras correspondientes en G están separadas por una frontera. Colorear el mapa G es equivalente a colorear los vértices de la gráfica dual G′.

18. Encuentre el dual del siguiente mapa.

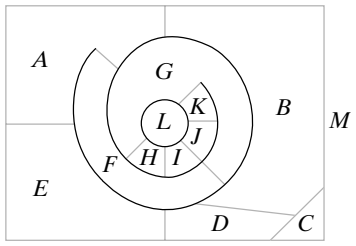


19. Demuestre que el dual de un mapa plano es una gráfica plana.

20. Demuestre que cualquier coloreado del mapa del ejercicio 18, excepto la región no acotada, requiere al menos 3 colores.

21. Coloree el mapa del ejercicio 18, excepto la región no acotada, usando tres colores.

22. Encuentre el dual del siguiente mapa.

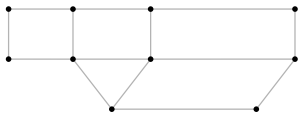


23. Demuestre que cualquier coloreado del mapa del ejercicio 22, excepto la región no acotada, requiere al menos 4 colores.

24. Coloree el mapa del ejercicio 22, excepto la región no acotada, usando 4 colores.

Una triangulación de una gráfica G plana simple se obtiene de G conectando el mayor número de vértices posible al tiempo que se mantiene la naturaleza plana al no introducir lazos o aristas paralelas.

25. Encuentre una triangulación de la siguiente gráfica.



26. Demuestre que si una triangulación G' de una gráfica G plana simple se puede colorear con n colores, y también G.

27. Demuestre que en una triangulación de una gráfica G plana simple, $3f = 2e$.

Appel y Haken probaron (vea [Appel]) que toda gráfica plana simple se puede colorear con cuatro colores. El problema se planteó a mediados del siglo XIX y durante años nadie tuvo éxito en dar una prueba. Quienes trabajan en el problema de cuatro colores en años recientes han tenido una ventaja sobre sus predecesores: el uso de las computadoras electrónicas rápidas. El siguiente ejercicio muestra cómo comienza la prueba.

Suponga que se tiene una gráfica plana simple que requiere más de cuatro colores para colorearla. Entre este tipo de gráficas, existe una con el número mínimo de vértices. Sea G una triangulación de esta gráfica. Entonces G también tiene el número mínimo de vértices y por el ejercicio 26, G requiere más de cuatro colores para colorearla.

28. Si el dual de un mapa tiene un vértice de grado 3, ¿cuál es la apariencia del mapa original?

29. Demuestre que G no tiene un vértice de grado 3.

★***30.*** Demuestre que G no tiene un vértice de grado 4.

★***31.*** Demuestre que G tiene un vértice de grado 5.

La contribución de Appel y Haken fue demostrar que sólo era necesario considerar y analizar un número finito de casos que incluyen el vértice de grado 5, y demostrar que todos se pueden colorear usando 4 colores. La reducción a un número finito de casos se facilitó al usar la computadora para ayudar a encontrar los casos que requieren análisis. De nuevo se usó la computadora para analizar los casos que resultan.

★**32.** Demuestre que cualquier gráfica simple plana se puede colorear usando 5 colores.

8.8 ➜ Locura instantánea†

Locura instantánea es un juego que consiste en cuatro cubos con cada cara pintada en uno de cuatro colores: rojo, blanco, azul o verde (vea la figura 8.8.1). (Existen diferentes versiones del juego, dependiendo de cómo se asignen los colores a las caras). El problema es apilar los cubos, uno encima del otro, de manera que sin importar desde dónde se vean (enfrente, atrás, derecha o izquierda), siempre se verán los cuatro colores (figura 8.8.2). Como son posibles 331,776 apilados diferentes (vea el ejercicio 12), una solución manual por prueba y error es impráctica. Aquí se presenta una solución, usando un modelo de gráficas, que hace posible descubrir la solución, si acaso existe, en unos cuantos minutos.

WWW

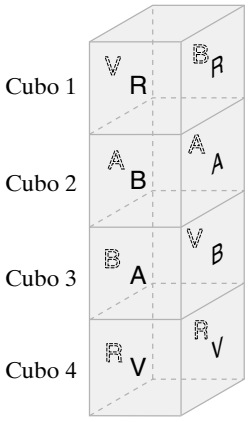


Figura 8.8.2 Una solución al juego de Locura instantánea de la figura 8.8.1.

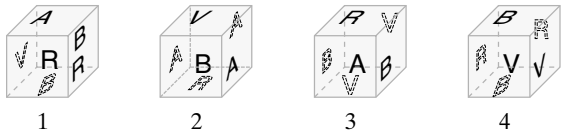
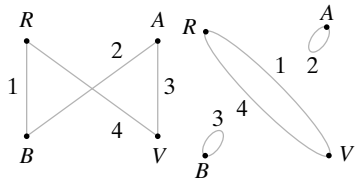


Figura 8.8.1 Juego de Locura instantánea.

Primero observe que un apilado en particular se puede representar mediante dos gráficas, una para los colores del frente/atrás y otra para los de derecha/izquierda. Por ejemplo, en la figura 8.8.3 se representa el apilado de la figura 8.8.2. Los vértices representan los colores, y las aristas conectan dos vértices si las caras opuestas tienen esos colores. Por ejemplo, en la gráfica del frente/atrás, la arista con etiqueta 1 conecta *R* y *B*, ya que las caras de enfrente y atrás del cubo 1 son roja (*R*) y blanca (*B*). Como otro ejemplo, en la gráfica de derecha/izquierda, *B* tiene un lazo, pues las dos caras, izquierda y derecha del cubo 3 son blancas.

También es posible construir un apilado a partir de un par de gráficas como las de la figura 8.8.3, que representan una solución del juego de Locura instantánea. Comenzamos con la gráfica frente/atrás. El cubo 1 debe tener rojo y blanco en caras opuestas. De manera arbitraria, se asigna uno de estos colores, digamos rojo, al frente. Entonces el cubo 1 tiene una cara blanca atrás. La otra arista incidente en *B* es 2, de manera que la cara del frente del cubo 2 debe ser blanca. Esto da al cubo 2 una cara azul atrás. La otra arista incidente en *A* es 3, y se hace azul la cara de enfrente del cubo 3. Esto da al cubo 3 una cara verde atrás. La otra arista incidente en *V* es 4. El cubo 4 obtiene entonces una cara verde al frente y una roja atrás. Las caras de enfrente y atrás tienen una alineación adecuada. En este punto, las caras de izquierda y derecha se arreglan al azar; sin embargo, se demostrará cómo orientar de manera correcta las caras de izquierda y derecha sin alterar los colores de enfrente y atrás.

El cubo 1 debe tener rojo y verde en las caras opuestas izquierda y derecha. Se asigna uno de estos colores, por ejemplo verde, a la izquierda. Entonces el cubo 1 tiene una cara roja en la derecha. Observe que al rotar el cubo se puede obtener esta orientación izquierda/derecha sin cambiar los colores de frente/atrás (figura 8.8.4). Se pueden orien-



(a) Frente/atrás (b) Izquierda/derecha

Figura 8.8.3 Gráficas que representan el apilado de la figura 8.8.2.

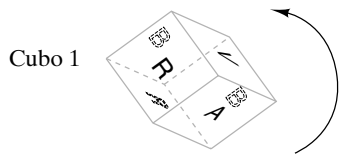


Figura 8.8.4 El cubo se gira para obtener una orientación izquierda/derecha, sin cambiar los colores del frente/atrás.

† Esta sección se puede omitir sin pérdida de continuidad.

tar los cubos 2, 3 y 4 de manera similar. Observe que los cubos 2 y 3 tienen los mismos colores en lados opuestos. Se reconstruyó el apilado de la figura 8.8.2.

Es aparente a partir del análisis anterior que se puede obtener una solución al juego de Locura instantánea si se encuentran dos gráficas como las de la figura 8.8.3. Las propiedades necesarias son:

- Cada vértice debe tener grado 2. **(8.8.1)**
- Cada cubo debe estar representado por una arista exactamente una vez en cada gráfica. **(8.8.2)**
- No debe haber aristas en común entre las dos gráficas. **(8.8.3)**

La propiedad (8.8.1) asegura que cada color se puede usar dos veces, una vez en el frente (o izquierda) y otra atrás (o derecha). La propiedad (8.8.2) asegura que cada cubo se usa exactamente una vez. La propiedad (8.8.3) asegura que, después de orientar los lados del frente y atrás, es posible orientar con éxito los lados izquierdo y derecho.

Para obtener una solución, primero se dibuja una gráfica G que representa todas las caras de todos los cubos. Los vértices de G representan los cuatro colores, y una arista con etiqueta i conecta dos vértices (colores) si las caras opuestas del cubo i tienen esos colores. En la figura 8.8.5 se dibujó la gráfica que representa a los cubos de la figura 8.8.1. Después, por inspección, se encuentran dos subgráficas de G que satisfacen las propiedades (8.8.1) a (8.8.3). Intente el método encontrando otra solución al juego representado en la figura 8.8.5.

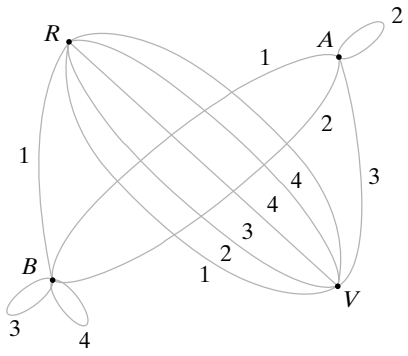


Figura 8.8.5 Representación gráfica del juego de Locura instantánea de la figura 8.8.1.

Ejemplo 8.8.1 ▶

Encuentre una solución al juego de Locura instantánea de la figura 8.8.6.

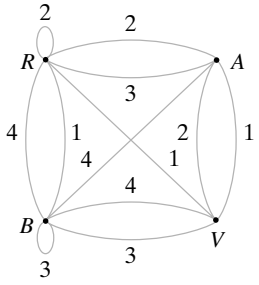


Figura 8.8.6 El juego de la Locura instantánea para el ejemplo 8.8.1.

Comenzamos por tratar de construir una subgráfica con las propiedades (8.8.1) y (8.8.2). De manera arbitraria se elige un vértice, digamos A, y dos aristas incidentes en A. Suponga que se seleccionan las dos aristas mostradas como líneas continuas en la figura 8.8.7. Ahora considere el problema de elegir dos aristas incidentes en R. No se pueden seleccionar aristas que inciden en A o V porque A y V deben tener grado 2. Como cada cubo debe aparecer en cada subgráfica exactamente una vez, no se puede seleccionar ninguna arista con etiqueta 1 o 2 porque ya se seleccionaron las aristas con estas etiquetas. Las aristas que inciden en R que no se pueden seleccionar se muestran punteadas en la figura 8.8.7. Esto deja sólo la arista con etiqueta 4. Dado que se necesitan dos aristas incidentes en R, la selección inicial de aristas incidentes en B debe revisarse.

Para el siguiente intento de elegir dos aristas incidentes en el vértice A, se seleccionan dos aristas etiquetadas 2 y 3, como se indica en la figura 8.8.8. Como esta elección incluye una arista incidente en R, debemos elegir una arista adicional que incida en R. Se tienen tres posibilidades para escoger la arista adicional (mostradas con líneas gruesas en la figura 8.8.8). (El lazo incidente en R cuenta como dos aristas y no se puede elegir). Si se selecciona la arista con etiqueta 1 incidente en R y V, se necesita un lazo en B con etiqueta 4. Como no hay tal lazo, no se selecciona esta arista. Si se elige la arista con etiqueta 1 incidente en R y B, se puede después elegir la arista con etiqueta 4, incidente en B y V (vea la figura 8.8.9). Con esto se obtiene una de las gráficas.

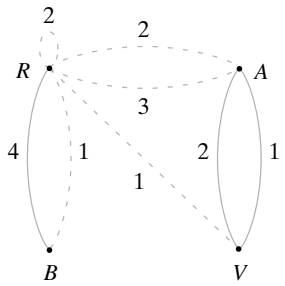


Figura 8.8.7 Intento para encontrar una subgráfica de la figura 8.8.6 que satisfaga (8.8.1) y (8.8.2).

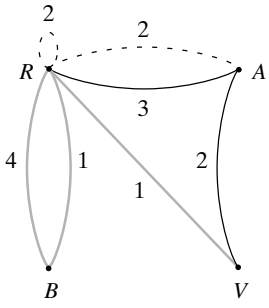


Figura 8.8.8 Otro intento para encontrar una subgráfica de la figura 8.8.6 que satisfaga (8.8.1) y (8.8.2).

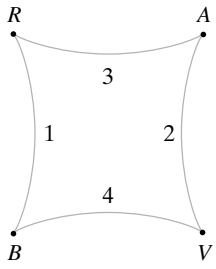


Figura 8.8.9 Subgráfica de la figura 8.8.6 que satisface (8.8.1) y (8.8.2).

Ahora se buscará una segunda gráfica que no tenga aristas en común con la gráfica que acaba de elegirse. De nuevo se inicia seleccionando dos aristas incidentes en *B*. Como no es posible reutilizar aristas, las opciones están limitadas a tres aristas (figura 8.8.10). Elegir las aristas con etiquetas 1 y 4 lleva a la gráfica de la figura 8.8.11. Las gráficas de las figuras 8.8.9 y 8.8.11 resuelven el juego de Locura instantánea de la figura 8.8.6.

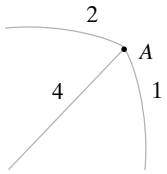


Figura 8.8.10 Aristas incidentes en *A* no usadas en la figura 8.8.9.

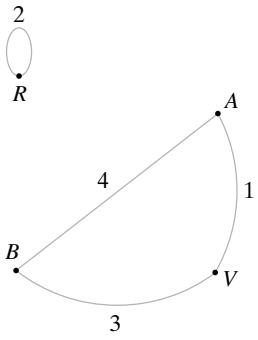


Figura 8.8.11 Otra subgráfica de la figura 8.8.6 sin aristas comunes con la figura 8.8.9, que satisface (8.8.1) y (8.8.2). Esta gráfica y la de la figura 8.8.9 resuelven el juego de Locura instantánea de la figura 8.8.6. ◀

Sección de ejercicios de repaso

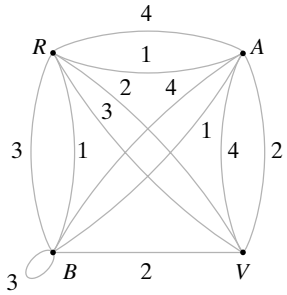
1. Describa el juego de Locura instantánea.

2. Describa cómo resolver el juego de Locura instantánea.

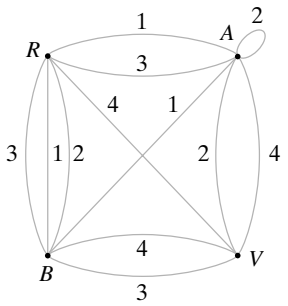
Ejercicios

Encuentre soluciones a los siguientes juegos de Locura instantánea.

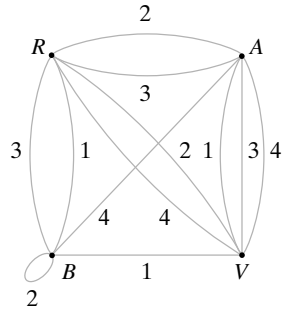
1.



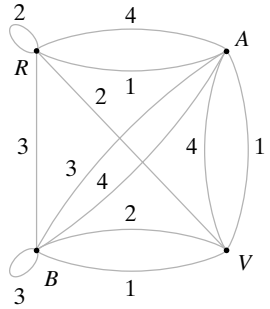
2.



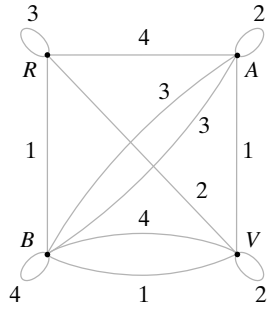
3.



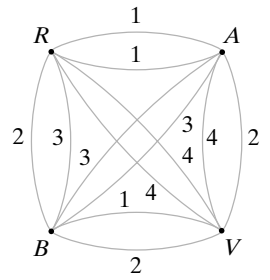
4.



5.



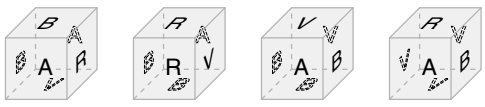
6.



7. *a)* Encuentre todas las subgráficas de la figura 8.8.5 que satisfacen las propiedades (8.8.1) y (8.8.2).

b) Encuentre todas las soluciones del juego de Locura instantánea de la figura 8.8.5.

8. *a)* Represente el juego de Locura instantánea mediante una gráfica.

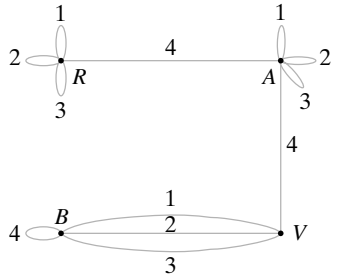


b) Encuentre una solución al juego.

c) Encuentre todas las subgráficas de la gráfica en el inciso *a)* que satisfacen las propiedades (8.8.1) y (8.8.2).

d) Use *c)* para demostrar que el juego tiene una solución única.

9. Demuestre que el siguiente juego de Locura instantánea no tiene solución, mediante un argumento para probar que ninguna subgráfica satisface las propiedades (8.8.1) y (8.8.2). Observe que no hay solución aun cuando cada cubo contiene los cuatro colores.



★**10.** Dé un ejemplo de un juego de Locura instantánea que satisfaga:

a) No tiene solución.

b) Cada cubo contiene los cuatro colores.

c) Hay una subgráfica que satisface las propiedades (8.8.1) y (8.8.2).

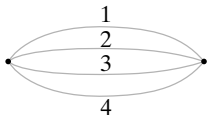
11. Demuestre que hay 24 orientaciones de un cubo.

12. Numere los cubos de un juego de Locura instantánea 1, 2, 3 y 4. Demuestre que el número de maneras en que se apilan los cubos 1, 2, 3 y 4, leyendo de abajo arriba, es 331,776.

★***13.*** ¿Cuántas gráficas de Locura instantánea hay; es decir, cuántas gráficas hay con cuatro vértices y 12 aristas, tres de cada uno de los cuatro tipos?

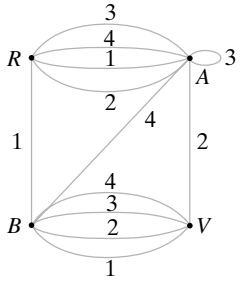
Los ejercicios 14 al 21 se refieren a una versión modificada de Locura instantánea donde una solución se define como un apilado que, al verse desde enfrente, atrás, izquierda o derecha muestra un solo color. (El frente, atrás, izquierda y derecha son de colores diferentes).

14. Dé un argumento que muestre que si se grafica el juego como la Locura instantánea normal, una solución para el juego modificado consiste en dos subgráficas de la forma mostrada en la figura, sin aristas o vértices en común.

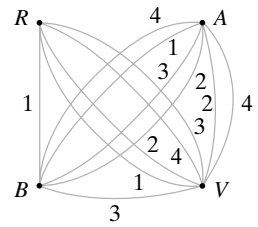


Encuentre soluciones a Locura instantánea modificado para los siguientes juegos.

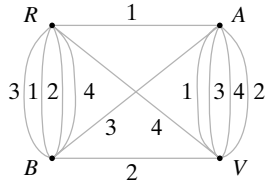
15.



16.



17.



18. Gráfica del ejercicio 6

19. Demuestre que para la figura 8.8.5, Locura instantánea, como se presenta en el texto, tiene una solución, pero la versión modificada no tiene solución.

★20. Demuestre que si Locura instantánea modificada tiene una solución, la versión presentada en el texto también debe tener una solución.

21. ¿Es posible que ninguna versión de Locura instantánea tenga una solución aun cuando cada cubo contenga los cuatro colores? Si la respuesta es sí, pruébelo; de otra manera, dé un contraejemplo.

Notas

Prácticamente cualquier referencia de matemáticas discretas contiene un capítulo o más sobre teoría de gráficas. Los libros específicos de teoría de gráficas son [Berge; Bondy; Chartrand; Deo; Even, 1979; Gibbons; Harari; König; Ore; West, y Wilson]. [Deo; Even, 1979; y Gibbons] ponen énfasis en los algoritmos de gráficas. [Brassard, Cormen y Johnsonbaugh] también estudian las gráficas y los algoritmos de gráficas.

[Akl; Leighton; Lester; Lewis; Miller, y Quinn] analizan las computadoras en paralelo y los algoritmos para computación en paralelo. Nuestros resultados acerca de las subgráficas del hipercubo se tomaron de [Saad].

El artículo original de Euler de los puentes de Königsberg, editado por J. R. Newman, se reimprimió como [Newman].

En [Gardner, 1959], los ciclos hamiltonianos se relacionan con el juego de la Torre de Hanoi.

En muchos casos, los métodos llamados de ramificación y acotamiento (vea por ejemplo, [Tucker]) con frecuencia proporcionan soluciones al problema del agente viajero de manera más eficiente que la búsqueda exhaustiva.

El algoritmo de la ruta más corta de Dijkstra aparece en [Dijkstra, 1959].

La complejidad del problema de isomorfismo entre gráficas se analiza en [Köbler].

Appel y Haken publicaron su solución para el problema de los cuatro colores en [Appel].

Repaso del capítulo

Sección 8.1

1. Gráfica $G = (V, E)$ (no dirigida y dirigida)
2. Vértice
3. Arista
4. La arista e incide en el vértice v
5. El vértice v es incidente en la arista e
6. v y w son vértices adyacentes
7. Aristas paralelas
8. Lazo
9. Vértice aislado
10. Gráfica simple
11. Gráfica ponderada
12. Peso de una arista
13. Longitud de una trayectoria en una gráfica ponderada
14. Similitud gráfica
15. Función de disimilitud
16. Cubo-n (hipercubo)
17. Computación en serie
18. Algoritmo en serie

19. Computación en paralelo
20. Algoritmo en paralelo
21. Gráfica completa sobre n vértices, K_n
22. Gráfica bipartita
23. Gráfica bipartita completa sobre m y n vértices, $K_{m,n}$

Sección 8.2

24. Trayectoria o ruta
25. Trayectoria simple
26. Ciclo
27. Ciclo simple
28. Gráfica conexa
29. Subgráfica
30. Componente de una gráfica
31. Grado de un vértice $\delta(v)$
32. Problema del puente de Königsberg
33. Ciclo de Euler
34. Una gráfica G tiene un ciclo de Euler si y sólo si G es conexa y todos los vértices tienen grado par.
35. La suma de los grados de todos los vértices en una gráfica es igual al doble del número de aristas.
36. En una gráfica, hay un número par de vértices de grado impar.
37. Una gráfica tiene una trayectoria sin aristas repetidas de v a w ($v \neq w$) que contiene a todas las aristas y vértices si y sólo si es conexa y v y w son los únicos vértices que tienen grado impar.
38. Si una gráfica G contiene un ciclo de v a v, G contiene un ciclo simple de v a v.

Sección 8.3

39. Ciclo de Hamilton
40. Problema del agente viajero
41. Modelo de anillo para computación en paralelo
42. Código Gray

Sección 8.4

43. Algoritmo de la ruta más corta de Dijkstra

Sección 8.5

44. Matriz de adyacencia
45. Si A es la matriz de una gráfica simple, el elemento ij de A^n es igual al número de trayectorias de longitud n del vértice i al vértice j
46. Matriz de incidencia

Sección 8.6

47. Las gráficas $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$ son isomorfas si existen funciones uno a uno y sobre f: $V_1 \rightarrow V_2$ y g: $E_1 \rightarrow E_2$ tales que $e \in E_1$ incide en v, $w \in V_1$ si y sólo si $g(e)$ incide en f(v) y $f(w)$.
48. Las gráficas G_1 y G_2 son isomorfas si y sólo si, para algún orden de sus vértices, sus matrices de adyacencia son iguales.
49. Modelo de rejilla para computación en paralelo
50. Invariante

Sección 8.7

51. Gráfica plana
52. Cara
53. Ecuación de Euler para gráficas planas conexas: $f = e - v + 2$
54. Aristas en serie
55. Reducción de serie
56. Gráficas homomorfas
57. Teorema de Kuratowski: una gráfica es plana si y sólo si no contiene una subgráfica homomorfa a K_5 o $K_{3,3}$.

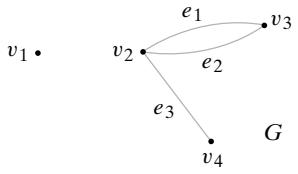
Sección 8.8

58. Locura instantánea
59. Cómo resolver un juego de Locura instantánea

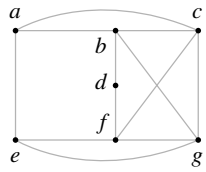
Autoevaluación del capítulo

Sección 8.1

1. Para la gráfica $G = (V, E)$, encuentre V, E y todas las aristas paralelas, ciclos y vértices aislados, y establezca si G es una gráfica simple. Además establezca en qué vértices incide e_3 y en qué aristas incide el vértice v_2.



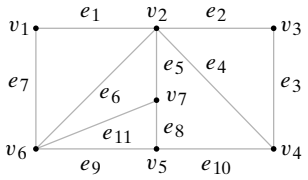
2. Explique por qué la gráfica no tiene una trayectoria de a a a que pase por cada arista exactamente una vez.



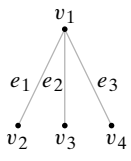
3. Dibuje $K_{2,5}$, la gráfica bipartita completa sobre 2 y 5 vértices.
4. Pruebe que el cubo-n es bipartita para toda $n \geq 1$.

Sección 8.2

5. Diga si la trayectoria $(v_2, v_3, v_4, v_2, v_6, v_1, v_2)$ en la gráfica es una trayectoria simple, un ciclo, un ciclo simple, o ninguno de éstos.



6. Dibuje todas las subgráficas que contienen exactamente dos aristas de la siguiente gráfica.

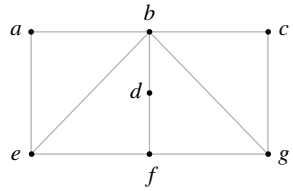


7. Encuentre una subgráfica conexa de la gráfica del ejercicio 5 que contenga todos los vértices de la gráfica original y tenga el menor número de aristas posible.
8. ¿Contiene la gráfica del ejercicio 5 un ciclo de Euler? Explique su respuesta.

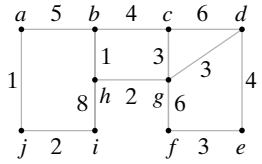
Sección 8.3

9. Encuentre un ciclo de Hamilton en la gráfica del ejercicio 5.
10. Encuentre un ciclo de Hamilton en el cubo-3.

11. Demuestre que la gráfica no tiene un ciclo hamiltoniano.

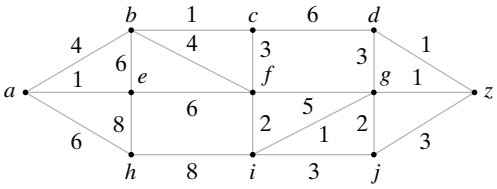


12. Demuestre que el ciclo $(a, b, c, d, e, f, g, h, i, j, a)$ proporciona una solución al problema del agente viajero para la gráfica mostrada.



Sección 8.4

Los ejercicios 13 al 16 se refieren a la gráfica siguiente.



13. Encuentre la longitud de una ruta más corta de a a i.
14. Encuentre la longitud de una ruta más corta de a a z.
15. Encuentre una ruta más corta de a a z.
16. Encuentre la longitud de una ruta más corta de a a z que pase por c.

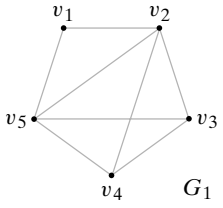
Sección 8.5

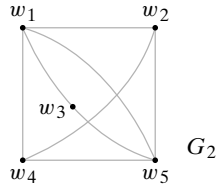
17. Escriba la matriz de adyacencia de la gráfica del ejercicio 5.
18. Escriba la matriz de incidencia de la gráfica del ejercicio 5.
19. Si A es la matriz de adyacencia de la gráfica del ejercicio 5, ¿qué representa el elemento en el renglón v_2 y la columna v_3 de A^3?
20. ¿Puede una columna de una matriz de incidencia tener sólo ceros? Explique.

Sección 8.6

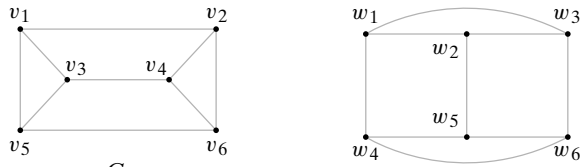
En los ejercicios 21 y 22, determine si las gráficas G_1 y G_2 son isomorfas. Si las gráficas son isomorfas, dé los ordenamientos de sus vértices que producen matrices de adyacencia iguales. Si las gráficas no son isomorfas, dé una invariante que no compartan las gráficas.

21.





22.



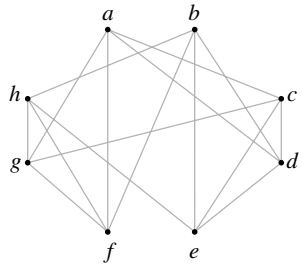
23. Dibuje todas las gráficas simples no isomorfas que tienen exactamente cinco vértices y dos aristas.

24. Dibuje todas las gráficas simples no isomorfas que tienen exactamente cinco vértices, dos componentes y ningún ciclo.

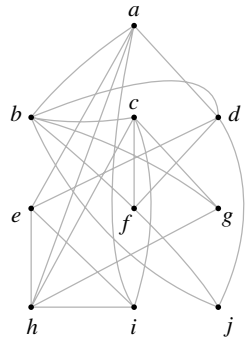
Sección 8.7

En los ejercicios 25 y 26, determine si la gráfica es plana. Si lo es, dibújela de nuevo de manera que las aristas no se crucen; de otra manera, encuentre una subgráfica homomorfa a K_5 o a $K_{3,3}$.

25.



26.

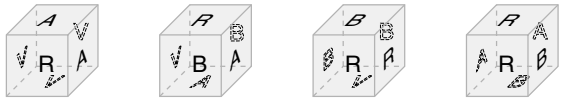


27. Demuestre que cualquier gráfica conexa simple con 31 aristas y 12 vértices no es plana.

28. Demuestre que el cubo-n es plano si $n \leq 3$ *y no plana si* $n > 3$.

Sección 8.8

29. Represente el juego de Locura instantánea mediante una gráfica.



1 2 3 4

30. Encuentre una solución al juego del ejercicio 29.

31. Encuentre todas las subgráficas de la gráfica del ejercicio 29 que satisfacen las propiedades (6.8.1) y (6.8.2).

32. A partir del ejercicio 31 determine cuántas soluciones tiene el juego del ejercicio 29.

Ejercicios para computadora

1. Escriba un programa que acepte como entrada cualquiera de los siguientes elementos:

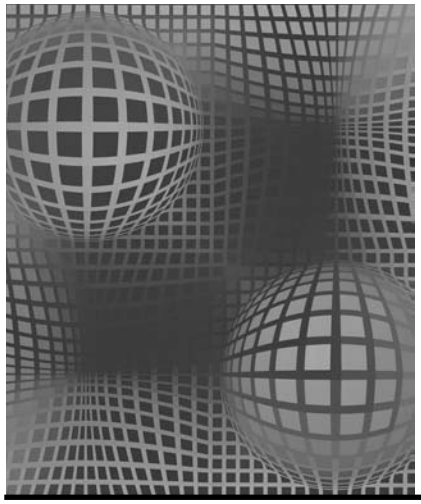
- una lista de las aristas de una gráfica dada como pares de enteros positivos
- la matriz de adyacencia
- la matriz de incidencia

y proporcione como salida los otros dos.

2. Escriba un programa que determine si una gráfica contiene un ciclo de Euler.

3. Escriba un programa que encuentre un ciclo de Euler en una gráfica conexa en la que todos los vértices tienen grado par.

4. Escriba un programa que genere de manera aleatoria una matriz de adyacencia de $n \times n$. Haga que imprima la matriz de adyacencia, el número de aristas, el número de lazos y el grado de cada vértice.
5. Escriba un programa que determine si una gráfica es bipartita. Si lo es, el programa debe listar los conjuntos ajenos de vértices.
6. Escriba un programa que acepte como entrada las aristas de una gráfica y luego dibuje la gráfica usando el despliegue gráfico de la computadora.
7. Escriba un programa que liste todas las trayectorias simples entre dos vértices dados.
8. Escriba un programa que determine si una trayectoria es una trayectoria simple, un ciclo, o un ciclo simple.
9. Escriba un programa que verifique si un ciclo propuesto es un ciclo de Hamilton.
10. Escriba un programa que verifique si una trayectoria propuesta es una trayectoria de Hamilton.
11. Escriba un programa que construya un código Gray.
12. Implemente el algoritmo de la ruta más corta de Dijkstra como un programa. Este programa debe encontrar una ruta más corta y su longitud.
13. Escriba un programa que pruebe si un isomorfismo propuesto es de hecho un isomorfismo.
14. Escriba un programa para determinar si una gráfica es plana.
15. Escriba un programa que resuelva un juego de Locura instantánea.



Capítulo 9

ÁRBOLES

Hablo con los árboles, pero no me escuchan.

DE PAINT YOUR WAGON

Los árboles forman una de las subclases de gráficas que más se utilizan. En particular, la ciencia de la computación hace uso de los árboles ampliamente. En computación, los árboles son útiles para organizar y relacionar datos en una base de datos (vea el ejemplo 9.1.7). Los árboles surgen en problemas teóricos como el tiempo óptimo para ordenar (vea la sección 9.7).

Este capítulo comienza con los requisitos de terminología. Se observan las subclases de árboles (como árboles con raíz y árboles binarios) y muchas aplicaciones de árboles (como árboles de expansión, árboles de decisión y árboles de juegos). El análisis de isomorfismos de árboles es una extensión del análisis de la sección 8.6 dedicada a los isomorfismos de gráficas.

9.1 ➜ Introducción

WWW

La figura 9.1.1 muestra los resultados de las semifinales y finales de la competencia de tenis clásico en Wimbledon, que incluyó cuatro de los mejores jugadores en la historia del tenis. En Wimbledon, cuando un jugador pierde, sale del torneo. Los ganadores siguen jugando hasta que queda sólo una persona: el campeón. (Esta competencia se conoce como *torneo por eliminación sencilla*). La figura 9.1.1 muestra que, en las semifinales, Mónica Seles venció a Martina Navratilova y Steffi Graf venció a Gabriela Sabatini. Las ganadoras, Seles y Graf, jugaron y Graf venció a Seles. Steffi Graf, al ser la única jugadora invicta se convirtió en la campeona de Wimbledon.

Si el torneo por eliminación sencilla de la figura 9.1.1 es visto como una gráfica (figura 9.1.2), se obtiene un **árbol**. Si se rota la figura 9.1.2, puede ser visto como un árbol natural (figura 9.1.3). La definición formal es la siguiente.

† Esta sección se puede omitir sin pérdida de continuidad.

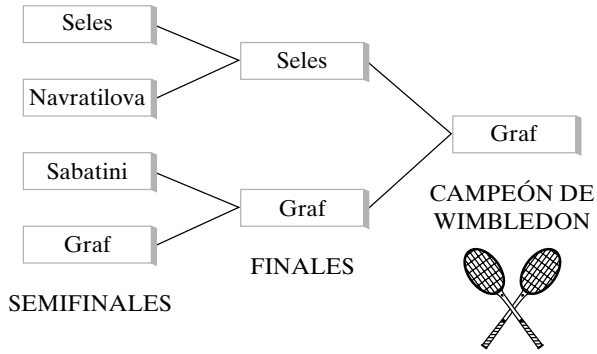


Figura 9.1.1 Semifinales y finales en Wimbledon.

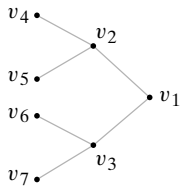
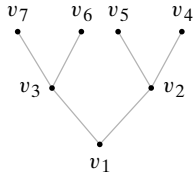
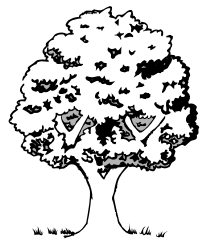


Figura 9.1.2 El torneo de la figura 9.1.1 como un árbol.



(a)



(b)

Figura 9.1.3 Árbol de la figura 9.1.2 *a)* girado, *b)* comparado con un árbol natural.

Definición 9.1.1 ▶

Un *árbol T (libre)* es una gráfica simple que satisface lo siguiente: si v y w son vértices en T, existe una trayectoria simple única de v a w.

Un *árbol con raíz* es un árbol en el que un vértice específico se designa como raíz. ◀

Ejemplo 9.1.2 ▶

Si se designa al ganador como raíz, el torneo por eliminación sencilla de la figura 9.1.1 (o figura 9.1.2) es un árbol con raíz. Observe que si v y w son vértices en esta gráfica, existe una trayectoria simple única de v a w. Por ejemplo, la trayectoria simple única de v_2 a v_7 es (v_2, v_1, v_3, v_7). ◀

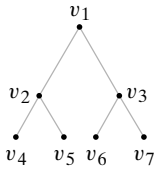


Figura 9.1.4 El árbol de la figura 9.1.3 *a)* con la raíz arriba.

Al contrario de los árboles naturales, cuyas raíces se localizan abajo, en teoría de gráficas los árboles con raíces suelen dibujarse con la raíz hacia arriba. La figura 9.1.4 presenta la forma en que se dibujaría el árbol de la figura 9.1.2 (con v_1 como raíz). Primero, se coloca la raíz v_1 arriba. Abajo de la raíz y al mismo nivel, se colocan los vértices v_2 y v_3, a los que se puede llegar desde la raíz por una trayectoria simple de longitud 1. Abajo de estos vértices y al mismo nivel se colocan los vértices v_4, v_5, v_6 y v_7, a los que se llega desde la raíz por trayectorias simples de longitud 2. Se continúa así hasta dibujar el árbol completo. Como la trayectoria simple de la raíz a cualquier vértice dado es única, cada vértice está en un nivel determinado de manera única. El nivel de la raíz es el nivel 0. Se dice que los vértices abajo de la raíz están en el nivel 1, y así sucesivamente. Entonces el **nivel de un vértice** v es la longitud de la trayectoria simple de la raíz a v. La **altura** de un árbol con raíz es el número máximo de nivel que ocurre.

Ejemplo 9.1.3 ▶

Los vértices v_1, v_2, v_3, v_4, v_5, v_6, v_7 en el árbol con raíz de la figura 9.1.4 están (respectivamente) en los niveles 0, 1, 1, 2, 2, 2, 2. La altura del árbol es 2. ◀

Ejemplo 9.1.4 ▶

Si se designa e como la raíz del árbol T de la figura 9.1.5, se obtiene el árbol con raíz T' de la figura 9.1.5. Los vértices a, b, c, d, e, f, g, h, i, j están (respectivamente) en los niveles 2, 1, 2, 1, 0, 1, 1, 2, 2, 3. La altura de T' es 3.

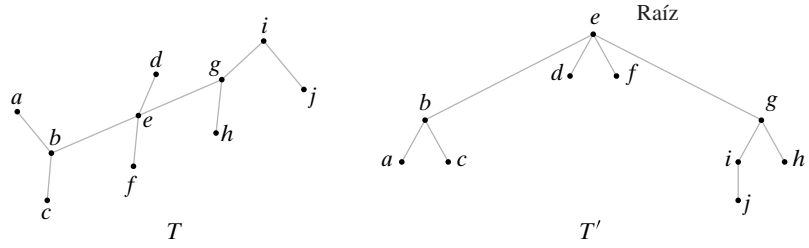


Figura 9.1.5 Un árbol T y un árbol con raíz T'. T' se obtiene de T designando a e como la raíz.

◀

Ejemplo 9.1.5 ▶

Con frecuencia se usa un árbol con raíz para especificar relaciones jerárquicas. Cuando se usa un árbol de esta manera, si un vértice a está en un nivel uno menos que el nivel del vértice b y a y b son adyacentes, entonces a está "justo arriba" de b y existe una relación lógica entre a y b: a domina a b o b es subordinado de a en alguna forma. Un ejemplo de este tipo de árbol, que es el organigrama administrativo de la organización de una universidad hipotética, se da en la figura 9.1.6.

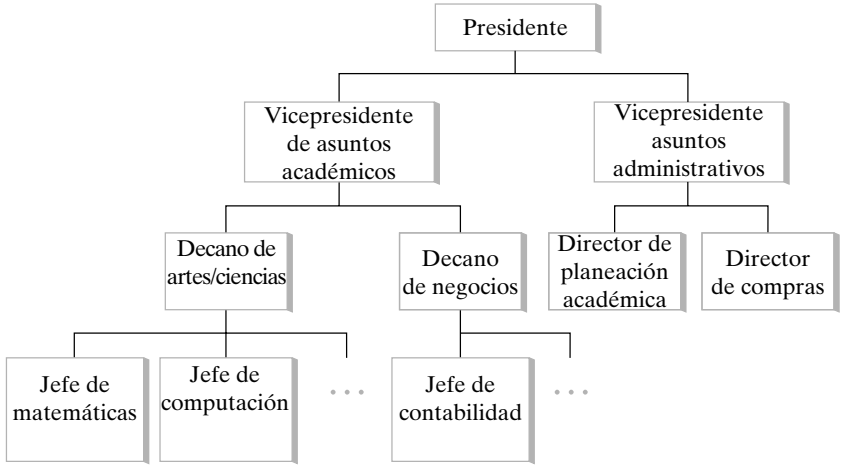


Figura 9.1.6 Organigrama administrativo organizacional.

◀

Ejemplo 9.1.6 ▶

Sistemas de archivos de computadora

Los sistemas operativos de las computadoras modernas organizan las carpetas y los archivos usando una estructura de árbol. Una *carpeta* contiene otras carpetas y archivos. La figura 9.1.7 muestra el explorador de Windows con el despliegue de carpetas a la izquierda y los archivos a la derecha en una computadora en particular. La figura 9.1.8 ilustra la misma estructura como un árbol con raíz. La raíz se llama *Desktop (Escritorio)*. Abajo de *Desktop* están *My Computer, Internet Explorer,* y otros (*My Computer* es lo único desplegado en la figura 9.1.7). Abajo de *My Computer* están $3\frac{1}{2}$ *Floppy* (*A:*), *Micron* (*C:*) y otros que no se muestran. Abajo de *Plug_ins*, que está resaltado, están los archivos *Afill32.api, Aform.js* y otros, que aparecen a la derecha de la figura 9.1.7.

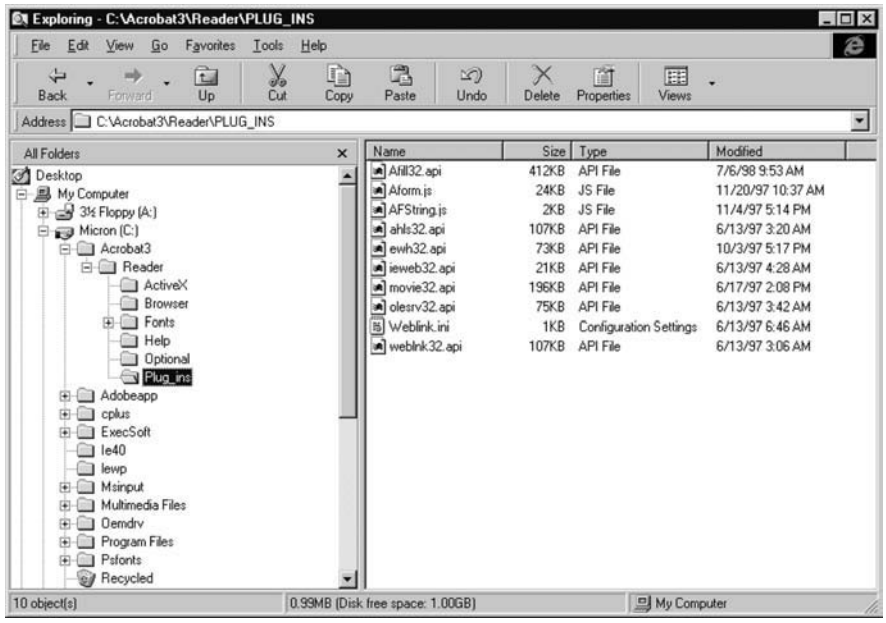


Figura 9.1.7 Despliegue del explorador de Windows de las carpetas y archivos en una computadora en particular. Las carpetas aparecen a la izquierda, y los archivos a la derecha. Bajo *Plig_ins,* que está resaltado, están los archivos *Alfill32.api, Aform.js* y los demás que aparecen a la derecha.

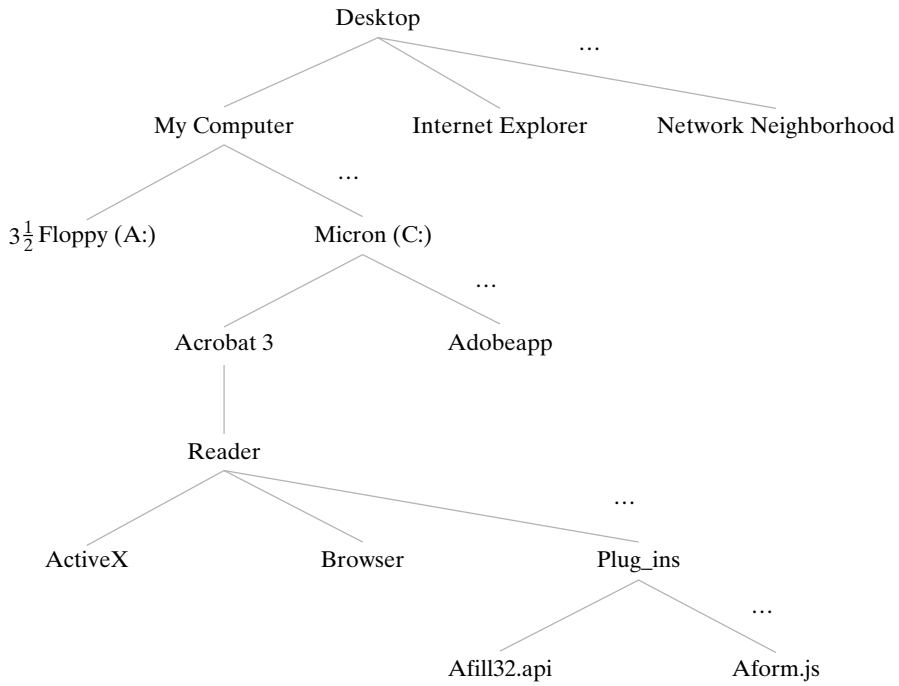


Figura 9.1.8 Estructura de la figura 9.1.7 mostrada como un árbol con raíz.

◀

Ejemplo 9.1.7 ▶

Árboles de definición jerárquica

La figura 9.1.9 es un ejemplo de un **árbol de definición jerárquica**. Estos árboles se usan para mostrar las relaciones entre los registros de una base de datos. [Recuerde que una base de datos es una colección de registros que maneja una computadora (sección 3.4)]. El árbol de la figura 9.1.9 se puede usar como modelo para establecer una base de datos para mantener los registros de los libros disponibles en varias bibliotecas.

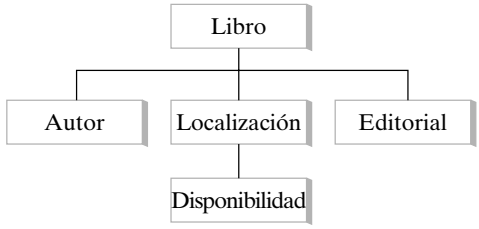


Figura 9.1.9 Árbol de definición jerárquica.

◀

Ejemplo 9.1.8 ▶

Códigos Huffman

La manera más común de representar caracteres internamente en una computadora es usando cadenas de bits de longitud fija. Por ejemplo, el código estándar para intercambio de información ASCII (siglas en inglés de *American Standard Code for Information Interchange*) representa los caracteres por una cadena de siete bits. En la tabla 9.1.1 se incluyen algunos ejemplos.

Los **códigos Huffman**, que representan caracteres por cadenas de bits de longitud variable, proporcionan alternativas al ASCII y otros códigos de longitud fija. La idea es usar cadenas de bits cortas para representar los caracteres que se usan con más frecuencia y cadenas de bits más largas para los caracteres de uso menos frecuente. De esta manera, en general es posible representar cadenas de caracteres, como texto y programas, en menos espacio que si se usara el ASCII. VCR Plus+, un dispositivo que programa automáticamente una videocasetera, usa un código Huffman para generar números que el usuario puede introducir para elegir qué programa grabar. Los números se publican en muchos listados de televisión.

WWW

TABLA 9.1.1 ■ Una parte de la tabla de ASCII.

Carácter	*Código ASCII*
A	100 0001
B	100 0010
C	100 0011
1	011 0001
2	011 0010
!	010 0001
*	010 1010

Un código Huffman se define con gran facilidad mediante un árbol con raíz (figura 9.1.10). Para decodificar una cadena de bits, comenzamos en la raíz y seguimos hacia abajo por el árbol hasta que se encuentra el carácter. El bit, 0 o 1, dice si debemos ir a la derecha o a la izquierda. Como ejemplo se decodificará la cadena

$$01010111. \qquad \textbf{(9.1.1)}$$

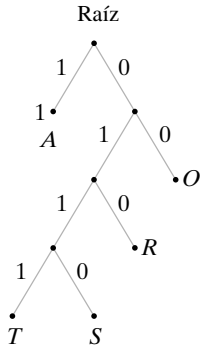


Figura 9.1.10 Código Huffman.

Se inicia en la raíz. Como el primer bit es 0, primero vamos a la derecha. Luego vamos a la izquierda y a la derecha. En este punto se encuentra el primer carácter *R*. Para decodificar el siguiente carácter, vamos de nuevo a la raíz. El siguiente bit es 1, entonces vamos a la izquierda y encontramos el siguiente carácter, *A*. Los últimos bits 0111 se decodifican como *T*. Por lo tanto, la cadena de bits (9.1.1) representa las letras *RAT*.

Dado un árbol que define un código Huffman, como el de la figura 9.1.10, cualquier cadena de bits [como (9.1.1)] se puede decodificar de manera única aun cuando los caracteres estén representados por cadenas de bits de longitud variable. Para el código Huffman definido por el árbol de la figura 9.1.10, el carácter *A* se representa por una cadena de bits de longitud 1 mientras que *S* y *T* se representan por una cadena de bits de longitud 4. (*A* se representa como 1, *S* se representa como 0110 y *T* como 0111). ◀

Huffman desarrolló un algoritmo (algoritmo 9.1.9) para construir un código Huffman a partir de la tabla que da la frecuencia de ocurrencia de los caracteres que se van a representar para que el código construido represente las cadenas de caracteres en el mínimo espacio, siempre que las cadenas representadas tengan frecuencias de caracteres idénticas a las frecuencias de la tabla. Un prueba de que el código construido es óptimo se encuentra en [Johnsonbaugh].

Algoritmo 9.1.9

Construcción de un código Huffman óptimo

Este algoritmo construye un código de Huffman óptimo a partir de una tabla con las frecuencias de ocurrencia de los caracteres que se van a representar. La salida es un árbol con raíz donde los vértices del nivel inferior se etiquetan con las frecuencias y las aristas se etiquetan con bits como en la figura 9.1.10. El árbol de codificación se obtiene sustituyendo cada frecuencia por un carácter que tiene esa frecuencia.

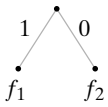


Figura 9.1.11 Caso $n = 2$ para el algoritmo 9.1.9.

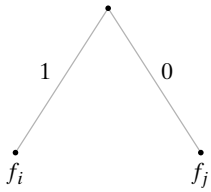


Figura 9.1.12 Caso $n > 2$ para el algoritmo 9.1.9.

Entrada: Sucesión de n frecuencias, $n \geq 2$
Salida: Árbol con raíz que define un código Huffman óptimo

```
huffman(f, n) {
    if(n == 2) {
        sean f1 y f2 las frecuencias
        sea T como en la figura 9.1.11
        return T
    }
    sean fi y fj las frecuencias más pequeñas
    se sustituyen fi y fj en la lista f por fi + fj
    T' = huffman(f, n − 1)
    se sustituye un vértice en T' etiquetado fi + fj por el
        árbol de la figura 9.1.12 para obtener el árbol T
    return T
}
```

Ejemplo 9.1.10 ▶

Se muestra cómo el algoritmo 9.1.9 construye un código Huffman óptimo usando la tabla 9.1.2.

TABLA 9.1.2 ■ Datos para el ejemplo 9.1.10.

Carácter	*Frecuencia*
!	2
@	3
#	7
$	8
%	12

El algoritmo comienza remplazando repetidas veces las dos frecuencias más pequeñas con la suma hasta que se obtiene una sucesión de dos elementos.

$$2, 3, 7, 8, 12 \rightarrow 2 + 3, 7, 8, 12$$
$$5, 7, 8, 12 \rightarrow 5 + 7, 8, 12$$
$$8, 12, 12 \rightarrow 8 + 12, 12$$
$$12, 20$$

Después, el algoritmo construye árboles trabajando hacia atrás, comenzando con la sucesión de dos elementos 12, 20 como se indica en la figura 9.1.13. Por ejemplo, el segundo árbol se obtiene del primero remplazando el vértice con etiqueta 20 por el árbol de la figura 9.1.14 ya que 20 surgió de la suma de 8 y 12. Por último, para obtener el árbol del código Huffman óptimo se sustituye cada frecuencia por un carácter que tenga la misma frecuencia (vea la figura 9.1.15).

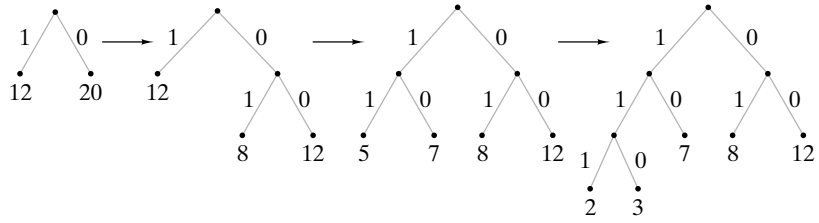


Figura 9.1.13 Construcción de código Huffman óptimo.

Observe que el árbol de Huffman para la tabla 9.1.2 no es único. Cuando 12 se sustituye por 5, 7, hay otra opción porque se tienen dos vértices con etiqueta 12. En la figura 9.1.13, se eligió uno de esos vértices de manera arbitraria. Si se elige el otro con etiqueta 12, se obtiene el árbol de la figura 9.1.16. Cualquiera de los dos árboles de Huffman da un código óptimo; es decir, cualquiera de los dos codificará texto con las frecuencias de la tabla 9.1.2 exactamente en el mismo espacio (óptimo).

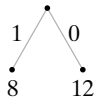


Figura 9.1.14 Árbol que sustituye el vértice con etiqueta 20 en la figura 9.1.13.

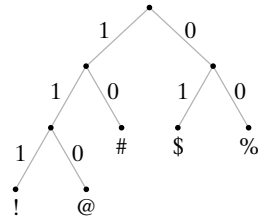


Figura 9.1.15 Árbol final de la figura 9.1.13 con cada frecuencia remplazada por un carácter que tiene esa frecuencia.

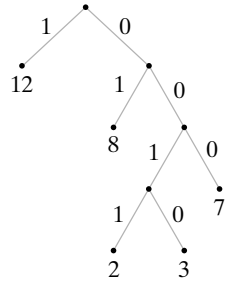


Figura 9.1.16 Otro árbol de Huffman óptimo para el ejemplo 9.1.10.

◀

Sección de ejercicios de repaso

1. Defina *árbol libre.*

2. Defina *árbol con raíz.*

3. ¿Cuál es el nivel de un vértice en un árbol con raíz?

4. ¿Cual es la altura de un vértice en un árbol con raíz?

5. Dé un ejemplo de árbol de definición jerárquica.

6. Explique cómo se organizan las carpetas y archivos en un sistema de cómputo en una estructura de árbol con raíz.

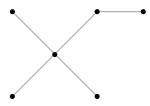
7. ¿Qué es un código Huffman?

8. Explique cómo se construye un código Huffman óptimo.

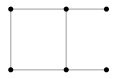
Ejercicios

¿Cuáles gráficas en los ejercicios 1 al 4 son árboles? Explique.

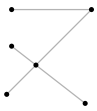
1.



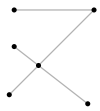
2.



3.



4.

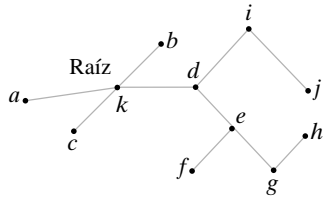


5. ¿Para qué valores de m y n la gráfica bipartita completa sobre m y n vértices es un árbol?

6. ¿Para qué valores de n la gráfica completa sobre n vértices es un árbol?

7. ¿Para qué valores de n es el cubo-n un árbol?

8. Encuentre el nivel de cada vértice en el árbol que sigue.



9. Encuentre la altura del árbol del ejercicio 8.

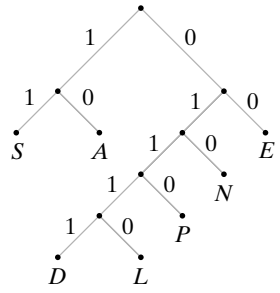
10. Dibuje el árbol T de la figura 9.1.5 como un árbol con raíz en a. ¿Cuál es la altura del árbol obtenido?

11. Dibuje el árbol T de la figura 9.1.15 como un árbol con raíz en b. ¿Cuál es la altura del árbol obtenido?

12. Dé un ejemplo similar al ejemplo 9.1.5 de un árbol que se usa para especificar relaciones de jerarquía.

13. Dé un ejemplo diferente al ejemplo 9.1.5 de un árbol de definición jerárquica.

Decodifique cada cadena de bits que usa el código Huffman dado.



14. 011000010

15. 01110100110

16. 01111001001110

17. 1110011101001111

Codifique cada palabra usando el código Huffman anterior.

18. DEN

19. NEED

20. LEADEN

21. PENNED

22. ¿Qué factores además de la cantidad de memoria usada deben considerarse cuando se elige un código, como ASCII o Huffman, para representar caracteres en una computadora?

23. ¿Qué técnicas además del uso de códigos Huffman se puede usar para ahorrar memoria cuando se almacena texto?

24. Construya un código Huffman óptimo para el conjunto de letras en la tabla.

Letra	*Frecuencia*	*Letra*	*Frecuencia*
α	5	δ	11
β	6	ε	20
γ	6		

25. Construya un código de Huffman óptimo para el conjunto de letras en la tabla

Letra	*Frecuencia*	*Letra*	*Frecuencia*
I	7.5	C	5.0
U	20.0	H	10.0
B	2.5	M	2.5
S	27.5	P	25.0

26. Use el código desarrollado en el ejercicio 25 para codificar las siguientes palabras (que tienen frecuencias consistentes con la tabla del ejercicio 25):

BUS, CUPS, MUSH, PUSS, SIP, PUSH,
CUSS, HIP, PUP, PUPS, HIPS.

27. Construya dos árboles de codificación de Huffman óptimos para la tabla del ejercicio 24, de diferentes alturas.

28. Construya un código Huffman óptimo para el conjunto de letras en la tabla.

Letra	*Frecuencia*	*Letra*	*Frecuencia*
a	2	*d*	8
b	3	*e*	13
c	5	*f*	21

29. El profesor Gig A. Byte necesita almacenar texto formado con los caracteres A, B, C, D, E, que ocurren con las siguientes frecuencias:

Letra	*Frecuencia*	*Letra*	*Frecuencia*
A	6	D	2
B	2	E	8
C	3		

El profesor Byte sugiere usar los códigos de longitud variable

Carácter	*Código*
A	1
B	00
C	01
D	10
E	0

los cuales, asegura, almacenan texto en menos espacio que el usado por un código Huffman óptimo. ¿Está en lo correcto el profesor? Explique su respuesta.

30. Demuestre que cualquier árbol con dos vértices o más tiene un vértice de grado 1.

31. Demuestre que un árbol es una gráfica plana.

32. Demuestre que un árbol es una gráfica bipartita.

33. Demuestre que los vértices de un árbol se pueden colorear con dos colores de manera que cada arista incida en vértices de diferentes colores.

La excentricidad de un vértice v en un árbol T es la longitud máxima de una trayectoria simple que comienza en v.

34. Encuentre la excentricidad de cada vértice en el árbol de la figura 9.1.5.

Un vértice v en un árbol T es un centro para T si la excentricidad de v es mínima.

35. Encuentre el centro o centros del árbol de la figura 9.1.5.

★36. Demuestre que un árbol tiene uno o dos centros.

★37. Demuestre que si un árbol tiene dos centros, éstos son adyacentes.

38. Defina el radio r de un árbol usando los conceptos de excentricidad y centro. El diámetro d de una gráfica se definió en el ejercicio 70, sección 8.2. ¿Es cierto siempre, de acuerdo con su definición de radio, que $2r = d$? Explique su respuesta.

39. Dé un ejemplo de un árbol T que no satisface la siguiente propiedad: Si v y w son vértices en T, existe una trayectoria única de v a w.

9.2 ➜ Terminología y caracterización de árboles

Una porción del árbol genealógico de los dioses griegos de la antigüedad se reproduce en la figura 9.2.1. (No se incluyeron todos los hijos). Como se aprecia, un árbol genealógico

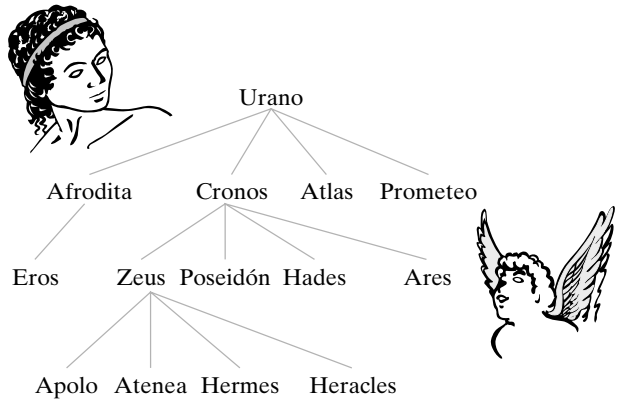


Figura 9.2.1 Una porción del árbol genealógico de los antiguos dioses griegos.

tiene la apariencia de un árbol con raíz. Los vértices adyacentes a un vértice v y en el siguiente nivel hacia abajo son los hijos de v. Por ejemplo, los hijos Cronos son Zeus, Poseidón Hades y Ares. La terminología adaptada del árbol genealógico se usa por rutina para cualquier árbol con raíz. La definición formal es la siguiente.

Definición 9.2.1 ▶

Sea T un árbol con raíz v_0. Suponga que x, y y z son vértices en T y que $(v_0, v_1, \ldots, v_n)$ es una trayectoria simple en T. Entonces

a) v_{n-1} es el *padre* de v_n.

b) $v_0, \ldots, v_{n-1}$ son *ancestros* de v_n.

c) v_n es un *hijo* de v_{n-1}.

d) Si x es un ancestro de y, y es un *descendiente* de x.

e) Si x y y son hijos de z, x y y son *hermanos*.

f) Si x no tiene hijos, x es un *vértice terminal* (o una *hoja*)

g) Si x no es un vértice terminal, x es un *vértice interno* (o una *rama*).

h) El *subárbol de T con raíz en x* es la gráfica con el conjunto de vértices V y el conjunto de aristas E, donde V es x junto con los descendientes de x y

$$E = \{e \mid e \text{ es una arista en una trayectoria simple de } x \text{ a algún vértice en } V\}. \quad \blacktriangleleft$$

Ejemplo 9.2.2 ▶

En el árbol con raíz de la figura 9.2.1,

a) El padre de Eros es Afrodita.

b) Los ancestros de Hermes son Zeus, Cronos y Urano.

c) Los hijos de Zeus son Apolo, Atenea, Hermes y Heracles.

d) Los descendientes de Cronos son Zeus, Poseidón, Hades, Ares, Apolo, Atenea, Hermes y Heracles.

e) Afrodita y Prometeo son hermanos.

f) Los vértices terminales son Eros, Apolo, Atenea, Hermes, Heracles, Poseidón, Hades, Ares, Atlas y Prometeo.

g) Los vértices internos son Urano, Afrodita, Cronos y Zeus.

h) El subárbol con raíz en Cronos se ilustra en la figura 9.2.2.

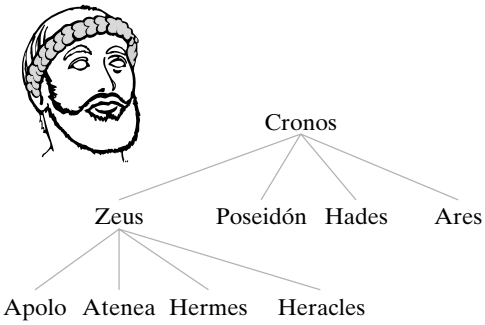


Figura 9.2.2 Subárbol con raíz en Cronos del árbol de la figura 9.2.1.

◀

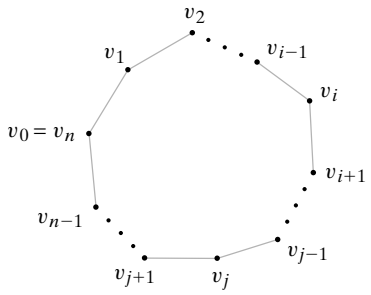


Figura 9.2.3 Ciclo simple.

El resto de esta sección se dedica a proporcionar caracterizaciones alternativas de los árboles. Sea T un árbol. Se ve que T es conexa ya que existe una trayectoria simple de cualquier vértice a cualquier otro. Más aún, es posible demostrar que T no contiene ciclos. Para esto, suponga que T contiene un ciclo C'. Por el Teorema 8.2.24, T contiene un ciclo simple (vea la figura 9.2.3)

$$C = (v_0, \ldots, v_n),$$

$v_0 = v_n$. Como T es una gráfica simple, C no puede ser un lazo; entonces C contiene al menos dos vértices distintos v_i y v_j, $i < j$. Ahora

$$(v_i, v_{i+1}, \ldots, v_j), \qquad (v_i, v_{i-1}, \ldots, v_0, v_{n-1}, \ldots, v_j)$$

son trayectorias simples distintas de v_i a v_j, lo cual contradice la definición de árbol. Por lo tanto, un árbol no puede contener un ciclo.

Una gráfica sin ciclos se llama **gráfica acíclica**. Se acaba de demostrar que un árbol es una gráfica conexa y acíclica. El inverso también es cierto; toda gráfica conexa y acíclica es un árbol. El siguiente teorema da ésta y otras caracterizaciones de los árboles.

Teorema 9.2.3

WWW

Sea T una gráfica con n vértices. Las siguientes son equivalentes.

a) T es un árbol.

b) T es conexa y acíclica.

c) T es conexa y tiene $n - 1$ aristas.

d) T es acíclica y tiene $n - 1$ aristas.

Demostración Para demostrar que *a)–d)* son equivalentes, se probarán cuatro resultados: si *a)*, entonces *b)*; si *b)*, entonces *c)*; si *c)*, entonces *d)*; si *d)*, entonces *a)*.

[Si *a)*, entonces *b)*]. La prueba de este resultado se dio antes de enunciar el teorema.

[Si *b)*, entonces *c)*]. Suponga que T es conexa y acíclica. Se probará que T tiene $n - 1$ por inducción sobre n.

Si $n = 1$, T consiste en un vértice y cero aristas, de manera que el resultado es verdadero para $n = 1$.

Ahora suponga que el resultado se cumple para una gráfica conexa acíclica con n vértices. Sea T una gráfica acíclica conexa con $n + 1$ vértices. Se selecciona una trayectoria P sin aristas repetidas de longitud máxima. Como T es acíclica, P no contiene ciclos. Por lo tanto, P tiene un vértice v de grado 1 (vea la figura 9.2.4). Se hace T^* igual que T pero eliminando v y la arista incidente en v. Entonces T^* es conexa y acíclica, y como T^* tiene n vértices, por la hipótesis inductiva T^* contiene $n - 1$ aristas. Por lo tanto, T contiene n aristas. El argumento inductivo queda completo al igual que esta parte de la demostración.

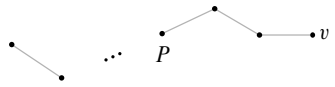


Figura 9.2.4 La demostración del Teorema 9.2.3 [si *b*), entonces *c*)]. P es una trayectoria simple.v y la arista incidente en v se eliminan de manera que pueda invocarse la hipótesis inductiva.

[Si *c)*, entonces *d)*]. Suponga que T es conexa y tiene $n - 1$ aristas. Debe demostrarse que T es acíclica.

Suponga que T contiene al menos un ciclo. Como eliminar una arista de un ciclo no desconecta la gráfica, se pueden eliminar aristas pero no vértices, del ciclo (o ciclos) en T hasta que la gráfica obtenida T^* sea conexa y acíclica. Ahora T^* es una gráfica conexa acíclica con n vértices. Se puede usar el resultado que se acaba de demostrar, *b)* implica *c)*, para concluir que T^* tiene $n - 1$ aristas. Pero T tiene más de $n - 1$ aristas. Esto es una contradicción. Por lo tanto, T es acíclica. Esta parte de la demostración queda completa.

[Si *d)*, entonces *a)*]. Suponga que T es acíclica y tiene $n - 1$ aristas. Debe demostrarse que T es un árbol, es decir, que T es una gráfica simple y que tiene una trayectoria simple única de cualquier vértice a cualquier otro.

La gráfica T no puede contener lazos porque los lazos son ciclos y T es acíclica. De manera similar, T no puede contener aristas distintas e_1 y e_2 que incidan en v y w porque se tendría el ciclo (v, e_1, w, e_2, v). Por lo tanto, T es una gráfica simple.

Suponga, a manera de contradicción, que T no es conexa (vea la figura 9.2.5). Sean

$$T_1, T_2, \ldots, T_k$$

las componentes de T. Como T no es conexa, $k > 1$. Suponga que T_i tiene n_i vértices. Cada T_i es conexa y acíclica, y podemos usar el resultado anterior, *b)* implica *c)*, para concluir que T_i tiene $n_i - 1$ aristas. Ahora bien

$$\begin{aligned} n - 1 &= (n_1 - 1) + (n_2 - 1) + \cdots + (n_k - 1) && \text{(conteo de aristas)} \\ &< (n_1 + n_2 + \cdots + n_k) - 1 && \text{(ya que } k > 1\text{)} \\ &= n - 1, && \text{(conteo de aristas)} \end{aligned}$$

lo cual es imposible. Por lo tanto, T es conexa.

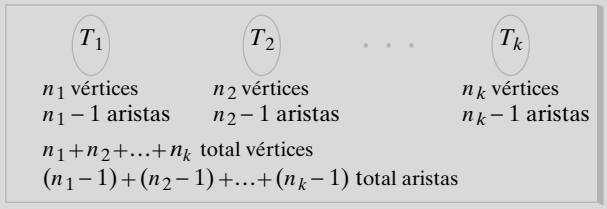


Figura 9.2.5 Demostración del Teorema 9.2.3 [si *d)* entonces *a)*]. Las T_i son componentes de T. T_i tiene n_i vértices y $n_i - 1$ aristas. Se obtiene una contradicción porque el número total de aristas debe ser igual que $n - 1$.

Suponga que existen trayectorias simples distintas P_1 y P_2 de a a b en T (vea la figura 9.2.6). Sea c el primer vértice después de a en P_1 que no está en P_2; sea d el vértice anterior a c en P_1; y sea e el primer vértice después de d en P_1 que también está en P_2. Sea

$$(v_0, v_1, \ldots, v_{n\ 1}, v_n)$$

la porción de P_1 de $d = v_0$ a $e = v_n$. Sea

$$(w_0, w_1, \ldots, w_{m\ 1}, w_m)$$

la porción de P_2 de $d = w_0$ a $e = w_n$. Ahora

$$(v_0, \ldots, v_n = w_m, w_{m-1}, \ldots, w_1, w_0) \qquad \textbf{(9.2.1)}$$

es un ciclo en T, lo cual es una contradicción. [De hecho, (9.2.1) es un ciclo simple ya que no se repiten vértices excepto v_0 y w_0]. Entonces, hay una trayectoria simple de cualquier vértice a cualquier otro en T. Por lo tanto, T es un árbol. Esto completa la prueba.

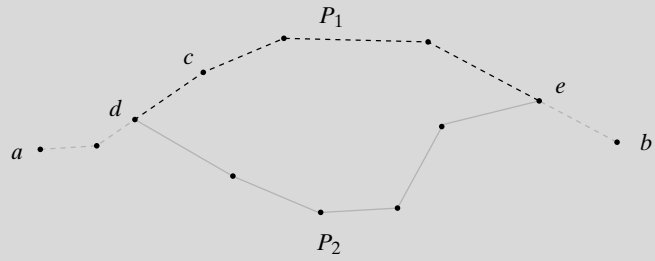


Figura 9.2.6 Prueba del Teorema 9.2.3 [si d), entonces a)]. P_1 (punteada) y P_2 (continua) son trayectorias simples diferentes de a a b; c es el primer vértice después de a en P_1 que no está en P_2; d es el vértice anterior a c en P_1; e es el primer vértice después de d en P_1 que también está en P_2. Como se aprecia, se obtiene un ciclo, que da la contradicción.

Sugerencias para resolver problemas

Esta sección introduce cierta terminología útil y proporciona varias caracterizaciones diferentes de los árboles. Si T es una gráfica con n vértices, las siguientes son equivalentes (Teorema 9.2.3):

a) T es un árbol

b) Si v y w son vértices en T, existe una trayectoria simple única de v a w (definición de árbol).

c) T es conexa y acíclica.

d) T es conexa y tiene $n - 1$ aristas.

e) T es acíclica y tiene $n - 1$ aristas.

Se pueden usar las caracterizaciones anteriores de varias maneras. Por ejemplo, una gráfica con cuatro aristas y seis vértices no puede ser un árbol porque viola los incisos *d)* y *e)*. Más aún, la gráfica es no conexa o contiene un ciclo. (Si es tanto conexa como acíclica, se-

ría un árbol y entonces tendría cinco aristas). Una gráfica conexa con n vértices y más aristas que $n - 1$ debe contener un ciclo. (Si fuera acíclica, sería un árbol y por lo mismo tendría $n - 1$ aristas).

Sección de ejercicios de repaso

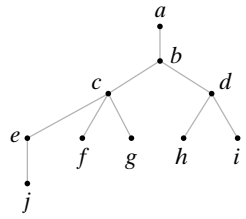
1. Defina *padre* en un árbol con raíz.
2. Defina *descendiente* en un árbol con raíz.
3. Defina *hermano* en un árbol con raíz.
4. Defina *vértice terminal* en un árbol con raíz.
5. Defina *vértice interno* en un árbol con raíz.
6. Defina *gráfica acíclica*.
7. Dé una caracterización alternativa de árboles.

Ejercicios

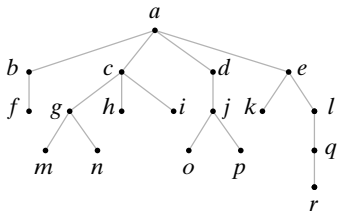
Responda las preguntas en los ejercicios 1 al 6 para el árbol en la figura 9.2.1.

1. Encuentre el padre de Poseidón.
2. Encuentre los ancestros de Eros.
3. Encuentre los hijos de Urano.
4. Encuentre los descendientes de Zeus.
5. Encuentre los hermanos de Ares.
6. Dibuje el subárbol con raíz en Afrodita.

Responda las preguntas en los ejercicios 7 al 15 para el siguiente árbol.



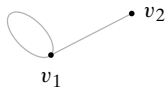
7. Encuentre los padres de c y h.
8. Encuentre los ancestros de c y j.
9. Encuentre los hijos de d y e.
10. Encuentre los descendientes de c y e.
11. Encuentre los hermanos de f y h.
12. Encuentre los vértices terminales.
13. Encuentre los vértices internos.
14. Dibuje el subárbol con raíz en j.
15. Dibuje el subárbol con raíz en e.
16. Responda a las preguntas en los ejercicios 7 al 15 para el siguiente árbol.



17. ¿Qué puede decir de dos vértices en un árbol con raíz que tienen el mismo padre?
18. ¿Qué puede decir de dos vértices en un árbol con raíz que tienen los mismos ancestros?
19. ¿Qué puede decir de un vértice en un árbol que no tiene ancestros?
20. ¿Qué pude decir de dos vértices en un árbol con raíz que tienen un descendiente en común?
21. ¿Qué puede decir de un vértice en un árbol con raíz que no tiene descendientes?

En los ejercicios 22 al 26, dibuje una gráfica que tenga las propiedades indicadas o explique por qué no existe tal gráfica.

22. Seis aristas; ocho vértices
23. Acíclica; cuatro aristas, seis vértices
24. Árbol; todos los vértices de grado 2
25. Árbol; seis vértices que tienen 1, 1, 1, 1, 3, 3 grados
26. Árbol; cuatro vértices internos, seis vértices terminales
27. Explique por qué, si se permiten ciclos de longitud 0, una gráfica que consiste en un solo vértice y ninguna arista no es acíclica.
28. Explique por qué, si se permite que los ciclos repitan aristas, una gráfica que consiste en una sola arista y dos vértices no es acíclica.
29. La gráfica conexa mostrada tiene una trayectoria simple única de cualquier vértice a cualquier otro, pero no es un árbol. Explique por qué.



Un bosque es una gráfica simple sin ciclos.

30. Explique por qué un bosque es una unión de árboles.
31. Si un bosque F *consiste en* m árboles y tiene n vértices, ¿cuántas aristas tiene F?
32. Si $P_1 = (v_0, \ldots, v_n)$ y $P_2 = (w_0, \ldots, w_m)$ son dos trayectorias simples de a a b diferentes, en una gráfica simple G, ¿es

$$(v_0, \ldots, v_n = w_m, w_{m-1}, \ldots, w_1, w_0)$$

necesariamente un ciclo? Explique su respuesta. (Este ejercicio es relevante para el último párrafo de la demostración del Teorema 9.2.3).

33. Demuestre que una gráfica G con n vértices y menos de $n - 1$ aristas no es conexa.

★34. Pruebe que T es un árbol si y sólo si T es conexa y cuando se agrega una arista entre cualesquiera dos vértices, se crea exactamente un ciclo.

35. Demuestre que si G es un árbol, todo vértice de grado 2 o más es un punto de articulación. ("Punto de articulación" se definió en el ejercicio 55, de la sección 8.2).
36. Dé un ejemplo para demostrar que el inverso del ejercicio 35 es falso, incluso si se supone que G es conexa.

Rincón de solución de problemas

Árboles

Problema

Sea T una gráfica simple. Pruebe que T es un árbol si y sólo si T es conexa pero al eliminar de ella cualquier arista (no vértices) T se desconecta.

Cómo atacar el problema

Debe aclararse qué se quiere probar. Como el enunciado es "si y sólo si", debemos probar dos afirmaciones:

Si T es árbol, entonces T es conexa pero al eliminar de ella cualquier arista (no vértices) T se desconecta. **(1)**

Si T es conexa pero al eliminar de ella cualquier arista (no vértices) T se desconecta, entonces T es un árbol. **(2)**

En (1), a partir de la suposición de que T es un árbol, debe deducirse que T es conexa, pero al eliminar cualquier artista (no vértice) T se desconecta. En (2), a partir de la suposición de que T es conexa pero al eliminar cualquier arista (no vértice) T se desconecta, debemos deducir que T es un árbol.

Al desarrollar una demostración, suele ser útil revisar las definiciones y otros resultados relacionados con la afirmación que se quiere probar. Aquí tienen relevancia directa la definición de un árbol y el Teorema 9.2.3, que da condiciones equivalentes para que una gráfica sea un árbol.

La definición 9.1.1 establece:

Un árbol T es una gráfica simple que satisface lo siguiente:

Si v y w son vértices en T, existe una trayectoria simple única de v a w. **(3)**

El Teorema 9.2.3 establece que las siguientes son equivalentes para una gráfica T con n vértices:

T es un árbol. **(4)**

T es conexa y acíclica **(5)**

T es conexa y tiene $n - 1$ aristas **(6)**

T es acíclica y tiene $n - 1$ aristas **(7)**

Cómo encontrar una solución

Primero se intentará probar (1). Se supone que T es un árbol. Debemos probar dos cosas: T es conexa, y al eliminar de T cualquier arista (no vértice) T se desconecta.

Las afirmaciones (5) y (6) dicen de inmediato que T es conexa. Ninguna de las afirmaciones (3) a la (7) dicen nada de la eliminación de aristas o de que una gráfica no esté conectada. Sin embargo, si se da un argumento por contradicción y se supone que eliminar de T algunas aristas (no vértices) no desconecta a T, entonces cuando se elimina una arista de T, la gráfica T' obtenida es conexa. En este caso, para la gráfica T', (5) es verdadera, pero (6) y (7) son falsas, lo cual es una contradicción ya que (5), (6) y (7) son todas verdaderas (y la gráfica es un árbol), o (5), (6) y (7) son todas falsas (y la gráfica no es un árbol).

Ahora considere probar (2). Se supone que T es conexa y que al eliminar de ella cualquier arista (no vértice) T se desconecta. Debe demostrarse que T es un árbol. Se intentará demostrar que T es conexa y acíclica. Después se puede recurrir a (5) para concluir que T es un árbol.

Como T es conexa, todo lo que debe hacerse es demostrar que T es acíclica. De nuevo, se adopta un enfoque por contradicción. Suponga que T tiene un ciclo. Sin perder de vista lo que se supone (la eliminación de T de cualquier arista desconecta a T), intente descubrir cómo deducir una contradicción a partir de que T tiene un ciclo, antes de seguir leyendo.

Si se quita una arista del ciclo de T, esta última seguirá conectada. Tal contradicción demuestra que T es acíclica. Por (5), T es un árbol.

Solución formal

Suponga que T no tiene vértices.

Suponga que T es un árbol. Entonces por el Teorema 9.2.3, T es conexa y tiene $n - 1$ aristas. Suponga que se puede eliminar una arista de T para obtener T' de manera que T' es conexa. Como T no contiene ciclos, T' tampoco contiene ciclos. Por el Teorema 9.2.3, T' es un árbol. De nuevo por el Teorema 9.2.3, T' tiene $n - 1$ aristas. Esto es una contradicción. Por lo tanto, T es conexa, pero al eliminar de T cualquier arista (no vértices) T se desconecta.

Si T es conexa y al eliminar de T una arista (no vértices) T se desconecta, entonces T no contiene ciclos. Por el Teorema 9.2.3, T es un árbol.

Resumen de las técnicas de solución de problemas

- Al tratar de desarrollar una demostración, escriba con cuidado lo que se supone y lo que se quiere probar.
- Al intentar desarrollar una demostración, considere el uso de definiciones y teoremas relacionados.
- Cuando intente desarrollar una demostración, revise las demostraciones de teoremas relacionados o similares.
- Si no se pueden aplicar las condiciones de las definiciones y teoremas potencialmente útiles, intente una prueba por contradicción. Cuando suponga la negación de la hipótesis, dispondrá de afirmaciones adicionales que pueden hacer que se apliquen algunas de las condiciones de las definiciones y teoremas.

9.3 ➜ Árboles de expansión

En esta sección se considera el problema de encontrar una subgráfica T de una gráfica G tal que T es un árbol que contiene todos los vértices de G. Este árbol se conoce como **árbol de expansión**. Se verá que los métodos para encontrar los árboles de expansión también son aplicables y se aplican a otros problemas.

WWW

Definición 9.3.1 ▶ Un árbol T es un *árbol de expansión* de una gráfica G si T es una subgráfica de G que contiene todos los vértices de G. ◀

Ejemplo 9.3.2 ▶ Un árbol de expansión de una gráfica G de la figura 9.3.1 se muestra con línea continua. ◀

Ejemplo 9.3.3 ▶ En general, una gráfica tiene varios árboles de expansión. Otro árbol de expansión de la gráfica G de la figura 9.3.1 se aprecia en la figura 9.3.2.

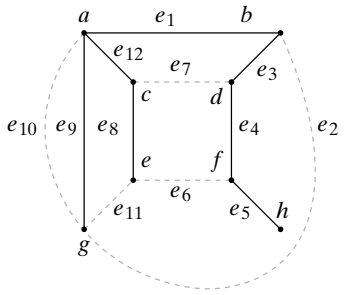


Figura 9.3.1 Una gráfica y un árbol de expansión mostrado por la línea continua.

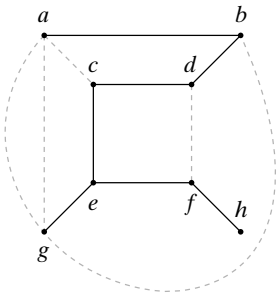


Figura 9.3.2 Otro árbol de expansión (línea continua) de la gráfica de la figura 9.3.1.

◀

Suponga que una gráfica G tiene un árbol de expansión T. Sean a y b vértices de G. Como a y b también son vértices de T y T es un árbol, existe una trayectoria P de a a b. Sin embargo, P también sirve como trayectoria de a a b en G; entonces G es conexa. El inverso también es cierto.

Teorema 9.3.4 *Una gráfica G tiene un árbol de expansión si y sólo si G es conexa.*

Demostración Ya se demostró que si G tiene un árbol de expansión, entonces G es conexa. Suponga que G es conexa. Si G es acíclica, por el Teorema 9.2.3, G es un árbol.

Suponga que G contiene un ciclo. Se elimina una arista (no vértices) de este ciclo. La gráfica producida todavía es conexa. Si es acíclica, nos detenemos. Si contiene un ciclo, se elimina una arista de este ciclo, y así sucesivamente, hasta producir una subgráfica T conexa y acíclica. Por el Teorema 9.2.3, T es un árbol. Como T contiene todos los vértices de G, T es un árbol de expansión de G.

Un algoritmo para encontrar un árbol de expansión basado en la demostración del Teorema 9.3.4 no sería muy eficiente; incluiría el tardado proceso de encontrar ciclos. Es posible hacer algo mejor. Se ilustrará el primer algoritmo para encontrar un árbol de expansión mediante un ejemplo y luego se establecerá el algoritmo.

Ejemplo 9.3.5 ▶ Encuentre el árbol de expansión para la gráfica G de la figura 9.3.1.

Se usará un método llamado **búsqueda a lo ancho** (algoritmo 9.3.6). La idea de la búsqueda a lo ancho es procesar todos los vértices en un nivel dado antes de moverse al nivel más alto que sigue.

WWW

Primero se selecciona un orden, por ejemplo *abcdefgh*, de los vértices de G. Se elige el primer vértice a y se etiqueta como raíz. T consiste en un sólo vértice a y ninguna

arista. Se agregan a T todas las aristas (a, x) y vértices en los cuales inciden, para $x = b$ hasta h, que no produzcan ciclos cuando se agregan a T. Entonces se agregan a T las aristas (a, b), (a, c) y (a, g). (Se puede usar cualquiera de las aristas paralelas que inciden en a y g). Se repite este proceso con los vértices del nivel 1 examinando cada uno en orden:

b: incluir (b, d).
c: incluir (c, e).
g: ninguna

Se repite el proceso con los vértices en el nivel 2:

d: incluir (d, f).
e: ninguna

Se repite el proceso con los vértices del nivel 3:

f: incluir (f, h).

Como no se pueden agregar más aristas al vértice h en el nivel 4, el proceso termina. Se ha encontrado el árbol de expansión de la figura 9.3.1. ◀

El método del ejemplo 9.3.5 se formaliza como el algoritmo 9.3.6.

Algoritmo 9.3.6

Búsqueda a lo ancho de un árbol de expansión

Este algoritmo encuentra un árbol de expansión usando el método de búsqueda a lo ancho.

Entrada: Gráfica conexa G con vértices ordenados

$$v_1, \quad v_2, \quad \ldots, \quad v_n$$

Salida: Árbol de expansión T.

```
bla(V, E) {
    //V = vértices ordenados v1, . . . , vn; E = aristas
    //V' = vértices del árbol de expansión T; E' = aristas del árbol de expansión T
    //v1 es la raíz del árbol de expansión
    //S es una lista ordenada
    S = (v1)
    V' = {v1}
    E' = ∅
    while(verdadero) {
        for x ∈ S, en orden
            for y ∈ V − V', en orden
                if((x, y) es una arista)
                    agregar arista (x, y) a E' y y a V'
        if (no se agregaron aristas)
            return T
        S = hijos de S ordenados siguiendo el orden original
    }
}
```

El ejercicio 16 pide un argumento para demostrar que el algoritmo 9.3.6 encuentra el árbol de expansión correctamente.

La búsqueda a lo ancho se puede usar para probar si una gráfica arbitraria G con n vértices es conexa (vea el ejercicio 26). Se usa el método del algoritmo 9.3.6 para producir un árbol T. Después G es conexa si y sólo si T tiene n vértices.

La búsqueda a lo ancho también resulta útil para encontrar trayectorias de longitud mínima en una gráfica ponderada desde un vértice fijo v a todos los demás vértices (vea el ejercicio 20). Se emplea el método del algoritmo 9.3.6 para generar un árbol de expansión con raíz en v. Se observa que la longitud de una ruta más corta de v al vértice en el nivel i del árbol de expansión es i. El algoritmo de la ruta más corta de Dijkstra para gráficas ponderadas (algoritmo 8.4.1) se puede considerar como una generalización de la búsqueda a lo ancho (vea el ejercicio 21).

WWW

Una alternativa para la búsqueda a lo ancho es la **búsqueda de altura**, que procede a los niveles sucesivos en un árbol en la oportunidad más cercana posible.

Algoritmo 9.3.7

Búsqueda a profundidad de un árbol de expansión

Este algoritmo encuentra un árbol de expansión usando el método de búsqueda a profundidad.

Entrada: Gráfica conexa G con vértices ordenados

$$v_1, \quad v_2, \quad \ldots, \quad v_n$$

Salida: Árbol de expansión T

```
bll(V, E) {
   //V' = vértices del árbol de expansión T; E' = aristas del árbol de expansión T
   //v1 es la raíz del árbol de expansión
   V' = {v1}
   E' = ∅
   w = v1
   while(verdadero) {
      while(hay arista (w, v) que al agregarla a T no crea
         un ciclo en T) {
         elegir la arista (w, vk) con k mínima, que si se
            agrega a T no crea un ciclo en T
         agregar(w, vk) a E'
         agregar vk a V'
         w = vk
      }
      if (w == v1)
         return T
      w = padre de w en T //hacia atrás
   }
}
```

El ejercicio 17 pide un argumento para probar que el algoritmo 9.3.7 encuentra un árbol de expansión correctamente.

Ejemplo 9.3.8 ▶

Use la búsqueda a profundidad (algoritmo 9.3.7) para encontrar un árbol de expansión para la gráfica de la figura 9.3.2 con el orden de vértices *abcdefgh*.

Se selecciona el primer vértice a y se le llama raíz (figura 9.3.2). Después se agrega al árbol la arista (a, x), con x mínima. En este caso se agrega la arista (a, b).

Se repite este proceso. Se agregan las aristas (b, d), (d, c), (c, e), (e, f) y (f, h). En este punto, no se puede agregar una arista de la forma (h, x), de manera que regresamos a f, el padre de h, y tratamos de agregar una arista de la forma (f, x). Una vez más, no es posible agregar una arista de la forma (f, x), así que regresamos a e, el padre de f. Esta vez se puede agregar la arista (e, g). En este punto, no hay más aristas que agregar, de manera que regresamos a la raíz y el procedimiento termina. ◀

En virtud de la línea en el algoritmo 9.3.7 donde se regresa por una arista hacia la raíz elegida el inicio, la búsqueda a profundidad también se llama **de regreso**. En el siguiente ejemplo, se usa el regreso para resolver un juego.

Ejemplo 9.3.9 ▶

Problema de las cuatro reinas

El problema de las cuatro reinas trata de colocar cuatro fichas en una cuadrícula de 4×4 de manera que no haya dos fichas en la misma fila, columna o diagonal. Construya un algoritmo de regreso para resolver el problema de las cuatro reinas. (Para usar la terminología del ajedrez, éste es el problema de colocar cuatro reinas en un tablero de 4×4 de manera que ninguna reina pueda atacar a otra).

WWW

La idea del algoritmo es colocar fichas sucesivamente en las columnas. Cuando es imposible colocar una ficha en una columna, se regresa y ajusta la ficha de la columna anterior. ◀

Algoritmo 9.3.10

Solución al problema de las cuatro reinas usando el regreso

Este algoritmo usa regresos para buscar un arreglo de cuatro fichas en una cuadrícula de 4×4 de manera que no haya dos fichas en la misma fila, columna o diagonal.

```
Entrada:  Un arreglo fila de tamaño 4
Salida:   verdadero, si hay solución
          Falso, si no hay solución
          [Si hay solución, la reina k está en la
          columna k y el renglón fila(k)].
cuatro_reinas(fila) {
   k = 1 //inicia en la columna 1

      //inicia en el renglón 1
   //como fila(k) se incrementa antes de usarla, se hace fila(1) igual a 0
   fila(1) = 0
   while (k > 0) {
      fila(k) = fila(k) + 1
      //se busca un movimiento legal en la columna k
      while (fila(k) ≤ 4 ∧ columna k, fila(k) conflicto)
         //se intenta el siguiente renglón
         fila(k) = fila(k) + 1
      if (fila(k) ≤ 4)
         if(k == 4)
            return verdadero
         else{//siguiente columna
            k = k + 1
            fila(k) = 0
         }
      else //regresar a columna anterior
         k = k − 1
   }
   return falso //no hay solución
}
```

El árbol que genera el algoritmo 9.3.10 se ilustra en la figura 9.3.3. La numeración indica el orden en que se generaron los vértices. La solución se encuentra en el vértice 8.

El problema de las n reinas es colocar las n fichas en una cuadrícula de $n \times n$ de manera que no haya dos fichas en la misma fila, columna o diagonal. Es directo verificar que no hay solución a los problemas de dos o tres reinas (vea el ejercicio 10). Se acaba de ver que el algoritmo 9.3.10 genera una solución para el problema de las cuatro reinas. Se han dado muchos desarrollos para generar una solución al problema de las n reinas para toda $n \geq 4$ (vea, por ejemplo, [Johnsonbaugh]).

La búsqueda a lo largo o de regreso es atractiva, en especial en un problema como el del ejemplo 9.3.9, donde todo lo que se quiere es una solución. Como una solución, si acaso existe, se encuentra en un vértice terminal, moviéndose a los vértices terminales tan pronto como sea posible, en general no se puede evitar generar algunos vértices innecesarios.

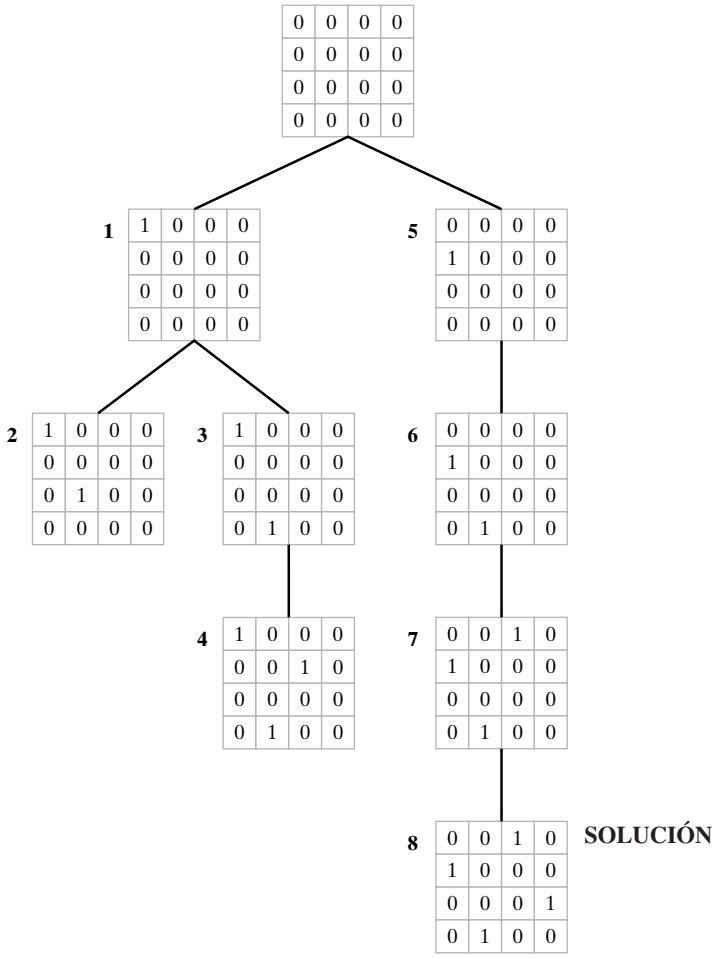


Figura 9.3.3 Árbol generado por el algoritmo de regreso (algoritmo 9.3.10) en la búsqueda de una solución al problema de las cuatro reinas.

Sugerencias para resolver problemas

La búsqueda a profundidad y la búsqueda a lo ancho son la base de muchos algoritmos de gráficas. Por ejemplo, cualquiera de los dos resulta útil para determinar si una gráfica es conexa: Si se puede visitar *cada* vértice en una gráfica a partir de un vértice inicial, la gráfica es conexa; de otra manera no es conexa (vea los ejercicios 26 y 27). La búsqueda a profundidad se puede emplear como algoritmo de búsqueda, en cuyo caso se llama de regreso. En el algoritmo 9.3.10, el regreso se usa para buscar una solución al problema de las 4 reinas; también se puede utilizar para buscar ciclos de Hamilton en una gráfica, para generar permutaciones y para determinar si dos gráficas son isomorfas (vea [Johnsonbaugh]).

Sección de ejercicios de repaso

1. ¿Qué es un árbol de expansión?

2. Establezca una condición necesaria y suficiente para que una gráfica tenga un árbol de expansión.

3. Explique cómo funciona la búsqueda a lo ancho.

4. Explique cómo funciona la búsqueda a profundidad.

5. ¿Qué significa ir de regreso?

Ejercicios

1. Use la búsqueda a lo ancho (algoritmo 9.3.6) con el orden de vértices *hgfedcba* para encontrar un árbol de expansión para la gráfica G de la figura 9.3.1.

2. Use la búsqueda a lo ancho (algoritmo 9.3.6) con el orden de vértices *hfdbgeca* para encontrar un árbol de expansión para la gráfica G de la figura 9.3.1.

3. Use la búsqueda a lo ancho (algoritmo 9.3.6) con el orden de vértices *chbgadfe* para encontrar un árbol de expansión para la gráfica G de la figura 9.3.1.

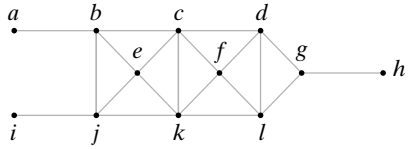
4. Use la búsqueda a profundidad (algoritmo 9.3.7) con el orden de vértices *hgfedcba* para encontrar un árbol de expansión para la gráfica G de la figura 9.3.1.

5. Use la búsqueda a profundidad (algoritmo 9.3.7) con el orden de vértices *hfdbgeca* para encontrar un árbol de expansión para la gráfica G de la figura 9.3.1.

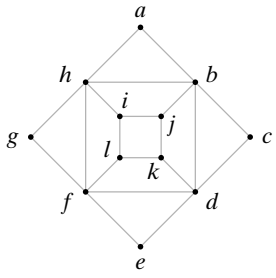
6. Use la búsqueda a profundidad (algoritmo 9.3.7) con el orden de vértices *dhcbefag* para encontrar un árbol de expansión para la gráfica G de la figura 9.3.1.

En los ejercicios 7 al 9, encuentre un árbol de expansión para cada gráfica.

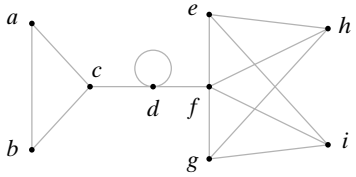
7.



8.



9.



10. Demuestre que no hay solución a los problemas de dos y tres reinas.

11. Encuentre una solución a los problemas de cinco y seis reinas.

12. ¿Falso o verdadero? Si G es una gráfica conexa y T es un árbol de expansión para G, existe un orden de los vértices de G tal que el algoritmo 9.3.6 produce T como una árbol de expansión. Si es verdadero, pruébelo; de otra manera, proporcione un contraejemplo.

13. ¿Falso o verdadero? Si G es una gráfica conexa y T es un árbol de expansión para G, existe un orden de los vértices de G tal que el algoritmo 9.3.7 produce T como un árbol de expansión. Si es verdadero, pruébelo; de otra manera, proporcione un contraejemplo.

14. Muestre, mediante un ejemplo, que el algoritmo 9.3.6 puede producir árboles de expansión idénticos para una gráfica conexa G a partir de dos ordenamientos de los vértices de G.

15. Muestre, mediante un ejemplo, que el algoritmo 9.3.7 puede producir árboles de expansión idénticos para una gráfica conexa G a partir de dos ordenamientos de los vértices de G.

16. Pruebe que el algoritmo 9.3.6 es correcto.

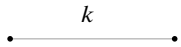
17. Pruebe que el algoritmo 9.3.7 es correcto.

18. ¿En qué condiciones una arista en una gráfica conexa G está contenida en todos los árboles de expansión de G?

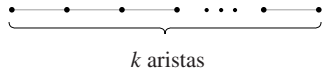
19. Sean T y T' dos árboles de expansión de una gráfica conexa G. Suponga que una arista x está en T pero no en T'. Demuestre que existe una arista y en T' pero no en T tal que $(T - \{x\}) \cup \{y\}$ y $(T' - \{y\}) \cup \{x\}$ son árboles de expansión de G.

20. Escriba un algoritmo basado en la búsqueda a lo ancho que encuentre la longitud mínima de cada trayectoria en una gráfica no ponderada de un vértice fijo v a todos los demás.

21. Sea G una gráfica ponderada en la que el peso de cada arista es un entero positivo. Sea G' la gráfica obtenida de G al sustituir cada arista



en G con peso k por k aristas no ponderadas en serie:



Demuestre que el algoritmo de Dijkstra para encontrar la longitud mínima de cada trayectoria en la gráfica ponderada G desde un vértice fijo v a todos lo demás (algoritmo 8.4.1) y realizar la búsqueda a lo ancho en una gráfica no ponderada G' comenzando en el vértice v son, de hecho, el mismo proceso.

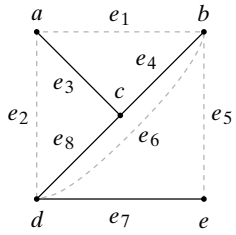
22. Sea T un árbol de expansión para una gráfica G. Demuestre que si una arista en G, pero no en T, se agrega a T, se produce un ciclo único.

Un ciclo como el descrito en el ejercicio 22 se llama ciclo fundamental. La matriz del ciclo fundamental de una gráfica G tiene renglones indexados según los ciclos fundamentales de G respecto a un árbol de expansión T para G y sus columnas indexadas según las aristas de G. El elemento ij es 1 si la arista j está en el i-ésimo ciclo fundamental y 0 de otra manera. Por ejemplo, la matriz del ciclo fundamental de la gráfica G de la figura 9.3.1 relativa al árbol de expansión mostrado en la figura 9.3.1 es

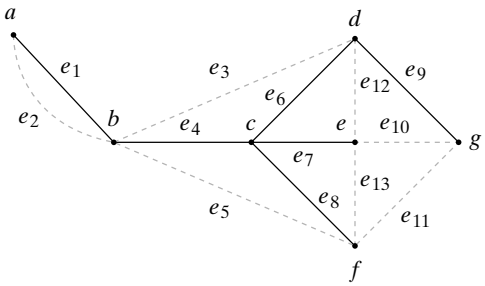
$$\begin{array}{c} \\ (abdca) \\ (efdbace) \\ (ageca) \\ (aga) \\ (abga) \end{array} \begin{array}{c} \begin{array}{cccccccccccc} e_7 & e_6 & e_{11} & e_{10} & e_2 & e_1 & e_3 & e_4 & e_5 & e_8 & e_9 & e_{12} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}.$$

Encuentre la matriz del ciclo fundamental de cada gráfica. El árbol de expansión que se usará está dibujado con línea continua.

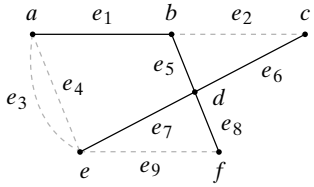
23.



24.



25.



26. Escriba un algoritmo de búsqueda a profundidad para probar si una gráfica es conexa.

27. Escriba un algoritmo de búsqueda a profundidad para probar si una gráfica es conexa.

28. Escriba un algoritmo de búsqueda a lo largo para encontrar todas las soluciones al problema de las cuatro reinas.

9.4 ➜ Árboles de expansión mínima

WWW

La gráfica ponderada G de la figura 9.4.1 muestra seis ciudades y los costos de construir carreteras entre ciertos pares de ciudades. Se desea construir el sistema de carreteras de menor costo que conecte las seis ciudades. La solución habrá de representarse en una subgráfica. Esta subgráfica debe ser un árbol de expansión ya que debe contener a todos los vértices (de manera que todas las ciudades queden comprendidas en el sistema de carreteras), debe ser conexa (para que se pueda llegar a cualquier ciudad desde cualquier otra) y debe tener una trayectoria simple única entre cada par de vértices (puesto que una gráfica que contiene trayectorias simples múltiples entre un par de vértices tal vez no represente un sistema de costo mínimo). Entonces, lo que se necesita es un árbol de expansión en el que la suma de los pesos sea mínima. Este árbol se llama **árbol de expansión mínima**.

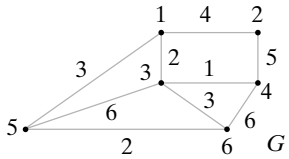


Figura 9.4.1 Seis ciudades (numeradas del 1 al 6) y los costos de construir carreteras entre ciertos pares de ellas.

Definición 9.4.1 ▶ Sea G una gráfica ponderada. Un *árbol de expansión mínima* de G es un árbol de expansión de G con peso mínimo. ◀

Ejemplo 9.4.2 ▶ El árbol T' mostrado en la figura 9.4.2 es un árbol de expansión para la gráfica G de la figura 9.4.1. El peso de T' es 20. Este árbol no es un árbol de expansión mínima, ya que el árbol de expansión T mostrado en la figura 9.4.3 tiene peso 12. Se verá más adelante que T es un árbol de expansión mínima para G.

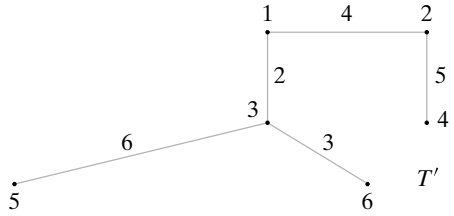


Figura 9.4.2 Árbol de expansión con peso 20 de la gráfica de la figura 9.4.1.

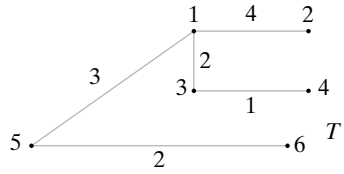


Figura 9.4.3 Árbol de expansión con peso 12 de la gráfica de la figura 9.4.1.

◀

El algoritmo para encontrar un árbol de expansión mínimo que se estudiará se conoce como **algoritmo de Prim** (algoritmo 9.4.3). Este algoritmo construye un árbol agregando aristas de manera iterativa hasta que se obtiene un árbol de expansión mínima. El algoritmo comienza con un solo vértice. Después, en cada iteración, agrega al árbol actual una arista de peso mínimo que no completa un ciclo. Otro algoritmo para encontrar el árbol de expansión mínima, conocido como **algoritmo de Kruskal**, se presenta en los ejercicios 20 al 22.

WWW

Algoritmo 9.4.3

Algoritmo de Prim

Este algoritmo encuentra un árbol de expansión mínima en una gráfica ponderada conexa.

Entrada: Gráfica ponderada conexa con vértices $1, \ldots, n$ y vértice inicial s. Si (i, j) es una arista, $w(i, j)$ es igual al peso de (i, j); si (i, j) no es una arista, $w(i, j)$ es igual a ∞ (un valor mayor que cualquier peso real).

Salida: Conjunto de aristas de E en un árbol de expansión mínima (aem)

```
    prim(w, n, s)  {
        //v(i) = 1 si el vértice i se agrega al aem
        //v(i) = 0 si el vértice i no se agrega al aem
1.      for i = 1 to n
2.          v(i) = 0
        //se agrega el vértice a; aem
3.      v(s) = 1
        //se comienza con un conjunto de aristas vacío
4.      E = ∅
        //se ponen n − 1 aristas en el aem
5.      for i = 1 to n − 1 {
            //se agrega la arista de peso mínimo con un vértice
            //en el aem y un vértice que no esté en el aem
6.          mín = ∞
7.          for j = 1 to n
8.              if (v(j) == 1)//si j es un vértice en el aem
9.                  for k = 1 to n
10.                     if(v(k) == 0 ∧ w(j, k) < mín) {
11.                         agregar_vértice = k
12.                         e = (j, k)
13.                         mín = w(j, k)
14.                     }
            //se pone el vértice y la arista en el aem
15.         v(agregar_vértice) = 1
16.         E = E ∪ {e}
17.     }
18.     return E
19. }
```

Ejemplo 9.4.4 ▶

Demuestre que el algoritmo de Prim encuentra un árbol de expansión mínima para la gráfica de la figura 9.4.1. Suponga que el vértice inicial s es 1.

En la línea 3 se agrega el vértice 1 al árbol de expansión mínima. La primera vez que se ejecuta el ciclo "for" en las líneas 7 a la 14, las aristas con un vértice en el árbol y un vértice fuera del árbol son

Arista	*Peso*
(1, 2)	4
(1, 3)	2
(1, 5)	3

Se selecciona la arista (1, 3) con peso mínimo. En las líneas 15 y 16 se agrega el vértice al árbol de expansión mínima y la arista (1, 3) a *E*.

La siguiente vez que se ejecuta el ciclo "for" en las líneas 7 a la 14, las aristas con un vértice en el árbol y otro fuera del árbol son

Arista	*Peso*
(1, 2)	4
(1, 5)	3
(3, 4)	1
(3, 5)	6
(3, 6)	3

Se selecciona la arista (3, 4) con peso mínimo. En las líneas 15 y 16 se agrega el vértice 4 al árbol de expansión mínima y la arista (3, 4) a *E*.

La siguiente vez que se ejecuta el ciclo "for" en las líneas 7 a la 14, las aristas con un vértice en el árbol y otro fuera del árbol son

Arista	*Peso*
(1, 2)	4
(1, 5)	3
(2, 4)	5
(3, 5)	6
(3, 6)	3
(4, 6)	6

Esta vez, dos aristas tienen peso mínimo de 3. Se construye un árbol de expansión mínima cuando se elige cualquiera de las dos aristas. En esta versión se seleccionó (1, 5). En las líneas 15 y 16 se agrega el vértice 5 al árbol de expansión mínima y la arista (1, 5) a *E*.

La siguiente vez que se ejecuta el ciclo "for" en las líneas 7 a la 14, las aristas con un vértice en el árbol y otro fuera son

Arista	*Peso*
(1, 2)	4
(2, 4)	5
(3, 6)	3
(4, 6)	6
(5, 6)	2

Se selecciona la arista (5, 6) con peso mínimo. En las líneas 15 y 16, se agrega el vértice 6 al árbol de expansión mínima y la arista (5, 6) se agrega a *E*.

La última vez que se ejecuta el ciclo "for" en las líneas 7 a la 14, las aristas con un vértice en el árbol y un vértice fuera son

Arista	*Peso*
(1, 2)	4
(2, 4)	5

Se selecciona la arista (1,2) con peso mínimo. En las líneas 15 y 16 se agrega el vértice 2 al árbol de expansión mínima y la arista (1,2) a *E*. El árbol de expansión mínima construido se muestra en la figura 9.4.3. ◀

El algoritmo de Prim proporciona un ejemplo de un **algoritmo ambicioso**. Un algoritmo ambicioso es uno que optimiza la elección en cada iteración. El principio se resume como "hacer lo mejor localmente". En el algoritmo de Prim, como se desea un árbol de ex-

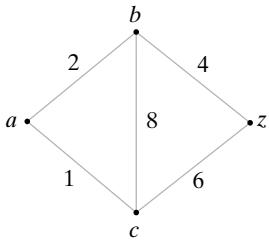


Figura 9.4.4 Gráfica que demuestra que seleccionar una arista con peso mínimo incidente en el último vértice que se agregó *no* necesariamente lleva a la ruta más corta. Comenzando en a, se obtiene (a, c, z), pero la ruta más corta de a a z es (a, b, z).

pansión mínima, en cada iteración simplemente se agrega una arista disponible con peso mínimo.

Optimizar en cada iteración no necesariamente ofrece una solución óptima al problema original. A continuación se mostrará (Teorema 9.4.5) que el algoritmo de Prim es correcto: sí se obtiene un árbol de expansión mínima. Como ejemplo de un algoritmo ambicioso que no lleva a una solución óptima, considere un "algoritmo de la ruta más corta" en el que en cada paso se selecciona una arista disponible con peso mínimo, incidente en el vértice que se agregó más recientemente. Si se aplica este algoritmo a la gráfica ponderada de la figura 9.4.4 para encontrar una ruta más corta de a a z, se seleccionaría la arista (a, c) y después la arista (c, z). Desafortunadamente, ésta no es la ruta más corta de a a z.

A continuación se prueba que el algoritmo de Prim es correcto.

Teorema 9.4.5

El algoritmo de Prim (algoritmo 9.4.3) es correcto; es decir, al terminar el algoritmo 9.4.3, T es un árbol de expansión mínima.

Demostración Sea T_i la gráfica construida por el algoritmo 9.4.3 después de la i-ésima iteración del ciclo "for", líneas 5 a la 17. De manera más precisa, el conjunto de aristas de T_i es el conjunto E construido después de la i-ésima iteración del ciclo "for", líneas 5 a la 17, y el conjunto de vértices de T_i es el conjunto de vértices sobre los cuales inciden las aristas en E. Sea T_0 una gráfica construida por el algoritmo 9.4.3 justo antes de entrar por primera vez al ciclo "for" en la línea 5; T_0 consiste en el único vértice s y ninguna arista. Más adelante en esta prueba, se suprime el conjunto de vértices y se hace referencia a una gráfica especificando su conjunto de aristas.

Por construcción, al terminar el algoritmo 9.4.3, la gráfica que se obtiene, T_{n-1}, es una subgráfica conexa acíclica de la gráfica dada G que contiene todos los vértices de G; así, T_{n-1} es un árbol de expansión de G.

Se usa inducción para demostrar que para toda $i = 0, \ldots, n-1$, T_i está contenida en un árbol de mínima expansión. Entonces, al terminar se deduce que T_{n-1} es un árbol de mínima expansión.

Si $i = 0$, T_0 consiste en un solo vértice. En este caso, T_0 está contenido en todo árbol de expansión mínima. Esto verifica el paso base.

Ahora, suponga que T_i está contenido en un árbol de expansión mínima T'. Sea V el conjunto de vértices en T_i. El algoritmo 9.4.3 selecciona una arista (j, k) de peso mínimo, donde $j \in V$ y $k \notin V$, y la agrega a T_i para producir T_{i+1}. Si (j, k) está en T', entonces T_{i+1} está contenida en el árbol de expansión mínima T'. Si (j, k) no está en T', $T' \cup \{(j, k)\}$ contiene un ciclo C. Elija una arista (x, y) en C, diferente de (j, k), con $x \in V$ y $y \notin V$. Entonces

$$w(x, y) \geq w(j, k). \tag{9.4.1}$$

En virtud de (9.4.1), la gráfica $T'' = [T' \cup \{(j, k)\}] - \{(x, y)\}$ tiene peso menor o igual que el peso de T'. Como T'' es un árbol de expansión, T'' es un árbol de expansión mínima. Como T_{i+1} está contenido en T'', se verifica el paso inductivo. La prueba queda completa.

Nuestra versión del algoritmo de Prim examina $\Theta(n^3)$ aristas en el peor caso (vea el ejercicio 6) para encontrar un árbol de mínima expansión para una gráfica que tiene n vértices. Es posible (vea el ejercicio 8) implementar el algoritmo de Prim de manera que sólo se examinen $\Theta(n^2)$ aristas en el peor caso. Como K_n tiene $\Theta(n^2)$ aristas, esta última versión es óptima.

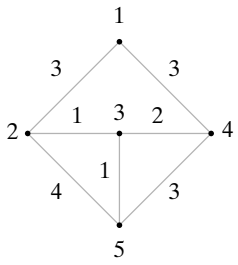
Sección de ejercicios de repaso

1. ¿Qué es un árbol de expansión mínima?
2. Explique cómo encuentra el algoritmo de Prim un árbol de expansión mínima.
3. ¿Qué es un algoritmo ambicioso?

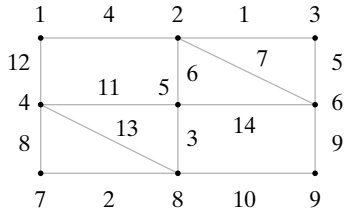
Ejercicios

En los ejercicios 1 al 5, encuentre el árbol de expansión mínima dado por el algoritmo 9.4.3 para cada gráfica.

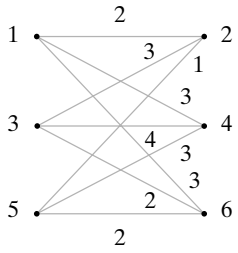
1.



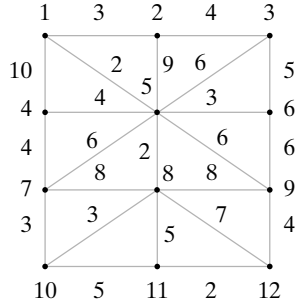
2.



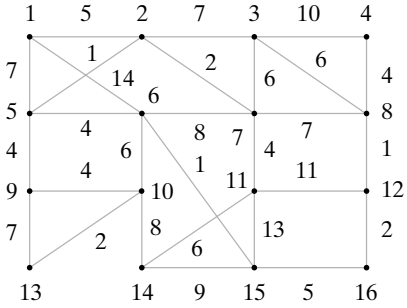
3.



4.



5.



6. Demuestre que el algoritmo 9.4.3 examina $\Theta(n^3)$ aristas en el peor caso.

Los ejercicios 7 al 9 se refieren a la versión alternativa del algoritmo de Prim (algoritmo 9.4.6).

Algoritmo 9.4.6

Versión alternativa del algoritmo de Prim

Este algoritmo encuentra un árbol de expansión mínima en un gráfica G conexa ponderada. En cada paso, algunos vértices tienen etiquetas temporales y otros tienen etiquetas permanentes. La etiqueta de un vértice i se denota por L_i.

Entrada: Gráfica conexa ponderada con vértices $1, \ldots, n$ y vértice inicial s. Si (i, j) es una arista, $w(i, j)$ es igual al peso de (i, j); si (i, j) no es una arista, $w(i, j)$ es igual a ∞ (un valor mayor que cualquier peso real)

Salida: Árbol de expansión mínima T

```
prim_altern(w, n, s) {
    sea T la gráfica con vértice s y ninguna arista
    for j = 1 to n {
        L_j = w(s, j)//estas etiquetas son temporales
        regreso(j) = s
    }
    L_j = 0
    se hace L_s permanente
    while (queden etiquetas temporales) {
        se elige la etiqueta temporal más pequeña L_i
        L_i se hace permanente
        se agrega la arista (i, regreso(i)) a T
        por cada etiqueta temporal L_k
            if (w(i, k) < L_k) {
                L_k = w(i, k)
                regreso (k) = i
            }
        }
        return T
    }
```

7. Muestre la manera en que el algoritmo 9.4.6 encuentra un árbol de expansión mínima para las gráficas de los ejercicios 1 al 5.

8. Demuestre que el algoritmo 9.4.6 examina $\Theta(n^2)$ aristas en el peor caso.

9. Pruebe que el algoritmo 9.4.6 es correcto; es decir, al terminar el algoritmo, T es un árbol de expansión mínima.

10. Sea G una gráfica conexa ponderada, sea v un vértice en G y sea e la arista con peso mínimo incidente en v. Demuestre que e está contenida en algún árbol de expansión mínima.

11. Sea G una gráfica conexa ponderada y sea v un vértice en G. Suponga que los pesos de las aristas incidentes en v son distintos. Sea e la arista con peso mínimo incidente en v. ¿Debe e estar contenida en todo árbol de expansión mínima?

12. Demuestre que cualquier algoritmo que encuentra un árbol de expansión mínima en K_n, cuando todos los pesos son iguales, debe examinar toda arista en K_n.

13. Demuestre que si todos los pesos en una gráfica conexa G son diferentes, G tiene un árbol de expansión mínima.

En los ejercicios 14 al 16, decida si la afirmación es falsa o verdadera. Si es verdadera, pruébela; de otra manera, dé un contraejemplo. En cada ejercicio, G es una gráfica conexa y ponderada.

14. Si todos los pesos en G son distintos, los árboles de expansión de G diferentes tienen pesos diferentes.

15. Si e es una arista en G cuyo peso es menor que el peso de cada tercera arista, e está en todo árbol de mínima expansión de G.

16. Si T es un árbol de expansión mínima de G, existe un etiquetado de los vértices de G de manera que el algoritmo 9.4.3 produce T.

17. Sea G una gráfica conexa ponderada. Demuestre que si, mientras sea posible, se elimina una arista de G con peso máximo y cuya eliminación no desconecta a G, el resultado es un árbol de expansión mínima para G.

★**18.** Escriba un algoritmo que encuentre un árbol de expansión mínima en una gráfica conexa ponderada.

19. Pruebe que su algoritmo del ejercicio 18 es correcto.

El algoritmo de Kruskal encuentra un árbol de expansión mínimo en una gráfica G conexa ponderada que tiene n vértices como sigue. La gráfica T inicialmente consiste en los vértices de G y ninguna arista. En cada iteración, se agrega una arista e a T con peso mínimo que no completa un ciclo en T. Cuando T tiene $n - 1$ aristas, se detiene.

20. Establezca formalmente el algoritmo de Kruskal.

21. Muestre cómo el algoritmo de Kruskal encuentra árboles de expansión mínima para las gráficas de los ejercicios 1 al 5.

22. Demuestre que el algoritmo de Kruskal es correcto; es decir, al terminar el algoritmo de Kruskal, T es un árbol de expansión mínima.

23. Sea V un conjunto de n vértices y sea s una "función de disimilitud" en $V \times V$ (vea el ejemplo 8.1.7). Sea G la gráfica ponderada completa que tiene vértices V y pesos $w(v_i, v_j) = s(v_i, v_j)$. Modifique el algoritmo de Kruskal de manera que agrupe datos en clases. Esta modificación se conoce como **método del vecino más cercano** (vea [Gosel]).

Los ejercicios 24 al 30 se refieren a la siguiente situación. Suponga que tenemos timbres de varias denominaciones y queremos elegir el mínimo número de timbres para completar una cantidad dada de importe postal. Considere un algoritmo ambicioso que selecciona timbres escogiendo todos los que puede de la denominación más grande, después todos los que puede de la denominación siguiente más grande, y así sucesivamente.

24. Demuestre que si las denominaciones disponibles son 1, 8 y 10 centavos, el algoritmo no siempre produce el menor número de timbres para completar un importe postal dado.

★**25.** Demuestre que si las denominaciones disponibles son 1, 5 y 25 centavos, el algoritmo produce el menor número de timbres para completar un importe dado.

26. Encuentre enteros positivos a_1 y a_2 tales que $a_1 > 2a_2 > 1$, a_2 no divide a a_1 y el algoritmo, con denominaciones disponibles 1, a_1, a_2, no siempre produce el menor número de timbres para completar el importe postal.

★***27.*** Encuentre enteros positivos a_1 y a_2 tales que $a_1 > 2a_2 > 1$, a_2 no divide a a_1 y el algoritmo, con denominaciones disponibles 1, a_1, a_2, produce el menor número de timbres para completar cualquier importe postal dado. Pruebe que sus valores en realidad dan una solución óptima.

★**28.** Suponga que las denominaciones disponibles son

$$1 = a_1 < a_2 < \cdots < a_n.$$

Demuestre, dando un contraejemplo, que la condición

$$a_i \geq 2a_{i-1} - a_{i-2}, \qquad 3 \leq i \leq n,$$

no es necesaria ni suficiente para que el algoritmo ambicioso sea óptimo.

★***29.*** Demuestre que el algoritmo ambicioso es óptimo para las denominaciones

$$1 = a_1 < a_2 < \cdots < a_m$$

si y sólo si ese algoritmo es óptimo para todas las denominaciones menores que $a_{m-1} + a_m$. (Este resultado se debe a William Seaman).

30. Demuestre que la cota $a_{m-1} + a_m$ en el ejercicio 29 no puede ser menor.

9.5 ➜ Árboles binarios

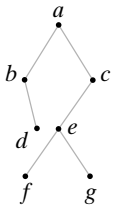


Figura 9.5.1 Árbol binario.

Los **árboles binarios** están entre los tipos especiales más importantes de árboles con raíz. Todo vértice en un árbol binario tiene cuando mucho dos hijos (vea la figura 9.5.1). Más aún, cada hijo se designa como un **hijo izquierdo** o un **hijo derecho**. Cuando se dibuja un árbol binario, un hijo izquierdo se dibuja a la izquierda y un hijo derecho se dibuja a la derecha. La definición formal es la siguiente.

Definición 9.5.1 ▶

Un *árbol binario* es un árbol con raíz en el que cada vértice tiene ningún hijo, un hijo o dos hijos. Si el vértice tiene un hijo se designa como un hijo izquierdo o como un hijo derecho (pero no ambos). Si un vértice tiene dos hijos, un hijo se designa como hijo izquierdo y el otro como hijo derecho. ◀

Ejemplo 9.5.2 ▶

En el árbol binario de la figura 9.5.1, el vértice b es el hijo izquierdo del vértice a y el vértice c es el hijo derecho del vértice a. El vértice d es el hijo derecho del vértice b; el vértice b no tiene hijo izquierdo. El vértice e es el hijo izquierdo del vértice c; el vértice c no tiene hijo derecho. ◀

Ejemplo 9.5.3 ▶

Un árbol que define un código de Huffman es un árbol binario. Por ejemplo, en el árbol del código de Huffman de la figura 9.1.8, moverse de un vértice a un hijo izquierdo corresponde a usar el bit 1, y moverse de un vértice a un hijo derecho corresponde a usar el bit 0. ◀

Un **árbol binario completo** es un árbol binario en el que cada vértice tiene dos o cero hijos. Un resultado fundamental acerca de loa árboles binarios completos es el siguiente teorema.

Teorema 9.5.4

Si T es un árbol binario completo con i vértices internos, entonces T tiene $i + 1$ vértices terminales y $2i + 1$ vértices en total.

Demostración Los vértices de T consisten en los vértices que son hijos (de algún padre) y los vértices que no son hijos (de ningún padre). Existe un vértice que no es hijo: la raíz. Como hay i vértices internos, cada uno con dos hijos, hay $2i$ hijos. Entonces, el número total de vértices de T es $2i + 1$, y el número total de vértices terminales es

$$(2i + 1) - i = i + 1.$$

Ejemplo 9.5.5 ▶

Un torneo por eliminación sencilla es un torneo en el que se elimina a un competidor después de una derrota. La gráfica de un torneo por eliminación sencilla es un árbol binario completo (vea la figura 9.5.2). Los nombres de los competidores se listan a la izquierda. Los ganadores avanzan a la derecha. En algún momento, habrá un solo ganador en la raíz. Si el número de competidores no es una potencia de 2, algunos de ellos reciben pase automático. En la figura 9.5.2, el concursante 7 tiene un pase automático en la primera ronda.

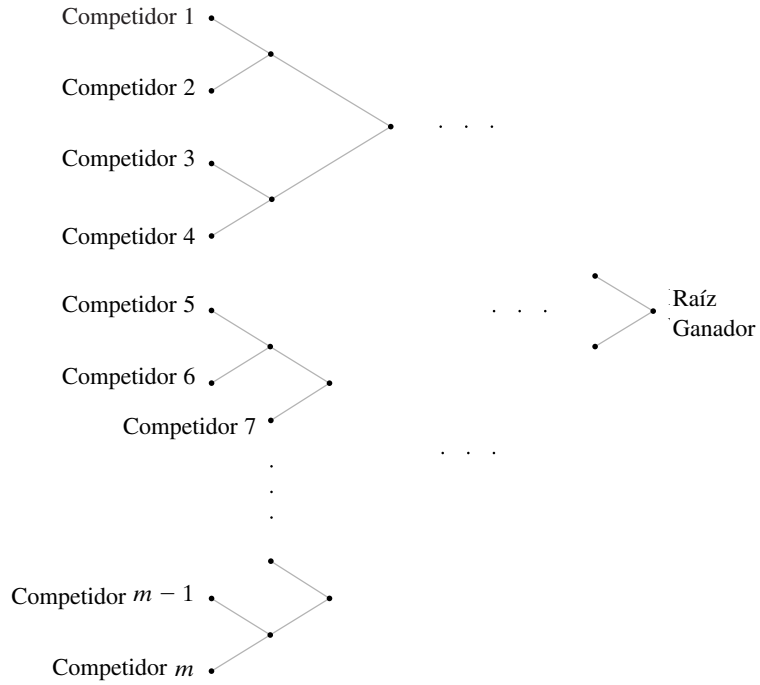


Figura 9.5.2 Gráfica de un torneo por eliminación sencilla (árbol binario completo).

Se demostrará que si hay n competidores en un torneo por eliminación sencilla, se juega un total de $n - 1$ juegos.

El número de competidores es el mismo que el número de vértices terminales, y el número de juegos i es igual al número de vértices internos. Entonces, por el Teorema 9.5.4,

$$n + i = 2i + 1,$$

de manera que $i = n - 1$. ◀

El siguiente resultado acerca de árboles binarios relaciona el número de vértices terminales con la altura.

Teorema 9.5.6

Si un árbol binario de altura h tiene t vértices terminales, entonces

$$\lg t \leq h. \tag{9.5.1}$$

Demostración Se probará la desigualdad equivalente

$$t \leq 2^h \tag{9.5.2}$$

por inducción sobre h. La desigualdad (9.5.1) se obtiene de (9.5.2) tomando el logaritmo base 2 en ambos lados de (9.5.2).

Si $h = 0$, el árbol binario consiste en un solo vértice. En este caso, $t = 1$ y, por lo tanto, (9.5.2) es verdadera.

Suponga que el resultado se cumple para un árbol binario cuya altura es menor que h. Sea T un árbol binario de altura $h > 0$ con t vértices terminales. Suponga primero que la raíz de T sólo tiene un hijo. Si se elimina la raíz y la arista incidente en la raíz, el árbol que se obtiene tiene altura $h - 1$ y el mismo número de terminales que T. Por inducción, $t \leq 2^{h-1}$. Como $2^{h-1} < 2^h$, (9.5.2) está establecida para este caso.

Ahora suponga que la raíz de T tiene hijos v_1 y v_2. Sea T_i el subárbol con raíz en v_i y suponga que T_i tiene altura h_i y t_i vértices terminales, $i = 1, 2$. Por inducción,

$$t_i \leq 2^{h_i}, \qquad i = 1, 2. \tag{9.5.3}$$

Los vértices terminales de T consisten en los vértices terminales de T_1 y T_2. Entonces

$$t = t_1 + t_2. \tag{9.5.4}$$

Al combinar (9.5.3) y (9.5.4), se obtiene

$$t = t_1 + t_2 \leq 2^{h_1} + 2^{h_2} \leq 2^{h-1} + 2^{h-1} = 2^h.$$

Esto verifica el paso inductivo y la prueba queda completa.

Ejemplo 9.5.7 ▶

El árbol binario de la figura 9.5.3 tiene altura $h = 3$ y el número de vértices terminales $t = 8$. Para este árbol, la desigualdad (9.5.1) se convierte en una igualdad.

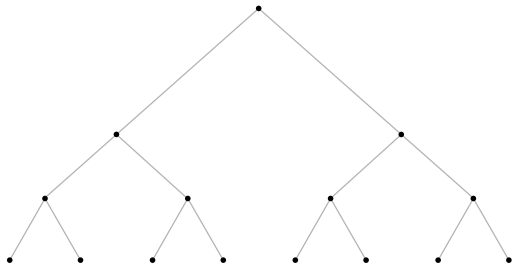


Figura 9.5.3 Árbol binario con altura $h = 3$ con $t = 8$ terminales. Para este árbol binario, $\lg t = h$.

◀

Suponga que se tiene un conjunto S cuyos elementos se pueden ordenar. Por ejemplo, si S consiste en números, se utiliza el orden normal definido sobre los números, y si S contiene cadenas de caracteres alfanuméricos, se emplea un orden lexicográfico. Los árboles binarios se usan ampliamente en ciencias de la computación para almacenar elementos de un conjunto ordenado como un conjunto de números o un conjunto de cadenas. Si el dato $d(v)$ se almacena en el vértice v y el dato $d(w)$ se guarda en el vértice w, entonces si v es un hijo izquierdo (o hijo derecho) de w, se garantiza que existe alguna relación de orden entre $d(v)$ y $d(w)$. Un ejemplo es un **árbol de búsqueda binaria**.

Definición 9.5.8 ▶

Un *árbol de búsqueda binaria* es un árbol binario T en el que se asocian datos a los vértices. Los datos están arreglados de manera que, para *cada* vértice v en T, cada dato en el subárbol de la izquierda de v es menor que el dato en v, y cada dato en el subárbol de la derecha de v es mayor que el dato en v. ◀

Ejemplo 9.5.9 ▶

Las palabras

OTRA PERSONA NO DIRÍA TODO JUNTO
LO TENDRÍA MEMORIZADO **(9.5.5)**

se pueden colocar en un árbol de búsqueda binaria como se ve en la figura 9.5.4. Observe que para cualquier vértice v, cada dato en el subárbol izquierdo de v es menor que (es decir, precede alfabéticamente) el dato en v y cada dato en el subárbol derecho de v es mayor que el dato en v.

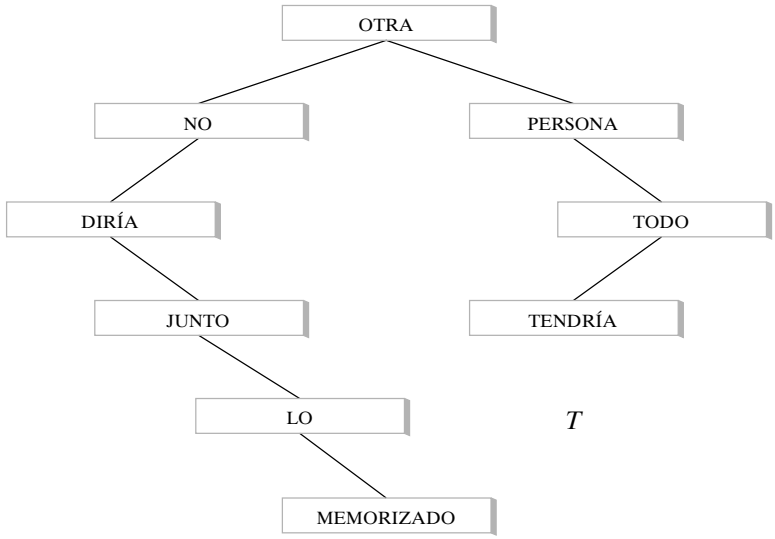


Figura 9.5.4 Un árbol de búsqueda binaria. ◀

En general, habrá muchas maneras de colocar datos en un árbol de búsqueda binaria. La figura 9.5.5 muestra otro árbol de búsqueda binario que almacena las palabras (9.5.5).

El árbol de búsqueda binaria T de la figura 9.5.4 se construyó de la siguiente manera. Se comienza con un **árbol vacío**, es decir, un árbol sin vértices ni aristas. Después se inspecciona cada palabra en (9.5.5) *en el orden en que aparecen*, OTRA primero, después PERSONA, después NO, etcétera. Para comenzar, se crea un vértice y la primera palabra (OTRA) se coloca ahí. Se designa este vértice como la raíz. En adelante, dada un palabra en la lista (9.5.5) se agrega un vértice v y una arista al árbol y se coloca la palabra en el vértice v. Para decidir

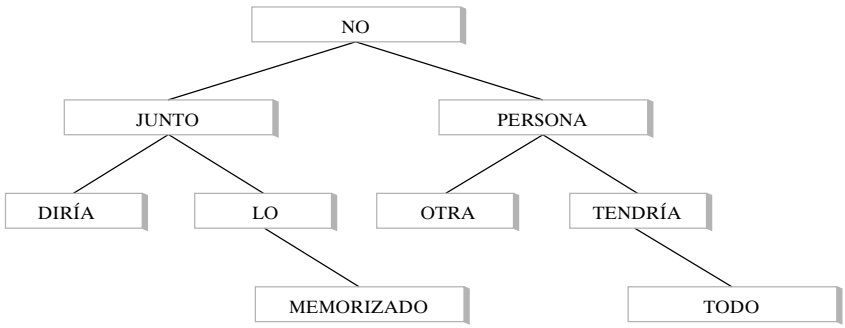


Figura 9.5.5 Otro árbol de búsqueda binaria que almacena las mismas palabras que el árbol en la figura 9.5.4.

dónde agregar el vértice y la arista, se comienza en la raíz. Si la palabra que se va agregar es menor que (según un orden lexicográfico) la palabra en la raíz, nos movemos al hijo izquierdo; si la palabra es mayor que la palabra en la raíz, nos movemos al hijo derecho. Si no hay hijo, se crea uno, se coloca una arista incidente en la raíz y un nuevo vértice, y se pone la palabra en el nuevo vértice. Si hay un hijo v, se repite este proceso; es decir, se compara la palabra que se va a agregar con la palabra en v y nos movemos al hijo izquierdo de v si la palabra es menor que la palabra en v; de otra manera, nos movemos al hijo derecho de v. Si no hay un hijo hacia donde ir, se crea uno, se coloca una arista incidente en v y el nuevo vértice, y se pone la palabra en este último. Si hay un hijo hacia donde moverse, se repite el proceso. En algún momento, se coloca la palabra en el árbol. Después se toma la siguiente palabra en la lista, se compara con la raíz, nos movemos a la izquierda o a la derecha, se compara con el nuevo vértice, nos movemos a la izquierda o a la derecha, y así sucesivamente; finalmente, esto se almacena como el árbol. De esta manera, se guardan todas las palabras en el árbol y se crea un árbol de búsqueda binaria. Se establece formalmente este método para construir un árbol de búsqueda binaria como el algoritmo 9.5.10.

Algoritmo 9.5.10

Construcción de un árbol de búsqueda binaria

Este algoritmo construye un árbol de búsqueda binaria. La entrada se lee en el orden que se entrega. Después de leer cada palabra, se inserta en árbol.

Entrada: Una sucesión $w_1, \ldots, w_n$ de palabras distintas y la longitud n de la sucesión
Salida: Un árbol de búsqueda binaria T

```
búsq_bin_libre(w, n) {
    sea T el árbol con un vértice, raíz
    se almacena w_1 en raíz
    for i = 2 to n {
        v = raíz
        busca = verdadero //encuentra lugar para w_i
        while (busca) {
            s = palabra en v
            if (w_i < s)
                if (v no tiene hijo izquierdo) {
                    agregar hijo izquierdo l a v
                    almacenar w_i en l
                    busca = falso //termina búsqueda
                }
                else
                    v = hijo izquierdo de v
            else //w_i > s
                if(v no tiene hijo derecho) {
                    agregar hijo derecho r a v
                    almacenar w_i en r
                    busca = falso //termina búsqueda
                }
                else
                    v = hijo derecho de v
        }//termina while
    } //termina for
    return T
}
```

Los árboles de búsqueda binaria son útiles para localizar datos. Esto es, a partir de un dato D, es fácil determinar si D es un árbol de búsqueda binaria y, si está presente, dónde se localiza. Para determinar si un dato D está en el árbol de búsqueda binaria, comenzaríamos en la raíz. Después compararíamos D repetidas veces con el dato en el vértice actual. Si D es igual al dato en el vértice actual, encontramos D y nos detenemos. Si D es

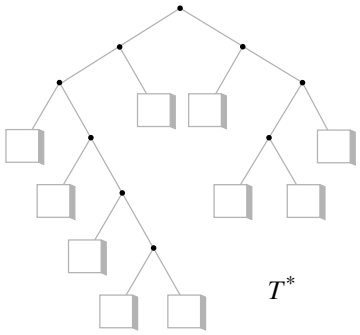


Figura 9.5.6 Expansión de un árbol de búsqueda binaria a un árbol binario completo.

menor que el dato en el vértice v actual, nos movemos al hijo izquierdo de v y repetimos el proceso. Si D es mayor que el dato en el vértice v actual, nos movemos al hijo derecho de v y repetimos el proceso. Si en cualquier punto el hijo al que vamos falta, se concluye que D no está en el árbol. (El ejercicio 2 pide un enunciado formal de este proceso).

El tiempo dedicado a buscar un artículo en un árbol de búsqueda binaria es mayor cuando el artículo no está presente y seguimos la trayectoria más larga desde la raíz. Entonces el tiempo máximo para buscar un artículo en un árbol de búsqueda binaria es aproximadamente proporcional a la altura del árbol. Por lo tanto, si la altura de un árbol de búsqueda binaria es pequeña, buscar en el árbol siempre será muy rápido (vea el ejercicio 21). Se conocen muchas formas de minimizar la altura de un árbol de búsqueda binaria (vea, por ejemplo, [Cormen]).

Se harán aseveraciones más precisas acerca del peor caso de búsqueda binaria. Sea T un árbol de búsqueda binaria con n vértices y sea T^* el árbol binario completo obtenido de T agregando hijos izquierdos y derechos a los vértices existentes en T siempre que sea posible. En la figura 9.5.6 aparece el árbol binario completo que se obtiene al modificar el árbol de búsqueda binaria de la figura 9.5.4. Los vértices agregados se dibujan como cuadros. Una búsqueda no exitosa en T corresponde a llegar a un vértice (cuadro) agregado en T^*. Defina el tiempo del peor caso necesario para ejecutar el procedimiento de búsqueda como la altura h del árbol T^*. Por el Teorema 9.5.6, $\lg t \leq h$, donde t es el número de vértices terminales en T^*. El árbol binario completo T^* tiene n vértices internos, de manera que por el Teorema 9.5.4, $t = n + 1$. Entonces, en el peor caso, el tiempo será igual a por lo menos $\lg t = \lg(n + 1)$. El ejercicio 3 muestra que si la altura de T se minimiza, el peor caso requiere un tiempo igual a $\lceil \lg(n + 1) \rceil$. Por ejemplo, como

$$\lceil \lg(2{,}000{,}000 + 1) \rceil = 21,$$

es posible almacenar 2 millones de elementos en un árbol de búsqueda binaria, o determinar que no está presente, en cuando mucho 21 pasos.

Sección de ejercicios de repaso

1. Defina *árbol binario*.
2. ¿Qué es un hijo izquierdo en un árbol binario?
3. ¿Qué es un hijo derecho en un árbol binario?
4. ¿Qué es un árbol binario completo?
5. Si T es un árbol binario completo con i vértices internos, ¿cuántos vértices terminales tiene T?
6. Si T es un árbol binario completo con i vértices internos, ¿cuántos vértices tiene T?
7. ¿Cómo se relaciona la altura de un árbol binario con el número de sus vértices terminales?
8. ¿Qué es un árbol de búsqueda binaria?
9. Dé un ejemplo de un árbol de búsqueda binaria.
10. Dé un algoritmo para construir un árbol de búsqueda binaria.

Ejercicios

1. Coloque las palabras FOTO SOBRE ARRIBA SIETE YA ANTES OBSERVA FELIZ BARATO FOMENTO, en el orden en el que aparecen, en un árbol de búsqueda binaria.
2. Escriba un algoritmo formal de búsqueda en un árbol de búsqueda binaria.
3. Escriba un algoritmo que almacene n palabras diferentes en un árbol de búsqueda binaria T de altura mínima. Demuestre que el árbol T^* obtenido, como se describe en el texto, tiene altura $\lceil \lg(n + 1) \rceil$.
4. ¿Falso o verdadero? Sea T un árbol binario. Si para cada vértice v en T el dato en v es mayor que el dato en el hijo izquierdo de v, y el dato en v es menor que el dato en el hijo derecho de v, entonces T es un árbol de búsqueda binaria. Explique.

En los ejercicios 5 al 7, dibuje una gráfica que tenga las propiedades indicadas o explique por qué no existe la gráfica.

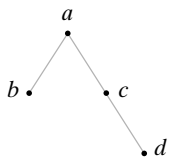
5. Árbol binario completo; 4 vértices internos; 5 vértices terminales
6. Árbol binario completo; altura = 3; 9 vértices terminales
7. Árbol binario completo; altura = 4; 9 vértices terminales
8. Un **árbol *m*-ario completo** es un árbol con raíz tal que todo padre tiene m hijos ordenados. Si T es un árbol m-ario completo con i vértices internos, ¿cuántos vértices tiene T? ¿Cuántos vértices terminales tiene T? Pruebe sus resultados.
9. Proporcione un algoritmo para construir un árbol binario completo con $n > 1$ vértices terminales.
10. Proporcione un algoritmo recursivo para insertar una palabra en un árbol de búsqueda binaria.
11. Encuentre la altura máxima de un árbol binario completo que tiene t vértices terminales.
12. Escriba un algoritmo que pruebe si un árbol binario en el que se almacenan datos en los vértices es un árbol de búsqueda binaria.
13. Sea T un árbol binario completo. Sea I la suma de las longitudes de las trayectorias simples de la raíz a los vértices internos. I se llama *longitud de trayectoria interna*. Sea E la suma de las longi-

tudes de las trayectorias simples de la raíz a los vértices terminales. E se llama *longitud de trayectoria externa*. Pruebe que si T tiene n vértices terminales, entonces $E = I - 2n$.

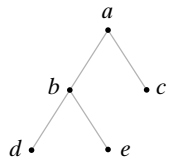
Un árbol binario T está balanceado si para cada vértice v en T, las alturas de los subárboles izquierdo y derecho de v difieren cuando mucho en 1. (Aquí, la altura de un árbol vacío se define como −1).

Establezca si cada árbol en los ejercicios 14 al 17 está balanceado o no.

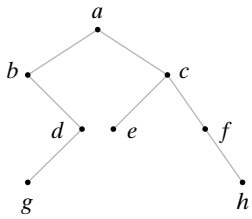
14.



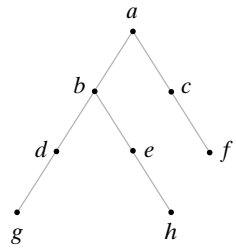
15.



16.



17.



En los ejercicios 18 al 20, N_h se define como el número mínimo de vértices en un árbol binario balanceado de altura h y $f_1, f_2, \ldots$ denota la sucesión de Fibonacci.

18. Demuestre que $N_0 = 1$, $N_1 = 2$ y $N_2 = 4$.

19. Demuestre que $N_h = 1 + N_{h-1} + N_{h-2}$, para $h \geq 0$.

20. Demuestre que $N_h = f_{h+3} - 1$, para $h \geq 0$.

★***21.*** Demuestre que la altura h de un árbol binario balanceado de n vértices satisface $h = O(\lg n)$. Este resultado indica que el tiempo de búsqueda, en el peor caso, en un árbol de búsqueda binaria balanceado de n vértices es $O(\lg n)$.

★***22.*** Pruebe que si un árbol binario de altura h tiene $n \geq 1$ vértices, entonces $\lg n < h + 1$. Este resultado, junto con el ejercicio 21, muestra que el tiempo de búsqueda en el peor caso en un árbol de búsqueda binaria balanceado de n vértices es $\Theta(\lg n)$.

9.6 ➜ Recorridos de árboles

La búsqueda a lo ancho y la búsqueda a profundidad proporcionan maneras de "caminar" por un árbol, es decir, de recorrerlo en forma sistemática de modo que se visite cada vértice exactamente una vez. En esta sección se consideran tres métodos adicionales para recorrer árboles. Estos recorridos se definen de manera recursiva.

Algoritmo 9.6.1

Recorrido preorden

Este algoritmo recursivo procesa los vértices de un árbol binario usando el recorrido preordenado.

Entrada: *PT*, la raíz de un árbol binario
Salida: Dependiente de cómo se interprete el "proceso" en la línea 3

```
   preorden(PT) {
1.    if (PT está vacío)
2.       return
3.    procesar PT
4.    l = hijo izquierdo de PT
5.    preorden(l)
6.    r = hijo derecho de PT
7.    preorden(r)
   }
```

Se examinará el algoritmo 9.6.1 para algunos casos sencillos. Si el árbol binario está vacío, nada se procesa ya que, en este caso, el algoritmo simplemente regresa a la línea 2.

Suponga que la entrada consiste en un árbol con un solo vértice. Se hace *PT* igual a la raíz y se invoca *preorden*(*PT*). Como *PT* no está vacío, se procede a la línea 3, donde se procesa la raíz. En la línea 5, se llama a *preorden* con *PT* igual al hijo izquierdo (vacío) de la raíz. Sin embargo, se acaba de ver que cuando entra un árbol vacío a *preorden*, nada se procesa. De manera similar para la línea 7, la entrada a *preorden* es un árbol vacío y nada se procesa. Entonces, cuando la entrada consiste en un árbol con un solo vértice, se procesa la raíz y se regresa.

Ahora suponga que la entrada es el árbol de la figura 9.6.1. Se hace *PT* igual a la raíz y se invoca a *preorden*(*PT*). Como *PT* no está vacío, se procede a la línea 3, conde se procesa la raíz. En la línea 5 se llama a *preorden* con *PT* igual al hijo izquierdo de la raíz (vea la figura 9.6.2). Se acaba de ver que si el árbol de entrada a *preorden* consiste en un solo vértice, *preorden* procesa ese vértice. Entonces, se procesa a continuación el vértice *B*. De manera similar, en la línea 7, se procesa el vértice *C*. Entonces los vértices se procesan en el orden *ABC*.

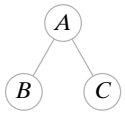


Figura 9.6.1 Entrada para el algoritmo 9.6.1.

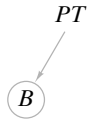


Figura 9.6.2 En la línea 5 del algoritmo 9.6.1, donde la entrada es el árbol de la figura 9.6.1.

Ejemplo 9.6.2 ▶

¿En qué orden se procesan los vértices del árbol de la figura 9.6.3 si se usa el recorrido preordenado?

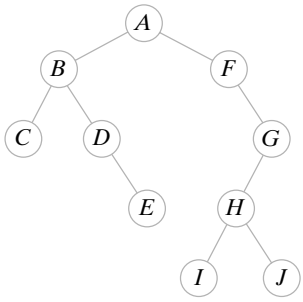


Figura 9.6.3 Un árbol binario. El preorden es *A B C D E F G H I J*. El entreorden es *C B D E A F I H J G*. El postorden es *C E D B I J H G F A*.

Siguiendo las líneas 3 a la 7 (raíz/izquierda/derecha) del algoritmo 9.6.1, el recorrido procede como se muestra en la figura 9.6.4. Entonces el orden de procesamiento es

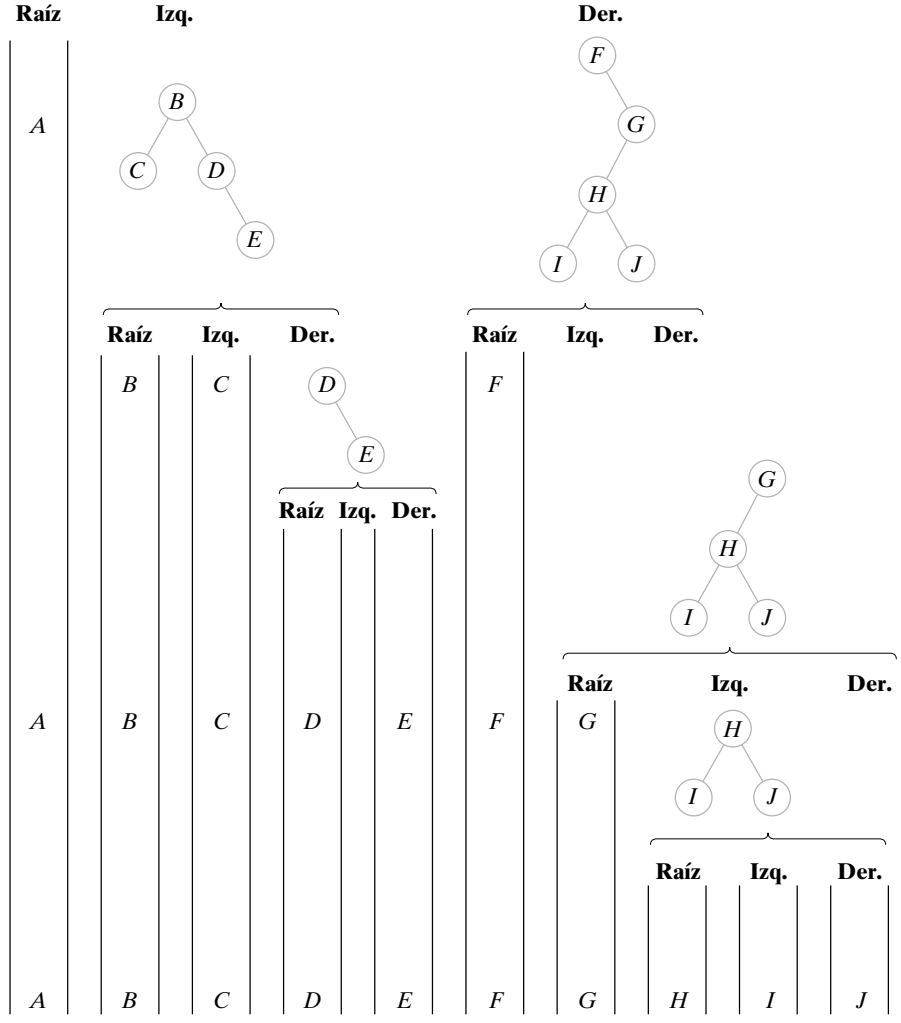


Figura 9.6.4 Recorrido preordenado del árbol de la figura 9.6.3.

◀

El recorrido entreorden y el postorden se obtienen cambiando la posición de la línea 3 (raíz) en el algoritmo 9.6.1. "Pre", "entre" y "post" se refieren a la posición de la raíz en el recorrido; es decir, "preorden" significa raíz primero, "entreorden" significa la raíz en segundo lugar y "postorden" quiere decir la raíz al último.

Algoritmo 9.6.3

Recorrido entreorden

Este algoritmo recursivo procesa los vértices de un árbol binario usando el recorrido entreorden.

Entrada: *PT*, la raíz de un árbol binario
Salida: Dependiente de cómo se interprete el "proceso" en la línea 5

```
   entreorden(PT) {
1.    if (PT está vacío)
2.       return
3.    l = hijo izquierdo de PT
4.    entreorden(l)
5.    procesar PT
6.    r = hijo derecho de PT
7.    entreorden(r)
   }
```

Ejemplo 9.6.4 ▶

¿En qué orden se procesan los vértices del árbol binario de la figura 9.6.3 si se usa el recorrido entreorden?

Siguiendo las líneas 3 a la 7 (izquierda/raíz/derecha) del algoritmo 9.6.3, se obtiene la lista de entreorden *C B D E A F I H J G*. ◀

Algoritmo 9.6.5

Recorrido postorden

Este algoritmo recursivo procesa los vértices de un árbol binario usando el recorrido postorden.

Entrada: *PT*, la raíz de un árbol binario
Salida: Dependiente de cómo se interprete el "proceso" en la línea 7

```
   postorden(PT) {
1.    if (PT está vacío)
2.       return
3.    l = hijo izquierdo de PT
4.    postorden(l)
5.    r = hijo derecho de PT
6.    postorden(r)
7.    procesar PT
   }
```

Ejemplo 9.6.6 ▶

¿En qué orden se procesan los vértices del árbol binario de la figura 9.6.3 si se usa el recorrido postorden?

Siguiendo las líneas 3 a la 7 (izquierdo/derecho/raíz) del algoritmo 9.6.5, se obtiene la lista postorden *C E D B I J H G F A*. ◀

Observe que el recorrido postorden se obtiene siguiendo la ruta mostrada en la figura 9.6.5, y que el recorrido postorden inverso se obtiene siguiendo la ruta mostrada en la figura 9.6.6.

Si los datos se almacenan en un árbol de búsqueda binaria, como se describe en la sección 9.5, el recorrido entreorden procesará los datos en orden, ya que la secuencia izquierdo/raíz/derecho está de acuerdo con el orden de los datos en el árbol.

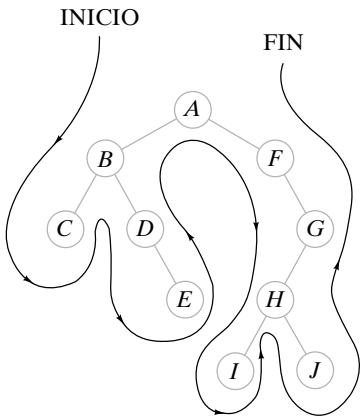


Figura 9.6.5 Recorrido preorden.

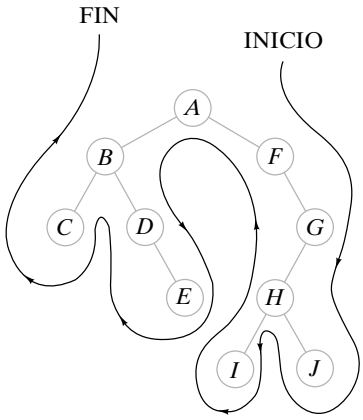


Figura 9.6.6 Recorrido postorden en reversa.

En el resto de esta sección se consideran representaciones mediante árboles binarios de expresiones aritméticas. Estas representaciones facilitan la evaluación de las expresiones por computadora.

Se restringirán los operadores a $+$, $-$, * y /. Un ejemplo de una expresión con estos operadores es

$$(A + B) * C - D/E. \qquad \textbf{(9.6.1)}$$

Esta manera estándar de representar expresiones se llama **forma de entrefijo de una expresión**. Se hace referencia a las variables A, B, C, D y E como **operandos**. Los **operadores** $+$, $-$, $*$ y / operan sobre pares de expresiones. En la forma de entrefijo de una expresión, un operador aparece entre sus operandos.

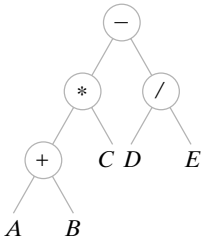


Figura 9.6.7 Representación del árbol binario de la expresión $(A + B)*C - D/E$.

Una expresión como (9.6.1) se puede representar como un árbol binario. Los vértices terminales corresponden a los operandos y los vértices internos a los operadores. La expresión (9.6.1) se representaría como se ilustra en la figura 9.6.7. En la representación de árbol binario de una expresión, un operador opera sobre sus subárboles izquierdo y derecho. Por ejemplo, en el subárbol cuya raíz es / en la figura 9.6.7, el operador división opera sobre los operandos D y E; es decir, D debe dividirse entre E. En el subárbol cuya raíz es * en la figura 9.6.7, el operador multiplicación opera sobre el subárbol encabezado por $+$, que en sí representa una expresión, y C.

En un árbol binario se distinguen los subárboles izquierdo y derecho de un vértice. Los subárboles izquierdo y derecho de un vértice corresponden a los operandos o expresiones izquierdo y derecho. Esta distinción izquierdo/derecho es importante en las expresiones. Por ejemplo, $4 - 6$ y $6 - 4$ son diferentes.

Si se recorre el árbol binario de la figura 9.6.7 usando entreorden e insertando un par de paréntesis para cada operación, se obtiene

$$(((A + B) * C) - (D/E)) .$$

Esta forma de expresión se llama **forma de la expresión con paréntesis**. En esta forma no se necesita especificar qué operaciones (como multiplicación) deben realizarse antes que otras (como suma), ya que los paréntesis dictan el orden sin ambigüedades.

Si se recorre el árbol de la figura 9.6.7 usando postorden, se obtiene

$$AB+C*DE/- .$$

WWW Esta forma de la expresión se llama **forma de posfijo de la expresión** (o **notación polaca inversa**). En posfijo, el operador sigue a los operandos. Por ejemplo, los primeros tres símbolos $AB+$ indican que A y B deben sumarse. Las ventajas de la forma posfijo sobre la de entrefijo son que en posfijo no se necesitan paréntesis y tampoco son necesarias convenciones respecto al orden de las operaciones. La expresión será evaluada sin ambigüedades. Por éstas y otras razones, muchos compiladores traducen las expresiones entrefijo a la forma posfijo. Además, algunas calculadoras requieren que se introduzcan expresiones en la forma posfijo.

Una tercera forma para una expresión se puede obtener aplicando el recorrido preorden a una representación por árbol binario de una expresión. En este caso, el resultado se llama **forma de prefijo de la expresión** (o **notación polaca**). Igual que en el posfijo, no se necesitan paréntesis ni convenciones en cuanto al orden de las operaciones. La forma de prefijo (9.6.1) que se obtiene aplicando el recorrido preorden al árbol de la figura 9.6.7 es

$$-*+ABC/DE.$$

Sección de ejercicios de repaso

1. ¿Qué es un recorrido preorden?

2. Desarrolle un algoritmo para ejecutar un recorrido preorden.

3. ¿Qué es un recorrido entreorden?

4. Desarrolle un algoritmo para ejecutar un recorrido entreorden.

5. ¿Qué es un recorrido postorden?

6. Desarrolle un algoritmo para ejecutar un recorrido postorden.

7. ¿Qué es una forma de prefijo de una expresión?

8. ¿Cuál es un nombre alternativo de la forma de prefijo de una expresión?

9. ¿Qué es la forma de entrefijo de una expresión?

10. ¿Qué es la forma de posfijo de una expresión?

11. ¿Cuál es un nombre alternativo para la forma de posfijo de una expresión?

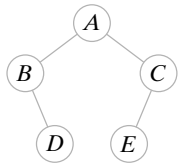
12. ¿Qué ventajas tienen las formas de prefijo y posfijo de una expresión sobre la forma de entrefijo?

13. Explique cómo se puede emplear un árbol para representar una expresión.

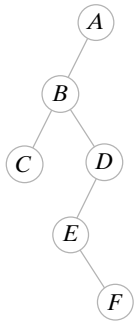
Ejercicios

En los ejercicios 1 al 5, enumere el orden en el que los vértices se procesan usando los recorridos preorden, entreorden y postorden.

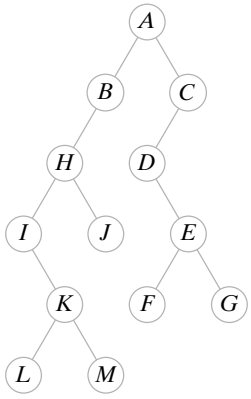
1.



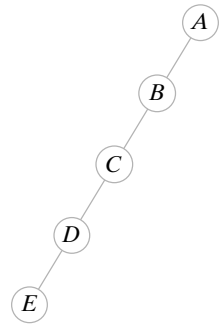
2.



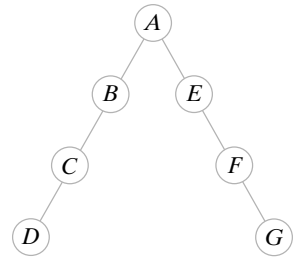
3.



4.



5.



En los ejercicios 6 al 10, represente la expresión como un árbol binario y escriba sus formas de prefijo y posfijo.

6. $(A + B) * (C - D)$

7. $((A - C) * D) / (A + (B + D))$

8. $(A * B + C * D) - (A/B - (D + E))$

9. $(((A + B) * C + D) * E) - ((A + B) * C - D)$

10. $(A * B - C/D + E) + (A - B - C - D * D)/(A + B + C)$

En los ejercicios 11 al 15, represente la expresión en posfijo como un árbol binario y escriba la forma de prefijo, la forma usual de entrefijo y la forma de entrefijo con paréntesis completos de la expresión.

11. $AB+C-$

12. $ABC+-$

13. $ABCD+*/E-$

14. $ABC**CDE+/-$

15. $AB+CD*EF/--A*$

En los ejercicios 16 al 21, encuentre el valor de la expresión en posfijo si $A = 1$, $B = 2$, $C = 3$ *y* $D = 4$.

16. $ABC+-$

17. $AB+C-$

18. $AB+CD*AA/--B*$

19. $ABC**ABC++-$

20. $ABAB*+*D*$

21. $ADBCD*-+*$

22. Demuestre, por ejemplo, que los árboles binarios distintos con vértices *A*, *B* y *C* pueden tener la misma lista de preorden *A B C*.

23. Demuestre que hay un árbol binario único con seis vértices cuya lista de vértices de preorden es *A B C E F D* y cuya lista de vértices de entreorden es *A C F E B D*.

★**24.** Escriba un algoritmo que reconstruya el árbol binario dados sus ordenamientos de preorden e entreorden.

25. Dé ejemplos de árboles binarios diferentes, B_1 y B_2, cada uno con dos vértices, donde la lista de vértices de preorden de B_1 es igual a la lista de preorden de B_2 y la lista de vértices de postorden de B_1 es igual a la lista de postorden de B_2.

26. Sean P_1 y P_2 permutaciones de *A B C D E F*. ¿Existe un árbol binario con vértices *A*, *B*, *C*, *D*, *E* y *F* cuya lista de preorden sea P_1 y cuya lista de entreorden sea P_2? Explique.

27. Escriba un algoritmo recursivo que imprima el contenido de los vértices terminales de un árbol binario de izquierda a derecha.

28. Escriba un algoritmo recursivo que intercambie todos los hijos izquierdos y derechos de un árbol binario.

29. Escriba un algoritmo recursivo que inicialice cada vértice de un árbol binario con el número de sus descendientes.

En los ejercicios 30 y 31, toda expresión implica sólo los operandos A, B, . . . , Z y las operaciones $+, -, *, /$.

★***30.*** Dé una condición necesaria y suficiente para que una cadena de símbolos sea una expresión en posfijo válida.

31. Escriba un algoritmo que, dada la representación por árbol binario de una expresión, produzca la forma de entrefijo con paréntesis completos de la expresión.

32. Escriba un algoritmo que imprima los caracteres y sus códigos dado un árbol de codificación de Huffman (vea el ejemplo 9.1.8). Suponga que cada vértice terminal almacena un carácter y su frecuencia.

Utilice las siguientes definiciones en los ejercicios 33 al 38.

Sea $G = (V, E)$ *un gráfica no dirigida simple. Una cubierta de los vértices de G es un subconjunto* V' *de V tal que para cada arista* $(v, w) \in E$, *ya sea* $v \in V'$ *o* $w \in V'$. *El tamaño de la cubierta de vértices* V' *es el número de vértices en* V'. *Una* cubierta de vértices óptima *es una cubierta de vértices de tamaño mínimo.*

Un conjunto ajeno de aristas *para G es un subconjunto* E' *de E tal que para cada par de aristas distintas* $e_1 = (v_1, w_1)$ *y* $e = (v_2, w_2)$ *en* E', *se tiene*

$$\{v_1, w_1\} \cap \{v_2, w_2\} = \emptyset.$$

33. Pruebe que para cada *n*, existe una gráfica conexa con *n* vértices que tiene una cubierta de vértices de tamaño 1.

34. Demuestre que el tamaño de una cubierta de vértices óptima de la gráfica completa con *n* vértices es $n - 1$.

35. El tamaño de una cubierta de vértices óptima de una gráfica con *n* vértices, ¿puede ser igual a *n*? Explique su respuesta.

★***36.*** Escriba un algoritmo que encuentre una cubierta de vértices óptima de un árbol $T = (V, E)$ cuyo tiempo en el peor caso es $\Theta(|E|)$.

37. Demuestre que si V' es cualquier cubierta de vértices de una gráfica *G* y E' es cualquier conjunto ajeno de aristas de *G*, entonces $|E'| \leq |V'|$.

38. Dé un ejemplo de una gráfica conexa en la que para cada cubierta de vértices V' y cada conjunto ajeno de aristas E', se tiene $|E'| < |V'|$. Pruebe que su ejemplo tiene la propiedad requerida.

39. Muestre cómo un árbol binario con *n* aristas se puede codificar como una cadena de $n + 1$ unos y $n + 1$ ceros donde, si se lee de izquierda a derecha, el número de ceros nunca excede al número de unos. Demuestre que cada cadena de este tipo representa un árbol binario. *Sugerencia:* Considere un recorrido preorden del árbol binario en el que un 1 significa que una arista está presente, y un 0 significa que una arista está ausente. Agregue un uno adicional al principio de la cadena, y elimine el último cero.

9.7 ➜ Árboles de decisiones y tiempo mínimo para ordenar

El árbol binario de la figura 9.7.1 da un algoritmo para seleccionar un restaurante. Cada vértice interno hace una pregunta. Si se inicia en la raíz, se responde cada pregunta, y se sigue la

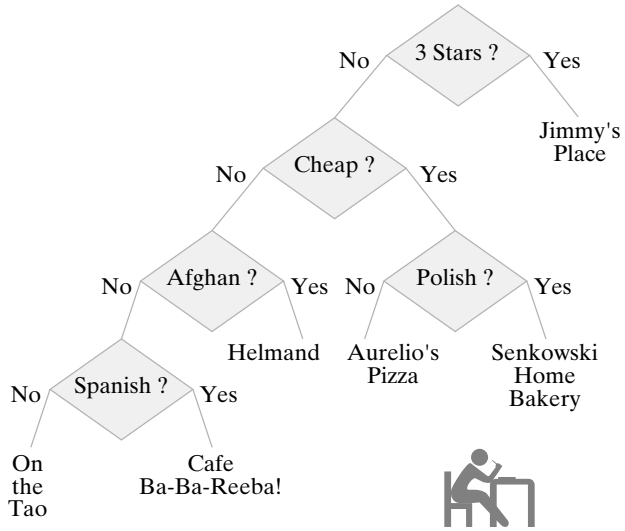


Figura 9.7.1 Un árbol de decisiones.

arista adecuada, en algún momento se llega a un vértice terminal que elige un restaurante. Un árbol de este tipo se llama **árbol de decisiones**. En esta sección se usan los árboles de decisión para especificar algoritmos y para obtener cotas inferiores sobre el tiempo del peor caso para ordenar así como par resolver ciertos juegos o acertijos. Se comienza con juegos de monedas.

Ejemplo 9.7.1 ▶

Acertijo de las cinco monedas

Cinco monedas tienen apariencia idéntica, pero una es más pesada o más ligera que las demás, que pesan todas lo mismo. El problema es identificar la moneda defectuosa y determinar si es más o menos pesada usando sólo una balanza (figura 9.7.2) que compara los pesos de dos conjuntos de monedas.

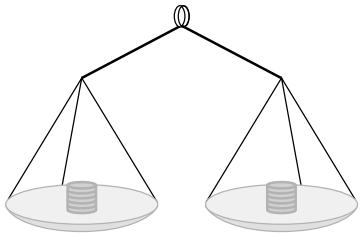


Figura 9.7.2 Balanza para comparar los pesos de las monedas.

La figura 9.7.3 proporciona un algoritmo para resolver el problema como un árbol de decisiones. Las monedas se etiquetan C_1, C_2, C_3, C_4, C_5. Como se ilustra, se comienza en la raíz y se coloca la moneda C_1 en el plato izquierdo y la moneda C_2 en el derecho. Una arista con la marca ↙ significa que el lado izquierdo de la balanza es más pesado que el derecho. De la misma manera, una arista con la marca ↘ significa que el lado derecho de la balanza es más pesado que el izquierdo, y una arista etiquetada — quiere decir que los dos lados pesan lo mismo. Por ejemplo, en la raíz, cuando se compara C_1 con C_2, si el lado izquierdo es más pesado que el derecho, se sabe que C_1 es la moneda pesada (H), o bien, que C_2 es la moneda ligera (L). En este caso, como se muestra en el árbol de decisiones, se compara C_1 con C_5 (que se sabe que es una buena moneda) y de inmediato se determina si la moneda defectuosa es C_1 o C_2 y si es más pesada o más ligera. Los vértices terminales dan la solución. Por ejemplo, cuando se compara C_1 con C_5 y la balanza se equilibra, se sigue la arista al vértice terminal con etiqueta C_2, L, que dice que la moneda defectuosa es C_2 y que es más ligera que las otras.

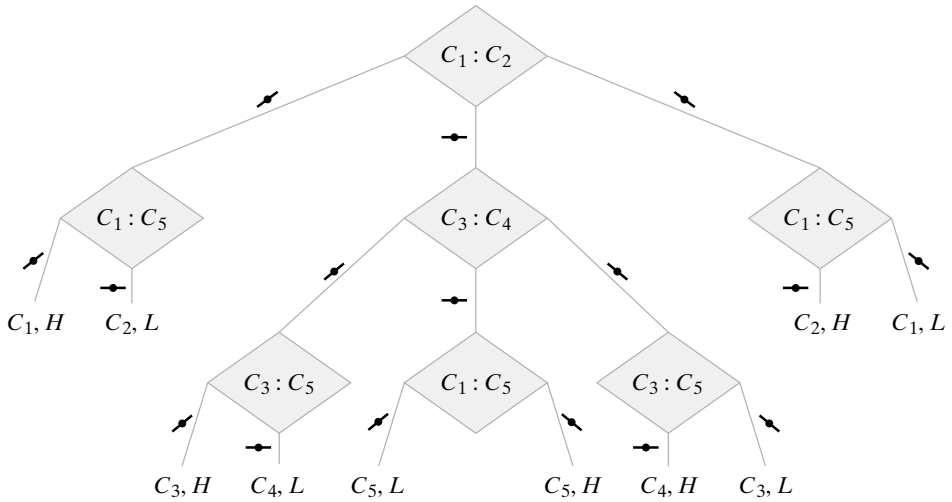


Figura 9.7.3 Algoritmo para resolver el acertijo de las cinco monedas. ◀

Si se define el tiempo en el peor caso para resolver el problema del peso de las monedas como el número de veces que se requiere pesar en el peor caso, es fácil determinar el tiempo en el peor caso a partir del árbol de decisiones; el tiempo en el peor caso es igual a la altura del árbol. Por ejemplo, la altura del árbol de decisiones de la figura 9.7.3 es 3, de manera que el tiempo en el peor caso para este algoritmo es igual a 3.

Se pueden usar árboles de decisiones para mostrar que el algoritmo dado en la figura 9.7.3 para resolver el problema de las cinco monedas es óptimo, es decir, que *ningún* algoritmo que resuelva este problema tiene un tiempo en el peor caso menor que 3.

Se da un argumento por contradicción para demostrar que ningún algoritmo que resuelve el problema de las cinco monedas tiene un tiempo en el peor caso menor que 3. Suponga que hay un algoritmo que resuelve el problema en el peor caso en dos o menos pesadas. El algoritmo puede describirse por un árbol de decisiones, y como el tiempo en el peor caso es 2 o menos, la altura del árbol es 2 o menos. Como cada vértice interno tiene cuando mucho tres hijos, este árbol puede tener cuando mucho nueve vértices terminales (figura 9.7.4). Ahora bien, los vértices terminales corresponden a los resultados posibles.

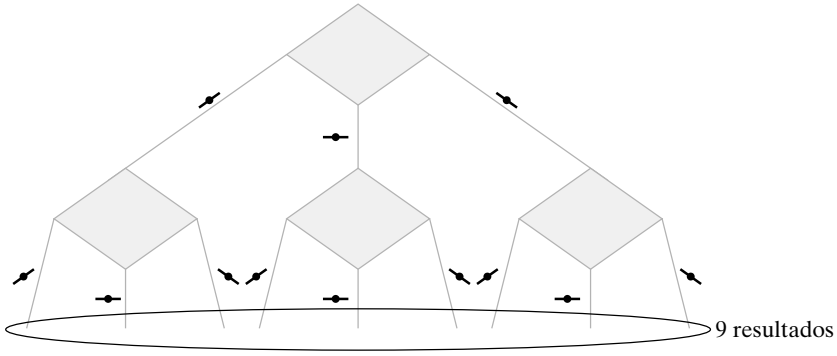


Figura 9.7.4 Algoritmo para el acertijo de las cinco monedas que requiere cuando mucho pesar 2 veces.

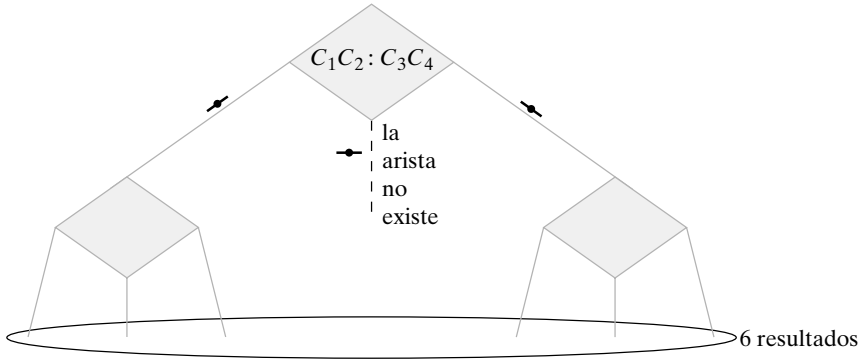


Figura 9.7.5 Algoritmo para el acertijo de cuatro monedas que comienza por comparar dos monedas con dos monedas.

Entonces, un árbol de decisión con altura 2 o menos explicará cuando mucho 9 resultados. Pero el acertijo de las cinco monedas tiene 10 resultados:

$$C_1, L, \quad C_1, H, \quad C_2, L, \quad C_2, H, \quad C_3, L,$$
$$C_3, H, \quad C_4, L, \quad C_4, H, \quad C_5, L, \quad C_5, H.$$

Ésta es una contradicción. Por lo tanto, ningún algoritmo que resuelve el acertijo de las cinco monedas tiene un tiempo en el peor caso menor que 3, y el algoritmo de la figura 9.7.3 es óptimo.

Se ha visto cómo se puede usar un árbol de decisiones para encontrar una cota inferior para el tiempo de resolución de un problema en el peor caso. Algunas veces, la cota inferior no se alcanza.

Considere el acertijo de cuatro monedas (todas las reglas son las mismas que con cinco monedas excepto que el número de monedas se redujo en uno). Como ahora hay 8 resultados en lugar de 10, se concluye que cualquier algoritmo que resuelva el problema de cuatro monedas requiere pesar al menos dos veces en el peor caso. (Esta vez *no se puede* concluir que se requiere pesar al menos 3 veces en el peor caso). Sin embargo, una inspección más detallada revela que, de hecho, se requiere pesar tres veces.

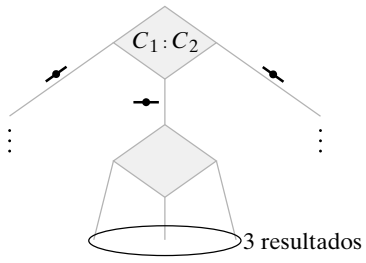


Figura 9.7.6 Un algoritmo para el acertijo de cuatro monedas que comienza comparando una moneda con otra.

La primera vez que se pesa se comparan dos monedas con otras dos. La figura 9.7.5 indica que, si se comienza por comparar dos monedas con otras dos, el árbol de decisiones explicará cuando mucho 6 resultados. Como existen 8 resultados, ningún algoritmo que comience comparando dos monedas contra dos resolverá el problema pesando dos veces o menos en el peor caso. De manera similar, la figura 9.7.6 muestra que si se comienza por comparar una moneda con otra y se equilibran, el árbol de decisión puede explicar sólo 3 resultados. Como son posibles 4 resultados después de identificar dos monedas buenas, ningún algoritmo que comience comparando una moneda con otra puede resolver el problema pesando dos veces o menos en el peor caso. Por lo tanto, cualquier algoritmo que resuelva el acertijo de cuatro monedas requiere pesar tres veces en el peor caso.

Si se modifica el acertijo de cuatro monedas requiriendo sólo identificar la moneda defectuosa (sin determinar si es más pesada o más ligera), es posible resolver el problema en dos pesadas en el peor caso (vea el ejercicio 1).

Ahora se estudiará el orden. Se pueden utilizar árboles de decisión para estimar el tiempo necesario para ordenar en el peor caso.

El problema de ordenar se describe con facilidad. Dados n artículos

$$x_1, \quad \ldots, \quad x_n,$$

arréglelos en orden no decreciente (o no creciente). Se restringe la atención a los algoritmos para ordenar que comparen dos elementos repetidas veces y, según el resultado de la comparación, se modifique la lista original.

Ejemplo 9.7.2 ▶

Un algoritmo para ordenar a_1, a_2, a_3 está dado por el árbol de decisiones de la figura 9.7.7. Cada arista se etiqueta con el arreglo de la lista basado en la respuesta a la pregunta en un vértice interno. Los vértices terminales dan el orden.

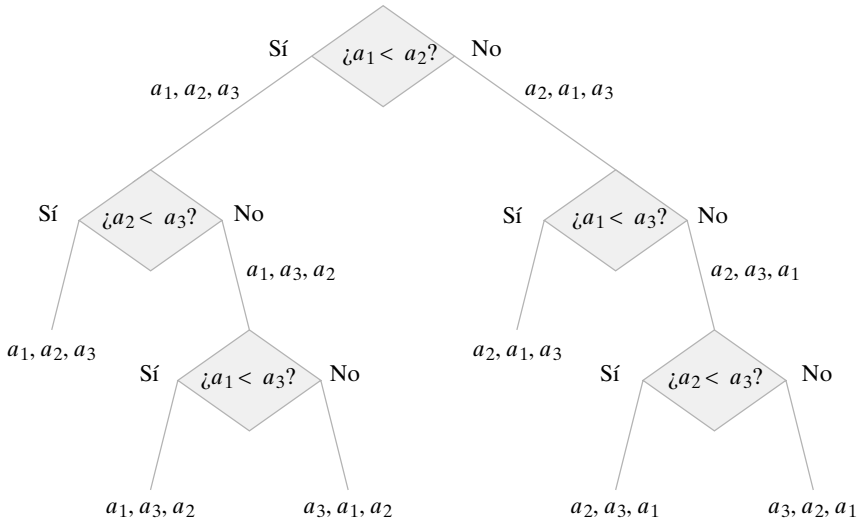


Figura 9.7.7 Un algoritmo para ordenar a_1, a_2, a_3.

Se definirá el tiempo en el peor caso para ordenar como un número de comparaciones en el peor caso. Igual que en el caso de los árboles de decisión que resuelven los acertijos de monedas, la altura del árbol de decisiones que resuelve el problema de ordenar es igual al tiempo en el peor caso. Por ejemplo, el tiempo en el peor caso para el algoritmo dado por el árbol de decisiones de la figura 9.7.7 es igual a 3. Se demuestra que este algoritmo es óptimo, es decir, que *ningún* algoritmo que ordene tres artículos tiene un tiempo en el peor caso menor que 3.

Se da un argumento por contradicción para demostrar que ningún algoritmo que ordene tres artículos tiene un tiempo en el peor caso menor que 3. Suponga que existe un algoritmo que ordena tres artículos en el peor caso en 2 comparaciones o menos. El algoritmo se puede describir por un árbol de decisiones y como el tiempo en el peor caso es 2 o menos, la altura del árbol es 2 o menos. Como cada vértice interno tiene a lo sumo 2 hijos, este árbol tendrá cuando mucho 4 vértices terminales (figura 9.7.8). Ahora, los vértices terminales corresponden a resultados posibles. Entonces un árbol de decisiones de altura 2 o menos explicará cuando mucho 4 resultados. Pero el problema de ordenar 3 artículos tie-

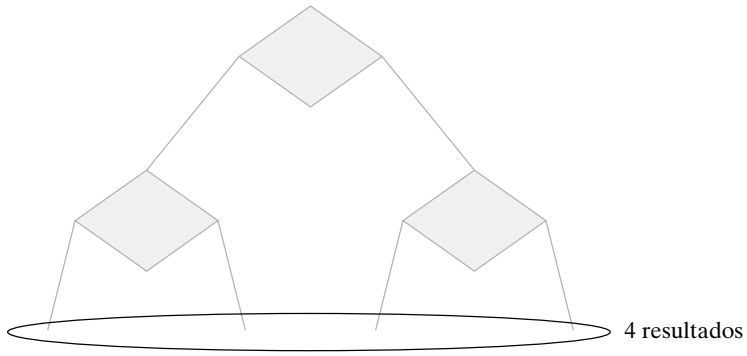


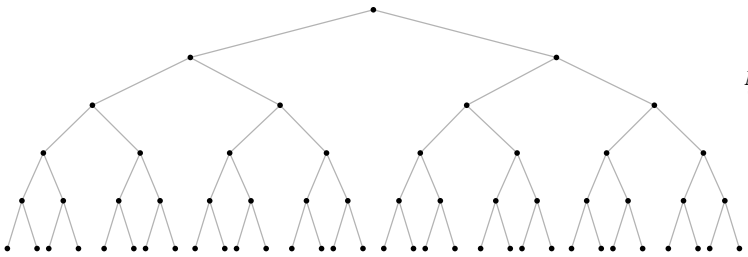
Figura 9.7.8 Un algoritmo para ordenar, que hace cuando mucho dos comparaciones.

ne 6 resultados posibles (cuando los artículos son distintos), lo que corresponde a $3! = 6$ maneras en que se pueden ordenar 3 artículos.

$$s_1, s_2, s_3, \quad s_1, s_3, s_2, \quad s_2, s_1, s_3, \quad s_2, s_3, s_1, \quad s_3, s_1, s_2, \quad s_3, s_2, s_1.$$

Esto es una contradicción. Por lo tanto, ningún algoritmo que ordene artículos tiene un tiempo en el peor caso menor que 3, y el algoritmo de la figura 9.7.7 es óptimo. ◀

Como $4! = 24$, existen 24 resultados posibles para el problema de ordenar cuatro artículos (cuando son diferentes). Para acomodar 24 vértices terminales debe tenerse un árbol por lo menos de altura 5 (vea la figura 9.7.9). Por lo tanto, cualquier algoritmo que ordene cuatro artículos requiere al menos cinco comparaciones en el peor caso. El ejercicio 9 pide un algoritmo que ordene cuatro artículos usando cinco comparaciones en el peor caso.



Nivel	*Número de vértices*
0	1
1	2
2	4
3	8
4	16
5	32

Figura 9.7.9 Nivel comparado con el máximo número de vértices en ese nivel en un árbol binario.

El método del ejemplo 9.7.2 resulta útil para dar una cota inferior al número de comparaciones requeridas en el peor caso para ordenar un número arbitrario de artículos.

Teorema 9.7.3

Si $f(n)$ es el número de comparaciones necesarias para ordenar n artículos en el peor caso por un algoritmo que ordena, entonces $f(n) = \Omega(n \lg n)$.

Demostración Sea T el árbol de decisiones que representa el algoritmo para una entrada de tamaño n y sea h la altura de T. Entonces, el algoritmo requiere h comparaciones en el peor caso, de manera que

$$h = f(n). \tag{9.7.1}$$

El árbol T tiene al menos $n!$ vértices terminales, de manera que por el Teorema 9.5.6,

$$\lg n! \leq h. \tag{9.7.2}$$

El ejemplo 4.3.9 muestra que $\lg n! = \Theta(n \lg n)$; así, para alguna constante positiva C,

$$Cn \lg n \leq \lg n! \tag{9.7.3}$$

para todos excepto un número finito n de enteros. Al combinar las expresiones (9.7.1) a (9.7.3), se obtiene

$$Cn \lg n \leq f(n)$$

para todos excepto un número finito n de enteros. Por lo tanto,

$$f(n) = \Omega(n \lg n).$$

El Teorema 7.3.10 establece que el merge sort (algoritmo 7.3.8) usa $\Theta(n \lg n)$ comparaciones en el peor caso y, por el Teorema 9.7.3, es óptimo. Se conocen muchos otros algoritmos para ordenar que también logran el número óptimo $\Theta(n \lg n)$ de comparaciones; uno de ellos, el orden de torneo, se describe en el ejercicio 12.

Sección de ejercicios de repaso

1. ¿Qué es un árbol de decisiones?

2. ¿Cómo se relaciona la altura de un árbol de decisiones que representa un algoritmo con el tiempo en el peor caso del algoritmo?

3. Use árboles de decisiones para explicar por qué ordenar en el peor caso requiere al menos $\Omega(n \lg n)$ comparaciones.

Ejercicios

1. Cuatro monedas son idénticas en apariencia, pero una está defectuosa y es más pesada o más ligera que las otras, que pesan lo mismo. Dibuje un árbol de decisiones para proporcionar un algoritmo que identifique, cuando mucho en dos pesadas, la moneda defectuosa (pero no necesariamente que determine si es más o menos pesada que las otras) usando sólo una balanza.

2. Demuestre que se requiere pesar al menos dos veces para resolver el problema del ejercicio 1.

3. Ocho monedas son idénticas en apariencia, pero una está defectuosa y es más pesada o más ligera que las otras, que pesan lo mismo. Dibuje un árbol de decisiones para proporcionar un algoritmo que identifique, cuando mucho en tres pesadas, la moneda defectuosa; y determine si es más pesada o más ligera que las otras usando sólo una balanza.

4. Doce monedas son idénticas en apariencia, pero una está defectuosa y es más pesada o más ligera que las otras, que pesan lo mismo. Dibuje un árbol de decisiones para proporcionar un algoritmo que identifique, cuando mucho en tres pesadas, la moneda defectuosa; y determine si es más pesada o más ligera que las otras usando sólo una balanza.

5. Diga qué está mal en el siguiente argumento que pretende demostrar que el acertijo de 12 monedas requiere al menos cuatro pesadas en el peor caso si se comienza por pesar cuatro monedas contra cuatro.

Si se pesan cuatro monedas contra cuatro y están balanceadas, debemos detectar la moneda defectuosa de las cuatro restantes. Pero el análisis en esta sección demostró que detectar una moneda defectuosa entre cuatro requiere pesar al menos tres veces en el peor caso. Por lo tanto, en el peor caso, si se comienza pesando cuatro monedas contra otras cuatro, el acertijo de 12 monedas requiere al menos cuatro pesadas.

★ **6.** Trece monedas son idénticas en apariencia, pero una está defectuosa y es más pesada o más ligera que las otras, que pesan lo mismo. ¿Cuántas veces se requiere pesar en el peor caso para encontrar la moneda defectuosa y determinar si es más o menos pesada que las otras usando sólo una balanza? Pruebe su respuesta.

7. Resuelva el ejercicio 6 para el acertijo de 14 monedas.

8. $(3^n - 3)/2$, $n \geq 2$ monedas son idénticas en apariencia, pero una está defectuosa y es más pesada o más ligera que las otras, que pesan lo mismo. [Kurosaka] proporcionó un algoritmo para encontrar la moneda defectuosa y determinar, en n pesadas en el peor caso, si es más o menos pesada que las otras usando sólo una balanza. Pruebe que no es posible encontrar la moneda e identificar como más pesada o más ligera en menos de n pesadas.

9. Dé un algoritmo que ordene cuatro artículos usando cinco comparaciones en el peor caso.

10. Utilice árboles de decisiones para encontrar una cota inferior para el número de comparaciones requeridas para ordenar cinco artículos en el peor caso. Dé un algoritmo que use este número de comparaciones para ordenar cinco artículos en el peor caso.

11. Utilice árboles de decisiones para encontrar una cota inferior para el número de comparaciones requeridas para ordenar seis artículos en el peor caso. Dé un algoritmo que emplee este número de comparaciones para ordenar seis artículos en el peor caso.

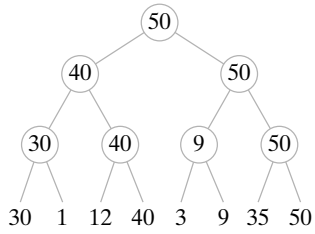
Los ejercicios 12 al 18 se refieren al orden de torneo.

Orden de torneo. Se tiene una secuencia

$$s_1, \quad \ldots, \quad s_{2^k}$$

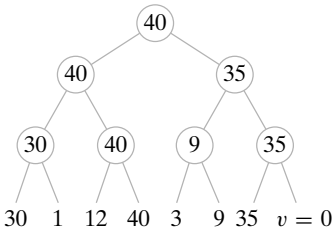
para ordenar en orden no decreciente.

Se construirá un árbol binario con vértices terminales etiquetados $s_1, \ldots, s_{2^k}$. Un ejemplo es el siguiente



Trabajando de derecha a izquierda, se crea un padre para cada par y se etiqueta con el máximo número de hijos. Se continúa de esta manera hasta llegar a la raíz. En este punto, se ha encontrado el valor más grande, m.

Para encontrar el segundo valor más grande, primero se escoge un valor v menor que todos los elementos de la sucesión. Se sustituye el vértice terminal w que contiene a m por v. Se etiquetan otra vez los vértices siguiendo la trayectoria de w a la raíz, como se indica. En este punto se ha encontrado el segundo valor más grande. Continúe hasta que la sucesión quede ordenada.



12. ¿Por qué es adecuado el nombre de "torneo"?

13. Dibuje los dos árboles que se crearían después del árbol adyacente cuando se aplica el orden de torneo.

14. ¿Cuántas comparaciones requiere el orden de torneo para encontrar el elemento más grande?

15. Demuestre que cualquier algoritmo que encuentra el valor más grande entre n elementos requiere al menos $n - 1$ comparaciones.

16. ¿Cuántas comparaciones requiere el orden de torneo para encontrar el segundo elemento más grande?

17. Escriba el orden de torneo como un algoritmo formal.

18. Demuestre que si n es una potencia de 2, el orden de torneo requiere $\Theta(n \lg n)$ comparaciones.

19. Dé un ejemplo de una situación real (como la de la figura 9.7.1) que se pueda modelar como un árbol de decisiones. Dibuje el árbol de decisión.

20. Dibuje un árbol de decisiones que se pueda usar para determinar quién debe entregar una declaración federal de impuestos.

21. Dibuje un árbol de decisiones que dé una estrategia razonable para jugar *blackjack* (vea, por ejemplo [Ainslie]).

9.8 ➜ Isomorfismos de árboles

En la sección 8.6 se determinó qué significa que dos gráficas sean isomorfas. (Tal vez quiera repasar la sección 8.6 antes de continuar). En esta sección se analizan los árboles isomorfos, los árboles con raíz isomorfos y los árboles binarios isomorfos.

WWW

El corolario 8.6.5 establece que dos gráficas simples G_1 y G_2 son isomorfas si y sólo si existe una función f uno a uno y sobre del conjunto de vértices de G_1 al conjunto de vértices de G_2 que preserva la relación de adyacencia en el sentido de que los vértices v_i y v_j son adyacentes en G_1 si y sólo si los vértices $f(v_i)$ y $f(v_j)$ son adyacentes en G_2. Como un árbol (libre) es una gráfica simple, los árboles T_1 y T_2 son isomorfos si y sólo si existe una función uno a uno y sobre del conjunto de vértices de T_1 al conjunto de vértices de T_2 que preserve la relación de adyacencia; es decir, los vértices v_i y v_j son adyacentes en T_1 si y sólo si los vértices $f(v_i)$ y $f(v_j)$ son adyacentes en T_2.

Ejemplo 9.8.1 ▶

La función f del conjunto de vértices del árbol T_1 que se ilustra en la figura 9.8.1 al conjunto de vértices del árbol T_2 que se aprecia en la figura 9.8.2 y que se define por

$$f(a) = 1, \quad f(b) = 3, \quad f(c) = 2, \quad f(d) = 4, \quad f(e) = 5$$

es una función uno a uno y sobre que preserva la relación de adyacencia. Así, los árboles T_1 y T_2 son isomorfos.

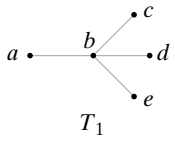


Figura 9.8.1 Un árbol.

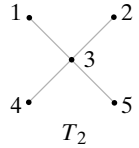


Figura 9.8.2 Un árbol isomorfo al árbol de la figura 9.8.1.

◀

Igual que en el caso de las gráficas, se demuestra que dos árboles no son isomorfos si se puede exhibir una invariante que los árboles no compartan.

Ejemplo 9.8.2 ▶

Los árboles T_1 y T_2 de la figura 9.8.3 no son isomorfos porque T_2 tiene un vértice (x) de grado 3 y T_1 no tiene un vértice de grado 3.

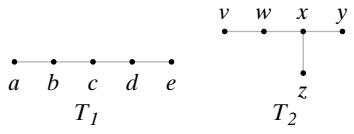


Figura 9.8.3 Árboles no isomorfos. T_2 tiene un vértice de grado 3, pero T_1 no lo tiene.

◀

Se puede ver que hay tres árboles no isomorfos con cinco vértices. Estos tres árboles no isomorfos se reproducen en las figuras 9.8.1 y 9.8.3.

Teorema 9.8.3 *Hay tres árboles isomorfos con cinco vértices.*

Demostración Se dará un argumento para demostrar que cualquier árbol con cinco vértices es isomorfo a uno de los árboles de las figuras 9.8.1 o 9.8.3.

Si T es un árbol con cinco vértices, por el Teorema 9.2.3, T tiene cuatro aristas. Si T tuviera un vértice v de grado mayor que 4, v sería incidente en más de cuatro aristas. Se deduce entonces que cada vértice en T tiene a lo sumo grado 4.

Primero se encontrarán todos los árboles no isomorfos con cinco vértices en los que el grado máximo de un vértice es 4. Después se encontrarán todos los árboles no isomorfos con cinco vértices en los que el grado máximo de vértice es 3, y así sucesivamente.

Sea T un árbol con cinco vértices y suponga que T tiene un vértice v de grado 4. Entonces hay cuatro aristas incidentes en v y, por el Teorema 9.2.3, éstas son todas las aristas. Se deduce que, en este caso, T es isomorfo al árbol de la figura 9.8.1.

Suponga que T es un árbol con cinco vértices y el grado máximo de vértice es 3. Sea v un vértice de grado 3. Entonces v incide en tres aristas como se observa en la figura 9.8.4. La cuarta arista no puede ser incidente en v ya que entonces v tendría grado 4. Así, la cuarta arista incide en uno de v_1, v_2 o v_3. Al agregar cualquier arista incidente en cualquiera de v_1, v_2 o v_3 se obtiene un árbol isomorfo al árbol T_2 de la figura 9.8.3.

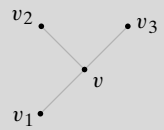


Figura 9.8.4 El vértice v tiene grado 3.

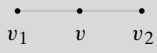


Figura 9.8.5 El vértice v tiene grado 2.

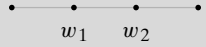


Figura 9.8.6 Se agrega una tercera arista a la gráfica de la figura 9.8.5.

Ahora suponga que T es un árbol con cinco vértices y que el grado máximo que ocurre en los vértices es 2. Sea v un vértice de grado 2. Entonces v incide en dos aristas, como se muestra en la figura 9.8.5. Una tercera arista no podría incidir en v; entonces debe incidir en v_1 o v_2. Agregar una tercera arista da la gráfica de la figura 9.8.6. Por la misma razón, la cuarta arista no puede incidir en uno de los vértices w_1 o w_2 de la figura 9.8.6. Agregar la última arista da un árbol isomorfo al árbol T_1 de la figura 9.8.3.

Como un árbol con cinco vértices debe tener un vértice de grado 2, se han encontrado todos los árboles no isomorfos con cinco vértices.

Para que dos árboles *con raíz* T_1 y T_2 sean isomorfos, debe existir una función f uno a uno y sobre de T_1 a T_2 que preserve la relación de adyacencia y que preserve la raíz. Esta última condición significa que f(raíz de T_1) = raíz de T_2. La definición formal es la siguiente.

Definición 9.8.4 ▶ Sea T_1 un árbol con raíz r_1 y sea T_2 un árbol con raíz r_2. Los árboles con raíz T_1 y T_2 son *isomorfos* si existe una función f uno a uno y sobre del conjunto de vértices de T_1 al conjunto de vértices de T_2 que satisface lo siguiente:

a) Los vértices v_i y v_j son adyacentes en T_1 si y sólo si los vértices $f(v_i)$ y $f(v_j)$ son adyacentes en T_2.

b) $f(r_1) = r_2$.

La función f recibe el nombre de *isomorfismo*. ◀

Ejemplo 9.8.5 ▶

Los árboles con raíz T_1 y T_2 en la figura 9.8.7 son isomorfos. Un isomorfismo es

$$f(v_1) = w_1, \quad f(v_2) = w_3, \quad f(v_3) = w_4, \quad f(v_4) = w_2,$$
$$f(v_5) = w_7, \quad f(v_6) = w_6, \quad f(v_7) = w_5.$$

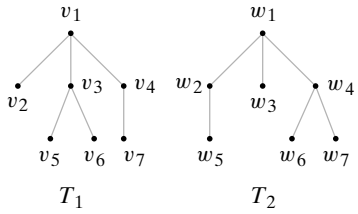


Figura 9.8.7 Árboles con raíz isomorfos. ◀

El isomorfismo del ejemplo 9.8.5 no es único. ¿Puede encontrar otro isomorfismo de los árboles con raíz de la figura 9.8.7?

Ejemplo 9.8.6 ▶

Los árboles con raíz T_1 y T_2 de la figura 9.8.8 no son isomorfos ya que la raíz de T_1 tiene grado 3, pero la raíz de T_2 tiene grado 2. Estos árboles son isomorfos como árboles *libres*. Cada uno es isomorfo al árbol T_2 de la figura 9.8.3.

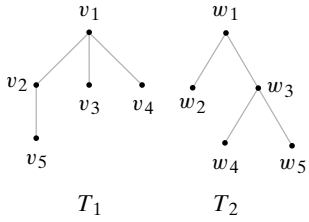


Figura 9.8.8 Árboles con raíz no isomorfos. (Los árboles son isomorfos como árboles *libres*). ◀

Con un argumento como el del Teorema 9.8.3, se puede demostrar que existen cuatro árboles de cuatro vértices con raíz no isomorfos.

Teorema 9.8.7

Existen cuatro árboles de cuatro vértices con raíz no isomorfos. Estos cuatro árboles con raíz se muestran en la figura 9.8.9.

Demostración Primero se encuentran todos los árboles de cuatro vértices con raíz en los que la raíz tiene grado 3; después se encuentran todos árboles con raíz no isomorfos con cuatro vértices en los que la raíz tiene grado 2; y así sucesivamente. Se observa que la raíz de un árbol de cuatro vértices con raíz no puede tener grado mayor que 3.

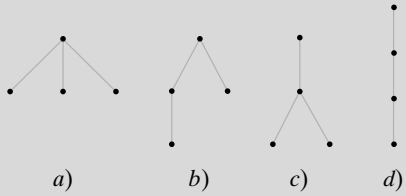


Figura 9.8.9 Los cuatro árboles de cuatro vértices con raíz no isomorfos.

Un árbol de cuatro vértices con raíz en el que la raíz tiene grado 3 debe ser isomorfo al árbol de la figura 9.8.9*a*).

Un árbol de cuatro vértices con raíz en el que la raíz tiene grado 2 debe ser isomorfo al árbol de la figura 9.8.9*b*).

Sea T un árbol con raíz con cuatro vértices en el que la raíz tiene grado 1. Entonces, la raíz incide en una arista. Las dos aristas restantes se pueden agregar de dos maneras [vea la figura 9.8.9*c*) y *d*)]. Por lo tanto, todos los árboles con raíz no isomorfos con cuatro vértices se muestran en la figura 9.8.9.

Los árboles binarios son un tipo especial de árboles con raíz; entonces un isomorfismo de árboles binarios debe preservar la relación de adyacencia y debe preservar las raíces. Sin embargo, en los árboles binarios un hijo se designa como hijo izquierdo o hijo derecho. Se requiere que un isomorfismo de árboles binarios preserve los hijos izquierdo y derecho. La siguiente es la definición formal.

Definición 9.8.8 ▶

Sea T_1 un árbol binario con raíz r_1 y sea T_2 un árbol binario con raíz r_2. Los árboles binarios T_1 y T_2 son *isomorfos* si existe una función f uno a uno y sobre del conjunto de vértices de T_1 al conjunto de vértices de T_2 que satisfagan lo siguiente:

a) Los vértices v_i y v_j son adyacentes en T_1 si y sólo si los vértices $f(v_i)$ y $f(v_j)$ son adyacentes en T_2.

b) $f(r_1) = r_2$.

c) v es un hijo izquierdo de w en T_1 si y sólo si $f(v)$ es un hijo izquierdo de $f(w)$ en T_2.

d) v es un hijo derecho de w en T_1 si y sólo si $f(v)$ es un hijo derecho de $f(w)$ en T_2.

Esta función f recibe el nombre de *isomorfismo*. ◀

Ejemplo 9.8.9 ▶

Los árboles binarios T_1 y T_2 en la figura 9.8.10 son isomorfos. El isomorfismo es $f(v_i) = w_i$ para $i = 1, \ldots, 4$.

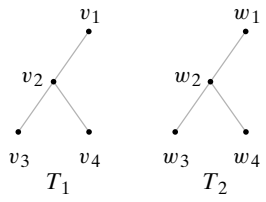


Figura 9.8.10 Árboles binarios isomorfos. ◀

Ejemplo 9.8.10 ▶

Los árboles binarios T_1 y T_2 de la figura 9.8.11 no son isomorfos. La raíz v_1 en T_1 tiene un hijo derecho, pero la raíz w_1 en T_2 no tiene un hijo derecho.

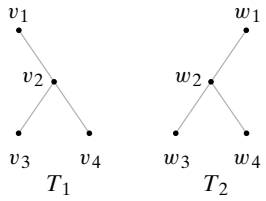


Figura 9.8.11 Árboles binarios no isomorfos. (Los árboles son isomorfos como árboles *con raíz* y como árboles *libres*).

◀

Los árboles T_1 y T_2 de la figura 9.8.11 *son* isomorfos como árboles con raíz y como árboles libres. Como árboles *con raíz*, cualquiera de los árboles de la figura 9.8.11 es isomorfo al árbol con raíz T de la figura 9.8.9*c*).

Con un argumento similar al de las demostraciones de los teoremas 9.8.3 y 9.8.7, se puede demostrar que hay cinco árboles binarios no isomorfos de tres vértices.

Teorema 9.8.11

Existen cinco árboles binarios isomorfos con tres vértices. Estos cinco árboles binarios se ilustran en la figura 9.8.12.

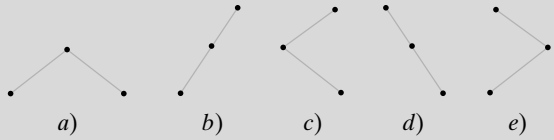


Figura 9.8.12 Los cinco árboles binarios isomorfos con tres vértices.

Demostración Primero se encuentran todos los árboles binarios con tres vértices en los que la raíz tiene grado 2. Después se encuentran todos los árboles binarios no isomorfos con tres vértices en los que la raíz tiene grado 1. Se observa que la raíz de cualquier árbol binario no puede tener grado mayor que 2.

Un árbol binario con tres vértices en el que la raíz tiene grado 2 debe ser un isomorfo al árbol de la figura 9.8.12*a*). En un árbol binario con tres vértices en el que la raíz tiene grado 1, la raíz tiene un hijo izquierdo y no uno derecho, o bien, tiene un hijo derecho y no uno izquierdo. Si la raíz tiene un hijo izquierdo, el hijo a su vez tiene un hijo izquierdo o un hijo derecho. Se obtienen los dos árboles binarios de la figura 9.8.12*b*) y *c*). De manera similar, si la raíz tiene un hijo derecho, el hijo a su vez tiene un hijo izquierdo o uno derecho. Esto da los dos árboles binarios de la figura 9.8.12*d*) y *e*). Por lo tanto, todos los árboles binarios no isomorfos con tres vértices se muestran en la figura 9.8.12.

Si S es un conjunto de árboles de un tipo específico (por ejemplo, S es un conjunto de árboles libres o S es un conjunto de árboles con raíz o S es un conjunto de árboles binarios) y se define una relación R en S mediante la regla T_1RT_2 si T_1 y T_2 son isomorfos, R es una relación de equivalencia. Cada clase de equivalencia consiste en un conjunto de árboles mutuamente isomorfos.

En el Teorema 9.8.3 se demostró que existen tres árboles libres no isomorfos que tienen cinco vértices. En el Teorema 9.8.7 se demostró que existen cuatro árboles con raíz no isomorfos que tienen cuatro vértices. En el Teorema 9.8.11 se demostró que existen cinco árboles binarios no isomorfos que tienen tres vértices. Podría preguntarse si existen fórmulas para el número de árboles no isomorfos con n vértices de un tipo específico. Existen fórmulas para el número de árboles libres de n vértices no isomorfos, para el número de árboles con raíz de n vértices no isomorfos y para el número de árboles binarios de n vértices no isomorfos. Las fórmulas para el número de árboles libres no isomorfos y para el número de árboles con raíz no isomorfos que tienen n vértices son bastante complicadas. Más aún, el desarrollo de estas fórmulas requiere técnicas más allá del alcance de las que se utilizan en este libro. Las fórmulas y demostraciones se encuentran en [Deo, sección 10-3]. Se derivará la fórmula para el número de árboles binarios con n vértices.

Teorema 9.8.12

Figura 9.8.13 Los dos árboles binarios no isomorfos con dos vértices.

Existen C_n árboles binarios no isomorfos con n vértices donde $C_n = C(2n, n)/(n + 1)$ es el n-ésimo número de Catalan.

Demostración Sea a_n el número de árboles binarios con n vértices. Por ejemplo $a_0 = 1$ ya que hay un árbol binario que no tiene vértices; $a_1 = 1$ ya que hay un árbol binario con un vértice; $a_2 = 2$ puesto que hay dos árboles binarios que tienen dos vértices (vea la figura 9.8.13); y $a_3 = 5$ porque hay cinco árboles binarios con tres vértices (vea la figura 9.8.12).

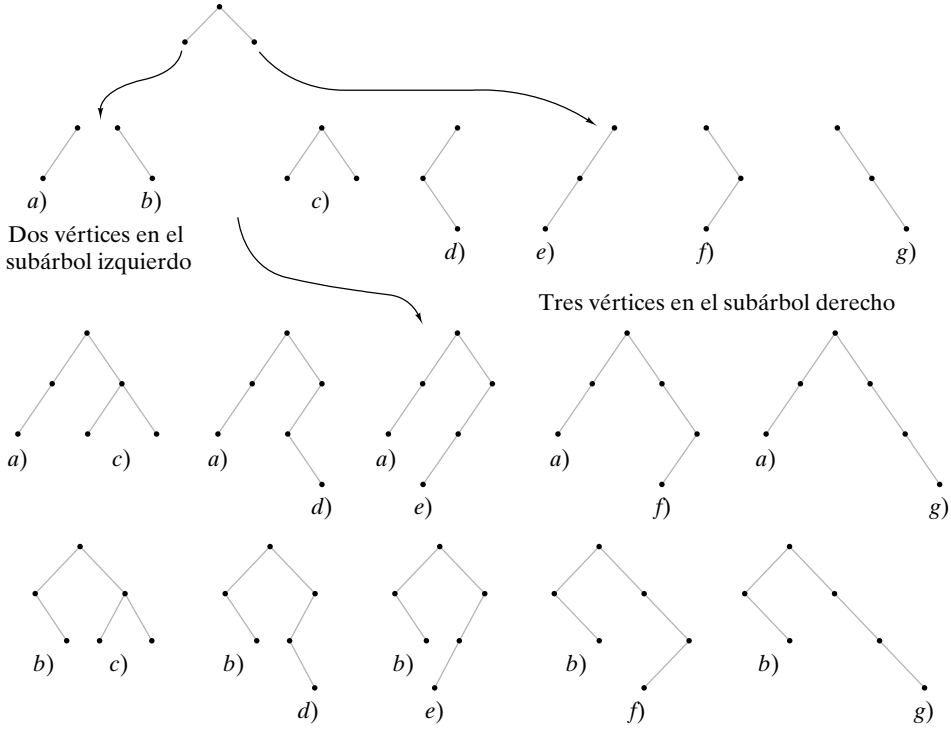


Figura 9.8.14 Prueba del Teorema 9.8.12 para el caso $n = 6$ vértices y $k = 2$ vértices en el subárbol izquierdo.

Se deriva una relación de recurrencia para la sucesión $a_0, a_1, \ldots$. Considere la construcción de un árbol binario con n vértices, $n > 0$. Un vértice debe ser la raíz. Como hay $n - 1$ vértices restantes, si el subárbol izquierdo tiene k vértices, el subárbol derecho debe tener $n - k - 1$ vértices. Se construye un árbol binario de n vértices cuyo subárbol izquierdo tiene k vértices y cuyo subárbol derecho tiene $n - k - 1$ vértices siguiendo un proceso de dos pasos: se construye el subárbol izquierdo; se construye el subárbol derecho. (La figura 9.8.14 muestra esta construcción para $n = 6$ y $k = 2$). Por el principio de la multiplicación, esta construcción puede realizarse de $a_k a_{n-k-1}$ maneras. Valores diferentes de k dan diferentes árboles binarios de n vértices, de manera que por el principio de la suma, el número total de árboles binarios de n vértices es

$$\sum_{k=0}^{n-1} a_k a_{n-k-1}.$$

Se obtiene la relación de recurrencia

$$a_n = \sum_{k=0}^{n-1} a_k a_{n-k-1}, \qquad n \geq 1.$$

Pero esta relación de recurrencia y la condición inicial $a_0 = 1$ definen la sucesión de números de Catalan (vea los ejemplos 6.2.23 y 7.1.7). Entonces a_n es igual al número de Catalan $C(2n, n)/(n + 1)$.

Al analizar los isomorfismos de gráficas en la sección 8.6, se resaltó que no se conoce un método eficiente para decidir si dos gráficas arbitrarias son isomorfas. La situación es diferente para los árboles. Es posible determinar en tiempo polinomial si dos árboles arbitrarios con isomorfos. Como caso especial, se da un algoritmo de tiempo lineal para determinar si dos árboles binarios T_1 y T_2 son isomorfos. El algoritmo se basa en el recorrido de preorden (vea la sección 9.6). Primero se verifica que T_1 y T_2 sean no vacíos, después de lo cual se verifica que los subárboles izquierdos de T_1 y T_2 sean isomorfos y que los subárboles derechos de T_1 y T_2 sean isomorfos.

Algoritmo 9.8.13

Prueba para verificar si dos árboles binarios son isomorfos

Entrada: Las raíces r_1 y r_2 de dos árboles binarios. (Si el primer árbol es vacío, r_1 tiene el valor especial de *nulo*. Si el segundo árbol es vacío, r_2 tiene el valor especial de *nulo*).

Salida: verdadero, si los árboles son isomorfos
falso, si los árboles no son isomorfos

```
árbol_bin_isom(r1, r2) {
1.    if (r1 == nulo ∧ r2 == nulo)
2.       return verdadero
      //ahora uno o ambos, r1 o r2, no es nulo
3.    if (r1 == nulo ∨ r2 == nulo)
4.       return falso
      //ahora ninguno de r1 o r2 es nulo
5.    hi_r1 = hijo izquierdo de r1
6.    hi_r2 = hijo izquierdo de r2
7.    hd_r1 = hijo derecho de r1
8.    hd_r2 = hijo derecho de r2
9.    return árbol_bin_isom(hi_r1, hi_r2) ∧ árbol_bin_isom(hd_r1, hd_r2)
}
```

Como una medida del tiempo que requiere el algoritmo 9.8.13, se cuenta el número de comparaciones con *nulo* en las líneas 1 y 3. Se demostrará que el algoritmo 9.8.13 es de tiempo lineal en el peor caso.

Teorema 9.8.14

El tiempo en el peor caso para el algoritmo 9.8.13 es $\Theta(n)$, *donde n es el número total de vértices en los dos árboles*.

Demostración Sea a_n el número de comparaciones con *nulo* que requiere el algoritmo 9.8.13 en el peor caso, donde n es el número total de vértices en los árboles de entrada. Se usa inducción matemática para probar que

$$a_n \leq 3n + 2 \qquad \text{para } n \geq 0.$$

Paso base ($n = 0$)

Si $n = 0$, los árboles de entrada al algoritmo 9.8.13 son vacíos ambos. En este caso, hay dos comparaciones con *nulo* en la línea 1, después de lo cual el procedimiento regresa. Entonces $a_0 = 2$ y la desigualdad se cumple cuando $n = 0$.

Paso inductivo

Suponga que

$$a_k \leq 3k + 2$$

cuando $k < n$. Debe demostrarse que

$$a_n \leq 3n + 2.$$

Primero se encuentra una cota superior para el número de comparaciones en el peor caso cuando el número total de vértices en los árboles de entrada al procedimiento es $n > 0$ y ninguno es vacío. En este caso, se hacen cuatro comparaciones en las líneas 1 y 3. Sea L la suma de los números de vértices en los dos subárboles izquierdos de los árboles de entrada y sea R la suma de los números de vértices en los dos subárboles derechos de los árboles de entrada. Entonces, en la línea 9 hay cuando mucho

$a_L + a_R$ comparaciones adicionales. Por lo tanto, en el peor caso se requieren cuando mucho $4 + a_L + a_R$ comparaciones. Por la suposición de inducción,

$$a_L \leq 3L + 2 \qquad \text{y} \qquad a_R \leq 3R + 2. \tag{9.8.1}$$

Ahora

$$2 + L + R = n \tag{9.8.2}$$

porque los vértices comprenden las dos raíces, los vértices en los subárboles izquierdos y los vértices en los subárboles derechos. Al combinar (9.8.1) y (9.8.2), se obtiene

$$4 + a_L + a_R \leq 4 + (3L + 2) + (3R + 2) = 3(2 + L + R) + 2 = 3n + 2.$$

Si cualquiera de los árboles es vacío, se requieren cuatro comparaciones en las líneas 1 y 3, después de lo cual el procedimiento regresa. Entonces, ya sea que uno de los árboles sea vacío o no, cuando mucho se requieren $3n + 2$ comparaciones en el peor caso. Por lo tanto,

$$a_n \leq 3n + 2,$$

y esto completa el paso inductivo. Se concluye que el tiempo en el peor caso del algoritmo 9.8.13 es $O(n)$.

Si n es par, digamos $n = 2k$, se puede usar inducción para demostrar (vea el ejercicio 24) que cuando dos árboles binarios isomorfos con k vértices se introducen al algoritmo 9.8.13, el número de comparaciones es igual a $3n + 2$. Usando este resultado, se puede demostrar (vea el ejercicio 25) que si n es impar, digamos $n = 2k + 1$, cuando los dos árboles binarios mostrados en la figura 9.8.15 se introducen al algoritmo 9.8.13, el número de comparaciones es igual a $3n + 1$. Así, el tiempo en el peor caso para el algoritmo 9.8.13 es $\Omega(n)$.

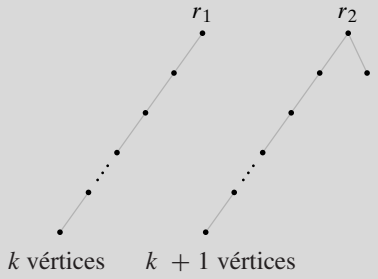


Figura 9.8.15 Dos árboles binarios que dan un tiempo en el peor caso de $3n + 1$ para el algoritmo 9.8.13 cuando $n = 2k + 1$ es impar.

Como el tiempo en el peor caso es $O(n)$ y $\Omega(n)$, el tiempo en el peor caso para el algoritmo 9.8.13 es $\Theta(n)$.

[Aho] da un algoritmo cuyo tiempo en el peor caso es lineal respecto al número de vértices, que determina si dos árboles con raíz arbitrarios (no necesariamente binarios) son isomorfos.

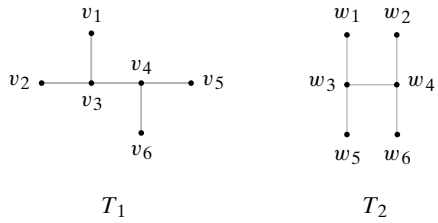
Sección de ejercicios de repaso

1. ¿Qué significa que dos árboles libres sean isomorfos?
2. ¿Qué significa que dos árboles con raíz sean isomorfos?
3. ¿Qué significa que dos árboles binarios sean isomorfos?
4. ¿Cuántos árboles binarios isomorfos de n vértices hay?
5. Describa un algoritmo en tiempo lineal para probar si dos árboles binarios son isomorfos.

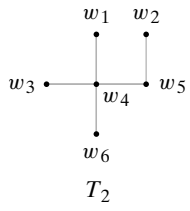
Ejercicios

En los ejercicios 1 al 6, determine si cada par de árboles libres es isomorfo. Si lo es, especifique un isomorfismo. Si el par no es isomorfo, dé una invariante que un árbol satisface y el otro no.

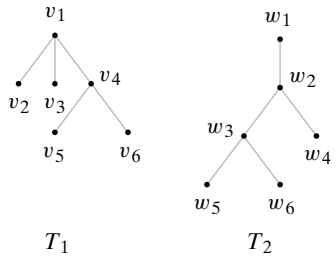
1.



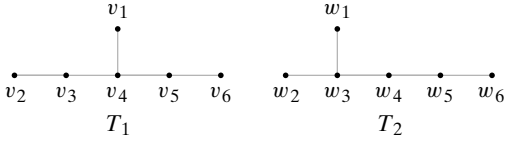
2. T_1 como en el ejercicio 1



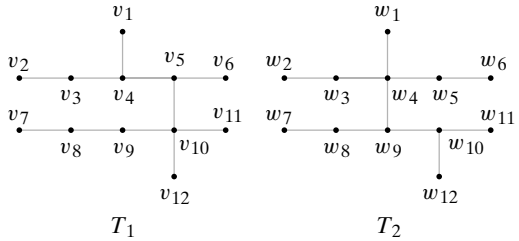
3.



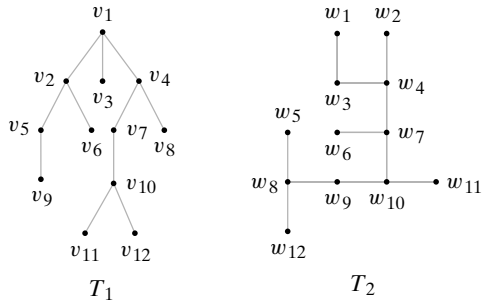
4.



5.

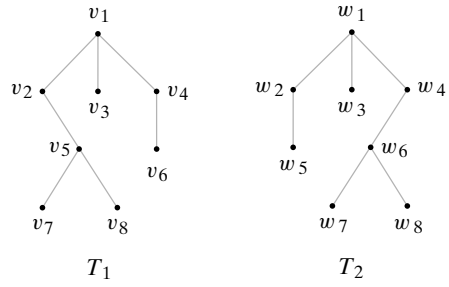


6.



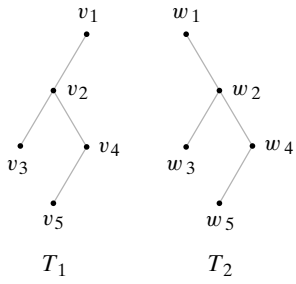
En los ejercicios 7 al 9, determine si cada par de árboles con raíz es isomorfo. Si lo es, especifique un isomorfismo. Si el par no es isomorfo, dé una invariante que un árbol satisfaga y el otro no. Además, determine si los árboles son isomorfos como árboles libres.

7.



8. T_1 y T_2 como en el ejercicio 3

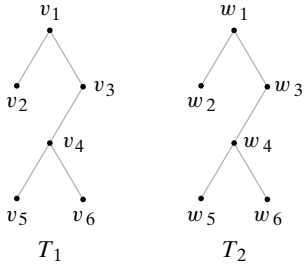
9.



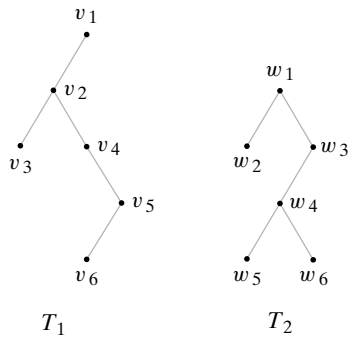
En los ejercicios 10 al 12, determine si cada par de árboles binarios es isomorfo. Si lo es, especifique un isomorfismo. Si el par no es isomorfo, dé una invariante que un árbol satisfaga y el otro no. Además, determine si los árboles son isomorfos como árboles libres o como árboles con raíz.

10. T_1 y T_2 como en el ejercicio 9

11.



12.



13. Dibuje todos los árboles libres no isomorfos de 3 vértices.

14. Dibuje todos los árboles libres no isomorfos de 4 vértices.

15. Dibuje todos los árboles libres no isomorfos de 6 vértices.

16. Dibuje todos los árboles con raíz no isomorfos de 3 vértices.

17. Dibuje todos los árboles con raíz no isomorfos de 5 vértices.

18. Dibuje todos los árboles binarios no isomorfos de 2 vértices.

19. Dibuje todos los árboles binarios no isomorfos de 4 vértices.

20. Dibuje todos los árboles binarios completos no isomorfos de 7 vértices. (Un árbol binario completo es un árbol binario en el que cada vértice interno tiene dos hijos).

21. Dibuje todos los árboles binarios completos no isomorfos de 9 vértices.

22. Encuentre una fórmula para el número de árboles completos no isomorfos de n vértices.

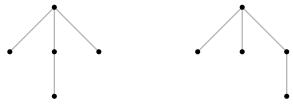
23. Encuentre todos los árboles de expansión (como árboles libres, no como árboles con raíz) no isomorfos par cada gráfica en los ejercicios 7 al 9, sección 9.3.

24. Utilice inducción para demostrar que cuando dos árboles binarios isomorfos de k vértices se introducen al algoritmo 9.8.13, el número de comparaciones con *nulo* es igual a $6k + 2$.

25. Demuestre que cuando los dos árboles binarios que aparecen en la figura 9.8.15 son la entrada del algoritmo 9.8.13, el número de comparaciones con *nulo* es igual a $6k + 4$.

26. Escriba un algoritmo para generar un árbol binario aleatorio de n vértices.

27. Un *árbol ordenado* es un árbol que toma en cuenta el orden de los hijos. Por ejemplo, los árboles ordenados



son *no* isomorfos. Demuestre que el número de árboles ordenados no isomorfos con n aristas es igual a C_n, el n-ésimo número de Catalan. *Sugerencia:* Considere un recorrido de preorden de árbol ordenado en el que 1 significa abajo y 0 significa arriba.

28. [*Proyecto*] Informe acerca de las fórmulas para el número de árboles libres no isomorfos y para el número de árboles con raíz no isomorfos de n vértices (vea [Deo]).

9.9 ➜ Árboles de juegos†

WWW

Los árboles son útiles en el análisis de juegos como gato, ajedrez y damas, donde los jugadores alternan movimientos. En esta sección se explica cómo se utilizan los árboles para desarrollar estrategias de juego. Este tipo de enfoque se emplea en el desarrollo de muchos programas de computadora que permiten a las personas jugar contra las computadoras o incluso una computadora contra otra.

Como ejemplo del enfoque general, considere una versión de juego de nim. Al inicio hay n pilas, cada una con un número de fichas idénticas. Los jugadores alternan movimientos. Un movimiento consiste en eliminar una o más fichas de cualquier pila. El jugador que elimina la última ficha pierde. Como caso específico, considere una distribución inicial con dos pilas: una con tres fichas y la otra con dos. Todas las secuencias de movimientos posibles se pueden listar en un **árbol del juego** (vea la figura 9.9.1). El primer jugador se representa por un cuadro y el segundo por un círculo. Cada vértice muestra una posición particular en el juego. En nuestro juego, la posición inicial se muestra como $\binom{3}{2}$. Una trayectoria representa una secuencia de movimientos. Si una posición se muestra en un cuadro, se trata del movimiento del primer jugador; si se indica en un círculo, corresponde a un movimiento del segundo jugador. Un vértice terminal representa el final de juego. En nim, si el vértice terminal es un círculo, el primer jugador eliminó la última ficha y perdió el juego. Si el vértice terminal es un cuadro, el segundo jugador pierde.

El análisis comienza con los vértices terminales. Se etiqueta cada vértice terminal con el valor de la posición del primer jugador. Si el vértice terminal es un círculo, como el primer jugador pierde, esta posición no vale nada para el primer jugador y se le asigna el valor 0 (vea la figura 9.9.2). Si el vértice terminal es un cuadro, como el primer jugador gana, esta posición es valiosa para el primer jugador y se etiqueta con un valor mayor que 0, por ejemplo, 1 (vea la figura 9.9.2). En este momento, todos los vértices terminales tienen valores asignados.

Ahora, considere el problema de asignar valores a los vértices internos. Suponga, por ejemplo, que se tiene un cuadro interno, cuyos hijos, en su totalidad, tienen un valor asignado.

† Esta sección se puede omitir sin pérdida de continuidad.

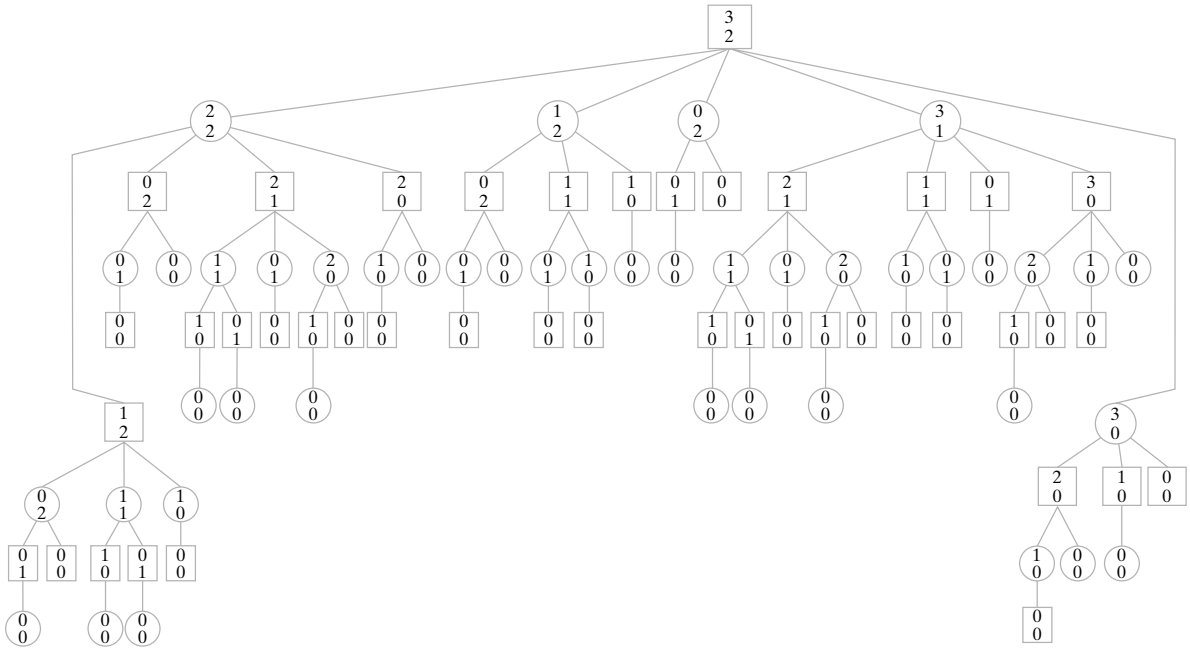


Figura 9.9.1 Un árbol de juego para nim. La distribución inicial es de dos pilas con tres y dos fichas, respectivamente.

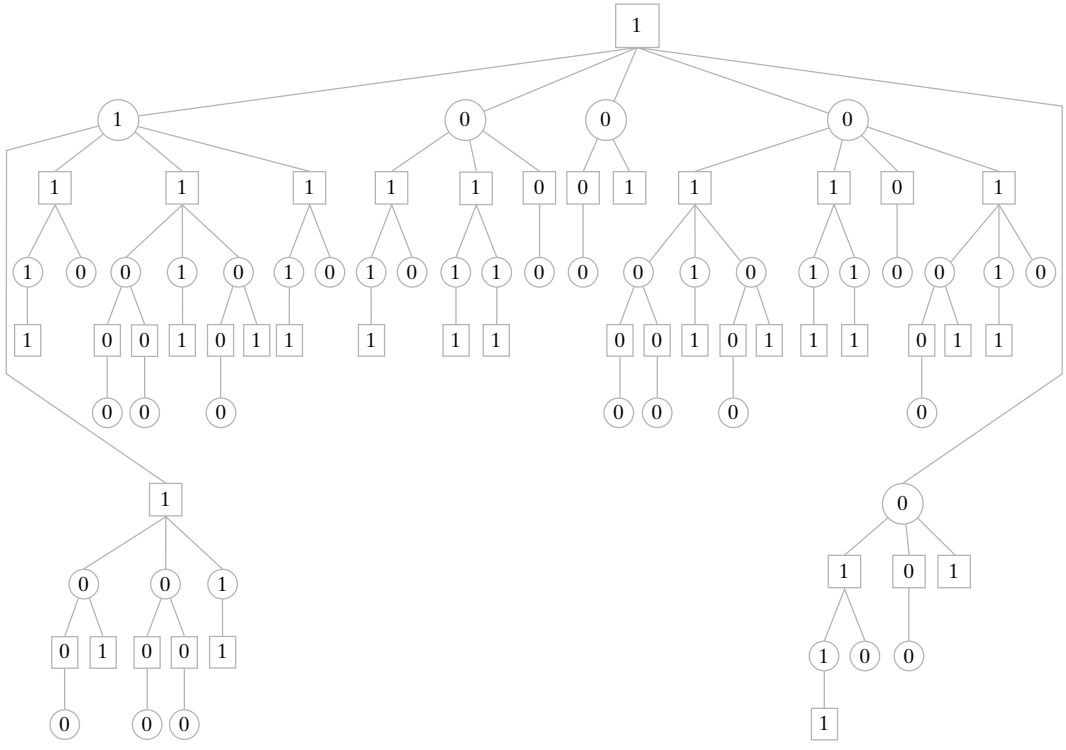


Figura 9.9.2 El árbol del juego de la figura 9.9.1 que muestra los valores de todos los vértices.

Por ejemplo, si se tiene la situación mostrada en la figura 9.9.3, el primer jugador (cuadro) debe moverse a la posición representada por el vértice *B*, ya que esta posición es la más valiosa. En otras palabras, el cuadro se mueve a la posición representada por un hijo con el máximo valor. Se asigna este valor máximo al vértice de cuadro.

Considere la situación desde el punto de vista del segundo jugador (círculo). Suponga que se presenta la situación descrita en la figura 9.9.4. El círculo debe moverse a la po-

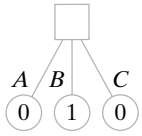


Figura 9.9.3 El primer jugador (cuadro) debe moverse a la posición *B* porque es la más valiosa. Este valor máximo (1) se asigna al cuadro.

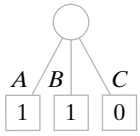


Figura 9.9.4 El segundo jugador (círculo) debe moverse a la posición *C* ya que es la menos valiosa (para el cuadro). Este valor mínimo (0) se asigna al círculo.

sición representada por el vértice *C*, ya que esta posición es la menos valiosa para el cuadro y por lo tanto la más valiosa para el círculo. En otras palabras, el círculo se mueve a una posición representada por un hijo con el mínimo valor. Se asigna este valor mínimo al vértice de círculo. El proceso por el que el círculo busca el mínimo de sus hijos y el cuadro busca el máximo de sus hijos se llama **procedimiento minimax**.

Trabajando hacia arriba desde los vértices terminales y usando el procedimiento minimax, es posible asignar valores a todos los vértices en el árbol del juego (vea la figura 9.9.2). Estos números representan el valor del juego, en cualquier posición, para el primer jugador. Observe que la raíz en la figura 9.9.2, que representa la posición original, tiene un valor de 1. Esto significa que el primer jugador siempre puede ganar usando la estrategia óptima. Esta estrategia óptima está contenida en el árbol del juego: el primer jugador siempre se mueve a una posición que maximice el valor de los hijos. No importa lo que el segundo jugador haga, el primer jugador siempre se podrá mover a un vértice con valor 1. Al final, se llega a un vértice terminal con valor 1 donde el primer jugador gana el juego.

Muchos juegos interesantes, como el ajedrez, tienen árboles de juego tan grandes que no es factible usar una computadora para generar el árbol completo. De cualquier forma, el concepto de un árbol de juego es siempre útil para analizar tales juegos.

Al utilizar un árbol de juego debe emplearse una búsqueda a profundidad. Si el árbol de juego es tan largo que no es factible llegar a un vértice terminal, la búsqueda se limita al nivel que se logre. Se dice que es una **búsqueda de nivel *n*** si se limita a *n* niveles hacia abajo del vértice dado. Como los vértices en el nivel más bajo quizá no sean vértices terminales, debe encontrarse algún método para asignarles un valor. Aquí es donde se utilizan las características específicas del juego. Se construye una **función de evaluación** *E* que asigna a cada posición *P* posible del juego el valor *E*(*P*) de la posición para el primer jugador. Después de que se asignan valores a los vértices en el nivel inferior usando la función *E*, es conveniente aplicar el procedimiento minimax para generar los valores de los otros vértices. Este concepto se ilustra con un ejemplo.

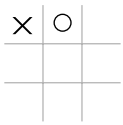
Ejemplo 9.9.1 ▶

Aplique el procedimiento minimax para encontrar el valor de la raíz en un juego de gato usando una búsqueda de dos niveles minimax a profundidad. Use la función de evaluación *E*, que asigna una posición al valor

$$N\text{X} - N\text{O}$$

donde *N*X (respectivamente, *N*O) es el número de renglones, columnas o diagonales que contienen X (respectivamente, *O*) que X (respectivamente, O) pueda completar. Por ejemplo, la posición *P* de la figura 9.9.5 tiene $N\text{X} = 2$, ya que X puede completar la columna o la diagonal, y $N\text{O} = 1$, porque O puede completar sólo una columna. Por lo tanto,

$$E(P) = 2 - 1 = 1.$$



P

Figura 9.9.5 El valor de la posición *P* es $E(P) = N\text{X} - N\text{O} = 2 - 1 = 1$.

En la figura 9.9.6, se dibujó el árbol de juego para gato hasta el nivel 2. Se omitieron las posiciones simétricas. Primero se asigna a los vértices del nivel 2 los valores dados por *E* (vea la figura 9.9.7). Después, se calcula el valor del círculo minimizando sobre los hijos. Por último, se calcula el valor de la raíz maximizando sobre los hijos. Empleando este análisis, el primer movimiento del primer jugador sería el cuadro del centro.

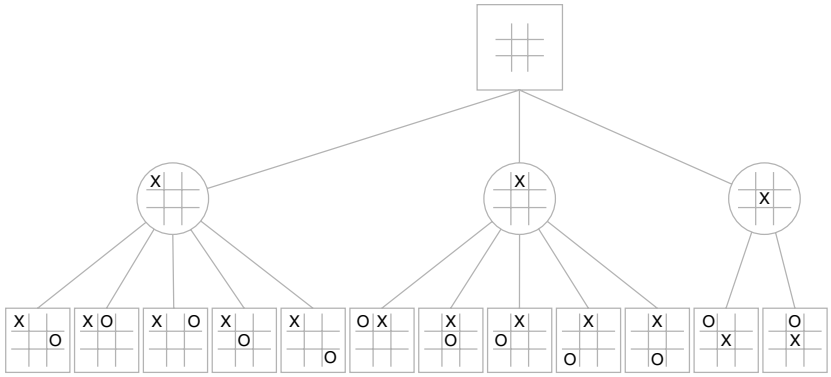


Figura 9.9.6 El árbol de juego para el gato hasta el nivel 2 con las posiciones simétricas omitidas.

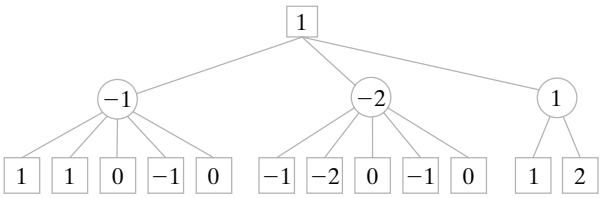


Figura 9.9.7 El árbol del juego de la figura 9.9.6 mostrando los valores de todos los vértices.

◀

La evaluación de un árbol de juego, o incluso una parte de él, puede ser tardada, de manera que cualquier técnica que reduzca el esfuerzo es bienvenida. La técnica más general se llama **recorte alfa-beta**. En general, el recorte alfa-beta permite evadir muchos vértices en un árbol de juego y de todas maneras encontrar el valor de un vértice. El valor obtenido es el mismo que si se hubieran evaluado todos los vértices.

Como ejemplo, considere el árbol de juego de la figura 9.9.8. Suponga que se quiere evaluar el vértice A usando una búsqueda a profundidad, de dos niveles. Se evalúan los hijos de izquierda a derecha. Se comienza abajo a la izquierda con los vértices E, F y G. Los valores mostrados se obtienen de una función de evaluación. El vértice B es 2, el mínimo de sus hijos. En este punto, se sabe que el valor x de A debe ser al menos 2, ya que el valor de A es el máximo de sus hijos; es decir,

$$x \geq 2 \qquad \textbf{(9.9.1)}$$

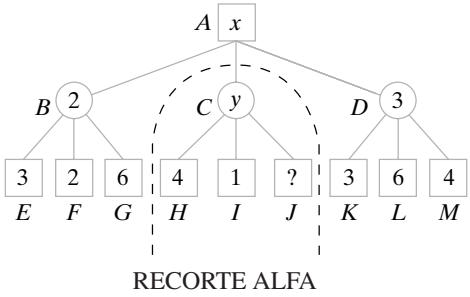


Figura 9.9.8 Evaluación del vértice A usando una búsqueda de dos niveles a profundidad con recorte alfa-beta. Un recorte alfa ocurre en el vértice C cuando se evalúa el vértice I, ya que el valor de I (1) es menor o igual que la cota inferior estimada actual (2) para el vértice A.

Esta cota inferior para A se llama **valor alfa** de A. Los siguientes vértices para evaluación son H, I y J. Cuando la evaluación de I es 1, se sabe que el valor y de C no puede exceder 1, ya que el valor de C es el mínimo de sus hijos; es decir,

$$y \leq 1 \qquad \textbf{(9.9.2)}$$

Se deduce de (9.9.1) y (9.9.2) que cualquiera que sea el valor de y, no afectará el valor de x; no necesitamos preocuparnos más por el subárbol con raíz en el vértice C. Se dice que ocurre un **recorte alfa**. Luego se evalúan los hijos de D y después el propio D. Por último, se encuentra que el valor de A es 3.

Para resumir, un recorte alfa ocurre en un vértice de cuadro v cuando un nieto w de v tiene un valor menor o igual al valor alfa de v. El subárbol cuya raíz es padre de w se puede eliminar (recortar). Este recorte no afectará el valor de v. Un valor alfa de un vértice v es sólo una cota inferior para el valor de v. El valor alfa de un vértice depende del estado actual de la búsqueda y cambia cuando la búsqueda avanza.

De manera similar, un **valor beta** de un vértice de círculo es una cota superior para v. Un **recorte beta** ocurre en un vértice de círculo cuando un nieto w de v tiene un valor mayor o igual que el valor beta de v. El subárbol cuya raíz es padre de w puede recortarse. Esta eliminación no afectará el valor de v. Un valor beta para el vértice v es sólo una cota superior para el valor de v. El valor beta de un vértice depende del estado actual de la búsqueda y cambia cuando la búsqueda avanza.

Ejemplo 9.9.2 ▶

Evalúe la raíz del árbol de la figura 9.9.9 usando la búsqueda a profundidad con recorte alfa-beta. Suponga que los hijos se evalúan de izquierda a derecha. Para cada vértice cuyo valor se calcula, escriba el valor en el vértice. Marque la raíz de cada subárbol recortado. El valor de cada vértice terminal está escrito bajo el vértice.

Se comienza por evaluar los vértices A, B, C y D (vea la figura 9.9.10). Después, se encuentra que el valor de E es 6. Esto da como resultado un valor beta de 6 para F. Luego se evalúa el vértice G. Como este valor es 8 y 8 excede el valor beta de F, se obtiene un

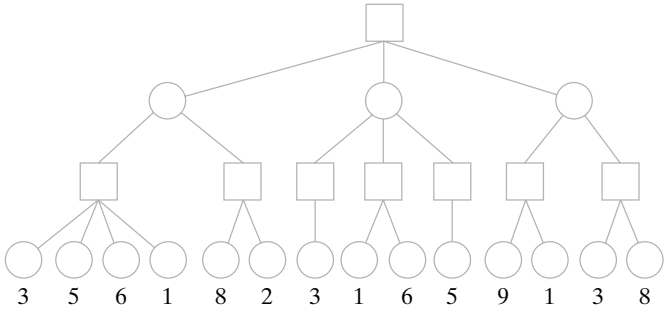


Figura 9.9.9 Árbol de juego para el ejemplo 9.9.2.

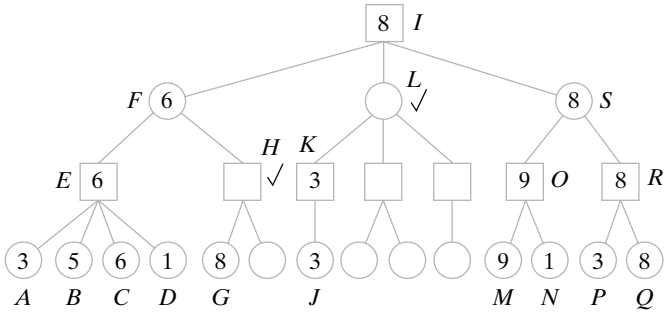


Figura 9.9.10 Para evaluar la raíz del árbol de juego de la figura 9.9.9 se usa la búsqueda a profundidad con recorte alfa-beta. Los vértices marcados son las raíces de los subárboles que se recortan. Los valores de los vértices evaluados se escriben dentro de los vértices.

recorte beta y se recorta el subárbol con raíz *H*. El valor de *F* es 6. Esto da como resultado un valor alfa de 6 para *I*. Después se evalúan los vértices *J* y *K*. Como el valor 3 de *K* es menor que el valor alfa 6 de *I*, ocurre un recorte alfa y el subárbol con raíz en *L* se elimina. Luego se evalúan *M*, *N*, *O*, *P*, *Q*, *R* y *S*. No es posible eliminar más vértices. Por último, se determina que la raíz *I* tiene un valor de 8. ◀

Se ha demostrado (vea [Pearl]) que para árboles de juegos en los que todo padre tiene *n* hijos y en los que los valores terminales tienen un orden aleatorio, para una cantidad dada de tiempo, el procedimiento alfa-beta permite una búsqueda a profundidad $4/3$ mayor que el procedimiento minimax puro, que evalúa todos los vértices. [Pearl] también demostró que para estos árboles de juegos, el procedimiento alfa-beta es óptimo.

Se han combinado otras técnicas con el recorte alfa-beta para facilitar la búsqueda en un árbol de juego. Una idea consiste en ordenar los hijos de los vértices que se van evaluar de manera que los movimientos más prometedores se examinen primero (vea los ejercicios 23 al 26). Otra idea es permitir la búsqueda a profundidad de una variable en donde la búsqueda explora hacia atrás cuando llega a una posición prometedora según la medida de alguna función.

Algunos programas de juegos han tenido un éxito increíble. Los mejores programas de ajedrez, backgamon y damas juegan a un nivel comparable al de los mejores jugadores humanos. El campeón mundial de damaz es un programa llamado Chinook desarrollado
WWW por un equipo de la Universidad de Alberta. En 1997 el programa de ajedrez de IBM, Deep Blue, venció a Garry Kasparov, que había sido campeón mundial desde 1985, en un encuentro de seis juegos. Deep Blue ganó dos juegos, empató tres y perdió uno.

Sección de ejercicios de repaso

1. ¿Qué es un árbol de juego?

2. ¿Qué es el procedimiento minimax?

3. ¿Qué es una búsqueda de nivel *n*?

4. ¿Qué es una función de evaluación?

5. Explique cómo funciona el recorte alfa-beta.

6. ¿Qué es un valor alfa?

7. ¿Qué es un recorte alfa?

8. ¿Qué es un valor beta?

9. ¿Qué es un recorte beta?

Ejercicios

1. Dibuje un árbol de juego completo para una versión de nim en la que la posición inicial consiste en una pila de seis fichas y un turno consiste en tomar una, dos o tres. Asigne valores a todos los vértices de manera que el árbol obtenido sea análogo al de la figura 9.9.2. Suponga que el último jugador en tomar una ficha pierde. ¿Ganará siempre el primero o el segundo jugador siguiendo una estrategia óptima? Describa una estrategia óptima para el jugador que gana.

2. Dibuje un árbol de juego completo para nim donde la posición inicial consiste en dos pilas de tres fichas cada una. Omita las posiciones simétricas. Suponga que el último jugador en tomar una ficha pierde. Asigne valores a todos los vértices de modo que el árbol que obtenga sea análogo al de la figura 9.9.2. ¿Ganará siempre el primero o el segundo jugador siguiendo una estrategia óptima? Describa una estrategia óptima para el jugador que gana.

3. Dibuje un árbol de juego completo para nim donde la posición inicial consiste en dos pilas, una con tres fichas y la otra con dos. Suponga que el último jugador en tomar una ficha gana. Asigne valores a todos los vértices de modo que el árbol que obtenga sea análogo al de la figura 9.9.2. ¿Ganará siempre el primero o el segundo jugador siguiendo una estrategia óptima? Describa una estrategia óptima para el jugador que gana.

4. Dibuje un árbol de juego completo para nim donde la posición inicial consiste en dos pilas de tres fichas cada una. Omita las posiciones simétricas. Suponga que el último jugador en tomar una ficha gana. Asigne valores a todos los vértices de modo que el árbol que obtenga sea análogo al de la figura 9.9.2. ¿Ganará siempre el primero o el segundo jugador siguiendo una estrategia óptima? Describa una estrategia óptima para el jugador que gana.

5. Dibuje un árbol de juego completo para la versión de nim descrita en el ejercicio 1. Suponga que el último jugador en tomar una ficha gana. Asigne valores a todos los vértices de modo que el árbol que obtenga sea análogo al de la figura 9.9.2. ¿Ganará siempre el primero o el segundo jugador siguiendo una estrategia óptima? Describa una estrategia óptima para el jugador que gana.

6. Dé un ejemplo de un árbol de juego completo (posiblemente hipotético) en el que un vértice terminal es 1 si el primer jugador gana y 0 si el primer jugador pierde con las siguientes propiedades: hay más ceros que unos en los vértices terminales, pero el primer jugador siempre puede ganar si sigue una estrategia óptima.

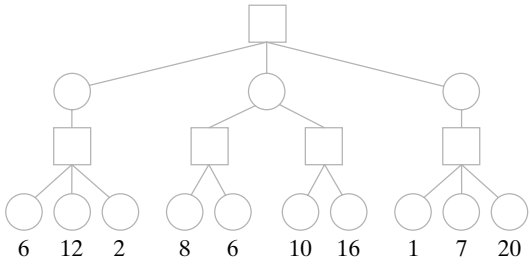
Los ejercicios 7 y 8 se refieren a nim y nim'. Nim es el juego que usa n pilas de fichas como se describe en esta sección, en donde el último jugador que mueve pierde. Nim' es el juego que usa n pilas de fichas como se describe en esta sección excepto que el último jugador que mueve gana. Se fijan n pilas con un número fijo de fichas. Se supone que al menos una pila tiene al menos dos fichas.

★ **7.** Demuestre que el primer jugador puede ganar siempre en *nim* si y sólo si el primer jugador puede ganar siempre en *nim*'.

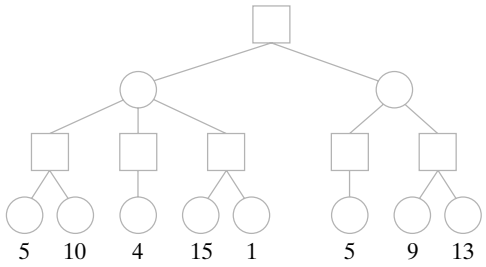
8. Dada una estrategia ganadora para un jugador de nim en particular, describa una estrategia ganadora para este jugador de nim'.

Evalúe cada vértice en cada árbol de juego. Los valores de los vértices terminales están dados.

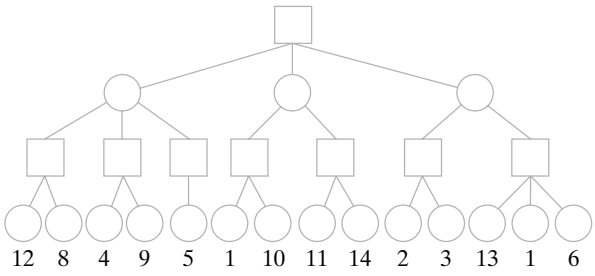
9.



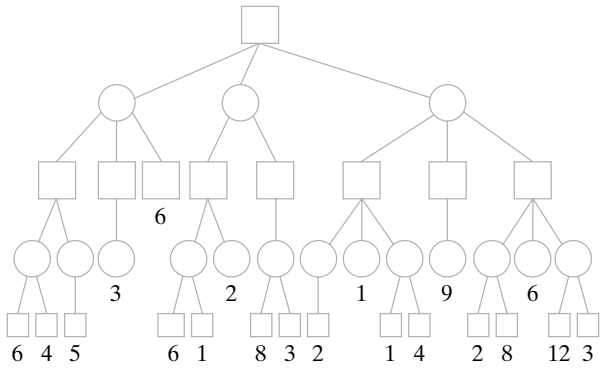
10.



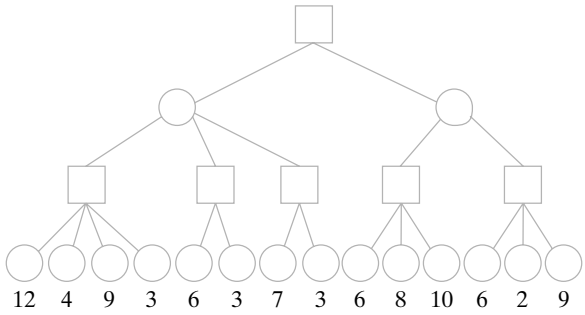
11.



12.



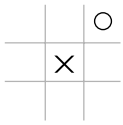
13.



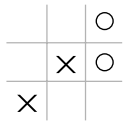
14. Evalúe la raíz de cada uno de los árboles de los ejercicios 9 al 13 mediante una búsqueda a profundidad con recorte alfa-beta. Suponga que los hijos se evalúan de izquierda a derecha. Para cada vértice cuyo valor se calcula, escriba el valor en el vértice. Marque la raíz de cada subárbol que se recorta. El valor de cada vértice terminal se escribe abajo del vértice.

En los ejercicios 15 al 18, determine el valor de la posición en el gato usando la función de evaluación del ejemplo 9.9.1.

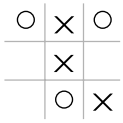
15.



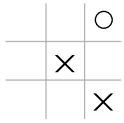
16.



17.



18.



19. Suponga que el primer jugador se mueve al centro del cuadro del gato. Dibuje un árbol de juego de dos niveles, con la raíz que tiene una X en el centro del cuadro. Omita las posiciones simétricas. Evalúe todos los vértices mediante la función de evaluación del ejemplo 9.9.1. ¿Hacia dónde se moverá O?

★**20.** Un programa de búsqueda de dos niveles basado en la función de evaluación E del ejemplo 9.9.1, ¿jugará un juego perfecto de gato? Si no es así, ¿puede alterar E de manera que un programa de búsqueda de dos niveles juegue un juego perfecto?

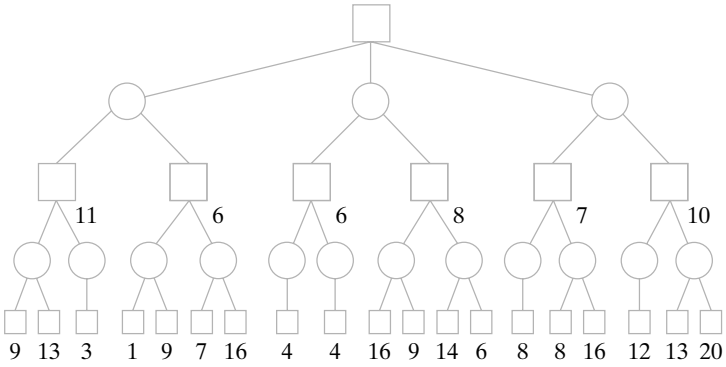
21. Escriba un algoritmo que evalúe los vértices de un árbol de juego hasta el nivel n usando una búsqueda a profundidad. Suponga que existe una función de evaluación E.

★***22.*** Escriba un algoritmo que evalúe la raíz de un árbol de juego usando una búsqueda a lo largo de nivel n con recorte alfa-beta. Suponga que existe una función de evaluación E.

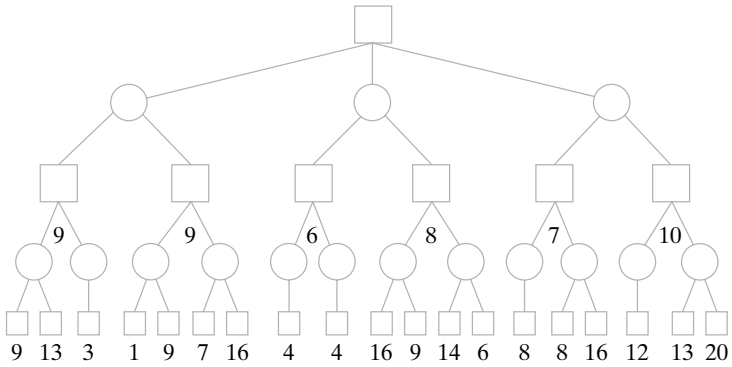
El siguiente enfoque con frecuencia conduce a más recortes que el de minimax alfa-beta puro. Primero, se realiza una búsqueda de dos niveles. Se evalúan los hijos de izquierda a derecha. En este punto, todos los hijos de la raíz tendrán valores. Después, se ordenan los hijos de la raíz con los movimientos más promisorios a la izquierda. Ahora, se utiliza una búsqueda a profundidad de n niveles con recorte alfa-beta. Se evalúan los hijos de izquierda a derecha.

Se realiza este procedimiento para $n = 4$ por cada árbol de juego de los ejercicios 23 al 25. Se coloca una marca junto a la raíz de cada subárbol que se recorta. El valor de cada vértice, según lo da la función de evaluación, se coloca abajo del vértice.

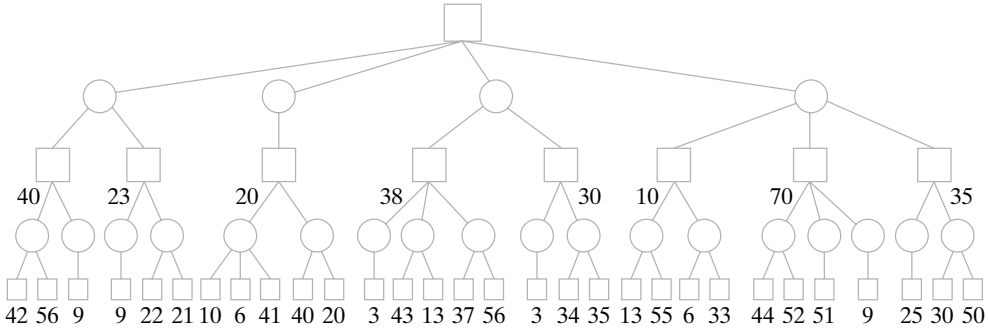
23.



24.

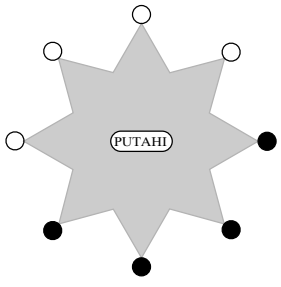


25.



26. Escriba un algoritmo para realizar el procedimiento descrito en el ejercicio 23.

Mu Torere es un juego de dos personas jugado por los maoríes (vea [Bell]). El tablero es una estrella de ocho picos con un área circular en el centro conocida como putahi.



El primer jugador tiene cuatro fichas negras y el segundo tiene cuatro fichas blancas. La posición inicial se muestra en la estrella. Un jugador que no puede hacer un movimiento pierde. Los jugadores alternan movimientos. Cuando mucho una ficha puede ocupar un pico de la estrella o el putahi. Un movimiento consiste en

a) Moverse al pico adyacente

b) Moverse del putahi a un pico

c) Moverse de un pico al putahi siempre que uno o ambos picos adyacentes contengan piezas del oponente

★27. Desarrolle una función de evaluación para Mu Torere.

★28. Combine la función de evaluación del ejercicio 27 con una búsqueda de dos niveles del árbol de juego para obtener un algoritmo que permita jugar Mu Torere. Evalúe la habilidad para jugar de este algoritmo.

★29. ¿Puede el primer jugador ganar siempre en Mu Torere?

★30. ¿Puede el primer jugador empatar siempre en Mu Torere?

31. [*Proyecto*] Según [Nilsson], el árbol de juego completo para el ajedrez tiene más de 10^{100} vértices. Haga un informe acerca de cómo se obtuvo esta estimación.

★**32.** [*Proyecto*] Desarrolle una función de evaluación para Kalah. (Vea las reglas en [Ainslie]).

★**33.** Desarrolle un algoritmo que juegue Kalah basado en la función de evaluación del ejercicio 32. Evalúe la habilidad para jugar de este algoritmo.

Notas

Las siguientes son referencias recomendadas de árboles: [Berge; Bondy; Deo; Even, 1979; Gibbons; Harary; Knuth, 1997; Liu, 1985; y Ore].

Vea en [Date] el uso de árboles para bases jerárquicas de datos.

[Johnsonbaugh] tiene información adicional de códigos Huffman y una demostración de que el algoritmo 9.1.9 construye un árbol de Huffman óptimo.

[Golomb, 1965] describe la exploración hacia atrás y contiene varios ejemplos y aplicaciones.

Se recomienda consultar [Tarjan] para los algoritmos de árboles de expansión mínima y sus implementaciones.

[Johnsonbaugh] analiza el tiempo mínimo para ordenar al igual que las cotas inferiores para otros problemas.

Se hace un estudio exhaustivo de los algoritmos clásicos para ordenar en [Knuth, 1998b]. Vea en [Akl; Leighton; Lester; Lewis; Miller; y Quinn] ordenamientos usando máquinas paralelas.

Algunas referencias buenas de árboles de juegos son [Nievergelt; Nilsson; y Slagel]. En [Frey] se aplica el procedimiento minimax a un solo juego. Se analizan y comparan varios métodos para acelerar la búsqueda del árbol de juego. Se proporcionan programas de computadora. [Berlekamp, 2001, 2003] contiene una teoría general de juegos así como análisis de muchos juegos específicos.

Repaso del capítulo

Sección 9.1

1. Árbol libre
2. Árbol con raíz
3. Nivel de un vértice en un árbol con raíz
4. Altura de un árbol con raíz
5. Árbol de definición de jerarquía
6. Código Huffman

Sección 9.2

7. Padre
8. Ancestro
9. Hijo
10. Descendiente
11. Hermano
12. Vértice terminal
13. Vértice interno
14. Subárbol
15. Gráfica acíclica
16. Caracterización alternativa de los árboles (Teorema 9.2.3)

Sección 9.3

17. Árbol de expansión
18. Una gráfica tiene un árbol de expansión si y sólo si es conexa.
19. Búsqueda a lo ancho
20. Búsqueda a profundidad
21. Búsqueda de regreso

Sección 9.4

22. Árbol de expansión mínima

23. Algoritmo de Prim para encontrar el árbol de expansión mínima

24. Algoritmo ambicioso

Sección 9.5

25. Árbol binario

26. Hijo izquierdo en un árbol binario

27. Hijo derecho en un árbol binario

28. Árbol binario completo

29. Si T es un árbol binario completo con i vértices internos, entonces T tiene $i + 1$ vértices terminales y $2i + 1$ vértices totales.

30. Si un árbol binario de altura h tiene t vértices terminales, entonces $\lg t \leq h$.

31. Árbol de búsqueda binario

32. Algoritmo para construir un árbol de búsqueda binario

Sección 9.6

33. Recorrido preorden

34. Recorrido entreorden

35. Recorrido postorden

36. Forma prefijo de una expresión (notación polaca)

37. Forma entrefijo de una expresión

38. Forma postfijo de una expresión (notación polaca inversa)

39. Representación de una expresión por un árbol

Sección 9.7

40. Árbol de decisiones

41. La altura de un árbol de decisiones que representa un algoritmo es proporcional al tiempo del algoritmo en el peor caso.

42. Cualquier algoritmo para ordenar requiere el menos $\Omega(n \lg n)$ comparaciones en el peor caso para ordenar n artículos.

Sección 9.8

43. Árboles libres isomorfos

44. Árboles con raíz isomorfos

45. Árboles binarios isomorfos

46. El número de Catalan $C(2n, n)/(n + 1)$ es igual al número de árboles binarios no isomorfos con n vértices.

47. Algoritmo de tiempo lineal (algoritmo 9.8.13) para probar si dos árboles binarios son isomorfos

Sección 9.9

48. Árbol de juego

49. Procedimiento minimax

50. Búsqueda de nivel n

51. Función de evaluación

52. Recorte alfa-beta

53. Valor alfa

54. Recorte alfa

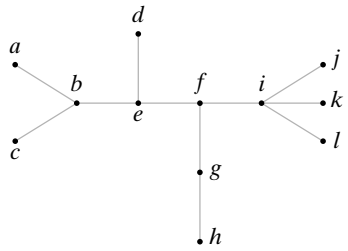
55. Valor beta

56. Recorte beta

Autoevaluación del capítulo

Sección 9.1

1. Dibuje un árbol libre como un árbol con raíz en c.



2. Encuentre el nivel de cada vértice en el árbol adyacente con raíz en c.
3. Encuentre la altura del árbol adyacente con raíz en c.
4. Construya un código Huffman óptimo para el conjunto de letras en la tabla.

Letra	*Frecuencia*
A	5
B	8
C	5
D	12
E	20
F	10

Sección 9.2

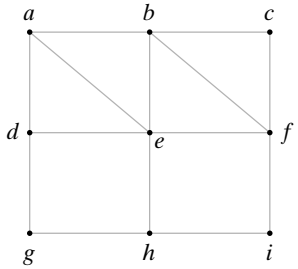
5. Dibuje el árbol libre del ejercicio 1 como un árbol con raíz en f. Encuentre
a) El padre de a.
b) Los hijos de b.
c) Los vértices terminales.
d) El subárbol con raíz en e.

Responda falso o verdadero en los ejercicios 6 al 8 y explique su respuesta.

6. Si T es un árbol con seis vértices, T debe tener cinco aristas.
7. Si T es un árbol con raíz de seis vértices, la altura de T es cuando mucho 5.
8. Una gráfica acíclica con ocho vértices tiene siete aristas.

Sección 9.3

9. Utilice la búsqueda a lo ancho (algoritmo 9.3.6) con el orden de vértices *eachgbdfi* para encontrar un árbol de expansión para la siguiente gráfica.

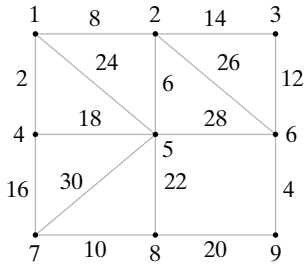


10. Utilice la búsqueda a profundidad (algoritmo 9.3.7) con el orden de vértices *eachgbdfi* para encontrar un árbol de expansión para la gráfica del ejercicio 9.
11. Utilice la búsqueda a lo ancho (algoritmo 9.3.6) con el orden de vértices *fdehagbci* para encontrar un árbol de expansión para la gráfica del ejercicio 9.

12. Utilice la búsqueda a profundidad (algoritmo 9.3.7) con el orden de vértices *fdehagbci* para encontrar un árbol de expansión para la gráfica del ejercicio 9.

Sección 9.4

13. Encuentre el árbol de expansión mínima para la siguiente gráfica.



14. ¿En qué orden se agregan las aristas mediante el algoritmo de Prim para la gráfica del ejercicio 13 si el vértice inicial es 1?

15. ¿En qué orden se agregan las aristas mediante el algoritmo de Prim para la gráfica del ejercicio 13 si el vértice inicial es 6?

16. Dé un ejemplo del uso del método ambicioso que no lleve a un algoritmo óptimo.

Sección 9.5

17. Dibuje un árbol binario con exactamente dos hijos izquierdos y un hijo derecho.

18. Un árbol binario completo tiene 15 vértices internos. ¿Cuántos vértices terminales tiene?

19. Coloque las palabras

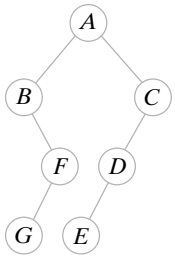
PROCESAR PALABRAS GENERA MANUSCRITOS LIMPIOS
PERO NO NECESARIAMENTE PROSA CLARA

en el orden en que aparecen, en un árbol de búsqueda binario.

20. Explique cómo se buscaría MÁS en el árbol de búsqueda binario del ejercicio 19.

Sección 9.6

Los ejercicios 21 al 23 se refieren al siguiente árbol binario.



21. Liste el orden en que se procesan los vértices usando el recorrido de preorden.

22. Liste el orden en que se procesan los vértices usando el recorrido de entreorden.

23. Liste el orden en que se procesan los vértices usando el recorrido de postorden.

24. Represente la expresión en forma de prefijo $- *E/BD - CA$ como un árbol binario. También escriba la forma de posfijo y la forma de entrefijo con paréntesis completos de la expresión.

Sección 9.7

25. Seis monedas son idénticas en apariencia, pero una es más pesada o más ligera que las otras, que pesan todas lo mismo. Pruebe que se requiere pesar al menos tres veces en el peor

caso para identificar la moneda defectuosa y determine si es más pesada o más ligera usando sólo una balanza.

26. Dibuje un árbol de decisiones que dé un algoritmo para resolver el acertijo de las monedas del ejercicio 25 en no más de tres pesadas en el peor caso.

27. El profesor E. Sabic asegura haber descubierto un algoritmo que usa cundo mucho $100n$ comparaciones en el peor caso para ordenar n artículos, para toda $n \geq 1$. El algoritmo del profesor compara repetidas veces dos elementos y según el resultado de la comparación, modifica la lista original. Dé un argumento que muestre que el profesor está equivocado.

28. El algoritmo de *orden de inserción binaria* ordena un arreglo de tamaño n de la siguiente manera. Si $n = 1$, 2 o 3, el algoritmo usa un orden óptimo. Si $n > 3$, el algoritmo ordena s_1, . . . , s_n como sigue. Primero s_1, . . . , s_{n-1} se ordenan en forma recursiva. Luego se usa la búsqueda binaria para determinar la posición correcta para s_n, después de lo cual se inserta s_n en su lugar. Determine el número de comparaciones usada por la inserción binaria en el peor caso para $n = 4, 5, 6$. ¿Algún otro algoritmo requiere menos comparaciones para $n = 4, 5, 6$?

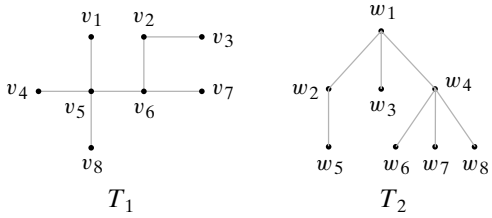
Sección 9.8

En los ejercicios 29 y 30, responda falso o verdadero y explique su repuesta.

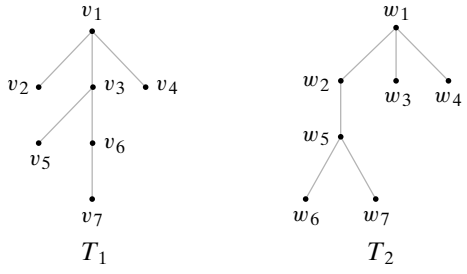
29. Si T_1 y T_2 son árboles con raíz isomorfos, entonces T_1 y T_2 son isomorfos como árboles libres.

30. Si T_1 y T_2 y son árboles con raíz que son isomorfos como árboles libres, entonces T_1 y T_2 son isomorfos como árboles con raíz.

31. Determine si los árboles libres son isomorfos. Si los árboles son isomorfos, dé el isomorfismo. Si los árboles no son isomorfos, dé una invariante que los árboles no compartan.

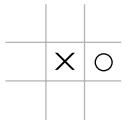


32. Determine si los árboles con raíz son isomorfos. Si los árboles son isomorfos, dé un isomorfismo. Si los árboles no son isomorfos, dé una invariante que los árboles no compartan.



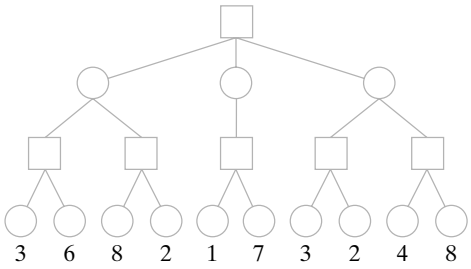
Sección 9.9

33. Encuentre el valor de la posición del gato usando la función de evaluación del ejemplo 9.9.1.



34. Proporcione una función de evaluación para una posición en el gato diferente a la del ejemplo 9.9.1. Intente discriminar más entre las posiciones que la función de evaluación de ese ejemplo.

35. Evalúe cada vértice en el árbol de juego. Los valores de los vértices terminales están dados.

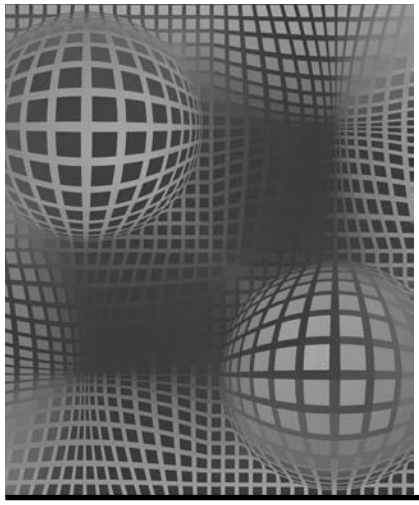


36. Evalúe la raíz del árbol del ejercicio 35 usando el procedimiento minimax con un recorte alfa-beta. Suponga que los hijos se evalúan de izquierda a derecha. Para cada vértice cuyo valor se calcula, escriba el valor en el vértice. Coloque una marca junto a la raíz de cada subárbol que se recorta.

Ejercicios para computadora

1. Escriba un programa que pruebe si una gráfica es un árbol.
2. Escriba un programa que, dada la matriz de adyacencia y un vértice v, dibuje el árbol con raíz en v usando el despliegue de gráficas.
3. Escriba un programa que, dada una tabla de frecuencias para los caracteres, construya un código Huffman óptimo.
4. Escriba un programa que codifique y decodifique texto dado un código Huffman.
5. Calcule una tabla de caracteres y frecuencias mediante el muestreo de un texto. Utilice su programa del ejercicio 3 para generar un código Huffman óptimo. Utilice su programa del ejercicio 4 para codificar una muestra de texto. Compare el número de bits usados para codificar el texto usando el código Huffman con el número de bits utilizados para codificar el texto en ASCII.
6. Escriba un programa que, dado un árbol T, calcule la excentricidad de cada vértice en T y encuentre el centro (o centros) de T.
7. Escriba un programa que, dado un árbol con raíz y un vértice v,
a) encuentre el padre de v
b) encuentre los ancestros de v
c) encuentre los hijos de v
d) encuentre los descendientes de v
e) encuentre los hermanos de v
f) determine si v es un vértice terminal.
8. Escriba un programa que encuentre un árbol de expansión en una gráfica.
9. Escriba un programa que determine si una gráfica es conexa.
10. Escriba un programa que encuentre las componentes de la gráfica.
11. Escriba un programa que resuelva el problema de n reinas.
12. Escriba un programa de búsqueda de regreso para determinar si dos gráficas son isomorfas.
13. Escriba un programa de búsqueda de regreso que determine si una gráfica se puede colorear con n colores y, si se puede, produzca un coloreado.
14. Escriba un programa para búsqueda de regreso que determine si una gráfica tiene un ciclo de Hamilton y si lo tiene, lo encuentre.
15. Escriba un programa que, dada una gráfica G y un árbol de expansión para G, calcule la matriz del ciclo fundamental de G.
16. Desarrolle el algoritmo de Prim como un programa.
17. Desarrolle el algoritmo de Kruskal (dado en el ejercicio 20, sección 9.4) como un programa.
18. Escriba un programa que acepte cadenas y las coloque en un árbol de búsqueda binaria.
19. Escriba un programa que construya todos los árboles binarios de n vértices.
20. Escriba un programa que genere un árbol binario de n vértices aleatorio.
21. Desarrolle los recorridos preorden, postorden e entreorden como programas.
22. Desarrolle el orden de torneo como un programa.

23. Implemente el algoritmo 9.8.13, que prueba si dos árboles binarios son isomorfos, como un programa.
24. Escriba un programa que genere el árbol de juego completo para nim en el que la posición inicial consiste en dos pilas de cuatro fichas cada una. Suponga que el último jugador en tomar un ficha pierde.
25. Desarrolle el procedimiento minimax como un programa.
26. Desarrolle el procedimiento minimax con recorte alfa-beta como un programa.
27. Desarrolle el método para jugar gato del ejemplo 9.9.1 como un programa.
28. Escriba un programa que juegue el juego perfecto de gato.
29. [*Proyecto*] Desarrolle un programa de computadora para jugar un juego que tenga reglas relativamente sencillas. Los juegos sugeridos son Cribbage, Otelo, The Mill, Battleship y Kalah.



Capítulo 10

MODELOS DE REDES†

Sólo sigue la corriente, Shel, sólo sigue la corriente.

DE THE IN-LAWS (LOS FOCKER)

En este capítulo se estudian los modelos de redes, que usan gráficas dirigidas. La mayor parte del capítulo se dedica al problema de maximizar el flujo a través de una red. La red puede ser una red de transporte por la que fluyen bienes, una tubería por la que fluye petróleo, una red de computadoras por la que fluyen datos, o cualquier cantidad de posibilidades diferentes. En cada caso el problema es encontrar un flujo máximo. Muchos otros problemas, que en apariencia no son problemas de flujo, de hecho, se pueden modelar como problemas de flujo en una red.

Maximizar el flujo en una red es un problema que pertenece tanto a la teoría de gráficas como a la investigación de operaciones. El problema del agente viajero proporciona otro ejemplo de un problema de teoría de gráficas e investigación de operaciones. La **investigación de operaciones** estudia la amplia categoría de problemas de optimización del desempeño de un sistema. Los problemas típicos estudiados en investigación de operaciones son problemas de redes, de asignación de recursos y de asignación de personal.

10.1 ➜ Introducción

WWW

Considere la gráfica dirigida de la figura 10.1.1, que representa una tubería de petróleo. El petróleo se descarga en el muelle a y se bombea por toda la red de la refinería z. Los vértices b, c, d y e representan estaciones de bombeo intermedias. Las aristas dirigidas representan subtuberías del sistema y muestran la dirección en que puede fluir el petróleo. Las etiquetas de las aristas indican las capacidades de las subtuberías. El problema es encontrar una manera de maximizar el flujo del muelle a la refinería y calcular el valor de este flujo máximo. La figura 10.1.1 proporciona un ejemplo de una **red de transporte**.

† La sección acerca de redes de Petri que apareció en ediciones anteriores de este libro se encuentra disponible en la página de Internet: www.depaul.edu/~rjohnson.

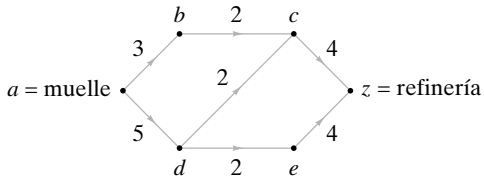


Figura 10.1.1 Una red de transporte.

Definición 10.1.1 ▶

Una *red de transporte* (o más sencillo, una *red*) es un gráfica dirigida, ponderada, simple que satisface las siguientes condiciones:

a) Un vértice designado, el *origen* o *fuente*, no tiene aristas entrantes.

b) Un vértice designado, *destino* o *sumidero*, no tiene aristas salientes.

c) El peso C_{ij} de la arista dirigida (i, j), llamado *capacidad* de (i, j), es un número no negativo. ◀

Ejemplo 10.1.2 ▶

La gráfica de la figura 10.1.1 es una red de transporte. El origen es el vértice a y el destino es el vértice z. La capacidad de la arista (a, b), $C_{a,b}$, es 3 y la capacidad de la arista (b, c), $C_{b,c}$, es 2. ◀

En todo este capítulo, si G es una red, el origen se denotará por a y el destino por z.

Un **flujo** en una red asigna un flujo a cada arista dirigida que no excede la capacidad de esa arista. Más aún, si se supone que el flujo que entra a un vértice v, que no es el origen ni el destino, es igual al flujo que sale de v. La siguiente definición precisa estas ideas.

Definición 10.1.3 ▶

Sea G una red de transporte. Sea C_{ij} la capacidad de la arista dirigida (i, j). Un *flujo* F en G asigna a cada arista dirigida (i, j) un número no negativo F_{ij} tal que:

a) $F_{ij} \leq C_{ij}$.

b) Para cada vértice j, que no es la fuente ni el destino,

$$\sum_i F_{ij} = \sum_i F_{ji}. \tag{10.1.1}$$

[En una suma como (10.1.1), a menos que se especifique lo contrario, se supone que la suma se toma sobre todos los vértices i. Además, si (i, j) no es un arista, se hace $F_{ij} = 0$].

F_{ij} recibe el nombre de *flujo en la arista* (i, j). Para cualquier vértice j,

$$\sum_i F_{ij}$$

se llama *flujo que entra a* j y

$$\sum_i F_{ji}$$

se llama *flujo que sale de* j. ◀

La propiedad expresada por la ecuación (10.1.1) se llama **conservación del flujo**. En el ejemplo del bombeo de petróleo de la figura 10.1.1, la conservación del flujo significa que el petróleo no se usa ni se suministra en las estaciones de bombeo b, c, d y e.

Ejemplo 10.1.4 ▶

Las asignaciones

$$F_{ab} = 2, \quad F_{bc} = 2, \quad F_{cz} = 3, \quad F_{ad} = 3,$$
$$F_{dc} = 1, \quad F_{de} = 2, \quad F_{ez} = 2,$$

definen un flujo para la red de la figura 10.1.1. Por ejemplo, el flujo que entra al vértice d,

$$F_{ad} = 3,$$

es el mismo que el flujo que sale del vértice d,

$$F_{dc} + F_{de} = 1 + 2 = 3. \quad \blacktriangleleft$$

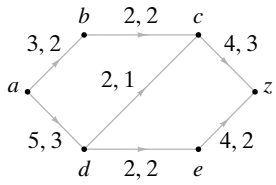


Figura 10.1.2 Flujo en una red. Las aristas tienen etiquetas x, y para indicar la capacidad x y el flujo y.

En la figura 10.1.2 se dibujó de nuevo la red de la figura 10.1.1 para mostrar el flujo del ejemplo 10.1.4. Una arista e se etiqueta "x, y" si la capacidad de e es x y el flujo en e es y. Esta notación se usará en todo el capítulo.

Advierta que en el ejemplo 10.1.4, el flujo que sale del origen

$$F_{ab} + F_{ad},$$

es el mismo que el flujo que entra al destino z,

$$F_{cz} + F_{ez};$$

ambos valores son 5. El siguiente teorema demuestra que siempre es cierto que el flujo que sale del origen es igual al flujo que entra al destino.

Teorema 10.1.5

Dado un flujo F en una red, el flujo que sale del origen a es igual al flujo que llega al destino z; es decir,

$$\sum_i F_{ai} = \sum_i F_{iz}.$$

Demostración Sea V el conjunto de vértices. Se tiene

$$\sum_{j\in V}\left(\sum_{i\in V} F_{ij}\right) = \sum_{j\in V}\left(\sum_{i\in V} F_{ji}\right),$$

ya que cada doble suma es

$$\sum_{e\in E} F_e,$$

donde E es el conjunto de aristas. Ahora

$$\begin{aligned} 0 &= \sum_{j\in V}\left(\sum_{i\in V} F_{ij} - \sum_{i\in V} F_{ji}\right) \\ &= \left(\sum_{i\in V} F_{iz} - \sum_{i\in V} F_{zi}\right) + \left(\sum_{i\in V} F_{ia} - \sum_{i\in V} F_{ai}\right) \\ &\quad + \sum_{\substack{j\in V \\ j\neq a,z}}\left(\sum_{i\in V} F_{ij} - \sum_{i\in V} F_{ji}\right) \\ &= \sum_{i\in V} F_{iz} - \sum_{i\in V} F_{ai} \end{aligned}$$

porque $F_{zi} = 0 = F_{ia}$, para toda $i \in V$, y (definición 10.1.3*b*)

$$\sum_{i\in V} F_{ij} - \sum_{i\in V} F_{ji} = 0 \qquad \text{si } j \in V - \{a, z\}.$$

A la luz del Teorema 10.1.5, se establece la siguiente definición.

Definición 10.1.6 ▶ Sea F un flujo en una red G. El valor

$$\sum_i F_{ai} = \sum_i F_{iz}$$

se llama el *valor del flujo F*. ◀

Ejemplo 10.1.7 ▶ El valor del flujo en la red de la figura 10.1.2 es 5. ◀

El problema para un red de transporte G se puede establecer como encontrar un flujo máximo en G; es decir, entre todos los flujos posibles en G, encuentre el flujo F tal que el valor de F sea máximo. En la siguiente sección se da un algoritmo que resuelve con eficiencia este problema. Para terminar la sección se dan ejemplos adicionales.

Ejemplo 10.1.8 ▶

Red de bombeo

La figura 10.1.3 representa una red de bombeo en la que se entrega agua para dos ciudades, A y B, desde tres pozos w_1, w_2 y w_3. Las capacidades de los sistemas intermedios se muestran en las aristas. Los vértices b, c y d representan las estaciones de bombeo intermedias. Modele este sistema como una red de transporte.

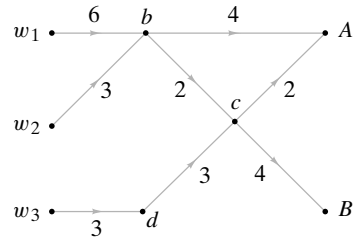


Figura 10.1.3 Red de bombeo. El agua para las ciudades A y B se entrega desde los pozos w_1, w_2 y w_3. Las capacidades se indican en las aristas.

Para obtener el origen y destino designados, se puede obtener una red de transporte equivalente uniendo los orígenes en un **superorigen** y los destinos en un **superdestino** (vea la figura 10.1.4). En ésta, ∞ representa una capacidad ilimitada.

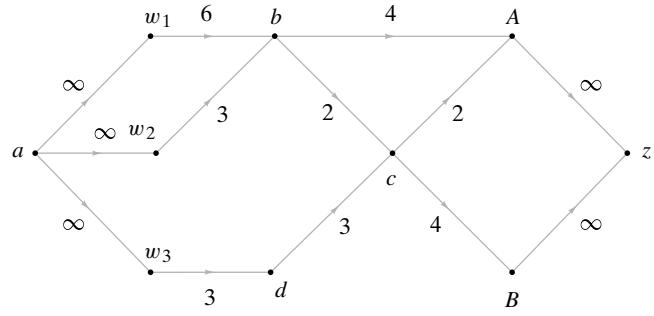


Figura 10.1.4 Red de la figura 10.1.3 con origen y destino designados.

◀

Ejemplo 10.1.9 ▶

Una red de flujo de tráfico

Es posible ir de la ciudad A a la ciudad C directamente o pasando por la ciudad B. Durante el periodo de 6:00 PM a 7:00 PM, los tiempos de viaje promedio son

A a B	15 minutos
B a C	30 minutos
A a C	30 minutos.

Las capacidades máximas de la rutas son

A a B	3000 vehículos
B a C	2000 vehículos
A a C	4000 vehículos.

Represente el flujo de tráfico de A a C durante el periodo de 6:00 PM a 7:00 PM como una red.

Un vértice representará una ciudad en un tiempo específico (vea la figura 10.1.5). Una arista conecta X, t_1 con Y, t_2 si se puede salir de la ciudad X a las t_1 PM y llegar a la ciudad Y a las t_2 PM. La capacidad de una arista es la capacidad de la ruta. Las aristas de capacidad infinita conectan a A, t_1 con A, t_2 y B, t_1 con B, t_2 para indicar que cualquier número de autos puede esperar en las ciudades A o B. Por último, se introduce un superorigen y un superdestino.

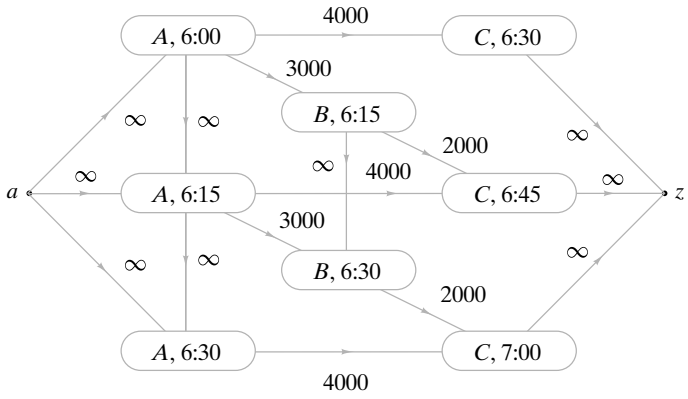


Figura 10.1.5 Red que representa el flujo de tráfico de la ciudad A a la ciudad C durante el periodo de 6:00 PM a 7:00 PM. ◀

Se han usado algunas variaciones del problema de flujo en redes en el diseño de redes de computadoras eficientes (vea [Jones; Kleinrock]). En un modelo de una red de computadoras, un vértice es un mensaje o un centro de conmutación, una arista representa un canal por el que se pueden trasmitir datos entre vértices, un flujo es el número promedio de bits por segundo que se trasmiten en un canal, y la capacidad de una arista es la capacidad del canal correspondiente.

Sección de ejercicios de repaso

1. ¿Qué es una red?

2. ¿Qué es la fuente o el origen en una red?

3. ¿Qué es un sumidero o destino en una red?

4. ¿Qué es una capacidad en una red?

5. ¿Qué es un flujo en una red?

6. ¿Qué es un flujo en una arista?

7. ¿Qué es un flujo en un vértice?

8. ¿Qué es un flujo que sale de un vértice?

9. ¿Qué es conservación del flujo?

10. Dado un flujo en una red, ¿cuál es la relación entre el flujo que sale del origen y el que llega al destino?

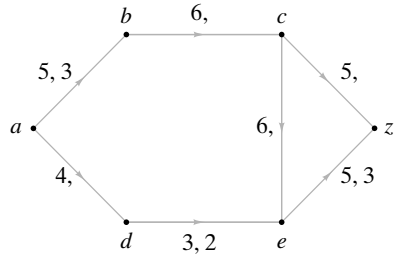
11. ¿Qué es un superorigen?

12. ¿Qué es un superdestino?

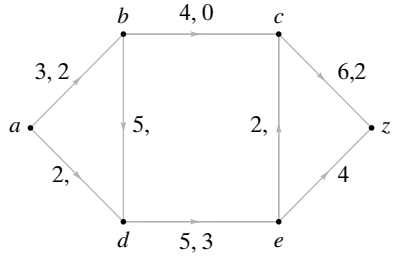
Ejercicios

En los ejercicios 1 al 3, encuentre los flujos en las aristas que faltan de manera que el resultado sea un flujo en la red dada. Determine los valores de los flujos.

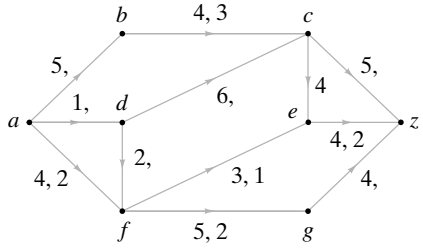
1.



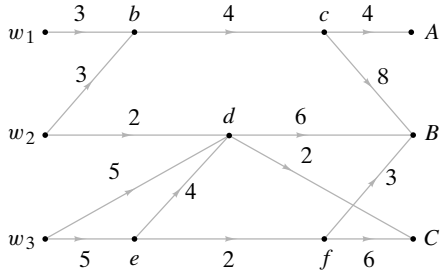
2.



3.



4. La siguiente gráfica representa una red de bombeo por la que se entrega petróleo a tres refinerías, A, B y C, desde tres pozos, w_1, w_2 y w_3. Las capacidades de los sistemas intermedios se indican en las aristas. Los vértices b, c, d, e y f representan las estaciones de bombeo intermedias. Modele este sistema como una red.



5. Modele el sistema del ejercicio 4 como una red suponiendo que el pozo w_1 puede bombear cuando mucho 2 unidades, el pozo w_2, cuando mucho 4 unidades y el pozo w_3 cuando mucho 7 unidades.

6. Modele el sistema del ejercicio 5 como una red suponiendo, además de las restricciones sobre los pozos, que la ciudad A requiere 4 unidades, la ciudad B requiere 3 unidades y la ciudad C, 4 unidades.

7. Modele el sistema del ejercicio 6 como una red suponiendo, además de las restricciones sobre los pozos y los requerimientos de las ciudades, que la estación de bombeo intermedia d puede bombear cuando mucho 6 unidades.

8. Hay dos rutas de la ciudad A a la ciudad D. Una ruta pasa por la ciudad B y la otra ruta pasa por la ciudad C. Durante el periodo de 7:00 AM a 8:00 AM, los tiempos de viaje promedio son

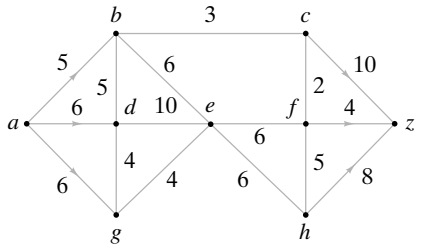
A a B	30 minutos
A a C	15 minutos
B a D	15 minutos
C a D	15 minutos.

Las capacidades máximas de las rutas son

A a B	1000 vehículos
A a C	3000 vehículos
B a D	4000 vehículos
C a D	2000 vehículos.

Represente el flujo de tráfico de A a D durante el periodo de 7:00 AM a 8:00 AM como una red.

9. En el sistema mostrado, se desea maximizar el flujo de a a z. Las capacidades se indican en las aristas. El flujo entre dos vértices, ninguno de los cuales es a o z, puede ser en cualquier dirección. Modele este sistema como una red.



10. Dé un ejemplo de una red con exactamente dos flujos máximos, donde cada F_{ij} es un entero no negativo.

11. ¿Cuál es el número máximo de aristas que puede tener una red de n vértices?

10.2 ➜ Algoritmo de flujo máximo

Si G es una red de transporte, un **flujo máximo** en G es un flujo con valor máximo. En general, habrá varios flujos que tienen el mismo valor máximo. En esta sección se presenta un algoritmo para encontrar un flujo máximo. La idea básica es sencilla: comenzar con un flujo inicial y aumentar, de manera iterativa, el valor del flujo hasta que no sea posible mejorarlo. El flujo obtenido será entonces el flujo máximo.

Se puede tomar el flujo inicial como uno en el que el flujo en cada arista es cero. Para aumentar el valor de un flujo dado, debe encontrarse una trayectoria del origen al destino y aumentar el flujo a lo largo de esta trayectoria.

En este momento resulta útil introducir cierta terminología. En esta sección, G denota una red con origen a, destino z y capacidad C. Por el momento, considere las aristas de G como no dirigidas y sea

$$P = (v_0, v_1, \ldots, v_n), \qquad v_0 = a, \qquad v_n = z,$$

una trayectoria de a a z en esta gráfica no dirigida. (Todas las trayectorias en esta sección se refieren a la gráfica no dirigida). Si una arista e en P es dirigida de v_{i-1} a v_i, se dice que e tiene la **orientación apropiada (respecto a *P*)**; de otra manera, se dice que e tiene la **orientación inapropiada (respecto a *P*)** (vea la figura 10.2.1).

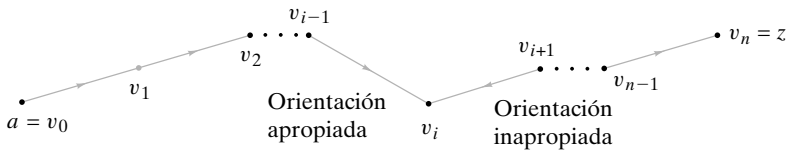


Figura 10.2.1 Aristas con orientación apropiada e inapropiada. La arista (v_{i-1}, v_i) tiene orientación apropiada porque está orientada en la dirección de a a z. La arista (v_i, v_{i+1}) tiene orientación inapropiada porque *no* tiene la dirección de a a z.

Si se puede encontrar una trayectoria P del origen al destino en la que todas las aristas tengan la orientación apropiada y el flujo de cada una sea menor que la capacidad de la arista, es posible aumentar el valor del flujo.

Ejemplo 10.2.1 ▶

Considere la trayectoria de a a z en la figura 10.2.2. Todas las aristas en P tienen la orientación adecuada. El valor del flujo en esta red se puede aumentar en 1, como se aprecia en la figura 10.2.3.

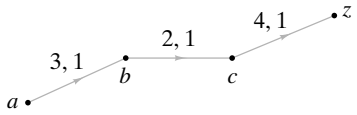


Figura 10.2.2 Una trayectoria cuyas aristas tienen la orientación apropiada.

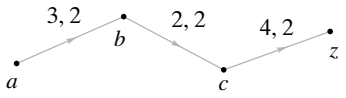


Figura 10.2.3 Después de aumentar en 1 el flujo de la figura 10.2.2. ◀

También es posible aumentar el flujo en ciertas trayectorias del origen al destino, en las que se tienen aristas con orientación apropiada e inapropiada. Sea P una trayectoria de a a z y sea x un vértice en P diferente de a y z (vea la figura 10.2.4). Existen cuatro posibilidades para la orientación de las aristas e_1 y e_2 incidentes en x. En el caso *a)*, ambas aristas tienen la orientación correcta. En este caso, si aumentamos en Δ el flujo en cada arista, el flujo que entra a x todavía será igual al flujo que sale de x. En el caso *b)*, si se incrementa

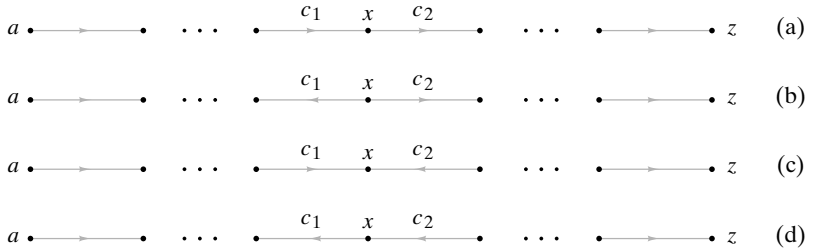


Figura 10.2.4 Las cuatro orientaciones posibles de las aristas incidentes en x.

en Δ el flujo de e_2, debe *disminuirse* en Δ el flujo en e_1 para que el flujo que llega a x siga siendo igual al que sale. El caso *c)* es similar al caso *b)* excepto que se aumenta en Δ el flujo en e_1 y se diminuye en Δ el flujo en e_2. En el caso *d)*, se disminuye en Δ el flujo en ambas aristas. En todos los casos, las asignaciones de aristas que se obtienen dan un flujo. Por supuesto que para llevar a cabo estas modificaciones debemos tener un flujo menor que la capacidad en un arista con la orientación apropiada y un flujo diferente de cero en la arista con la orientación inapropiada.

Ejemplo 10.2.2 ▶

Considere la trayectoria de a a z en la figura 10.2.5. Las aristas (a, b), (c, d) y (d, z) tienen la orientación apropiada y la arista (c, b) tiene la orientación inapropiada. Se disminuye en 1 el flujo de la arista con orientación inapropiada (c, b) y se aumenta en 1 el flujo de las aristas orientadas apropiadamente (a, b), (c, d) y (d, z) (vea la figura 10.2.6). El valor del nuevo flujo es 1 unidad mayor que el original.

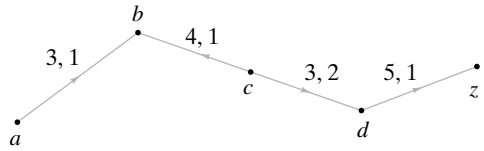


Figura 10.2.5 Una trayectoria con una arista orientada inapropiadamente: (c, b).

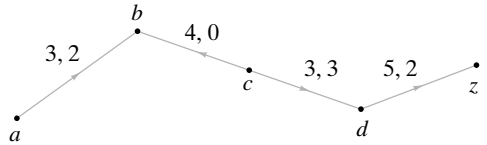


Figura 10.2.6 Después de aumentar en 1 unidad el flujo de la figura 10.2.5.

◀

WWW Se resume el método de los ejemplos 10.2.1 y 10.2.2 como un teorema.

Teorema 10.2.3

Sea P una trayectoria del vértice a al z en una red G que satisface las siguientes condiciones:

a) Para cada arista (i, j) con orientación apropiada en P,

$$F_{ij} < C_{ij}.$$

b) Para cada arista (i, j) con orientación inadecuada en P,

$$0 < F_{ij}.$$

Sea

$$\Delta = \text{mín } X,$$

donde X consiste en los números $C_{ij} - F_{ij}$, para aristas (i, j) con orientación apropiada en P, y F_{ij}, para aristas (i, j) con orientación inapropiada en P. Defina

$$F_{ij}^{*} = \begin{cases} F_{i,j} & \text{si } (i, j) \text{ no está en } P \\ F_{ij} + \Delta & \text{si } (i, j) \text{ tiene orientación apropiada en } P \\ F_{ij} - \Delta & \text{si } (i, j) \text{ no tiene la orientación apropiada en } P. \end{cases}$$

Entonces F es un flujo cuyo valor es Δ mayor que el valor de F.*

Demostración (Vea las figuras 10.2.2, 10.2.3, 10.2.5 y 10.2.6). El argumento que asegura que F^* es un flujo se presentó justo antes del ejemplo 10.2.2. Como la arista (a, v) en P se incrementa Δ, el valor de F^* es Δ unidades mayor que el valor de F.

En la siguiente sección se demuestra que si no hay trayectorias que satisfagan las condiciones del Teorema 10.2.3, el flujo es máximo. Entonces es posible construir un algoritmo basado en el Teorema 10.2.3. La descripción es la siguiente:

1. Se comienza con un flujo (por ejemplo, uno en el que el flujo de cada arista es 0).
2. Se busca una trayectoria que satisfaga las condiciones del Teorema 10.2.3. Si no existe, hay que detenerse; el flujo es máximo.
3. Se incrementa el flujo en toda la trayectoria por Δ unidades, donde Δ se define como en el Teorema 10.2.3, y se va a la línea 2.

En el algoritmo formal, se busca una trayectoria que satisfaga las condiciones del Teorema 10.2.3 al mismo tiempo que se van registrando las cantidades

$$C_{ij} - F_{ij}, F_{ij}.$$

Algoritmo 10.2.4

Encontrar un flujo máximo en una red

Este algoritmo encuentra un flujo máximo en una red. La capacidad de cada arista es un entero no negativo.

Entrada: Una red con origen a, destino z, capacidad C, vértices $a = v_0, \ldots, v_n = z$, y n

Salida: Un flujo máximo F

flujo_máx(a, z, C, v, n) {
  //etiqueta de v es (*predecesor*(v), *val*(v))
  //inicia con flujo cero
1. para cada arista (i, j)
2. $F_{ij} = 0$
3. while(verdadero) {
  //eliminar todas las etiquetas
4. for $i = 0$ to n {
5. *predecesor*$(v_i) = nulo$
6. *val*$(v_i) = nulo$
7. }
  //etiquetar a
8. *predecesor*$(a) = -$
9. *val*$(a) = \infty$
  //U es conjunto de vértices etiquetados no examinados
10. $U = \{a\}$
  //continuar hasta etiquetar z
11. while (*val*$(z) ==$ *nulo*) {
12. if ($U == \emptyset$)//flujo es máximo
13. return F
14. elegir v en U
15. $U = U - \{v\}$
16. $\Delta =$ *val*(v)
17. para cada arista (v, w) con *val*$(w) ==$ *nulo*
18. if ($F_{vw} < C_{vw}$) {
19. *predecesor*$(w) = v$
20. *val*$(w) =$ mín$\{\Delta, C_{vw} - F_{vw}\}$
21. $U = U \cup \{w\}$
22. }
23. para cada arista (w, v) con *val*$(w) ==$ *nulo*
24. if ($F_{wv} > 0$) {
25. *predecesor*$(w) = v$
26. *val*$(w) =$ mín$\{\Delta, F_{wv}\}$
27. $U = U \cup \{w\}$
28. }
29. } //fin del ciclo while (*val*$(z) ==$ *nulo*)
  //encontrar trayectoria P de a a z para revisar flujo
30. $w_0 = z$
31. $k = 0$
32. while ($w_k \neg = a$) {
33. $w_{k+1} =$ *predecesor*(w_k)
34. $k = k + 1$
35. }

```
36.            P = (w_{k+1}, w_k, . . . , w_0)
37.            Δ = val(z)
38.            for i = 1 to k + 1 {
39.                e = (w_i, w_{i−1})
40.                if (e tiene orientación correcta en P)
41.                    F_e = F_e + Δ
42.                else
43.                    F_e = F_e − Δ
44.            }
45.        } //fin del ciclo while(verdadero)
    }
```

Una demostración de que el algoritmo 10.2.4 termina, se deja como ejercicio 19. Si se permite que las capacidades sean números racionales no negativos, el algoritmo también termina; sin embargo, si se permiten números reales no negativos para las capacidades y se permite que las aristas en la línea 17 se examinen en cualquier orden, el algoritmo puede no terminar (vea [Ford, pp. 21−22]).

Con frecuencia nos referimos al algoritmo 10.2.4 como **procedimiento de etiquetado**. Se ilustrará el algoritmo con dos ejemplos.

Ejemplo 10.2.5 ▶

En este análisis, si el vértice v satisface

$$predecesor(v) = p \quad \text{y} \quad val(v) = t,$$

la etiqueta de v en la gráfica se indica como (p, t).

En las líneas 1 y 2, se inicializa el flujo en 0 en cada arista (vea la figura 10.2.7). Después, en las líneas 4 a la 7, las etiquetas se establecen como *nulo*. En las líneas 8 y 9 el vértice a se etiqueta como $(-, \infty)$. En la línea 10 se hace $U = \{a\}$. Después se llega al ciclo "while" (línea 11).

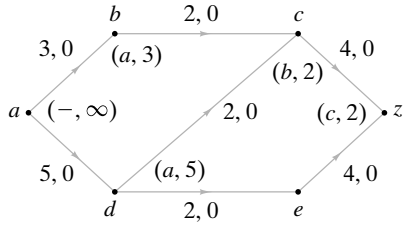


Figura 10.2.7 Después de la primera etiquetada. El vértice v se etiqueta $(predecesor(v), val(v))$.

Como z no está etiquetado y U no está vacío, se pasa a la línea 14 donde se elige el vértice a en U y se elimina de U en la línea 15. En este punto, $U = \emptyset$. Se establece que Δ es igual a ∞ [$= val$(a)] en la línea 16. En la línea 17 se examinan las aristas (a, b) y (a, d) puesto que b y d no están etiquetadas. Para la arista (a, b) se tiene

$$F_{ab} = 0 < C_{ab} = 3.$$

En las líneas 19 y 20, se etiqueta el vértice b como $(a, 3)$ ya que

$$predecesor(b) = a$$

y

$$val(b) = \text{mín}\{\Delta, 3 - 0\} = \text{mín}\{\infty, 3 - 0\} = 3.$$

En la línea 21, se agrega b a U. De manera similar, se etiqueta el vértice d como $(a, 5)$ y se agrega a U. En este punto, $U = \{b, d\}$.

Después se regresa al principio del ciclo "while" (línea 11). Como z no tiene etiqueta y U no está vacío, se pasa a la línea 14, donde se elige un vértice en U. Suponga que se elige b. Se elimina b de U en la línea 15. Se hace Δ igual a 3 [$= val(b)$] en la línea 16. En la línea 17 se examina la arista (b, c). En las líneas 19 y 20 se etiqueta el vértice c como $(b, 2)$ puesto que

$$predecesor(c) = b$$

y

$$val(c) = \text{mín}\{\Delta, 2 - 0\} = \text{mín}\{3, 2 - 0\} = 2.$$

En la línea 21 se agrega c a U. En este punto, $U = \{c, d\}$.

Después se regresa al principio del ciclo "while" (línea 11). Como z no tiene etiqueta y U no está vacío, se pasa a la línea 14, donde se elige un vértice en U. Suponga que se elige c. Se elimina c de U en la línea 15. Se hace Δ igual a 2 [$= val(c)$] en la línea 16. En la línea 17 se examina la arista (c, z). En las líneas 19 y 20 se etiqueta el vértice z como $(c, 2)$. En la línea 21 se agrega z a U. En este punto, $U = \{d, z\}$.

Después se regresa al principio del ciclo "while" (línea 11). Como z está etiquetada, se procede a la línea 30. En las líneas 30 a la 36, siguiendo los predecesores desde z, se encuentra la trayectoria

$$P = (a, b, c, z)$$

de a a z. En la línea 37 se hace Δ igual a 2. Como todas las aristas en P tienen la orientación apropiada, en la línea 41 se aumenta el flujo de cada arista en P en $\Delta = 2$ para obtener la figura 10.2.8.

Después se regresa al principio del ciclo "while" (línea 3). En las líneas 4 a la 7 se hacen todas las etiquetas igual a *nulo*. Luego, en las líneas 8 y 9 se etiqueta el vértice a como $(-, \infty)$ (vea la figura 10.2.8). En la línea 10 se hace $U = \{a\}$. Después se llega al ciclo "while" (línea 11).

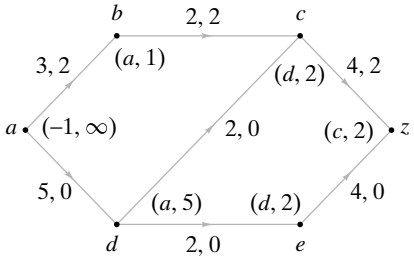


Figura 10.2.8 Después de aumentar el flujo en la trayectoria (a, b, c, z) en 2 unidades y el segundo etiquetado.

Como z no está etiquetado y U es no vacío, se pasa a la línea 14, donde se elige el vértice a en U y se elimina de U en la línea 15. En las líneas 19 y 20 se etiqueta el vértice b como $(a, 1)$ y se etiqueta el vértice d como $(a, 5)$. Se agregan b y d a U de modo que $U = \{b, d\}$.

Después se regresa al principio del ciclo "while" (línea 11). Como z no está etiquetado y U es no vacío, se pasa a la línea 14, donde se elige un vértice en U. Suponga que se selecciona b. Se elimina b de U en la línea 15. En la línea 17 se examina la arista (b, c). Como $F_{bc} = C_{bc}$, no se etiqueta el vértice c en este punto. Ahora $U = \{d\}$.

Entonces se regresa al principio del ciclo "while" (línea 11). Como z no tiene etiqueta y U es no vacío, se pasa a la línea 14, donde se elige el vértice d en U y se elimina de U en la línea 15. En las líneas 19 y 20 se etiqueta el vértice c como $(d, 2)$ y se etiqueta el vértice e como $(d, 2)$. Se agregan c y e a U de modo que $U = \{c, e\}$.

De nuevo se regresa al principio de ciclo "while" (línea 11). Como z no está etiquetado y U es no vacío, se pasa a la línea 14, donde se elige un vértice en U. Suponga que se selecciona c y se elimina de U en la línea 15. En las líneas 19 y 20 se etiqueta el vértice z como $(c, 2)$. Se agrega z a U de manera que $U = \{z, e\}$.

Se regresa al principio del ciclo "while" (línea 11). Como z está etiquetado, se procede a la línea 30. En la línea 36 se encuentra que

$$P = (a, d, c, z).$$

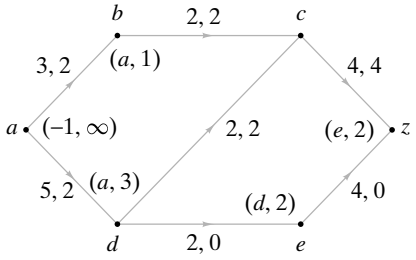


Figura 10.2.9 Después de aumentar el flujo en la trayectoria (a, d, c, z) en 2 unidades y el tercer etiquetado.

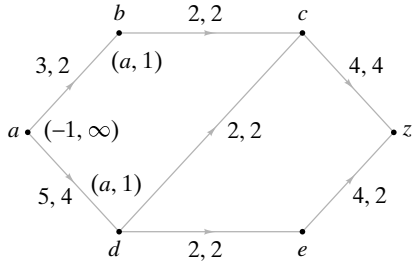


Figura 10.2.10 Después de aumentar el flujo en la trayectoria (a, d, e, z) en 2 unidades y el cuarto etiquetado. El flujo es máximo.

Como todas las aristas en P tienen la orientación adecuada, en la línea 41 se aumenta el flujo en cada una en $\Delta = 2$ para obtener la figura 10.2.9.

Debe verificarse que la siguiente iteración del algoritmo produzca las etiquetas mostradas en la figura 10.2.9. Aumentar el flujo en $\Delta = 2$ produce la figura 10.2.10.

Ahora se regresa al principio del ciclo "while" (línea 3). Después, en las líneas 4 a la 7 se hacen todas las etiquetas iguales a *nulo*. Luego, en las líneas 8 y 9 se etiqueta el vértice a como $(-, \infty)$ (vea la figura 10.2.10). En la línea 10 se hace $U = \{a\}$. Después se llega al ciclo "while" (línea 11).

Como z no tiene etiqueta y U es no vacío, se pasa a la línea 14, donde se elige el vértice a en U y se elimina de U en la línea 15. En las líneas 19 y 20 se etiqueta el vértice b como $(a, 1)$ y se etiqueta el vértice d como $(a, 1)$. Se agregan b y d a U de manera que $U = \{b, d\}$.

Se regresa el principio del ciclo "while" (línea 11). Como z no tiene etiqueta y U es no vacío, se pasa a la línea 14, donde se elige un vértice en U. Suponga que se elige b. Se elimina b de U en la línea 15. En la línea 17, se examina la arista (b, c). Como $F_{bc} = C_{bc}$, no se etiqueta el vértice c. Ahora $U = \{d\}$.

Se regresa al principio del ciclo "while" (línea 11). Como z no está etiquetado y U es no vacío, se pasa a la línea 14, donde se elige el vértice d en U y se elimina de U en la línea 15. En la línea 17 se examinan las aristas (d, c) y (d, e). Como $F_{dc} = C_{dc}$ y $F_{de} = C_{de}$, no se etiqueta el vértice c ni el e. Ahora $U = \emptyset$.

Se regresa al principio del ciclo "while" (línea 11). Como z no tiene etiqueta, se pasa a la línea 12. Como U es vacío, el algoritmo termina. El flujo de la figura 10.2.10 es máximo. ◀

El último ejemplo muestra cómo modificar el algoritmo 10.2.4 para generar un flujo máximo a partir de un flujo dado.

Ejemplo 10.2.6 ▶

Sustituya el flujo cero en las líneas 1 y 2 del algoritmo 10.2.4 con el flujo de la figura 10.2.11 y luego encuentre un flujo máximo.

Después de inicializar el flujo dado, se pasa a las líneas 4 a la 7, donde se establecen todas las etiquetas en *nulo*. Después, en las líneas 8 y 9 se etiqueta el vértice a como $(-, \infty)$ (vea la figura 10.2.11). En la línea 10 se hace $U = \{a\}$. Después se llega al ciclo "while" (línea 11).

Como z no tiene etiqueta y U es no vacío, se pasa a la línea 14, donde se elige el vértice a en U y se elimina de U en la línea 15. En las líneas 19 y 20, se etiqueta el vértice b como $(a, 1)$ y se etiqueta el vértice d como $(a, 1)$. Se agregan b y d a U de modo que $U = \{b, d\}$.

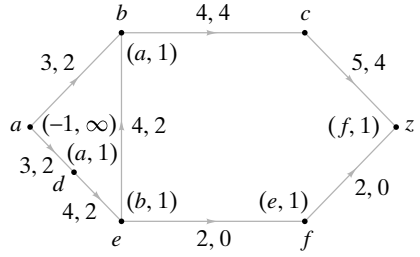


Figura 10.2.11 Después de etiquetar.

Después se regresa al principio del ciclo "while" (línea 11). Como z no tiene etiqueta y U es no vacío, se pasa a la línea 14, donde se elige un vértice de U. Suponga que se elige b. Se elimina b de U en la línea 15. En la línea 17 se examinan las aristas (b, c) y (e, b). Como $F_{bc} = C_{bc}$, no se etiqueta el vértice c. En las líneas 25 y 26 se etiqueta el vértice e como $(b, 1)$ ya que

$$val(e) = \text{mín}\{val(b), F_{eb}\} = \text{mín}\{1, 2\} = 1.$$

Se regresa entonces al principio del ciclo "while" (línea 11). Al final se etiqueta z (vea la figura 10.2.11) y en la línea 36 se encuentra la trayectoria

$$P = (a, b, e, f, z).$$

Las aristas (a, b), (e, f) y (f, z) tienen la orientación apropiada, por lo que el flujo en cada una se aumenta en 1. Como la arista (e, b) tiene la orientación inapropiada, su flujo se disminuye en 1. Se obtiene el flujo de la figura 10.2.12.

Otra iteración del algoritmo produce el flujo máximo mostrado en la figura 10.2.13.

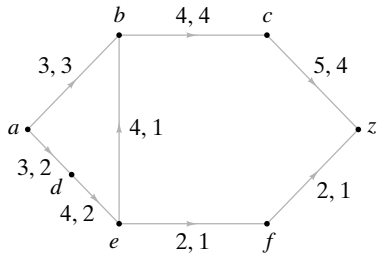


Figura 10.2.12 Después de aumentar en 1 el flujo en la trayectoria (a, b, e, f, z). Observe que la arista (e, b) está orientada inapropiadamente por lo que su flujo *disminuye* en 1.

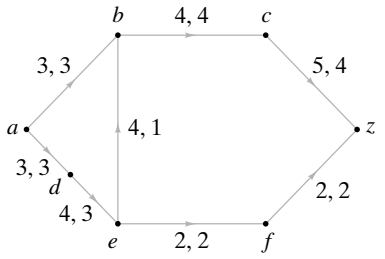


Figura 10.2.13 Después de aumentar en 1 el flujo en la trayectoria (a, d, e, f, z). El flujo es máximo.

◀

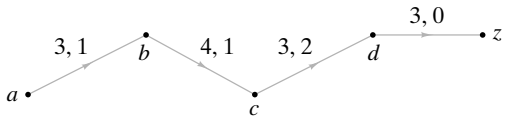
Sección de ejercicios de repaso

1. ¿Qué es un flujo máximo?
2. ¿Qué es una arista con orientación apropiada respecto a una trayectoria?
3. ¿Qué es una arista con orientación inapropiada respecto a una trayectoria?
4. ¿Cuándo se puede incrementar el flujo en una trayectoria del origen al destino?
5. Explique cómo se incrementa el flujo en las condiciones del ejercicio 4.
6. Explique cómo encontrar un flujo máximo en una red.

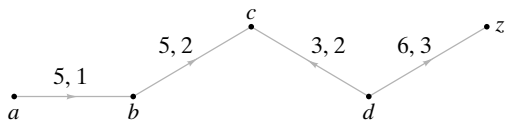
Ejercicios

En los ejercicios 1 al 3 se da una trayectoria del origen a al destino z en una red. Encuentre el incremento máximo posible en el flujo que se puede obtener modificando los flujos en las aristas de la trayectoria.

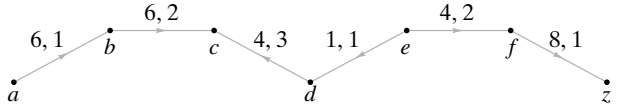
1.



2.



3.

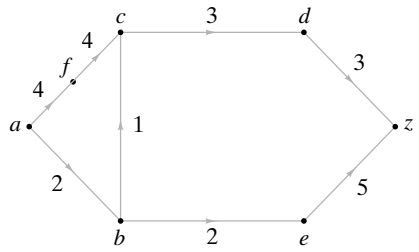


En los ejercicios 4 al 12, utilice el algoritmo 10.2.4 para encontrar un flujo máximo en cada red.

4. Figura 10.1.4

5. Figura 10.1.5

6.



7. Ejercicio 5, sección 10.1

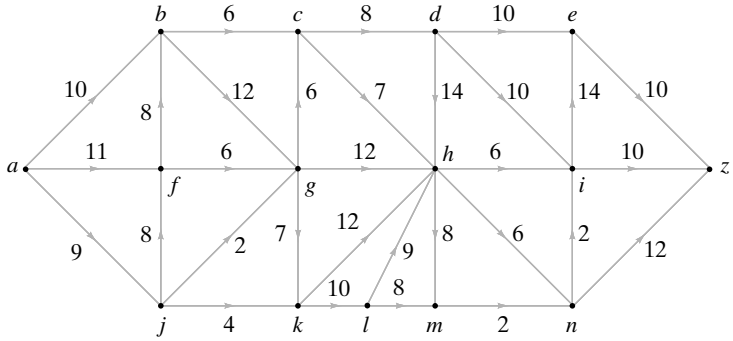
8. Ejercicio 6, sección 10.1

9. Ejercicio 7, sección 10.1

10. Ejercicio 8, sección 10.1

11. Ejercicio 9, sección 10.1

12.



En los ejercicios 13 al 18, encuentre un flujo máximo en cada red comenzando con el flujo dado.

13. Figura 10.1.2

14. Ejercicio 1, sección 10.1

15. Ejercicio 2, sección 10.1

16. Ejercicio 3, sección 10.1

17. Figura 10.1.4 con los flujos

$$F_{a,w_1} = 2, \quad F_{w_1,b} = 2, \quad F_{bA} = 0, \quad F_{cA} = 0,$$
$$F_{Az} = 0, \quad F_{a,w_2} = 0, \quad F_{w_2,b} = 0, \quad F_{bc} = 2,$$
$$F_{cB} = 4, \quad F_{Bz} = 4, \quad F_{a,w_3} = 2, \quad F_{w_3,d} = 2,$$
$$F_{dc} = 2.$$

18. Figura 10.2.4 con los flujos

$$F_{a,w_1} = 1, \quad F_{w_1,b} = 1, \quad F_{bA} = 4,$$
$$F_{cA} = 2, \quad F_{Az} = 6, \quad F_{a,w_2} = 3,$$
$$F_{w_2,b} = 3, \quad F_{bc} = 0, \quad F_{cB} = 1,$$
$$F_{Bz} = 1, \quad F_{a,w_3} = 3, \quad F_{w_3,d} = 3,$$
$$F_{dc} = 3.$$

19. Demuestre que el algoritmo 10.2.4 termina.

10.3 ➜ Teorema de flujo máximo y corte mínimo

En esta sección se muestra que a la terminación del algoritmo 10.2.4, el flujo en la red es máximo. Al mismo tiempo se definirán y analizarán los cortes en las redes.

Sea G una red y considere el flujo F a la terminación del algoritmo 10.2.4. Algunos vértices están etiquetados y otros no. Sea P ($\overline{P}$) el conjunto de vértices etiquetados (no etiquetados). (Recuerde que $\overline{P}$ denota el complemento de P). Entonces el origen a está en P y el destino z está en $\overline{P}$. El conjunto S de aristas (v, w), con $w \in P$ y $w \in \overline{P}$, se llama un **corte**, y la suma de las capacidades de las aristas en S se llama **capacidad del corte**. Se verá que este corte tiene capacidad mínima y, como un corte mínimo corresponde a un flujo máximo (Teorema 10.3.9), el flujo F es máximo. Se comienza con la definición formal de corte.

En toda esta sección, G es una red con origen a y destino z. La capacidad de la arista (i, j) es C_{ij}.

Definición 10.3.1 ▶

Un *corte* $(P, \overline{P})$ en G consiste en un conjunto P de vértices y el complemento $\overline{P}$ de P, con $a \in P$ y $z \in \overline{P}$. ◀

Ejemplo 10.3.2 ▶

Considere la red G de la figura 10.3.1. Si se hace $P = \{a, b, d\}$, entonces $\overline{P} = \{c, e, f, z\}$ y $(P, \overline{P})$ es un corte en G. Como se muestra, algunas veces se indica un corte dibujando una línea punteada para dividir los vértices.

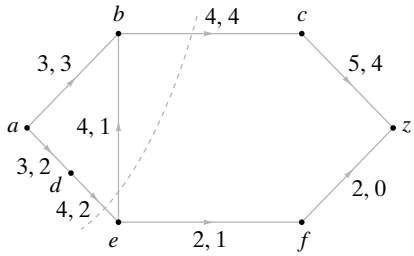


Figura 10.3.1 Un corte en una red. La línea punteada divide los vértices en dos conjuntos $P = \{a, b, d\}$ y $\overline{P} = \{c, e, f, z\}$, lo que produce el corte $(P, \overline{P})$.

◀

Ejemplo 10.3.3 ▶

La figura 10.2.10 muestra las etiquetas al terminar el algoritmo 10.2.4 para una red en particular. Si P ($\overline{P}$) denota el conjunto de vértices etiquetados (no etiquetados), se obtiene el corte mostrado en la figura 10.3.2.

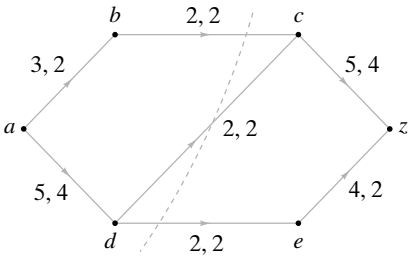


Figura 10.3.2 Una red a la terminación del algoritmo 10.2.4. El corte $(P, \overline{P})$, $P = \{a, b, d\}$, se obtiene cuando P es el conjunto de vértices etiquetados.

◀

Ahora se definirá la capacidad de un corte.

Definición 10.3.4 ▶

La capacidad del corte $(P, \overline{P})$ es el número

$$C(P, \overline{P}) = \sum_{i \in P} \sum_{j \in \overline{P}} C_{ij}.$$

◀

Ejemplo 10.3.5 ▶

La capacidad del corte de la figura 10.3.1 es

$$C_{bc} + C_{de} = 8.$$

◀

Ejemplo 10.3.6 ▶

La capacidad del corte de la figura 10.3.2 es

$$C_{bc} + C_{dc} + C_{de} = 6.$$

◀

El teorema siguiente demuestra que la capacidad de cualquier corte es siempre mayor o igual que el valor de cualquier flujo.

Teorema 10.3.7

Sea F un flujo en G y sea $(P, \overline{P})$ *una cortadura en G. Entonces la capacidad de* $(P, \overline{P})$ *es mayor o igual que el valor de F; es decir,*

$$\sum_{i\in P}\sum_{j\in\overline{P}} C_{ij} \geq \sum_{i} F_{ai}. \qquad \textbf{(10.3.1)}$$

(La notación $\sum_i$ *significa la suma sobre todos los vértices i).*

Demostración Observe que

$$\sum_{j\in P}\sum_{i\in P} F_{ji} = \sum_{j\in P}\sum_{i\in P} F_{ij},$$

ya que cualquiera de los lados de la ecuación es simplemente la suma de F_{ij} sobre toda $i, j \in P$.

Ahora

$$\begin{aligned}\sum_i F_{ai} &= \sum_{j\in P}\sum_{i} F_{ji} - \sum_{j\in P}\sum_{i} F_{ij}\\ &= \sum_{j\in P}\sum_{i\in P} F_{ji} + \sum_{j\in P}\sum_{i\in\overline{P}} F_{ji} - \sum_{j\in P}\sum_{i\in P} F_{ij} - \sum_{j\in P}\sum_{i\in\overline{P}} F_{ij}\\ &= \sum_{j\in P}\sum_{i\in\overline{P}} F_{ji} - \sum_{j\in P}\sum_{i\in\overline{P}} F_{ij} \leq \sum_{j\in P}\sum_{i\in\overline{P}} F_{ji} \leq \sum_{j\in P}\sum_{i\in\overline{P}} C_{ji}.\end{aligned}$$

Ejemplo 10.3.8 ▶

En la figura 10.3.1, el valor 5 del flujo es menor que la capacidad de 8 del corte. ◀

Un **corte mínimo** es un corte que tiene capacidad mínima.

Teorema 10.3.9

Flujo máximo, corte mínimo

Sea F un flujo en G y sea $(P, \overline{P})$ *un corte en G. Si se cumple la igualdad en (10.3.1), entonces el flujo es máximo y el corte es mínimo. Más aún, la igualdad se cumple en (10.3.1) si y sólo si*

a) $F_{ij} = C_{ij}$, para $i \in P, j \in \overline{P}$

y

b) $F_{ij} = 0$, para $i \in \overline{P}, j \in P$.

Demostración La primera afirmación se deduce directamente.

La prueba del Teorema 10.3.7 indica que la igualdad se cumple precisamente cuando

$$\sum_{j\in P}\sum_{i\in\overline{P}} F_{ij} = 0 \qquad \text{y} \qquad \sum_{j\in P}\sum_{i\in\overline{P}} F_{ji} = \sum_{j\in P}\sum_{i\in\overline{P}} C_{ji};$$

entonces la última afirmación también se cumple.

Ejemplo 10.3.10 ▶

En la figura 10.3.2, el valor del flujo y la capacidad del corte son 6 los dos; por lo tanto, el flujo es máximo y el corte es mínimo. ◀

El Teorema 10.3.9 resulta útil para demostrar que el algoritmo 10.2.4 produce un flujo máximo.

Teorema 10.3.11

Al terminar, el algoritmo 10.2.4 produce un flujo máximo. Todavía más, si P (respectivamente, $\overline{P}$) es el conjunto de vértices etiquetados (respectivamente, no etiquetados) al terminar el algoritmo 10.2.4, el corte ($P, \overline{P}$) es mínimo.

Demostración Sea P ($\overline{P}$) el conjunto de vértices etiquetados (no etiquetados) de G a la terminación del algoritmo 10.2.4. Considere una arista (i, j), donde $i \in P, j \in \overline{P}$. Como i está etiquetado, debe tenerse

$$F_{ij} = C_{ij};$$

de otra manera, se habría etiquetado j en las líneas 19 y 20. Ahora considere una arista (j, i), donde $j \in \overline{P}$, $i \in P$. Como i está etiquetado, debe tenerse

$$F_{ji} = 0;$$

de otra manera se habría etiquetado j en las líneas 25 y 26. Por el Teorema 10.3.9, el flujo al terminar el algoritmo 10.2.4 es máximo y el corte ($P, \overline{P}$) es mínimo.

Sección de ejercicios de repaso

1. ¿Qué es un corte en una red?

2. ¿Qué es la capacidad de un corte?

3. ¿Cuál es la relación entre la capacidad de un corte y el valor de cualquier flujo?

4. ¿Qué es un corte mínimo?

5. Enuncie el teorema del flujo máximo y corte mínimo.

6. Explique cómo demuestra el teorema de flujo máximo y corte mínimo que el algoritmo de la sección 8.2 encuentra correctamente un flujo máximo en una red.

Ejercicios

En los ejercicios 1 al 3, encuentre la capacidad del corte ($P, \overline{P}$). Además, determine si el corte es mínimo.

1. $P = \{a, d\}$ para el ejercicio 1, sección 10.1

2. $P = \{a, d, e\}$ para el ejercicio 2, sección 10.1

3. $P = \{a, b, c, d\}$ para el ejercicio 3, sección 10.1

En los ejercicios 4 al 16, encuentre un corte mínimo en cada red.

4. Figura 10.1.1

5. Figura 10.1.4

6. Figura 10.1.5

7. Ejercicio 1, sección 10.1

8. Ejercicio 2, sección 10.1

9. Ejercicio 3, sección 10.1

10. Ejercicio 4, sección 10.1

11. Ejercicio 5, sección 10.1

12. Ejercicio 6, sección 10.1

13. Ejercicio 7, sección 10.1

14. Ejercicio 8, sección 10.1

15. Ejercicio 9, sección 10.1

16. Ejercicio 12, sección 10.2

Los ejercicios 17 al 22 se refieren a una red G que, además de tener capacidades enteras no negativas C_{ij}, tiene requerimientos mínimos m_{ij} de flujo en enteros no negativos en las aristas. Esto es, un flujo F debe satisfacer

$$m_{ij} \leq F_{ij} \leq C_{ij}$$

para todas las aristas (i, j).

17. Dé un ejemplo de una red G, en la que $m_{ij} \leq C_{ij}$ para todas las aristas (i, j) para las que no hay un flujo.

Defina

$$C(\overline{P}, P) = \sum_{i \in \overline{P}} \sum_{j \in P} C_{ij},$$

$$m(P, \overline{P}) = \sum_{i \in P} \sum_{j \in \overline{P}} m_{ij}, \qquad m(\overline{P}, P) = \sum_{i \in \overline{P}} \sum_{j \in P} m_{ij}.$$

18. Demuestre que el valor V de cualquier flujo satisface

$$m(P, \overline{P}) - C(\overline{P}, P) \leq V \leq C(P, \overline{P}) - m(\overline{P}, P)$$

para cualquier cortadura ($P, \overline{P}$).

19. Demuestre que si existe un flujo en G, existe un flujo máximo en G con valor

$$\text{mín } \{C(P, \overline{P}) - m(\overline{P}, P) \mid (P, \overline{P}) \text{ es un corte en } G.$$

20. Suponga que G tiene un flujo F. Desarrolle un algoritmo para encontrar un flujo máximo en G.

21. Demuestre que si existe un flujo en G, existe un flujo mínimo en G con valor

$$\text{máx}\{m(P, \overline{P}) - C(\overline{P}, P) \mid (P, \overline{P}) \text{ es un corte en } G\}.$$

22. Suponga que G tiene un flujo F. Desarrolle un algoritmo para encontrar un flujo mínimo en G.

23. ¿Falso o verdadero? Si F es un flujo en una red G y $(P, \overline{P})$ es un corte en G y la capacidad de $(P, \overline{P})$ excede el valor del flujo F, entonces el corte $(P, \overline{P})$ no es mínimo y el flujo F no es máximo. Si esto es verdadero, pruébelo; de otra manera, dé un contraejemplo.

10.4 ➜ Acoplamiento

En esta sección se estudia el problema de acoplar elementos de un conjunto con elementos de otro conjunto. Se verá que este problema se reduce a encontrar un flujo máximo en una red. Se comienza con un ejemplo.

Ejemplo 10.4.1 ▶

Suponga que cuatro personas A, B, C y D llenan su solicitud de trabajo para cinco puestos J_1, J_2, J_3, J_4 y J_5. Suponga que el candidato A está calificado para los puestos J_2 y J_5; el candidato B está calificado para los puestos J_2 y J_5; el candidato C está calificado para los puestos J_1, J_3, J_4 y J_5, y el candidato D está calificado para los puestos J_2 y J_5. ¿Es posible encontrar un puesto para cada persona?

La situación se ha modelado con la gráfica de la figura 10.4.1. Los vértices representan los candidatos y los puestos. Una arista conecta un candidato con un puesto para el que está calificado. Se puede demostrar que no es posible asignar un puesto para cada candidato si se consideran los candidatos A, B y D, que están calificados para los puestos J_2 y J_5. Si A y B se asignan cada uno a un puesto, no queda lugar para D. Por lo tanto, no existen asignaciones para A, B, C y D.

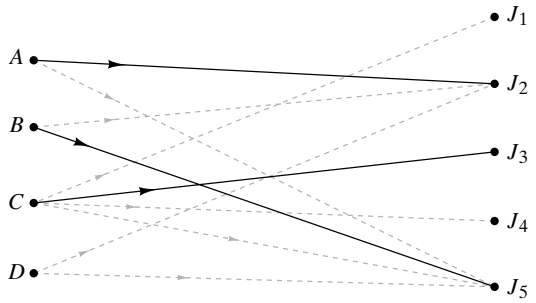


Figura 10.4.1 Candidatos (A, B, C, D) y puestos (J_1, J_2, J_3, J_4, J_5). Una arista conecta a un candidato con un puesto para el que está calificado. Las líneas continuas muestran una asignación máxima (es decir, el máximo número de candidatos tienen un puesto de trabajo).

◀

En el ejemplo 10.4.1 una asignación consiste en encontrar puestos para personas calificadas. Una asignación máxima para la gráfica de la figura 10.4.1 se muestra con las líneas continuas. Una asignación completa encuentra puestos para todos. Se mostró que la gráfica de la figura 10.4.1 no tiene una asignación completa. Las siguientes son las definiciones formales.

Definición 10.4.2 ▶

Sea G una gráfica bipartita dirigida con conjuntos ajenos de vértices V y W en los que las aristas se dirigen de vértices en V a vértices en W. (Cualquier vértice en G está en V o en W). Una *asignación* para G es un conjunto de aristas E que no tienen vértices en común. Una *asignación máxima* para G es una asignación E, donde E contiene el máximo número de aristas. Una *asignación completa* para G es una asignación E que tiene la propiedad de que si $v \in V$, entonces $(v, w) \in E$ para alguna $w \in W$. ◀

Ejemplo 10.4.3 ▶

La asignación para la gráfica de la figura 10.4.2, mostrada con líneas continuas, es una asignación máxima y completa.

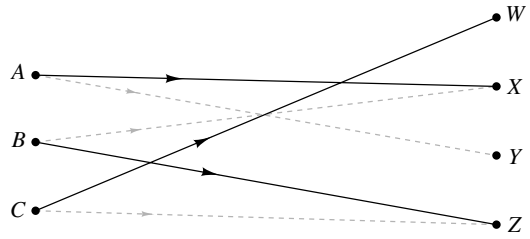


Figura 10.4.2 Las líneas continuas muestran un acoplamiento máximo (se usa el número máximo de aristas) y completa (cada uno de A, B y C tienen pareja).

◀

El siguiente ejemplo ilustra cómo se puede modelar el problema de acoplamiento como una problema de redes.

Ejemplo 10.4.4 ▶

Una red de acoplamiento

Modele el problema de asignación del ejemplo 10.4.1 como una red.

Primero se asigna una capacidad de 1 a cada arista en la gráfica de la figura 10.4.1 (vea la figura 10.4.3). Después se agrega un superorigen a y aristas de capacidad 1 de a a cada vértice A, B, C y D. Por último, se introduce un superdestino z y aristas de capacidad 1 de cada vértice J_1, J_2, J_3, J_4 y J_5 a z. Una red como la de la figura 10.4.3 se llama **red de asignación** o **red de acoplamiento**.

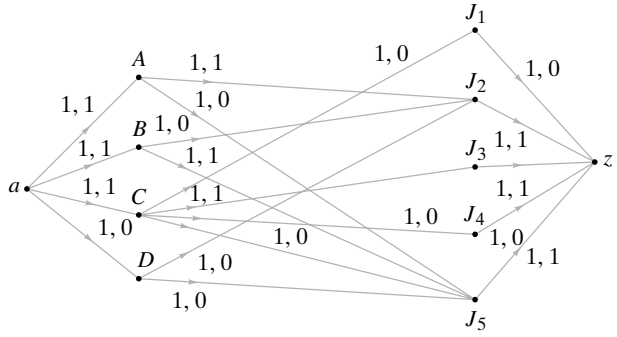


Figura 10.4.3 El problema de acoplamiento (figura 10.4.1) como un problema de red de acoplamiento.

◀

El siguiente teorema relaciona las redes de acoplamiento y los flujos.

Teorema 10.4.5

Sea G una gráfica bipartita dirigida con conjuntos ajenos de vértices V y W donde las aristas están dirigidas de los vértices en V a los vértices en W. (Cualquier vértice en G está en V o bien en W).

a) Un flujo en la red de acoplamiento da una asignación en G. El vértice $v \in V$ se asigna al vértice $w \in W$ si y sólo si el flujo en la arista (v, w) es 1.

b) Un flujo máximo corresponde a un acoplamiento máximo.

c) Un flujo cuyo valor es $|V|$ corresponde a un acoplamiento completo.

Demostración Sea a (z) el origen (destino) en la red de acoplamiento y suponga que se da un flujo.

Suponga que la arista (v, w), $v \in V$, $w \in W$, tiene un flujo de 1. La única arista que llega al vértice v es (a, v). Esta arista debe tener un flujo de 1; entonces el flujo que llega al vértice v es 1. Como el flujo que sale de v también es 1, la única arista de la forma (v, x) que tiene flujo 1 es (v, w).

De manera similar, la única arista de la forma (x, w) con flujo 1 es (v, w). Por lo tanto, si E es el conjunto de aristas de la forma (v, w) cuyo flujo es 1, los miembros de E no tienen vértices en común; entonces, E es un acoplamiento para G.

Los incisos *b)* y *c)* se deducen del hecho de que el número de vértices en V acoplados es igual al valor del flujo correspondiente.

Como un flujo máximo da un acoplamiento máximo, el algoritmo 10.2.4 aplicado a una red de acoplamiento produce un acoplamiento máximo. En la práctica, la implementación del algoritmo 10.2.4 se simplifica usando la matriz de adyacencia de la gráfica (vea el ejercicio 11).

Ejemplo 10.4.6 ▶

El acoplamiento de la figura 10.4.1 se representa como un flujo en la figura 10.4.3. Como el flujo es máximo, el acoplamiento es máximo. ◀

Ahora se estudiará la existencia de un acoplamiento completo en una gráfica bipartita dirigida G con conjuntos de vértices V y W. Si $S \subseteq V$, sea

$$R(S) = \{w \in W \mid v \in S \text{ y } (v, w) \text{ es una arista en } G\}.$$

Suponga que G tiene un acoplamiento completo. Si $S \subseteq V$, debe tenerse

$$|S| \leq |R(S)|.$$

Resulta que si $|S| \leq |R(S)|$ para todos los subconjuntos S de V, entonces G tiene un acoplamiento completo. El matemático inglés Philip Hall fue el primero en dar este resultado que se conoce como **teorema de Hall del matrimonio**, dado que si V es un conjunto de hombres y W es un conjunto de mujeres y existen aristas de $v \in V$ a $w \in W$ si v y w son compatibles, el teorema da una condición para que cada hombre se pueda casar con una mujer compatible.

WWW

Teorema 10.4.7

Teorema de Hall del matrimonio

Sea G una gráfica dirigida bipartita con conjuntos ajenos de vértices V y W en los que las aristas están dirigidas de los vértices en V a los vértices en W. (Cualquier vértice en G pertenece a V o bien a W). Existe un acoplamiento completo en G si y sólo si

$$|S| \leq |R(S)| \quad \textit{para todo } S \subseteq V. \tag{10.4.1}$$

Demostración Se ha señalado que si hay un acoplamiento completo en G, la condición (10.4.1) se cumple.

Suponga que la condición (10.4.1.) se cumple. Sea $n = |V|$ y sea $(P, \overline{P})$ un corte mínimo en la red de asignación. Si se puede demostrar que la capacidad de este corte es n, un flujo máximo tendrá el valor de n. La asignación que corresponde a este flujo máximo será un acoplamiento completo.

Se da un argumento por contradicción. Suponga que la capacidad del corte mínimo $(P, \overline{P})$ es menor que n. La capacidad de este corte es el número de aristas en el conjunto

$$E = \{(x, y) \mid x \in P, y \in \overline{P}\}$$

(vea la figura 10.4.4). Un miembro de E es de uno de los tres tipos:

Tipo I: $(a, v), v \in V$;

Tipo II: $(v, w), v \in V, w \in W$;

Tipo III: $(w, z), w \in W$.

Se estimará el número de aristas de cada tipo.

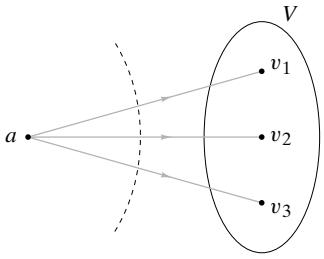


Figura 10.4.5 Prueba del Teorema 10.4.7 para $n = 3$. Si $V \subseteq \overline{P}$, como se muestra, la capacidad del corte es n. Como se está suponiendo que la capacidad del corte es menor que n, este caso no puede ocurrir. Por lo tanto, $V \cap P$ es no vacío.

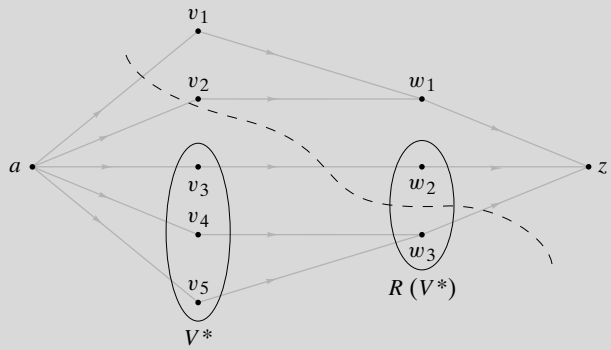


Figura 10.4.4 Prueba del Teorema 10.4.7.
$V = \{v_1, v_2, v_3, v_4, v_5\}$; $n = |V| = 5$; $W = \{w_1, w_2, w_3\}$; el corte es $(P, \overline{P})$, donde $P = \{a, v_3, v_4, v_5, w_3\}$;.
$V^* = V \cap P = \{v_3, v_4, v_5\}$; $R(V^*) = \{w_2, w_3\}$;
$W_1 = R(V^*) \cap P = \{w_3\}$; $W_2 = R(V^*) \cap \overline{P} = \{w_2\}$;
$E = \{(a, v_1), (a, v_2), (v_3, w_2), (w_3, z)\}$.
La capacidad del corte es $|E| = 4 < n$. Las aristas tipo I son (a, v_1) y (a, v_2). La arista (v_3, w_2) es la única de tipo II y la arista (w_3, z) es la única arista de tipo III.

Si $V \subseteq \overline{P}$, la capacidad del corte es n (vea la figura 10.4.5); entonces

$$V^* = V \cap P$$

es no vacío. Se deduce que hay $n - |V^*|$ aristas en E de tipo I.

Se hace una partición de $R(V^*)$ en los conjuntos

$$W_1 = R(V^*) \cap P \qquad \text{y} \qquad W_2 = R(V^*) \cap \overline{P}.$$

Entonces, existen al menos $|W_1|$ aristas de tipo III en E. Así, hay menos de

$$n - (n - |V^*|) - |W_1| = |V^*| - |W_1|$$

aristas de tipo II en E. Como cada miembro de W_2 contribuye cuando mucho con una arista tipo II,

$$|W_2| < |V^*| - |W_1|.$$

Entonces

$$|R(V^*)| = |W_1| + |W_2| < |V^*|,$$

que contradice la expresión (10.4.1). Por lo tanto, existe un acoplamiento completo.

Ejemplo 10.4.8 ▶

Para la gráfica de la figura 10.4.1, si $S = \{A, B, D\}$, se tiene $R(S) = \{J_2, J_5\}$ y

$$|S| = 3 > 2 = |R(S)|.$$

Por el Teorema 10.4.7, no hay un acoplamiento completo para la gráfica de la figura 10.4.1. ◀

Ejemplo 10.4.9 ▶

Se tienen n computadoras y n discos duros. Cada computadora es compatible con $m > 0$ discos duros y cada disco duro es compatible con m computadoras. ¿Es posible acoplar cada computadora con un disco duro compatible?

Sea V el conjunto de computadoras y W el conjunto de discos duros. Existe una arista de $v \in V$ a $w \in W$ si v y w son compatibles. Observe que todo vértice tiene grado m.

Sea $S = \{v_1, \ldots v_k\}$ un subconjunto de V. Entonces existen km aristas desde el conjunto S. Si $R(S) = \{w_1, \ldots, w_j\}$, entonces $R(S)$ recibe cuando mucho jm aristas desde S. Por lo tanto,

$$km \leq jm.$$

Ahora bien

$$|S| = k \leq j = |R(S)|.$$

Por el Teorema 10.4.7 hay una acoplamiento completo. Por lo tanto, sí es posible acoplar cada computadora con un disco duro compatible. ◀

Sección de ejercicios de repaso

1. ¿Qué es un acoplamiento?

2. ¿Qué es un acoplamiento máximo?

3. ¿Qué es un acoplamiento completo?

4. ¿Cuál es la relación entre flujos y acoplamiento?

5. Enuncie el teorema de Hall del matrimonio.

Ejercicios

1. Demuestre que el flujo de la figura 10.4.3 es máximo exhibiendo un corte mínimo con capacidad 3.

2. Encuentre el flujo correspondiente al acoplamiento de la figura 10.4.2. Demuestre que este flujo es máximo exhibiendo un corte mínimo con capacidad 3.

Los ejercicios 3 al 7 se refieren a la figura 10.4.1, donde se invierte la dirección de las aristas de manera que estén dirigidas de los puestos a los candidatos.

3. ¿Qué representa un acoplamiento?

4. ¿Qué representa un acoplamiento máximo?

5. Muestre un acoplamiento máximo.

6. ¿Qué representa un acoplamiento completo?

7. ¿Existe un acoplamiento completo? Si la hay, muestre una. Si no hay un acoplamiento completo, explique por qué.

8. El candidato A está calificado para los trabajos J_1 y J_4; B está calificado para los puestos J_2, J_3 y J_6; C está calificado para los puestos J_1, J_3, J_5 y J_6; D puede estar en los puestos J_1, J_3 y J_4; y E está calificado para los puestos J_1, J_3 y J_6.

a) Modele esta situación como una red de acoplamiento.

b) Use el algoritmo 10.2.4 para encontrar un acoplamiento máximo.

c) ¿Existe un acoplamiento completo?

9. El candidato A está calificado para los trabajos J_1, J_2, J_4 y J_5; B está calificado para los puestos J_1, J_4 y J_5; C está calificado para los puestos J_1, J_4, y J_5; D puede estar en los puestos J_1, y J_5; E está calificado para los puestos J_2, J_3 y J_5; y F está calificado para los puestos J_4 y J_5. Conteste los incisos *a)* al *c)* del ejercicio 8 para esta situación.

10. El candidato A está calificado para los trabajos J_1, J_2 y J_4; B está calificado para los puestos J_3, J_4, J_5 y J_6; C está calificado para los puestos J_1 y J_5; D puede estar en los puestos J_1, J_3, J_4 y J_8; E está calificado para los puestos J_1, J_2, J_4, J_6 y J_8; F está calificado para los puestos J_4 y J_6; y G está calificado para J_3, J_5 y J_7. Conteste los incisos *a)* y *c)* del ejercicio 8 para esta situación.

11. Cinco estudiantes, V, W, X, Y y Z, son miembros de cuatro comités, C_1, C_2 C_3 y C_4. Los miembros de C_1 son V, X y Y; los integrantes de C_2 son X y Z; los de C_3 son V, Y y Z; y los miembros de C_4 son V, W, X y Z. Cada comité debe enviar un representante a la administración. Ningún estudiante puede representar a dos comités.

a) Modele esta situación como red de acoplamiento.

b) ¿Cuál es la interpretación de un acoplamiento máximo?

c) ¿Cuál es la interpretación de un acoplamiento completo?

d) Use al algoritmo 10.2.4 para encontrar un acoplamiento máximo.

e) ¿Existe un acoplamiento completo?

12. Demuestre que con un orden apropiado de los vértices, la matriz de adyacencia de una gráfica bipartita se puede escribir

$$\begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix},$$

donde 0 es una matriz con sólo ceros y A^T es la transpuesta de la matriz A.

En los ejercicios 13 al 15, G es una gráfica bipartita, A es la matriz del ejercicio 12, y F es un flujo en la red de acoplamiento asociada. Etiquete cada elemento de A que represente una arista con flujo 1.

13. ¿Qué tipo de etiquetas corresponde a un acoplamiento?

14. ¿Qué tipo de etiquetas corresponde a un acoplamiento completo?

15. ¿Qué tipo de etiquetas corresponde a un acoplamiento máximo?

16. Enuncie de nuevo el algoritmo 10.2.4, aplicado a una red de acoplamiento, en términos de las operaciones en la matriz A del ejercicio 12.

Sea G una gráfica bipartita dirigida con conjuntos ajenos de vértices V y W en los que las aristas están dirigidas de los vértices de V a los vértices de W. (Cualquier vértice de G está en V o bien esta en W). La deficiencia se define como

$$\delta(G) = \text{máx}\{|S| - |R(S)| \mid S \subseteq V\}.$$

17. Demuestre que G tiene un acoplamiento completo si y sólo si $\delta(G) = 0$.

★ **18.** Demuestre que el máximo número de vértices en V que se pueden acoplar a vértices en W es $|V| - \delta(G)$.

19. ¿Falso o verdadero? Cualquier acoplar está contenido en un acoplamiento máximo. Si es verdadero, pruébelo; si es falso, dé un contraejemplo.

Rincón de solución de problemas

Acoplamiento

Problema

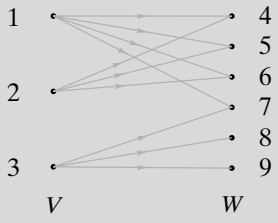
Sea G una gráfica dirigida bipartita con conjuntos ajenos de vértices V y W en los que las aristas están dirigidas de los vértices en V a los vértices en W. (Cualquier vértice en G está en V o bien está en W). Sea M_W el grado máximo que ocurre entre los vértices en W, y sea m_V el grado mínimo que ocurre entre los vértices en V. Demuestre que si $0 < M_W \leq m_V$, entonces G tiene un acoplamiento completo.

Cómo atacar el problema

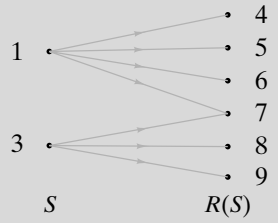
El teorema de Hall del matrimonio (Teorema 10.4.7) dice que una gráfica bipartita dirigida con conjuntos ajenos de vértices V y W tiene un acoplamiento completo si y sólo si $|S| \leq |R(S)|$ para toda $S \subseteq V$. Por tanto, una manera posible de resolver el problema es demostrar que la condición dada $M_W \leq m_V$ implica que $|S| \leq |R(S)|$ para toda $S \subseteq V$.

Cómo encontrar una solución

La meta es probar que si $M_W \leq m_V$, entonces $|S| \leq |R(S)|$ para toda $S \subseteq V$. Se comienza con el ejemplo de una gráfica G para la que $M_W \leq m_V$; de hecho, se puede hacer que $M_W = 2$ y $m_V = 3$:



Considere un ejemplo de subconjunto $S = \{1, 3\}$ de V y las aristas incidentes en los vértices en S:



El hecho de que el grado mínimo de los vértices en V sea 3 significa que para *cualquier* subconjunto S de V, *al menos* tres aristas inciden en cada vértice en S. En general, habrá al menos $3|S| = m_V|S|$ aristas incidentes en los vértices en S. En el ejemplo, $3|S| = 6$, pero hay en realidad 7 aristas incidentes en los vértices en S. La expresión $m_V|S|$ siempre es una *cota inferior* para el número de aristas incidentes en los vértices en S.

El hecho de que el grado máximo de los vértices en W sea 2 significa que para *cualquier* subconjunto S de V, *cuando más* dos aristas inciden en cada vértice en $R(S)$. En general, habrá cuando mucho $2|R(S)| = M_W|R(S)|$ aristas incidentes en los vértices en $R(S)$. En el ejemplo, $2|R(S)| = 12$, pero existen en realidad 10 aristas incidentes en los vértices en $R(S)$. Como las aristas incidentes en los vértices en S son un subconjunto de las aristas incidentes en los vértices en $R(S)$, la expresión $M_W|R(S)|$ siempre es una *cota superior* para el número de aristas incidentes en los vértices en S.

Se tienen dos maneras de estimar el número de aristas incidentes en los vértices en S. La primera, usando S, da una cota inferior $m_V|S|$ sobre el número de aristas, y la segunda, usando $R(S)$, da una cota superior $M_W|R(S)|$ sobre el número de aristas. Al comparar estas estimaciones se obtiene la desigualdad

$$m_V|S| \leq M_W|R(S)|.$$

No se puede deducir $|S| \leq |R(S)|$, pero no se ha usado la hipótesis todavía

$$M_W \leq m_V$$

Al combinar las últimas dos desigualdades se tiene

$$m_V|S| \leq M_W|R(S)| \leq m_V|R(S)|.$$

Si ahora se cancela m_V en ambos lados de la desigualdad, se obtiene

$$|S| \leq |R(S)|,$$

que es justo la desigualdad que se quiere probar.

Solución formal

Sea $S \subseteq V$. Cada vértice en S incide en al menos $m_V|S|$ aristas; entonces, existen al menos $m_V|S|$ aristas incidentes en los vértices de S. Cada vértice en $R(S)$ incide en cuando mucho M_W aristas; entonces, existen cuando mucho $M_W|R(S)|$ aristas incidentes en los vértices de $R(S)$. Se deduce que $m_V|S| \leq M_W|R(S)|$. Como $M_W \leq m_V$, $|R(S)|M_W \leq |R(S)|m_V$. Por lo tanto, $m_V|S| \leq m_v|R(S)|$, y $|S| \leq |R(S)|$. Por el Teorema 10.4.7, G tiene un acoplamiento completo.

Resumen de las técnicas para la solución de problemas

- Observe ejemplos de gráficas.
- Al ver ejemplos, es buena idea asignar valores diferentes a los parámetros del problema para poder manejarlos. (En el ejemplo se estableció que $M_W = 2$ y $m_V = 3$).
- Intente reducir las condiciones dadas a las que se usan en un teorema útil. (Se redujeron las condiciones dadas en este problema a las condiciones indicadas en el teorema de Hall del matrimonio).
- Algunas veces, una desigualdad puede probarse estimando el tamaño de algún conjunto de dos maneras diferentes. Si una estimación da una cota superior M y otra da una cota inferior m, se deduce que $m \leq M$.

Comentarios

La última técnica de solución de problemas resumida proporciona un método para probar una desigualdad. De manera similar, en ocasiones una desigualdad se puede probar contando el número de elementos en algún conjunto de dos maneras diferentes. Si una manera de contar da c_1 y la otra c_2, se deduce que $c_1 = c_2$. Estas técnicas se emplean ampliamente y jamás podría exagerarse su utilidad. Por ejemplo, en la sección 6.2 se derivó una fórmula para $C(n, r)$ contando, de dos maneras diferentes, el número de permutaciones r de un conjunto de n elementos.

Ejercicio

1. Dé un ejemplo de una gráfica bipartita G que tenga un acoplamiento completo pero que no satisfaga la condición $M_W \leq m_V$.

Notas

Las referencias generales que contienen secciones acerca de modelos de redes son [Berge; Deo; Liu, 1968, 1985; y Tucker]. El trabajo clásico sobre redes es [Ford]; muchos resultados de redes, en especial los primeros, se deben a Ford y Fulkerson, los autores de este libro. [Tarjan] analiza los algoritmos de flujo en redes y los detalles de su implementación.

Vea en [Bachelis] una prueba directa agradable del teorema de Hall del matrimonio (Teorema 10.4.7) mediante inducción matemática.

El problema de encontrar un flujo máximo en una red G, con origen a, destino z y capacidades C_{ij}, se puede establecer como sigue:

$$\text{maximizar} \sum_j F_{aj}$$

sujeta a

$$0 \leq F_{ij} \leq C_{ij} \qquad \text{para toda } i, j,$$

$$\sum_i F_{ij} = \sum_i F_{ji} \qquad \text{para toda } j.$$

Un problema de este tipo es un ejemplo de un **problema de programación lineal**. En un problema de programación lineal se quiere maximizar (o minimizar) una expresión lineal, como $\sum_j F_{aj}$,, sujeta a restricciones de igualdades y desigualdades lineales, como $0 \leq F_{ij} \leq C_{ij}$ y $\sum_i F_{ij} = \sum_i F_{ji}$. Aunque el **algoritmo símplex** suele ser una manera eficiente de resolver un problema general de programación lineal, los problemas de redes de transporte se resuelven con mayor eficiencia si se usa el algoritmo 10.2.4. Vea en [Hillier] una exposición del algoritmo símplex.

Suponga que para cada arista (i, j) en una red G, c_{ij} representa el costo del flujo de una unidad a través de la arista (i, j). Suponga que se desea encontrar el flujo máximo, con un costo mínimo.

$$\sum_i \sum_j c_{ij} F_{ij}.$$

Este problema, llamado **problema de transporte**, de nuevo es un problema de programación lineal y, al igual que con el problema del flujo máximo, se puede usar un algoritmo específico para obtener una solución que, en general, sea más eficiente que el algoritmo símplex (vea [Hillier]).

Repaso del capítulo

Sección 10.1

1. Red (transporte)
2. Origen
3. Destino
4. Capacidad
5. Flujo en una red
6. Flujo en una arista
7. Flujo que entra a un vértice
8. Flujo que sale de un vértice
9. Conservación del flujo

10. Dado un flujo F en una red, el flujo que sale del origen es igual al flujo que llega al destino. Este valor común se llama valor del flujo F.
11. Superorigen
12. Superdestino

Sección 10.2

13. Flujo máximo
14. Arista con orientación apropiada respecto a la trayectoria
15. Arista con orientación inapropiada respecto a la trayectoria
16. Cómo incrementar el flujo en una trayectoria del origen al destino cuando *a)* para cada arista con orientación apropiada el flujo es menor que la capacidad y *b)* cada arista con orientación inapropiada tiene flujo positivo (vea el Teorema 10.2.3)
17. Cómo encontrar un flujo máximo en una red (algoritmo 10.2.4)

Sección 10.3

18. Corte en una red
19. Capacidad de un corte
20. La capacidad de cualquier corte es mayor o igual que el valor de cualquier flujo (Teorema 10.3.7).
21. Cortadura mínima
22. Teorema de flujo máximo, corte mínimo (Teorema 10.3.9)
23. A la terminación del algoritmo de flujo máximo, algoritmo 10.2.4, el conjunto de vértices etiquetados define un corte mínimo.

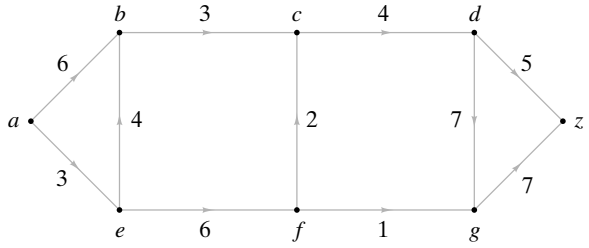
Sección 10.4

24. Acoplamiento
25. Acoplamiento máximo
26. Asignación completo
27. Red de acoplamiento
28. Relación entre flujos y acoplamientos (Teorema 10.4.5)
29. Teorema de Hall del matrimonio (Teorema 10.4.7)

Autoevaluación del capítulo

Sección 10.1

Los ejercicios 1 al 4 se refieren a la siguiente red. Las capacidades se indican en las aristas.



1. Explique por qué

$$F_{a,e} = 2, \quad F_{e,b} = 2, \quad F_{b,c} = 3, \quad F_{c,d} = 3, \quad F_{d,z} = 3, \quad F_{a,b} = 1,$$

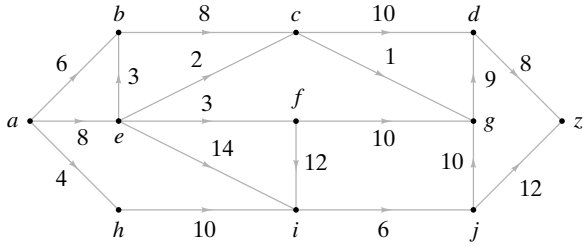
con todos los otros $F_{x,y} = 0$, constituye un flujo.
2. ¿Cuál es el flujo que llega a b?
3. ¿Cuál es el flujo que sale de c?
4. ¿Cuál es el valor del flujo F?

Sección 10.2

5. Para el flujo del ejercicio 1, encuentre una trayectoria de a a z que satisfaga lo siguiente: *a)* para cada arista con la orientación apropiada el flujo es menor que la capacidad y *b)* cada arista con la orientación inapropiada tiene flujo positivo.
6. Modifique sólo los flujos en las aristas de la trayectoria del ejercicio 5 para encontrar un flujo con un valor mayor que F.

7. Utilice el algoritmo 10.2.4 para encontrar un flujo máximo en la red del ejercicio 1 (comenzando con el flujo compuesto por flujos iguales a cero en cada arista).

8. Utilice el algoritmo 10.2.4 para encontrar un flujo máximo en la siguiente red (comenzando con el flujo compuesto por flujos iguales a cero en cada arista).



Sección 10.3

9. En los incisos *a)* al *d)*, conteste verdadero si la afirmación es cierta para toda red; de otra manera, conteste falso.

a) Si la capacidad de un corte en una red es igual a *Ca*, entonces el valor de cualquier flujo es menor o igual que *Ca*.

b) Si la capacidad de un corte en una red es igual a *Ca*, entonces el valor de cualquier flujo es mayor o igual que *Ca*.

c) Si la capacidad de un corte en una red es igual a *Ca*, entonces el valor de algún flujo es mayor o igual que *Ca*.

d) Si la capacidad de un corte en una red es igual a *Ca*, entonces el valor de algún flujo es menor o igual que *Ca*.

10. Encuentre la capacidad del corte $(P, \overline{P})$ en la red del ejercicio 1, donde $P = \{a, b, e, f\}$.

11. ¿Es mínimo el corte $(P, \overline{P})$, $P = \{a, b, e, f\}$, en la red del ejercicio 1? Explique.

12. Encuentre un corte mínimo en la red del ejercicio 8.

Sección 10.4

Los ejercicios 13 al 16 se refieren a la siguiente situación. El candidato A está calificado para los puestos J_2, J_4 y J_5; el candidato B puede hacer los trabajos J_1 y J_3; el candidato C está calificado para los trabajos J_1, J_3 y J_5; y el candidato D para los puestos J_3 y J_5.

13. Modele la situación como una red de acoplamiento.

14. Use el algoritmo 10.2.4 para encontrar un acoplamiento máximo.

15. ¿Existe un acoplamiento completo?

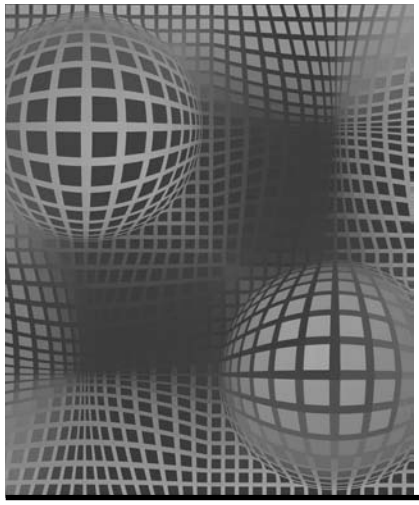
16. Encuentre una cortadura mínima en la red de acoplamiento.

Ejercicios para computadora

1. Escriba un programa que acepte como entrada una red con un flujo dado y produzca todas las trayectorias posibles del origen al destino para las que se pueda aumentar el flujo.

2. Implemente el algoritmo 10.2.4 que encuentra un flujo máximo en una red como un programa. Haga que el programa produzca el corte mínimo, así como el flujo máximo.

3. Escriba un programa que calcule la deficiencia en una red.



Capítulo 11

ÁLGEBRAS BOOLEANAS Y CIRCUITOS COMBINATORIOS

Él es indigno, deshonesto, egoísta, engañoso, despreciable; pero él está allá y yo estoy acá. Dicen que él es normal y yo no. Bueno, si eso es normal, no lo quiero.

DE MILAGRO EN LA CALLE 34

Varias definiciones honran al matemático del siglo XIX George Boole; álgebra booleana, funciones booleanas, expresión booleana y anillo booleano, por nombrar unas cuantas. Boole es una de las personas de una larga cadena histórica que se preocuparon por formalizar y mecanizar el proceso del pensamiento lógico. De hecho, en 1854 Boole escribió un libro titulado *The Laws of Thought* (*Las leyes del pensamiento*). La contribución de Boole fue el desarrollo de una teoría de lógica que usa símbolos en lugar de palabras. Un análisis del trabajo de Boole se encuentra en [Hailperin].

Casi un siglo después del trabajo de Boole, C. E. Shannon observó en 1938 (vea [Shannon]) que el álgebra booleana se podía usar para analizar circuitos eléctricos. Fue así como el álgebra booleana se convirtió en una herramienta indispensable para el análisis y WWW diseño de las computadoras electrónicas en las siguientes décadas. En este capítulo se exploran las relaciones del álgebra booleana con los circuitos.

11.1 ➜ Circuitos combinatorios

En una computadora digital sólo hay dos posibilidades, que se escriben como 0 y 1, para el objeto indivisible más pequeño. En última instancia, todos los programas y datos se pueden reducir a combinaciones de bits. A través de los años se ha usado una variedad de dispositivos en las computadoras digitales para almacenar bits. Los circuitos electrónicos permiten que estos dispositivos de almacenamiento se comuniquen entre sí. Un bit en una parte del circuito es trasmitido a otra parte del circuito como un voltaje. Entonces se necesitan dos niveles de voltaje; por ejemplo, un voltaje alto puede comunicar un 1 y un voltaje bajo, un 0.

WWW

En esta sección analizaremos los **circuitos combinatorios**. La salida de un circuito combinatorio se define de manera única para cada combinación de entradas. Un circuito de este tipo carece de memoria; las entradas anteriores y el estado del sistema no afectan su salida. Los circuitos para los cuales la salida es una función, no sólo de las entradas sino también del estado del sistema, se llaman **circuitos secuenciales** y se estudiarán en el capítulo 12.

Los circuitos combinatorios se pueden construir usando dispositivos de estado sólido, llamados **compuertas**, que son capaces de cambiar los niveles de voltaje (bits). Se comenzará por analizar las compuertas AND (y), OR (o) y NOT (no).

Definición 11.1.1 ▶

Una *compuerta* AND recibe entradas x_1 y x_2, donde x_1 y x_2 son bits, y produce una salida denotada por $x_1 \wedge x_2$, donde

$$x_1 \wedge x_2 = \begin{cases} 1 & \text{si } x_1 = 1 \text{ y } x_2 = 1 \\ 0 & \text{de otra manera.} \end{cases}$$

Una compuerta AND se dibuja como se indica en la figura 11.1.1.

Figura 11.1.1 Compuerta AND.

◀

Definición 11.1.2 ▶

Una *compuerta* OR recibe entradas x_1 y x_2, donde x_1 y x_2 son bits, y produce una salida denotada por $x_1 \vee x_2$, donde

$$x_1 \vee x_2 = \begin{cases} 1 & \text{si } x_1 = 1 \text{ o } x_2 = 1 \\ 0 & \text{de otra manera.} \end{cases}$$

Una compuerta OR se dibuja como se indica en la figura 11.1.2.

Figura 11.1.2 Compuerta OR.

◀

Definición 11.1.3 ▶

Una *compuerta* NOT (o *inversor*) recibe una entrada x, donde x es un bit, y produce una salida denotada por $\overline{x}$, donde

$$\overline{x} = \begin{cases} 1 & \text{si } x = 0 \\ 0 & \text{si } x = 1. \end{cases}$$

Una compuerta NOT se dibuja como se indica en la figura 11.1.3. ◀

Figura 11.1.3 Compuerta NOT.

La **tabla lógica** de un circuito combinatorio lista todas las entradas posibles junto con las salidas producidas.

Ejemplo 11.1.4 ▶

A continuación aparecen las tablas lógicas para los circuitos AND, OR y NOT básicos (figuras 11.1.1 a la 11.1.3).

x_1	x_2	$x_1 \wedge x_2$
1	1	1
1	0	0
0	1	0
0	0	0

x_1	x_2	$x_1 \vee x_2$
1	1	1
1	0	1
0	1	1
0	0	0

x	$\overline{x}$
1	0
0	1

Se observa que realizar la operación AND (OR) es lo mismo que tomar el mínimo (máximo) de dos bits x_1 y x_2. ◀

Ejemplo 11.1.5 ▶

El circuito de la figura 11.1.4 es un ejemplo de un circuito combinatorio, ya que la salida y se define de manera única para cada combinación de entradas x_1, x_2 y x_3.

Figura 11.1.4 Un circuito combinatorio.

La tabla lógica para este circuito combinatorio es la siguiente.

x_1	x_2	x_3	y
1	1	1	0
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	0
0	0	0	1

Observe que se listan todas las combinaciones posibles de los valores de las entradas x_1, x_2 y x_3. Para un conjunto determinado de entradas, el valor de la salida y se calcula rastreando el flujo a través del circuito. Por ejemplo, la cuarta línea de la tabla da el valor de la salida y para los valores de entrada

$$x_1 = 1, \quad x_2 = 0, \quad x_3 = 0.$$

Si $x_1 = 1$ y $x_2 = 0$, la salida de la compuerta AND es 0 (vea la figura 11.1.5). Como $x_3 = 0$, las entradas a la compuerta OR son ambas 0. Por lo tanto, la salida de la compuerta OR es 0. Como la entrada a la compuerta NOT es 0, se produce una salida $y = 1$.

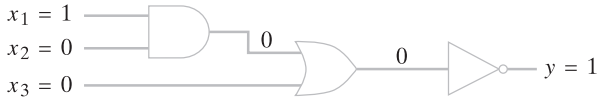


Figura 11.1.5 Circuito de la figura 11.1.4 cuando $x_1 = 1$ y $x_2 = x_3 = 0$.

◀

Ejemplo 11.1.6 ▶

El circuito de la figura 11.1.6 no es un circuito combinatorio porque la salida y no es única para cada combinación de entradas x_1 y x_2. Por ejemplo, suponga que $x_1 = 1$ y $x_2 = 0$. Si la salida de la compuerta AND es 0, entonces $y = 0$. Por otro lado, si la salida de la compuerta AND es 1, entonces $y = 1$. Un circuito de este tipo sirve para almacenar un bit.

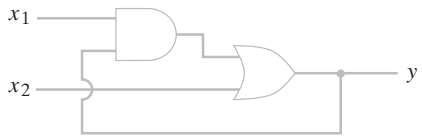


Figura 11.1.6 Un circuito que no es combinatorio.

◀

Ejemplo 11.1.7 ▶

Los circuitos combinatorios individuales se pueden interconectar. Es posible combinar los circuitos combinatorios C_1, C_2 y C_3 de la figura 11.1.7, como se muestra, para obtener el circuito combinatorio C.

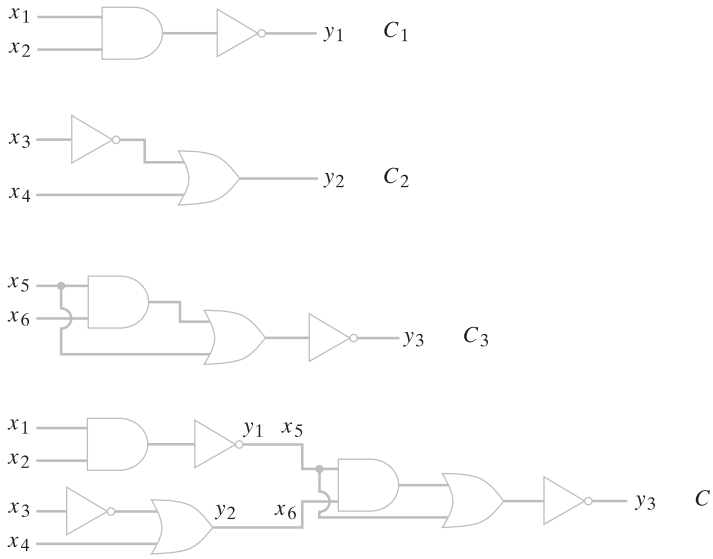


Figura 11.1.7 El circuito combinatorio C se obtiene interconectando los circuitos combinatorios C_1, C_2 y C_3. ◀

Ejemplo 11.1.8 ▶

Un circuito combinatorio con una salida, como el de la figura 11.1.4, se representa mediante una expresión que usa los símbolos $\wedge$, $\vee$ y $\neg$. Se sigue el flujo del circuito simbólicamente. Primero se aplica AND a x_1 y x_2 (figura 11.1.8), lo que produce la salida $x_1 \wedge x_2$. Esta salida después se une por OR con x_3 para producir la salida $(x_1 \wedge x_2) \vee x_3$. Después se aplica NOT a esta salida. Entonces la salida y puede ser

$$y = \overline{(x_1 \wedge x_2) \vee x_3}. \tag{11.1.1}$$

Las expresiones como (11.1.1) se llaman **expresiones booleanas**.

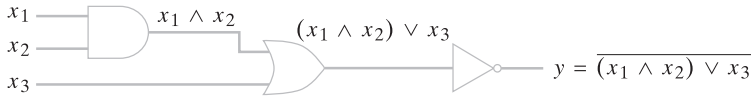


Figura 11.1.8 Representación de un circuito combinatorio mediante una expresión booleana. ◀

Definición 11.1.9 ▶

Las *expresiones booleanas* en los símbolos $x_1, \ldots, x_n$ se definen de manera recursiva como sigue.

$$0,\ 1,\ x_1,\ \ldots,\ x_n \tag{11.1.2}$$

son expresiones booleanas. Si X_1 y X_2 son expresiones booleanas, entonces

$$a)\ (X_1),\qquad b)\ \overline{X}_1,\qquad c)\ X_1 \vee X_2,\qquad d)\ X_1 \wedge X_2 \tag{11.1.3}$$

son expresiones booleanas.

Si X es una expresión booleana en los símbolos $x_1, \ldots, x_n$, algunas veces se escribe

$$X = X(x_1, \ldots, x_n).$$

Cualquiera de los símbolos x o $\overline{x}$ se llama *literal*. ◀

Ejemplo 11.1.10 ▶

Utilice la definición 11.1.9 para demostrar que el lado derecho de (11.1.1) es una expresión booleana en x_1, x_2 y x_3.

Por (11.1.2), x_1 y x_2 son expresiones booleanas. Por (11.1.3*d*), $x_1 \wedge x_2$ es una expresión booleana. Por (11.1.3*a*), $(x_1 \wedge x_2)$ es una expresión booleana. Por (11.1.2), x_3 es una expresión booleana. Como $(x_1 \wedge x_2)$ y x_3 son expresiones booleanas, por (11.1.3*c*), también lo es $(x_1 \wedge x_2) \vee x_3$. Por último, se aplica (11.1.3*b*) para concluir que

$$\overline{(x_1 \wedge x_2) \vee x_3}$$

es una expresión booleana. ◀

Si $X = X(x_1, \ldots, x_n)$ es una expresión booleana y se asignan valores $a_1, \ldots, a_n$ en $\{0, 1\}$ a $x_1, \ldots, x_n$, se pueden usar las definiciones 11.1.1 a la 11.1.3 para calcular el valor de X. Este valor se denota por $X(a_1, \ldots, a_n)$ o $X(x_i = a_i)$.

Ejemplo 11.1.11 ▶ Para $x_1 = 1$, $x_2 = 0$ y $x_3 = 0$, la expresión booleana $X(x_1, x_2, x_3) = \overline{(x_1 \wedge x_2) \vee x_3}$ de (11.1.1) se convierte en

$$\begin{aligned} X(1, 0, 0) &= \overline{(1 \wedge 0) \vee 0} \\ &= \overline{0 \vee 0} && \text{ya que } 1 \wedge 0 = 0 \\ &= \overline{0} && \text{ya que } 0 \vee 0 = 0 \\ &= 1 && \text{ya que } \overline{0} = 1. \end{aligned}$$

Se calculó de nuevo el cuarto renglón de la tabla en el ejemplo 11.1.5. ◀

En una expresión booleana en la que no se usan paréntesis para especificar el orden de las operaciones, se supone que $\wedge$ es evaluado antes de $\vee$.

Ejemplo 11.1.12 ▶ Para $x_1 = 0$, $x_2 = 0$ y $x_3 = 1$, el valor de la expresión booleana $x_1 \wedge x_2 \vee x_3$ es

$$x_1 \wedge x_2 \vee x_3 = 0 \wedge 0 \vee 1 = 0 \vee 1 = 1.$$ ◀

El ejemplo 11.1.8 mostró cómo se representa un circuito combinatorio con una salida como una expresión booleana. El siguiente ejemplo muestra cómo se construye un circuito combinatorio que representa una expresión booleana.

Ejemplo 11.1.13 ▶ Encuentre el circuito combinatorio correspondiente a la expresión booleana

$$(x_1 \wedge (\overline{x}_2 \vee x_3)) \vee x_2$$

y escriba la tabla lógica para el circuito obtenido.

Se comienza con la expresión $\overline{x}_2 \vee x_3$ en el paréntesis interior. Esta expresión se convierte en un circuito combinatorio como el mostrado en la figura 11.1.9.

Se aplica AND a la salida de este circuito y x_1 para producir el circuito dibujado en la figura 11.1.10. Por último, se aplica OR a la salida de este circuito y x_2 para dar el circuito deseado de la figura 11.1.11. La tabla lógica es la que sigue.

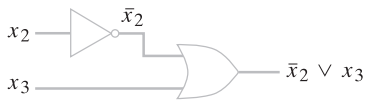


Figura 11.1.9 Circuito combinatorio correspondiente a la expresión booleana $\overline{x}_2 \vee x_3$.

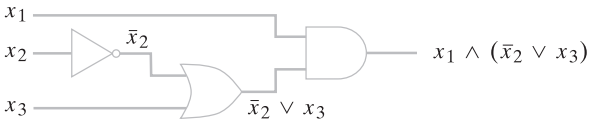


Figura 11.1.10 Circuito combinatorio correspondiente a la expresión booleana $x_3 \wedge (\overline{x}_2 \vee x_3)$

x_1	x_2	x_3	$(x_1 \wedge (\overline{x}_2 \vee x_3)) \vee x_2$
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	1
0	1	1	1
0	1	0	1
0	0	1	0
0	0	0	0

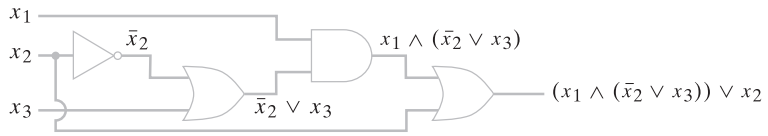


Figura 11.1.11 Circuito combinatorio correspondiente a la expresión booleana $(x_1 \wedge (\overline{x}_2 \vee x_3)) \vee x_2$.

◀

Sección de ejercicios de repaso

1. ¿Qué es un circuito combinatorio?

2. ¿Qué es un circuito secuencial?

3. ¿Qué es una compuerta AND?

4. ¿Qué es una compuerta OR?

5. ¿Qué es una compuerta NOT?

6. ¿Qué es un inversor?

7. ¿Qué es una tabla lógica de un circuito combinatorio?

8. ¿Qué es una expresión booleana?

9. ¿Qué es una literal?

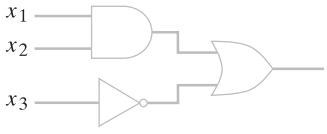
Ejercicios

En los ejercicios 1 al 6, escriba la expresión booleana que representa el circuito combinatorio, escriba la tabla lógica y escriba la salida de cada compuerta simbólicamente como en la figura 11.1.8.

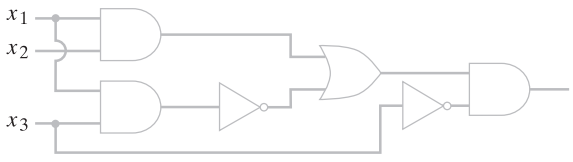
1.

2.

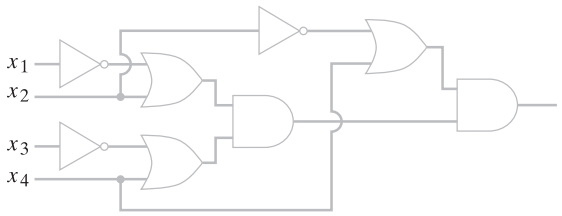
3.



4.



5.



6. El circuito inferior de la figura 11.1.7.

Los ejercicios 7 al 9 se refieren al circuito

7. Demuestre que este circuito no es un circuito combinatorio.

8. Demuestre que si $x = 0$, la salida y está determinada de manera única.

9. Demuestre que si $x = 1$, la salida y es indeterminada.

En los ejercicios 10 al 14, encuentre el valor de las expresiones booleanas para

$$x_1 = 1, \quad x_2 = 1, \quad x_3 = 0, \quad x_4 = 1.$$

10. $\overline{x_1 \wedge x_2}$

11. $x_1 \vee (\overline{x}_2 \wedge x_3)$

12. $(x_1 \wedge \overline{x}_2) \vee (x_1 \vee \overline{x}_3)$

13. $(x_1 \wedge (x_2 \vee (x_1 \wedge \overline{x}_2))) \vee ((x_1 \wedge \overline{x}_2) \vee (\overline{x_1 \wedge \overline{x}_3}))$

14. $(((x_1 \wedge x_2) \vee (x_3 \wedge \overline{x}_4)) \vee (\overline{(\overline{x_1 \vee x_3}) \wedge (\overline{x}_2 \vee x_3)})) \vee (x_1 \wedge \overline{x}_3)$

15. Usando la definición 11.1.9, demuestre que cada expresión en los ejercicios 10 al 14 es una expresión booleana.

En los ejercicios 16 al 20, determine si la expresión indicada es booleana. Si lo es, utilice la definición 11.1.9 para demostrarlo.

16. $x_1 \wedge (x_2 \vee x_3)$

17. $x_1 \wedge \overline{x}_2 \vee x_3$

18. (x_1)

19. $((x_1 \wedge x_2) \vee \overline{x}_3)$

20. $((x_1))$

21. Encuentre el circuito combinatorio correspondiente a cada expresión booleana en los ejercicios 10 al 14 y escriba la tabla lógica.

Un circuito de conmutación es una red eléctrica que consiste en interruptores cada uno de los cuales está abierto o cerrado. Un ejemplo aparece en la figura 11.1.12. Si el interruptor X está abierto (cerrado) se escribe $X = 0$ ($X = 1$). Los interruptores etiquetados con la misma letra, como B en la figura 11.1.12, están todos abiertos o todos cerrados. El interruptor X, como A en la figura 11.1.12, está abierto si y sólo si el interruptor $\overline{X}$, como $\overline{A}$, está cerrado. Si puede fluir corriente entre las terminales extremas izquierda y derecha del circuito, se dice que la salida del circuito es 1; de otra manera, se dice que la salida del circuito es 0. Una tabla de conmutación da la salida del circuito para todos los valores de los interruptores. La tabla de conmutación para la figura 11.1.12 es la siguiente:

A	B	C	*Salida del circuito*
1	1	1	1
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	1
0	1	0	1
0	0	1	1
0	0	0	1

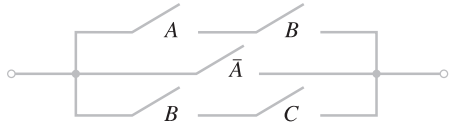


Figura 11.1.12 Circuito conmutador.

22. Dibuje un circuito con dos interruptores A y B que tienen la propiedad de que la salida del circuito es 1 precisamente cuando ambos, A y B, están cerrados. Esta configuración se etiqueta $A \wedge B$ y se llama **circuito en serie**.

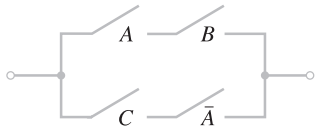
23. Dibuje un circuito con dos interruptores A y B que tienen la propiedad de que la salida del circuito es 1 justo cuando uno de ellos, A o B, está cerrado. Esta configuración se etiqueta $A \vee B$ y se llama **circuito en paralelo**.

24. Demuestre que el circuito de la figura 11.1.12 se puede representar simbólicamente como

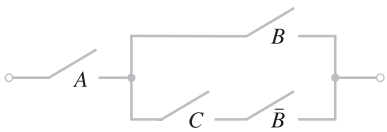
$$(A \wedge B) \vee \overline{A} \vee (B \wedge C).$$

Represente cada circuito en los ejercicios 25 al 29 simbólicamente y dé su tabla de conmutación.

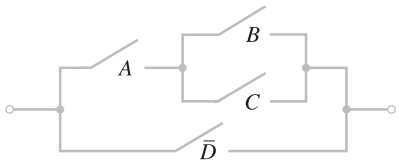
25.



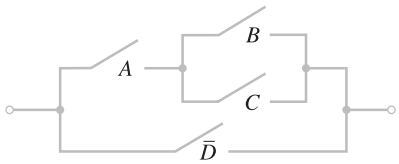
26.



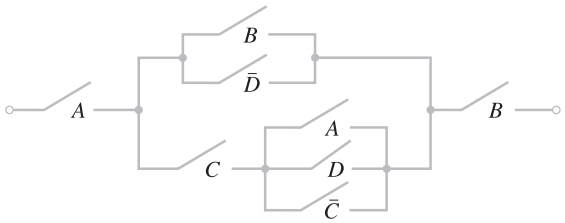
27.



28.



29.



Represente las expresiones en los ejercicios 30 al 34 como circuitos de conmutación y escriba las tablas de conmutación.

30. $(A \vee \overline{B}) \wedge A$

31. $A \vee (\overline{B} \wedge C)$

32. $(\overline{A} \wedge B) \vee (C \wedge A)$

33. $(A \wedge ((B \wedge \overline{C}) \vee (\overline{B} \wedge C))) \vee (\overline{A} \wedge B \wedge C)$

34. $A \wedge ((B \wedge C \wedge \overline{D}) \vee ((\overline{B} \wedge C) \vee D) \vee (\overline{B} \wedge \overline{C} \wedge D)) \wedge (B \vee \overline{D})$

11.2 ➜ Propiedades de los circuitos combinatorios

WWW

En la sección anterior se definieron dos operadores binarios $\wedge$ y $\vee$ en $Z_2 = \{0, 1\}$ y un operador unitario $^-$ en Z_2. (En el resto de este capítulo Z_2 denota el conjunto $\{0, 1\}$). Se vio que estos operadores podían implementarse en los circuitos como compuertas. En esta sección se analizan algunas propiedades del sistema que consiste en Z_2 y los operadores $\wedge$, $\vee$ y $^-$.

Teorema 11.2.1

Si $\wedge$, $\vee$ y $^-$ son los operadores de las definiciones 11.1.1 a la 11.1.3, entonces se cumplen las siguientes propiedades.

a) Leyes asociativas:

$$(a \vee b) \vee c = a \vee (b \vee c)$$
$$(a \wedge b) \wedge c = a \wedge (b \wedge c) \qquad \text{para todo } a, b, c \in Z_2.$$

b) Leyes conmutativas:

$$a \vee b = b \vee a, \qquad a \wedge b = b \wedge a \qquad \text{para todo } a, b \in Z_2.$$

c) Leyes distributivas:

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$$
$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c) \qquad \text{para todo } a, b, c \in Z_2.$$

d) Leyes de identidad:

$$a \vee 0 = a, \qquad a \wedge 1 = a \qquad \text{para todo } a \in Z_2.$$

e) Leyes de complementos:

$$a \vee \overline{a} = 1, \qquad a \wedge \overline{a} = 0 \qquad \text{para todo } a \in Z_2.$$

Demostración Las demostraciones son verificaciones directas. Se probará la primera ley distributiva y las otras ecuaciones se dejan como ejercicios (vea los ejercicios 16 y 17).

Debe demostrarse que

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c) \qquad \text{para todo } a, b, c \in z_2. \qquad \textbf{(11.2.1)}$$

Simplemente se evalúan ambos lados de (11.2.1) para todos los valores posibles de *a, b* y *c* en Z_2 y se verifica que en cada caso se obtenga el mismo resultado. La tabla proporciona los detalles.

a	b	c	$a \wedge (b \vee c)$	$(a \wedge b) \vee (a \wedge c)$
1	1	1	1	1
1	1	0	1	1
1	0	1	1	1
1	0	0	0	0
0	1	1	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	0	0

Ejemplo 11.2.2 ▶

Utilice el Teorema 11.2.1 para demostrar que los circuitos combinatorios de la figura 11.2.1 tienen salidas idénticas para entradas idénticas dadas.

Las expresiones booleanas que representan los circuitos son, respectivamente,

$$x_1 \vee (x_2 \wedge x_3), \qquad (x_1 \vee x_2) \wedge (x_1 \vee x_3).$$

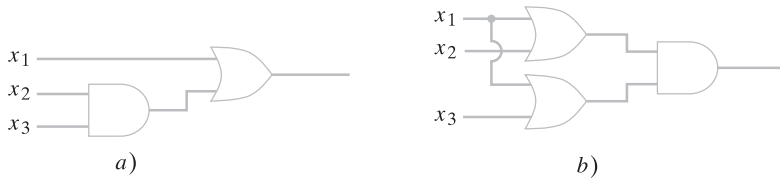


Figura 11.2.1 Los circuitos combinatorios *a)* y *b)* tienen salidas idénticas para entradas idénticas dadas y se dice que son equivalentes.

Por el Teorema 11.2.1*c)*,

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c) \qquad \textit{para todo } a, b, c \in Z_2 \qquad \textbf{(11.2.2)}$$

Pero (11.2.2) dice que los circuitos combinatorios de la figura 11.2.1 tienen salidas idénticas para valores idénticos de entrada. ◀

Las expresiones booleanas arbitrarias se definen iguales si tienen los mismos valores para todas las asignaciones posibles de bits a las literales.

Definición 11.2.3 ▶

Sean

$$X_1 = X_1(x_1, \ldots, x_n) \qquad \text{y} \qquad X_2 = X_2(x_1, \ldots, x_n)$$

expresiones booleanas. Se dice que X_1 es *igual* a X_2, y se escribe

$$X_1 = X_2$$

si

$$X_1(a_1, \ldots, a_n) = X_2(a_1, \ldots, a_n) \qquad \text{para todo } a_i \in Z_2.$$

◀

Ejemplo 11.2.4 ▶

Demuestre que

$$(\overline{x \vee y}) = \overline{x} \wedge \overline{y}. \qquad \textbf{(11.2.3)}$$

Según la definición 11.2.3, la ecuación (11.2.3) se cumple si es cierta para todas las opciones de x y y en Z_2. Entonces se puede simplemente elaborar una tabla que liste todas las posibilidades para verificar (11.2.3).

x	y	$(\overline{x} \vee y)$	$\overline{x} \wedge \overline{y}$
1	1	0	0
1	0	0	0
0	1	0	0
0	0	1	1

◀

Si se define una relación R en el conjunto de expresiones booleanas mediante la regla $X_1 \, R \, X_2$ si $X_1 = X_2$, R es una relación de equivalencia. Cada clase de equivalencia consiste en un conjunto de expresiones booleanas donde cada una es igual a cualquier otra.

Por las leyes asociativas, Teorema 11.2.1*a)*, se puede escribir sin ambigüedad

$$a_1 \vee a_2 \vee \cdots \vee a_n \qquad \textbf{(11.2.4)}$$

o bien,

$$a_1 \wedge a_2 \wedge \cdots \wedge a_n \qquad \textbf{(11.2.5)}$$

para $a_i \in Z_2$. El circuito combinatorio correspondiente a (11.2.4) se dibuja como en la figura 11.2.2 y el circuito combinatorio correspondiente a (11.2.5) se dibuja como en la figura (11.2.3).

Figura 11.2.2 Compuerta OR con n entradas.

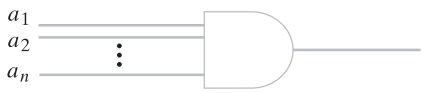


Figura 11.2.3 Compuerta AND con n entradas.

Las propiedades listadas en el Teorema 11.2.1 se cumplen en muchos sistemas. Un sistema que satisface estas propiedades se llama una **álgebra booleana**. Las álgebras booleanas abstractas se examinarán en la sección 11.3.

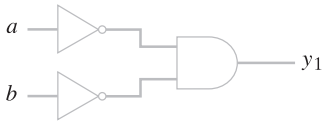
Una vez definida la igualdad de las expresiones booleanas, se define la equivalencia de los circuitos combinatorios.

Definición 11.2.5 ▶ Se dice que dos circuitos combinatorios, cada uno con entradas $x_1, \ldots, x_n$ y una sola salida, son *equivalentes* si, siempre que los circuitos reciban las mismas entradas, producen las mismas salidas. ◀

Ejemplo 11.2.6 ▶ Los circuitos combinatorios de las figuras 11.2.4 y 11.2.5 son equivalentes ya que, como se observa, tienen tablas lógicas idénticas.

a	b	y_1
1	1	0
1	0	0
0	1	0
0	0	1

Figura 11.2.4 Un circuito combinatorio y su tabla lógica.



a	b	y_1
1	1	0
1	0	0
0	1	0
0	0	1

Figura 11.2.5 Circuito combinatorio y su tabla lógica, que es idéntica a la tabla lógica de la figura 11.2.4. Se dice que los circuitos de las figuras 11.2.4 y 11.2.5 son equivalentes porque tienen tablas lógicas idénticas. ◀

Si se define una relación R en un conjunto de circuitos combinatorios por la regla C_1 R C_2 si C_1 y C_2 son equivalentes (en el sentido de la definición 11.2.5), R es una relación de equivalencia. Cada clase de equivalencia consiste en un conjunto de circuitos combinatorios mutuamente equivalentes.

El ejemplo 11.2.6 indica que es posible que los circuitos equivalentes no tengan el mismo número de compuertas. En general, es deseable emplear el menor número posible de compuertas para minimizar el costo de las componentes.

Se deduce de inmediato, a partir de las definiciones, que los circuitos combinatorios son equivalentes si y sólo si las expresiones booleanas que los representan generan tablas lógicas idénticas.

Teorema 11.2.7

Sea C_1 y C_2 circuitos combinatorios representados respectivamente por las expresiones booleanas $X_1 = X_1(x_1, \ldots, x_n)$ y $X_2 = X_2(x_1, \ldots, x_n)$. Entonces C_1 y C_2 son equivalentes si y sólo si $X_1 = X_2$.

Demostración El valor $X_1(a_1, \ldots, a_n)$ [respectivamente, $X_2(a_1, \ldots, a_n)$] para $a_i \in Z_2$ es la salida del circuito C_1 (respectivamente, C_2) para entradas $a_1, \ldots, a_n$.

De acuerdo con la definición 11.2.5, los circuitos C_1 y C_2 son equivalentes si y sólo si tienen las mismas salidas $X_1(a_1, \ldots, a_n)$ y $X_2(a_1, \ldots, a_n)$ para todas las entradas posibles $a_1, \ldots, a_n$. Entonces, los circuitos C_1 y C_2 son equivalentes si y sólo si

$$X_1(a_1, \ldots, a_n) = X_2(a_1, \ldots, a_n) \quad \text{para todo } a_1 \in Z_2. \qquad \textbf{(11.2.6)}$$

Pero por la definición 11.2.3, la ecuación (11.2.6) se cumple si y sólo si $X_1 = X_2$.

Ejemplo 11.2.8 ▶

En el ejemplo 11.2.4 se demostró que

$$(\overline{x \vee y}) = \overline{x} \wedge \overline{y}.$$

Por el Teorema 11.2.7, los circuitos combinatorios (figuras 11.2.4 y 11.2.5) correspondientes a estas expresiones son equivalentes. ◀

Sección de ejercicios de repaso

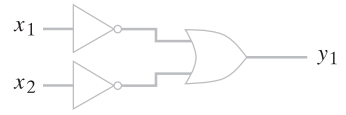
1. Establezca las leyes asociativas para $\wedge$ y $\vee$.
2. Establezca las leyes conmutativas para $\wedge$ y $\vee$.
3. Establezca las leyes distributivas para $\wedge$ y $\vee$.
4. Establezca las leyes de identidad para $\wedge$ y $\vee$.
5. Establezca las leyes de complementos para $\wedge$, $\vee$ y $^-$.
6. ¿Cuándo son iguales dos expresiones booleanas?
7. ¿Qué son expresiones combinatorias equivalentes?
8. ¿Cuál es la relación entre las expresiones combinatorias y las expresiones booleanas que las representan?

Ejercicios

Demuestre que los circuitos combinatorios de los ejercicios 1 al 5 son equivalentes.

1.

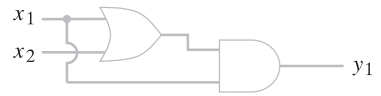
a)



b)

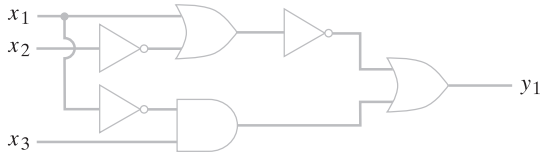
2.

a)

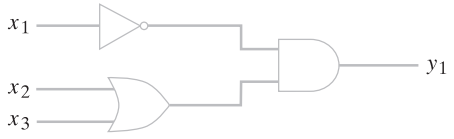


b)

3.

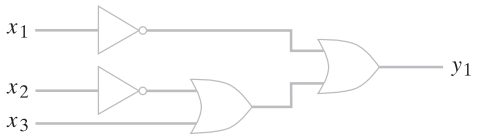


a)

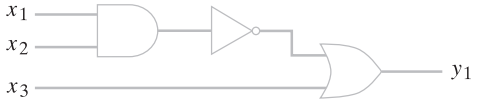


b)

4.

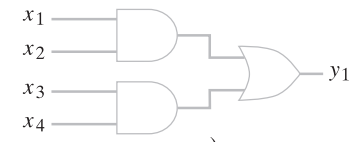


a)

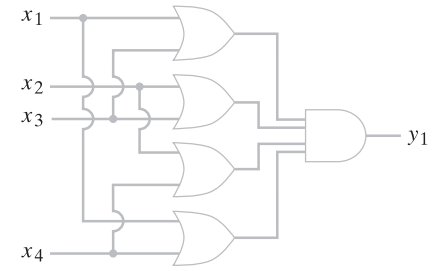


b)

5.



a)



b)

Verifique las ecuaciones en los ejercicios 6 al 10.

6. $x_1 \vee x_1 = x_1$

7. $x_1 \vee (x_1 \wedge x_2) = x_1$

8. $x_1 \wedge \overline{x}_2 = (\overline{\overline{x}_1 \vee x_2})$

9. $x_1 \wedge (\overline{x_2 \wedge x_3}) = (x_1 \wedge \overline{x}_2) \vee (x_1 \wedge \overline{x}_3)$

10. $(x_1 \vee x_2) \wedge (x_3 \vee x_4) = (x_3 \wedge x_1) \vee (x_3 \wedge x_2) \vee (x_4 \wedge x_1) \vee (x_4 \wedge x_2)$

Pruebe o desapruebe las ecuaciones en los ejercicios 11 al 15.

11. $\overline{\overline{x}} = x$

12. $\overline{x}_1 \wedge \overline{x}_2 = x_1 \vee x_2$

13. $\overline{x}_1 \wedge ((x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_3)) = x_2 \wedge x_3$

14. $(\overline{(\overline{x}_1 \wedge x_2) \vee (x_1 \wedge \overline{x}_3)}) = (x_1 \vee \overline{x}_2) \wedge (x_1 \vee \overline{x}_3)$

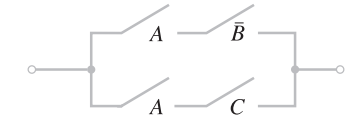
15. $(x_1 \vee x_2) \wedge (\overline{\overline{x}_3 \vee x_4}) \wedge (x_3 \wedge \overline{x}_2) = 0$

16. Pruebe la segunda afirmación del Teorema 11.2.1*c)*.

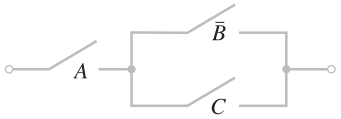
17. Pruebe los incisos *a)*, *b)* y *e)* del Teorema 11.2.1.

Se dice que dos circuitos de conmutación son equivalentes si las expresiones booleanas que los representan son iguales.

18. Demuestre que los circuitos de conmutación son equivalentes.



(a)

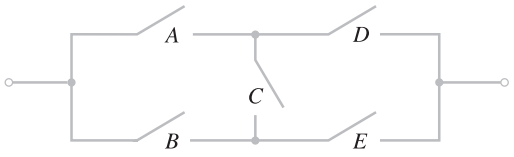


b)

19. Para cada circuito de conmutación en los ejercicios 25 al 29 de la sección 11.1, encuentre un circuito de conmutación equivalente usando circuitos en paralelo y en serie que tengan el menor número posible de interruptores.

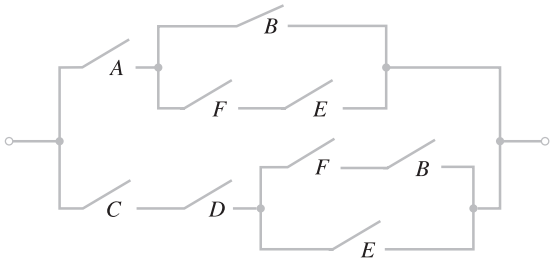
20. Para cada expresión booleana en los ejercicios 30 al 34 de la sección 11.1, encuentre un circuito de conmutación usando circuitos paralelos y en serie con el menor número posible de interruptores.

Un circuito puente es un circuito de conmutación, como el que se ilustra en seguida, que usa circuitos no paralelos y no en serie.

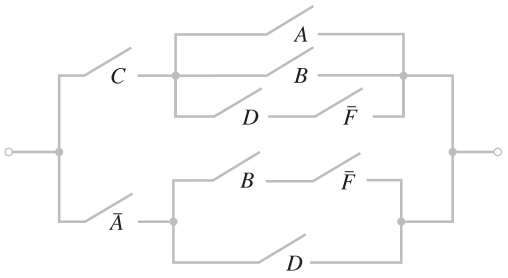


Para cada circuito de conmutación, encuentre un circuito de conmutación equivalente usando circuitos puente que tengan el menor número de interruptores.

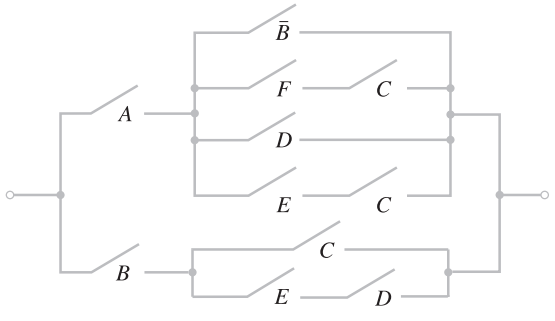
21.



22.



★ **23.**



24. Para cada expresión booleana en los ejercicios 30 al 34 de la sección 11.1, encuentre un circuito de conmutación usando circuitos puente con el menor número posible de interruptores.

11.3 ➜ Álgebras booleanas

WWW

En esta sección se consideran los sistemas generales que tienen propiedades como las indicadas en el Teorema 11.2.1. Se verá que, en apariencia, varios sistemas obedecen estas mismas leyes. Estos sistemas reciben el nombre de álgebras booleanas.

Definición 11.3.1 ▶

Un *álgebra booleana B* consiste en un conjunto S que contiene elementos distintos 0 y 1, operadores binarios $+$ y $\cdot$ en S, y un operador unitario $'$ en S que satisface las siguientes leyes.

a) Leyes asociativas:

$$(x + y) + z = x + (y + z)$$
$$(x \cdot y) \cdot z = x \cdot (y \cdot z) \qquad \text{para todo } x, y, z \in S.$$

b) Leyes conmutativas:

$$x + y = y + x, \qquad x \cdot y = y \cdot x \qquad \text{para todo } x, y \in S.$$

c) Leyes distributivas:

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$
$$x + (y \cdot z) = (x + y) \cdot (x + z) \qquad \text{para todo } x, y, z \in S.$$

d) Leyes de identidad:

$$x + 0 = x, \qquad x \cdot 1 = x \qquad \text{para todo } x \in S.$$

e) Leyes de complementos:

$$x + x' = 1, \qquad x \cdot x' = 0 \qquad \text{para todo } x \in S.$$

Si B es un álgebra booleana, se escribe $B = (S, +, \cdot, ', 0, 1)$. ◀

Las leyes asociativas se pueden omitir de la definición 11.3.1, puesto que se deducen de las otras leyes (vea el ejercicio 24).

Ejemplo 11.3.2 ▶

Por el Teorema 11.2.1, $(Z_2, \wedge, \vee, ^-, 0, 1)$ es un álgebra booleana. (Se está estableciendo que Z_2 denota el conjunto $\{0, 1\}$). Los operadores $+, \cdot, '$ en la definición 11.3.1 son $\wedge, \vee, ^-$, respectivamente. ◀

Como es costumbre, suele abreviarse $a \cdot b$ como ab. También se supone que $\cdot$ se evalúa antes que $+$. Esto permite eliminar algunos paréntesis. Por ejemplo, $(xy) + z$ se escribe de manera más sencilla como $xy + z$.

Es adecuado hacer algunos comentarios respecto a la definición 11.3.1. En primer lugar, 0 y 1 son sólo nombres simbólicos y, en general, no tienen relación con los números 0 y 1. Este mismo comentario se aplica a $+$ y $\cdot$, que sólo denotan operadores binarios y, en general, no tienen relación con la suma y multiplicación comunes.

Ejemplo 11.3.3 ▶

Sea U un conjunto universal y sea $S = \mathcal{P}(U)$ el conjunto potencia de U. Si se definen las siguientes operaciones

$$X + Y = X \cup Y, \qquad X \cdot Y = X \cap Y, \qquad X' = \overline{X}$$

en S, entonces $(S, \cup, \cap, ^-, \emptyset, U)$ es un álgebra booleana. El conjunto vacío $\emptyset$ asume el papel de 0 y el conjunto universal U hace el papel de 1. Si X, Y y Z son subconjuntos de S, las propiedades *a)* a la *e)* de la definición 11.3.1 se convierten en las siguientes propiedades de conjuntos (vea el Teorema 2.1.12):

a′) $(X \cup Y) \cup Z = X \cup (Y \cup Z)$
$(X \cap Y) \cap Z = X \cap (Y \cap Z)$ para todo $X, Y, Z \in \mathcal{P}(U)$.

b′) $X \cup Y = Y \cup X, \qquad X \cap Y = Y \cap X$ para todo $X, Y \in \mathcal{P}(U)$.

c′) $X \cap (Y \cup Z) = (X \cap Y) \cup (X \cap Z)$
$X \cup (Y \cap Z) = (X \cup Y) \cap (X \cup Z)$ para todo $X, Y, Z \in \mathcal{P}(U)$.

d′) $X \cup \emptyset = X, \quad X \cap U = X$ para todo $X \in \mathcal{P}(U)$.

e′) $X \cup \overline{X} = U, \quad X \cap \overline{X} = \emptyset$ para todo $X \in \mathcal{P}(U)$. ◀

En este punto se derivan algunas otras propiedades de las álgebras booleanas. Primero se demuestra que el elemento x' en la definición 11.3.1*e)* es único.

Teorema 11.3.4

En un álgebra booleana, el elemento x′de la definición 11.3.1e) es único. Específicamente, si $x + y = 1$ y $xy = 0$, entonces $y = x'$.

Demostración

$y = y1$	definición 11.3.1*d)*
$= y(x + x')$	definición 11.3.1*e)*
$= yx + yx'$	definición 11.3.1*c)*
$= xy + yx'$	definición 11.3.1*b)*
$= 0 + yx'$	proporcionado
$= xx' + yx'$	definición 11.3.1*e)*
$= x'x + x'y$	definición 11.3.1*b)*
$= x'(x \Downarrow y)$	definición 11.3.1*c)*
$= x'1$	proporcionado
$= x'$	definición 11.3.1*d)*

Definición 11.3.5 ▶ En un álgebra booleana, el elemento x' recibe el nombre de complemento de x. ◀

Ahora es posible derivar varias propiedades adicionales de las álgebras booleanas.

Teorema 11.3.6

Sea $B = (S, +, \cdot, ', 0, 1)$ un álgebra booleana. Las siguientes propiedades se cumplen.

a) Leyes de idempotencia:

$x + x = x, \quad xx = x$ para todo $x \in S$.

b) Leyes de acotación:

$x + 1 = 1, \quad x0 = 0$ para todo $x \in S$.

c) Leyes de absorción:

$x + xy = x, \quad x(x + y) = x$ para todo $x, y \in S$.

d) Leyes de involución:

$(x')' = x$ para todo $x \in S$.

e) Leyes de 0 y 1:

$0' = 1, \quad 1' = 0.$

f) Leyes de De Morgan para álgebras booleanas:

$(x + y)' = x'y', \quad (xy)' = x' + y'$ para todo $x, y \in S$.

Demostración Se probará *b)* y la primera afirmación de los incisos *a)*, *c)* y *f)* y se dejará el resto como ejercicio (vea los ejercicios 18 al 20).

a)

$x = x + 0$	definición 11.3.1*d)*
$= x + (xx')$	definición 11.3.1*e)*
$= (x + x)(x + x')$	definición 11.3.1*c)*
$= (x + x)1$	definición 11.3.1*e)*
$= x + x$	definición 11.3.1*d)*

b)

$x + 1 = (x + 1)1$	definición 11.3.1*d)*
$= (x + 1)(x + x')$	definición 11.3.1*e)*
$= x + 1x'$	definición 11.3.1*c)*
$= x + x'1$	definición 11.3.1*b)*
$= x + x'$	definición 11.3.1*d)*
$= 1$	definición 11.3.1*e)*
$x0 = x0 + 0$	definición 11.3.1*d)*
$= x0 + xx'$	definición 11.3.1*e)*
$= x(0 + x')$	definición 11.3.1*c)*
$= x(x' + 0)$	definición 11.3.1*b)*
$= xx'$	definición 11.3.1*d)*
$= 0$	definición 11.3.1*e)*

c)

$x + xy = x1 + xy$	definición 11.3.1*d)*
$= x(1 + y)$	definición 11.3.1*c)*
$= x(y + 1)$	definición 11.3.1*b)*
$= x1$	inciso *b)*
$= x$	definición 11.3.1*d)*

f) Si se demuestra que

$$(x + y)(x'y') = 0 \tag{11.3.1}$$

y

$$(x + y) + x'y' = 1, \tag{11.3.2}$$

se deduce del Teorema 11.3.4 que $x'y' = (x + y)'$. Ahora bien,

$(x + y)(x'y') = (x'y')(x + y)$	definición 11.3.1*b)*
$= (x'y')x + (x'y')y$	definición 11.3.1*c)*
$= x(x'y') + (x'y')y$	definición 11.3.1*b)*
$= (xx')y' + x'(y'y)$	definición 11.3.1*a)*
$= (xx')y' + x'(yy')$	definición 11.3.1*b)*
$= 0y' + x'0$	definición 11.3.1*e)*
$= y'0 + x'0$	definición 11.3.1*b)*
$= 0 + 0$	inciso *b)*
$= 0$	definición 11.3.1*d)*

Por lo tanto, (11.3.1) se cumple.

Ahora se verifica la ecuación (11.3.2).

$(x + y) + x'y' = ((x + y) + x')((x + y) + y')$	definición 11.3.1 *c)*
$= ((y \lrcorner\ x) + x')((x + y) + y')$	definición 11.3.1 *b)*
$= (y + (x + x'))(x + (y + y'))$	definición 11.3.1 *a)*
$= (y + 1)(x + 1)$	definición 11.3.1 *e)*
$= 1 \cdot 1$	inciso *b)*
$= 1$	definición 11.3.1 *d)*

Por el Teorema 11.3.4, $x'\,y' = (x + y)'$.

Ejemplo 11.3.7 ▶

Como se explicó en el ejemplo 11.3.3, si U es un conjunto, $\mathcal{P}(U)$ se puede considerar un álgebra booleana. Por lo tanto, las leyes de De Morgan, que para conjuntos se pueden enunciar

$$(\overline{X \cup Y}) = \overline{X} \cap \overline{Y}, \quad (\overline{X \cap Y}) = \overline{X} \cup \overline{Y} \qquad \text{para todo } X, Y \in \mathcal{P}(U),$$

se cumplen. Estas ecuaciones se verifican de manera directa (vea el Teorema 2.1.12), pero el Teorema 11.3.6 demuestra que son una consecuencia de otras leyes. ◀

Sin duda el lector ha observado que las ecuaciones que incluyen elementos de un álgebra booleana vienen en pares. Por ejemplo, las leyes de identidad [definición 11.3.1*d*)] son

$$x + 0 = x, \qquad x1 = x.$$

Se dice que estos pares son **duales**.

Definición 11.3.8 ▶

El *dual* de una afirmación que incluye expresiones booleanas se obtiene sustituyendo 0 por 1, 1 por 0, $+$ por $\cdot$ y $\cdot$ por $+$. ◀

Ejemplo 11.3.9 ▶

El dual de

$$(x + y)' = x'y'$$

es

$$(xy)' = x' + y'. \qquad \blacktriangleleft$$

Cada condición en la definición de álgebra booleana (definición 11.3.1) incluye su dual. Por lo tanto, se tiene el siguiente resultado.

Teorema 11.3.10

El dual de un teorema de álgebras booleanas también es un teorema.

Demostración Suponga que T es un teorema de álgebras booleanas. Entonces existe una prueba P de T que involucra sólo las definiciones de un álgebra booleana (definición 11.3.1). Sea P' la secuencia de afirmaciones obtenidas al sustituir cada enunciado en P por su dual. Entonces P' es una prueba del dual de T.

Ejemplo 11.3.11 ▶

El dual de

$$x + x = x \tag{11.3.3}$$

es

$$xx = x. \tag{11.3.4}$$

Antes se probó (11.3.3) [vea la prueba del Teorema 11.3.6*a*)]. Si se escribe el dual de cada afirmación en la demostración de (11.3.3), se obtiene la siguiente demostración de (11.3.4):

$$\begin{aligned} x &= x1 \\ &= x(x + x') \\ &= xx + xx' \\ &= xx + 0 \\ &= xx. \end{aligned}$$ ◀

Ejemplo 11.3.12 ▶

Las demostraciones dadas en el Teorema 11.3.6 de las dos afirmaciones del inciso *b)* son duales entre sí. ◀

Sección de ejercicios de repaso

1. Defina *álgebra booleana*.

2. ¿Qué son las leyes de idempotencia para álgebras booleanas?

3. ¿Qué son las leyes de acotación para álgebras booleanas?

4. ¿Qué son las leyes de absorción para álgebras booleanas?

5. ¿Qué son las leyes de involución para álgebras booleanas?

6. ¿Qué son las leyes 0 / 1 para álgebras booleanas?

7. ¿Qué son las leyes de De Morgan para álgebras booleanas?

8. ¿Cómo se obtiene el dual de una expresión booleana?

9. ¿Qué puede decirse del dual de un teorema sobre álgebras booleanas?

Ejercicios

1. Verifique las propiedades *a')* a *c')* del ejemplo 11.3.3.

2. Sea $S = \{1, 2, 3, 6\}$. Defina

$$x + y = \text{mcm}(x, y), \qquad x \cdot y = \text{mcd}(x, y), \qquad x' = \frac{6}{x}$$

para $x, y \in S$ (mcm denota el mínimo común múltiplo y mcd el máximo común divisor). Demuestre que $(S, +, \cdot, ', 1, 6)$ es un álgebra booleana.

3. $S = \{1, 2, 4, 8\}$. Defina $+$ y $\cdot$ como en el ejercicio 2 y defina $x' = 8/x$. Demuestre que $(S, +, \cdot, ', 1, 8)$ no es un álgebra booleana.

Sea $S_n = \{1, 2, \ldots, n\}$. *Defina*

$$x + y = \text{máx}\{x, y\}, \qquad x \cdot y = \text{mín}\{x, y\}.$$

4. Demuestre que los incisos *a)* a *c)* de la definición 11.3.1 se cumplen para S_n.

5. Demuestre que es posible definir 0, 1 y $'$ de manera que $(S_n, +, \cdot, ', 0, 1)$ es un álgebra booleana si y sólo si $n = 2$.

6. Rescriba las condiciones del Teorema 11.3.6 para conjuntos como los del ejemplo 11.3.3.

7. Interprete el Teorema 11.3.6 para conjuntos como los del ejemplo 11.3.3.

Escriba el dual de cada afirmación en los ejercicios 8 al 14.

8. $(x + y)(x + 1) = x + xy + y$

9. $(x' + y')' = xy$

10. Si $x + y = x + z$ y $x' + y = x' + z$, entonces $y = z$.

11. $xy' = 0$ si y sólo si $xy = x$.

12. Si $x + y = 0$, entonces $x = 0 = y$.

13. $x = 0$ si y sólo si $y = xy' + x'y$ para toda y.

14. $x + x(y + 1) = x$

15. Pruebe la afirmación del ejercicio 8 al 14.

16. Pruebe los duales de las afirmaciones de los ejercicios 8 al 14.

17. Escriba el dual del Teorema 11.3.4. ¿Cómo se relaciona el dual con el Teorema 11.3.4 en sí?

18. Pruebe las segundas afirmaciones de los incisos *a)*, *c)* y *f)* del Teorema 11.3.6.

19. Pruebe las segundas afirmaciones de los incisos *a)*, *c)* y *f)* del Teorema 11.3.6 obteniendo el dual de las demostraciones de las primeras afirmaciones.

20. Pruebe el Teorema 11.3.6, incisos *d)* y *e)*.

★**21.** Deduzca el inciso *a)* de la definición 11.3.1 a partir de los incisos *b)* a *e)* de la definición 11.3.1.

22. Sea U el conjunto de enteros positivos. Sea S la colección de subconjuntos X de U con uno de X o $\overline{X}$ finito. Demuestre que $(S, \cup, \cap, \bar{}, \emptyset, U)$ es un álgebra booleana.

★**23.** Sea n un entero positivo. Sea S el conjunto de todos los divisores de n, incluyendo 1 y n. Defina $+$ y $\cdot$ como en el ejercicio 2 y defina $x' = n/x$. ¿Qué condición debe satisfacer n para que $(S, +, \cdot, ', 1, n)$ sea un álgebra booleana?

★**24.** Demuestre que las leyes asociativas se deducen de las otras leyes de la definición 11.3.1.

Rincón de solución de problemas

Álgebras booleanas

Problema

Sea $(S, +, \cdot, ', 0, 1)$ un álgebra booleana y sea A un subconjunto de S. Demuestre que $(A, +, \cdot, ', 0, 1)$ es un álgebra booleana si y sólo si $1 \in A$ y $xy' \in A$ para todo $x, y \in A$.

Cómo atacar el problema

Como la afirmación dada es del tipo "si y sólo si", hay que probar dos afirmaciones:

Si $(A, +, \cdot, ', 0, 1)$ es un álgebra booleana, entonces $1 \in A$ y $xy' \in A$ para todo $x, y \in A$. **(1)**

Si $1 \in A$ y $xy' \in A$ para todo $x, y \in A$, entonces $(A, +, \cdot, ', 0, 1)$ es un álgebra booleana. **(2)**

Para probar la afirmación (1), resultan útiles las leyes especificadas en la definición de "álgebra booleana" (definición 11.3.1) y las leyes derivadas del Teorema 11.3.6 que deben obedecer los elementos de un álgebra booleana. Para probar que $(A, +, \cdot, ', 0, 1)$ es un álgebra booleana, se verificará que se satisfacen las leyes especificadas en la definición 11.3.1. Antes de continuar con la lectura es recomendable repasar la definición 11.3.1 y el Teorema 11.3.6.

Cómo encontrar una solución

Primero se intentará probar la afirmación (1). Se supone que $(A, +, \cdot, ', 0, 1)$ es un álgebra booleana y se quiere probar que

- $1 \in A$

y

- $xy' \in A$ para todo $x, y \in A$.

La definición 11.3.1 dice que un álgebra booleana contiene el 1. Como $(A, +, \cdot, ', 0, 1)$ es un álgebra booleana, $1 \in A$.

Ahora suponga que $x, y \in A$. La definición 11.3.1 dice que $'$ es un operador unitario en A. Esto significa que $y' \in A$. La definición 11.3.1 también dice que $\cdot$ es un operador binario en A. Esto quiere decir que $xy' \in A$. Esto completa la prueba de la afirmación (1).

Ahora se intentará probar el enunciado (2). Esta vez se supone que $1 \in A$ y xy' en A para todo $x, y \in A$ y se quiere probar que $(A, +, \cdot, ', 0, 1)$ es un álgebra booleana. Según la definición 11.3.1, se debe probar que

A contiene elementos distintos 0 y 1 **(3)**

$+$ y $\cdot$ son operadores binarios en A. **(4)**

$'$ es un operador unitario en A. **(5)**

Las leyes asociativas se cumplen. **(6)**

Las leyes conmutativas se cumplen. **(7)**

Las leyes distributivas se cumplen. **(8)**

Las leyes de identidad se cumplen. **(9)**

Las leyes de complementos se cumplen. **(10)**

A contiene el 1 por suposición. Para demostrar la afirmación (3), debe probarse que $0 \in A$. Se tienen sólo dos suposiciones acerca de A: $1 \in A$ y si $x, y \in A$, entonces $xy' \in A$. Todo lo que podemos hacer en este punto es combinar estas suposiciones; es decir, tomar $x = y = 1$ y examinar la conclusión: $11' \in A$. Ahora el Teorema 11.3.6*e*) [aplicado al álgebra booleana $(S, +, \cdot, ', 0, 1)$] dice que $1' = 0$. Sustituyendo $1'$ ahora se sabe que $10 \in A$. Pero el Teorema 11.3.6*b*) dice que para cualquier x, $x\,0 = 0$. Entonces $10 = 0$ está en A. ¡Se tuvo éxito! A contiene el 1 y el 0. 0 y 1 son distintos porque son elementos del álgebra booleana $(S, +, \cdot, ', 0, 1)$. Por lo tanto, la afirmación (3) queda demostrada.

Para demostrar la aseveración (4), debe probarse que $+$ y $\cdot$ son operadores binarios en A; es decir, si $x, y \in A$, entonces $x + y$ y xy están en A. Considere probar que $\cdot$ es un operador binario en A. Se sabe que si $x, y \in A$, entonces $xy' \in A$, que es muy cercano a lo que se desea probar. Si se pudiera de alguna manera sustituir y' por y en la expresión xy', se podría concluir que $xy \in A$. Lo que se desea hacer es suponer que $x, y \in A$, para deducir posteriormente

$$x, y' \in A, \qquad \textbf{(11)}$$

y luego concluir

$$xy = xy'' \in A.$$

Para deducir la expresión (11), es necesario demostrar que si $y \in A$, entonces $y' \in A$. Pero esto es la afirmación (5). ¡Desviación! Se trabajará en esta última.

Se supondrá que $y \in A$ y se intentará probar que $y' \in A$. Si se pudiera eliminar esa molesta x (en la hipótesis $x, y \in A$ implica $xy' \in A$), se tendría exactamente lo que se quiere. De hecho se puede eliminar x haciendo $x = 1$ puesto que $1y = y$. Formalmente, se argumenta lo siguiente. Sea y en A. Como $1 \in A$, $y' = 1y' \in A$. [$y' = 1y'$ por la definición 11.3.1*b*) y 11.3.1*d*)]. La afirmación (5) queda demostrada.

Ahora de regreso a la aseveración (4). Sean $x, y \in A$. Por la propiedad (5) que se acaba de probar, $y' \in A$. Por la condición dada, $xy = xy'' \in A$ [$y = y''$ por el teorema 11.3.6*d*)]. Esto demuestra que $\cdot$ es un operador binario en A.

Las leyes de De Morgan [Teorema 11.3.6*f*)], de hecho, permiten intercambiar $+$ y $\cdot$, para poder usarlos a fin de probar que si $x, y \in A$, entonces $x + y \in A$. Formalmente, se argumenta lo siguiente. Suponga que $x, y \in A$. Por la afirmación (5), se sabe que x' y y' están ambos en A. Como ya se probó que $\cdot$ es un operador binario en A, $x'y' \in A$. Por la aseveración (5), $(x'y')' \in A$. Por las leyes de De Morgan [Teorema 11.3.6*f*)] y el Teorema 11.3.6*d*), $x + y = x'' + y'' = (x'y')' \in A$. Por lo tanto, $+$ es un operador binario en A. Esto prueba la aseveración (4).

La siguiente afirmación que se debe probar es (6), que trata de verificar las leyes asociativas.

$$(x + y) + z = x + (y + z),$$
$$(xy)z = x(yz) \qquad \text{para todo } x, y, z \in A.$$

Ahora bien, $(S, +, \cdot, ', 0, 1)$ es un álgebra booleana y por ello las leyes asociativas se cumplen en S. Como A es un subconjunto de S, las leyes asociativas sin duda se cumplen en A. Entonces la afirmación (6) se cumple. Por la misma razón, las propiedades (7) a (10) también se cumplen en A. Por lo tanto, $(A, +, \cdot, ', 0, 1)$ es un álgebra booleana.

Solución formal

Suponga que $(A, +, \cdot, ', 0, 1)$ es un álgebra booleana. Entonces $1 \in A$. Suponga que $x, y \in A$. Entonces $y' \in A$. Por tanto, $xy' \in A$.

Ahora suponga que $1 \in A$ y $xy' \in A$ para todo $x, y \in A$. Haciendo $x = y = 1$, se obtiene $0 = 11' \in A$. Tomando $x = 1$, se obtiene $y' = 1y' \in A$. Entonces $'$ es un operador unitario en A. Sustituyendo y por y', se obtiene $xy = xy'' \in A$. Así, $\cdot$ es un operador binario en A. Ahora bien, $x + y = x'' + y'' = (x'y')' \in A$. Entonces $+$ es un operador binario en A. Los incisos *a)* a *e)* de la definición 11.3.1 se cumplen automáticamente en A, ya que se cumplen en S. Por lo tanto $(A, +, \cdot, ', 0, 1)$ es un álgebra booleana.

Resumen de las técnicas de solución de problemas

- Al intentar desarrollar una demostración, escriba con cuidado las suposiciones y qué se quiere probar.
- Al intentar desarrollar una demostración, examine definiciones y teoremas que tengan relación.
- Para probar que algo es un álgebra booleana, vaya directamente a la definición (definición 11.3.1).
- Considere demostrar las afirmaciones en un orden diferente al dado. En este problema fue más fácil probar la afirmación (5) antes que la (4).
- Intente diferentes sustituciones para las variables en una afirmación cuantificada universalmente. (Después de todo, "cuantificada universalmente" significa que la afirmación se cumple *para todos* los valores). Al hacer $x = y = 1$ en la afirmación

  $$xy' \in A \quad \text{para todo } x, y \in A,$$

  se pudo demostrar que $0 \in A$.

11.4 ➜ Funciones booleanas y simplificación de circuitos

Un circuito permite realizar una tarea específica. Si se quiere construir un circuito combinatorio, el problema puede darse en términos de entradas y salidas. Por ejemplo, suponga que se desea construir un circuito combinatorio para calcular el **OR-exclusivo** de x_1 y x_2. Se puede establecer el problema haciendo una lista de las entradas y salidas que define el OR-exclusivo. Esto equivale a elaborar la tabla lógica deseada.

Definición 11.4.1 ▶

El *OR-exclusivo* de x_1 y x_2 escrito $x_1 \oplus x_2$ se define en la tabla 11.4.1. ◀

Una tabla lógica, con una salida, es una función. El dominio es el conjunto de entradas y el recorrido o imagen es el conjunto de salidas. Para la función OR-exclusivo dada en la tabla 11.4.1, el dominio es el conjunto

$$\{(1, 1), (1, 0), (0, 1), (0, 0)\}$$

y el rango es el conjunto

$$Z_2 = \{0, 1\}.$$

TABLA 11.4.1 ■
OR-exclusivo.

x_1	x_2	$x_1 \oplus x_2$
1	1	0
1	0	1
0	1	1
0	0	0

Si se pudiera desarrollar una fórmula para la función OR-exclusivo de la forma

$$x_1 \oplus x_2 = X(x_1, x_2),$$

donde X es un expresión booleana, se podría resolver el problema de la construcción del circuito combinatorio. Se podría simplemente construir el circuito correspondiente a X.

Las funciones que se pueden representar por expresiones booleanas se llaman **funciones booleanas**.

Ejemplo 11.4.2 ▶

Sea $X(x_1, \ldots, x_n)$ un expresión booleana. Una función f de la forma

$$f(x_1, \ldots, x_n) = X(x_1, \ldots, x_n)$$

se llama *función booleana*. ◀

Ejemplo 11.4.3 ▶

La función f: $Z_2^3 \rightarrow Z_2$ definida por

$$f(x_1, x_2, x_3) = x_1 \wedge (\overline{x}_2 \vee x_3)$$

es una función booleana. Las entradas y salidas se dan en la siguiente tabla.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
1	1	1	1
1	1	0	0
1	0	1	1
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

◀

En el siguiente ejemplo se muestra cómo una función f: $Z_2^n \rightarrow Z_2$ puede interpretarse como una función booleana.

Ejemplo 11.4.4 ▶ Demuestre que la función f dada por la siguiente tabla es una función booleana.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	0
0	0	0	0

Considere el primer renglón de la tabla y la combinación

$$x_1 \wedge x_2 \wedge x_3. \tag{11.4.1}$$

Observe que si $x_1 = x_2 = x_3 = 1$, como se indica en el primer renglón de la tabla, entonces la expresión (11.4.1) es 1. Los valores de x_i dados por cualquier otro renglón de la tabla dan un valor de 0 a la expresión (11.4.1). De manera similar, para el cuarto renglón de la tabla se puede construir la combinación

$$x_1 \wedge \overline{x}_2 \wedge \overline{x}_3. \tag{11.4.2}$$

La expresión (11.4.2) tiene el valor 1 para los valores de x_i dados por el cuarto renglón de la tabla, mientras que los valores de x_i dados por cualquier otro renglón de la tabla dan el valor 0 para (11.4.2).

El procedimiento es claro. Se considera un renglón R de la tabla cuya salida es 1. Después se forma la combinación $x_1 \wedge x_2 \wedge x_3$ y se coloca una barra sobre cada x_i cuyo valor sea 0 en el renglón R. La combinación formada es 1 si y sólo si las x_i tienen los valores dados en el renglón R. Entonces, para el renglón 6, se obtiene la combinación

$$\overline{x}_1 \wedge x_2 \wedge \overline{x}_3. \tag{11.4.3}$$

Después, se aplica OR a los términos de (11.4.1) a (11.4.3) para obtener la expresión booleana

$$(x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \overline{x}_2 \wedge \overline{x}_3) \vee (\overline{x}_1 \wedge x_2 \wedge \overline{x}_3). \tag{11.4.4}$$

Se asegura que $f(x_1, x_2, x_3)$ y (11.4.4) son iguales. Para verificarlo, primero se supone que x_1, x_2 y x_3 tienen los valores dados por un renglón de la tabla para el que $f(x_1, x_2, x_3) = 1$. Entonces una de las expresiones (11.4.1) a la (11.4.3) es 1, de manera que el valor de (11.4.4) es 1. Por otro lado, si x_1, x_2, x_3 tienen los valores dados por un renglón de la tabla para el que $f(x_1, x_2, x_3) = 0$, todas las combinaciones (11.4.1) a (11.4.3) son 0, de manera que el valor de (11.4.4) es 0. Entonces f y la expresión booleana (11.4.4) están de acuerdo en Z_2^3; por lo tanto,

$$f(x_1, x_2, x_3) = (x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \overline{x}_2 \wedge \overline{x}_3) \vee (\overline{x}_1 \wedge x_2 \wedge \overline{x}_3),$$

como se aseguró. ◀

Después de una definición más, se mostrará que el método del ejemplo 11.4.4 se puede usar para representar cualquier $f: Z_2^n \to Z_2$.

Definición 11.4.5 ▶ Un *mintérmino* en los símbolos $x_1, \ldots, x_n$ es una expresión booleana de la forma

$$y_1 \wedge y_2 \wedge \cdots \wedge y_n,$$

donde cada y_i es ya sea x_i o $\overline{Xi}$. ◀

Teorema 11.4.6

Si f: $Z_2^n \to Z_2$, *entonces f es una función booleana. Si f no es idénticamente cero, sea* $A_1, \ldots, A_k$ *los elementos* A_j *de* Z_2^n *para los cuales* $f(A_i) = 1$. *Para cada* $A_i = (a_1, \ldots, a_n)$, sea

$$m_i = y_1 \wedge \cdots \wedge y_n,$$

donde

$$y_j = \begin{cases} x_j & si\ a_j = 1 \\ \overline{x}_j & si\ a_j = 0. \end{cases}$$

entonces

$$f(x_1, \ldots, x_n) = m_1 \vee m_2 \vee \cdots \vee m_k. \qquad \textbf{(11.4.5)}$$

Demostración Si $f(x_1, \ldots, x_n) = 0$ para todo x_i, entonces f es una función booleana, ya que 0 es una expresión booleana.

Suponga que f no es idénticamente cero. Sea $m_i(a_1, \ldots, a_n)$ el valor obtenido de m_i al sustituir cada x_j por a_j. Se deduce de la definición de m_i que

$$m_i(A) = \begin{cases} 1 & \text{si } A = A_i \\ 0 & \text{si } A \neq A_i. \end{cases}$$

Sea $A \in Z_2^n$. Si $A = A_i$ para alguna $i \in \{1, \ldots, k\}$, entonces $f(A) = 1$, $m_i(A) = 1$ y

$$m_1(A) \vee \cdots \vee m_k(A) = 1.$$

Por otro lado, si $A \neq A_i$ para cualquier $i \in \{1, \ldots, k\}$, entonces $f(A) = 0$, $m_i(A) = 0$ para $i = 1, \ldots, k$ y

$$m_1(A) \vee \cdots \vee m_k(A) = 0.$$

Por lo tanto, (11.4.5) se cumple.

Definición 11.4.7 ▶

La representación (11.4.5) de una función booleana f: $Z_2^n \to Z_2$ se llama *forma disyuntiva normal* de la función f. ◀

Ejemplo 11.4.8 ▶

Diseñe un circuito combinatorio que calcule el OR-exclusivo de x_1 y x_2.

La tabla lógica para la función OR-exclusivo $x_1 \oplus x_2$ se reproduce en la tabla 11.4.1. La forma disyuntiva normal de esta función es

$$x_1 \oplus x_2 = (x_1 \wedge \overline{x}_2) \vee (\overline{x}_1 \wedge x_2). \qquad \textbf{(11.4.6)}$$

El circuito combinatorio correspondiente a (11.4.6) se presenta en la figura 11.4.1.

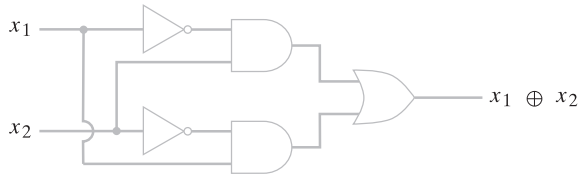


Figura 11.4.1 Circuito combinatorio para el OR-exclusivo.

◀

Suponga que una función está dada por una expresión booleana como

$$f(x_1, x_2, x_3) = (x_1 \vee x_2) \wedge x_3$$

y se desea encontrar la forma disyuntiva normal de f. Se podría escribir la tabla lógica de f y después usar el Teorema 11.4.6. De forma alternativa, se puede manejar directamente la expresión booleana usando las definiciones y resultados de las secciones 11.2 y 11.3. Se comenzará por distribuir los términos x_3 como sigue:

$$(x_1 \vee x_2) \wedge x_3 = (x_1 \wedge x_3) \vee (x_2 \wedge x_3).$$

Aunque esto representa la expresión booleana como una combinación de términos de la forma $y \wedge z$, no está en la forma disyuntiva normal, ya que cada término no contiene todos los símbolos x_1, x_2 y x_3. Sin embargo, esto tiene remedio de la siguiente forma:

$$\begin{aligned}(x_1 \wedge x_3) \vee (x_2 \wedge x_3) &= (x_1 \wedge x_3 \wedge 1) \vee (x_2 \wedge x_3 \wedge 1)\\ &= (x_1 \wedge x_3 \wedge (x_2 \vee \overline{x}_2)) \vee (x_2 \wedge x_3 \wedge (x_1 \vee \overline{x}_1))\\ &= (x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \overline{x}_2 \wedge x_3)\\ &\quad \vee (x_1 \wedge x_2 \wedge x_3) \vee (\overline{x}_1 \wedge x_2 \wedge x_3)\\ &= (x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \overline{x}_2 \wedge x_3)\\ &\quad \vee (\overline{x}_1 \wedge x_2 \wedge x_3).\end{aligned}$$

Esta expresión está en la forma disyuntiva normal de f.

El Teorema 11.4.6 tiene un dual. En este caso la función f se expresa como

$$f(x_1, \ldots, x_n) = M_1 \wedge M_2 \wedge \cdots \wedge M_k. \qquad \textbf{(11.4.7)}$$

Cada M_i es de la forma

WWW

$$y_1 \vee \cdots \vee y_n, \qquad \textbf{(11.4.8)}$$

donde y_j es ya sea x_j o $\overline{x}_j$. Un término de la forma (11.4.8) se llama **maxtérmino** y la representación de f (11.4.7) se llama **forma conjuntiva normal**. Los ejercicios 24 al 28 exploran con mayor detalle los maxtérmino y la forma conjuntiva normal .

Sección de ejercicios de repaso

1. Defina el *OR-exclusivo*.
2. ¿Qué es una función booleana?
3. ¿Qué es un mintérmino?
4. ¿Qué es la forma disyuntiva normal de una función booleana?
5. ¿Cómo se puede obtener la forma disyuntiva normal de una función booleana?
6. ¿Qué es un maxtérmino?
7. ¿Qué es la forma conjuntiva normal de una función booleana?

Ejercicios

En los ejercicios 1 al 10. encuentre la forma normal disyuntiva de cada función y dibuje el circuito combinatorio correspondiente a esa forma normal disyuntiva.

1.

x	y	$f(x, y)$
1	1	1
1	0	0
0	1	1
0	0	1

2.

x	y	$f(x, y)$
1	1	0
1	0	1
0	1	0
0	0	1

3.

x	y	z	$f(x, y, z)$
1	1	1	1
1	1	0	1
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	1
0	0	0	1

4.

x	y	z	$f(x, y, z)$
1	1	1	1
1	1	0	1
1	0	1	0
1	0	0	1
0	1	1	1
0	1	0	1
0	0	1	0
0	0	0	0

5.

x	y	z	$f(x, y, z)$
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	0
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	1

6.

x	y	z	$f(x, y, z)$
1	1	1	0
1	1	0	1
1	0	1	1
1	0	0	1
0	1	1	1
0	1	0	1
0	0	1	1
0	0	0	0

7.

x	y	z	$f(x, y, z)$
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	1

8.

x	y	z	$f(x, y, z)$
1	1	1	0
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	1
0	1	0	1
0	0	1	1
0	0	0	0

9.

w	x	y	z	$f(w, x, y, z)$
1	1	1	1	1
1	1	1	0	0
1	1	0	1	1
1	1	0	0	0
1	0	1	1	0
1	0	1	0	0
1	0	0	1	0
1	0	0	0	1
0	1	1	1	1
0	1	1	0	0
0	1	0	1	0
0	1	0	0	0
0	0	1	1	1
0	0	1	0	0
0	0	0	1	0
0	0	0	0	0

10.

w	x	y	z	$f(w, x, y, z)$
1	1	1	1	0
1	1	1	0	0
1	1	0	1	1
1	1	0	0	1
1	0	1	1	1
1	0	1	0	1
1	0	0	1	0
1	0	0	0	1
0	1	1	1	0
0	1	1	0	1
0	1	0	1	1
0	1	0	0	1
0	0	1	1	0
0	0	1	0	1
0	0	0	1	0
0	0	0	0	1

En los ejercicios 11 al 20, encuentre la forma disyuntiva normal de cada función usando las técnicas algebraicas. ($a \wedge b$ se abrevia ab.)

11. $f(x, y) = x \vee xy$

12. $f(x, y) = (x \vee y)(\overline{x} \vee \overline{y})$

13. $f(x, y, z) = x \vee y(x \vee \overline{z})$

14. $f(x, y, z) = (yz \vee x\overline{z})(\overline{x\overline{y} \vee z})$

15. $f(x, y, z) = (\overline{x}y \vee \overline{xz})(\overline{x \vee yz})$

16. $f(x, y, z) = x \vee (\overline{y} \vee (x\overline{y} \vee x\overline{z}))$

17. $f(x, y, z) = (x \vee \overline{x}y \vee \overline{x}y\overline{z})(xy \vee \overline{xz})(y \vee xy\overline{z})$

18. $f(x, y, z) = (\overline{x}y \vee \overline{xz})(\overline{xyz \vee y\overline{z}})(x\overline{y}z \vee x\overline{y} \vee x\overline{yz} \vee \overline{xyz})$

19. $f(w, x, y, z) = wy \vee (w\overline{y} \vee z)(x \vee \overline{w}z)$

20. $f(w, x, y, z) = (\overline{w}x\overline{y}z \vee x\overline{y}\,\overline{z})(\overline{\overline{w}\overline{y}z \vee xy\overline{z} \vee yxz})(\overline{w}z \vee xy \vee \overline{w}\,\overline{y}z \vee xy\overline{z} \vee \overline{x}yz)$

21. ¿Cuántas funciones booleanas de Z_2^n en Z_2 existen?

Sea F el conjunto de todas las funciones de Z_2^n en Z_2. Defina

$$\begin{aligned}
(f \vee g)(x) &= f(x) \vee g(x) && x \in Z_2^n \\
(f \wedge g)(x) &= f(x) \wedge g(x) && x \in Z_2^n \\
\overline{f}(x) &= \overline{f(x)} && x \in Z_2^n \\
0(x) &= 0 && x \in Z_2^n \\
1(x) &= 1 && x \in Z_2^n.
\end{aligned}$$

22. ¿Cuántos elementos tiene F?

23. Demuestre que $(F, \vee, \wedge, \bar{}, 0, 1)$ es un álgebra booleana.

24. Trabajando con el dual en el procedimiento del ejemplo 11.4.4, explique cómo se encuentra la forma conjuntiva normal de una función booleana de Z_2^n en Z_2.

25. Encuentre la forma conjuntiva normal de cada función en los ejercicios 1 al 10.

26. Usando métodos algebraicos, encuentre la forma conjuntiva normal de cada función en los ejercicios 11 al 20.

27. Demuestre que si $m_1 \vee \cdots \vee m_k$ es la forma disyuntiva normal de $f(x_1, \ldots, x_n)$, entonces $\overline{m}_1 \wedge \cdots \wedge \overline{m}_k$ es la forma conjuntiva normal de $\overline{f(x_1, \ldots, x_n)}$.

28. Con el método del ejercicio 27, encuentre la forma conjuntiva normal de $\bar{f}$ para cada función f de los ejercicios 1 al 10.

29. Demuestre que la forma disyuntiva normal (11.4.5) es única; es decir, demuestre que si se tiene una función booleana

$$f(x_1, \ldots, x_n) = m_1 \vee \cdots \vee m_k = m'_1 \vee \cdots \vee m'_j,$$

donde cada m_i, m'_i es un mintérmino, entonces $k = j$ y los subíndices en las m'_i se pueden permutar de manera que $m_i = m'_i$ para $i = 1, \ldots, k$.

11.5 ➜ Aplicaciones

WWW

En la sección anterior se mostró cómo diseñar un circuito combinatorio usando las compuertas AND, OR y NOT que calculan una función arbitraria de Z_2^n en Z_2, donde $Z_2 = \{0, 1\}$. En este sección se considera el uso de otros tipos de compuertas para implementar un circuito. También se estudia el problema de un diseño eficiente. Se concluye con la revisión de varios circuito útiles que tienen salidas múltiples. En toda la sección, $a \wedge b$ se escribe ab.

Antes de considerar alternativas para las compuertas AND, OR y NOT, debe darse una definición precisa de "compuerta".

Definición 11.5.1 ▶ Una *compuerta* es una función de Z_2^n en Z_2. ◀

Ejemplo 11.5.2 ▶ La compuerta AND es la función $\wedge$ de Z_2^2 en Z_2 definida como en la definición 11.1.1. La compuerta NOT es la función $\bar{}$ de Z_2 en Z_2 como en la definición 11.1.3. ◀

La atención se centra en las compuertas que permiten construir circuitos combinatorios arbitrarios.

Definición 11.5.3 ▶ Se dice que un conjunto de compuertas $\{g_1, \ldots, g_k\}$ es *funcionalmente completo* si, dado cualquier entero positivo n y una función f de Z_2^n en Z_2, es posible construir un circuito combinatorio que calcule f usando sólo las compuertas $g_1, \ldots, g_k$. ◀

Ejemplo 11.5.4 ▶ El Teorema 11.4.6 demuestra que un conjunto de compuertas {AND, OR, NOT} es funcionalmente completo. ◀

Un hecho interesante es que se pueda eliminar ya sea AND o bien OR del conjunto {AND, OR, NOT} y todavía obtener un conjunto de compuertas funcionalmente completo.

Teorema 11.5.5

Los conjuntos de compuertas

{AND, NOT} {OR, NOT}

son funcionalmente completos.

Demostración Se demostrará que el conjunto de compuertas {AND, NOT} es funcionalmente completo y se deja para los ejercicios el problema de demostrar que el otro conjunto es funcionalmente completo (vea el ejercicio 1).

Se tiene

$$\begin{aligned} x \vee y &= \bar{\bar{x}} \quad \bar{\bar{y}} && \text{ley de involución} \\ &= \overline{\bar{x}\,\bar{y}} && \text{ley de De Morgan.} \end{aligned}$$

Por lo tanto, una compuerta OR se puede sustituir por una compuerta AND y tres compuertas NOT. (El circuito combinatorio se ilustra en la figura 11.5.1).

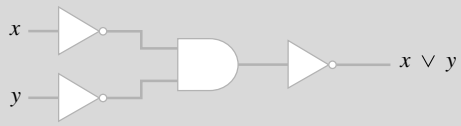


Figura 11.5.1 Circuito combinatorio que usa sólo las compuertas AND y NOT para calcular $x \vee y$.

Dada cualquier función f: $Z^n_2 \to Z_2$, por el Teorema 11.4.6 se puede construir un circuito combinatorio C usando las compuertas AND, OR y NOT, que calcula f. Pero la figura 11.5.1 muestra que cada compuerta OR se puede sustituir por compuertas AND y NOT. Por lo tanto, el circuito C se puede modificar de manera que consista sólo de compuertas AND y NOT. Entonces, el conjunto de compuertas {AND, NOT} es funcionalmente completo.

Aunque ninguno de AND, OR o NOT por sí solos forma un conjunto funcionalmente completo (vea los ejercicios 2 al 4), es posible definir una nueva compuerta que, por sí misma, forme un conjunto funcionalmente completo.

Definición 11.5.6 ▶

Una *compuerta NAND* recibe entradas x_1 y x_2, donde x_1 y x_2 son bits, y produce una salida denotada por $x_1 \uparrow x_2$, donde

$$x_1 \uparrow x_2 = \begin{cases} 0 & \text{si } x_1 = 1 \text{ y } x_2 = 1 \\ 1 & \text{de otra manera} \end{cases}$$

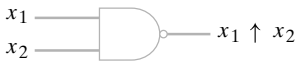


Figura 11.5.2 Compuerta NAND.

Una compuerta NAND se dibuja como se muestra en la figura 11.5.2. ◀

Muchos circuitos básicos usados hoy en las computadoras digitales se construyen a partir de compuertas NAND.

Teorema 11.5.7

El conjunto {NAND} es un conjunto de compuertas funcionalmente completo.

Demostración Primero se observa que

$$x \uparrow y = \overline{xy}.$$

Por lo tanto,

$$\overline{x} = \overline{xx} = x \uparrow x \tag{11.5.1}$$

$$x \vee y = \overline{\overline{x}\,\overline{y}} = \overline{x} \uparrow \overline{y} = (x \uparrow x) \uparrow (y \uparrow y). \tag{11.5.2}$$

Las ecuaciones (11.5.1) y (11.5.2) muestran que tanto OR como NOT se pueden escribir en términos de NAND. Por el Teorema 11.5.5, el conjunto {OR, NOT} es funcionalmente completo. Se concluye que el conjunto {NAND} también es funcionalmente completo.

Ejemplo 11.5.8 ▶

Diseñe circuitos combinatorios usando compuertas NAND para comparar las funciones $f_1(x) = \overline{x}$ y $f_2(x, y) = x \vee y$.

Los circuitos combinatorios, derivados de las ecuaciones (11.5.1) y (11.5.2), se ilustran en la figura 11.5.3.

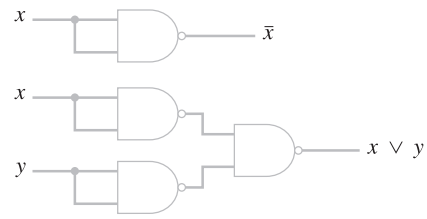


Figura 11.5.3 Circuitos combinatorios usando sólo compuertas NAND que calculan $\overline{x}$ y $x \vee y$.

◀

Considere el problema de diseñar un circuito combinatorio usando compuertas AND, OR y NOT para calcular la función f.

x	y	z	$f(x, y, z)$
1	1	1	1
1	1	0	1
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

La forma disyuntiva normal de f es

$$f(x, y, z) = xyz \vee xy\overline{z} \vee x\overline{y}\,\overline{z}. \tag{11.5.3}$$

El circuito combinatorio correspondiente a (11.5.3) se presenta en la figura 11.5.4.

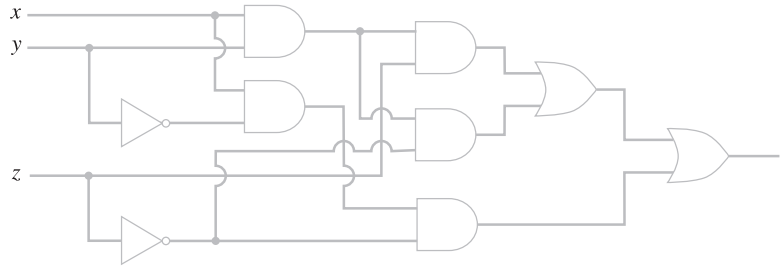


Figura 11.5.4 Circuito combinatorio que calcula $f(x, y, z) = xyz \vee xy\overline{z} \vee x\overline{y}\,\overline{z}$.

El circuito combinatorio de la figura 11.5.4 tiene nueve compuertas. Como se demostrará, es posible diseñar un circuito con menos compuertas. El problema de encontrar el mejor circuito se llama **problema de minimización**. Existen muchas definiciones de "el mejor".

Para encontrar un circuito más sencillo equivalente al de la figura 11.5.4, se intenta simplificar la expresión booleana (11.5.3) correspondiente. Las ecuaciones

$$Ea \vee E\overline{a} = E \tag{11.5.4}$$

$$E = E \vee Ea, \tag{11.5.5}$$

donde E representa una expresión booleana arbitraria, son útiles al simplificar expresiones booleanas.

La ecuación (11.5.4) se deriva como sigue:

$$Ea \vee E\overline{a} = E(a \vee \overline{a}) = E1 = E$$

usando las propiedades de álgebras booleanas. La ecuación (11.5.5) es en esencia la ley de absorción [Teorema 11.3.6*c*)].

Mediante las ecuaciones (11.5.4) y (11.5.5), se puede simplificar la ecuación (11.5.3) como sigue:

$$\begin{aligned} xyz \vee xy\overline{z} \vee x\overline{y}\,\overline{z} &= xy \vee x\overline{y}\,\overline{z} && \text{por (11.5.4)} \\ &= xy \vee xy\overline{z} \vee x\overline{y}\,\overline{z} && \text{por (11.5.5)} \\ &= xy \vee x\overline{z}. && \text{por (11.5.4).} \end{aligned}$$

Es posible una simplificación más,

$$xy \vee x\overline{z} = x(y \vee \overline{z}), \qquad \textbf{(11.5.6)}$$

aplicando la ley distributiva [definición 11.3.1*c*)]. La figura 11.5.5 muestra el circuito combinatorio correspondiente a (11.5.6), que requiere sólo tres compuertas.

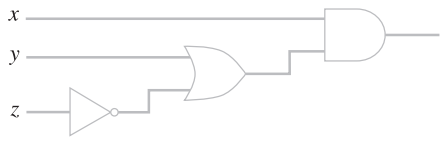


Figura 11.5.5 Circuito combinatorio con tres compuertas equivalente al de la figura 11.5.4.

Ejemplo 11.5.9 ▶

El circuito combinatorio en la figura 11.4.1 usa cinco compuertas AND, OR y NOT para calcular el OR-exclusivo, $x \oplus y$, de x y y. Diseñe un circuito que calcule $x \oplus y$ usando menos compuertas AND, OR y NOT.

Por desgracia, las expresiones (11.5.4) y (11.5.5) no ayudan a simplificar la forma disyuntiva normal $x\overline{y} \vee \overline{x}y$ de $x \oplus y$. Entonces debemos experimentar con varias reglas booleanas hasta producir una expresión que requiera menos de cinco compuertas. Una solución está dada por la expresión

$$(x \vee y)\overline{xy}$$

cuya implementación requiere sólo cuatro compuertas. Este circuito combinatorio se muestra en la figura 11.5.6.

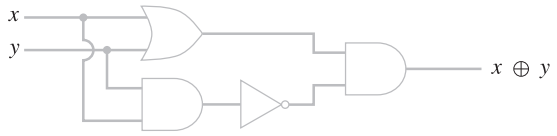


Figura 11.5.6 Circuito combinatorio de cuatro compuertas que calcula el OR-exclusivo $x \oplus y$ de x y y.

◀

El conjunto de compuertas disponible determina el problema de minimización. Como el estado de la tecnología determina las compuertas disponibles, el problema de minimización cambia con el tiempo. En los años 50, el problema típico consistía en minimizar circuitos considerando compuertas AND, OR y NOT. Se desarrollaron soluciones como el método de Quine-McCluskey y el método de mapas de Karnaugh. Se recomienda al lector consultar en [Mendelson] los detalles de estos métodos.

Los avances en la tecnología de estado sólido han hecho posible fabricar componentes muy pequeños, llamados **circuitos integrados**, que en sí son circuitos completos. Actualmente, diseñar un circuito consiste en combinar compuertas básicas como AND, OR, NOT y NAND y los circuitos integrados para calcular las funciones deseadas. El álgebra booleana sigue siendo una herramienta esencial, como lo mostraría una hojeada a cualquier libro de diseño lógico como [McCalla].

Se concluye esta sección con el análisis de varios circuitos combinatorios útiles que tienen salidas múltiples. Un circuito con n salidas se puede caracterizar por n expresiones booleanas, como se aprecia el ejemplo siguiente.

Ejemplo 11.5.10 ▶

Escriba dos expresiones booleanas para describir el circuito combinatorio de la figura 11.5.7.

La salida y_1 se describe por la expresión

$$y_1 = \overline{ab},$$

y y_2 se describe por la expresión

$$y_2 = bc \vee \overline{ab}.$$

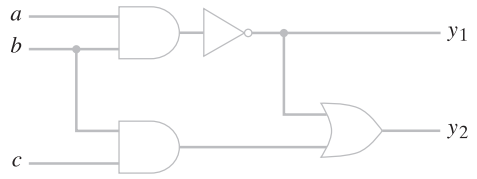


Figura 11.5.7 Circuito combinatorio con dos salidas.

◀

El primer circuito se llama **medio sumador (half adder)** o **semisumador**.

Definición 11.5.11 ▶

Un *medio sumador* acepta como entrada dos bits x y y para producir como salida la suma binaria cs de x y y. El término cs es un número binario de dos bits; s es el bit de la suma y c es el bit de acarreo. ◀

Ejemplo 11.5.12 ▶

Circuito medio sumador

Diseñe un circuito combinatorio sumador parcial.

La tabla del medio sumador es la siguiente:

x	y	c	s
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0

Esta función tiene dos salidas, c y s. Se observa que $c = xy$ y $s = x \oplus y$. Entonces se obtiene el circuito medio sumador de la figura 11.5.8. Se usó el circuito de la figura 11.5.6 para considerar el OR-exclusivo.

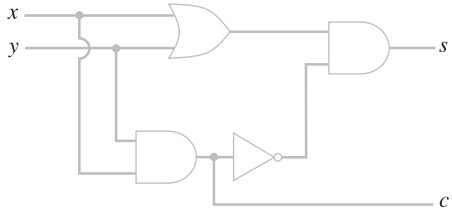


Figura 11.5.8 Circuito medio sumador.

◀

Un **sumador completo** suma tres bits y se emplea para sumar dos bits y un tercer bit de acarreo de una suma anterior.

Definición 11.5.13 ▶

Un *sumador completo* acepta como entrada tres bits x, y y z y produce como salida la suma binaria cs de x, y y z. El término cs es un número binario de dos bits. ◀

Ejemplo 11.5.14 ▶ **Circuito sumador completo**

Diseñe un circuito combinatorio sumador completo.

La tabla para el circuito sumador completo es la siguiente:

x	y	z	c	s
1	1	1	1	1
1	1	0	1	0
1	0	1	1	0
1	0	0	0	1
0	1	1	1	0
0	1	0	0	1
0	0	1	0	1
0	0	0	0	0

Al verificar las ocho posibilidades, se observa que

$$s = x \oplus y \oplus z;$$

así, se pueden utilizar dos circuitos OR-exclusivo para calcular s.

Para calcular c, primero se encuentra la forma disyuntiva normal

$$c = xyz \vee xy\overline{z} \vee x\overline{y}z \vee \overline{x}yz \tag{11.5.7}$$

de c. Después se utilizan las ecuaciones (11.5.4) y (11.5.5) para simplificar la (11.5.7) como sigue:

$$\begin{aligned} xyz \vee xy\overline{z} \vee x\overline{y}z \vee \overline{x}yz &= xy \vee x\overline{y}z \vee \overline{x}yz \\ &= xy \vee xyz \vee x\overline{y}z \vee \overline{x}yz \\ &= xy \vee xz \vee \overline{x}yz \\ &= xy \vee xz \vee xyz \vee \overline{x}yz \\ &= xy \vee xz \vee yz. \end{aligned}$$

Es posible eliminar las compuertas adicionales si se escribe

$$c = xy \vee z(x \vee y).$$

Se obtiene el circuito sumador completo de la figura 11.5.9.

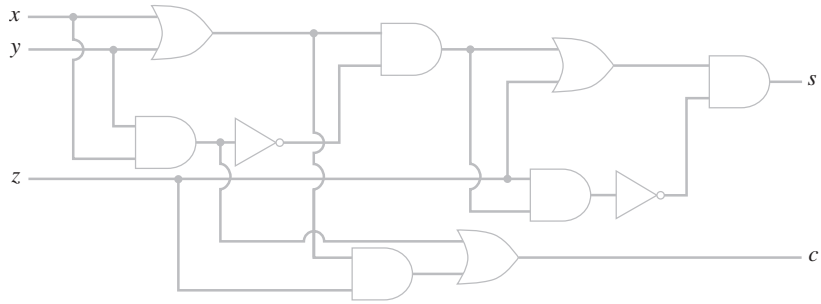


Figura 11.5.9 Circuito sumador completo.

◀

El último ejemplo muestra cómo se emplean los circuitos medio sumador y sumador completo para construir un circuito que sume números binarios.

Ejemplo 11.5.15 ▶

Circuito para sumar números binarios

Empleando los circuitos sumador parcial y sumador completo, diseñe un circuito combinatorio que calcule la suma de dos números de tres bits.

Sea $M = x_3x_2x_1$ y $N = y_3y_2y_1$ los números que deben sumarse y sea $z_4z_3z_2z_1$ la suma. El circuito que calcula la suma de M y N se ilustra en la figura 11.5.10. Se trata de una implementación del algoritmo estándar para sumar números ya que, de hecho, el "bit de acarreo" se *acarrea* a la siguiente suma binaria.

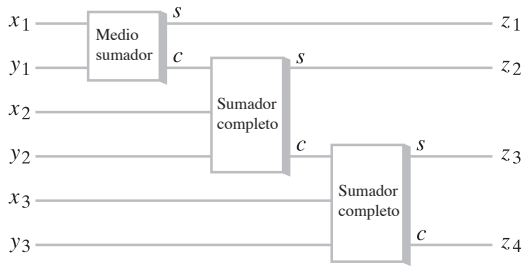


Figura 11.5.10 Un circuito combinatorio que calcula la suma de dos números de tres bits.

◀

Si se estuvieran utilizando registros de tres bits para la suma, de manera que la suma de dos números de tres bits fuera cuando mucho de tres bits, se podría usar el bit z_4 en el ejemplo 11.5.15 como una bandera de saturación. Si $z_4 = 1$, ocurrió un desbordamiento; si $z_4 = 0$, no hubo saturación.

En el siguiente capítulo (ejemplo 12.1.3), se estudia un circuito secuencial que utiliza un memoria interna primitiva para sumar números binarios.

Sección de ejercicios de repaso

1. ¿Qué es una compuerta?
2. ¿Qué es un conjunto de compuertas funcionalmente completo?
3. Dé ejemplos de conjuntos de compuertas funcionalmente completos.
4. ¿Qué es una compuerta NAND?
5. ¿El conjunto {NAND} es funcionalmente completo?
6. ¿Cuál es el problema de minimización?
7. ¿Qué es un circuito integrado?
8. Describa un circuito medio sumador.
9. Describa un circuito sumador completo.

Ejercicios

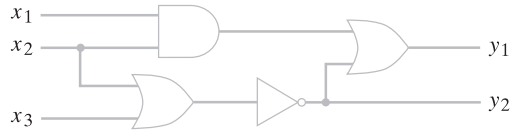
1. Demuestre que el conjunto de compuertas {OR, NOT} es funcionalmente completo.

Demuestre que cada conjunto de compuertas en los ejercicios 2 al 5 no es funcionalmente completo.

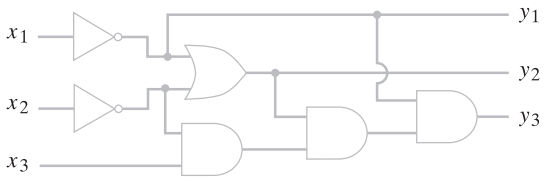
2. {AND}
3. {OR}
4. {NOT}
5. {AND, OR}
6. Dibuje un circuito con sólo compuertas NAND que calcule xy.
7. Escriba xy usando sólo $\uparrow$.
8. Pruebe o desapruebe: $x \uparrow (y \uparrow z) = (x \uparrow y) \uparrow z$, para todo $x, y, z \in Z_2$.

Escriba expresiones booleanas para describir los circuitos de salidas múltiples en los ejercicios 9 al 11.

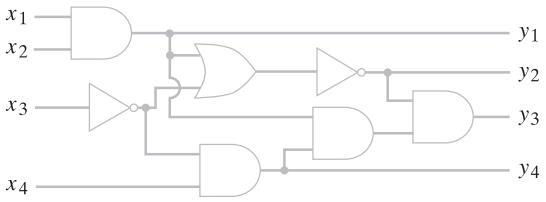
9.



10.



11.



12. Diseñe circuitos usando sólo compuertas NAND para calcular las funciones de los ejercicios 1 al 10, sección 11.4.

13. ¿Puede reducir el número de compuertas NAND incluidas en cualquiera de los circuitos del ejercicio 12?

14. Diseñe circuitos que usen el menor número de compuertas AND, OR y NOT tanto como sea posible para calcular las funciones de los ejercicios 1 al 10, sección 11.4.

15. Diseñe un circuito medio sumador usando sólo compuertas NAND.

★ **16.** Diseñe un circuito medio sumador usando cinco compuertas NAND.

Una compuerta NOR recibe entradas x_1 *y* x_2*, donde* x_1 *y* x_2 *son bits, y produce una salida denotada por* $x_1 \downarrow x_2$*, donde*

$$x_1 \downarrow x_2 = \begin{cases} 0 & si\ x_1 = 1\ o\ x_2 = 1 \\ 1 & de\ otra\ manera. \end{cases}$$

17. Escriba xy, $x \vee y$, $\overline{x}$ y $x \uparrow y$ en términos de $\downarrow$.

18. Escriba $x \downarrow y$ en términos de $\uparrow$.

19. Escriba la tabla lógica para la función nor.

20. Demuestre que el conjunto de compuertas {NOR} es funcionalmente completo.

21. Diseñe circuitos usando sólo compuertas NOR para calcular las funciones de los ejercicios 1 al 10, sección 11.4.

22. ¿Puede reducir el número de compuertas NOR empleadas en cualquiera de sus circuitos para el ejercicio 21?

23. Diseñe un circuito medio sumador con sólo compuertas NOR.

★ **24.** Diseñe un circuito medio sumador con cinco compuertas NOR.

25. Diseñe un circuito con tres entradas que produce 1 justo cuando dos o tres entradas tienen valor 1.

26. Diseñe un circuito que multiplique los números binarios x_2x_1 y y_2 y_1. La salida será de la forma $z_4z_3z_2z_1$.

27. Un **módulo de 2** es un circuito que acepta como entrada dos bits b y FLAGIN y produce bits c y FLAGOUT. Si FLAGIN $= 1$, entonces $c = \overline{b}$ y FLAGOUT $= 1$. Si FLAGIN $= 0$ y $b = 1$, entonces FLAGOUT $= 1$. Si FLAGIN $= 0$ y b $= 0$, entonces FLAGOUT $= 0$. Si FLAGIN $= 0$, entonces $c = b$. Diseñe un circuito para implementar el módulo de 2.

El complemento de 2 de un número binario se calcula utilizando el siguiente algoritmo.

Algoritmo 11.5.16

Cómo encontrar el complemento de 2

Este algoritmo calcula el complemento de 2 $C_NC_{N-1} \cdots C2C1$ del número binario $M = B_NB_{N-1} \cdots B_2B_1$. El número M se barre de derecha a izquierda y los bits se copian hasta que se encuentra 1. En adelante, si $B_i = 0$, se hace $C_i = 1$ y si $B_i = 0$, se hace $C_i = 0$. La bandera F indica si se encontró un 1 ($F =$ **verdadero**) o no ($F =$ **falso**).

Entrada: $B_NB_{N-1} \cdots B_1$
Salida: $C_NC_{N-1} \cdots C_1$

```
complemento_dos(B) {
    F = falso
    i = 1
    while (¬F ∧ i ≤ n) {
        C_i = B_i
        if (B_i == 1)
            F = verdadero
        i = i + 1
    }
    while (i ≤ n) {
        C_i = B_i ⊕ 1
        i = i + 1
    }
    return C
}
```

Encuentre el complemento de 2 de los números en los ejercicios 28 al 30 usando el algoritmo 11.15.16.

28. 101100

29. 11011

30. 011010110

31. Utilizando módulos de 2, diseñe un circuito que calcule el complemento de 2 $y_3y_2y_1$ del número binario de tres bits $x_3x_2x_1$.

★ ***32.*** Sea $*$ un operador binario en un conjunto S que contiene 0 y 1. Escriba un conjunto de axiomas para $*$, modelado tomando en cuenta las reglas que satisface NAND, de manera que si se define

$$\overline{x} = x * x$$
$$x \vee y = (x * x) * (y * y)$$
$$x \wedge y = (x * y) * (x * y),$$

entonces $(S, \vee, \wedge, \overline{\ }, 0, 1)$ es un álgebra booleana.

★ **33.** Sea $*$ un operador binario en S que contiene 0 y 1. Escriba un conjunto de axiomas para $*$, modelado tomando en cuenta las reglas que satisface NOR, y las definiciones de $\overline{\ }$, $\vee$ y $\wedge$ de manera que $(S, \vee, \wedge, \overline{\ }, 0, 1)$ sea un álgebra booleana.

★ ***34.*** Demuestre que $\{\rightarrow\}$ es funcionalmente completo (vea la definición 1.2.3).

★ **35.** Sea $B(x, y)$ una expresión booleana en las variables x y y que sólo usa el operador $\leftrightarrow$ (vea la definición 1.2.8).

a) Demuestre que si B contiene un número par de x, los valores de $B(\overline{x}, y)$ y $B(x, y)$ son iguales para toda x y y.

b) Demuestre que si B contiene un número impar de x, los valores de $B(\overline{x}, y)$ y $\overline{B(x.\ y)}$ son iguales para toda x y y.

c) Utilice los incisos *a)* y *b)* para demostrar que $\{\leftrightarrow\}$ *no* es funcionalmente completo.

Paul Pluznikov contribuyó con este ejercicio.

Notas

Algunas referencias generales de álgebras booleanas son [Hohn; y Mendelson]. [Mendelson] contiene más de 150 referencias de álgebras booleanas y circuitos combinatorios. Los libros de diseño lógico incluyen [Kohavi; McCalla; y Ward].

[Hailperin] presenta un análisis técnico de las matemáticas de Boole. También proporciona referencias adicionales. El libro de Boole, *The Laws of Thought* (*Las leyes del pensamiento*), se ha reeditado (vea [Boole]).

A causa de nuestro interés en las aplicaciones del álgebra booleana, la mayor parte del análisis se limitó al álgebra booleana $(Z_2, \vee, \wedge, \bar{}, 0, 1)$.. Sin embargo, las versiones de la mayoría de nuestros resultados siguen siendo válidas para álgebras booleanas finitas arbitrarias.

Las **expresiones booleanas** en símbolos $x_1, \ldots, x_n$ sobre un álgebra booleana arbitraria $(S, +, \cdot, ', 0, 1)$ se definen de manera recursiva como

- Para cada $s \in S$, s es una expresión booleana.
- $x_1, \ldots, x_n$ son expresiones booleanas.

Si X_1 y X_2 son expresiones booleanas, también lo son

$$(X_1), \quad X_1', \quad X_1 + X_2, \quad X_1 \cdot X_2.$$

Una **función booleana** sobre S se define como una función de S^n a S de la forma

$$f(x_1, \ldots, x_n) = X(x_1, \ldots, x_n),$$

donde X es un expresión booleana en los símbolos $x_1, \ldots, x_n$ sobre S. Se puede definir una forma disyuntiva normal para f. Otro resultado es que si X y Y son expresiones booleanas sobre S y

$$X(x_1, \ldots, x_n) = Y(x_1, \ldots, x_n)$$

para todo $x_i \in S$, entonces Y se puede derivar de X usando la definición de un álgebra booleana (definición 11.3.1). Otros resultados son que cualquier álgebra booleana finita tiene 2^n elementos y que si dos álgebras booleanas tienen 2^n elementos, en esencia, son la misma. Se concluye que cualquier álgebra booleana finita es esencialmente el ejemplo 11.3.3, el álgebra booleana de los subconjuntos de un conjunto universal finito U. Las demostraciones de estos resultados se encuentran en [Mendelson].

Repaso del capítulo

Sección 11.1

1. Circuito combinatorio
2. Circuito secuencial
3. Compuerta AND
4. Compuerta OR
5. Compuerta NOT (inversor)
6. Tabla lógica de un circuito combinatorio
7. Expresión booleana
8. Literal

Sección 11.2

9. Propiedades de $\wedge$, $\vee$ y $\bar{}$: leyes asociativas, conmutativas, distributivas, de identidad, de complemento (vea el Teorema 11.2.1)
10. Expresiones booleanas iguales
11. Expresiones booleanas equivalentes
12. Las expresiones combinatorias son equivalentes si y sólo si las expresiones booleanas que las representan generan tablas lógicas idénticas.

Sección 11.3

13. Álgebra booleana
14. x': complemento de x
15. Propiedades de álgebras booleanas: leyes de idempotencia, de acotación, de absorción, de involución, 0 y 1, y leyes De Morgan

16. Dual de afirmación que incluye expresiones booleanas
17. El dual de un teorema de álgebras booleanas también es un teorema.

Sección 11.4

18. OR-exclusivo
19. Función booleana
20. Mintérmino: $y_1 \wedge y_2 \wedge \cdots \wedge y_n$, , donde cada y_i es x_i o $\overline{x}_i$
21. Forma disyuntiva normal
22. Cómo escribir una función booleana en la forma disyuntiva normal (Teorema 11.4.6)
23. Maxtérmino: $y_1 \vee y_2 \vee \cdots \vee y_n$, , donde cada y_i es x_i o $\overline{x}_i$
24. Forma conjuntiva normal

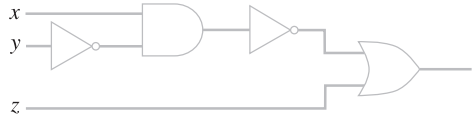
Sección 11.5

25. Compuerta
26. Conjunto de compuertas funcionalmente completo
27. Los conjuntos de compuertas {AND, not} y {OR, NOT} son funcionalmente completos.
28. Compuerta NAND
29. El conjunto {NAND} es un conjunto de compuerta funcionalmente completo.
30. Problema de minimización
31. Circuito integrado
32. Circuito medio sumador
33. Circuito sumador completo

Autoevaluación del capítulo

Sección 11.1

1. Escriba una expresión booleana que represente el circuito combinatorio y escriba la tabla lógica.



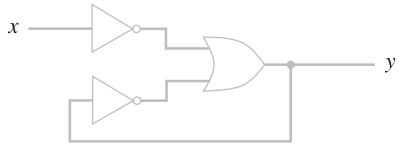
2. Encuentre el valor de la expresión booleana

$$(x_1 \wedge x_2) \vee (\overline{x}_2 \wedge x_3)$$

si $x_1 = x_2 = 0$ y $x_3 = 1$.

3. Encuentre un circuito combinatorio correspondiente a las expresiones booleanas del ejercicio 2.

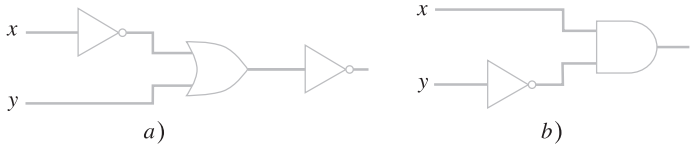
4. Demuestre que el siguiente circuito no es combinatorio.



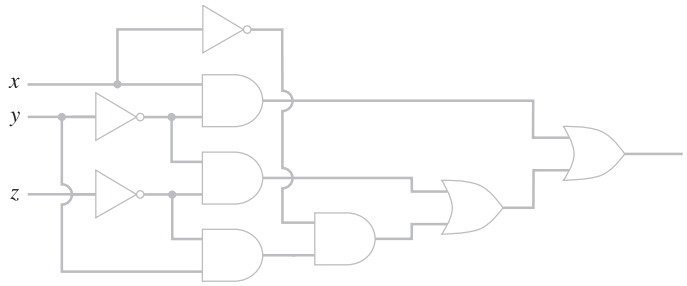
Sección 11.2

¿Son equivalentes los circuitos combinatorios en los ejercicios 5 y 6? Explique.

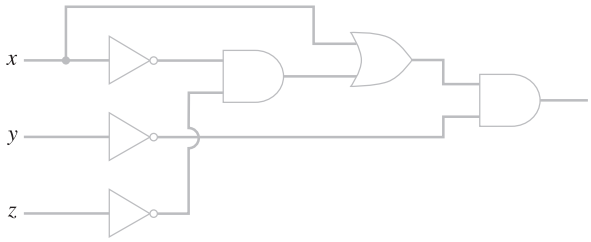
5.



6.



a)



b)

Pruebe o desapruebe las ecuaciones en los ejercicios 7 y 8.

7. $(x \wedge y) \vee (\overline{x} \wedge z) \vee (\overline{x} \wedge y \wedge \overline{z}) = y \vee (\overline{x} \wedge z)$

8. $(x \wedge y \wedge z) \vee (\overline{x \vee z}) = (x \wedge z) \vee (\overline{x} \wedge \overline{z})$

Sección 11.3

9. Si U es un conjunto universal y $S = \mathcal{P}(U)$, el conjunto potencia de U, entonces

$$(S, \cup, \cap, \overline{\ }, \emptyset, U)$$

es un álgebra booleana. Establezca las leyes de frontera y absorción para esta álgebra booleana.

10. Pruebe que en cualquier álgebra booleana, $(x(x + y \cdot 0))' = x'$ para toda x y y.

11. Escriba el dual de la afirmación del ejercicio 10 y pruébelo.

12. Sea U el conjunto de enteros positivos. Sea S una colección de subconjuntos finitos de U. ¿Por qué $(S, \cup, \cap, \overline{\ }, \emptyset, U)$ no es un álgebra booleana?

Sección 11.4

En los ejercicios 13 al 16, encuentre la forma disyuntiva normal de una expresión booleana que tiene una tabla lógica igual a la tabla que se incluye y dibuje el circuito correspondiente a la forma disyuntiva normal.

13.

x_1	x_2	x_3	y
1	1	1	0
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

14.

x_1	x_2	x_3	y
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

15.

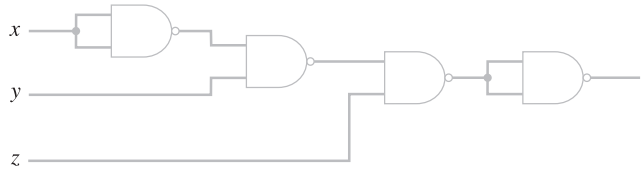
x_1	x_2	x_3	y
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	1

16.

x_1	x_2	x_3	y
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	1
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	0

Sección 11.5

17. Escriba la tabla lógica para el circuito



18. Encuentre una expresión booleana en la forma disyuntiva normal para el circuito del inciso *a)* del ejercicio 6. Use métodos algebraicos para simplificar la forma disyuntiva normal. Dibuje el circuito correspondiente a la expresión simplificada.

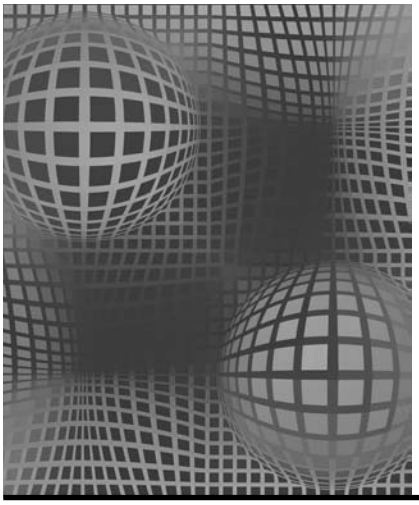
19. Diseñe un circuito sólo con compuertas NAND para calcular $x \oplus y$.

20. Diseñe un circuito sumador completo que use dos sumadores parciales y una compuerta OR.

Ejercicios para computadora

1. Escriba un programa que tenga como entrada una expresión booleana en x y y e imprima la tabla lógica de la expresión.
2. Escriba un programa que reciba como entrada una expresión booleana en x, y y z e imprima la tabla lógica de la expresión.

3. Escriba un programa que produzca la forma disyuntiva normal de una expresión booleana $p(x, y)$.
4. Escriba un programa que produzca la forma disyuntiva normal de una expresión booleana $p(x, y)$.
5. Escriba un programa que produzca la forma disyuntiva normal de una expresión booleana $p(x, y, z)$.
6. Escriba un programa que produzca la forma disyuntiva normal de una expresión booleana $p(x, y, z)$.
7. Escriba un programa que calcule los dos complementos de un número binario de n bits.



Capítulo 12

AUTÓMATAS, GRAMÁTICAS Y LENGUAJES

De hecho, siempre he tenido un vocabulario bastante extenso, sin mencionar una comprensión fenomenal de la gramática y un dominio superlativo de la sintaxis. Simplemente elijo no utilizarlos.

DE THE LITTLE RASCALS

En el capítulo 11 se analizaron los circuitos combinatorios en los que la salida depende sólo de la entrada. Estos circuitos carecen de memoria. En el presente capítulo se comenzará por estudiar los circuitos en los que la salida depende no sólo de la entrada, sino también del estado del sistema en el momento en que se introduce la entrada. El estado del sistema está determinado por el proceso anterior. En este sentido, estos circuitos tienen memoria; se conocen como *circuitos secuenciales* y su importancia es evidente en el diseño de computadoras.

Las máquinas de estado finito son modelos abstractos de máquinas con una memoria interna primitiva. Un autómata es un tipo especial de máquina de estado finito que tiene una relación estrecha con un tipo específico de lenguaje. En la última parte de este capítulo, se analizarán con cierto detalle las máquinas de estado finito, los autómatas de estado finito y los lenguajes.

12.1 ➜ Circuitos secuenciales y máquinas de estado finito

WWW

Las operaciones dentro de una computadora digital se llevan a cabo en intervalos discretos. La salida depende del estado del sistema al igual que de la entrada. Se supondrá que el estado del sistema cambia sólo en el tiempo $t = 0, 1, \ldots$. Una manera sencilla de introducir la secuenciación en los circuitos es introducir un **retraso unitario de tiempo**.

Definición 12.1.1 ▶

Un *retraso unitario de tiempo* acepta como entrada un bit x_t en el tiempo t y produce x_{t-1}, el bit recibido como entrada en el tiempo $t - 1$. El retraso unitario de tiempo se dibuja como se muestra en la figura 12.1.1.

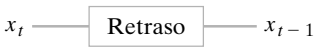


Figura 12.1.1 Retraso unitario de tiempo. ◀

Como ejemplo del uso del retraso unitario de tiempo, se analizará el **sumador en serie**.

Definición 12.1.2 ▶

Un *sumador en serie* acepta como entrada dos números binarios

$$x = 0x_N x_{N-1} \cdots x_0 \qquad \text{y} \qquad y = 0y_N y_{N-1} \cdots y_0$$

y produce la suma $z_{N+1}z_N \cdots z_0$ de x y y. Los números x y y se introducen de manera secuencial por pares, $x_0, y_0; \ldots; x_N, y_N; 0, 0$. Se produce la suma $z_0, z_1, \ldots, z_{N+1}$. ◀

Ejemplo 12.1.3 ▶

Circuito sumador en serie

En la figura 12.1.1 se ilustra un circuito que usa un retraso unitario de tiempo para implementar un sumador en serie.

Se mostrará la manera en que el sumador en serie calcula la suma de

$$x = 010 \quad \text{y} \quad y = 011.$$

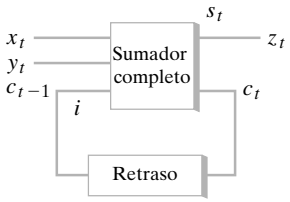


Figura 12.1.2 Circuito sumador en serie.

Primero se establece $x_0 = 0$ y $y_0 = 1$. (Se supone que en este instante $i = 0$. Esto se puede preparar haciendo $x = y = 0$). El estado del sistema se observa en la figura 12.1.3*a)*. Después, se hace $x_1 = y_1 = 1$. El retraso unitario de tiempo envía $i = 0$ como el tercer bit al sumador completo. El estado del sistema se ve en la figura 12.1.3*b)*. Por último, se hace $x_2 = y_2 = 0$. Esta vez el retraso unitario de tiempo envía $i = 1$ como el tercer bit al sumador completo. El estado del sistema se observa en la figura 12.1.3*c)*. Se obtiene la suma $z = 101$.

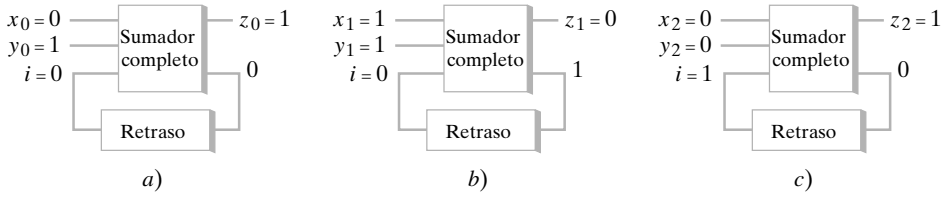


Figura 12.1.3 Cálculo de 010 + 011 con el circuito sumador en serie. ◀

Una **máquina de estado finito** es un modelo abstracto de una máquina con una memoria interna primitiva.

Definición 12.1.4 ▶

Una *máquina de estado finito M* consiste en

a) Un conjunto finito $\mathcal{I}$ de *símbolos de entrada*.
b) Un conjunto finito $\mathcal{O}$ de *símbolos de salida*.
c) Un conjunto finito $\mathcal{S}$ de *estados*.
d) Una *función f del siguiente estado* de $\mathcal{S} \times \mathcal{I}$ en $\mathcal{S}$.
e) Una *función g de salida* de $\mathcal{S} \times \mathcal{I}$ en $\mathcal{S}$.
f) Un *estado inicial* $\sigma \in \mathcal{S}$.

Se escribe $M = (\mathcal{I}, \mathcal{O}, \mathcal{S}, f, g, \sigma)$. ◀

Ejemplo 12.1.5 ▶

Sean $\mathcal{I} = \{a, b\}, \mathcal{O} = \{0, 1\}$, y $\mathcal{S} = \{\sigma_0, \sigma_1\}$. Defina un par de funciones $f: \mathcal{S} \times \mathcal{I} \to \mathcal{S}$ y $\mathcal{S} \times \mathcal{I} \to \mathcal{O}$ por las reglas dadas en la tabla 12.1.1.

TABLA 12.1.1 ■

	f		g	
$\mathcal{S}$ \ $\mathcal{I}$	a	b	a	b
σ_0	σ_0	σ_1	0	1
σ_1	σ_1	σ_1	1	0

Entonces $M = (\mathcal{I}, \mathcal{O}, \mathcal{S}, f, g, \sigma_0)$ es una máquina de estado finito.

La tabla 12.1.1 se interpreta como sigue:

$$\begin{aligned} f(\sigma_0, a) &= \sigma_0 \qquad g(\sigma_0, a) = 0, \\ f(\sigma_0, b) &= \sigma_1 \qquad g(\sigma_0, b) = 1, \\ f(\sigma_1, a) &= \sigma_1 \qquad g(\sigma_1, a) = 1, \\ f(\sigma_1, b) &= \sigma_1 \qquad g(\sigma_1, b) = 0. \end{aligned}$$

◀

Las funciones del siguiente estado y de salida también se pueden definir mediante un **diagrama de transición**. Antes de dar una definición formal de diagrama de transición, se ilustrará cómo se construye uno.

Ejemplo 12.1.6 ▶

Dibuje un diagrama de transición para la máquina de estado finito del ejemplo 12.1.5.

El diagrama de transición es una digráfica. Los vértices son los estados (figura 12.1.4). El estado inicial se indica con una flecha. Si estamos en el estado σ e introducir i produce una salida o y nos mueve al estado σ', se dibuja una arista dirigida del vértice σ al vértice σ' y se etiqueta i/o. Por ejemplo, si estamos en el estado σ_0 e introducimos a, la tabla 12.1.1 dice que producimos 0 y permanecemos en el estado σ_0. Entonces dibujamos un lazo dirigido en el vértice σ_0 y se etiqueta $a/0$ (figura 12.1.4). Por otro lado, si estamos en el estado σ_0 y se introduce b, producimos 1 y nos movemos al estado σ_1. Entonces, dibujamos una arista dirigida de σ_0 a σ_1 y se etiqueta $b/1$. Al considerar todas estas posibilidades, obtenemos el diagrama de transición de la figura 12.1.4. ◀

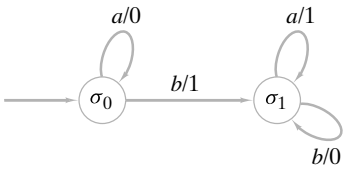


Figura 12.1.4 Diagrama de transición.

Definición 12.1.7 ▶

Sea $M = (\mathcal{I}, \mathcal{O}, \mathcal{S}, f, g, \sigma)$ una máquina de estado finito. El *diagrama de transición* de M es una digráfica G cuyos vértices son los miembros de $\mathcal{S}$. Una flecha designa el estado inicial σ. Se tiene una arista dirigida (σ_1, σ_2) en G si existe una entrada i con $f(\sigma_1, i) = \sigma_2$. En este caso, si $g(\sigma_1, i) = o$, la arista (σ_1, σ_2) tiene etiqueta i/o. ◀

Se puede ver la máquina de estado finito $M = (\mathcal{I}, \mathcal{O}, \mathcal{S}, f, g, \sigma)$ como una computadora sencilla. Iniciamos en el estado σ, introducimos una cadena sobre $\mathcal{I}$ y producimos una cadena de salida.

Definición 12.1.8 ▶

Sea $M = (\mathcal{I}, \mathcal{O}, \mathcal{S}, f, g, \sigma)$ una máquina de estado finito. Una *cadena de entrada* para M es una cadena sobre $\mathcal{I}$. La cadena

$$y_1 \cdots y_n$$

es la *cadena de salida* para M correspondiente a la cadena de entrada

$$\alpha = x_1 \cdots x_n$$

Si existen estados $\sigma_0, \ldots, \sigma_n \in \mathcal{S}$ con

$$\begin{aligned} \sigma_0 &= \sigma \\ \sigma_i &= f(\sigma_{i-1}, x_i) \qquad \text{for } i = 1, \ldots, n; \\ y_i &= g(\sigma_{i-1}, x_i) \qquad \text{for } i = 1, \ldots, n. \end{aligned}$$

◀

Ejemplo 12.1.9 ▶

Encuentre la cadena de salida correspondiente a la cadena de entrada

$$aababba \tag{12.1.1}$$

para la máquina de estado finito del ejemplo 12.1.5.

Al inicio estamos en el estado σ_0. El primer símbolo de entrada es a. Localizamos la arista que sale de σ_0 en el diagrama de transición de M (figura 12.1.4) etiquetado a/x, que dice que si entra a, sale x. En este caso, 0 es la salida. La arista señala el siguiente estado, σ_0. Después, a es de nuevo la entrada. Como antes, se produce 0 y nos quedamos en el estado σ_0. Luego, la entrada es b.

En este caso, producimos 1 y cambiamos al estado σ_1. Continuando de esta manera, se encuentra que la cadena de salida es

$$0011001. \tag{12.1.2}$$

◀

Ejemplo 12.1.10 ▶

Máquina de estado finito sumador en serie

Diseñe una máquina de estado finito que realice la suma en serie.

Se representará la máquina de estado finito por su diagrama de transición. Como el sumador en serie acepta pares de bits, el conjunto de entrada será

$$\{00, 01, 10, 11\}.$$

El conjunto de salida es

$$\{0, 1\}.$$

Dada una entrada xy, se toma una de dos acciones: se suman x y y, o se suman x, y y 1, dependiendo de si el bit de acarreo es 0 o 1. Entonces existen dos estados que se llamarán C (acarreo) y NC (ningún acarreo). El estado inicial es NC. En este punto se pueden dibujar los vértices y designar el estado inicial en el diagrama de transición (vea la figura 12.1.5).

Después se consideran las entradas posibles en cada vértice. Por ejemplo, si 00 es entrada a NC, debe producirse 0 y seguir en el estado NC. Entonces NC tiene un ciclo con etiqueta 00/0. Como otro ejemplo, si entra 11 a C, se calcula $1 + 1 + 1 = 11$. En este caso, la salida es 1 y se continúa en el estado C. Así, C tiene un ciclo con etiqueta 11/1. Como un último ejemplo, si estamos en el estado NC y la entrada es 11, debe producirse 0 y nos movemos al estado C. Al considerar todas las posibilidades, se llega al diagrama de transición de la figura 12.1.6.

Figura 12.1.5 Dos estados para la máquina de estado finito sumador en serie.

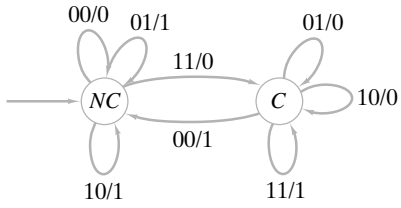


Figura 12.1.6 Máquina de estado finito que realiza la suma en serie.

◀

Ejemplo 12.1.11 ▶

El flip-flop *SR*

Un **flip-flop** es un componente básico de los circuitos digitales puesto que sirve como celda de memoria de un bit. El **flip-flop *SR*** (o **flip-flop set-reset**) se define mediante la tabla

S	R	Q
1	1	No permitido
1	0	1
0	1	0
0	0	$\begin{cases} 1 \text{ si el último valor de } S \text{ es 1} \\ 0 \text{ si el último valor de } R \text{ es 1} \end{cases}$

El flip-flop SR "recuerda" si el último valor de S o R era 1. (Si $Q = 1$, el último valor de S era 1; si $Q = 0$, el último valor de R era 1). Se puede modelar el flip-flop SR como una máquina de estado finito definiendo dos estados: "el último valor de S era 1" o "el último valor de R era 1" (vea la figura 12.1.7). Se define la entrada como los nuevos valores de S y R; la notación sr significa que $S = s$ y $R = r$. Se define Q como la salida. Se ha designado de manera arbitraria el estado inicial como "S era igual a 1". Una implementación del circuito secuencial del flip-flop SR se muestra en la figura 12.1.8.

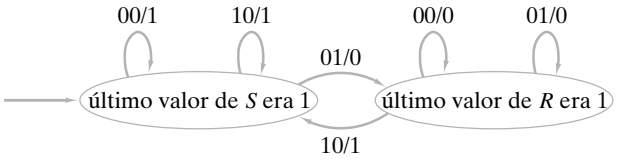


Figura 12.1.7 El flip-flop *SR* como máquina de estado finito.

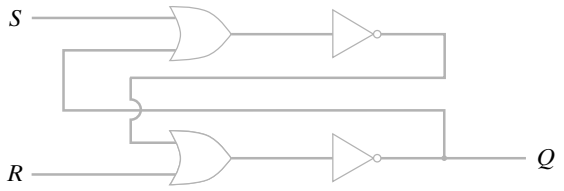


Figura 12.1.8 Implementación de un circuito secuencial del flip-flop *SR*. ◀

Sección de ejercicios de repaso

1. ¿Qué es un retraso unitario de tiempo?
2. ¿Qué es un sumador en serie?
3. Defina *máquina de estado finito*.
4. ¿Qué es un diagrama de transición?
5. ¿Qué es un flip-flop *SR*?

Ejercicios

En los ejercicios 1 al 5, dibuje el diagrama de transición de la máquina de estado finito $(\mathcal{I}, \mathcal{O}, \mathcal{S}, f, g, \sigma_0)$.

1. $\mathcal{I} = \{a, b\}, \mathcal{O} = \{0, 1\}, \mathcal{S} = \{\sigma_0, \sigma_1\}$

	f		g	
$\mathcal{S}$ \ $\mathcal{I}$	a	b	a	b
σ_0	σ_1	σ_1	1	1
σ_1	σ_0	σ_1	0	1

2. $\mathcal{I} = \{a, b\}, \mathcal{O} = \{0, 1\}, \mathcal{S} = \{\sigma_0, \sigma_1\}$

	f		g	
$\mathcal{S}$ \ $\mathcal{I}$	a	b	a	b
σ_0	σ_1	σ_0	0	0
σ_1	σ_0	σ_0	1	1

3. $\mathcal{I} = \{a, b\}, \mathcal{O} = \{0, 1\}, \mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2\}$

	f		g	
$\mathcal{S}$ \ $\mathcal{I}$	a	b	a	b
σ_0	σ_1	σ_1	0	1
σ_1	σ_2	σ_1	1	1
σ_2	σ_0	σ_0	0	0

4. $\mathcal{I} = \{a, b, c\}, \mathcal{O} = \{0, 1\}, \mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2\}$

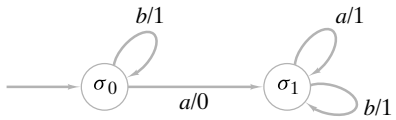
	f			g		
$\mathcal{S}$ \ $\mathcal{I}$	a	b	c	a	b	c
σ_0	σ_0	σ_1	σ_2	0	1	0
σ_1	σ_1	σ_1	σ_0	1	1	1
σ_2	σ_2	σ_1	σ_0	1	0	0

5. $\mathcal{I} = \{a, b, c\}, \mathcal{O} = \{0, 1, 2\}, \mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}$

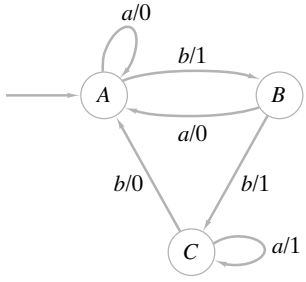
	f			g		
$\mathcal{S}$ \ $\mathcal{I}$	a	b	c	a	b	c
σ_0	σ_1	σ_0	σ_2	1	1	2
σ_1	σ_0	σ_2	σ_2	2	0	0
σ_2	σ_3	σ_3	σ_0	1	0	1
σ_3	σ_1	σ_1	σ_0	2	0	2

En los ejercicios 6 al 10, encuentre los conjuntos $\mathcal{I}$, $\mathcal{O}$ y $\mathcal{S}$, el estado inicial y la tabla que define el siguiente estado y las funciones de salida para cada máquina de estado finito.

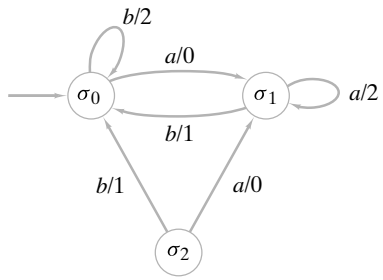
6.



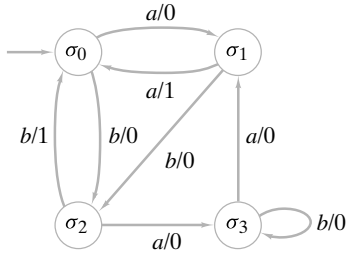
7.



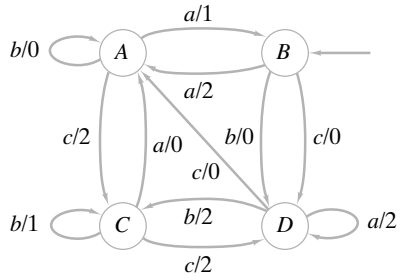
8.



9.



10.



En los ejercicios 11 al 20, encuentre la cadena de salida para la cadena de entrada que se indica y la máquina de estado finito.

11. *abba*; ejercicio 1

12. *abba*; ejercicio 2

13. *aabbaba*; ejercicio 3

14. *aabbcc*; ejercicio 4

15. *aabaab*; ejercicio 5

16. *aaa*; ejercicio 6

17. *aabbabaab*; ejercicio 7

18. *baaba*; ejercicio 8

19. *bbababbabaaa*; ejercicio 9

20. *cacbccbaabac*; ejercicio 10

En los ejercicios 21 al 26, diseñe una máquina de estado finito que tenga las propiedades indicadas. La entrada siempre es una cadena de bits.

21. Produce 1 si entra un número par de unos; de otra manera produce 0.

22. Produce 1 si entran k unos, donde k es un múltiplo de 3; de otra manera produce 0.

23. Produce 1 si entran dos unos o más; de otra manera produce 0.

24. Produce 1 cuando ve 101; de otra manera produce 0.

25. Produce 1 cuando ve 101 y en adelante; de otra manera produce 0.

26. Produce 1 cuando ve primero 0 y hasta que ve otro 0; en adelante, produce 0; en el resto de los casos produce 0.

27. Sea $\alpha = x_1 \cdots x_n$ una cadena de bits. Sea $\beta = y_1 \cdots y_n$, donde

$$y_i = \begin{cases} a & \text{si } x_i = 0 \\ b & \text{si } x_i = 1 \end{cases}$$

para $i = 1, \ldots, n$. Sea $\gamma = y_n \cdots y_1$.

Demuestre que si γ alimenta a la máquina de estado finito de la figura 12.1.4, la salida es el complemento de 2 de α (considere el algoritmo 11.5.16 una descripción del complemento de 2).

★28. Demuestre que no existe una máquina de estado finito que reciba una cadena de bits y produzca 1 siempre que el número de unos sea igual al número de ceros en la entrada, y produzca 0 de otra manera.

★29. Demuestre que no hay una máquina de estado finito que realice la multiplicación en serie. En particular, demuestre que no hay una máquina de estado finito que alimenta números binarios $X = x_1 \cdots x_n$, $Y = y_1 \cdots y_n$,, como una secuencia de números binarios de dos bits

$$x_n y_n, \quad x_{n-1} y_{n-1}, \quad \ldots, \quad x_1 y_1, \quad 00, \quad \ldots, \quad 00,$$

donde hay 00, y produce $z_{2n}, \ldots, z_1$, donde $Z = z_1 \cdots z_{2n} = XY$.

Ejemplo: Si existe tal máquina, para multiplicar 101 × 1001 se alimentaría 11,00,10,01,00,00,00,00. El primer par 11 es el par de los bits en la extrema derecha (10<u>1</u>, 100<u>1</u>); el segundo par 00 es el siguiente par de bits (1<u>0</u>1, 10<u>0</u>1); y así sucesivamente. Se amortigua la cadena de entrada con cuatro pares 00 —la longitud del número más grande 1001 que debe multiplicarse—. Como 101 × 1001 = 101101, se asevera que se obtiene la salida mostrada en la tabla adyacente.

Entrada	*Salida*
11	1
00	0
10	1
01	1
00	0
00	1
00	0
00	0

12.2 ➜ Autómata de estado finito

WWW

Un **autómata de estado finito** es un tipo especial de máquina de estado finito. Los autómatas de estado finito son de gran interés por su relación con los lenguajes, como se verá en la sección 12.5.

Definición 12.2.1 ▶ Un *autómata de estado finito* $A = (\mathcal{I}, \mathcal{O}, \mathcal{S}, f, g, \sigma_0)$ es una máquina de estado finito en la que el conjunto de símbolos de salida es $\{0, 1\}$ y donde el estado actual determina la última salida. Aquellos estados para los que la última salida fue 1 se llaman *estados de aceptación*. ◀

Ejemplo 12.2.2 ▶ Dibuje un diagrama de transición de la máquina de estado finito A definida por la tabla. El estado inicial es σ_0. Muestre que A es un autómata de estado finito y determine el conjunto de estados de aceptación.

	f		g	
$\mathcal{S}$ \ $\mathcal{I}$	a	b	a	b
σ_0	σ_1	σ_0	1	0
σ_1	σ_2	σ_0	1	0
σ_2	σ_2	σ_0	1	0

El diagrama de transición se muestra en la figura 12.2.1. Si estamos en el estado σ_0, la última salida fue 0. Si estamos en el estado σ_1 o σ_2, la última salida fue 1; entonces A es un autómata de estado finito. Los estados de aceptación son σ_1 y σ_2.

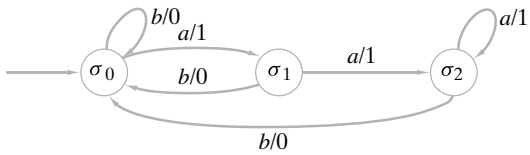


Figura 12.2.1 Diagrama de transición para el ejemplo 12.2.2.

◀

El ejemplo 12.2.2 muestra que la máquina de estado finito definida por un diagrama de transición será un autómata de estado finito si el conjunto de símbolos de salida es $\{0, 1\}$ y si, para cada estado σ, todas las aristas que llegan a σ tienen la misma etiqueta de salida.

El diagrama de transición de un autómata de estado finito suele dibujarse con los estados de aceptación en círculos dobles y sin los símbolos de salida. Cuando volvemos a dibujar el diagrama de transición de la figura 12.2.1 de esta manera, obtenemos el diagrama de transición de la figura 12.2.2.

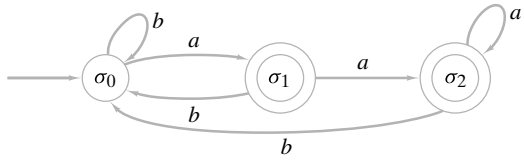


Figura 12.2.2 Diagrama de transición de la figura 12.2.1 vuelto a dibujar con los estados de aceptación en círculos dobles y sin los símbolos de salida.

Ejemplo 12.2.3 ▶ Dibuje el diagrama de transición del autómata de estado finito de la figura 12.2.3 como un diagrama de transición de una máquina de estado finito.

Como σ_2 es un estado de aceptación, se etiquetan todas sus aristas entrantes con salida 1 (vea la figura 12.2.4). Los estados σ_0 y σ_1 no son de aceptación, de modo que se etiquetan sus aristas entrantes con salida 0. Se obtiene el diagrama de transición de la figura 12.2.4.

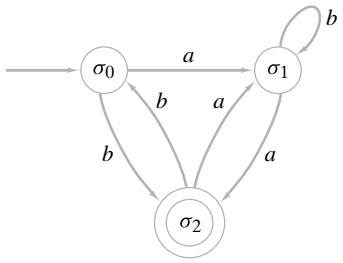


Figura 12.2.3 Autómata de estado finito.

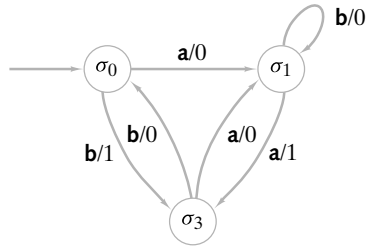


Figura 12.2.4 Autómata de estado finito de la figura 12.2.3 vuelto a dibujar como diagrama de transición de una máquina de estado finito.

◀

Como alternativa para la definición 12.2.1, se puede considerar un autómata de estado finito A como consistente en

1. Un conjunto finito $\mathcal{I}$ de *símbolos de entrada*
2. Un conjunto finito $\mathcal{S}$ de *estados*
3. Una *función f del siguiente estado* de $\mathcal{S} \times \mathcal{I}$ en $\mathcal{S}$
4. Un subconjunto $\mathcal{A}$ de $\mathcal{S}$ de *estados aceptantes.*
5. Un *estado inicial* $\sigma \in \mathcal{S}$.

Si usamos esta caracterización, se escribe $A = (\mathcal{I}, \mathcal{S}, f, \mathcal{A}, \sigma)$.

Ejemplo 12.2.4 ▶

En la figura 12.2.5 se ilustra el diagrama de transición del autómata de estado finito $A = (\mathcal{I}, \mathcal{S}, f, \mathcal{A}, \sigma)$, donde

$$\mathcal{I} = \{a, b\}, \quad \mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2\}, \quad \mathcal{A} = \{\sigma_2\}, \quad \sigma = \sigma_0,$$

y f está dada por la siguiente tabla:

	f	
$\mathcal{S}$ \ $\mathcal{I}$	a	b
σ_0	σ_0	σ_1
σ_1	σ_0	σ_2
σ_2	σ_0	σ_2

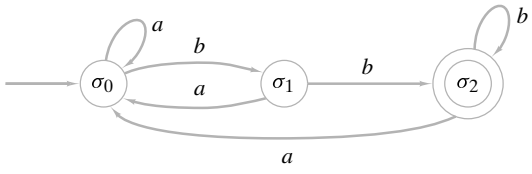


Figura 12.2.5 Diagrama de transición para el ejemplo 12.2.4.

◀

Si una cadena se alimenta a un autómata de estado finito, terminará ya sea en un estado de aceptación o en un estado de no aceptación. Este estado final determina si el autómata de estado finito **acepta** la cadena.

Definición 12.2.5 ▶

Sea $A = (\mathcal{I}, \mathcal{S}, f, \mathcal{A}, \sigma)$ un autómata de estado finito. Sea $\alpha = x_1, \ldots, x_n$ una cadena en $\mathcal{I}$. Si existen estados $\sigma_0, \ldots, \sigma_n$ que satisfacen

a) $\sigma_0 = \sigma$

b) $f(\sigma_{i-1}, x_i) = \sigma_i$ para $i = 1, \ldots, n$

c) $\sigma_n \in \mathcal{A}$,

se dice que A *acepta* a α. La cadena nula se acepta si y sólo si $\sigma \in \mathcal{A}$. Sea Ac(A) el conjunto de cadenas aceptadas por A; se dice que A *acepta* Ac(A).

Sea $\alpha = x_1, \ldots, x_n$ una cadena en $\mathcal{I}$. Defina los estados $\sigma_0, \ldots, \sigma_n$ por las condiciones *a)* y *b)* anteriores. La trayectoria (directa) $\sigma_0, \ldots, \sigma_n$ recibe el nombre de trayectoria que *representa* α en A. ◀

De la definición 12.2.5 se deduce que si la trayectoria P representa la cadena α en un autómata de estado finito A, entonces A acepta a α si y sólo si P termina en un estado de aceptación.

Ejemplo 12.2.6 ▶

¿La cadena *abaa* es aceptada por el autómata de estado finito de la figura 12.2.2?

Se comienza en el estado σ_0. Cuando se alimenta a, nos movemos al estado σ_1. Cuando se alimenta b, nos movemos al estado σ_0. Cuando se alimenta a, nos movemos al estado σ_1. Por último, cuando se alimenta el último símbolo a, nos movemos al estado σ_2. La trayectoria (σ_0, σ_1, σ_0, σ_1, σ_2) representa la cadena *abaa*. Como el estado final σ_2 es un estado de aceptación, la cadena *abaa* es aceptada por el autómata de estado finito de la figura 12.2.2. ◀

Ejemplo 12.2.7 ▶

¿La cadena $\alpha = abbabba$ es aceptada por el autómata de estado finito de la figura 12.2.3?

La trayectoria que representa α termina en σ_1. Como σ_1 no es un estado de aceptación, el autómata de estado finito de la figura 12.2.3 no acepta a la cadena α. ◀

Ahora se darán dos ejemplos que ilustran problemas de diseño.

Ejemplo 12.2.8 ▶

Diseñe un autómata de estado finito que acepte precisamente aquellas cadenas en $\{a, b\}$ que no contienen símbolos a.

La idea es usar dos conjuntos de estados:

A: Se encontró una a.
NA: No se encontró una a.

El estado NA es el estado inicial y el único estado de aceptación. Ahora es sencillo dibujar las aristas (vea la figura 12.2.6). Observe que el autómata de estado finito acepta correctamente la cadena nula. ◀

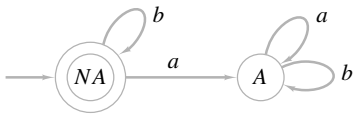


Figura 12.2.6 Autómata de estado finito que acepta de manera precisa las cadenas en $\{a, b\}$ que no contienen a.

Ejemplo 12.2.9 ▶

Diseñe un autómata de estado finito que acepte precisamente aquellas cadenas en $\{a, b\}$ que contienen un número impar de símbolos a.

Esta vez los dos estados son

E: Se encontró un número par de símbolos a.
O: Se encontró un número impar de símbolos a.

El estado inicial es E y el estado de aceptación es O. Se obtiene el diagrama de transición que aparece en la figura 12.2.7. ◀

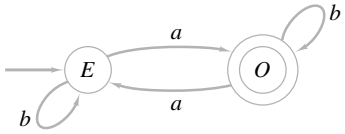


Figura 12.2.7 Autómata de estado finito que acepta precisamente esas cadenas en $\{a, b\}$ que contienen un número impar de símbolos a.

En esencia, un autómata de estado finito es un algoritmo para decidir si se acepta o no una cadena determinada. Como ejemplo, se convertirá el diagrama de transición de la figura 12.2.7 en un algoritmo.

Algoritmo 12.2.10

Este algoritmo determina si una cadena en $\{a, b\}$ es aceptada por un autómata de estado finito cuyo diagrama de transición está dado en la figura 12.2.7.

Entrada: n, longitud de la cadena ($n = 0$ designa la cadena nula); la cadena $s_1 s_2 \cdots s_n$
Salida: "aceptar" si la cadena se acepta
"rechazar" si la cadena no se acepta

```
fsa(s, n) {
    estado = 'E'
    for i = 1 to n {
        if (estado == 'E' ∧ s_i == 'a')
            estado = 'O'
        if (estado == 'O' ∧ s_i == 'a')
            estado = 'E'
    }
    if (estado == 'O')
        return "aceptar"
    else
        return "rechazar"
}
```

Si dos autómatas de estado finito aceptan precisamente las mismas cadenas, se dice que los autómatas son **equivalentes**.

Definición 12.2.11 ▶ Los autómatas de estado finito A y A' son *equivalentes* si $\text{Ac}(A) = \text{Ac}(A')$. ◀

Ejemplo 12.2.12 ▶ Se puede verificar que los autómatas de estado finito de la figura 12.2.6 y 12.2.8 son equivalentes (vea el ejercicio 33).

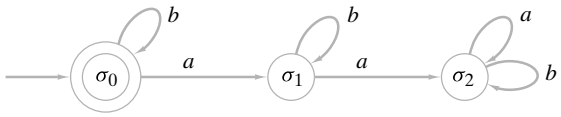


Figura 12.2.8 Autómata de estado finito equivalente al de la figura 12.2.6.

◀

Si se define una relación R en el conjunto de autómatas de estado finito por la regla $A \; R \; A'$ si A y A' son equivalentes (en el sentido de la definición 12.2.11), R es una relación de equivalencia. Cada clase de equivalencia consiste en un conjunto de autómatas de estado finito mutuamente equivalentes.

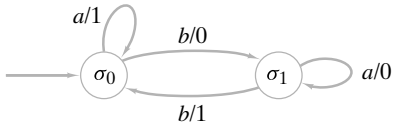
Sección de ejercicios de repaso

1. Defina *autómata de estado finito*.
2. ¿Qué significa para una cadena ser aceptada por un autómata de estado finito?
3. ¿Qué son autómatas de estado finito equivalentes?

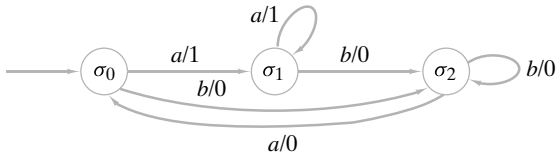
Ejercicios

En los ejercicios 1 al 3, demuestre que cada máquina de estado finito es un autómata de estado finito y dibuje de nuevo el diagrama como corresponde.

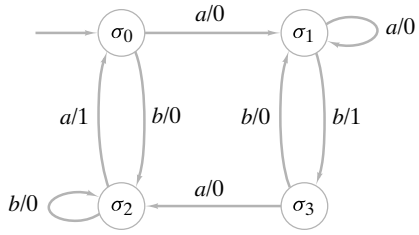
1.



2.

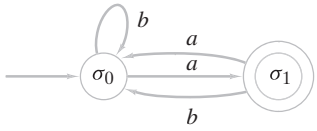


3.

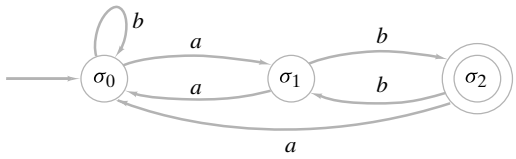


En los ejercicios 4 al 6, dibuje de nuevo el diagrama del autómata de estado finito como el diagrama de transición de una máquina de estado finito.

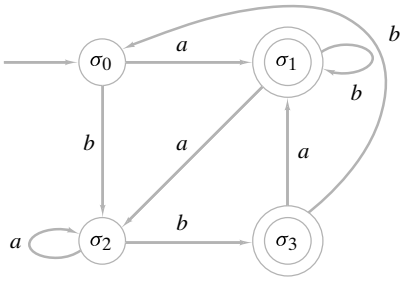
4.



5.



6.



En los ejercicios 7 al 9, dibuje el diagrama de transición del autómata de estado finito $(\mathcal{I}, \mathcal{S}, f, \mathcal{A}, \sigma_0)$.

7. $\mathcal{I} = \{a, b\}, \mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2\}, \mathcal{A} = \{\sigma_0\}$

	f	
$\mathcal{S}$ \ $\mathcal{I}$	a	b
σ_0	σ_1	σ_0
σ_1	σ_2	σ_0
σ_2	σ_0	σ_2

8. $\mathcal{I} = \{a, b\}, \mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2\}, \mathcal{A} = \{\sigma_0, \sigma_2\}$

	f	
$\mathcal{S}$ \ $\mathcal{I}$	a	b
σ_0	σ_1	σ_1
σ_1	σ_0	σ_2
σ_2	σ_0	σ_1

9. $\mathcal{I} = \{a, b, c\}, \mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}, \mathcal{A} = \{\sigma_0, \sigma_2\}$

	f		
$\mathcal{S}$ \ $\mathcal{I}$	a	b	c
σ_0	σ_1	σ_0	σ_2
σ_1	σ_0	σ_3	σ_0
σ_2	σ_3	σ_2	σ_0
σ_3	σ_1	σ_0	σ_1

10. Para cada autómata de estado finito en los ejercicios 1 al 6, encuentre los conjuntos $\mathcal{I}$, $\mathcal{S}$ y $\mathcal{A}$, el estado inicial, y la tabla que define la función del siguiente estado.

11. ¿Cuáles de las máquinas de estado finito de los ejercicios 1 al 10, de la sección 12.1, son autómatas de estado finito?

12. ¿Cómo debe verse la tabla de una máquina de estado finito *M* para que *M* sea un autómata de estado finito?

En los ejercicios 13 al 17, determine si la cadena que se indica es aceptada por el autómata de estado finito dado.

13. *abbaa*; figura 12.2.2

14. *abbaa*; figura 12.2.3

15. *aabaabb*; figura 12.2.5

16. *aaabbbaab*; ejercicio 5

17. *aaababbab*; ejercicio 6

18. Demuestre que el autómata de estado finito de la figura 12.2.2 acepta una cadena α en $\{a, b\}$ si y sólo si α termina con *a*.

19. Demuestre que el autómata de estado finito de la figura 12.2.5 acepta una cadena α en $\{a, b\}$ si y sólo si α termina con *bb*.

★**20.** Caracterice las cadenas aceptadas por el autómata de estado finito de los ejercicios 1 al 9.

En los ejercicios 21 al 31, dibuje el diagrama de transición de un autómata de estado finito que acepte el conjunto dado de cadenas en $\{a, b\}$.

21. Número par de símbolos *a*

22. Exactamente una *b*

23. Al menos una *b*

24. Exactamente dos símbolos *a*

25. Al menos dos símbolos *a*

26. Contiene *m* símbolos *a*, donde *m* es múltiplo de 3

27. Comienza con *baa*

★**28.** Contiene *abba*

29. Toda *b* va seguida por *a*

★***30.*** Termina con *aba*

★**31.** Comienza con *ab* y termina con *baa*

32. Escriba algoritmos similares al algoritmo 12.2.10, que decidan si los autómatas de estado finito de los ejercicios 1 al 9 aceptan o no una cadena dada.

33. Dé un argumento formal para mostrar que los autómatas de estado finito de las figuras 12.2.6 y 12.2.8 son equivalentes.

34. Sea *L* un conjunto de cadenas en $\{a, b\}$. Demuestre que existe un autómata de estado finito que acepta a *L*.

35. Sea L el conjunto de cadenas aceptadas por el autómata de estado finito del ejercicio 6. Sea S el conjunto de todas las cadenas en $\{a, b\}$. Diseñe un autómata de estado finito que acepte $S - L$.

36. Sea L_i el conjunto de cadenas aceptadas por el autómata de estado finito $A_i = (\mathcal{I}, \mathcal{S}_i, f_i, \mathcal{A}_i, \sigma_i)$, $i = 1, 2$. Sea

$$A = (\mathcal{I}, \mathcal{S}_1 \times \mathcal{S}_2, f, \mathcal{A}, \sigma),$$

donde

$$f((S_1, S_2), x) = (f_1(S_1, x), f_2(S_2, x))$$
$$\mathcal{A} = \{(A_1, A_2) \mid A_1 \in \mathcal{A}_1 \text{ y } A_2 \in \mathcal{A}_2\}$$
$$\sigma = (\sigma_1, \sigma_2).$$

Demuestre que $\text{Ac}(A) = L_1 \cap L_2$.

37. Sea L_i el conjunto de cadenas aceptadas por el autómata de estado finito $A_i = (\mathcal{I}, \mathcal{S}_i, f_i, \mathcal{A}_i, \sigma_i)$, $i = 1, 2$. Sea

$$A = (\mathcal{I}, \mathcal{S}_1 \times \mathcal{S}_2, f, \mathcal{A}, \sigma),$$

donde

$$f((S_1, S_2), x) = (f_1(S_1, x), f_2(S_2, x))$$
$$\mathcal{A} = \{(A_1, A_2) \mid A_1 \in \mathcal{A}_1 \text{ o } A_2 \in \mathcal{A}_2\}$$
$$\sigma = (\sigma_1, \sigma_2).$$

Demuestre que $\text{Ac}(A) = L_1 \cup L_2$.

En los ejercicios 38 al 42, sea $L_i = \text{Ac}(A_i)$, $i = 1, 2$. Dibuje el diagrama de transición del autómata de estado finito que acepta $L_1 \cap L_2$ y $L_1 \cup L_2$.

38. A_1 dado en el ejercicio 4; A_2 dado en el ejercicio 5

39. A_1 dado en el ejercicio 4; A_2 dado en el ejercicio 6

40. A_1 dado en el ejercicio 5; A_2 dado en el ejercicio 6

41. A_1 dado en el ejercicio 6; A_2 dado en el ejercicio 6

42. A_1 dado por la figura 12.5.7, sección 12.5; A_2 dado en el ejercicio 6

12.3 ➜ Lenguajes y gramáticas

Según el *Webster's New Collegiate Dictionary*, lenguaje es un "sistema de palabras y métodos para combinar palabras, usado y comprendido por una comunidad de tamaño considerable". Estos lenguajes, con frecuencia, se conocen como **lenguajes naturales** para distinguirlos de los **lenguajes formales**, que se usan para modelar los lenguajes naturales y comunicarse con las computadoras. Las reglas de un lenguaje natural son muy complejas y su caracterización completa es difícil. Por otro lado, es posible especificar por completo las reglas que sigue la construcción de ciertos lenguajes formales. Comenzaremos con una definición de lenguaje formal.

Definición 12.3.1 ▶

Sea A un conjunto finito. Un *lenguaje (formal)* L sobre A es un subconjunto de A^*, el conjunto de todas las cadenas sobre A. ◀

Ejemplo 12.3.2 ▶

Sea $A = \{a, b\}$. El conjunto L de todas las cadenas sobre A que contienen un número impar de símbolos a es un lenguaje sobre A. Como se vio en el ejemplo 12.2.9, L es precisamente el conjunto de cadenas sobre A aceptadas por el autómata de estado finito de la figura 12.2.7. ◀

Una manera de definir un lenguaje consiste en dar una lista de las reglas que se supone que el lenguaje debe obedecer.

Definición 12.3.3 ▶

Una *gramática de estructura de frases* (o simplemente *gramática*) G consiste en

a) Un conjunto finito N de *símbolos no terminales*

b) Un conjunto finito T de *símbolos terminales* donde $N \cap T = \emptyset$

c) Un subconjunto finito P de $[(N \cup T)^* - T^*] \times (N \cup T)^*$, llamado conjunto de *producciones*

d) Un *símbolo de inicio* $\sigma \in N$.

Se escribe $G = (N, T, P, \sigma)$ ◀

Una producción $(A, B) \in P$ suele escribirse

$$A \to B.$$

La definición 12.3.3*c)* establece que en la producción $A \to B$, $A \in (N \cup T)^* - T^*$ y $B \in (N \cup T)^*$; entonces, A debe incluir al menos un símbolo no terminal, mientras que B puede consistir en cualquier combinación de símbolos no terminales y terminales.

Ejemplo 12.3.4 ▶

Sea

$$N = \{\sigma, S\}$$
$$T = \{a, b\}$$
$$P = \{\sigma \to b\sigma, \sigma \to aS, S \to bS, S \to b\}.$$

Entonces $G = (N, T, P, \sigma)$ es un gramática. ◀

Dada una gramática G, se puede construir un lenguaje $L(G)$ a partir de G usando las producciones para derivar las cadenas que constituyen $L(G)$. La idea es comenzar con el símbolo de inicio y luego usar las producciones repetidas veces hasta que se obtiene una cadena de símbolos terminales. El lenguaje $L(G)$ es el conjunto de todas estas cadenas obtenidas. La definición 12.3.5 da los detalles formales.

Definición 12.3.5 ▶

Sea $G = (N, T, P, \sigma)$ una gramática.

Si $\alpha \to \beta$ es una producción y $x\alpha y \in (N \cup T)^*$, 6 decimos que $x\beta y$ *se puede derivar directamente* de $x\ \alpha\ y$ y se escribe

$$x\alpha y \Rightarrow x\beta y.$$

Si $\alpha_1 \in (N \cup T)^*$ para $i = 1, \ldots, n$, y α_{i+1} se deriva directamente de α_i para $i = 1, \ldots, n - 1$, se dice que α_n se *puede derivar de* α_1 y se escribe

$$\alpha_1 \Rightarrow \alpha_n.$$

Llamamos a

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_n$$

la *derivación de* α_n (*a partir de* α_1). Por convención, cualquier elemento de $(N \cup T)^*$ se puede derivar de sí mismo.

El *lenguaje generado por G*, escrito $L(G)$, consiste en todas las cadenas sobre T que se pueden derivar de σ. ◀

Ejemplo 12.3.6 ▶

Sea G la gramática del ejemplo 12.3.4.

La cadena *abSbb* se puede derivar directamente de *aSbb*, escrito

$$aSbb \Rightarrow abSbb,$$

usando la producción $S \to bS$.

La cadena *bbab* se puede derivar de σ, escrito

$$\sigma \Rightarrow bbab.$$

La derivación es

$$\sigma \Rightarrow b\sigma \Rightarrow bb\sigma \Rightarrow bbaS \Rightarrow bbab.$$

Las únicas derivaciones de σ son

$$\begin{aligned}
&\sigma \Rightarrow b\sigma \\
&\quad\vdots \\
&\Rightarrow b^n\sigma && n \geq 0 \\
&\Rightarrow b^n aS \\
&\quad\vdots \\
&\Rightarrow b^n ab^{m-1}S \\
&\Rightarrow b^n ab^m && n \geq 0, \quad m \geq 1.
\end{aligned}$$

Entonces, $L(G)$ consiste en las cadenas sobre $\{a, b\}$ que contienen precisamente una a y terminan con b. ◀

Una manera alternativa para establecer la producción de una gramática es usar la **forma regular de Backus** (o **forma de Backus-Naur** o **BNF**). En la forma BNF, los símbolos no terminales suelen comenzar con "<" y terminar con ">". La producción $S \to T$ se escribe $S ::= T$. Las producciones de la forma

WWW

$$S ::= T_1, \quad S ::= T_2, \quad \ldots, \quad S ::= T_n$$

se puede combinar como

$$S ::= T_1 \mid T_2 \mid \cdots \mid T_n.$$

La barra "|" se lee "o".

Ejemplo 12.3.7 ▶

Una gramática para enteros

Un entero se define como una cadena consistente en un signo opcional (+ o −) seguido de una cadena de dígitos (0 al 9). La siguiente gramática genera todos los enteros.

$$\begin{aligned}
&< \text{dígito} > ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9\\
&< \text{entero} > ::= < \text{entero con signo} > \mid < \text{entero sin signo} >\\
&< \text{entero con signo} > ::= + < \text{entero sin signo} > \mid - < \text{entero sin signo} >\\
&< \text{entero sin signo} > ::= < \text{dígito} > \mid < \text{dígito} > < \text{entero sin signo} >
\end{aligned}$$

El símbolo de inicio es < entero >.

Por ejemplo, la derivación del entero –901 es

$$\begin{aligned}
< \text{entero} > &\Rightarrow < \text{dígito con signo} >\\
&\Rightarrow - < \text{entero sin signo} >\\
&\Rightarrow - < \text{dígito} > < \text{entero sin signo} >\\
&\Rightarrow - < \text{dígito} > < \text{dígito} > < \text{entero sin signo} >\\
&\Rightarrow - < \text{dígito} > < \text{dígito} > < \text{dígito} >\\
&\Rightarrow - 9 < \text{dígito} > < \text{dígito} >\\
&\Rightarrow - 90 < \text{dígito} >\\
&\Rightarrow - 901.
\end{aligned}$$

En la notación de la definición 12.3.3, este lenguaje consiste en

1. El conjunto $N = \{< \text{dígito} >, < \text{entero} >, < \text{entero con signo}>, < \text{entero sin signo} >\}$ de símbolos no terminales
2. El conjunto $T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -\}$ de símbolos terminales
3. Las producciones

$$\begin{aligned}
&< \text{dígito} > \to 0, \ldots, < \text{dígito} > \to 9\\
&< \text{entero} > \to < \text{entero con signo} >\\
&< \text{entero} > \to < \text{entero sin signo} >\\
&< \text{entero con signo} > \to + < \text{entero sin signo} >\\
&< \text{entero con signo} > \to - < \text{entero sin signo} >\\
&< \text{entero sin signo} > \to < \text{dígito} >\\
&< \text{entero sin signo} > \to < \text{dígito} > < \text{entero sin signo} >
\end{aligned}$$

4. El símbolo de inicio < entero>. ◀

Es típico que los lenguajes de computadora, como FORTRAN, Pascal y C++ se especifiquen en BNF. El ejemplo 12.3.7 muestra cómo se puede especificar en BNF una constante entera en un lenguaje de computadora.

Las gramáticas se clasifican según los tipos de producciones que las definen.

Definición 12.3.8 ▶

Sea G una gramática y sea λ la cadena nula.

a) Si cada producción es de la forma

$$\alpha A\beta \to \alpha\delta\beta, \qquad \text{donde } \alpha, \beta \in (N \cup T)^*, \quad A \in N, \quad \delta \in (N \cup T)^* - \{\lambda\}, \tag{12.3.1}$$

G recibe el nombre de *gramática sensible al contexto* (o *tipo* 1).

b) Si cada producción es de la forma

$$A \to \delta, \qquad \text{donde } A \in N, \quad \delta \in (N \cup T)^*, \tag{12.3.2}$$

G recibe el nombre de *gramática libre de contexto* (o tipo 2).

c) Si cada producción es de la forma

$$A \to a \text{ o } A \to aB \text{ o } A \to \lambda, \qquad \text{donde } A, B \in N, \quad a \in T,$$

G recibe el nombre de *gramática regular* (o *tipo* 3). ◀

De acuerdo con la forma (12.3.1), en una gramática sensible al contexto, se puede sustituir A por δ si A está en el contexto de α y β. En una gramática libre de contexto, (12.3.2) establece que se puede sustituir A por δ en cualquier momento. Una gramática regular tiene reglas de sustitución especialmente sencillas: se sustituye un símbolo no terminal por un símbolo terminal, por un símbolo terminal seguido de un símbolo no terminal, o por la cadena nula.

Observe que una gramática regular es una gramática libre de contexto y que una gramática libre de contexto sin producciones de la forma $A \to \lambda$ es una gramática sensible al contexto.

Algunas definiciones permiten sustituir a por una cadena de terminales en la definición 12.3.8*c)*; sin embargo, se puede demostrar (vea el ejercicio 32) que las dos definiciones producen los mismos lenguajes.

Ejemplo 12.3.9 ▶

La gramática G definida por

$$T = \{a, b, c\}, \qquad N = \{\sigma, A, B, C, D, E\},$$

con producciones

$$\sigma \to aAB, \quad \sigma \to aB, \quad A \to aAC, \quad A \to aC, \quad B \to Dc,$$
$$D \to b, \quad CD \to CE, \quad CE \to DE, \quad DE \to DC, \quad Cc \to Dcc$$

y símbolo de inicio σ, es sensible al contexto. Por ejemplo, la producción $CE \to DE$ dice que se puede sustituir C por D si C va seguido de E, y la producción $Cc \to Dcc$ dice que se puede sustituir C por Dc si C va seguido de c.

Se puede derivar DC a partir de CD, ya que

$$CD \Rightarrow CE \Rightarrow DE \Rightarrow DC.$$

La cadena $a^3b^3c^3$ está en $L(G)$, ya que se tiene

$$\sigma \Rightarrow aAB \Rightarrow aaACB \Rightarrow aaaCCDc \Rightarrow aaaDCCc \Rightarrow aaaDCDcc$$
$$\Rightarrow aaaDDCcc \Rightarrow aaaDDDccc \Rightarrow aaabbbccc.$$

Se puede demostrar (vea el ejercicio 33) que

$$L(G) = \{a^n b^n c^n \mid n = 1, 2, \ldots\}.$$ ◀

Es natural permitir que el lenguaje $L(G)$ herede una propiedad de una gramática G. La siguiente definición precisa este concepto.

Definición 12.3.10 ▶

Un lenguaje L es *sensible al contexto* (respectivamente, *libre de contexto, regular*) si existe una gramática sensible al contexto (respectivamente, libre de contexto, regular) G con $L = L(G)$. ◀

Ejemplo 12.3.11 ▶

Según el ejemplo 12.3.9, el lenguaje

$$L = \{a^n b^n c^n \mid n = 1, 2, \ldots\}$$

es sensible al contexto. Se puede demostrar (vea [Hopcroft]) que no existe una gramática libre de contexto G con $L = L(G)$; así, L no es un lenguaje libre de contexto. ◀

Ejemplo 12.3.12 ▶

La gramática G definida por

$$T = \{a, b\}, \qquad N = \{\sigma\},$$

con producciones

$$\sigma \to a\sigma b, \qquad \sigma \to ab$$

y símbolo de inicio σ, es libre de contexto. Las únicas derivaciones de σ son

$$\begin{aligned} \sigma &\Rightarrow a\sigma b \\ &\;\;\vdots \\ &\Rightarrow a^{n-1}\sigma b^{n-1} \\ &\Rightarrow a^{n-1}abb^{n-1} = a^n b^n. \end{aligned}$$

Entonces, $L(G)$ consiste en las cadenas sobre $\{a, b\}$ de la forma $a^n b^n$, $n - 1, 2 \ldots$. Este lenguaje es libre de contexto. En la sección 12.5 (vea el ejemplo 12.5.6), mostraremos que $L(G)$ no es regular. ◀

A partir de los ejemplos 12.3.11 y 12.3.12, se deduce que el conjunto de lenguajes libres de contexto que no contienen la cadena nula es un subconjunto propio del conjunto de lenguajes sensibles al contexto y que el conjunto de lenguajes regulares es un subconjunto propio del conjunto de los lenguajes libres de contexto. También se puede demostrar que hay lenguajes que no son sensibles al contexto.

Ejemplo 12.3.13 ▶

La gramática G definida en el ejemplo 12.3.4 es regular. Entonces el lenguaje

$$L(G) = \{b^n ab^m \mid n = 0, 1, \ldots; m = 1, 2, \ldots\}$$

que genera es regular. ◀

Ejemplo 12.3.14 ▶

La gramática del ejemplo 12.3.7 es libre de contexto pero no regular. Sin embargo, si se cambian las producciones a

$$\begin{aligned} &< \text{entero} > ::= + < \text{entero sin signo} > \mid - < \text{entero sin signo} > \mid \\ &\qquad 0 < \text{dígito} > \mid 1 < \text{dígito} > \mid \cdots \mid 9 < \text{dígito} > \\ &< \text{entero sin signo} > ::= 0 < \text{dígito} > \mid 1 < \text{dígito} > \mid \cdots \mid 9 < \text{dígito} > \\ &< \text{dígito} > ::= 0 < \text{dígito} > \mid 1 < \text{dígito} > \mid \cdots \mid 9 < \text{dígito} > \mid \lambda, \end{aligned}$$

la gramática que se obtiene es regular. Como el lenguaje generado no cambia, se concluye que el conjunto de cadenas que representan enteros es un lenguaje regular. ◀

El ejemplo 12.3.14 origina la siguiente definición.

Definición 12.3.15 ▶

Las gramáticas G y G' son *equivalentes* si $L(G) = L(G')$. ◀

Ejemplo 12.3.16 ▶

Las gramáticas de los ejemplos 12.3.7 y 12.3.14 son equivalentes. ◀

Si se define una relación R en un conjunto de gramáticas por la regla $G\,R\,G'$ si G y G' son equivalentes (en el sentido de la definición 12.3.15), R es una relación de equivalencia. Cada clase de equivalencia consiste en un conjunto de gramáticas mutuamente equivalentes.

Se cierra esta sección con una introducción breve de otro tipo de gramáticas que resultan útiles para generar curvas fractales.

Definición 12.3.17 ▶

Una *gramática de Lindenmayer interactiva libre de contexto* consiste en

a) Un conjunto finito N de *símbolos no terminales*

b) Un conjunto finito T de *símbolos terminales*, donde $N \cap T = \emptyset$

WWW *c)* Un conjunto finito P de las *producciones* $A \to B$, donde $A \in N \cup T$ y $B \in (N \cup T)^*$

d) Un *símbolo de inicio* $\sigma \in N$. ◀

La diferencia entre una gramática de Lindenmayer interactiva libre de contexto y una gramática libre de contexto es que la primera permite producciones de la forma $A \to B$, donde A es una terminal o no terminal. (En una gramática libre de contexto, A debe ser una no terminal).

Las reglas para derivar las cadenas en una gramática de Lindenmayer interactiva libre de contexto son diferentes de las reglas para derivar cadenas en una gramática de estructura de frases (vea la definición 12.3.5). En una gramática de Lindenmayer interactiva libre de contexto, para derivar la cadena β de todas las cadenas α, *todos* los símbolos en α debe sustituirse *simultáneamente*. La definición formal es la siguiente.

Definición 12.3.18 ▶

Sea $G = (N, T, P, \sigma)$ una gramática de Lindenmayer interactiva libre de contexto. Si

$$\alpha = x_1 \cdots x_n$$

y hay producciones

$$x_i \to \beta_i$$

en P, para $i = 1, \ldots, n$, se escribe

$$\alpha \Rightarrow \beta_1 \cdots \beta_n$$

y se dice que $\beta_1 \cdot \cdot \beta_n$ se *puede derivar directamente* de α. Si α_{i+1} se puede derivar de α_i para $i = 1, \ldots, n-1$, se dice que α_n se *puede derivar de* α_1 y se escribe

$$\alpha_1 \Rightarrow \alpha_n.$$

Las implicaciones

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_n$$

se llaman *derivación de* α_n (*a partir de* α_1). Los *lenguajes generados por* G, escrito $L(G)$, consisten en todas las cadenas sobre T que se pueden derivar de σ. ◀

Ejemplo 12.3.19 ▶

Copo de nieve de von Koch

Sea

$$N = \{D\}$$
$$T = \{d, +, -\}$$
$$P = \{D \to D - D + + D - D,\ D \to d,\ + \to +,\ - \to -\}.$$

WWW Se interpreta $G(N, T, P, D)$ como una gramática de Lindenmayer libre de contexto. Como un ejemplo de una derivación a partir de D se tiene

$$D \Rightarrow D - D + + D - D \Rightarrow d - d + + d - d.$$

Entonces $d - d + + d - d \in L(G)$.

Ahora se impone un significado para las cadenas en $L(G)$. El símbolo d se interpreta como un comando para dibujar una línea recta de longitud fija en la dirección actual; el signo $+$ se interpreta como un comando para girar a la derecha 60°; el signo $-$ se interpre-

ta como un comando para girar a la izquierda 60°. Si comenzamos a la izquierda y el primer movimiento es horizontal a la derecha, cuando la cadena $d - d + + d - d$ se interpreta, se obtiene la curva mostrada en la figura 12.3.1*a)*.

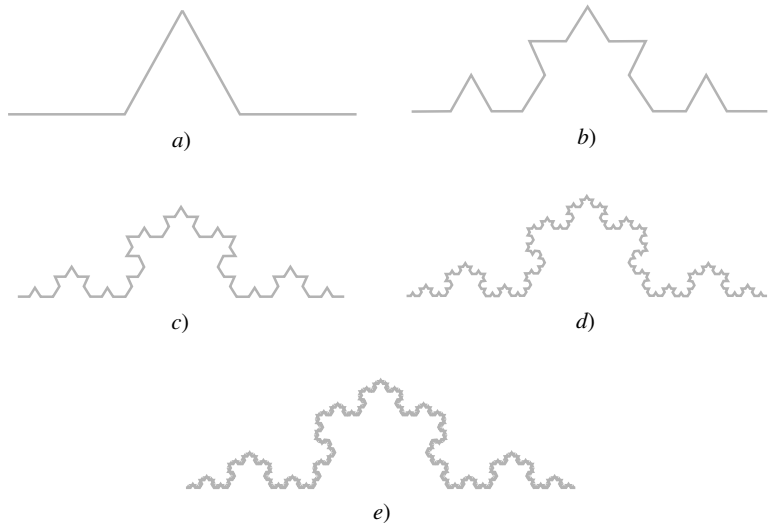


Figura 12.3.1 Copos de nieve de von Koch.

La siguiente cadena más larga en $L(G)$ es

$$d - d + + d - d - d - d + + d - d + + d - d + + d - d - d - d + + d - d,$$

cuya derivación es

$$\begin{aligned} D &\Rightarrow D - D + + D - D \\ &\Rightarrow D - D + + D - D - D - D + + D - D + + D \\ &\quad - D + + D - D - D - D + + D - D \\ &\Rightarrow d - d + + d - d - d - d + + d - d + + d \\ &\quad - d + + d - d - d - d + + d - d. \end{aligned}$$

No es posible encontrar una cadena más corta porque *todos* los símbolos deben sustituirse al mismo tiempo usando las producciones (definición 12.3.18). Si se sustituyen algunas D por d y otras D por $D - D + + D - D$, no se puede derivar una cadena de la cadena que resulta, mucho menos la cadena terminal, ya que d no ocurre en el lado izquierdo de cualquier producción.

Cuando se interpreta la cadena

$$d - d + + d - d - d - d + + d - d + + d - d + + d - d - d - d + + d - d$$

se obtiene la curva mostrada en la figura 12.3.1*b)*.

Las curvas obtenidas por interpretación de las cadenas más largas siguientes en $L(G)$ se ilustran en la figura 12.3.1*c)* a *e)*. Estas curvas se conocen como **copos de nieve de von Koch**. ◀

Las curvas como los copos de nieve de von Koch se llaman **curvas fractales** (vea [Peitgen]). Una característica de las curvas fractales es que una parte del todo es semejante al todo. Por ejemplo, como se observa en la figura 12.3.2, cuando la parte indicada del copo de nieve de von Koch se extrae y amplifica, se parece al original.

Las gramáticas de Lindenmayer interactivas sensibles al contexto fueron inventadas en 1968 por A. Lindenmayer (vea [Lindenmayer]) para modelar el crecimiento de las plantas. Como sugiere el ejemplo 12.3.19, estas gramáticas se pueden emplear en gráficas de computadora para generar imágenes (vea [Prusinkiewicz 1986, 1988; Smith]). Se puede demostrar (vea [Wood, p. 503]) que la clase de lenguajes generados por gramáticas de Lindenmayer sensibles al contexto es exactamente la misma que la clase de lenguajes generados por gramáticas de estructura de frases.

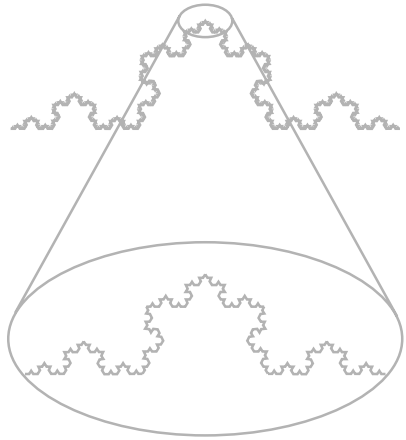


Figura 12.3.2 Naturaleza de fractal del copo de nieve de von Koch. Cuando se extrae y amplifica la parte superior del copo de nieve, se parece al original.

Sección de ejercicios de repaso

1. Compare los lenguajes natural y formal.
2. Defina una *gramática de estructura de frases*.
3. ¿Qué es una cadena que se puede derivar directamente?
4. ¿Qué es una cadena que se puede derivar?
5. ¿Qué es una derivación?
6. ¿Qué es el lenguaje generado por una gramática?
7. ¿Qué es la forma regular de Backus?
8. Defina *gramática sensible al contexto*.
9. Defina *gramática libre de contexto*.
10. Defina *gramática regular*.
11. ¿Qué gramática es equivalente a una gramática tipo 1?
12. ¿Qué gramática es equivalente a una gramática tipo 2?
13. ¿Qué gramática es equivalente a una gramática tipo 3?
14. Defina *lenguaje sensible al contexto.*
15. Defina *lenguaje libre de contexto.*
16. Defina *lenguaje regular.*
17. ¿Qué es una gramática de Lindenmayer interactiva libre de contexto?
18. ¿Cómo se genera el copo de nieve de von Koch?
19. ¿Qué es una curva fractal?

Ejercicios

En los ejercicios 1 al 6, determine si la gramática dada es sensible al contexto, libre de contexto, regular o ninguna de ellas. Dé todas las caracterizaciones pertinentes.

1. $T = \{a, b\}$, $N = \{\sigma, A\}$, con producciones

$$\sigma \to b\sigma, \qquad \sigma \to aA, \qquad A \to a\sigma,$$
$$A \to bA, \qquad A \to a, \qquad \sigma \to b,$$

y símbolo de inicio σ.

2. $T = \{a, b, c\}$, $N = \{\sigma, A, B\}$, con producciones

$$\sigma \to AB, \qquad AB \to BA, \qquad A \to aA,$$
$$B \to Bb, \qquad A \to a, \qquad B \to b,$$

y símbolo de inicio σ.

3. $T = \{a, b\}$, $N = \{\sigma, A, B\}$, con producciones

$$\sigma \to A, \qquad \sigma \to AAB, \qquad Aa \to ABa,$$
$$A \to aa, \qquad Bb \to ABb, \qquad AB \to ABB,$$
$$B \to b,$$

y símbolo de inicio σ.

4. $T = \{a, b, c\}$, $N = \{\sigma, A, B\}$, con producciones

$$\sigma \to BAB, \qquad \sigma \to ABA, \qquad A \to AB,$$
$$B \to BA, \qquad A \to aA, \qquad A \to ab,$$
$$B \to b,$$

y símbolo de inicio σ.

5.

$$< S > ::= b < S > | \, a < A > | \, a$$
$$< A > ::= a < S > | \, b < B >$$
$$< B > ::= b < A > | \, a < S > | \, b$$

y símbolo de inicio $< S >$.

6. $T = \{a, b\}$, $N = \{\sigma, A, B\}$, con producciones

$$\sigma \to AA\sigma, \qquad AA \to B, \qquad B \to bB, \qquad A \to a$$

y símbolo de inicio σ.

En los ejercicios 7 al 11, demuestre que la cadena α está en $L(G)$ para la gramática G dando una derivación de α.

7. *bbabbab*, ejercicio 1 **8.** *abab*, ejercicio 2

9. *aabbaab*, ejercicio 3 **10.** *abbbaabab*, ejercicio 4

11. *abaabbabba*, ejercicio 5

12. Escriba las gramáticas de los ejemplos 12.3.4 y 12.3.9 y los ejercicios 1 al 4 y 6 en la BNF.

★**13.** Sea G la gramática del ejercicio 1. Demuestre que $\alpha \in L(G)$ si y sólo si α no es nula y contiene un número par de símbolos a.

★**14.** Sea G la gramática del ejercicio 5. Caracterice $L(G)$.

En los ejercicios 15 al 24, escriba una gramática que genere las cadenas que tienen la propiedad señalada.

15. Cadenas sobre $\{a, b\}$ que inician con a

16. Cadenas sobre $\{a, b\}$ que terminan con ba

17. Cadenas sobre $\{a, b\}$ que contienen ba

★**18.** Cadenas sobre $\{a, b\}$ que no terminan con ba

19. Enteros sin ceros a la izquierda

20. Números de punto flotante (números como .294, 89., 67.284)

21. Números exponenciales (números que incluyen punto flotante y números como 6.9E3, 8E12, 9.6E–4, 9E–10)

22. Expresiones booleanas en $X_1, \ldots, X_n$

23. Todas las cadenas sobre $\{a, b\}$

24. Cadenas $x_1, \ldots, x_n$ sobre $\{a, b\}$ con $x_1 \cdot \cdot \cdot x_n = x_n \cdot \cdot \cdot x_1$

Cada gramática en los ejercicios 25 al 31 se propone como generadora del conjunto L de cadenas sobre {a, b} que contienen el mismo número de símbolos a y b. Si la gramática genera L, demuéstrelo. Si la gramática no genera L, dé un contraejemplo y pruebe que su contraejemplo es correcto. En cada gramática, S es el símbolo de inicio.

25. $S \to aSb \mid bSa \mid \lambda$

26. $S \to aSb \mid bSa \mid SS \mid \lambda$

27. $S \to aB \mid bA \mid \lambda,\ B \to b \mid bA,\ A \to a \mid aB$

28. $S \to abS \mid baS \mid aSb \mid bSa \mid \lambda$

29. $S \to aSb \mid bSa \mid abS \mid baS \mid Sab \mid Sba \mid \lambda$

30. $S \to aB \mid bA,\ A \to a \mid SA,\ B \to b \mid SB$

31. $S \to aSbS \mid bSaS \mid \lambda$

★**32.** Sea G una gramática y sea λ la cadena nula. Demuestre que si cada producción es de la forma

$$A \to \alpha \text{ o } A \to \alpha B \text{ o } A \to \lambda,$$
$$\text{donde } A, B \in N, \quad \alpha \in T^* - \{\lambda\},$$

existe una gramática regular G' con $L(G) = L(G')$.

★**33.** Sea la gramática del ejemplo 12.3.9. Demuestre que

$$L(G) = \{a^n b^n c^n \mid n = 1, 2, \ldots\}.$$

34. Demuestre que el lenguaje

$$\{a^n b^n c^k \mid n, k \in \{1, 2, \ldots\}\}$$

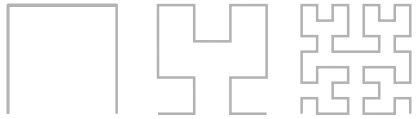
es un lenguaje libre de contexto.

35. Sea

$$N = \{S, D\}$$
$$T = \{d, +, -\}$$
$$P = \{S \to D + D + D + D,$$
$$D \to D + D - D - DD + D + D - D \mid d,$$
$$+ \to +, - \to -\}.$$

Considere a $G = (N, T, P, S)$ como una gramática de Lindenmayer libre de contexto. Interprete el símbolo d como un comando para dibujar una línea recta de longitud fija en la dirección actual; interprete $+$ como un comando para girar 90° a la derecha, e interprete – como un comando para girar 90° a la izquierda. Genere las dos cadenas más pequeñas en $L(G)$ y dibuje las curvas correspondientes. Estas curvas se conocen como *islas cuadráticas de Koch*.

★**36.** La siguiente figura



representa las primeras tres etapas de la *curva de Hilbert*. Defina una gramática de Lindenmayer libre de contexto que genere cadenas que cuando se interpretan de manera adecuada generen la curva de Hilbert.

12.4 ➜ Autómata de estado finito no determinístico

En esta sección y la siguiente, se muestra que las gramáticas regulares y los autómatas de estado finito son en esencia lo mismo, en el sentido de que cualquiera es una especificación de un lenguaje regular. Se comienza con un ejemplo que ilustra cómo se puede convertir un autómata de estado finito en una gramática regular.

Ejemplo 12.4.1 ▶

Escriba la gramática regular dada por el autómata de estado finito de la figura 12.2.7.

Los símbolos terminales son los símbolos de entrada $\{a, b\}$. Los estados E y O se convierten en símbolos no terminales. El estado inicial E se convierte en el símbolo de inicio. Las producciones corresponden a las aristas dirigidas. Si hay una arista dirigida con etiqueta x de S a S', se escribe la producción

$$S \to xS'.$$

En nuestro caso, se obtienen las producciones

$$E \to bE, \qquad E \to aO, \qquad O \to aE, \qquad O \to bO. \tag{12.4.1}$$

Además, si S es un estado de aceptación, se incluye en la producción

$$S \to \lambda.$$

En nuestro caso, se obtiene la producción adicional

$$O \to \lambda. \tag{12.4.2}$$

Entonces la gramática $G = (N, T, P, E)$, con $N = \{O, E\}$, $T = \{a, b\}$ y P que consiste en las producciones (12.4.1) y (12.4.2), genera el lenguaje $L(G)$, que es el mismo que el conjunto de cadenas aceptadas por el autómata de estado finito de la figura 12.2.7. ◀

Teorema 12.4.2

Sea A un autómata de estado finito dado como diagrama de transición. Sea σ el estado inicial. Sea T el conjunto de símbolos de entrada y sea N el conjunto de estados. Sea P el conjunto de producciones

$$S \to xS'$$

si existe una arista etiquetada x de S a S′ y

$$S \to \lambda$$

si S es un estado de aceptación. Sea G la gramática regular

$$G = (N, T, P, \sigma).$$

Entonces, el conjunto de cadenas aceptadas por A es igual a L(G).

Demostración Primero se demuestra que $\text{Ac}(A) \subseteq L(G)$. Sea $\alpha \in \text{Ac}(A)$. Si α es la cadena nula, entonces σ es un estado de aceptación. En este caso, G contiene la producción

$$\sigma \to \lambda.$$

La derivación

$$\sigma \Rightarrow \lambda \tag{12.4.3}$$

muestra que $\alpha \in L(G)$.

Ahora suponga que $\alpha \in \text{Ac}(A)$ y α no es la cadena nula. Entonces $\alpha = x_1 \cdot \cdot \cdot x_n$ para algunas $x_i \in T$. Como A acepta a α, existe una trayectoria $(\sigma, S_1, \ldots, S_n)$, donde S_n es un estado de aceptación, con aristas etiquetadas sucesivamente $x_1, \ldots, x_n$. Se deduce que G contiene las producciones

$$\begin{aligned} \sigma &\to x_1 S_1 \\ S_{i-1} &\to x_i S_i \qquad \text{para } i = 2, \ldots, n. \end{aligned}$$

Como S_n es un estado de aceptación, G también contiene la producción

$$S_n \to \lambda\ .$$

La derivación

$$\begin{aligned} \sigma &\Rightarrow x_1 S_1 \\ &\Rightarrow x_1 x_2 S_2 \\ &\ \ \vdots \\ &\Rightarrow x_1 \cdots x_n S_n \\ &\Rightarrow x_1 \cdots x_n \end{aligned} \tag{12.4.4}$$

muestra que $\alpha \in L(G)$.

La prueba se completa al demostrar que $L(G) \subseteq \text{Ac}(A)$. Suponga que $\alpha \in L(G)$. Si α es la cadena nula, α debe obtenerse de la derivación (12.4.3), ya que la derivación que inicia con cualquier otra producción llevaría a una cadena no nula. Entonces la producción $\sigma \to \lambda$ está en la gramática. Por lo tanto, σ es un estado de aceptación en A. Se deduce que $\alpha \in \text{Ac}(A)$.

Ahora suponga que $\alpha \in L(G)$ y que α no es la cadena nula. Entonces $\alpha = x_1 \cdot \cdot \cdot x_n$ para alguna $x_i \in T$. Se deduce que hay una derivación de la forma (12.4.4). Si, en el diagrama de transición, se comienza en σ y se traza la trayectoria $(\sigma, S_1, \ldots, S_n)$, se puede generar la cadena α. La última producción empleada en (12.4.4) es $S_n \to \lambda$; entonces el último estado alcanzado es un estado aceptante. Por lo tanto, α es aceptada por A, de manera que $L(G) \subseteq \text{Ac}(A)$. Esto completa la prueba.

Ahora se considera la situación inversa. Dada una gramática regular G, se desea construir un autómata de estado finito A de manera que $L(G)$ sea precisamente el conjunto de cadenas aceptadas por A. Podría parecer, a primera vista, que bastaría con invertir el procedimiento del Teorema 12.4.2. Sin embargo, el siguiente ejemplo indica que la situación es un poco más compleja.

Ejemplo 12.4.3 ▶

Considere la gramática regular definida por

$$T = \{a, b\}, \qquad N = \{\sigma, C\},$$

con producciones

$$\sigma \to b\sigma, \qquad \sigma \to aC, \qquad C \to bC, \qquad C \to b$$

y símbolo inicial α.

Las no terminales se convierten en estados con α como el estado inicial. Para cada producción de la forma

$$S \to xS',$$

se dibuja una arista del estado S al estado S' y se etiqueta con x. Las producciones

$$\sigma \to b\sigma, \qquad \sigma \to aC, \qquad C \to bC$$

dan la gráfica que se reproduce en la figura 12.4.1. La producción $C \to b$ es equivalente a las dos producciones

$$C \to bF, \qquad F \to \lambda,$$

donde F es un símbolo adicional no terminal. Las producciones

$$\sigma \to b\sigma, \qquad \sigma \to aC, \qquad C \to bC, \qquad C \to bF$$

dan la gráfica que aparece en la figura 12.4.2. La producción

$$F \to \lambda$$

nos dice que F debe ser un estado de aceptación (vea la figura 12.4.2). ◀

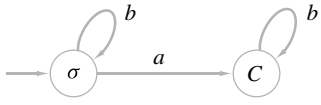


Figura 12.4.1 Gráfica correspondiente a las producciones $\sigma \to b\sigma, \sigma \to aC, C \to bC$.

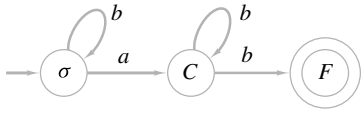


Figura 12.4.2 Autómata de estado finito no determinístico correspondiente a la gramática $\sigma \to b\sigma, \sigma \to aC, C \to bC, C \to b$.

Desafortunadamente, la gráfica de la figura 12.4.2 no es un autómata de estado finito. Existen varios problemas. El vértice C no tiene arista que sale etiquetada a y el vértice F no tiene aristas de salida. Además, al vértice C tiene dos aristas de salida etiquetadas con b. Un diagrama como el de la figura 12.4.2 define otro tipo de autómata llamado **autómata de estado finito no determinístico**. La razón del calificativo "no determinístico" es que cuando estamos en un estado que tiene múltiples aristas de salida todas con la misma etiqueta x, si x se alimenta, la situación es no determinística, pues se tienen varias opciones para los siguientes estados. Por ejemplo, si en la figura 12.4.2 estamos en el estado C y se alimenta b, se tienen opciones para los siguientes estados: se puede permanecer en el estado C o ir al estado F.

Definición 12.4.4 ▶ *Un autómata A de estado finito no determinístico* consiste en

a) Un conjunto finito $\mathcal{I}$ de *símbolos de entrada*

b) Un conjunto finito $\mathcal{S}$ de *estados*

c) Una *función f del siguiente estado* de $\mathcal{S} \times \mathcal{I}$ en $\mathcal{P}(\mathcal{S})$

d) Un subconjunto $\mathcal{A}$ *de* $\mathcal{S}$ de *estados de aceptación*

e) Un *estado inicial* $\alpha \in \mathcal{S}$.

Se escribe $A = (\mathcal{I}, \mathcal{S}, f, \mathcal{A}, \sigma)$. ◀

La única diferencia entre un autómata de estado finito no determinístico y un autómata de estado finito es que en este último la función del siguiente estado lleva al estado definido de manera única, mientras que en un autómata de estado finito no determinístico la función del siguiente estado lleva a un conjunto de estados.

Ejemplo 12.4.5 ▶ Para el autómata de estado finito no determinístico de la figura 12.4.2, se tiene

$$\mathcal{I} = \{a, b\}, \qquad \mathcal{S} = \{\sigma, C, F\}, \qquad \mathcal{A} = \{F\}.$$

El estado inicial es σ y la función del siguiente estado está dada por

	f	
$\mathcal{S}$ \ $\mathcal{I}$	a	b
σ	$\{C\}$	$\{\sigma\}$
C	$\emptyset$	$\{C, F\}$
F	$\emptyset$	$\emptyset$

◀

Dibujar el diagrama de transición de un autómata de estado finito no determinístico es similar a dibujar el del autómata de estado finito. Se coloca una arista del estado S a cada estado en el conjunto $f(S, x)$ y se etiqueta cada una con x.

Ejemplo 12.4.6 ▶ El diagrama de transición del autómata de estado finito no determinístico

$$\mathcal{I} = \{a, b\}, \qquad \mathcal{S} = \{\sigma, C, D\}, \qquad \mathcal{A} = \{C, D\}$$

con estado inicial σ y función del siguiente estado

	f	
$\mathcal{S}$ \ $\mathcal{I}$	a	b
σ	$\{\sigma, C\}$	$\{D\}$
C	$\emptyset$	$\{C\}$
D	$\{C, D\}$	$\emptyset$

se muestra en la figura 12.4.3.

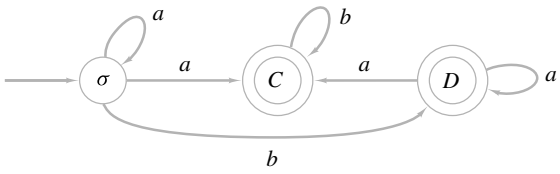


Figura 12.4.3 Diagrama de transición del autómata de estado finito no determinístico del ejemplo 12.4.6. ◀

Una cadena α es aceptada por un autómata de estado finito no determinístico A si existe una trayectoria que represente a α en el diagrama de transición de A que comience en el estado inicial y termine en un estado de aceptación. La siguiente es la definición formal.

Definición 12.4.7 ▶

Sea $A = (\mathcal{I}, \mathcal{S}, f, \mathcal{A}, \sigma)$ un autómata de estado finito no determinístico. La cadena nula es *aceptada* por A si y sólo si $\sigma \in \mathcal{A}$. Si $\alpha = x_1 \cdot \cdot \cdot x_n$ es una cadena no nula sobre $\mathcal{I}$ y existen estados $\sigma_0, \ldots, \sigma_n$ que satisfacen las siguientes condiciones:

a) $\sigma_0 = \sigma$

b) $\sigma_i \in f(\sigma_{i-1}, x_i)$ para $i = 1, \ldots, n$

c) $\sigma_n \in \mathcal{A}$,

se dice que α es *aceptada* por A. Si se iguala $\text{Ac}(A)$ al conjunto de cadenas aceptadas por A, se dice que A *acepta* a $\text{Ac}(A)$.

Si A y A' son autómatas de estado finito no determinísticos y $\text{Ac}(A) = \text{Ac}(A')$, se dice que A y A' son *equivalentes*.

Si $\alpha = x_1 \cdot \cdot \cdot x_n$ es una cadena sobre $\mathcal{I}$ y existen estados $\sigma_0, \ldots, \sigma_n$ que satisfacen las condiciones *a)* y *b)*, la trayectoria $(\sigma_0, \ldots, \sigma_n)$ se llama *trayectoria que representa a* α en A. ◀

Ejemplo 12.4.8 ▶

La cadena

$$\alpha = bbabb$$

es aceptada por el autómata de estado finito no determinístico de la figura 12.4.2, ya que la trayectoria $(\sigma, \sigma, \sigma, C, C, F)$, que termina en un estado aceptante, representa a α. Observe que la trayectoria $P = (\sigma, \sigma, \sigma, C, C, C)$ también representa a α, pero P no termina en un estado de aceptación. De todas maneras, la cadena α se acepta porque existe al menos una trayectoria que representa a α y que termina en un estado aceptante. Una cadena β no será aceptada si ninguna trayectoria la representa o si todas las trayectorias que la representan terminan en un estado de no aceptación. ◀

Ejemplo 12.4.9 ▶

La cadena $\alpha = aabaabbb$ es aceptada por el autómata de estado finito no determinístico de la figura 12.4.3. El lector debería localizar la trayectoria que representa a α, que termina en el estado C. ◀

Ejemplo 12.4.10 ▶

La cadena $\alpha = abba$ no es aceptada por el autómata de estado finito no determinístico de la figura 12.4.3. Comenzando en σ, cuando se alimenta a se tienen dos opciones: ir a C o permanecer en σ. Si se va a C, cuando se alimentan dos b, los movimientos están determinados y se permanece en C. Pero ahora, cuando se alimenta la a final, no hay arista por la cual moverse. Por otro lado, suponga que cuando se alimenta la primera a, se permanece en σ. Entonces, cuando se alimenta b nos movemos a D. Pero ahora cuando se alimenta la siguiente b, no hay arista por la cual moverse. Como no hay una trayectoria que represente α en la figura 12.4.3, la cadena α no es aceptada por el autómata de estado finito no determinístico de la figura 12.4.3. ◀

Se formulará la construcción del ejemplo 12.4.3 como un teorema.

Teorema 12.4.11

Sea $G = (N, T, P, \sigma)$ *una gramática regular. Sea*

$$\mathcal{I} = T$$
$$\mathcal{S} = N \cup \{F\}, \text{ donde } F \notin N \cup T$$
$$f(S, x) = \{S' \mid S \to xS' \in P\} \cup \{F \mid S \to x \in P\}$$
$$\mathcal{A} = \{F\} \cup \{S \mid S \to \lambda \in P\}.$$

Entonces el autómata de estado finito no determinístico $A = (\mathcal{I}, \mathcal{S}, f, \mathcal{A}, \sigma)$ acepta precisamente las cadenas $L(G)$.

Demostración La prueba, en esencia, es la misma que la del Teorema 12.4.2 y, por lo tanto, se omite.

Puede parecer que un autómata de estado finito no determinístico es un concepto más general que un autómata de estado finito; sin embargo, en la siguiente sección se demostrará que dado un autómata de estado finito no determinístico A, se puede construir un autómata de estado finito que sea equivalente a A.

Sección de ejercicios de repaso

1. Dado un autómata de estado finito A, ¿cómo se puede construir una gramática regular G de manera que el conjunto de cadenas aceptadas por A sea igual al lenguaje generado por G?
2. ¿Qué es un autómata de estado finito no determinístico?
3. ¿Qué significa que una cadena sea aceptada por un autómata de estado finito no determinístico?
4. ¿Qué son autómatas de estado finito no determinísticos equivalentes?
5. Dada una gramática regular G, ¿cómo se puede construir un autómata A de estado finito no determinístico de manera que el lenguaje generado por G sea igual a las cadenas aceptadas por A?

Ejercicios

En los ejercicios 1 al 5, dibuje el diagrama de transición del autómata de estado finito no determinístico $(\mathcal{I}, \mathcal{S}, f, \mathcal{A}, \sigma_0)$.

1. $\mathcal{I} = \{a, b\}, \mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2\}, \mathcal{A} = \{\sigma_0\}$

$\mathcal{S}$ \ $\mathcal{I}$	a	b
σ_0	$\emptyset$	$\{\sigma_1, \sigma_2\}$
σ_1	$\{\sigma_2\}$	$\{\sigma_0, \sigma_1\}$
σ_2	$\{\sigma_0\}$	$\emptyset$

2. $\mathcal{I} = \{a, b\}, \mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2\}, \mathcal{A} = \{\sigma_0, \sigma_1\}$

$\mathcal{S}$ \ $\mathcal{I}$	a	b
σ_0	$\{\sigma_1\}$	$\{\sigma_0, \sigma_2\}$
σ_1	$\emptyset$	$\{\sigma_2\}$
σ_2	$\{\sigma_1\}$	$\emptyset$

3. $\mathcal{I} = \{a, b\}, \mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}, \mathcal{A} = \{\sigma_1\}$

$\mathcal{S}$ \ $\mathcal{I}$	a	b
σ_0	$\emptyset$	$\{\sigma_3\}$
σ_1	$\{\sigma_1, \sigma_2\}$	$\{\sigma_3\}$
σ_2	$\emptyset$	$\{\sigma_0, \sigma_1, \sigma_3\}$
σ_3	$\emptyset$	$\emptyset$

4. $\mathcal{I} = \{a, b, c\}, \mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2\}, \mathcal{A} = \{\sigma_0\}$

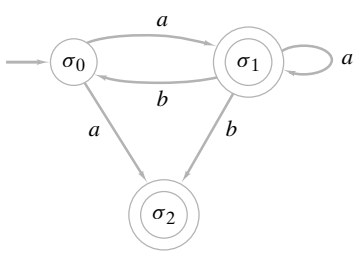
$\mathcal{S}$ \ $\mathcal{I}$	a	b	c
σ_0	$\{\sigma_1\}$	$\emptyset$	$\emptyset$
σ_1	$\{\sigma_0\}$	$\{\sigma_2\}$	$\{\sigma_0, \sigma_2\}$
σ_2	$\{\sigma_0, \sigma_1, \sigma_2\}$	$\{\sigma_0\}$	$\{\sigma_0\}$

5. $\mathcal{I} = \{a, b, c\}, \mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}, \mathcal{A} = \{\sigma_0, \sigma_3\}$

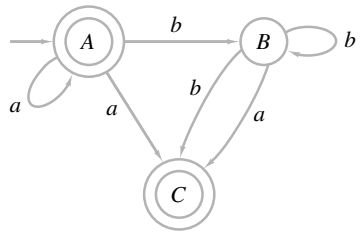
$\mathcal{S}$ \ $\mathcal{I}$	a	b	c
σ_0	$\{\sigma_1\}$	$\{\sigma_0, \sigma_1, \sigma_3\}$	$\emptyset$
σ_1	$\{\sigma_2, \sigma_3\}$	$\emptyset$	$\emptyset$
σ_2	$\emptyset$	$\{\sigma_0, \sigma_3\}$	$\{\sigma_1, \sigma_2\}$
σ_3	$\emptyset$	$\emptyset$	$\{\sigma_0\}$

Para cada autómata de estado finito no determinístico en los ejercicios 6 al 10, encuentre los conjuntos $\mathcal{I}$, $\mathcal{S}$, y $\mathcal{A}$, el estado inicial y la tabla que define la función del siguiente estado.

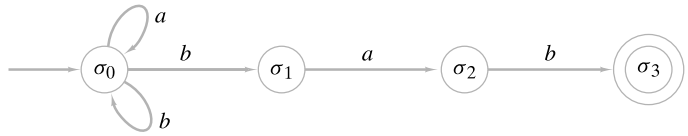
6.



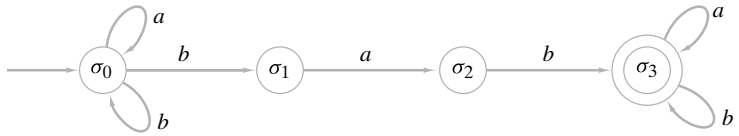
7.



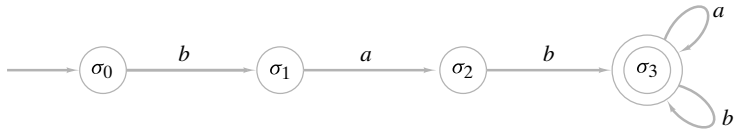
8.



9.



10.



11. Escriba las gramáticas regulares dadas por el autómata de estado finito de los ejercicios 4 al 9, sección 12.2.

12. Represente las gramáticas de los ejercicios 1 y 5, sección 12.3 y el ejemplo 12.3.14 por un autómata de estado finito no determinístico.

13. ¿El autómata de estado finito no determinístico de la figura 12.4.2 acepta la cadena *bbabbb*? Pruebe su respuesta.

14. ¿El autómata de estado finito no determinístico de la figura 12.4.2 acepta la cadena *bbabab*? Pruebe su respuesta.

15. Demuestre que una cadena α sobre $\{a, b\}$ es aceptada por el autómata de estado finito no determinístico de la figura 12.4.2 si y sólo si α contiene exactamente una a y termina con b.

16. ¿La cadena *aaabba* es aceptada por el autómata de estado finito no determinístico de la figura 12.4.3? Pruebe su respuesta.

17. ¿La cadena *aaaab* es aceptada por el autómata de estado finito no determinístico de la figura 12.4.3? Pruebe su respuesta.

18. Caracterice las cadenas aceptadas por el autómata de estado finito no determinístico de la figura 12.4.3.

19. Muestre que las cadenas aceptadas por el autómata de estado finito no determinístico del ejercicio 8 son precisamente las cadenas sobre $\{a, b\}$ que terminan con *bab*.

★20. Caracterice las cadenas aceptadas por el autómata de estado finito no determinístico de los ejercicios 1 al 7, 9 y 10.

Diseñe un autómata de estado finito no determinístico que acepte las cadenas sobre $\{a, b\}$ que tienen las propiedades especificadas en los ejercicios 21 al 29.

21. Comienzan con *abb* o con *ba*

22. Terminan con *abb* o con *ba*

23. Contienen *abb* o *ba*

★24. Contienen *bab* y *bb*

25. Cada *b* está precedida y seguida por una *a*

26. Comienzan con *abb* y terminan con *ab*

★27. Comienzan con *ab* pero no terminan con *ab*

28. No contienen *ba* ni *bbb*

★29. No contienen *abba* ni *bbb*

30. Escriba las gramáticas regulares que generan las cadenas de los ejercicios 21 al 29.

12.5 ➜ Relaciones entre lenguajes y autómatas

En la sección anterior se demostró (Teorema 12.4.2) que si A es un autómata de estado finito, existe una gramática regular G, con $L(G) = \text{Ac}(A)$. Como un inverso parcial, se demostró (Teorema 12.4.11) que si G es una gramática regular, existe un autómata A de estado finito no determinístico con $L(G) = \text{Ac}(A)$. En esta sección se demuestra (Teorema 12.5.4) que si G es una gramática regular, existe un autómata de estado finito A con $L(G) = \text{Ac}(A)$. Este resultado se deducirá del Teorema 12.4.11, demostrando que cualquier autómata de

estado finito no determinístico se puede convertir en un autómata de estado finito equivalente (Teorema 12.5.3). Primero se ilustra el método mediante un ejemplo.

Ejemplo 12.5.1 ▶

Encuentre un autómata de estado finito equivalente al autómata de estado finito no determinístico de la figura 12.4.2.

El conjunto de símbolos de entrada no cambia. Los estados consisten en todos los subconjuntos

$$\emptyset, \quad \{\sigma\}, \quad \{C\}, \quad \{F\}, \quad \{\sigma, C\}, \quad \{\sigma, F\}, \quad \{C, F\}, \quad \{\sigma, C, F\}$$

del conjunto original de estados $\mathcal{S} = \{\sigma, C, F\}$. El estado inicial es $\{\sigma\}$. Los estados de aceptación son todos los subconjuntos

$$\{F\}, \quad \{\sigma, F\}, \quad \{C, F\}, \quad \{\sigma, C, F\}$$

de $\mathcal{S}$ que contienen un estado aceptante del autómata de estado finito no determinístico original.

Se dibuja una arista de X a Y y se etiqueta x si $X = \emptyset = Y$ o si

$$\bigcup_{S \in X} f(S, x) = Y.$$

Se obtiene el autómata de estado finito de la figura 12.5.1. Los estados

$$\{\sigma, F\}, \quad \{\sigma, C\}, \quad \{\sigma, C, F\}, \quad \{F\},$$

que no se alcanzan nunca, se pueden eliminar. Entonces, se obtiene el autómata de estado finito equivalente simplificado de la figura 12.5.2.

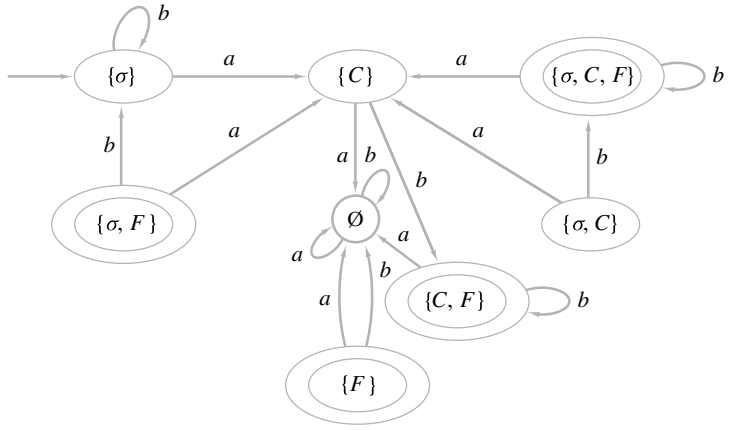


Figura 12.5.1 Autómata de estado finito equivalente al autómata de estado finito no determinístico de la figura 12.4.2.

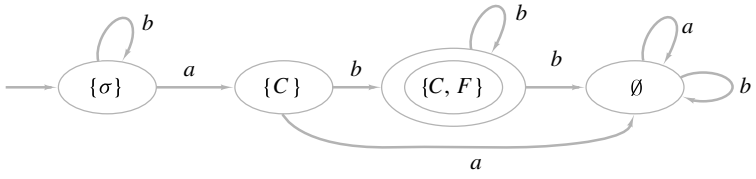


Figura 12.5.2 Versión simplificada de la figura 12.5.1 (eliminando los estados que no se alcanzan).

◀

Ejemplo 12.5.2 ▶

El autómata de estado finito equivalente al autómata de estado finito no determinístico del ejemplo 12.4.6 se ilustra en la figura 12.5.3.

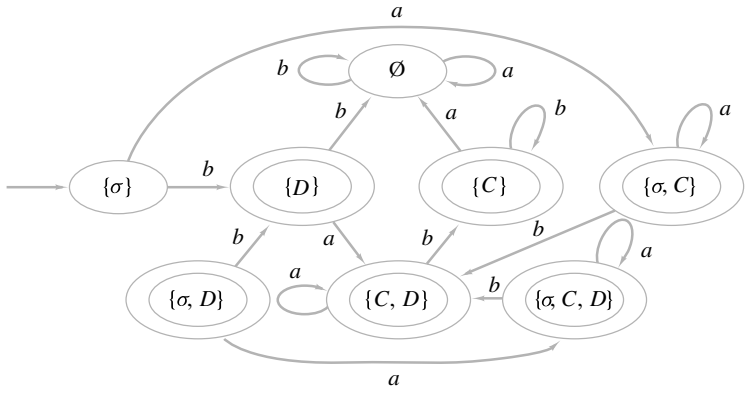


Figura 12.5.3 Autómata de estado finito equivalente al autómata de estado finito no determinístico de la figura 12.4.6.

◀

Ahora se dará la justificación formal del método de los ejemplos 12.5.1 y 12.5.2.

Teorema 12.5.3

Sea $A = (\mathcal{I}, \mathcal{S}, f, \mathcal{A}, \sigma)$ *un autómata de estado finito no determinístico. Sea*

a) $\mathcal{S}' = \mathcal{P}(\mathcal{S})$

b) $\mathcal{I}' = \mathcal{I}$

c) $\sigma' = \{\sigma\}$

d) $\mathcal{A}' = \{X \subseteq \mathcal{S} \mid X \cap \mathcal{A} \neq \emptyset\}$

e) $f'(X, x) = \begin{cases} \emptyset & si\ X = \emptyset \\ \bigcup_{S \in X} f(S, x) & si\ X \neq \emptyset. \end{cases}$

Entonces el autómata de estado finito $A' = (\mathcal{I}', \mathcal{S}', f', \mathcal{A}', \sigma')$ *es equivalente a* A.

Demostración Suponga que la cadena $\alpha = x_1 \cdots x_n$ es aceptada por A. Entonces existen estados $\sigma_0, \ldots, \sigma_n \in \mathcal{S}$ con

$$\begin{aligned} &\sigma_0 = \sigma \\ &\sigma_i \in f(\sigma_{i-1}, x_i) \quad \text{para } i = 1, \ldots, n \\ &\sigma_n \in \mathcal{A}. \end{aligned}$$

Se establece que $Y_0 = \{\sigma_0\}$ y

$$Y_i = f'(Y_{i-1}, x_i) \quad \text{para } i = 1, \ldots, n.$$

Como

$$Y_1 = f'(Y_0, x_1) = f'(\{\sigma_0\}, x_1) = f(\sigma_0, x_1),$$

se deduce que $\sigma_1 \in Y_1$. Ahora

$$\sigma_2 \in f(\sigma_1, x_2) \subseteq \bigcup_{S \in Y_1} f(S, x_2) = f'(Y_1, x_2) = Y_2.$$

De nuevo,

$$\sigma_3 \in f(\sigma_2, x_3) \subseteq \bigcup_{S \in Y_2} f(S, x_3) = f'(Y_2, x_3) = Y_3.$$

El argumento puede continuar (formalmente, se usaría inducción) para demostrar que $\sigma_n \in Y_n$. Como σ_n es un estado de aceptación en A, Y_n es un estado de aceptación en A'.

Entonces, en A' se tiene

$$\begin{aligned} f'(\sigma', x_1) &= f'(Y_0, x_1) = Y_1 \\ f'(Y_1, x_2) &= Y_2 \\ &\vdots \\ f'(Y_{n-1}, x_n) &= Y_n. \end{aligned}$$

Por lo tanto, α es aceptado por A'.

Ahora suponga que la cadena $\alpha = x_1 \cdot \cdot \cdot x_n$ es aceptada por A'. Entonces existen subconjuntos $Y_0, \ldots, Y_n$ de $\mathcal{S}$ tales que

$$\begin{aligned} Y_0 &= \sigma' = \{\sigma\} \\ f'(Y_{i-1}, x_i) &= Y_i \quad \text{para } i = 1, \ldots, n; \end{aligned}$$

además, existe un estado $\sigma_n \in Y_n \cap \mathcal{A}$. Como

$$\sigma_n \in Y_n = f'(Y_{n-1}, x_n) = \bigcup_{S \in Y_{n-1}} f(S, x_n),$$

existe $\sigma_{n-1} \in Y_{n-1}$ con $\sigma_n \in f(\sigma_{n-1}, x_n)$. De manera similar, como

$$\sigma_{n-1} \in Y_{n-1} = f'(Y_{n-2}, x_{n-1}) = \bigcup_{S \in Y_{n-2}} f(S, x_{n-1}),$$

existe $\sigma_{n-2} \in Y_{n-2}$ con $\sigma_{n-1} \in f(\sigma_{n-2}, x_{n-1})$. Si se continúa de esta manera, se obtiene

$$\sigma_i \in Y_i \quad \text{para } i = 0, \ldots, n,$$

con

$$\sigma_i \in f(\sigma_{i-1}, x_i) \quad \text{para } i = 1, \ldots, n.$$

En particular,

$$\sigma_0 \in Y_0 = \{\sigma\}.$$

Entonces $\sigma_0 = \sigma$, el estado inicial en A. Como σ_n es un estado de aceptación en A, la cadena α es aceptada por A.

El siguiente teorema resume estos resultados y los de la sección anterior.

Teorema 12.5.4

Un lenguaje L es regular si y sólo si existe un autómata de estado finito que acepte precisamente las cadenas en L.

Demostración Este teorema plantea de nuevo los teoremas 12.4.2, 12.4.11 y 12.5.3.

Ejemplo 12.5.5 ▶

Encuentre un autómata A de estado finito que acepte precisamente las cadenas generadas por la gramática regular G que tiene producciones

$$\sigma \to b\sigma, \qquad \sigma \to aC, \qquad C \to bC, \qquad C \to b.$$

El símbolo inicial es σ, el conjunto de símbolos terminales es $\{a, b\}$, y el conjunto de símbolos no terminales es $\{\sigma, C\}$.

El autómata A' de estado finito no determinístico que acepta $L(G)$ se ilustra en la figura 12.4.2. Un autómata A' de estado finito equivalente se muestra en la figura 12.5.1, y un autómata A de estado finito simplificado equivalente se observa en la figura 12.5.2. El autómata A de estado finito acepta precisamente las cadenas generadas por G. ◀

Esta sección termina con algunas aplicaciones de los métodos y la teoría desarrollada por los autores.

Ejemplo 12.5.6 ▶

Un lenguaje que no es regular

Demuestre que el lenguaje

$$L = \{a^n b^n \mid n = 1, 2, \ldots\}$$

no es regular.

Si L es regular, existe un autómata A de estado finito tal que $\text{Ac}(A) = L$. Suponga que A tiene k estados. A acepta la cadena $\alpha = a^k b^k$. Considere la trayectoria P que representa α. Como hay k estados, algún estado σ se visita de nuevo por la trayectoria que representa a^k. Entonces hay un ciclo C, para el que todas las aristas tienen la etiqueta a, que contiene a σ. Se cambia la trayectoria P para obtener una trayectoria P' como sigue. Al llegar a σ en P, se sigue C. Después de regresar a σ en C, se sigue sobre P hasta el final. Si la longitud de C es j, la trayectoria P' representa la cadena $\alpha' = a^{j+k} b^k$. Como P y P' terminan en el mismo estado σ', y σ' es un estado de aceptación, α' es aceptada por A. Esto es una contradicción, ya que α' no es de la forma $a^n b^n$. Por lo tanto, L no es regular. ◀

Ejemplo 12.5.7 ▶

Sea L el conjunto de cadenas aceptadas por el autómata A de estado finito de la figura 12.5.4. Construya un autómata de estado finito que acepte las cadenas

$$L^R = \{x_n \cdots x_1 \mid x_1 \cdots x_n \in L\}.$$

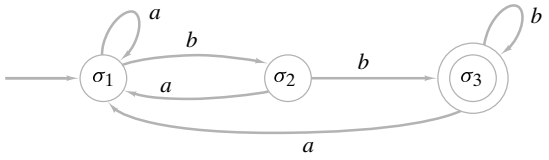


Figura 12.5.4 Autómata de estado finito para el ejemplo 12.5.7 que acepta L.

Se quiere convertir A en un autómata de estado finito que acepte L^R. La cadena $\alpha = x_1 \cdots x_n$ es aceptada por A si y sólo si existe una trayectoria P en A que representa a α que comienza en σ_1 y termina en σ_3. Si se inicia en σ_3 y se sigue P en reversa, se puede terminar en σ_1 y procesar las aristas en el orden $x_n, \ldots, x_1$. Entonces se necesita sólo invertir todas las flechas en la figura 12.5.4 y considerar σ_3 el estado inicial y σ_1 el estado de aceptación (vea la figura 12.5.5). El resultado es un autómata de estado finito no determinístico que acepta a L^R.

Después de encontrar un autómata de estado finito equivalente y eliminar los estados que no se pueden alcanzar, se obtiene el autómata de estado finito equivalente de la figura 12.5.6.

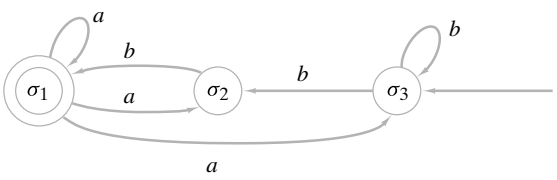


Figura 12.5.5 Autómata de estado finito no determinístico que acepta a L^R.

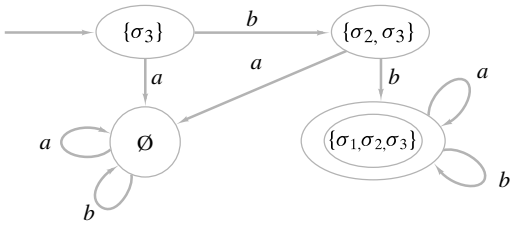


Figura 12.5.6 Autómata de estado finito que acepta a L^R.

◀

Ejemplo 12.5.8 ▶

Sea L el conjunto de cadenas aceptadas por el autómata A de estado finito de la figura 12.5.7. Construya un autómata de estado finito no determinístico que acepte las cadenas

$$L^R = \{x_n \cdots x_1 \mid x_1 \cdots x_n \in L\}.$$

Si A tiene sólo un estado de aceptación, se puede usar el procedimiento del ejemplo 12.5.7 para construir el autómata de estado finito no determinístico deseado. Entonces primero se construye un autómata de estado finito no determinístico equivalente a A con un estado de aceptación. Para esto se introduce un estado adicional σ_5. Después, se arregla que las trayectorias que terminan en σ_3 o σ_4 tengan la opción de terminar en σ_5 (figura 12.5.8). El autómata de estado finito no determinístico deseado se obtiene de la figura 12.5.8 por el método del ejemplo 12.5.7 (figura 12.5.9). Por supuesto, si se desea, es posible construir un autómata de estado finito equivalente.

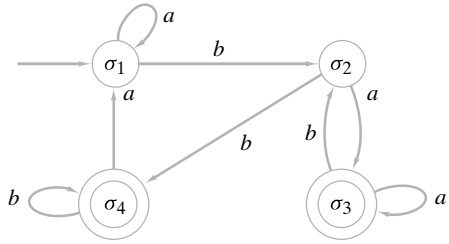


Figura 12.5.7 Autómata de estado finito para el ejemplo 12.5.8 que acepta a L.

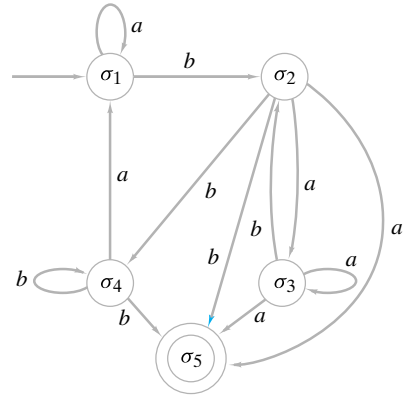


Figura 12.5.8 Autómata de estado finito no determinístico con un estado de aceptación equivalente al autómata de estado finito de la figura 12.5.7.

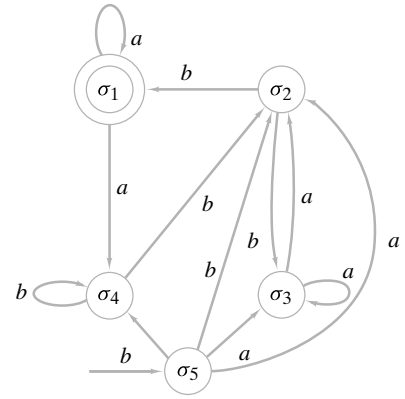


Figura 12.5.9 Autómata de estado finito no determinístico que acepta a L^R.

◀

Sección de ejercicios de repaso

1. Dado un autómata de estado finito no determinístico, ¿cómo se puede construir un autómata de estado finito determinístico equivalente?
2. Dé una condición en términos de autómata de estado finito para que un lenguaje sea regular.

Ejercicios

1. Encuentre el autómata de estado finito equivalente al autómata de estado finito no determinístico en los ejercicios 1 al 10, de la sección 12.4.

En los ejercicios 2 al 6, encuentre el autómata de estado finito que acepte las cadenas generadas por gramáticas regulares.

2. La gramática del ejercicio 1, sección 12.3.
3. La gramática del ejercicio 5, sección 12.3.
4. $$< S > ::= a < A > \mid a < B >$$
$$< A > ::= a < B > \mid b < S > \mid b$$
$$< B > ::= b < S > \mid b$$
con símbolo de inicio $< S >$

5.

$$< S > ::= a < S > | \; a < A > | \; b < C > | \; a$$
$$< A > ::= b < A > | \; a < C >$$
$$< B > ::= a < S > | \; a$$
$$< C > ::= a < B > | \; a < C >$$

con símbolo de inicio $< S >$

6.

$$< S > ::= a < A > | \; a < B >$$
$$< A > ::= b < S > | \; b$$
$$< B > ::= a < B > | \; a < C >$$
$$< C > ::= a < S > | \; b < A > | \; a < C > | \; a$$

con símbolo de inicio $< S >$

7. Encuentre autómatas de estado finito que acepten las cadenas de los ejercicios 21 al 29, de la sección 12.4.

8. Elimine los estados que no se alcanzan del autómata de estado finito de la figura 12.5.3 para encontrar un autómata de estado finito equivalente, más sencillo.

9. Demuestre que el autómata de estado finito no determinístico de la figura 12.5.5 acepta una cadena α sobre $\{a, b\}$ si y sólo si α comienza con bb.

★***10.*** Caracterice las cadenas aceptadas por los autómatas de estado finito no determinísticos de las figuras 12.5.7 y 12.5.9.

En los ejercicios 11 al 21, encuentre un autómata de estado finito no determinístico que acepte el conjunto dado de cadenas. Si S_1 y S_2 son conjuntos de cadenas, sea

$$S_1^+ = \{u_1 u_2 \cdots u_n \mid u_i \in S_1, n \in \{1, 2, \ldots\}\};$$
$$S_1 S_2 = \{uv \mid u \in S_1, v \in S_2\}.$$

11. $\text{Ac}(A)^R$, donde A es el autómata del ejercicio 4, sección 12.2

12. $\text{Ac}(A)^R$, donde A es el autómata del ejercicio 5, sección 12.2

13. $\text{Ac}(A)^R$, donde A es el autómata del ejercicio 6, sección 12.2

14. $\text{Ac}(A)^+$, donde A es el autómata del ejercicio 4, sección 12.2

15. $\text{Ac}(A)^+$, donde A es el autómata del ejercicio 5, sección 12.2

16. $\text{Ac}(A)^+$, donde A es el autómata del ejercicio 6, sección 12.2

17. $\text{Ac}(A)^+$, donde A es el autómata de la figura 12.5.7

18. $\text{Ac}(A_1)\text{Ac}(A_2)$, donde A_1 es el autómata del ejercicio 4, sección 12.2, y A_2 es el autómata del ejercicio 5, sección 12.2

19. $\text{Ac}(A_1)\text{Ac}(A_2)$, donde A_1 es el autómata del ejercicios 5, sección 12.2, y A_2 es el autómata del ejercicio 6, sección 12.2

20. $\text{Ac}(A_1)\text{Ac}(A_1)$, donde A_1 es el autómata del ejercicios 6, sección 12.2

21. $\text{Ac}(A_1)\text{Ac}(A_2)$, donde A_1 es el autómata de la figura 12.5.7 y A_2 es el autómata del ejercicio 5, sección 12.2

22. Encuentre una gramática regular que genere el lenguaje L^R, donde L es el lenguaje generado por la gramática del ejercicio 5, sección 12.3.

23. Encuentre una gramática regular que genere el lenguaje L^+, donde L es el lenguaje generado por la gramática del ejercicio 5, sección 12.3.

24. Sea L_1 (respectivamente, L_2) el lenguaje generado por la gramática del ejercicio 5, sección 12.3 (respectivamente, ejemplo 12.5.5). Encuentre una gramática regular que genere el lenguaje $L_1 L_2$.

★***25.*** Demuestre que el conjunto

$$L = \{x_1 \cdots x_n \mid x_1 \cdots x_n = x_n \cdots x_1\}$$

de cadenas sobre $\{a, b\}$ no es un lenguaje regular.

26. Demuestre que si L_1 y L_2 son lenguajes regulares sobre $\mathcal{I}$ y S es el conjunto de todas las cadenas sobre $\mathcal{I}$, entonces cada uno de $S - L_1$, $L_1 \cup L_2$, $L_1 \cap L_2$, L_1^+ y $L_1 L_2$ es un lenguaje regular.

★***27.*** Demuestre, con un ejemplo, que existen lenguajes libres de contexto L_1 y L_2 tales que $L_1 \cap L_2$ no es libre de contexto.

★***28.*** Pruebe o desapruebe: si L es un lenguaje regular, también lo es

$$\{u^n \mid u \in L, n \in \{1, 2, \ldots\}\}.$$

Notas

Las referencias generales sobre autómatas, gramáticas y lenguajes son [Carroll, Cohen, Davis, Hopcroft, Kelley, McNaughton, Sudkamp y Wood].

El desarrollo sistemático de una geometría fractal fue iniciado por Benoit B. Mandelbrot (vea [Mandelbrot, 1977, 1982]).

Una máquina de estado finito tiene una memoria interna primitiva en el sentido de que recuerda en qué estado se encuentra. Al permitir una memoria externa en la que la máquina puede leer y escribir datos, es posible definir máquinas más poderosas. Otras mejoras se logran si se permite que la máquina lea por rastreo (escaneo) la cadena de entrada en cualquier dirección y que altere la cadena de entrada. Con esto es posible definir las clases de las máquinas que aceptan lenguajes libres de contexto, lenguajes sensibles al contexto y lenguajes generados por gramáticas de estructura de frases.

Las **máquinas Turing** son una clase importante de máquinas. Igual que una máquina de estado finito, una máquina Turing siempre guarda un estado en particular. Se supone que la cadena de entrada a una máquina Turing reside en una cinta que es infinita en ambas direcciones. Una máquina Turing lee un caracter a la vez y después de cada lectura, la máquina se detiene o realiza alguna de las siguientes funciones, ninguna o todas: altera el carácter, se mueve una posición a la derecha o a la izquierda, cambia estados. En particular, se puede cambiar la cadena de entrada. Una máquina Turing T de aceptación una cadena α si, cuando se alimenta α a T, T se detiene en un estado de aceptación. Se puede demostrar que un lenguaje L se genera por una gramática de estructura de frases si y sólo si existe una máquina de Turing que acepte a L.

La importancia real de las máquinas de Turing se debe a la creencia generalizada de que cualquier función que puede calcularse con alguna computadora digital, quizá hipotética, se pue-

de calcular en una máquina de Turing. Esta última aseveración se conoce como **hipótesis de Turing** o **tesis de Church**. La tesis de Church implica que una máquina de Turing es el modelo de abstracción correcto de una computadora digital. Estas ideas también llevan a la siguiente definición formal de algoritmo. Un **algoritmo** es una máquina de Turing que, dada una cadena de entrada, en algún momento se detiene.

Repaso del capítulo

Sección 12.1

1. Retraso unitario de tiempo
2. Sumador en serie
3. Máquina de estado finito
4. Símbolo de entrada
5. Símbolo de salida
6. Estado
7. Función del siguiente estado
8. Función de salida
9. Estado inicial
10. Diagrama de transición
11. Cadenas de entrada y salida para una máquina de estado finito
12. Flip flop *SR*

Sección 12.2

13. Autómata de estado finito
14. Estado de aceptación
15. Cadena aceptada por un autómata de estado finito
16. Autómata de estado finito equivalente

Sección 12.3

17. Lenguaje natural
18. Lenguaje formal
19. Gramática de estructura de frases
20. Símbolo no terminal
21. Símbolo terminal
22. Producción
23. Símbolo de inicio
24. Cadena que se puede derivar directamente
25. Cadena que se puede derivar
26. Derivación
27. Lenguaje generado por una gramática
28. Forma regular de Backus (= forma Backus−Nahur = BNF)
29. Gramática sensible al contexto (= gramática tipo 1)
30. Gramática libre de contexto (= gramática tipo 2)
31. Gramática regular (= gramática tipo 3)
32. Lenguaje sensible al contexto
33. Lenguaje libre de contexto
34. Lenguaje regular
35. Gramática de Lindenmayer interactiva libre de contexto
36. Copo de nieve de von Koch
37. Curvas fractales

Sección 12.4

38. Dado un autómata de estado finito *A*, la forma de construir una gramática regular *G*, tal que el conjunto de cadenas aceptadas por *A* es igual al lenguaje generado por *G* (vea el Teorema 12.4.2)
39. Autómata de estado finito no determinístico
40. Cadena aceptada por una autómata de estado finito no determinístico
41. Autómata de estado finito no determinístico equivalente
42. Dada una gramática regular *G*, la forma de construir un autómata *A* de estado finito no determinístico tal que el lenguaje generado por *G* es igual al conjunto de cadenas aceptadas por *A* (vea el Teorema 12.4.11)

Sección 12.5

43. Dado un autómata de estado finito no determinístico, la forma de construir un autómata de estado finito determinístico equivalente (vea el Teorema 12.5.3)

44. Un lenguaje L es regular si y sólo si existe un autómata de estado finito que acepte las cadenas en L.

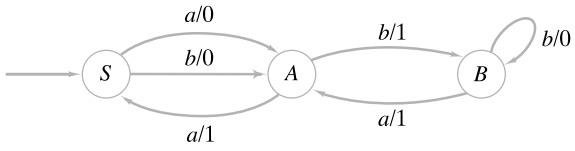
Autoevaluación del capítulo

Sección 12.1

1. Dibuje el diagrama de transición de la máquina de estado finito $(\mathcal{I}, \mathcal{O}, \mathcal{S}, f, g, \sigma_0)$, donde $\mathcal{I} = \{a, b\}$, $\mathcal{O} = \{0, 1\}$ y $\mathcal{S} = \{\sigma_0, \sigma_1\}$.

	f		g	
$\mathcal{S}$ \ $\mathcal{I}$	a	b	a	b
σ_0	σ_1	σ_0	0	1
σ_1	σ_0	σ_1	1	0

2. Encuentre los conjuntos $\mathcal{I}$, $\mathcal{O}$ y $\mathcal{S}$, el estado inicial y la tabla que define las funciones del siguiente estado y de salida para la máquina de estado finito que sigue.



3. Para la máquina de estado finito del ejercicio 1, encuentre la cadena de salida para la cadena de entrada *bbaa*.

4. Diseñe una máquina de estado finito cuya entrada sea una cadena de bits que produce 0 cuando ve 001 y en adelante; de otra manera, produce 1.

Sección 12.2

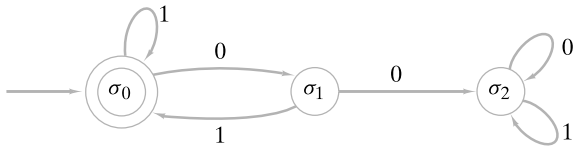
5. Dibuje un diagrama de transición para el autómata de estado finito $(\mathcal{I}, \mathcal{S}, f, \mathcal{A}, \mathcal{S})$, donde $\mathcal{I} = \{0, 1\}$, $\mathcal{S} = \{S, A, B\}$ y $\mathcal{A} = \{A\}$.

	f	
$\mathcal{S}$ \ $\mathcal{I}$	0	1
S	A	S
A	S	B
B	A	S

6. ¿El autómata de estado finito del ejercicio 5 acepta la cadena 11010?

7. Dibuje el diagrama de transición de un autómata de estado finito que acepta el conjunto de cadenas sobre $\{0, 1\}$ que contienen un número par de ceros y un número impar de unos.

8. Defina el conjunto de cadenas aceptadas por el siguiente autómata de estado finito.



Sección 12.3

9. La gramática

$$S \to aSb, \qquad S \to Ab, \qquad A \to aA, \qquad A \to b, \qquad A \to \lambda$$

¿es sensible al contexto, libre de contexto, regular o ninguna de las tres? Dé todas las características pertinentes.

10. Demuestre que la cadena $\alpha = aaaabbbb$ está en el lenguaje generado por la gramática del ejercicio 9 dando una derivación de α.

11. Defina el lenguaje generado por la gramática del ejercicio 9.

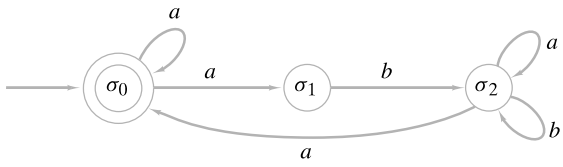
12. Escriba una gramática que genere todas las cadenas no nulas sobre $\{0, 1\}$ que tienen un número igual de ceros y unos.

Sección 12.4

13. Dibuje un diagrama de transición del autómata de estado finito no determinístico $(\mathcal{I}, \mathcal{S}, f, \mathcal{A}, \sigma_0)$, donde $\mathcal{I} = \{a, b\}$, $\mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2\}$ y $\mathcal{A} = \{\sigma_2\}$.

$\mathcal{S}$ \ $\mathcal{I}$	a	b
σ_0	$\{\sigma_0\}$	$\{\sigma_2\}$
σ_1	$\{\sigma_0, \sigma_1\}$	$\emptyset$
σ_2	$\{\sigma_2\}$	$\{\sigma_0, \sigma_1\}$

14. Encuentre los conjuntos $\mathcal{I}$, $\mathcal{S}$, y $\mathcal{A}$, el estado inicial y la tabla que define la función del siguiente estado para el autómata de estado finito no determinístico que sigue



15. ¿El autómata de estado finito no determinístico del ejercicio 14 acepta la cadena *aabaaba*?

16. Diseñe un autómata de estado finito no determinístico que acepte todas las cadenas sobre $\{0, 1\}$ que comienzan con 01 y contienen a 110.

Sección 12.5

17. Encuentre un autómata de estado finito equivalente al autómata de estado finito no determinístico del ejercicio 13.

18. Encuentre un autómata de estado finito equivalente al autómata de estado finito no determinístico del ejercicio 14.

19. Explique cómo construir un autómata de estado finito no determinístico que acepta el lenguaje

$$L_1L_2 = \{\alpha\beta \mid \alpha \in L_1, \beta \in L_2\},$$

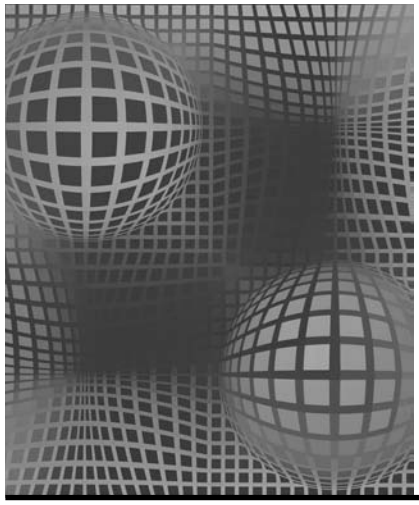
dado el autómata de estado finito que acepta los lenguajes regulares L_1 y L_2.

20. Pruebe que cualquier lenguaje regular que no contiene la cadena nula es aceptado por un autómata de estado finito no determinístico con exactamente un estado aceptante. Dé un ejemplo para mostrar que esta afirmación es falsa para lenguajes regulares arbitrarios (es decir, si se permite que la cadena nula pertenezca al lenguaje regular).

Ejercicios para computadora

1. Escriba un programa que simule una máquina de estado finito arbitraria. Al inicio el programa debe recibir como entrada la función del siguiente estado, la función de salida y el estado inicial. Después, el programa debe aceptar cadenas, simular la acción de la máquina de estado finito y producir la cadena generada por la máquina de estado finito.

2. Escriba un programa que simule un autómata de estado finito arbitrario. Al inicio, el programa debe recibir como entrada la función del siguiente estado, el conjunto de estados de aceptación y el estado inicial. Después, el programa debe aceptar cadenas, simular la acción del autómata de estado finito e imprimir mensajes que indiquen si la cadena fue aceptada.
3. Escriba un programa para dibujar un fractal dada la gramática de Lindenmayer interactiva libre de contexto que lo genera (vea la sección 12.3).
4. Haga un informe del algoritmo de Knuth-Morris-Pratt (vea [Johnsonbaugh]) que determine si una cadena contiene una subcadena específica. Este algoritmo utiliza un autómata de estado finito.



Capítulo 13

GEOMETRÍA PARA CÁLCULO

Yo demostré, sin lugar a dudas y con lógica geométrica, que sí existía una llave.

DE THE CAINE MUTINY

La **geometría para cálculo** se refiere al diseño y análisis de los algoritmos para resolver problemas geométricos. Los algoritmos geométricos eficientes son útiles en campos como gráficas por computadora, estadística, procesamiento de imágenes y diseño de la integración a gran escala (VLSI, siglas en inglés para *very large scale integration*). En este capítulo se presenta una introducción a este fascinante tema.

El problema del par más cercano proporciona un ejemplo de un problema de geometría para cálculo: dados n puntos en el plano, hay que encontrar el par más cercano. Además del problema del par más cercano, se considera el problema de encontrar el casco convexo.

13.1 ➜ Problema del par más cercano

El **problema del par más cercano** tiene un enunciado sencillo: dados n puntos en el plano, encuentre un par más cercano (vea la figura 13.1.1). (Se habla de *un* par más cercano porque es posible que varios pares logren la misma distancia mínima). La medida de la distancia aquí, es la distancia euclidiana normal.

WWW

Una manera de resolver este problema es enumerar la distancia entre cada par y elegir la mínima distancia de la lista. Como hay $C(n, 2) = n(n - 1)/2 = \Theta(n^2)$ pares, el tiempo de este algoritmo de "listar todos" es $\Theta(n^2)$. Se puede lograr algo mejor; se dará un algoritmo de "divide y vencerás" para el par más cercano cuyo tiempo en el peor caso es $\Theta(n \lg n)$. Primero se analiza el algoritmo y luego se da una descripción más precisa usando un seudocódigo.

Nuestro algoritmo comienza por encontrar una línea vertical l que divide los puntos en dos partes casi iguales (figura 13.1.1). [Si n es impar, se dividen los puntos en dos partes, cada una con $n/2$ puntos. Si n es impar, se dividen los puntos en dos partes, una con $(n + 1)/2$ puntos y la otra con $(n - 1)/2$ puntos].

Después se resuelve el problema de manera recursiva para cada parte. Sea δ_L la distancia entre un par más cercano en la parte izquierda; sea δ_R la distancia entre un par más cercano en la parte derecha; se hace

$$\delta = \text{mín}\{\delta_L, \delta_R\}.$$

Desafortunadamente, δ quizá no sea la distancia entre un par más cercano en el conjunto original de puntos porque un par de puntos, uno de la parte izquierda y otro de la derecha,

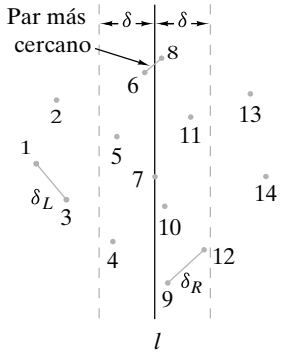


Figura 13.1.1 Con n puntos en el plano, el problema es encontrar un par más cercano. Para este conjunto, el par más cercano es 6 y 8. La línea l divide los puntos en dos partes aproximadamente iguales. El par más cercano en la mitad de la izquierda es 1 y 3, con una distancia de δ_L. El par más cercano en la mitad derecha está separado δ_R. Cualquier par (como 6 y 8) más cercano que $\delta = \text{mín}\{\delta_L, \delta_R\}$ debe estar en la franja vertical con 2δ de ancho centrada en l.

podría ser más cercano que δ (vea la figura 13.1.1). Entonces deben considerarse las distancias entre puntos en los lados opuestos de la línea l.

Primero se observa si la distancia entre un par de puntos es menor que δ, los puntos deben estar en una franja vertical con 2δ de ancho centrada en l (figura 13.1.1). (Cualquier punto fuera de esta franja está al menos a una distancia δ de todos los puntos en el otro lado de l). Entonces se puede restringir la búsqueda de un par más cercano que δ a los puntos en esta franja.

Si hay n puntos en la franja y se verifican *todos* los pares en ella, el tiempo en el peor caso para procesar los puntos en la franja es $\Theta(n^2)$. En este caso, el tiempo en el peor caso de nuestro algoritmo será $\Omega(n^2)$, que es al menos tan malo como la búsqueda exhaustiva; entonces, debe evitarse verificar todos los pares de la franja.

Se colocan los puntos en la franja en orden no decreciente respecto de sus coordenadas y. Después se examinan los puntos en este orden. Cuando se examina un punto p en la franja, cualquier punto q que sigue a p cuya distancia a p es menor que δ, debe estar estrictamente dentro de, o en la base del rectángulo con altura δ cuya base contiene a p y cuyos lados verticales están a una distancia δ de l (figura 13.1.2). (No es necesario calcular la distancia entre p y los puntos por debajo de p. Estas distancias ya se consideraron puesto que se están examinando los puntos en orden no decreciente de sus coordenadas y). Se demostrará que este rectángulo contiene cuando mucho ocho puntos, incluyendo p, de manera que si se calculan las distancias entre p y los siguientes siete puntos en la franja, se puede tener la seguridad de que se calculará la distancia entre p y todos los puntos en el rectángulo. Por supuesto, si menos de siete puntos siguen a p en la lista, se calculan las distancias entre p y todos los puntos restantes. Al restringir la búsqueda en la franja de esta manera, el tiempo para procesar los puntos en la franja es $O(n)$. (Como hay cuando mucho n puntos en la franja, el tiempo necesario para procesar los puntos en ella es a lo sumo de $7n$).

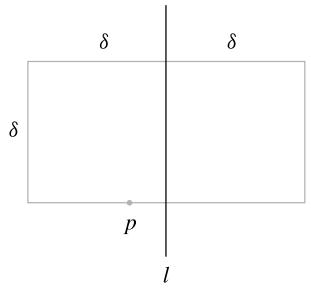


Figura 13.1.2 Cualquier punto q que sigue a p cuya distancia a p es menor que δ debe estar dentro del rectángulo.

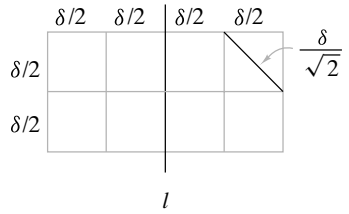


Figura 13.1.3 El rectángulo grande contiene cuando mucho ocho puntos porque cada cuadrado contiene a lo sumo un punto.

Se demuestra que el rectángulo de la figura 13.1.2 contiene cuando mucho ocho puntos. La figura 13.1.3 muestra el rectángulo de la figura 13.1.2 dividido en ocho cuadros iguales. Observe que la longitud de una diagonal de un cuadrado es

$$\left(\left(\frac{\delta}{2}\right)^2 + \left(\frac{\delta}{2}\right)^2\right)^{1/2} = \frac{\delta}{\sqrt{2}} < \delta;$$

así que cada cuadrado contiene, cuando mucho, un punto. Por lo tanto, el rectángulo de $2\delta \times \delta$ contiene cuando mucho ocho puntos.

Ejemplo 13.1.1 ▶

Se demuestra la manera en que el algoritmo del par más cercano encuentra dicho par para la entrada de la figura 13.1.1.

El algoritmo comienza por encontrar una línea vertical l que divida los puntos en dos partes iguales,

$$S_1 = \{1, 2, 3, 4, 5, 6, 7\}, \qquad S_2 = \{8, 9, 10, 11, 12, 13, 14\}.$$

Para esta entrada existen muchas opciones posibles para la línea divisoria. Sucede que la línea específica elegida aquí pasa por el punto 7.

Ahora se resuelve el problema de manera recursiva para S_1 y S_2. El par más cercano de los puntos en S_2 es 1 y 3. Sea δ_L la distancia entre los puntos 1 y 3. El par más cercano en S_2 es 9 y 12. Sea δ_R la distancia entre 9 y 12. Se establece que

$$\delta = \text{mín}\{\delta_L, \delta_R\} = \delta_L.$$

Ahora se ordenan los puntos en la franja vertical con ancho 2δ centrada en l, en orden no decreciente de sus coordenadas y:

$$9, \quad 12, \quad 4, \quad 10, \quad 7, \quad 5, \quad 11, \quad 6, \quad 8.$$

Después se examinan los puntos en este orden. Se calculan las distancias entre cada punto y los siete que le siguen, o entre cada punto y los restantes si le siguen menos de siete.

Primero se calculan las distancias de 9 a cada uno de 12, 4, 10, 7, 5, 11 y 6. Como cada una de ellas excede a δ, en este punto no se ha encontrado un par más cercano.

Después se calculan las distancias de 12 a cada uno de 4, 10, 7, 5, 11, 6 y 8. Como cada distancia excede a δ, en este punto todavía no se encuentra un par más cercano.

Luego se calculan las distancias de 4 a cada uno de 10, 7, 5, 11, 6 y 8. Como todas estas distancias exceden a δ, todavía no se encuentra un par más cercano en este punto.

Ahora se calculan las distancias de 10 a cada uno de 7, 5, 11, 6 y 8. Como la distancia entre los puntos 10 y 7 es menor que δ, se ha descubierto un par más cercano. Se actualiza δ a la distancia entre 10 y 7.

Después se calcula la distancia entre 7 y cada uno de 5, 11, 6 y 8. Como cada una de estas distancias excede δ, no se ha encontrado un par más cercano.

Después se calcula la distancia entre 5 y cada uno de 11, 6 y 8. Como cada una de estas distancias excede a δ, no se ha encontrado un par más cercano.

Luego se calculan las distancias de 11 a cada uno de 6 y 8. Como estas distancias exceden a δ, no se ha encontrado un par más cercano.

Ahora se calcula la distancia entre 6 y 8. Como esta distancia es menor que δ, se descubrió un par más cercano. Se actualiza δ con la distancia entre los puntos 6 y 8. Como no hay más puntos que considerar en la franja, el algoritmo termina. El par más cercano es 6 y 8 y la distancia entre ellos es δ. ◀

Antes de dar un enunciado formal del algoritmo del par más cercano, existen varios puntos técnicos que resolver.

Para terminar el proceso recursivo, se verifica el número de puntos en la entrada, y si hay tres puntos o menos se encuentra el par más cercano directamente. Dividir la entrada y resolver en forma recursiva sólo si hay cuatro puntos o más asegura que cada parte contenga al menos un par de puntos y, por lo tanto, que hay un par más cercano en cada parte.

Antes de invocar el procedimiento recursivo, se ordena el conjunto completo de puntos según su coordenada x. Esto facilita dividir los puntos en dos partes casi iguales.

Se usa el merge sort (vea la sección 7.3) para ordenar por la coordenada y. Sin embargo, en lugar de ordenar cada vez que se examinan los puntos en la franja vertical, se supone, como en el merge sort, que cada mitad está ordenada por la coordenada y. Después simplemente se fusionan las dos mitades para ordenar todos los puntos por la coordenada y.

Ahora se puede establecer, de manera formal, el algoritmo del par más cercano. Para simplificar la descripción, nuestra versión produce sólo la distancia entre un par más cercano, no un par más cercano. Se deja este detalle como ejercicio (vea el ejercicio 5).

Algoritmo 13.1.2

Encontrar la distancia entre un par más cercano de puntos

Entrada: $p_1, \ldots, p_n$ ($n \geq 2$ puntos en el plano)
Salida: d, distancia entre un par más cercano de puntos

```
par_más_cercano( p, n) {
    ordena p_1, . . . , p_n por la coordenada x
    return rec_par_cerc( p, 1, n)
}
```

```
rec_par_cerc( p, i, j) {
    // la entrada es la sucesión p_i, . . . , p_j de puntos en el plano
    // ordenados por la coordenada x

    // al terminar rec_par–cerc, la sucesión está ordenada por
    // la coordenada y

    // rec_par_cerc regresa la distancia entre un par más
    // cercano en la entrada

    // denota la coordenada x del punto p por p.x

    // caso trivial (3 puntos o menos)
    if (j − i < 3) {
        ordena p_i, . . . , p_j por la coordenada y
        encuentra la distancia δ entre un par más cercano
        return δ
    }
    // divide
    k = ⌊(i + j)/2⌋
    l = p_k.x
    δ_L = rec_par_cerc( p, i, k)
    δ_R = rec_par_cerc( p, k + 1, j)
    δ = mín{δ_L, δ_R}
    // p_i, . . . , p_k está ordenado según la coordenada y
    // p_{k+1}, . . . , p_j está ordenado según la coordenada y
    se fusionan p_i, . . . , p_k y p_{k+1}, . . . , p_j por la coordenada y
    // suponga que el resultado de la fusión está ordenado como p_i, . . . , p_j

    // ahora p_i, . . . , p_j está ordenado según la coordenada y

    // se almacenan los puntos en la granja vertical en v
    t = 0
    for k = i to j
        if( p_k.x > l − δ ∧ p_k.x < l + δ) {
            t = t + 1
            v_t = p_k
        }
    // los puntos en la franja son v_1, . . . , v_t
    // se buscan pares más cercanos en la franja
    // se compara cada uno con los siguientes siete puntos
    for k = 1 to t − 1
        for s = k + 1 to mín{t, k + 7}
            δ = mín{δ, dist(v_k, v_s)}
    return δ
}
```

Se demuestra que el tiempo en el peor caso para el algoritmo del par más cercano es $\Theta(n \lg n)$. El procedimiento *par_más_cercano* comienza por ordenar los puntos según la coordenada x. Si se usa un ordenamiento óptimo (como merge sort), el tiempo en el peor caso será $\Theta(n \lg n)$. Después, *par_más_cercano* invoca *rec_par_cerc*. Se denota como a_n el tiempo en el peor caso de *rec_par_cerc* para la entrada de tamaño n. Si $n > 3$, *rec_par_cerc* primero se llama a sí mismo con una entrada de tamaño $\lfloor n/2 \rfloor$ y $\lfloor (n + 1)/2 \rfloor$. Fusionar los puntos, extraer los puntos de la franja y verificar las distancias en la franja son acciones que llevan, cada una, un tiempo de $O(n)$. Entonces se obtiene la relación de recurrencia

$$a_n \leq a_{\lfloor n/2 \rfloor} + a_{\lfloor (n+1)/2 \rfloor} + cn, \qquad n > 3.$$

Ésta es la misma recurrencia que satisface el merge sort, de manera que se concluye que *rec_par_cerc* tiene el mismo tiempo en el peor caso $O(n \lg n)$ que el merge sort. Como el tiempo en el peor caso del ordenamiento de los puntos según la coordenada x es $\Theta(n \lg n)$ y el tiempo en el peor caso de *rec_par_cerc* es $O(n \lg n)$, el tiempo en el peor caso de *par_más_cercano* es $\Theta(n \lg n)$.

Haciendo suposiciones razonables acerca de qué tipos de cálculos se permiten y usando un modelo de árbol de decisiones junto con métodos avanzados, Preparata (vea [Preparata, 1985: Teorema 5.2, p. 188]) demuestra que cualquier algoritmo que resuelve el problema del par más cercano es $\Omega(n \lg n)$; entonces el algoritmo 13.1.2 es óptimo de manera asintótica.

Se puede demostrar (ejercicio 10) que existen cuando mucho seis puntos en el rectángulo de la figura 13.1.2 cuando se incluye la base y los otros lados se excluyen. Este resultado es el mejor posible ya que se pueden colocar seis puntos en el rectángulo (ejercicio 8). Considerando los lugares posibles para los puntos en el rectángulo, D. Lerner y R. Johnsonbaugh demostraron que basta con comparar cada punto en la franja con los siguientes tres puntos (en lugar de los siguientes siete). Este resultado es el mejor posible puesto que verificar los siguientes dos puntos no conduce a un algoritmo correcto (ejercicio 7).

Sección de ejercicios de repaso

1. ¿Qué es geometría para el cálculo?

2. Enuncie el problema del par más cercano.

3. Describa un algoritmo de fuerza bruta para el par más cercano.

4. Describa el algoritmo de divide y vencerás para el par más cercano.

5. Compare los tiempos del peor caso de los algoritmos de fuerza bruta y divide y vencerás para el par más cercano.

Ejercicios

1. Describa cómo el algoritmo del par más cercano encuentra el par más cercano entre los puntos si la entrada es (8, 4), (3, 11), (12, 10), (5, 4), (1, 2), (17, 10), (8, 7), (8, 9), (11, 3), (1, 5), (11, 7), (5, 9), (1, 9), (7, 6), (3, 7), (14, 7),

2. ¿Qué se puede concluir de la entrada al algoritmo del par más cercano cuando la salida es cero para las distancias entre un par más cercano?

3. Dé un ejemplo de una entrada para la que el algoritmo del par más cercano coloca algunos puntos que están sobre la línea divisoria l en la mitad izquierda y otros en la derecha.

4. Explique por qué, en algunos casos, al dividir un conjunto de puntos con una línea vertical en dos partes casi iguales, es necesario que la línea contenga algunos de los puntos.

5. Escriba un algoritmo del par más cercano que encuentre un par más cercano al igual que la distancia entre el par de puntos.

6. Escriba un algoritmo que encuentre la distancia entre un par más cercano de puntos en una línea (recta).

7. Dé un ejemplo de la entrada para la cual comparar cada punto en la franja con los siguientes dos puntos produce una salida incorrecta.

8. Dé un ejemplo para demostrar que es posible colocar seis puntos en el rectángulo de la figura 13.1.2 cuando se incluye la base y se excluyen los otros lados.

9. Cuando se calculan las distancias entre un punto p en la franja y los puntos que le siguen, ¿se puede dejar de calcular las distancias desde p si se encuentra un punto q tal que la distancia entre p y q excede a δ? Explique su respuesta.

★***10.*** Demuestre que hay cuando mucho seis puntos en el rectángulo de la figura 13.1.2 cuando se incluye la base y los otros lados se excluyen.

11. Escriba un algoritmo $\Theta(n \lg n)$ que encuentre la distancia δ entre un par más cercano y que, si $\delta > 0$, también encuentre *todos* los pares que están a una distancia δ.

12. Escriba un algoritmo $\Theta(n \lg n)$ que encuentre la distancia δ entre un par más cercano y *todos* los pares que están a menos de 2δ.

13. Escriba un algoritmo $\Theta(n \lg n)$ que encuentre la distancia δ entre un par más cercano y todos los pares a menos de 2δ, donde cada par que está a menos de 2δ se lista exactamente una vez.

14. Explique por qué el siguiente algoritmo no encuentra la distancia δ entre un par más cercano y todos los pares a menos de 2δ.

```
ejercicio14(p, n) {
    δ = par_más_cercano( p, n)
    for i = 1 to n − 1
        for j = i + 1 to mín{n, i + 31}
            if (dist(p_i, p_j) < 2*δ)
                println(i + " " + j)
}
```

15. Escriba un algoritmo que combine encontrar los puntos en la franja y buscar en la franja. De esta manera, el tamaño de la franja cambia en forma dinámica de modo que, en general, esta versión verifica menos puntos en la franja que el algoritmo 13.1.2.

16. Suponga que la distancia entre un par más cercano en un conjunto de n puntos en el plano es $\delta > 0$. Demuestre que el número de pares que están exactamente a una distancia δ es $O(n)$.

13.2 ➜ Algoritmo para calcular el casco convexo

Un problema fundamental en geometría para cálculo es el de calcular los puntos que "acotan" o del contorno, de manera formal, el **casco convexo**, de un conjunto finito de puntos en el plano. (Vea la figura 13.2.1, donde los puntos que forman el casco convexo tienen un círculo). El casco convexo tiene aplicaciones en muchas áreas, que incluyen estadística, gráficas por computadora y procesamiento de imágenes. Por ejemplo, en estadística, los puntos de un conjunto de datos que determinan su casco convexo pueden ser puntos extremos que en realidad no son representativos de los datos y se pueden descartar. En esta sección se presenta el algoritmo de Graham para calcular el casco convexo. En toda la sección, "conjunto de puntos" significa un "conjunto de puntos *distintos*". Comenzaremos con las definiciones.

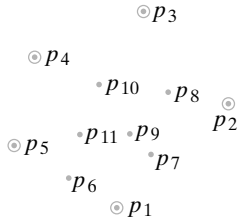


Figura 13.2.1 Casco convexo de los puntos $p_1, \ldots, p_{11}$ es p_1, p_2, p_3, p_4, p_5.

Definición 13.2.1 ▶

Dado un conjunto finito S de puntos en el plano, un punto p en S es un *punto del casco* si existe una línea (recta) L que pasa por p tal que todos los puntos en S excepto p están de un lado de L (y a excepción de p, ninguno está sobre L). ◀

Ejemplo 13.2.2 ▶

En la figura 13.2.2, p_1 es un punto en el casco ya que podemos encontrar una línea L_1 que pasa por p_1 tal que todos los otros puntos están estrictamente en un lado de L_1. El punto p_8 no es un punto del casco porque toda línea L que pasa por p_8 tiene puntos en ambos lados. El punto p_6 tampoco está en el casco. Como se ve en la figura 13.2.2, es posible colocar una línea L_2 a través de p_6 tal que un lado de L_2 no contiene puntos, pero L_2 no cumple con las condiciones de la definición 13.2.1, puesto que contiene puntos distintos de p_6.

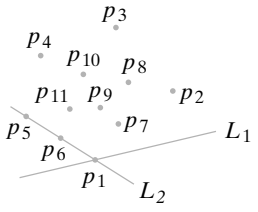


Figura 13.2.2 Los puntos del casco son p_1, p_2, p_3, p_4, p_5. El resto de los puntos no son puntos del casco.

◀

El casco convexo de un conjunto finito S de puntos en el plano consiste en los puntos del casco enumerados en orden al recorrer la frontera de S. En la figura 13.2.2, el casco convexo es la sucesión de puntos p_1, p_2, p_3, p_4, p_5. La siguiente definición precisa este concepto de ordenar los puntos del casco.

Definición 13.2.3 ▶

El *casco convexo* de un conjunto finito S de puntos en el plano es la sucesión $p_1, p_2, \ldots, p_n$ de puntos en el casco de S listados en el siguiente orden. El punto p_1 es el punto con la

coordenada y mínima. Si varios puntos tienen la misma coordenada y mínima, p_1 es el que tiene la coordenada x mínima. (Observe que p_1 es un punto en el casco). Para $i \geq 2$, sea a_i el ángulo desde la horizontal al segmento de recta p_1, p_i (vea la figura 13.2.3). Los puntos $p_1, p_2, \ldots, p_n$ están ordenados de manera que $\alpha_2, \alpha_3, \ldots, \alpha_n$ es una sucesión creciente.

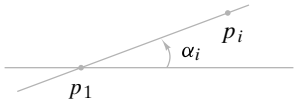


Figura 13.2.3 α_i es el ángulo de la horizontal al segmento de recta p_1, p_i.

Ejemplo 13.2.4 ▶

En la figura 13.2.4, el punto p_1 tiene la coordenada y mínima, de manera que es el primer punto en la lista del casco convexo. Como se muestra, los ángulos desde la horizontal a los segmentos de recta p_1, p_2; p_1, p_3; p_1, p_4; p_1, p_5 aumentan. De esta forma, el casco convexo del conjunto de puntos en la figura 13.2.4 es, p_1, p_2, p_3, p_4, p_5.

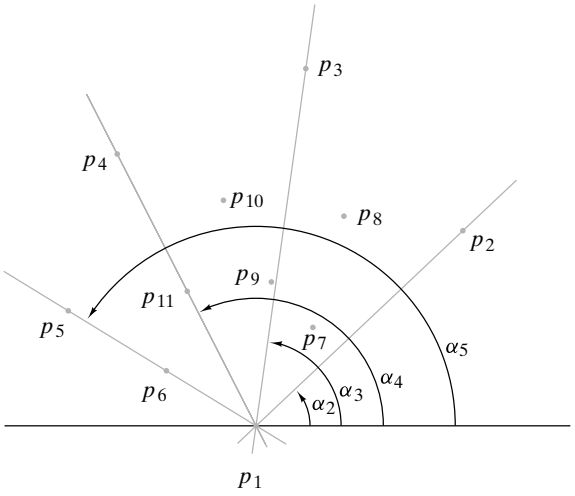


Figura 13.2.4 El casco convexo es p_1, p_2, p_3, p_4, p_5 porque éstos son puntos del casco y los ángulos correspondientes α_2, α_3, α_4, α_5 aumentan, donde α_i es el ángulo de la horizontal al segmento de recta p_1, p_i.

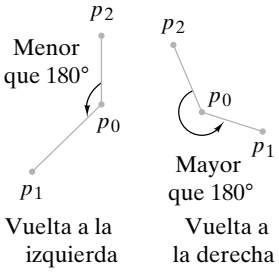


Figura 13.2.5 El primer giro es a la izquierda porque el ángulo de p_0, p_2 a p_0, p_1 es menor que 180°. La segunda vuelta es a la derecha porque el ángulo de p_0, p_2 a p_0, p_1 es mayor que 180°.

La definición 13.2.3 sugiere un algoritmo para calcular el casco convexo de un conjunto finito S en el plano. Primero se encuentra el punto p_1 con la coordenada y mínima. Si varios puntos tienen la misma coordenada y, se elige el que tiene la coordenada x mínima. Después, se ordenan *todos* los puntos p en S según el ángulo desde la horizontal hasta el segmento de recta p_1, p. Por último, se examinan los puntos en orden y se descartan los que no están en el casco convexo. El resultado será el casco convexo. Ésta es la estrategia del algoritmo de Graham. Para que esta idea se convierta en un algoritmo, deben analizarse dos aspectos importantes. Debe describirse una manera de comparar ángulos, y debe desarrollarse un método para probar si los puntos están en el casco convexo. Primero se estudiará el problema de comparar ángulos.

Suponga que se visitan los puntos diferentes p_1, p_0, p_2 en el plano, en este orden. Si después de salir de p_0 nos movemos a la izquierda, se dice que los puntos p_1, p_0, p_2 *hacen un giro a la izquierda* (vea la figura 13.2.5). De manera más precisa, los puntos p_1, p_0, p_2 hacen un giro a la izquierda si el ángulo del segmento de recta p_0, p_2 al segmento de recta p_0, p_1, medido en sentido contrario a las manecillas del reloj, es menor que 180°. De ma-

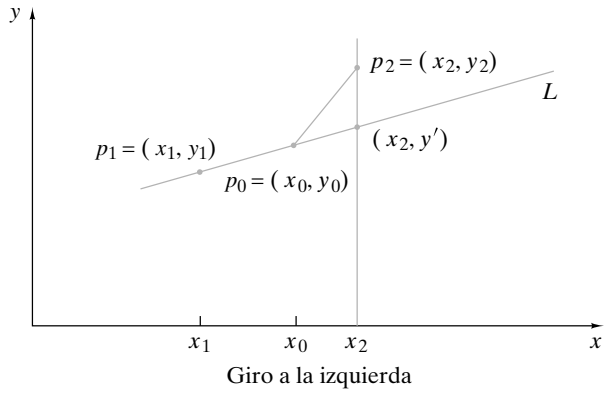


Figura 13.2.6 Se prueba si los puntos p_1, p_0, p_2 hacen un giro a la izquierda o a la derecha. La figura supone que $x_1 < x_0$. L es la línea que pasa por p_0 y p_1. Como se muestra, en este caso un giro a la izquierda ocurre precisamente cuando p_2 está arriba de L.

nera similar, si después de salir de p_0 nos movemos a la derecha, se dice que los puntos p_1, p_0, p_2 *hacen un giro a la derecha* (vea la figura 13.2.5); es decir, los puntos p_1, p_0, p_2 giran a la derecha si el ángulo del segmento de recta p_0, p_2 al segmento de recta p_0, p_1, medido en sentido contrario a las manecillas del reloj, es mayor que 180°.

Se pueden usar métodos de geometría analítica para derivar una prueba a fin de determinar si los puntos p_1, p_0, p_2 giran a la izquierda o a la derecha. Sean (x_i, y_i) las coordenadas del punto p_i, $i = 0, 1, 2$ (figura 13.2.6). Suponga primero que $x_1 < x_0$. La ecuación de la línea L que pasa por p_0 y p_1 es

$$y = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0).$$

Ahora p_1, p_0, p_2 giran a la izquierda precisamente cuando p_2 está arriba de L o cuando

$$y_2 > y' = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_0).$$

Esta desigualdad se escribe como

$$y_2 - y_0 > \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_0).$$

Al multiplicar por $x_1 - x_0$, que es negativo, y mover todos los términos a un lado de la desigualdad se tiene

$$(y_2 - y_0)(x_1 - x_0) - (y_1 - y_0)(x_2 - x_0) < 0.$$

Si se define el **producto cruz** de los puntos p_1, p_0, p_2 como

$$\text{cruz}(p_1, p_0, p_2) = (y_2 - y_0)(x_1 - x_0) - (y_1 - y_0)(x_2 - x_0),,$$

se ha demostrado lo siguiente:

Si los puntos p_1, p_0, p_2 giran a la izquierda, entonces cruz$(p_1, p_0, p_2) < 0$.

De manera similar, se puede demostrar que

Si los puntos p_1, p_0, p_2 giran a la derecha, entonces cruz$(p_1, p_0, p_2) > 0$,

y

Si los puntos p_1, p_0, p_2 son colineales, entonces cruz$(p_1, p_0, p_2) = 0$.

También es posible demostrar el inverso de estas afirmaciones. Por ejemplo, para probar que si cruz$(p_1, p_0, p_2) < 0$, entonces los puntos p_1, p_0, p_2 hacen un giro a la izquierda, se argumenta lo siguiente. Suponga que cruz$(p_1, p_0, p_2) < 0$. Como los puntos no giran a la derecha [si lo hicieran, cruz(p_1, p_0, p_2) sería positivo] y no son colineales [si lo fueran, cruz(p_1, p_0, p_2) sería cero], deben hacer un giro a la izquierda. Entonces,

Los puntos p_1, p_0, p_2 giran a la izquierda si y sólo si cruz$(p_1, p_0, p_2) < 0$.
Los puntos p_1, p_0, p_2 giran a la derecha si y sólo si cruz$(p_1, p_0, p_2) > 0$.
Los puntos p_1, p_0, p_2 son colineales si y sólo si cruz$(p_1, p_0, p_2) = 0$.

(13.2.1)

Se demostró (13.2.1) suponiendo que $x_1 < x_0$. Si $x_1 = x_0$ o $x_1 > x_0$, la conclusión (13.2.1) sigue siendo válida (vea los ejercicios 2 y 3). Estas conclusiones se resumen como un teorema.

Teorema 13.2.5

Si p_1, p_0, p_2 son puntos distintos en el plano,

a) p_1, p_0, p_2 dan un giro a la izquierda si y sólo si cruz$(p_1, p_0, p_2) < 0$.
b) p_1, p_0, p_2 dan un giro a la derecha si y sólo si cruz$(p_1, p_0, p_2) > 0$.
c) p_1, p_0, p_2 son colineales si y sólo si cruz$(p_1, p_0, p_2) = 0$.

Demostración La demostración precede al enunciado del teorema.

El algoritmo de Graham comienza por encontrar el punto p_1 con la menor coordenada y. Si varios puntos tienen la misma coordenada y mínima, el algoritmo elige la que tiene la menor coordenada x. Después el algoritmo ordena *todos* los puntos p en S según el ángulo de la horizontal al segmento de recta p_1, p. Por último, el algoritmo examina los puntos en orden y descarta los que no están en el casco convexo.

Se puede usar el producto cruz para comparar puntos $p \neq q$ en el ordenamiento. Para comparar los puntos p y q, se calcula el producto cruz(p_1, p, q). Si cruz$(p_1, p, q) < 0$, entonces p, p_1, q giran a la izquierda. Respecto a los ángulos con la horizontal, esto significa que $p > q$ (vea la figura 13.2.7). Si cruz$(p_1, p, q) > 0$, entonces $p < q$. Si cruz$(p_1, p, q) = 0$, entonces p, p_1, q son colineales. En el último caso, se define $p > q$ si p está más lejos de p_1 que q; y se define $p < q$ si q está más lejos de p_1 que p.

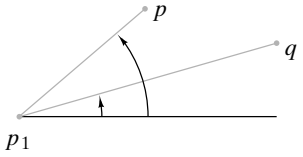


Figura 13.2.7 El caso $p > q$ ocurre cuando p, p_1, q hacen un giro a la izquierda.

También se puede usar el producto cruz para determinar si debe descartarse un punto en el casco convexo. Al examinar los puntos en orden, se conservarán los puntos que estarían en el casco convexo si no hubiera más puntos que examinar. Después, se examina el siguiente punto p. Por ejemplo, en la figura 13.2.8 suponga que se han conservado $p_1, \ldots, p_5$ y el siguiente punto que se examina es p. Como p_4, p_5, p hacen un giro a la izquierda, se conserva p_5. Después se examina el punto que sigue a p.

En otro ejemplo, suponga que se han conservado $p_1, \ldots, p_5$ (vea la figura 13.2.9). Esta vez, como p_4, p_5, p hacen un giro a la derecha, se descarta p_5. Después se regresa a examinar p_3, p_4, p. Como estos puntos también dan un giro a la derecha, se descarta p_4. Des-

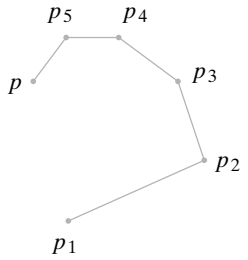


Figura 13.2.8 Una situación en el algoritmo del casco convexo cuando se examina el punto p. Antes de examinar p, el casco convexo de los puntos examinados hasta ahora es p_1, p_2, p_3, p_4, p_5. Como p_4, p_5, p dan un giro a la izquierda, p_5 se queda en el casco convexo de los puntos examinados hasta ahora, es decir, se conserva. El casco convexo actual es p_1, p_2, p_3, p_4, p_5, p. El algoritmo continúa examinando el punto que sigue a p.

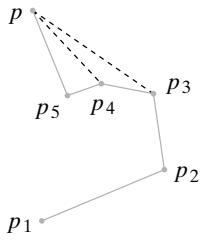


Figura 13.2.9 Una situación en el algoritmo del casco convexo cuando se examina el punto p. Antes de examinar p, el casco convexo de los puntos examinados hasta ahora es p_1, p_2, p_3, p_4, p_5. Como p_4, p_5, p dan un giro a la derecha, p_5 se descarta. Esto deja los puntos p_3, p_4, p, que también dan un giro a la derecha; entonces, también se descarta p_4. Esto deja los puntos p_2, p_3, p, que dan un giro a la izquierda; entonces, p_3 se conserva. El casco convexo actual es p_1, p_2, p_3, p. El algoritmo continúa examinando el punto que sigue a p.

pués, se regresa a examinar p_2, p_3, p. Como estos puntos hacen un giro a la izquierda, se conserva p_3. El proceso continúa examinando el punto que sigue a p. El seudocódigo para el algoritmo de Graham está dado como el algoritmo 13.2.6.

Algoritmo 13.2.6

Algoritmo de Graham para calcular el caso convexo

Este algoritmo calcula el casco convexo de los puntos $p_1, \ldots, p_n$ en el plano. Las coordenadas x y y del punto p se denotan por $p.x$ y $p.y$, respectivamente.

WWW

Entrada: $p_1, \ldots, p_n$ y n
Salida: $p_1, \ldots, p_k$ (el casco convexo de $p_1, \ldots, p_n$) y k

```
barrido_graham(p, n, k) {
    // caso trivial
    if (n == 1) {
        k = 1
        return
    }
    // encuentra el punto con coordenada y mínima
    mín = 1
    for i = 2 to n
        if(p_i · y < p_mín · y)
            mín = i
    // entre todos estos puntos encuentre el que tiene
    // coordenada x mínima
    for i = 1 to n
        if (p_i · y == p_mín · y ∧ p_i · x < p_mím · x)
            mín = i
    cambia(p_1, p_mín)
    // ordena según el ángulo de la horizontal a p_1, p_i
    ordena p_2, . . . , p_n
```

```
    // p_0 es un punto adicional para prevenir que el algoritmo
    // regrese sin parar
    p_0 = p_n
    // se descartan los puntos que no están en el casco convexo
    k = 2
    for i = 3 to n {
        while (p_{k-1}, p_k, p_i no giran a la derecha)
            // se descarta p_k
            k = k − 1
        k = k + 1
        cambia(p_i, p_k)
    }
}
```

Ejemplo 13.2.7 ▶ La figura 13.2.10 muestra el algoritmo de Graham en acción.

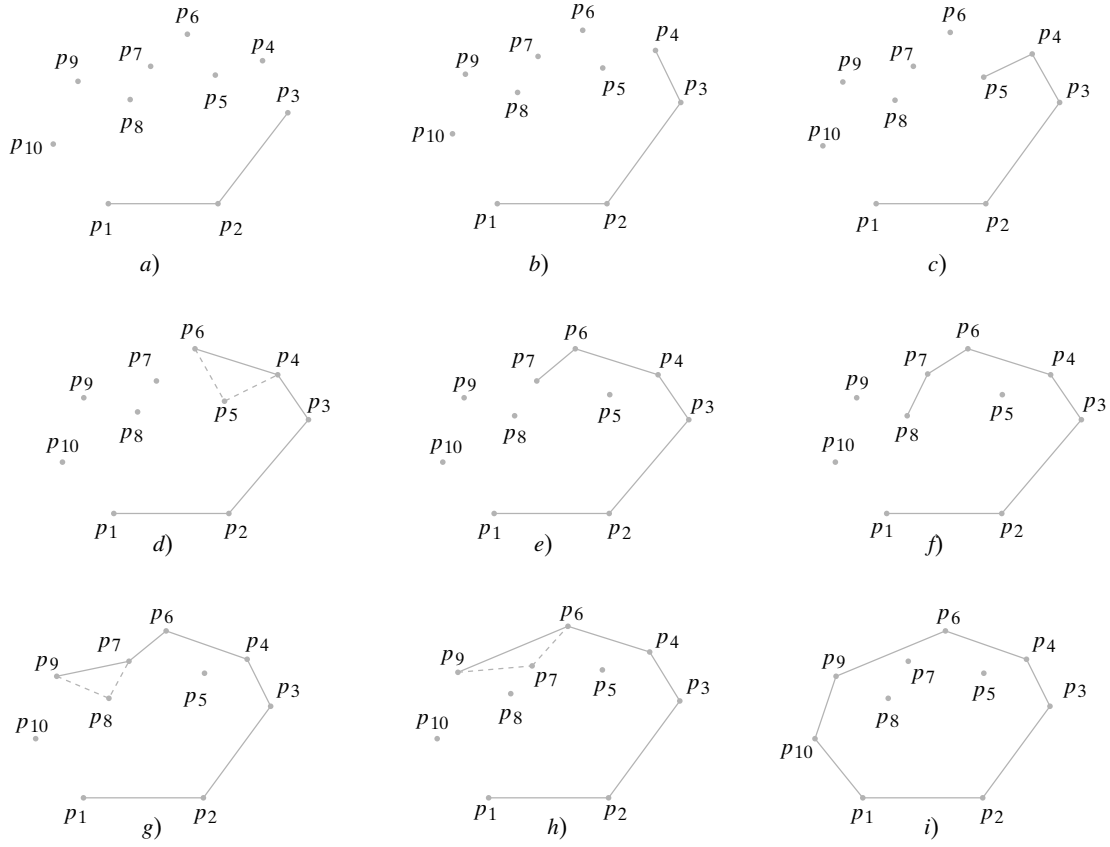


Figura 13.2.10 Algoritmo de Graham para calcular el casco convexo. El punto p_1 tiene la coordenada y mínima y, entre los puntos de este tipo, tiene la coordenada x mínima. Los puntos ordenados según el ángulo de la horizontal a p_1, p_i son $p_2, p_3, \ldots, p_{10}$. En *a)*, el algoritmo comienza por examinar p_1, p_2, p_3. Se ve que p_1, p_2, p_3 dan un giro a la izquierda, y p_2 se conserva. Después en *b)*, p_2, p_3, p_4 se examinan. Se observa que p_2, p_3, p_4 dan un giro a la izquierda, por lo que p_3 se conserva. Luego en *c)*, p_3, p_4, p_5 se examinan. Se observa que p_3, p_4, p_5 dan un giro a la izquierda, y p_4 se conserva. Después en *d)*, p_4, p_5, p_6 se examinan. Ahora, p_4, p_5, p_6 dan un giro a la derecha y p_5 se descarta. El algoritmo regresa a p_3, p_4, p_6. Los puntos p_3, p_4, p_6 dan un giro a la izquierda y p_4 se conserva. Luego en *e)*, se examinan p_4, p_6, p_7. Se ve que p_4, p_6, p_7 dan un giro a la izquierda, por lo que p_6 se conserva. Después en *f)*, p_6, p_7, p_8 se examinan, y se observa que dan un giro a la izquierda, de manera que p_7 se conserva. En *g)*, se examinan p_7, p_8, p_9, que dan un giro a la derecha, por lo que p_8 se descarta. En *h)*, el algoritmo regresa a p_6, p_7, p_9. Los puntos p_6, p_7, p_9 también hacen un giro a la derecha y p_7 se descarta. El algoritmo regresa a los puntos p_4, p_6, p_9, que dan un giro a la izquierda, por lo que p_6 se conserva. Por último, en *i)*, el algoritmo concluye con el examen de p_6, p_9, p_{10}. Estos últimos hacen un giro a la izquierda, por lo que p_9 se conserva. El casco convexo es $p_1, p_2, p_3, p_4, p_6, p_9, p_{10}$.

◀

El algoritmo de Graham comienza con dos ciclos "for" que toman un tiempo $\Theta(n)$ cada uno. Si se utiliza un orden óptimo como el merge sort, el tiempo para ordenar en el peor caso será $\Theta(n \lg n)$. El tiempo de ejecución *total* del último ciclo "while" es $O(n)$ ya que se puede descartar un punto cuando mucho una vez y se tienen n puntos. El ciclo "for" en sí se ejecuta $\Theta(n)$ veces; entonces el tiempo total para los últimos ciclos "for" y "while" es $\Theta(n)$. Se ve que el tiempo para ordenar domina, de manera que el tiempo en el peor caso para el algoritmo de Graham es $\Theta(n \lg n)$. El tiempo, después de ordenar los puntos, es $\Theta(n)$, de manera que si los puntos llegan preordenados, el algoritmo de Graham calculará el casco convexo en un tiempo lineal.

El último teorema establece que cualquier algoritmo que calcule el casco convexo de n puntos en el plano tiene un tiempo en el peor caso de $\Omega(n \lg n)$, por lo que el algoritmo de Graham es óptimo.

Teorema 13.2.8

Cualquier algoritmo que calcule el casco convexo de n puntos en el plano tiene un tiempo en el peor caso de $\Omega(n \lg n)$.

Demostración Sea A un algoritmo que calcula el casco convexo de un conjunto finito de puntos en el plano. Se demostrará que en el peor caso este algoritmo toma tanto tiempo como el algoritmo para ordenar y, por lo tanto, tiene la misma cota inferior $\Omega(n \lg n)$ que el problema de ordenar.

Considere una sucesión arbitraria de números reales

$$y_1, \quad y_2, \quad \ldots, \quad y_n,$$

donde cada y_i está entre 0 y 1. Se usa el algoritmo del casco convexo, algoritmo A, para construir un algoritmo, el algoritmo B, que ordena esta sucesión. Primero, el algoritmo B proyecta estos números reales en el círculo unitario (vea la figura 13.2.11). Después, el algoritmo B llama al algoritmo A para encontrar el casco convexo $h_1, h_2, \ldots, h_n$ de los puntos en el círculo. Por último, al algoritmo B produce las coordenadas y de $h_1, h_2, \ldots, h_n$ en este orden. Observe que la salida del algoritmo B es el ordenamiento de la sucesión de entrada

$$y_1, \quad y_2, \quad \ldots, \quad y_n.$$

Como el algoritmo B es para ordenar, por el Teorema 9.7.3 su tiempo t_n en el peor caso satisface

$$t_n \geq Cn \lg n.$$

Por otro lado, el algoritmo B consiste en dos ciclos $\Theta(n)$ (uno para proyectar los puntos en el círculo unitario y el otro para producir las coordenadas y del casco convexo) y la llamada al algoritmo A que toma un tiempo, digamos, s_n en el peor caso. Entonces

$$t_n = 2n + s_n.$$

Por lo tanto,

$$s_n = t_n - 2n \geq Cn \lg n - 2n = \Omega(n \lg n),$$

y la prueba queda completa.

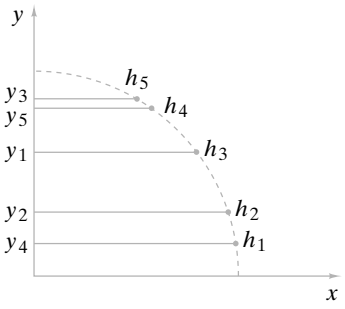


Figura 13.2.11 Ordenamiento usando el algoritmo del casco convexo. Los puntos y_1, y_2, y_3, y_4, y_5 se proyectan primero en el círculo unitario. Los puntos obtenidos en el círculo unitario se denotan por h_i. Después, se usa el algoritmo del casco convexo para encontrar el casco convexo h_1, h_2, h_3, h_4, h_5. Las coordenadas y del casco convexo (en el orden de h_1, h_2, h_3, h_4, h_5) son y_4, y_2, y_1, y_5, y_3, es decir, el orden de las y.

Sección de ejercicios de repaso

1. ¿Qué es un punto del casco?
2. Defina *casco convexo*.
3. ¿Qué es un producto cruz?
4. Describa la manera como el algoritmo de Graham calcula el casco convexo.
5. ¿Cuál es el tiempo en el peor caso de cualquier algoritmo que calcula el casco convexo de n puntos en el plano?
6. Explique cómo se deriva el tiempo en el peor caso de cualquier algoritmo que calcula el casco convexo de n puntos en el plano.

Ejercicios

1. Sea S un conjunto finito de puntos en el plano. Sea p_1 el punto con la coordenada y mínima. Si varios puntos tienen la misma coordenada y mínima, elija el que tiene la menor coordenada x. Pruebe que p_1 está en el casco convexo de S.

2. Pruebe el Teorema 13.2.5 para $x_1 = x_0$.

3. Pruebe el Teorema 13.2.5 para $x_1 > x_0$.

4. Use el algoritmo de Graham para encontrar el casco convexo de los puntos
(10, 1), (7, 7), (3, 13), (6, 10), (16, 4), (10, 5), (7, 13), (13, 8), (4, 4), (2, 2), (1, 8), (10, 13), (7, 1), (4, 8), (12, 3), (16, 10), (14, 5), (10, 9).

5. Utilice el algoritmo de Graham para encontrar el casco convexo de los puntos
(7, 8), (9, 8), (3, 11), (5, 1), (7, 11), (9, 5), (9, 1), (6, 7), (4, 5), (2, 1), (10, 17), (7, 3), (7, 14), (4, 8), (11, 3), (10, 12).

6. Suponga que el algoritmo de Graham se usó para encontrar el casco convexo de un conjunto S de n puntos. Demuestre que si se agrega un punto a S para obtener S', el casco convexo de S' se puede encontrar en un tiempo $\Theta(n)$.

Los ejercicios 7 al 10 se refieren a la marcha de Jarvis, otro algoritmo que calcula el casco convexo de un conjunto finito de puntos en el plano. Al igual que el algoritmo de Graham, comienza por encontrar el punto con la coordenada y mínima. Si ésta es igual para varios puntos se elige la que tiene la coordenada x mínima. Después, la marcha de Jarvis encuentra el punto p_2 tal que el ángulo de la horizontal al segmento de recta p_1, p_2 es mínimo. (En caso de empates, se selecciona el punto más alejado de p_1). Después de encontrar $p_1, \ldots, p_i$, la marcha de Jarvis encuentra el punto p_{i+1} tal que p_{i-1}, p_i, p_{i+1} hacen el menor giro a la izquierda. (En caso de empates, se elige el punto más alejado de p_i).

7. Demuestre que la marcha de Jarvis, de hecho, encuentra el casco convexo.

8. Escriba un seudocódigo para la marcha de Jarvis.

9. Encuentre el tiempo en el peor caso para la marcha de Jarvis.

10. ¿Existen conjuntos para los que la marcha de Jarvis sea más rápida que el algoritmo de Graham? Explique su respuesta.

Notas

[De Berg; Preparata, 1985; y Edelsbrunner] son libros de geometría computacional.

El algoritmo del par más cercano en la sección 13.1 se debe a M. I. Shamos y aparece en [Preparata, 1985]. Esta última referencia también proporciona un algoritmo de $\Theta(n \lg n)$ para encontrar el par más cercano en un número arbitrario de dimensiones.

El algoritmo de Graham (vea [Graham, 1972]) que apareció en 1972 fue uno de los primeros algoritmos de cascos convexos en el plano con $\Theta(n \lg n)$. La marcha de Jarvis se encuentra en [Jarvis]. Calcular el casco convexo de un conjunto de puntos en un espacio de más de dos dimensiones es mucho más complicado que el problema en el plano. El primer algoritmo de tres dimensiones óptimo fue desarrollado en 1977 por [Preparata, 1977]. En 1981, [Siedel] publicó un algoritmo de casco convexo en n dimensiones que es óptimo para n par.

Repaso del capítulo

Sección 13.1

1. Geometría para cálculo
2. Problema del par más cercano
3. Algoritmo del par más cercano

Sección 13.2

4. Punto del casco
5. Casco convexo
6. Producto cruz
7. Algoritmo de Graham para calcular el casco convexo
8. Cualquier algoritmo que calcula el casco convexo de n puntos en el plano tiene un tiempo en peor caso $\Omega(n \lg n)$ (Teorema 13.2.8).

Autoevaluación del capítulo

Sección 13.1

1. Describa la manera como el algoritmo del par más cercano encuentra el par más cercano de puntos si la entrada es la del ejercicio 4, de la sección 13.2.

2. Para terminar las iteraciones recursivas en el algoritmo del par más cercano, si hay tres puntos o menos en la entrada, se encuentra el par más cercano en forma directa. ¿Por qué no se puede sustituir "tres" por "dos"?

3. Demuestre que existen cuando mucho cuatro puntos en la mitad inferior del rectángulo de la figura 13.1.3.

4. ¿Cuál sería el tiempo en el peor caso del algoritmo del par más cercano (algoritmo 13.1.2) si en lugar de fusionar $p_i, \ldots, p_k$ y $p_{k+1}, \ldots, p_j$, se usará el merge sort para ordenar $p_i, \ldots, p_j$?

Sección 13.2

5. Sea S un conjunto finito de puntos en el plano. Sea p el punto en S con coordenada x máxima. Si varios puntos tienen la misma coordenada x, se elige el que tiene la coordenada y máxima. Pruebe que p no está en el casco convexo de S.

6. Sea S un conjunto finito de puntos distintos en el plano. Suponga que S contiene al menos dos puntos. Sean p y q puntos en S que están separados una distancia máxima. Pruebe que p y q están en el casco convexo de S.

7. Utilice el algoritmo de Graham para encontrar el casco convexo del conjunto de puntos del ejercicio 1, de la sección 13.1.

8. Suponga que se usó el algoritmo de Graham para encontrar el casco convexo de un conjunto de $n \geq 2$ puntos. Demuestre que si se elimina un punto distinto de p_1 del algoritmo 13.2.6 de S para obtener S', el casco convexo se puede encontrar en un tiempo $\Theta(n)$.

Ejercicios para computadora

1. Implemente el algoritmo del par más cercano (algoritmo 13.1.2) como un programa que encuentre no sólo la distancia más corta, sino también el par más cercano.

2. Escriba un programa para encontrar todos los pares más cercanos en el plano.

3. Escriba un programa para encontrar el par más cercano en el espacio de tres dimensiones. Un algoritmo está dado en [Preparata].

4. Implemente el algoritmo de Graham para encontrar el casco convexo (algoritmo 13.2.6) como un programa.

5. Implemente la marcha de Jarvis, que encuentra el casco convexo, como un programa (vea los ejercicios 7 al 10, sección 13.2).

Apéndice A

MATRICES

Es una práctica común organizar los datos en columnas y renglones. En matemáticas, este arreglo se llama **matriz**. En este apéndice resumiremos algunas definiciones y propiedades elementales de las matrices. Primero daremos la definición de "matriz".

Definición A.1 ▶

Una *matriz*

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \tag{A.1}$$

es un arreglo rectangular de datos.

Si A tiene m renglones y n columnas, se dice que el *tamaño* de A es m por n (escrito $m \times n$). ◀

Con frecuencia, abreviaremos la ecuación (A.1) como $A = (a_{ij})$. En esta ecuación, a_{ij} denota el elemento de A que aparece en el renglón i y la columna j.

Ejemplo A.2 ▶

La matriz

$$A = \begin{pmatrix} 2 & 1 & 0 \\ -1 & 6 & 14 \end{pmatrix}$$

tiene dos renglones y tres columnas, de manera que su tamaño es 2×3. Si se escribe $A = (a_{ij})$, tendríamos, por ejemplo,

$$a_{11} = 2, \qquad a_{21} = -1, \qquad a_{13} = 0.$$

◀

Definición A.3 ▶

Dos matrices A y B son *iguales* ($A = B$), si son del mismo tamaño y sus elementos correspondientes son iguales. ◀

Ejemplo A.4 ▶

Determine w, x, y y z de manera que

$$\begin{pmatrix} x + y & y \\ w + z & w - z \end{pmatrix} = \begin{pmatrix} 5 & 2 \\ 4 & 6 \end{pmatrix}.$$

De acuerdo con la definición A.3, como las matrices son del mismo tamaño, serán iguales siempre que

$$\begin{aligned} x + y &= 5 & y &= 2 \\ w + z &= 4 & w - z &= 6. \end{aligned}$$

Al resolver estas ecuaciones se obtiene

$$w = 5, \qquad x = 3, \qquad y = 2, \qquad z = -1. \quad \blacktriangleleft$$

Enseguida describimos algunas operaciones que se pueden realizar con matrices. La **suma** de dos matrices se obtiene sumando los elementos correspondientes. El **producto escalar** se obtiene multiplicando cada elemento de la matriz por un número fijo.

Definición A.5 ▶

Sean $A = (a_{ij})$ y $B = (b_{ij})$ dos matrices de $m \times n$. La *suma* de A y B está definida como

$$A + B = (a_{ij} + b_{ij}).$$

El *producto escalar* de un número c y una matriz $A = (a_{ij})$ está definido como

$$cA = (ca_{ij}).$$

Si A y B son matrices, se define $-A = (-1)A$ y $A - B = A + (-B)$. ◀

Ejemplo A.6 ▶

Si

$$A = \begin{pmatrix} 4 & 2 \\ -1 & 0 \\ 6 & -2 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & -3 \\ 4 & 4 \\ -1 & -3 \end{pmatrix},$$

entonces

$$A + B = \begin{pmatrix} 5 & -1 \\ 3 & 4 \\ 5 & -5 \end{pmatrix}, \qquad 2A = \begin{pmatrix} 8 & 4 \\ -2 & 0 \\ 12 & -4 \end{pmatrix}, \qquad -B = \begin{pmatrix} -1 & 3 \\ -4 & -4 \\ 1 & 3 \end{pmatrix}.$$

◀

La multiplicación es otra operación importante con matrices.

Definición A.7 ▶

Sea $A = (a_{ij})$ una matriz de $m \times n$ y sea $B = (b_{jk})$ una matriz de $n \times l$. La *matriz producto* de A y B se define como la matriz de $m \times l$

$$AB = (c_{ik}),$$

donde

$$c_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk}.$$

◀

Para multiplicar la matriz A por la matriz B, la definición A.7 requiere que el número de columnas de A sea igual al número de renglones de B.

Ejemplo A.8 ▶

Sea

$$A = \begin{pmatrix} 1 & 6 \\ 4 & 2 \\ 3 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 2 & -1 \\ 4 & 7 & 0 \end{pmatrix}.$$

La matriz producto AB está definida ya que el número de columnas de A es el mismo que el número de renglones de B; ambos son iguales a 2. El elemento c_{ik} en el producto de AB se obtiene usando el renglón i de A y la columna k de B. Por ejemplo, el elemento c_{31} se calcula usando el tercer renglón

$$(3 \quad 1)$$

de A y la primera columna

$$\begin{pmatrix} 1 \\ 4 \end{pmatrix}$$

de B. Después se multiplica, consecutivamente, cada elemento en el tercer renglón de A por cada elemento de la primera columna de B y después sumamos para obtener

$$3 \cdot 1 + 1 \cdot 4 = 7.$$

Como el número de columnas de A es el mismo que el número de renglones de B, los elementos se aparean correctamente. Procediendo de la misma manera, se obtiene el producto

$$AB = \begin{pmatrix} 25 & 44 & -1 \\ 12 & 22 & -4 \\ 7 & 13 & -3 \end{pmatrix}. \quad \blacktriangleleft$$

Ejemplo A.9 ▶

La matriz producto

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad \text{es} \quad \begin{pmatrix} ax + by \\ cx + dy \end{pmatrix}. \quad \blacktriangleleft$$

Definición A.10 ▶

Sea A una matriz de $n \times n$. Si m es un entero positivo, la m-ésima *potencia* de A se define como la matriz producto

$$A^m = \underbrace{A \cdots A}_{m \ A\text{'s}}. \quad \blacktriangleleft$$

Ejemplo A.11 ▶

Si

$$A = \begin{pmatrix} 1 & -3 \\ -2 & 4 \end{pmatrix},$$

entonces

$$A^2 = AA = \begin{pmatrix} 1 & -3 \\ -2 & 4 \end{pmatrix} \begin{pmatrix} 1 & -3 \\ -2 & 4 \end{pmatrix} = \begin{pmatrix} 7 & -15 \\ -10 & 22 \end{pmatrix}$$

$$A^4 = AAAA = A^2A^2 = \begin{pmatrix} 7 & -15 \\ -10 & 22 \end{pmatrix} \begin{pmatrix} 7 & -15 \\ -10 & 22 \end{pmatrix} = \begin{pmatrix} 199 & -435 \\ -290 & 634 \end{pmatrix}$$

◀

Ejercicios

1. Calcule la suma

$$\begin{pmatrix} 2 & 4 & 1 \\ 6 & 9 & 3 \\ 1 & -1 & 6 \end{pmatrix} + \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}.$$

En los ejercicios 2 al 8, defina

$$A = \begin{pmatrix} 1 & 6 & 9 \\ 0 & 4 & -2 \end{pmatrix}, \qquad B = \begin{pmatrix} 4 & 1 & -2 \\ -7 & 6 & 1 \end{pmatrix}$$

y calcule cada expresión.

2. $A + B$

3. $B + A$

4. $-A$

5. $3A$

6. $-2B$

7. $2B + A$

8. $B - 6A$

En los ejercicios 9 al 13, calcule los productos.

9. $\begin{pmatrix} 1 & 2 & 3 \\ -1 & 2 & 3 \\ 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} 2 & 8 \\ -1 & 1 \\ 6 & 0 \end{pmatrix}$

10. $\begin{pmatrix} 1 & 6 \\ -8 & 2 \\ 4 & 1 \end{pmatrix} \begin{pmatrix} 4 & 1 \\ 7 & -6 \end{pmatrix}$

11. A^2, donde $A = \begin{pmatrix} 1 & -2 \\ 6 & 2 \end{pmatrix}$

12. $\begin{pmatrix} 2 & -4 & 6 & 1 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ -2 \\ 6 \\ 4 \end{pmatrix}$

13. $\begin{pmatrix} 2 & 4 & 1 \\ 6 & 9 & 3 \\ 1 & -1 & 6 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix}$

14. *a)* Diga cuál es el tamaño de cada matriz.

$$A = \begin{pmatrix} 1 & 4 & 6 \\ 0 & 1 & 7 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 4 & 7 \\ 8 & 2 & 1 \\ 0 & 1 & 6 \end{pmatrix},$$

$$C = \begin{pmatrix} 4 & 2 \\ 0 & 0 \\ 2 & 9 \end{pmatrix}$$

b) Usando las matrices del inciso *a)*, decida cuáles de los productos

$$A^2, \quad AB, \quad BA, \quad AC, \quad CA, \quad AB^2,$$
$$BC, \quad CB, \quad C^2$$

están definidos y calcule estos productos.

15. Determine x, y y z de manera que la ecuación

$$\begin{pmatrix} x+y & 3x+y \\ x+z & x+y-2z \end{pmatrix} = \begin{pmatrix} -1 & 1 \\ 9 & -17 \end{pmatrix}$$

se cumpla.

16. Determine w, x, y y z de manera que la ecuación

$$\begin{pmatrix} 2 & 1 & -1 & 7 \\ 6 & 8 & 0 & 3 \end{pmatrix} \begin{pmatrix} x & 2x \\ y & -y+z \\ x+w & w-2y+x \\ z & z \end{pmatrix} = -\begin{pmatrix} 45 & 46 \\ 3 & 87 \end{pmatrix}$$

se cumpla.

17. Defina la matriz de $n \times n$ $I_n = (a_{ij})$ por

$$a_{ij} = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j. \end{cases}$$

La matriz I_n se llama **matriz identidad** $n \times n$.

Demuestre que si A es una matriz de $n \times n$ (que recibe el nombre de **matriz cuadrada**), entonces

$$AI_n = A = I_n A.$$

Se dice que una matriz A de $n \times n$ es invertible si existe una matriz B de $n \times n$ que satisface

$$AB = I_n = BA.$$

(La matriz I_n se definió en el ejercicio 17).

18. Demuestre que la matriz

$$\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$$

se puede invertir.

★19. Demuestre que la matriz

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

es invertible si y sólo si $ad - bc \neq 0$.

20. Suponga que se quiere resolver el sistema

$$AX = C,$$

donde

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

$$X = \begin{pmatrix} x \\ y \end{pmatrix}$$

$$C = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

para x y y.

Demuestre que si A es invertible, el sistema tiene solución.

21. La **transpuesta** de una matriz $A = (a_{ij})$ es la matriz $A^T = (a'_{ji})$, donde $a'_{ji} = a_{ij}$. *Ejemplo:*

$$\begin{pmatrix} 1 & 3 \\ 4 & 6 \end{pmatrix}^T = \begin{pmatrix} 1 & 4 \\ 3 & 6 \end{pmatrix}.$$

Si A y B son matrices de $m \times k$ y $k \times n$, respectivamente, demuestre que

$$(AB)^T = B^T A^T.$$

Apéndice B
REPASO DE ÁLGEBRA

En este apéndice se repasa el álgebra básica: reglas para combinar y simplificar expresiones, fracciones, exponentes, factorización, ecuaciones cuadráticas, desigualdades y logaritmos. Un tratamiento más extenso de álgebra básica se encuentra en [Bleau; Lial; Sullivan].

Agrupamiento

Los términos con un símbolo en común se pueden combinar:

$$ac + bc = (a + b)c, \qquad ac - bc = (a - b)c.$$

Técnicamente, estas ecuaciones se conocen como **leyes distributivas**.

Ejemplo B.1 ▶

$$2x + 3x = (2 + 3)x = 5x \quad \blacktriangleleft$$

Las leyes distributivas, rescritas como

$$a(b + c) = ab + ac, \qquad a(b - c) = ab - ac,$$

son útiles para simplificar expresiones.

Ejemplo B.2 ▶

$$2(x + 1) = 2x + 2 \cdot 1 = 2x + 2 \quad \blacktriangleleft$$

Ejemplo B.3 ▶

$$2(x + 1) + 2(x - 1) = 2x + 2 + 2x - 2 = 4x \quad \blacktriangleleft$$

Fracciones

Las fórmulas útiles para sumar, restar y multiplicar fracciones se dan como el teorema B.4.

Teorema B.4

Combinación de fracciones

a) $\dfrac{a}{c} + \dfrac{b}{c} = \dfrac{a + b}{c}$

b) $\dfrac{a}{c} - \dfrac{b}{c} = \dfrac{a - b}{c}$

c) $\dfrac{a}{c} \quad \dfrac{b}{d} = \dfrac{ad + bc}{cd}$

d) $\dfrac{a}{c} - \dfrac{b}{d} = \dfrac{ad - bc}{cd}$

e) $\dfrac{a}{c} \cdot \dfrac{b}{d} = \dfrac{ab}{cd}$

Ejemplo B.5 ▶ Al usar el Teorema B.4*a)* se obtiene

$$\frac{x-1}{2} + \frac{x+1}{2} = \frac{(x-1)+(x+1)}{2} = \frac{2x}{2} = x. \quad \blacktriangleleft$$

Ejemplo B.6 ▶ Al usar el Teorema B.4*b)* se obtiene

$$\frac{x-1}{2} - \frac{x+1}{2} = \frac{(x-1)-(x+1)}{2} = \frac{-2}{2} = -1. \quad \blacktriangleleft$$

Ejemplo B.7 ▶ Al usar el Teorema B.4*c)* se obtiene

$$\frac{x-1}{2} + \frac{x+1}{3} = \frac{3(x-1)+2(x+1)}{2 \cdot 3} = \frac{5x-1}{6}. \quad \blacktriangleleft$$

Ejemplo B.8 ▶ Al usar el Teorema B.4*d)* se obtiene

$$\frac{x-1}{2} - \frac{x+1}{3} = \frac{3(x-1)-2(x+1)}{2 \cdot 3} = \frac{x-5}{6}. \quad \blacktriangleleft$$

Ejemplo B.9 ▶ Al usar el Teorema B.4*e)* se obtiene

$$\frac{2}{x} \cdot \frac{4}{y} = \frac{8}{xy}. \quad \blacktriangleleft$$

Exponentes

Si n es un entero positivo y a es un número real, a^n se define como

$$a^n = \underbrace{a \cdot a \cdots a}_{n\ a\text{'s}}.$$

Si a es un número real diferente de cero, se define $a^0 = 1$. Si n es un entero negativo y a es un número real diferente de cero, a^n se define como

$$a^n = \frac{1}{a^{-n}}.$$

Ejemplo B.10 ▶ Si a es un número real,

$$a^4 = a \cdot a \cdot a \cdot a.$$

Como ejemplo específico,

$$2^4 = 2 \cdot 2 \cdot 2 \cdot 2 = 16.$$

Si a es un número real diferente de cero,

$$a^{-4} = \frac{1}{a^4}.$$

Como ejemplo específico,

$$2^{-4} = \frac{1}{2^4} = \frac{1}{16}. \quad \blacktriangleleft$$

Si a es un número real positivo y n es un entero positivo, $a^{1/n}$ se define como el número positivo b que satisface

$$b^n = a.$$

b se llama *raíz n-ésima de a*.

Ejemplo B.11 ▶

$3^{1/4}$ con nueve cifras significativas es 1.316074013 porque $(1.316074013)^4$ es aproximadamente 3. ◀

Si a es un número real positivo, m es un entero, y n es un entero positivo, se define

$$a^{m/n} = (a^{1/n})^m.$$

La ecuación anterior define a^q para todos los números reales positivos a y los *números racionales q*. (Recuerde que un número racional es el cociente de dos enteros).

Ejemplo B.12 ▶

Como $3^{1/4}$ con nueve cifras significativas es 1.316074013.

$$3^{9/4} = (1.316074013)^9 = 11.84466612.$$

Los valores decimales son aproximaciones. ◀

Si a es un número real, la definición de a^x se puede extender para incluir todos los números reales x (racionales e irracionales). El siguiente teorema incluye cinco leyes de exponentes importantes.

Teorema B.13

Leyes de exponentes

Sean a y b números reales positivos, y sean x y y números reales. Entonces

a) $a^{x+y} = a^x a^y$
b) $(a^x)^y = a^{xy}$
c) $\dfrac{a^x}{a^y} = a^{x\ \ y}$
d) $a^x b^x = (ab)^x$
e) $\dfrac{a^x}{b^x} = \left(\dfrac{a}{b}\right)^x.$

Ejemplo B.14 ▶

Sea $a = 3$, $x = 2$ y $y = 4$. Entonces $a^x = 9$, $a^y = 81$ y $a^{x+y} = 3^{2+4} = 729$. Ahora

$$a^{x+y} = 729 = 9 \cdot 81 = a^x a^y,$$

lo que ilustra el Teorema B.13*a)*. ◀

Ejemplo B.15 ▶

Sea $a = 3$, $x = 2$ y $y = 4$. Entonces $a^x = 9$, $a^{xy} = 3^8 = 6561$. Ahora

$$(a^x)^y = 9^4 = 6561 = a^{xy},$$

lo que ilustra el Teorema B.13*b)*. ◀

Ejemplo B.16 ▶

Sea $a = 3$, $x = 2$ y $y = 4$. Entonces $a^x = 9$, $a^y = 81$ y $a^{x-y} = 3^{-2} = 1/9$. Ahora

$$\frac{a^x}{a^y} = \frac{9}{81} = \frac{1}{9} = a^{x-y},$$

lo que ilustra el Teorema B.13*c)*. ◀

Ejemplo B.17 ▶ Sea $a = 3$, $b = 4$ y $x = 2$. Entonces $a^x = 9$, $b^x = 16$ y $(ab)^x = 12^2 = 144$. Ahora

$$a^x b^x = 9 \cdot 16 = 144 = (ab)^x,$$

lo que ilustra el Teorema B.13*d)*. ◀

Ejemplo B.18 ▶ Sea $a = 3$, $b = 4$ y $x = 2$. Entonces $a^x = 9$, $b^x = 16$ y

$$\left(\frac{a}{b}\right)^x = \left(\frac{3}{4}\right)^2 = \frac{9}{16}.$$

Ahora

$$\frac{a^x}{b^x} = \frac{9}{16} = \left(\frac{a}{b}\right)^x,$$

lo que ilustra el Teorema B.13*e)*. ◀

Ejemplo B.19 ▶

$$2^x 2^x = 2^{x+x} = 2^{2x} = (2^2)^x = 4^x$$ ◀

Factorización

Se puede usar la ecuación

$$(x + b)(x + d) = x^2 + (b + d)x + bd$$

para factorizar una expresión de la forma $x^2 + c_1 x + c_2$.

Ejemplo B.20 ▶ Factorice $x^2 + 3x + 2$.

Se buscan constantes enteras en la factorización. De acuerdo con la ecuación anterior, $x^2 + 3x + 2$ se factoriza como $(x + b)(x + d)$, donde $b + d = 3$ y $bd = 2$. Si $bd = 2$, y b y d son enteros, las únicas opciones para b y d son 1, 2 y –1, −2. Se encuentra que $b = 1$ y $d = 2$ satisfacen $b + d = 3$ y $bd = 2$. Entonces

$$x^2 + 3x + 2 = (x + 1)(x + 2).$$ ◀

Algunos casos especiales de

$$(x + b)(x + d) = x^2 + (b + d)x + bd$$

son

$$(x + b)^2 = x^2 + 2bx + b^2$$
$$(x - b)^2 = x^2 - 2bx + b^2$$
$$(x + b)(x - b) = x^2 - b^2.$$

Ejemplo B.21 ▶ Utilizando la ecuación $(x + b)^2 = x^2 + 2bx + b^2$, se tiene

$$(x + 9)^2 = x^2 + 18x + 81.$$ ◀

Ejemplo B.22 ▶ Factorice $x^2 - 36$.

Como $36 = 6^2$, se tiene

$$x^2 - 36 = (x + 6)(x - 6).$$ ◀

Se puede usar la ecuación

$$(ax + b)(cx + d) = (ac)x^2 + (ad + bc)x + bd$$

para factorizar una expresión de la forma $c_0x^2 + c_1x + c_2$.

Ejemplo B.23 ▶

Factorice $6x^2 - x - 2$.

Se buscan las constantes enteras en la factorización. Usando la notación anterior, se debe tener

$$ac = 6, \qquad ad + bc = -1, \qquad bd = -2.$$

Como $ac = 6$, las posibilidades para a y c son

$$1, 6 \quad 2, 3 \quad -1, -6 \quad -2, -3.$$

Como $bd = -2$, las únicas posibilidades para b y d son 1, -2, y -1, 2. Puesto que debe tenerse $ad + bc = -1$, se encuentra que $a = 2$, $b = 1$, $c = 3$, y $d = -2$ proporcionan una solución. Por lo tanto, la factorización es

$$6x^2 - x - 2 = (2x + 1)(3x - 2). \quad \blacktriangleleft$$

Ejemplo B.24 ▶

Demuestre que

$$\left[\frac{n(n+1)}{2}\right]^2 + (n+1)^3 = \left[\frac{(n+1)(n+2)}{2}\right]^2.$$

Se muestra la manera como es posible rescribir el lado izquierdo de la ecuación como el lado derecho. Por el teorema B13*d)* y *e)*, obtenemos

$$\left[\frac{n(n+1)}{2}\right]^2 + (n+1)^3 = \frac{n^2(n+1)^2}{4} + (n+1)^3.$$

Como $(n + 1)^2$ es un factor común del lado derecho de esta ecuación, se escribe

$$\frac{n^2(n+1)^2}{4} + (n+1)^3 = (n+1)^2\left[\frac{n^2}{4} + (n+1)\right].$$

Puesto que

$$\frac{n^2}{4} + (n+1) = \frac{n^2 + 4n + 4}{4} = \frac{(n+2)^2}{4},$$

se deduce que

$$(n+1)^2\left[\frac{n^2}{4} + (n+1)\right] = (n+1)^2\left[\frac{(n+2)^2}{4}\right] = \left[\frac{(n+1)(n+2)}{2}\right]^2. \quad \blacktriangleleft$$

Solución de una ecuación cuadrática

Una **ecuación cuadrática** es una ecuación de la forma

$$ax^2 + bx + c = 0.$$

Una **solución** es un valor de x que satisface esta ecuación.

Ejemplo B.25 ▶

El valor $x = -3$ es una solución de la ecuación cuadrática

$$2x^2 + 2x - 12 = 0$$

porque

$$2(-3)^2 + 2(-3) - 12 = 2 \cdot 9 - 6 - 12 = 18 - 18 = 0. \quad \blacktriangleleft$$

Si una ecuación cuadrática se factoriza con facilidad, sus soluciones se obtienen de inmediato.

Ejemplo B.26 ▶

Resuelva la ecuación cuadrática

$$3x^2 - 10x + 8 = 0.$$

Se puede factorizar $3x^2 - 10x + 8$ como

$$3x^2 - 10x + 8 = (x - 2)(3x - 4).$$

Para que esta expresión sea igual a cero, ya sea $x - 2$ o $3x - 4$ debe ser igual a cero. Si $x - 2 = 0$, debe tenerse $x = 2$. Si $3x - 4 = 0$, debe tenerse $x = 4/3$. Entonces las soluciones de la ecuación cuadrática dada son

$$x = 2 \qquad \text{y} \qquad x = \frac{4}{3}.$$ ◀

Las soluciones de una ecuación cuadrática *siempre* se pueden obtener con la **fórmula cuadrática**.

Teorema B.27

Fórmula cuadrática

Las soluciones de

$$ax^2 + bx + c = 0$$

son

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Ejemplo B.28 ▶

La fórmula cuadrática de las soluciones de

$$x^2 - x - 1 = 0$$

como

$$x = \frac{-(-1) \pm \sqrt{(-1)^2 - 4 \cdot 1 \cdot (-1)}}{2 \cdot 1} = \frac{1 \pm \sqrt{1 + 4}}{2} = \frac{1 \pm \sqrt{5}}{2}.$$

Entonces estas soluciones son

$$x = \frac{1 + \sqrt{5}}{2} \qquad \text{y} \qquad x = \frac{1 - \sqrt{5}}{2}.$$

◀

Desigualdades

Si a es **menor que** b, se escribe $a < b$. Si a es **menor o igual que** b, se escribe $a \leq b$. Si a es **mayor que** b, se escribe $a > b$. Si a es **mayor o igual que** b, se escribe $a \geq b$.

Ejemplo B.29 ▶

Suponga que $a = 2$, $b = 8$, $c = 2$. Se tiene

$$a < b, \qquad b > a, \qquad a \leq b, \qquad b \geq a, \qquad a \leq c, \qquad a \geq c.$$ ◀

Las leyes de desigualdades se exponen en el Teorema B.30.

Teorema B.30

Leyes de desigualdades

a) Si $a < b$ y c es cualquier número, entonces $a + c < b + c$.
b) Si $a \leq b$ y c es cualquier número, entonces $a + c \leq b + c$.
c) Si $a > b$ y c es cualquier número, entonces $a + c > b + c$.
d) Si $a \geq b$ y c es cualquier número, entonces $a + c \geq b + c$.
e) Si $a < b$ y $c > 0$, entonces $ac < bc$.
f) Si $a \leq b$ y $c > 0$, entonces $ac \leq bc$.
g) Si $a < b$ y $c < 0$, entonces $ac > bc$.
h) Si $a \leq b$ y $c < 0$, entonces $ac \geq bc$.
i) Si $a > b$ y $c > 0$, entonces $ac > bc$.
j) Si $a \geq b$ y $c > 0$, entonces $ac \geq bc$.
k) Si $a > b$ y $c < 0$, entonces $ac < bc$.
l) Si $a \geq b$ y $c < 0$, entonces $ac \leq bc$.
m) Si $a < b$ y $b < c$, entonces $a < c$.
n) Si $a < b$ y $b \leq c$, entonces $a < c$.
o) Si $a \leq b$ y $b < c$, entonces $a < c$.
p) Si $a \leq b$ y $b \leq c$, entonces $a \leq c$.
q) Si $a > b$ y $b \geq c$, entonces $a > c$.
r) Si $a > b$ y $b > c$, entonces $a > c$.
s) Si $a \geq b$ y $b > c$, entonces $a > c$.
t) Si $a \geq b$ y $b \geq c$, entonces $a \geq c$.

Ejemplo B.31 ▶

Resuelva la desigualdad

$$x - 5 < 6.$$

Por el Teorema B.30*a)*, se puede sumar 5 en ambos lados de la desigualdad para obtener la solución

$$x < 11.$$ ◀

Ejemplo B.32 ▶

Resuelva la desigualdad

$$3x + 4 < x + 10.$$

Por el Teorema B.30*a)*, se puede sumar $-x$ en ambos lados de la desigualdad para obtener

$$2x + 4 < 10.$$

De nuevo, por el Teorema B.30*a)*, se puede sumar -4 en ambos lados de la desigualdad para obtener

$$2x < 6.$$

Por último, se utiliza el Teorema B.30*e)* para multiplicar ambos lados de la desigualdad por $1/2$ y obtener la solución

$$x < 3.$$ ◀

Ejemplo B.33 ▶

Demuestre que si $n > 2m$ y $m > 2p$, entonces $n > 4p$.

Se utiliza el Teorema B.30*i)* para multiplicar ambos lados de $m > 2p$ por 2 para obtener

$$2m > 4p.$$

Como

$$n > 2m,$$

se puede usar el Teorema B.30*q)* para obtener

$$n > 4p\ .$$ ◀

Ejemplo B.34 ▶

Demuestre que

$$\frac{n+2}{n+1} < \frac{4(n+1)^2}{(2n+1)^2}$$

para todo entero positivo n.

Como $(n+1)(2n+1)^2$ es positivo, por el Teorema B.30*e)*,

$$(n+1)(2n+1)^2 \cdot \frac{n+2}{n+1} < (n+1)(2n+1)^2 \cdot \frac{4(n+1)^2}{(2n+1)^2},$$

que se rescribe como

$$(2n+1)^2(n+2) < (n+1)4(n+1)^2.$$

Expandiendo cada lado de la desigualdad, se obtiene

$$4n^3 + 12n^2 + 9n + 2 < 4n^3 + 12n^2 + 12n + 4.$$

Por el Teorema B.30*a)*, se puede sumar $-4n^3 - 12n^2 - 9n - 2$ en ambos lados de la desigualdad para obtener

$$0 < 3n + 2.$$

Esta última desigualdad es cierta para todos los enteros positivos n porque el lado derecho siempre es por lo menos 5. Como los pasos son reversibles (es decir, si se comienza con $0 < 3n + 2$ se obtiene la desigualdad original usando el Teorema B.30), se ha probado la desigualdad. ◀

Logaritmos

En este apartado, b es un número real positivo diferente de 1. Si x es un número real positivo, el **logaritmo base** b **de** x es el exponente al cual debe elevarse b para obtener x. El logaritmo base b de x se denota como $\log_b x$. Entonces si se tiene $y = \log_b x$, la definición establece que $b^y = x$.

Ejemplo B.35 ▶

Se tiene que $\log_2 8 = 3$ porque $2^3 = 8$. ◀

Ejemplo B.36 ▶

A partir de

$$2^{2^x} = n,$$

donde n es un entero positivo, obtenga x.

Sea lg el logaritmo base 2. Entonces, a partir de la definición de logaritmo,

$$2^x = \lg n.$$

De nuevo, a partir de la definición de logaritmo,

$$x = \lg(\lg n).$$ ◀

El siguiente teorema incluye las leyes de logaritmos.

Teorema B.37

Leyes de logaritmos

Suponga que $b > 0$ *y* $b \neq 1$. *Entonces*

a) $b^{\log_b x} = x$

b) $\log_b(xy) = \log_b x + \log_b y$

c) $\log_b \left(\frac{x}{y}\right) = \log_b x - \log_b y$

d) $\log_b (x^y) = y \log_b x$

e) Si $a > 0$ *y* $a \neq 1$, *se tiene* $\log_a x = \frac{\log_b x}{\log_b a}$

f) Si $b > 1$ *y* $x > y > 0$, *entonces* $\log_b x > \log_b y$.

El Teorema B.37*e)* se conoce como **fórmula para cambio de base de logaritmos**. Si se sabe cómo calcular los logaritmos base b, se pueden realizar los cálculos en el lado derecho de la ecuación para obtener el logaritmo base a. El Teorema B.37*f)* dice que si $b > 1$, $\log_b(x)$ es una función creciente.

Ejemplo B.38 ▶

Sean $b = 2$ y $x = 8$. Entonces $\log_b x = 3$. Ahora

$$b^{\log_b x} = 2^3 = 8 = x,$$

que ilustra el Teorema B.37*a)*. ◀

Ejemplo B.39 ▶

Sean $b = 2$, $x = 8$ y $y = 16$. Entonces $\log_b x = 3$, $\log_b y = 4$, y $\log_b(xy) = \log_2 128 = 7$. Ahora

$$\log_b(xy) = 7 = 3 + 4 = \log_b x + \log_b y,$$

que ilustra el Teorema B.37*b)*. ◀

Ejemplo B.40 ▶

Sean $b = 2$, $x = 8$ y $y = 16$. Entonces $\log_b x = 3$, $\log_b y = 4$, y

$$\log_b \left(\frac{x}{y}\right) = \log_2 \frac{1}{2} = -1.$$

Ahora

$$\log_b \left(\frac{x}{y}\right) = -1 = \log_b x - \log_b y,$$

que ilustra el Teorema B.37*c)*. ◀

Ejemplo B.41 ▶

Sean $b = 2$, $x = 4$ y $y = 3$. Entonces $\log_b x = 2$ y

$$\log_b \left(x^y\right) = \log_2 64 = 6.$$

Ahora

$$\log_b \left(x^y\right) = 6 = 3 \cdot 2 = y \log_b x,$$

que ilustra el Teorema B.37*d)*. ◀

Ejemplo B.42 ▶

Suponga que se tiene una calculadora que tiene una tecla de logaritmo para calcular logaritmos base 10, pero que no puede calcular logaritmos base 2. Se usa el Teorema B.37*e)* para calcular $\log_2 40$.

Usando nuestra calculadora, obtenemos

$$\log_{10} 40 = 1.602060, \qquad \log_{10} 2 = 0.301030.$$

El Teorema B.37*e)* ahora da

$$\log_2 40 = \frac{\log_{10} 40}{\log_{10} 2} = \frac{1.602060}{0.301030} = 5.321928. \quad \blacktriangleleft$$

Ejemplo B.43 ▶ Demuestre que si k y n son enteros positivos que satisfacen

$$2^{k-1} < n < 2^k,$$

entonces

$$k - 1 < \lg n < k,$$

donde lg denota el logaritmo base 2.

Por el Teorema B.37*f)*, la función logaritmo es creciente. Por lo tanto,

$$\lg(2^{k-1}) < \lg n < \lg(2^k).$$

Por el Teorema B.37*d)*,

$$\lg(2^{k-1}) = (k-1)\lg 2.$$

Como

$$\lg 2 = \log_2 2 = 1,$$

obtenemos

$$\lg(2^{k-1}) = (k-1)\lg 2 = k - 1.$$

De manera similar,

$$\lg(2^k) = k.$$

De aquí se deduce la desigualdad dada. $\blacktriangleleft$

Ejercicios

En los ejercicios 1 al 3, simplifique las expresiones combinando términos semejantes.

1. $8x - 12x$ **2.** $8y + 3a - 4y - 9a$

3. $6(a+b) - 8(a-b)$

En los ejercicios 4 al 6, combine las fracciones dadas.

4. $\dfrac{8x-4b}{3} + \dfrac{7x+b}{3}$ **5.** $\dfrac{8x-4b}{2} - \dfrac{7x+b}{4}$

6. $\dfrac{8x-4b}{3} \cdot \dfrac{7x+b}{3}$

7. Demuestre que

$$\frac{1}{n} - \frac{1}{n+1} = \frac{1}{n(n+1)}.$$

Utilice este hecho para demostrar que

$$\sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1}.$$

Encuentre el valor de cada expresión en los ejercicios 8 al 13 sin usar una calculadora.

8. 3^4 **9.** 3^{-4} **10.** $(-3)^4$

11. $(-3)^{-4}$ **12.** 1^{10} **13.** 1000^0

14. ¿Cuáles expresiones son iguales?

a) $3^4 3^{10}$ *b)* $(3^4)^{10}$ *c)* 3^{14}

d) $4^3 10^3$ *e)* $2^3 20^3$ *f)* 3^{40}

g) 2187^2

15. Demuestre que $5^n + 4 \cdot 5^n = 5^{n+1}$ para todo entero positivo n.

En los ejercicios 16 al 24, expanda la expresión dada.

16. $(x+3)(x+5)$ **17.** $(x-3)(x+4)$

18. $(2x+3)(3x-4)$ ***19.*** $(x+4)^2$

20. $(x-4)^2$ **21.** $(3x+4)^2$

22. $(x-2)(x+2)$ **23.** $(x+a)(x-a)$

24. $(2x-3)(2x+3)$

En los ejercicios 25 al 36, factorice las expresiones dadas.

25. $x^2 + 6x + 5$ **26.** $x^2 - 3x - 10$

27. $x^2 + 6x + 9$ ***28.*** $x^2 - 8x + 16$

29. $x^2 - 81$

30. $x^2 - 4b^2$

31. $2x^2 + 11x + 5$

32. $6x^2 + x - 15$

33. $4x^2 - 12x + 9$

34. $4x^2 - 9$

35. $9a^2 - 4b^2$

36. $12x^2 - 50x + 50$

37. Demuestre que

$$(n+1)! + (n+1)(n+1)! = (n+2)!$$

para todo entero positivo n.

38. Demuestre que

$$\frac{n(n+1)(2n+1)}{6} + (n+1)^2 = \frac{(n+1)(n+2)(2n+3)}{6}$$

para todo entero positivo n.

39. Demuestre que

$$\frac{n}{2n+1} + \frac{1}{(2n+1)(2n+3)} = \frac{n+1}{2n+3}$$

para todo entero positivo n.

40. Demuestre que

$$7(3 \cdot 2^{n-1} - 4 \cdot 5^{n-1}) - 10(3 \cdot 2^{n-2} - 4 \cdot 5^{n-2}) = 3 \cdot 2^n - 4 \cdot 5^n$$

para todo entero positivo n.

41. Simplifique $2r(n-1)r^{n-1} - r^2(n-2)r^{n-2}$.

En los ejercicios 42 al 44, resuelva la ecuación cuadrática.

42. $x^2 - 6x + 8 = 0$

43. $6x^2 - 7x + 2 = 0$

44. $2x^2 - 4x + 1 = 0$

En los ejercicios 45 al 47, resuelva la desigualdad dada.

45. $2x + 3 \le 9$

46. $2x - 8 > 3x + 1$

47. $\dfrac{x-3}{6} < \dfrac{4x+3}{2}$

48. Demuestre que $\sum_{i=1}^{n} i \le n^2$.

49. Demuestre que

$$(1+ax)(1+x) \ge 1 + (a+1)x$$

para cualquier x y $a \ge 0$.

50. Demuestre que

$$\left(\frac{3}{2}\right)^{n-2}\left(\frac{5}{2}\right) > \left(\frac{3}{2}\right)^{n}$$

para todo entero $n \ge 2$.

51. Demuestre que

$$\frac{2n+1}{(n+2)n^2} > \frac{2}{(n+1)^2}$$

para todo entero positivo n.

52. Demuestre que $6n^2 < 6n^2 + 4n + 1$ para todo entero positivo n.

53. Demuestre que $6n^2 + 4n + 1 \le 11n^2$ para todo entero positivo n.

Encuentre el valor de cada expresión en los ejercicios 54 al 58 sin usar una calculadora (lg *significa* $\log_2$).

54. $\lg 64$

55. $\lg \frac{1}{128}$

56. $\lg 2$

57. $2^{\lg 10}$

58. $\lg 2^{1000}$

Dado que $\lg 3 = 1.584962501$ *y* $\lg 5 = 2.321928095$, *encuentre el valor de cada expresión en los ejercicios 59 al 63* (lg *significa* $\log_2$).

59. $\lg 6$

60. $\lg 30$

61. $\lg 59049$

62. $\lg 0.6$

63. $\lg 0.0375$

Utilice una calculadora con una tecla de logaritmo para encontrar el valor de cada expresión en los ejercicios 64 al 67.

64. $\log_5 47$

65. $\log_7 0.30881$

66. $\log_9 8.888^{100}$

67. $\log_{10}(\log_{10} 1054)$

En los ejercicios 68 al 70, use una calculadora con una tecla de logaritmo para encontrar el valor de x.

68. $5^x = 11$

69. $5^{2x}6^x = 811$

70. $5^{11^x} = 10^{100}$

71. Demuestre que $x^{\log_b y} = y^{\log_b x}$.

Apéndice C
SEUDOCÓDIGO

En este apéndice se describe el seudocódigo usado en este libro.

El signo = denota el **operador asignación**. En seudocódigo, $x = y$ significa "copia el valor de y en x" o, de manera equivalente, "sustituye el valor actual de x con el valor de y". Cuando $x = y$ se ejecuta, el valor de y no cambia.

Ejemplo C.1 ▶

Suponga que el valor de x es 5 y que el valor de y es 10. Después de

$$x = y$$

el valor de x es 10 y el valor de y, que no cambia, también es 10. ◀

En la **instrucción "if"**

if(*condición*)
    acción

si la *condición* es verdadera, se ejecuta la *acción* y el control pasa a la declaración que sigue a la *acción*. Si la *condición* es falsa, la *acción* no se ejecuta y el control pasa de inmediato a la instrucción que sigue a *acción*. Como se aprecia, las sangrías sirven para identificar las declaraciones que forman una *acción*.

Ejemplo C.2 ▶

Suponga que el valor de x es 5, el valor de y es 10 y el valor de z es 15. Considere el segmento.

if ($y > x$)
    $z = x$
$y = z$

Como $y > x$ es verdadera,

$$z = x$$

se ejecuta y el valor de z se hace 5. Luego

$$y = z$$

se ejecuta y el valor de y se hace 5. Ahora, cada una de las variables x, y y z tiene el valor 5. ◀

Las palabras reservadas (como "if") se muestran en tipo normal y las palabras elegidas por el usuario (por ejemplo, variables como x) en *cursivas*.

Se usan los operadores normales de la aritmética $+$, $-$, $*$ (para multiplicación), y $/$ además de los operadores relacionales $==$ (igual a), $\neg =$ (no igual a), $<$, $>$, $\leq$ y $\geq$ y los operadores lógicos $\wedge$ (y), $\vee$ (o) y $\neg$ (no). Se usará $==$ para denotar el operador igualdad

y = para denotar el operador asignación. En ocasiones se utilizarán instrucciones menos formales en aras de clarificar el significado. (*Ejemplo:* Elegir un elemento *x* en *S*).

Ejemplo C.3 ▶

Suponga que el valor de *x* es 5, el valor de *y* es 10 y el valor de *z* es 15. Considere el segmento

```
if (y == x)
    z = x
y = z
```

Como $y == x$ es falso,

```
z = x
```

no se ejecuta. Luego

```
y = z
```

se ejecuta, y el valor de *y* se establece en 15. Ahora el valor de *x* es 5, y cada una de las variables *y* y *z* tiene valor 15. ◀

En una instrucción "if", si la *acción* implica varias instrucciones, éstas se encierran entre corchetes. Un ejemplo de una *acción* con instrucciones múltiples en un "if" es

```
if (x ≥ 0) {
    x = x + 1
    a = b + c
}
```

Una forma alternativa de la instrucción "if" es **"if else"** o "si, de otra manera". En la siguiente instrucción

```
if (condición)
    acción1
else
    acción2
```

si la *condición* es cierta, se ejecuta la *acción1* (pero no la *acción2*) y el control pasa a la instrucción que sigue a la *acción2*. Si *condición* es falsa, se ejecuta la *acción2* (pero no la *acción1*) y el control pasa a la instrucción que sigue a la *acción2*. Si la *acción1* o la *acción2* consiste en instrucciones múltiples, éstas se encierran entre corchetes.

Ejemplo C.4 ▶

Suponga que el valor de *x* es 5, el valor de *y* es 10 y el valor de *z* es 15. Considere el segmento

```
if (y ¬ = x)
    y = x
else
    z = x
a = z
```

Como $y \neg = x$ es cierta, se ejecuta

```
y = x
```

y *y* se hace igual a 5. La instrucción

```
z = x
```

no se ejecuta. Luego

```
a = z
```

se ejecuta y a se establece en 15. Ahora el valor de cada una de x y y es igual a 5, y el valor de a y z es igual a 15. ◀

Ejemplo C.5 ▶

Suponga que el valor de x es 5, el valor de y es 10 y el valor de z es 15. Considere el segmento

```
if (y < x)
    y = x
else
    z = x
a = z
```

Como $y < x$ es falso,

```
    y = x
```

no se ejecuta. En su lugar, se ejecuta

```
    z = x
```

y z se hace igual a 5. Luego se ejecuta

```
    a = z
```

y a se hace igual a 5. Ahora el valor de cada una de x, z y a es 5 y el valor de y es igual a 10. ◀

Dos diagonales, //, señalan el inicio de un **comentario**, que luego se extiende hasta el final de la línea. Los comentarios ayudan al lector a entender el código pero no se ejecutan.

Ejemplo C.6 ▶

En el segmento

```
if (x ≥ 0) { // si x no es negativa, se actualizan x y a
    x = x + 1
    a = b + c
}
```

la parte

```
// si x no es negativa, se actualizan x y a
```

es un comentario. El segmento se ejecuta como si estuviera escrito

```
if (x ≥ 0) {
    x = x + 1
    a = b + c
}
```

◀

El **ciclo "while"** se escribe

```
while (condición)
    acción
```

Si la *condición* es verdadera, se ejecuta la *acción* y la secuencia se repite; es decir, si la *condición* es cierta, la *acción* se ejecuta de nuevo. Esta secuencia se repite hasta que la *condición* se convierte en falsa. Entonces, el control pasa de inmediato a la instrucción que sigue a *acción*. Si la *acción* consiste en instrucciones múltiples, éstas se encierran entre corchetes.

Ejemplo C.7 ▶

Sea $s_1, \ldots, s_n$ una sucesión. Una vez que el segmento

```
grande = s1
i = 2
while (i ≤ n) {
    if (si > grande)
        grande = si
    i = i + 1
}
```

se ejecuta, *grande* es igual al elemento mayor en la sucesión. La idea es recorrer toda la sucesión y guardar el valor más grande encontrado hasta el momento en *grande*. ◀

En el ejemplo C.7 se recorrió una sucesión usando la variable *i* que tomó valores enteros de 2 a *n*. Este tipo de ciclo es tan común que casi siempre se usa un ciclo especial llamado **ciclo "for"**, en lugar del ciclo "while". La forma del **ciclo "for"** es

```
for var = inicio a límite
    acción
```

Igual que en la instrucción "if" y el ciclo "while" anteriores, si la *acción* consiste en varias instrucciones, estas últimas se colocan entre corchetes. Cuando el ciclo "for" se ejecuta, la *acción* se ejecuta para los valores de *var* que van de *inicio* a *límite*. De manera más precisa, *inicio* y *límite* son expresiones que tienen valores enteros. La variable *var* primero se iguala al valor *inicio*. Si *var* ≤ *límite*, se ejecuta la *acción* y luego se suma 1 a *var*. Este proceso se repite. La repetición continúa hasta que *var* > *límite*. Observe que si *inicio* > *límite*, la *acción* nunca se ejecutará.

Ejemplo C.8 ▶

El segmento del ejemplo C.7 se rescribe usando un ciclo "for" como sigue

```
grande = s1
for i = 2 to n
    if (si > grande)
        grande = si
```

◀

Una **función** es una unidad de código que puede recibir una entrada, realizar cálculos y producir una salida. Los **parámetros** describen los datos, variables, etcétera, que se alimentan a y salen de la función. La sintaxis es

```
nombre función(parámetros separados por comas) {
    código para realizar cálculos
}
```

Ejemplo C.9 ▶

La siguiente función, llamada *máx1*, encuentra el número más grande entre *a*, *b* y *c*. Los parámetros *a*, *b* y *c* son parámetros de entrada (es decir, son valores asignados antes de ejecutar la función), y el parámetro *x* es una parámetro de *salida* (es decir, la función asignará un valor a *x*, a saber, el más grande de los números *a*, *b* y *c*).

```
máx1(a, b, c) {
    x = a
    if (b > x) //si b es mayor que x, se actualiza x
        x = b
    if (c > x) //si c es mayor que x, se actualiza x
        x = c
}
```

◀

La **instrucción "return"**

return *x*

termina una función y regresa el valor de *x* al invocador de la función. La instrucción de regreso

return

(sin la *x*) simplemente termina una función. Si no hay instrucción de regresar, la función termina justo antes del corchete de cierre.

Ejemplo C.10 ▶

La función del ejemplo C.9 se rescribe usando la instrucción "return" como sigue

```
máx2(a, b, c) {
   x = a
   if (b > x) //si b es mayor que x, se actualiza x
      x = b
   if (c > x) //si c es mayor que x, se actualiza x
      x = c
   return x
}
```

En lugar de usar un parámetro para el mayor de *a*, *b* y *c*, *máx2* regresa el valor más grande. ◀

Se usan las funciones *print* y *println* para imprimir la salida. La función *println* agrega una nueva línea después de imprimir su argumento (con ello la siguiente salida ocurre a la izquierda de la siguiente línea); por lo demás, las funciones son iguales. El operador + une cadenas. Las cadenas están delimitadas por comillas (como "y"). Si exactamente uno de los operandos de + es una cadena, el otro argumento se convierte en cadena, después de lo cual ocurre la concatenación. El operador de concatenación resulta útil en las funciones de impresión.

Ejemplo C.11 ▶

El segmento

```
for i = 1 to n
   println(s_i)
```

imprime los valores de la sucesión $s_1, \ldots, s_n$, uno por línea. ◀

Ejemplo C.12 ▶

El segmento

```
for i = 1 to n
   println(s_i + "")
println()
```

imprime los valores en la sucesión $s_1, \ldots, s_n$, separados por un espacio, en una línea. Los valores de la sucesión van seguidos por una línea nueva. ◀

Ejercicios

1. Muestre la manera en que el segmento del ejemplo C.7 encuentra el elemento más grande en la sucesión

$$s_1 = 2, \quad s_2 = 3, \quad s_3 = 8, \quad s_4 = 6.$$

2. Muestre la manera en que el segmento del ejemplo C.7 encuentra el elemento más grande en la sucesión

$$s_1 = 8, \quad s_2 = 8, \quad s_3 = 8, \quad s_4 = 1.$$

3. Muestre la manera en que el segmento del ejemplo C.7 encuentra el elemento más grande en la sucesión

$$s_1 = 1, \quad s_2 = 1, \quad s_3 = 1, \quad s_4 = 1.$$

4. Muestre cómo la función *máx1* del ejemplo C.9 encuentra el mayor de los números $a = 4$, $b = -3$ y $c = 5$.

5. Muestre cómo la función *máx1* del ejemplo C.9 encuentra el mayor de los números $a = b = 4$ y $c = 2$.

6. Muestre cómo la función *máx1* del ejemplo C.9 encuentra el mayor de los números $a = b = c = 8$.

7. Escriba una función que regrese el mínimo de a y b.

8. Escriba una función que regrese el máximo de a y b.

9. Escriba una función que intercambie los valores de a y b. (Aquí, a y b son parámetros tanto de entrada como de salida).

10. Escriba una función que imprima todos los números impares entre 1 y n, inclusive.

11. Escriba una función que imprima todos los números negativos, uno por línea, que ocurren en la sucesión de números $s_1, \ldots, s_n$.

12. Escriba una función que imprima todos los índices de los valores en la sucesión $s_1, \ldots, s_n$ que son iguales al valor *val*.

13. Escriba una función que regrese el producto de la sucesión de números $s_1, \ldots, s_n$.

14. Escriba una función que regrese un valor sí y uno no de la sucesión $s_1, s_2, \ldots, s_n$ (es decir, $s_1, s_3, s_5, \ldots$), un valor por línea.

REFERENCIAS

AGARWAL, M., N. SAXENA y N. KAYAL, "PRIMES is in P", http://www.cse.iitk.ac.in/news/primality.html.

AHO, A., J. HOPCROFT y J. ULLMAN, *Data Structures and Algorithms,* Addison-Wesley, Reading, Mass., 1983.

AINSLIE, T., *Ainslie's Complete Hoyle,* Simon and Schuster, 1975.

AKL, S. G., *The Design and Analysis of Parallel Algorithms*, Prentice Hall, Englewood Cliffs, N.J., 1989.

APPEL, K. y W. HAKEN, "Every planar map is four-colorable", *Illinois J. Math.,* núm. 21, 1977, pp. 429-567.

BAASE, S. y A. VAN GELDER, *Computer Algorithms: Introduction to Design and Analysis,* 3a. ed., Addison-Wesley, Reading, Mass., 2000.

BABAI, L. y T. KUCERA, "Canonical labelling of graphs in linear average time", *Proc. 20th Symposium on the Foundations of Computer Science,* 1979, pp. 39-46.

BACHELIS, G. F., "A short proof of Hall's theorem on SDRs", *Amer. Math. Monthly,* núm. 109, 2002, pp. 473-474.

BAIN, V., "An algorithm for drawing the *n*-cube", *College Math. J.,* núm. 29, 1998, pp. 320-322.

BARKER, S. F., *The Elements of Logic,* 5a. ed., McGraw-Hill, Nueva York, 1989.

BELL, R. C., *Board and Table Games for Many Civilizations*, ed. rev., Dover, Nueva York, 1979.

BENTLEY, J., *Programming Pearls,* 2a. ed., Addison-Wesley, Reading, Mass., 2000.

BERGE, C., *Graphs and Hypergraphs,* North-Holland, Amsterdam, 1979.

BERLEKAMP, E. R., J. H. CONWAY y R. K. GUY, *Winning Ways*, vol. 1, 2a. ed., A. K. Peters, Nueva York, 2001.

BERLEKAMP, E. R., J. H. CONWAY y R. K. GUY, *Winning Ways*, vol. 2, 2a. ed., A. K. Peters, Nueva York, 2003.

BILLINGSLEY, P., *Probability and Measure,* 3a. ed., Wiley, Nueva York, 1995.

BLEAU, B. L., *Forgotten Algebra*, 2a. ed., Barron's, Hauppauge, Nueva York, 1994.

BONDY, J. A. y U. S. R. MURTY, *Graph Theory with Applications*, American Elsevier, Nueva York, 1976.

BOOLE, G., *The Laws of Thought*, reimpreso por Dover, Nueva York, 1951.

BRASSARD, G. y P. BRATLEY, *Fundamentals of Algorithms*, Prentice Hall, Upper Saddle River, N.J., 1996.

BRUALDI, R. A., *Introductory Combinatorics,* 3a. ed., Prentice Hall, Upper Saddle River, N.J., 1999.

CARMONY, L., "Odd pie fights", *Math. Teacher*, 72 (1979), 61-64.

CARROLL, J. y D. LONG, *Theory of Finite Automata*, Prentice Hall, Englewood Cliffs, N.J., 1989.

CHARTRAND, G. y L. LESNIAK, *Graphs and Digraphs*, 2a. ed., Wadsworth, Belmont, Calif., 1986.

CHRYSTAL, G., *Textbook of Algebra*, vol. II, 7a. ed., Chelsea, Nueva York, 1964.

CHU, I. P. y R. JOHNSONBAUGH, "Tiling deficient boards with trominoes", *Math. Mag*., núm. 59, 1986, pp. 34-40.

CODD, E. F., "A relational model of data for large shared databanks", *Comm. ACM*, núm. 13, 1970, pp. 377-387.

COHEN, D. I. A., *Introduction to Computer Theory,* 2a. ed., Wiley, Nueva York, 1997.

COPI, I. M. y C. COHEN, *Introduction to Logic*, 10a. ed., Prentice Hall, Upper Saddle River, N.J., 1998.

CORMEN, T. H., C. E. LEISERSON, R. L. RIVEST y C. STEIN, *Introduction to Algorithms*, 2a. ed., MIT Press, Cambridge, Mass., 2001.

CULL, P. y E. F. ECKLUND, JR., "Towers of Hanoi and analysis of algorithms", *Amer. Math. Monthly*, núm. 92, 1985, pp. 407-420.

D'ANGELO, J. P. y D. B. WEST, *Mathematical Thinking: Problem Solving and Proofs*, 2a. ed., Prentice Hall, Upper Saddle River, N.J., 2000.

DATE, C. J., *An Introduction to Database Systems*, 6a. ed., Addison-Wesley, Reading, Mass., 1995.

DAVIS, M. D., R. SIGAL y E. J. WEYUKER, *Computability, Complexity, and Languages*, 2a. ed., Academic Press, San Diego, 1994.

DE BERG, M., M. VAN KREVELD, M. OVERMARS y O. SCHWARZKOPF, *Computational Geometry*, 2a. ed. rev., Springer, Berlín, 2000.

DEO, N., *Graph Theory and Applications to Engineering and Computer Science*, Prentice Hall, Englewood Cliffs, N.J., 1974.

DIJKSTRA, E. W., "A note on two problems in connection with graphs", *Numer. Math.,* núm. 1, 1959, pp. 260-271.

DIJKSTRA, E. W., "Cooperating sequential processes," in *Programming Languages,* F. Genuys, ed., Academic Press, Nueva York, 1968.

EDELSBRUNNER, H., *Algorithms in Combinatorial Geometry,* Springer-Verlag, Nueva York, 1987.

EDGAR, W. J., *The Elements of Logic*, SRA, Chicago, 1989.

ENGLISH, E. y S. HAMILTON, "Network security under siege, the timing attack", *Computer,* marzo, 1996, pp. 95-97.

EVEN, S., *Algorithmic Combinatorics,* Macmillan, Nueva York, 1973.

EVEN, S., *Graph Algorithms,* Computer Science Press, Rockville, Md., 1979.

EZEKIEL, M., "The cobweb theorem", *Quart. J. Econom.,* núm. 52, 1938, pp. 255-280.

FORD, L. R., JR. y D. R. FULKERSON, *Flows in Networks,* Princeton University Press, Princeton, N.J., 1962.

FOWLER, P. A., "The Königsberg bridges-250 years later", *Amer. Math. Monthly*, núm. 95, 1988, pp. 42-43.

FREY, P., "Machine-problem solving-Part 3: The alpha-beta procedure", *BYTE*, núm. 5, noviembre, 1980, pp. 244-264.

FUKUNAGA, K., *Introduction to Statistical Pattern Recognition,* 2a. ed., Academic Press, Nueva York, 1990.

GALLIER, J. H., *Logic for Computer Science*, Harper & Row, Nueva York, 1986.

GARDNER, M., *Mathematical Puzzles and Diversions,* Simon and Schuster, 1959.

GARDNER, M., "A new kind of cipher that would take millions of years to break", *Sci. Amer*., febrero, 1977, pp. 120-124.

GARDNER, M., *Mathematical Circus*, Mathematical Association of America, Washington, D. C., 1992.

GENESERETH, M. R. y N. J. NILSSON, *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann, Los Altos, Calif., 1987.

GHAHRAMANI, S., *Fundamentals of Probability,* 2a. ed., Prentice Hall, Upper Saddle River, N.J., 2000.

GIBBONS, A., *Algorithmic Graph Theory*, Cambridge University Press, Cambridge, 1985.

GOLDBERG, S., *Introduction to Difference Equations*, Wiley, Nueva York, 1958.

GOLOMB, S. W., "Checker boards and polyominoes", *Amer. Math. Monthly*, núm. 61, 1954, pp. 675-682.

GOLOMB, S. y L. BAUMERT, "Backtrack programming" *J. ACM*, núm. 12, 1965, pp. 516-524.

GOSE, E., R. JOHNSONBAUGH y S. JOST, *Pattern Recognition and Image Analysis*, Prentice Hall, Upper Saddle River, N.J., 1996.

GRAHAM, R. L., "An efficient algorithm for determining the convex hull of a finite planar set", *Info. Proc. Lett.*, núm. 1, 1972, pp. 132-133.

GRAHAM, R. L., D. E. KNUTH y O. PATASHNIK, *Concrete Mathematics: A Foundation for Computer Science*, 2a. ed., Addison-Wesley, Reading, Mass., 1994.

GRIES, D., *The Science of Programming*, Springer-Verlag, Nueva York, 1981.

HAILPERIN, T., "Boole's algebra isn't Boolean algebra", *Math. Mag*., núm. 54, 1981, pp. 137-184.

HALMOS, P. R., *Naive Set Theory*, Springer-Verlag, Nueva York, 1974.
HARARY, F., *Graph Theory*, Addison-Wesley, Reading, Mass., 1969.
HELL, P., "Absolute retracts in graphs", en *Graphs and Combinatorics,* R. A. Bari y F. Harary, eds., Lecture Notes in Mathematics, vol. 406, Springer-Verlag, Nueva York, 1974.
HILLIER, F. S. y G. J. LIEBERMAN, *Introduction to Operations Research,* 6a. ed., McGraw-Hill, Nueva York, 1995.
HINZ, A. M., "The Tower of Hanoi", *Enseignement Math*., núm. 35, 1989, pp. 289-321.
HOHN, F., *Applied Boolean Algebra,* 2a. ed., Macmillan, Nueva York, 1966.
HOPCROFT, J. E., R. MOTWANI y J. D. ULLMAN, *Introduction to Automata Theory, Languages, and Computation*, 2a. ed., Addison-Wesley, Boston, 2001.
HU, T. C., *Combinatorial Algorithms*, Addison-Wesley, Reading, Mass., 1982.
JACOBS, H. R., *Geometry*, 2a. ed., W. H. Freeman, San Francisco, 1987.
JARVIS, R. A., "On the identification of the convex hull of a finite set of points in the plane", *Info. Proc. Lett*., núm. 2, 1973, pp. 18-21.
JOHNSONBAUGH, R. y M. SCHAEFER, *Algorithms*, Prentice Hall, Upper Saddle River, N.J., 2004.
JONES, R. H. y N. C. STEELE, *Mathematics in Communication Theory*, Ellis Horwood, Chichester, Inglaterra, 1989.
KELLEY, D., *Automata and Formal Languages*, Prentice Hall, Upper Saddle River, N.J., 1995.
KELLY, D. G., *Introduction to Probability*, Prentice Hall, Upper Saddle River, N.J., 1994.
KLEINROCK, L., *Queueing Systems*, vol. 2: *Computer Applications*, Wiley, Nueva York, 1976.
KLINE, M., *Mathematical Thought from Ancient to Modern Times*, Oxford University Press, Nueva York, 1972.
KNUTH, D. E., "Algorithms", *Sci. Amer,* abril, 1977, pp. 63-80.
KNUTH, D. E., "Algorithmic thinking and mathematical thinking", *Amer. Math. Monthly*, núm. 92, 1985, pp. 170-181.
KNUTH, D. E., *The Art of Computer Programming*, vol. 1: *Fundamental Algorithms*, 3a. ed., Addison-Wesley, Reading, Mass., 1997.
KNUTH, D. E., *The Art of Computer Programming*, vol. 2: *Seminumeric Algorithms*, 3a. ed., Addison-Wesley, Reading, Mass., 1998a.
KNUTH, D. E., *The Art of Computer Programming*, vol. 3: *Sorting and Searching*, 2a. ed., Addison-Wesley, Reading, Mass., 1998b.
KÖBLER, J., U. SCHÖNING y J. TORÁN, *The Graph Isomorphism Problem: Its Structural Complexity*, Birkhäuser Verlag, Basilea, Suiza, 1993.
KOHAVI, Z., *Switching and Finite Automata Theory*, 2a. ed., McGraw-Hill, Nueva York, 1978.
KÖNIG, D., *Theorie der endlichen und unendlichen Graphen,* Akademische Verlags-gesellschaft, Leipzig, 1936. (Reimpreso en 1950 por Chelsea, Nueva York). (Traducción al ingés: *Theory of Finite and Infinite Graphs*, Birkhäuser Boston, Cambridge, Mass., 1990).
KRANTZ, S. G., *Techniques of Problem Solving*, American Mathematical Society, Providence, R.I., 1997.
KROENKE, D. M., *Database Processing: Fundamentals, Design and Implementation*, 7a. ed., Prentice Hall, Upper Saddle River, N.J., 2000.
KRUSE, R. L. y A. RYBA, *Data Structures and Program Design in C++*, Prentice Hall, Upper Saddle River, N.J., 1999.
KUROSAKA, R. T., "A ternary state of affairs", *BYTE*, núm. 12, febrero, 1987, pp. 319-328.
LEIGHTON, F. T., *Introduction to Parallel Algorithms and Architectures*, Morgan Kaufmann, San Mateo, Calif., 1992.
LESTER, B. P., *The Art of Parallel Programming*, Prentice Hall, Upper Saddle River, N.J., 1993.
LEWIS, T. G. y H. EL-REWINI, *Introduction to Parallel Computing*, Prentice Hall, Upper Saddle River, N.J., 1992.
LIAL, M. L., E. J. HORNSBY, D. I. SCHNEIDER y C. D. MILLER, *College Algebra,* 7a. ed., Addison-Wesley, Nueva York, 1997.

LINDENMAYER, A., "Mathematical models for cellular interaction in development", partes I y II, J. *Theoret. Biol*., núm. 18, 1968, pp. 280-315.

LIPSCHUTZ, S., *Schaum's Outline of Theory and Problems of Set Theory and Related Topics*, 2a. ed., McGraw-Hill, Nueva York, 1998.

LIU, C. L., *Introduction to Combinatorial Mathematics*, McGraw-Hill, Nueva York, 1968.

LIU, C. L., *Elements of Discrete Mathematics*, 2a. ed., McGraw-Hill, Nueva York, 1985.

MANBER, U., *Introduction to Algorithms*, Addison-Wesley, Reading, Mass., 1989.

MANDELBROT, B. B., *Fractals: Form, Chance, and Dimension*, W. H. Freeman, San Francisco, 1977.

MANDELBROT, B. B., *The Fractal Geometry of Nature*, W. H. Freeman, San Francisco, 1982.

MARTIN, G. E., *Polyominoes: A Guide to Puzzles and Problems in Tiling*, Mathematical Association of America, Washington, D. C., 1991.

MCCALLA, T. R., *Digital Logic and Computer Design*, Merrill, Nueva York, 1992.

MCNAUGHTON, R., *Elementary Computability, Formal Languages, and Automata*, Prentice Hall, Englewood Cliffs, N.J., 1982.

MENDELSON, E., *Boolean Algebra and Switching Circuits,* Schaum, Nueva York, 1970.

MILLER, R. y L. BOXER, A *Unified Approach to Sequential and Parallel Algorithms*, Prentice Hall, Upper Saddle River, N.J., 2000.

MITCHISON, G. J., "Phyllotaxis and the Fibonacci series", *Science*, núm. 196, 1977, pp. 270-275.

NADLER, M. y E. P. SMITH, *Pattern Recognition Engineering,* Wiley, Nueva York, 1993.

NAYLOR, M., "Golden, $\sqrt{2}$, and π flowers: A, spiral story", *Math. Mag*., núm. 75, 2002, pp. 163-172.

NEWMAN, J. R., "Leonhard Euler and the Koenigsberg bridges", *Sci. Amer*. núm. 7, julio, 1953, pp. 66-70.

NIEVERGELT, J., J. C. FARRAR y E. M. REINGOLD, *Computer Approaches to Mathematical Problems*, Prentice Hall, Englewood Cliffs, N.J., 1974.

NILSSON, N. J., *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill, Nueva York, 1971.

NIVEN, I., *Mathematics of Choice*, Mathematical Association of America, Washington, D. C., 1965.

NIVEN, I. y H. S. ZUCKERMAN, *An Introduction to the Theory of Numbers*, 4a. ed., Wiley, Nueva York, 1980.

NYHOFF, L. R., *C++: An Introduction to Data Structures*, Prentice Hall, Upper Saddle River, N.J., 1999.

ORE, O., *Graphs and Their Uses*, Mathematical Association of America, Washington, D. C., 1963.

PEARL, J., "The solution for the branching factor of the alpha-beta pruning algorithm and its optimality", *Comm. ACM*, núm. 25, 1982, pp. 559-564.

PEITGEN, H. y D. SAUPE, eds., *The Science of Fractal Images*, Springer-Verlag, Nueva York, 1988.

PFLEEGER, C. P., *Security in Computing*, 2a. ed., Prentice Hall, Upper Saddle River, N.J., 1997.

PREPARATA, F. P. y S. J. HONG, "Convex hulls of finite sets of points in two and three dimensions", *Comm. ACM*, núm. 20, 1977, pp. 87-93.

PREPARATA, F. P. y M. I. SHAMOS, *Computational Geometry*, Springer-Verlag, Nueva York, 1985. Problema 1186, *Math. Mag.,* núm. 58, 1985, pp. 112-114.

PRODINGER, H. y R. TICHY, "Fibonacci numbers of graphs", *Fibonacci Quarterly*, núm. 20, 1982, pp. 16-21.

PRUSINKIEWICZ, P., "Graphical applications of L-systems", *Proc. of Graphics Interface* 1986-*Vision Interface,* 1986, pp. 247-253.

PRUSINKIEWICZ, P. y J. HANAN, "Applications of L-systems to computer imagery", en *Graph Grammars and Their Application to Computer Science; Third International Workshop*, H. Ehrig, M. Nagl, A. Rosenfeld y G. Rozenberg, eds., Springer-Verlag, Nueva York, 1988.

QUINN, M. J., *Designing Efficient Algorithms for Parallel Computers,* McGraw-Hill, Nueva York, 1987.

READ, R. C. y D. G. CORNEIL, "The graph isomorphism disease", *J. Graph Theory*, núm. 1, 1977, pp. 339-363.

REINGOLD, E., J. NIEVERGELT y N. DEO, *Combinatorial Algorithms*, Prentice Hall, Englewood Cliffs, N.J., 1977.

RIORDAN, J., *An Introduction to Combinatorial Analysis*, Wiley, Nueva York, 1958.

ROBERTS, F. S., *Applied Combinatorics,* Prentice Hall, Englewood Cliffs, N.J., 1984.

ROBINSON, J. A., "A machine oriented logic based on the resolution principle", *J. ACM*, núm. 12, 1965, pp. 23-41.

ROSS, S. M., *A First Course in Probability*, 5a. ed., Prentice Hall, Upper Saddle River, N.J., 1998.

ROZANOV, Y. A., *Probability Theory:* A *Concise Course*, Dover, Nueva York, 1969.

SAAD, Y. y M. H. SCHULTZ, "Topological properties of hypercubes", *IEEE Trans. Computers*, núm. 37, 1988, pp. 867-872.

SCHUMER, P., "The Josephus problem: Once more around," *Math. Mag*., núm. 75, 2002, pp. 12-17.

SCHWENK, A. J., "Which rectangular chessboards have a knight's tour?", *Math. Mag*., 64 (1991), 325-332.

SEIDEL, R., "A convex hull algorithm optimal for points in even dimensions", tesis M.S., informe técnico. 81-14, Dept. of Comp. Sci., Univ. of British Columbia, Vancouver, Canadá, 1981.

SHANNON, C. E., "A symbolic analysis of relay and switching circuits", *Trans. Amer. Inst. Electr. Engrs*., núm. 47, 1938, pp. 713-723.

SIGLER, L., *Fibonacci's Liber Abaci*, Springer-Verlag, Nueva York, 2003.

SLAGLE, J. R., *Artificial Intelligence: The Heuristic Programming Approach,* McGraw-Hill, Nueva York, 1971.

SMITH, A. R., "Plants, fractals, and formal languages", *Computer Graphics*, núm. 18, 1984, pp. 1-10.

SOLOW, D., *How to Read and* Do *Proofs*, 2a. ed., Wiley, Nueva York, 1990.

STANDISH, T. A., *Data Structures in Java*, Addison-Wesley, Reading, Mass., 1997.

STOLL, R. R., *Set Theory and Logic*, Dover, Nueva York, 1979.

SUDKAMP, T. A., *Languages and Machines: An Introduction to the Theory of Computer Science*, 2a. ed., Addison-Wesley, Reading, Mass., 1996.

SULLIVAN, M., *College Algebra*, 5a. ed., Prentice Hall, Upper Saddle River, N.J., 1999.

TARJAN, R. E., *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, Filadelfia, 1983.

TAUBES, G., "Small army of code-breakers conquers a 129-digit giant", *Science*, núm. 264, 1994, pp. 776-777.

TUCKER, A., *Applied Combinatorics*, 3a. ed., Wiley, Nueva York, 1995.

ULLMAN, J. D. y J. WIDOM, *A First Course in Database Systems*, Prentice Hall, Upper Saddle River, N.J., 1997.

VILENKIN, N. Y., *Combinatorics*, Academic Press, Nueva York, 1971.

WAGON, S., "Fourteen proofs of a result about tiling a rectangle", *Amer. Math. Monthly*, núm. 94, 1987, pp. 601-617. (Reimpreso: R. K. Guy y R. E. Woodrow, eds., *The Lighter Side of Mathematics*, Mathematical Association of America, Washington, D.C., 1994, pp. 113-128).

WARD, S. A. y R. H. HALSTEAD, JR., *Computation Structures*, MIT Press, Cambridge, Mass., 1990.

WEST, D., *Introduction to Graph Theory*, 2a. ed., Prentice Hall, Upper Saddle River, N.J., 2000.

WILSON, R. J., *Introduction to Graph Theory*, 4a. ed., Addison-Wesley, Reading, Mass., 1996.

WONG, D. F. y C. L. LIU, "A new algorithm for floorplan design", 23rd Design Automation Conference, 1986, pp. 101-107.

WOOD, D., *Theory of Computation*, Harper & Row, Nueva York, 1987.

WOS, L., R. OVERBEEK, E. LUSK y J. BOYLE, *Automated Reasoning*, Prentice Hall, Englewood Cliffs, N.J., 1984.

SUGERENCIAS Y SOLUCIONES PARA EJERCICIOS SELECCIONADOS

Sección 1.1 Repaso

1. Una proposición es una afirmación que es verdadera o falsa, pero no ambas.

2. La tabla de verdad de una proposición P formada con las proposiciones $p_1, \ldots, p_n$ lista todas las combinaciones de los valores de verdad para $p_1, \ldots, p_n$, donde V denota verdadero y F, falso, y para cada combinación da el valor de verdad de P.

3. La conjunción de las proposiciones p y q es la proposición p *y* q. Se denota por $p \wedge q$.

4.

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

5. La disyunción de las proposiciones p y q es la proposición p *o* q. Se denota por $p \vee q$.

6.

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

7. La negación de la proposición p es la proposición *no* p. Se denota por $\neg p$.

8.

p	$\neg p$
V	F
F	V

Sección 1.1

1. Es una proposición. Negación: $2 + 5 \neq 19$

4. Es una proposición. Negación: Audrey Meadows no era la "Alice" original en "The Honeymooners".

7. Es una proposición. Un entero par mayor que 4 no es la suma de dos primos.

9. No se obtuvieron 10 caras. (Alternativa: se obtuvo al menos una cruz).

12. No se obtuvieron caras. (Alternativa: se obtuvieron 10 cruces).

13. Verdadero

16. Verdadero

19.

p	q	$p \wedge \neg q$
V	V	F
V	F	V
F	V	F
F	F	F

22.

p	q	$(p \wedge q) \wedge \neg q$
V	V	F
V	F	F
F	V	F
F	F	F

25.

p	q	$(p \vee q) \wedge (\neg p \vee q) \wedge (p \vee \neg q) \wedge (\neg p \vee \neg q)$
V	V	F
V	F	F
F	V	F
F	F	F

27. $p \wedge q$; falso

30. Leo no toma ciencias de la computación.

33. Leo toma ciencias de la computación o Leo no toma matemáticas.

36. Hoy es lunes o está lloviendo.

39. (Hoy es lunes y está lloviendo) y no ocurre que (hace calor o que hoy es lunes).

41. $\neg p$

44. $\neg p \wedge \neg q$

47. $p \wedge r$

50. $(p \vee q) \wedge \neg r$

54. No, si se supone la interpretación: estará fuera de la ley una persona que tenga más de tres [3] perros y más de tres [3] gatos en su propiedad dentro de la ciudad. Un juez determinó que el decreto era "ambiguo". Puede decirse que el significado que se quería era: "Estará fuera de la ley una persona que tenga más de tres [3] perros *o* más de tres [3] gatos en su propiedad dentro de la ciudad."

55. nacionales Y parques Y (norte O sur) Y dakota.

Sección 1.2 Repaso

1. Si p y q son proposiciones, la proposición condicional es la proposición si p entonces q. Se denota por $p \rightarrow q$.

2.

p	q	$p \rightarrow q$
V	V	V
V	F	F
F	V	V
F	F	V

3. En la proposición condicional $p \rightarrow q$, p es la hipótesis.

4. En la proposición condicional $p \rightarrow q$, q es la conclusión.

5. En la proposición condicional $p \rightarrow q$, q es la condición necesaria.

6. En la proposición condicional $p \rightarrow q$, p es la condición suficiente.

7. La recíproca de $p \rightarrow q$ es $q \rightarrow p$.

8. Si p y q son proposiciones, la proposición bicondicional es la proposición p si y sólo si q. Se denota por $p \leftrightarrow q$.

9.

p	q	$p \leftrightarrow q$
V	V	V
V	F	F
F	V	F
F	F	V

10. Si las proposiciones P y Q está formadas por las proposiciones $p_1, \ldots, p_n$, P y Q son equivalentes lógicos siempre que dados cualesquiera valores de verdad de $p_1, \ldots, p_n$, ya sea que P y Q son ambas verdaderas o P y Q son ambas falsas.

11. $\neg(p \vee q) \equiv \neg p \wedge \neg q, \neg(p \wedge q) \equiv \neg p \vee \neg q$

12. La contrapositiva de $p \rightarrow q$ es $\neg q \rightarrow \neg p$.

Sección 1.2

1. Si José estudia duro, entonces pasará el examen de matemáticas discretas.

4. Si Katia pasa matemáticas discretas, entonces tomará el curso de algoritmos.

7. Si el programa se puede leer, entonces está bien estructurado.

8. (Para el ejercicio 1) Si José pasa el examen de matemáticas discretas, entonces estudió duro.

10. Verdadera **13.** Falsa **16.** Falsa

18. Verdadera **21.** Verdadera **24.** Verdadera

27. Verdadera **28.** $p \rightarrow q$ **31.** $q \leftrightarrow (p \wedge \neg r)$

32. Si hoy es lunes, entonces está lloviendo.

35. No ocurre que hoy es lunes o está lloviendo si y sólo si hace calor.

38. Sea p: $4 < 6$ y q: $9 > 12$.

La afirmación dada : $p \rightarrow q$; falsa.

Recíproca: $q \rightarrow p$; si $9 > 12$, entonces $4 < 6$; verdadera.

Contrapositiva: $\neg q \rightarrow \neg p$; si $9 \leq 12$, entonces $4 \geq 6$; falsa.

41. Sea p: $|4| < 3$ y q: $-3 < 4 < 3$.

La afirmación dada: $q \rightarrow p$; verdadera.

Recíproca: $p \rightarrow q$; si $|4| < 3$, entonces $-3 < 4 < 3$; verdadera

Contrapositiva: $\neg p \rightarrow \neg q$, si $|4| \geq 3$, entonces $-3 \geq 4$ o $4 \geq 3$; verdadera.

42. $P \not\equiv Q$ **45.** $P \not\equiv Q$

48. $P \not\equiv Q$ **51.** $P \not\equiv Q$

52.

p	q	p *impl* q	q *impl* p
V	V	V	V
V	F	F	F
F	V	F	F
F	F	V	V

Como p *impl* q es verdadera precisamente cuando q *impl* p es verdadera, p *impl* $q \equiv q$ *impl* p.

55.

p	q	$p \rightarrow q$	$\neg p \vee q$
V	V	V	V
V	F	F	F
F	V	V	V
F	F	V	V

Como $p \rightarrow q$ es verdadera precisamente cuando $\neg p \vee q$ es verdadera, $p \rightarrow q \equiv \neg p \vee q$.

Sección 1.3 Repaso

1. Si $P(x)$ es una afirmación que incluye la variable x, P se llama una función proposicional si para cada x en el dominio de discurso, $P(x)$ es una proposición.

2. Un dominio de discurso para una función proposicional P es un conjunto D tal que $P(x)$ está definida para todo x en D.

3. Una afirmación cuantificada universalmente es una afirmación de la forma para todo x en el dominio de discurso, $P(x)$.

4. Un contraejemplo para la afirmación $\forall x\, P(x)$ es un valor de x para el que $P(x)$ es falsa.

5. Una afirmación cuantificada existencialmente es una afirmación de la forma para alguna x en el dominio de discurso, $P(x)$.

6. $\neg(\forall x\, P(x))$ y $\exists x \neg P(x)$ tienen los mismos valores de verdad.

$\neg(\exists x\, P(x))$ y $\forall x \neg P(x)$ tienen los mismos valores de verdad.

7. Para probar que una afirmación cuantificada universalmente $\forall x\, P(x)$ es verdadera, demuestre que para todo x en el dominio de discurso, la proposición $P(x)$ es verdadera.

8. Para probar que la afirmación cuantificada existencialmente $\exists x\, P(x)$ es verdadera, encuentre un valor de x en el dominio de discurso para el que $P(x)$ es verdadera.

9. Para probar que una afirmación cuantificada universalmente $\forall x\, P(x)$ es falsa, encuentre un valor de x en el dominio de discurso para el que la proposición $P(x)$ es falsa.

10. Para probar que la afirmación cuantificada existencialmente $\exists x\, P(x)$ es falsa, demuestre que para toda x en el dominio de discurso, la proposición $P(x)$ es falsa.

Sección 1.3

1. Es una función proposicional. El dominio de discurso podría ser todos los enteros.

4. Es una función proposicional. El dominio de discurso es el conjunto de todas las películas.

7. 11 divide a 77. Verdadera.

10. Para todo entero positivo n, n divide a 77. Falsa.

12. Todo estudiante está en el curso de matemáticas.

15. Algún estudiante no está en el curso de matemáticas.

18. (Para el ejercicio 2) $\exists x \neg P(x)$. Algún estudiante no está en el curso de matemáticas.

19. Todo atleta profesional juega fútbol. Falsa.

22. Alguien no juega fútbol o algún jugador de fútbol es un atleta profesional. Verdadera.

25. Todos son atletas profesionales y jugadores de fútbol. Falsa.

27. (Para el ejercicio 19) $\exists x(P(x) \wedge \neg Q(x))$. Alguien es un atleta profesional y no juega fútbol.

28. $\forall x(P(x) \rightarrow Q(x))$.

31. $\exists x(P(x) \wedge Q(x))$.

32. (Para el ejercicio 28) $\exists x(P(x) \wedge \neg Q(x))$. Algún contador no es dueño de un Porsche.

33. Falso. Un contraejemplo es $x = 0$.

36. Verdadero. El valor $x = 2$ hace que $(x > 1) \rightarrow (x^2 > x)$ sea verdadera.

39. (Para el ejercicio 33) $\exists x(x^2 \leq x)$. Existe x tal que $x^2 \leq x$.

41. El significado literal es: Ningún hombre engaña a su esposa. El significado deseado es: Algún hombre no engaña a su esposa. Sea $P(x)$ la afirmación "x es un hombre" y sea $Q(x)$ la afirmación "x engaña a su esposa". En símbolos, la afirmación aclarada es $\exists x(P(x) \wedge \neg Q(x))$.

44. El significado literal es: Ningún problema ambiental es una tragedia. El significado deseado es: Algún problema ambiental no es una tragedia. Sea $P(x)$ la afirmación "x es un problema ambiental" y sea $Q(x)$ la afirmación "x es una tragedia". En símbolos, la afirmación aclarada es $\exists x(P(x) \wedge \neg Q(x))$.

47. El significado literal es: Todo no es dulzura y miel. El significado deseado es: No todo es dulzura y miel. Sea $P(x)$ la afirmación "x es dulzura y miel". En símbolos, la afirmación aclarada es $\exists x \neg P(x)$.

Sección 1.4 Repaso

1. Para toda x y para toda y, $P(x, y)$. La afirmación es verdadera si para toda x y para toda y en el dominio de discurso, $P(x, y)$ es verdadera. La afirmación es falsa si existe al menos una x y al menos una y en el dominio de discurso tales que $P(x, y)$ es falsa.

2. Para toda x, existe y tal que $P(x, y)$. La afirmación es verdadera si, para toda x en el dominio de discurso, existe al menos una y en el dominio de discurso para la cual $P(x, y)$ es verdadera. La afirmación es falsa si existe al menos una x en el dominio de discurso tal que $P(x, y)$ es falsa para toda y en el dominio de discurso.

3. Existe x tal que para toda y, $P(x, y)$. La afirmación es cierta si existe al menos una x en el dominio de discurso tal que $P(x, y)$ es verdadera para toda y en el dominio de discurso. La afirmación es falsa si, para toda x en el dominio de discurso, existe al menos una y en el dominio de discurso tal que $P(x, y)$ es falsa.

4. Existe x y existe y tal que $P(x, y)$. La afirmación es cierta si existe al menos una x y al menos una y en el dominio de discurso tales que $P(x, y)$ es verdadera. La afirmación es falsa si, para toda x y para toda y en el dominio de discurso, $P(x, y)$ es falsa.

5. Sea $P(x, y)$ la función proposicional "$x \leq y$" con los enteros como dominio de discurso. Entonces $\forall x \exists y P(x, y)$ es verdadera ya que, para todo entero x, existe un entero y (por ejemplo, $y = x$) tal que $x \leq y$ es verdadera. Por otro lado, $\exists x \forall y P(x, y)$ es falsa. Para todo entero x, existe un entero y (por ejemplo, $y = x - 1$) tal que $x \leq y$ es falsa.

6. $\exists x \exists y \neg P(x, y)$ **7.** $\exists x \forall y \neg P(x, y)$

8. $\forall x \exists y \neg P(x, y)$ **9.** $\forall x \forall y \neg P(x, y)$

10. Dada una función proposicional cuantificada, usted y su oponente, a quien llamamos Fernando, juegan el juego de lógica. Su meta es intentar hacer que la función proposicional sea cierta, y la meta de Fernando es intentar hacerla falsa. El juego comienza con el primer cuantificador (izquierda). Si el cuantificador es $\forall$, Fernando elige un valor para esa variable; si el cuantificador es $\exists$, usted escoge un valor para esa variable. El juego continúa con el segundo cuantificador. Después de elegir valores para todas las variables, si la función proposicional es verdadera, usted gana; si es falsa, Fernando gana. Si usted puede ganar siempre sin importar cómo selecciona Fernando valores para las variables, la función proposicional cuantificada es verdadera; pero si Fernando puede elegir valores para las variables de manera que usted no pueda ganar, la función proposicional cuantificada es falsa.

Sección 1.4

1. Todos son más altos que todos los demás. Falsa.

4. Alguien es más alto que alguien más. Verdadera.

5. (Para el ejercicio 1) En símbolos: $\exists x \exists y \neg T(x, y)$. Observe que en palabras $\neg T(x, y)$ es: x no es más alto que y o $x = y$. Entonces, la negación en palabras es: Alguien no es más alto que alguien más o dos personas (no necesariamente distintas) son iguales.

6. Todos son más altos o de la misma altura que todos. Falsa.

9. Alguien es más alto que o de la misma altura que alguien más. Verdadera.

10. (Para el ejercicio 6) En símbolos: $\exists x \exists y \neg T(x, y)$. Observe que en palabras, $\neg T(x, y)$ es: x es más bajo que y. Entonces, la negación en palabras es: Alguien es más bajo que alguien más.

11. $\exists x \forall y L(x, y)$. Verdadera (piense en un santo).

14. $\forall x \exists y L(x, y)$. Verdadera (según la canción de Dean Martin "Everybody Loves Somebody Sometime" o todos aman a alguien alguna vez).

15. (Para el ejercicio 11) Todos no aman a alguien. $\forall x \exists y \neg L(x, y)$

16. Falsa **19.** Verdadera

20. (Para el ejercicio 16) $\exists x \exists y \neg P(x, y)$ o $\exists x \exists y (x < y)$

21. Falsa. Un contraejemplo es $x = 2$, $y = 0$.

24. Verdadera. Tome $x = y = 0$.

27. Falsa. Un contraejemplo es $x = y = 2$.

30. Verdadera. Tome $x = 1$, $y = \sqrt{8}$.

33. Verdadera. Tome $x = 0$. Después, para toda y, $x^2 + y^2 \geq 0$.

36. Verdadera. Para cualquier x, si se hace $y = x - 1$, la proposición condicional si $x < y$, entonces $x^2 < y^2$, es cierta porque la hipótesis es falsa.

39. (Para el ejercicio 21) $\exists x \exists y (x^2 \geq y + 1)$

42. (Para el ejercicio 21) Como ambos cuantificadores son $\forall$, Fernando elige valores para ambos x y y. Como Fernando puede elegir valores que hagan $x^2 < y + 1$ falsa (por ejemplo, $x = 2$, $y = 0$), Fernando puede ganar el juego. Por lo tanto, la proposición es falsa.

45. Como los primeros dos cuantificadores son $\forall$, Fernando elige valores para ambos x y y. El último cuantificador es $\exists$, de manera que usted elige un valor para z. Fernando puede elegir valores (por ejemplo, $x = 1$, $y = 2$) de manera que no importa qué valor seleccione usted para z, la expresión

$$(x < y) \rightarrow ((z > x) \wedge (z < y))$$

es falsa. Como Fernando puede elegir valores para las variables de forma que usted no pueda ganar, la afirmación cuantificada es falsa.

47. $\forall x\, \exists y\, P(x, y)$ debe ser verdadera. Como $\forall x\, \forall y\, P(x, y)$ es verdadera, sin importar qué valor de x se seleccione, $P(x, y)$ es cierta *para toda* y. Entonces, para cualquier x, $P(x, y)$ es cierta *para cualquier* y específica.

50. $\forall x\, \forall y\, P(x, y)$ puede ser falsa. Sea $P(x, y)$ la expresión $x > y$. Si el dominio de discurso es el conjunto de los enteros, $\forall x\, \exists y\, P(x, y)$ es verdadera; sin embrago, $\forall x\, \forall y\, P(x, y)$ es falsa.

53. $\forall x\, \forall y\, P(x, y)$ puede ser falsa. Sea $P(x, y)$ la expresión $x \leq y$. Si el dominio de discurso es el conjunto de enteros positivos, $\exists x\, \forall y\, P(x, y)$ es verdadera; sin embargo, $\forall x\, \forall y\, P(x, y)$ es falsa.

56. $\forall x\, \forall y\, P(x, y)$ puede ser falsa. Sea $P(x, y)$ la expresión $x \leq y$. Si el dominio de discurso es el conjunto de enteros positivos, $\exists x\, \exists y\, P(x, y)$ es verdadera; sin embargo, $\forall x\, \forall y\, P(x, y)$ es falsa.

59. $\exists x\, \neg(\forall y\, P(x, y))$ no es un equivalente lógico de $\neg(\forall x\, \exists y\, P(x, y))$. Sea $P(x, y)$ la expresión $x < y$. Si el dominio de discurso es el conjunto de enteros, $\exists x\, \neg(\forall y\, P(x, y))$ es verdadera; sin embargo, $\neg(\forall x\, \exists y\, P(x, y))$ es falsa.

62. $\exists x\, \exists y\, \neg P(x, y)$ no es un equivalente lógico de $\neg(\forall x\, \exists y\, P(x, y))$. Sea $P(x, y)$ la expresión $x < y$. Si el domino de discurso es el conjunto de enteros, $\exists x\, \exists y\, \neg P(x, y)$ es verdadera; sin embargo, $\neg(\forall x\, \exists y\, P(x, y))$ es falsa.

Sección 1.5 Repaso

1. Un sistema matemático consiste en axiomas, definiciones y términos no definidos.

2. Un axioma es una proposición que se supone verdadera.

3. Una definición crea un nuevo concepto en términos de los existentes.

4. Un término no definido es un término que no tiene una definición explícita, sino que está definido de manera implícita por los axiomas.

5. Un teorema es una proposición que debe demostrarse que es verdadera.

6. Una demostración es un argumento que establece la verdad de un teorema.

7. Un lema es un teorema que suele no ser muy interesante por sí mismo, pero que resulta útil para demostrar otro teorema.

8. Una prueba directa supone que las hipótesis son ciertas y entonces, usando las hipótesis, otros axiomas, definiciones y teoremas demostrados antes, prueba directamente que la conclusión es verdadera.

9. Un entero n es par si existe un entero k tal que $n = 2k$.

10. Un entero n es impar si existe un entero k tal que $n = 2k + 1$.

11. Una demostración por contradicción supone que las hipótesis son verdaderas y que la conclusión es falsa y después usa la hipótesis y la negación de la conclusión, junto con otros axiomas, definiciones y teoremas demostrados antes, para obtener una contradicción.

12. "Demostración indirecta" es otro nombre para la demostración por contradicción.

13. Para probar $p \rightarrow q$, la demostración por contrapositiva prueba la afirmación equivalente $\neg q \rightarrow \neg p$.

14. En lugar de probar

$$(p_1 \vee p_2 \vee \cdots \vee p_n) \rightarrow q,$$

en la demostración por casos, se prueba

$$(p_1 \rightarrow q) \wedge (p_2 \rightarrow q) \wedge \cdots \wedge (p_n \rightarrow q).$$

15. Para probar

$$\exists x\, P(x),$$

se encuentra un miembro x en el dominio de discurso que hace a $P(x)$ verdadera.

16. El razonamiento deductivo se refiere al proceso de delinear una conclusión a partir de una secuencia de proposiciones.

17. En el argumento $p_1, p_2, \ldots, p_n/\therefore q$, las hipótesis son $p_1, p_2, \ldots, p_n$.

18. "Premisa" es otro nombre para hipótesis.

19. En el argumento $p_1, p_2, \ldots, p_n/\therefore q$, la conclusión es q.

20. El argumento $p_1, p_2, \ldots, p_n/\therefore q$ es válido siempre que si p_1 y p_2 y $\ldots$ y p_n son todas verdaderas, entonces q también es verdadera.

21. Un argumento inválido es un argumento que no es válido.

22.
$$\begin{array}{l} p \rightarrow q \\ \underline{p} \\ \therefore q \end{array}$$

23.
$$\begin{array}{l} p \rightarrow q \\ \underline{\neg q} \\ \therefore \neg p \end{array}$$

24.
$$\begin{array}{l} \underline{p} \\ \therefore p \vee q \end{array}$$

25.
$$\begin{array}{l} \underline{p \wedge q} \\ \therefore p \end{array}$$

26.
$$\begin{array}{l} p \\ \underline{q} \\ \therefore p \wedge q \end{array}$$

27.
$$\begin{array}{l} p \rightarrow q \\ \underline{q \rightarrow r} \\ \therefore p \rightarrow r \end{array}$$

28.
$$\begin{array}{l} p \vee q \\ \underline{\neg p} \\ \therefore q \end{array}$$

29.
$$\begin{array}{l} \underline{\forall x\, P(x)} \\ \therefore P(d) \text{ si } d \text{ está en } D \end{array}$$

30.
$$\begin{array}{l} \underline{P(d) \text{ para todo } d \text{ en } D} \\ \therefore \forall x\ P(x) \end{array}$$

31.
$$\begin{array}{l} \underline{\exists x\, P(x)} \\ \therefore P(d) \text{ para alguna } d \text{ en } D \end{array}$$

32.
$$\begin{array}{l} \underline{P(d) \text{ para alguna } d \text{ en } D} \\ \therefore \exists x\ P(x) \end{array}$$

Sección 1.5

1. Si tres puntos no son colineales, entonces hay exactamente un plano que los contiene.

4. Si x es un número real no negativo y n es un entero positivo, $x^{1/n}$ es el número no negativo y que satisface $y^n = x$.

7. Sean m y n enteros pares. Entonces existen k_1 y k_2 tales que $m = 2k_1$ y $n = 2k_2$. Ahora

$$m + n = 2k_1 + 2k_2 = 2(k_1 + k_2).$$

Por lo tanto, $m + n$ es par.

10. Sean m y n enteros impares. Entonces existen k_1 y k_2 tales que $m = 2k_1 + 1$ y $n = 2k_2 + 1$. Ahora

$$\begin{aligned} mn = (2k_1 + 1)(2k_2 + 1) &= 4k_1k_2 + 2k_1 + 2k_2 + 1 \\ &= 2(2k_1k_2 + k_1 + k_2) + 1. \end{aligned}$$

Por lo tanto, mn es impar.

13.
$$\begin{aligned} x \cdot 0 + 0 &= x \cdot 0 && \text{porque } b + 0 = b \text{ para todo número real } b \\ &= x \cdot (0 + 0) && \text{porque } b + 0 = b \text{ para todo número real } b \\ &= x \cdot 0 + x \cdot 0 && \text{porque } a(b + c) = ab + ac \text{ para todos los números reales } a, b, c \end{aligned}$$

Tomando $a = c = x \cdot 0$ y $b = 0$, la ecuación anterior se convierte en $a + b = a + c$; por tanto, $0 = b = c = x \cdot 0$.

16. Suponga que la conclusión es falsa, es decir, que no hay dos bolsas que contienen el mismo número de monedas. Suponga que se arreglan las bolsas en orden creciente del número de monedas que contienen. Entonces la primera bolsa contiene al menos una moneda; la segunda bolsa contiene al menos dos monedas; y así sucesivamente. Entonces el número total de monedas es al menos

$$1 + 2 + 3 + \cdots + 9 = 45.$$

Esto contradice la hipótesis de que hay 40 monedas. La prueba por contradicción queda completa.

19. La afirmación es verdadera y se demuestra usando la prueba por contradicción. Suponga que para toda j, $s_j \le A$. Como $s_j \le A$ para toda j y $s_i < A$,

$$s_1 + \cdots + s_i + \cdots + s_n < A + \cdots + A + \cdots + A = nA.$$

Al dividir entre n se obtiene

$$\frac{s_1 + \cdots + s_n}{n} < A,$$

que es una contradicción.

22. Se consideran tres casos: $x > 0$, $x = 0$ y $x < 0$.

Si $x > 0$, $|x| = x$ y $\text{sgn}(x) = 1$. Por lo tanto,

$$|x| = x = 1 \cdot x = \text{sgn}(x)x.$$

Si $x = 0$, $|x| = 0$ y $\text{sgn}(x) = 0$. Por lo tanto,

$$|x| = 0 = 0 \cdot 0 = \text{sgn}(x)x.$$

Si $x < 0$, $|x| = -x$ y $\text{sgn}(x) = -1$. Por lo tanto,

$$|x| = -x = -1 \cdot x = \text{sgn}(x)x.$$

En todos los casos se tiene $|x| = \text{sgn}(x)x$.

25. Se consideran dos casos: $x \ge y$ y $x < y$.

Si $x \ge y$,

$$\text{máx}\{x, y\} = x \quad \text{y} \quad \text{mín}\{x, y\} = y.$$

Por lo tanto,

$$\text{máx}\{x, y\} + \text{mín}\{x, y\} = x + y.$$

Si $x < y$,

$$\text{máx}\{x, y\} = y \quad \text{y} \quad \text{mín}\{x, y\} = x.$$

Por lo tanto,

$$\text{máx}\{x, y\} + \text{mín}\{x, y\} = y + x = x + y.$$

En cualquier caso,

$$\text{máx}\{x, y\} + \text{mín}\{x, y\} = x + y.$$

28.
$$\begin{aligned}\text{máx}\{x, y\} + \text{mín}\{x, y\} &= \frac{x + y + |x - y|}{2} + \frac{x + y - |x - y|}{2} \\ &= \frac{x + y + |x - y| + x + y - |x - y|}{2} \\ &= \frac{2x + 2y}{2} = x + y.\end{aligned}$$

31. Válido
$$\begin{array}{l} p \to q \\ \underline{p} \\ \therefore q \end{array}$$

34. Inválido
$$\begin{array}{l} (p \vee r) \to q \\ \underline{q} \\ \therefore \neg p \to r \end{array}$$

36. Válido. Si 4 megabytes es mejor que nada de memoria, entonces compraremos una nueva computadora. Si 4 megabytes es mejor que cero memoria, entonces compraremos más memoria. Por lo tanto, si 4 megabytes es mejor que cero memoria, entonces compraremos una nueva computadora y compraremos más memoria.

39. Inválido. Si no compramos una nueva computadora, entonces 4 megabytes no es mejor que cero memoria. Compraremos una nueva computadora. Por lo tanto, 4 megabytes es mejor que cero memoria.

41. Inválido

44. Inválido

47. Un análisis del argumento debe tomar en cuenta el hecho de que "nada" se usa de dos maneras muy diferentes.

49. Suma

53. Sea p la proposición "El auto tiene gasolina", sea q la proposición "Iré a la tienda", y sea r la proposición "Compraré un refresco". Entonces las hipótesis son:

$$\begin{array}{l} p \to q \\ q \to r \\ p \end{array}$$

De $p \to q$ y $q \to r$, se puede usar el silogismo hipotético para concluir que $p \to r$. De $p \to r$ y p, se puede usar modus ponens para concluir r. Como r representa la proposición "Compraré un refresco", se concluye que la conclusión se deriva de las hipótesis.

56. Sea $P(x)$ la función proposicional "x tiene una calculadora que grafica" y sea $Q(x)$ la función proposicional "x entiende las funciones trigonométricas". Las hipótesis son $\forall x\, P(x)$ y $\forall x(P(x) \to Q(x))$. Por ejemplificación universal, se tiene P(Rafael) y P(Rafael) $\to Q$(Rafael). La regla de inferencia modus ponens ahora da Q(Rafael), que representa la proposición "Rafael entiende las funciones trigonométricas". Se concluye que la conclusión se deriva de las hipótesis.

59. Se construye una tabla de verdad para todas las proposiciones implicadas:

p	q	$p \to q$	$\neg q$	$\neg p$
V	V	V	F	F
V	F	F	V	F
F	V	V	F	V
F	F	V	V	V

Se observa que siempre que la hipótesis $p \to q$ y $\neg q$ sean verdaderas, la conclusión $\neg p$ también es verdadera; por tanto, el argumento es válido.

62. Se construye una tabla de verdad para todas las proposiciones implicadas:

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

Se observa que siempre que las hipótesis p y q son verdaderas, la conclusión $p \wedge q$ también es verdadera; por lo tanto, el argumento es válido.

65. Por definición, la proposición $\forall x\, P(x)$ es verdadera cuando $P(x)$ es verdadera para toda x en el dominio de discurso. Se sabe que $P(d)$ es verdadera para cualquier d en el dominio de discurso D. Por lo tanto, $\forall x\, P(x)$ es verdadera.

Sección 1.6 Repaso

1. $p \vee q,\ \neg p \vee r / \therefore q \vee r$

2. Una cláusula consiste en términos separados por o, donde cada término es una variable o la negación de una variable.

3. Una demostración por resolución procede aplicando repetidas veces la regla en el ejercicio 1 a pares de afirmaciones para derivar nuevas afirmaciones hasta obtener la conclusión.

Sección 1.6

1.

p	q	r	$p \vee q$	$\neg p \vee r$	$q \vee r$
V	V	V	V	V	V
V	V	F	V	F	V
V	F	V	V	V	V
V	F	F	V	F	F
F	V	V	V	V	V
F	V	F	V	V	V
F	F	V	F	V	V
F	F	F	F	V	

2. 1. $\neg p \vee q \vee r$
2. $\neg q$
3. $\neg r$
4. $\neg p \vee r$ de 1 y 2
5. $\neg p$ de 3 y 4

5. Primero se ve que $p \to q$ es lógicamente equivalente a $\neg\, p \vee q$. Ahora se argumenta lo siguiente:

1. $\neg p \vee q$
2. $p \vee q$
3. q de 1 y 2

7. (Para el ejercicio 2)

1. $\neg p \vee q \vee r$ hipótesis
2. $\neg q$ hipótesis
3. $\neg r$ hipótesis
4. p negación de conclusión
5. $\neg p \vee r$ de 1 y 2
6. $\neg p$ de 3 y 5

Ahora 4 y 6 se combinan para dar una contradicción.

Sección 1.7 Repaso

1. Suponga que se tiene una función proposicional $S(n)$ cuyo dominio de discurso es el conjunto de enteros positivos. Suponga que $S(1)$ es verdadera y, para todo $n \geq 1$, si $S(n)$ es verdadera, entonces $S(n + 1)$ es verdadera. Entonces $S(n)$ es verdadera para todo entero positivo n.

2. Primero se verifica que $S(1)$ es verdadera (paso base). Después se supone que $S(n)$ es verdadera para probar que $S(n + 1)$ es verdadera (paso inductivo).

3. $\dfrac{n(n+1)}{2}$

4. La suma geométrica es la suma

$$a + ar^1 + ar^2 + \cdots + ar^n.$$

Es igual a

$$\frac{a(r^{n+1} - 1)}{r - 1}.$$

Sección 1.7

1. **Paso base** $1 = 1^2$

Paso inductivo Suponga verdad para n.

$$1 + \cdots + (2n - 1) + (2n + 1) = n^2 + 2n + 1 = (n + 1)^2$$

4. **Paso base** $1^2 = (1 \cdot 2 \cdot 3)/6$

Paso inductivo Suponga verdad para n.

$$1^2 + \cdots + n^2 + (n + 1)^2 = \frac{n(n+1)(2n+1)}{6} + (n + 1)^2 = \frac{(n+1)(n+2)(2n+3)}{6}$$

7. **Paso base** $1/(1 \cdot 3) = 1/3$

Paso inductivo Suponga verdad para n.

$$\frac{1}{1 \cdot 3} + \cdots + \frac{1}{(2n-1)(2n+1)} + \frac{1}{(2n+1)(2n+3)} = \frac{n}{2n+1} + \frac{1}{(2n+1)(2n+3)} = \frac{n+1}{2n+3}$$

10. **Paso base** $\cos x = \dfrac{\cos[(x/2) \cdot 2]\,\text{sen}(x/2)}{\text{sen}(x/2)}$

Paso inductivo Suponga verdad para n. Entonces

$$\cos x + \cdots + \cos nx + \cos(n + 1)x = \frac{\cos[(x/2)(n+1)]\,\text{sen}(nx/2)}{\text{sen}(x/2)} + \cos(n + 1)x. \quad (*)$$

Debe demostrarse que el lado derecho de $(*)$ es igual a

$$\frac{\cos[(x/2)(n+2)]\,\text{sen}[(n+1)x/2]}{\text{sen}(x/2)}.$$

Esto es lo mismo que demostrar que [después de multiplicar por el término $\text{sen}(x/2)$]

$$\cos\left[\frac{x}{2}(n+1)\right]\text{sen}\frac{nx}{2} + \cos(n+1)x\,\text{sen}\frac{x}{2} = \cos\left[\frac{x}{2}(n+2)\right]\text{sen}\left[\frac{(n+1)x}{2}\right].$$

Si $\alpha = (x/2)(n + 1)$ y $\beta = x/2$, debe demostrarse que

$$\cos\alpha\,\text{sen}(\alpha - \beta) + \cos 2\alpha\,\text{sen}\,\beta = \cos(\alpha + \beta)\,\text{sen}\,\alpha.$$

Esta última ecuación se puede verificar reduciendo cada lado a términos que implican α y β.

12. **Paso base** $1/2 \leq 1/2$

Paso inductivo Suponga verdad para n.

$$\frac{1 \cdot 3 \cdot 5 \cdots (2n-1)(2n+1)}{2 \cdot 4 \cdot 6 \cdots (2n)(2n+2)} \geq \frac{1}{2n} \cdot \frac{2n+1}{2n+2} = \frac{2n+1}{2n} \cdot \frac{1}{2n+2} \geq \frac{1}{2n+2}$$

15. **Paso base** $(n = 4)\quad 2^4 = 16 \geq 16 = 4^2$

Paso inductivo Suponga verdad para n.

$$(n + 1)^2 = n^2 + 2n + 1 \leq 2^n + 2n + 1 \leq 2^n + 2^n \quad \text{por el ejercicio 14} = 2^{n+1}$$

18. $r^0 + r^1 + \cdots r^n = \dfrac{1 - r^{n+1}}{1 - r} < \dfrac{1}{1 - r}$

21. **Paso base** $7^1 - 1 = 6$ es divisible entre 8.

Paso inductivo Suponga que 8 divide $7^n - 1$. Ahora

$$7^{n+1} - 1 = 7 \cdot 7^n - 1 = 7^n - 1 + 6 \cdot 7^n.$$

Como 6 divide a los dos, $7^n - 1$ y $6 \cdot 7^n$, divide a su suma, que es $7^{n+1} - 1$.

24. **Paso base** $3^1 + 7^1 - 2 = 8$ es divisible ente 8.

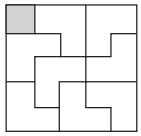
Paso inductivo Suponga que 8 divide a $3^n + 7^n - 2$. Ahora

$$3^{n+1} + 7^{n+1} - 2 = 3(3^n + 7^n - 2) + 4(7^n + 1).$$

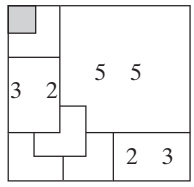
Por la suposición inductiva, 8 divide a $3^n + 7^n - 2$. Se puede usar inducción matemática para demostrar que 2 divide a $7^n + 1$ para toda $n \geq 1$ (el argumento es similar al dado en la sugerencia para el ejercicio 21). Después se deriva que 8 divide a $4(7^n + 1)$. Como 8 divide tanto a $3(3^n + 7^n - 2)$ como a $4(7^n + 1)$, también divide a su suma, que es $3^{n+1} + 7^{n+1} - 2$.

26. En el paso inductivo cuando la línea $(n + 1)$ se agrega, debido a las suposiciones, la línea cruzará cada una de las otras n líneas. Ahora, imagine recorrer la línea $(n + 1)$. Cada vez que pasa por una de las regiones originales, se divide en dos regiones.

29.



32. Se denota el cuadrado en el renglón i, columna j por (i, j). Entonces, por simetría, deben considerarse sólo tableros de 7×7 con cuadrados (i, j) eliminados donde $i \leq j \leq 4$. La solución cuando el cuadrado (1, 1) se elimina se muestra en la siguiente figura.



No se muestran todos los trominos del enlosado. Por el ejercicio 31, los subtableros de 3×2 tienen enlosados. Por el ejercicio 29, el subtablero de 5×5 con un cuadrado en la esquina eliminado tiene un enlosado. En esencia, la misma figura da los enlosados si el cuadrado (1, 2) o (2, 2) se elimina.

Un argumento similar da los enlosados para los casos restantes.

35. **Paso base** $(n = 1)$ El tablero es un tromino.

Paso inductivo Suponga que cualquier tablero deficiente de $2^n \times 2^n$ se puede enlosar con trominos. Se debe probar que cualquier tablero deficiente de $2^{n+1} \times 2^{n+1}$ se puede enlosar con trominos.

Dado un tablero deficiente de $2^{n+1} \times 2^{n+1}$, se divide el tablero en cuatro subtableros de $2^n \times 2^n$ como se muestra en la figura 1.7.5. Por la suposición inductiva, se puede enlosar el subtablero que contiene el cuadrado que falta. Los tres subtableros restantes forman una L de $2^n \times 2^n$, que se puede enlosar usando el ejercicio 34. Entonces, el tablero deficiente de $2^{n+1} \times 2^{n+1}$ se enlosa. El paso inductivo queda completo.

36. Numere los cuadros como se muestra:

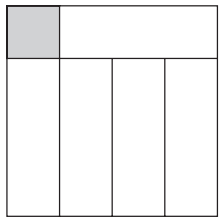
1	2	3	1
2	3	1	2
3	1	2	3
1	2	3	1

Observe que cada tromino cubre exactamente un 1, un 2 y un 3. Por lo tanto, si existe un enlosado, se cubren cinco números 2. Como se requieren cinco trominos, el cuadro que falta no puede ser 2. De manera similar, el cuadro faltante no puede ser 3.

El mismo argumento aplicado a

1	3	2	1
2	1	3	2
3	2	1	3
1	3	2	1

muestra que la única posibilidad para el cuadro que falta sea una esquina. Este tablero se puede enlosar:



39. Se prueba que $pow = a^{i-1}$ es un ciclo invariante para el ciclo "while". Justo antes de que el ciclo "while" inicie su ejecución , $i = 1$ y $pow = 1$, de manera que $pow = a^{1-1}$. Se ha probado el paso base.

Suponga que $pow = a^{i-1}$. Si $i \leq n$ (de forma que el cuerpo del ciclo se ejecuta otra vez), *pow* se convierte en

$$pow * a = a^{i-1} * a = a^i,$$

e i se convierte en $i + 1$. Se ha probado el paso inductivo. Por lo tanto, $pow = a^{i-1}$ es invariante para el ciclo "while".

El ciclo "while" termina cuando $i = n + 1$. Como $pow = a^{i-1}$ es invariante, en este punto $pow = a^n$.

43. *a)* $S_1 = 0 \neq 2$;

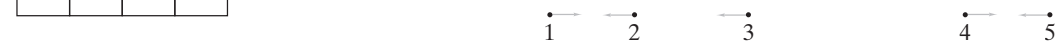
$$\begin{aligned} 2 + \cdots + 2n + 2(n + 1) &= S_n + 2n + 2 \\ &= (n + 2)(n - 1) + 2n + 2 \\ &= (n + 3)n = S_{n+1}. \end{aligned}$$

b) Debe tenerse $S'_n = S'_{n-1} + 2n$; entonces

$$\begin{aligned} S'_n &= S'_{n-1} + 2n \\ &= [S'_{n-2} + 2(n - 1)] + 2n \\ &= S'_{n-2} + 2n + 2(n - 1) \\ &= S'_{n-3} + 2n + 2(n - 1) + 2(n - 2) \\ &\vdots \\ &= S'_1 + 2[n + (n - 1) + \cdots + 2] \\ &= C' + 2\left[\frac{n(n + 1)}{2} - 1\right] \\ &= n^2 + n + C. \end{aligned}$$

47. Si $n = 2$, cada persona lanza un pastel a la otra y no hay sobrevivientes.

50. La afirmación es falsa. En



1 y 5 son los más lejanos, pero ninguno es sobreviviente.

52. Sean x y y puntos en $X \cap Y$. Entonces x está en X y y está en X. Como X es convexo, el segmento de recta de x a y está en X. De manera similar, el segmento de recta de x a y está en Y. Por lo tanto, el segmento de recta de x a y está en $X \cap Y$. Así, $X \cap Y$ es convexa.

55. $x_1, \ldots, x_n$ denota los n puntos y X_i el círculo de radio 1 centrado en x_i. Aplique el teorema de Helly a $X_1, \ldots, X_n$.

57. 1

60. **Paso base** $(i = 1)$ Como se elimina 2, 1 sobrevive. Entonces $J(2) = 1$.

Paso inductivo Suponga verdad para i. Ahora suponga que 2^{i+1} personas están colocadas en círculo. Se comienza por eliminar 2, 4, 6, . . . , 2^{i+1}. Entonces se tienen 2^i personas colocadas en círculo y, comenzando con 1, se elimina la segunda persona, después la cuarta, y así sucesivamente. Por la suposición inductiva, 1 sobrevive. Por lo tanto,

$$J(2^{i+1}) = J(2^i) = 1.$$

63. La potencia más grande de 2 menor o igual que 100,000 es 2^{16}. Entonces, en la notación del ejercicio 61, $n = 100{,}000$, $i = 16$ y

$$j = n - 2^i = 100{,}000 - 2^{16} = 100{,}000 - 65{,}536 = 34{,}464.$$

Por el ejercicio 61,

$$J(100{,}000) = J(n) = 2j + 1 = 2 \cdot 34{,}464 + 1 = 68{,}929.$$

64.

$$\begin{aligned} b_1 + b_2 + \cdots + b_n &= (a_2 - a_1) + (a_3 - a_2) \\ &\quad + \cdots + (a_{n+1} - a_n) \\ &= -a_1 + a_{n+1} = a_{n+1} - a_1 \end{aligned}$$

ya que $a_1, \ldots, a_n$ se cancelan.

67. Sea

$$a_n = \frac{1}{n}.$$

Entonces

$$\Delta a_n = a_{n+1} - a_n = \frac{1}{n+1} - \frac{1}{n} = \frac{-1}{n(n+1)}.$$

Sea $b_n = \Delta a_n$. Por el ejercicio 64,

$$\begin{aligned} \frac{-1}{1 \cdot 2} + \cdots + \frac{-1}{n(n+1)} &= \Delta a_1 + \cdots + \Delta a_n \\ &= b_1 + \cdots + b_n \\ &= a_{n+1} - a_1 \\ &= \frac{1}{n+1} - 1 = \frac{-n}{n+1}. \end{aligned}$$

Multiplicando por -1 se llega a la fórmula deseada.

Sección 1.8 Repaso

1. Suponga que se tiene una función proposicional $S(n)$ cuyo dominio de discurso es el conjunto de enteros mayores o iguales que n_0. Suponga que $S(n_0)$ es verdadera; y para toda $n > n_0$, si $S(k)$ es verdadera para toda k, $n_0 \le k < n$, entonces $S(n)$ es verdadera. Por lo tanto, $S(n)$ es verdadera para todo entero $n \ge n_0$.

2. Todo conjunto no vacío de enteros no negativos tiene al menos un elemento.

3. Si d y n son enteros, $d > 0$, existen enteros únicos q (cociente) y r (residuo) que satisfacen $n = dq + r,\ 0 \le r < d$.

Sección 1.8

1. **Paso base** $(n = 6, 7)$ Se puede reunir 6 centavos de importe postal usando tres timbres de 2 centavos. Se puede reunir 7 centavos de importe usando un timbre de 7 centavos.

Paso inductivo Se supone que $n \ge 8$ y se puede lograr un importe de k centavos o más usando sólo timbres de 2 y 7 centavos para $6 \le k < n$. Por la suposición inductiva, se puede lograr un importe de $n - 2$ centavos. Al agregar un timbre de 2 centavos se tiene el importe de n centavos.

3. **Paso base** $(n = 4)$ Se puede reunir un importe postal de 4 centavos usando timbres de 2 centavos.

Paso inductivo Se supone que se puede lograr un importe de n centavos y se prueba que se puede reunir un importe de $n + 1$ centavos.

Si entre los timbres que forman el importe de n centavos hay al menos un timbre de 5 centavos, se sustituye un timbre de 5 centavos por tres de 2 centavos para tener $n + 1$ centavos. Si no hay timbres de 5 centavos entre los que forman el importe de n centavos, hay al menos dos timbres de 2 centavos (porque $n \ge 4$). Se sustituyen los dos timbres de 2 centavos por uno de 5 centavos para reunir $n + 1$ centavos de importe postal.

6. $c_2 = 4, c_3 = 9, c_4 = 20, c_5 = 29$

8. $c_2 = 2, c_3 = 3, c_4 = 12, c_5 = 13$

11. **Paso base** $(n = 4)$ El primer jugador remueve una carta (de cualquier pila). Luego el segundo jugador quita la última carta y gana el juego.

Paso inductivo Suponga que $n > 1$ y siempre que haya dos pilas de $k > n$ cartas, el segundo jugador siempre puede ganar el juego.

Suponga que hay dos pilas de n cartas. El primer jugador retira i cartas de una de las pilas. Si $i = n$ (es decir, el primer jugador retira todas las cartas de una pila), el segundo jugador puede ganar removiendo todas las cartas de la pila que queda. Si $i < n$, el segundo jugador retira i cartas de la otra pila, dejando dos pilas cada una con $n - i$ cartas. El juego entonces prosigue con el primer jugador ante dos pilas, cada una con $k = n - i < n$ cartas. Por la suposición inductiva, el segundo jugador puede ganar el juego. La prueba inductiva queda completa.

12. $q = 5,\ r = 2$

15. $q = -1,\ r = 2$

18. $\dfrac{5}{6} = \dfrac{1}{2} + \dfrac{1}{3} = \dfrac{1}{2} + \dfrac{1}{4} + \dfrac{1}{12}$

22. Se puede suponer que $p/q > 1$. Se elige el entero mayor n que satisface

$$\frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n} \le \frac{p}{q}.$$

(El rincón de solución de problemas anterior muestra que la suma

$$\frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n}$$

es no acotada; de manera que existe tal n.) Si se obtiene una desigualdad, p/q está en forma egipcia, entonces suponga que

$$\frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n} < \frac{p}{q}. \qquad (*)$$

Defina

$$D = \frac{p}{q} - \left(\frac{1}{1} + \cdots + \frac{1}{n} \right).$$

Es claro que $D > 0$. Como n es el entero más grande que satisface $(*)$,

$$\frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n} + \frac{1}{n+1} \ge \frac{p}{q}.$$

Entonces

$$\begin{aligned} D &= \frac{p}{q} - \left(\frac{1}{1} + \cdots + \frac{1}{n} \right) \\ &\le \left(\frac{1}{1} + \cdots + \frac{1}{n} + \frac{1}{n+1} \right) - \left(\frac{1}{1} + \cdots + \frac{1}{n} \right) \\ &= \frac{1}{n+1}. \end{aligned}$$

En particular, $D < 1$. Por el ejercicio 20, D debe escribirse en forma egipcia:

$$D = \frac{1}{n_1} + \cdots + \frac{1}{n_k},$$

donde las n_i son distintas. Dado que

$$\frac{1}{n_i} \leq D \leq \frac{1}{1+n}, \text{ para } i = 1, \ldots, k,$$

$n < n + 1 \leq n$ para $i = 1, \ldots, k$. Se deduce que

$$1, 2, \ldots, n, n_1, \ldots, n_k$$

son distintas. Entonces

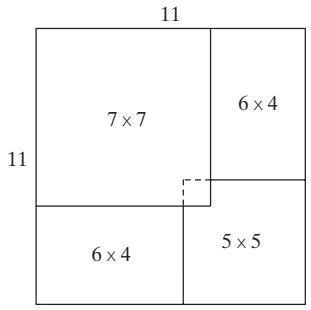
$$\frac{p}{q} = D + \frac{1}{1} + \cdots + \frac{1}{n} = \frac{1}{n_1} + \cdots + \frac{1}{n_k} + \frac{1}{1} + \cdots + \frac{1}{n}$$

está representada en forma egipcia.

23. En esta solución, la inducción es sobre el conjunto X de enteros impares $n > 5$, donde 3 divide a $n^2 - 1$. Este tipo de inducción se puede justificar al considerar que la primera afirmación se trata del entero más pequeño en X, la segunda afirmación es acerca del segundo entero más pequeño en X, etcétera.

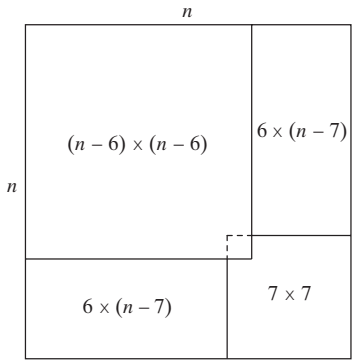
Pasos base ($n = 7$, 11) El ejercicio 32, sección 1.7, da una solución si $n = 7$.

Si $n = 11$, se encierra el cuadro que falta en la esquina del subtablero de 7×7 (vea la figura siguiente). Se enlosa este subtablero usando el resultado del ejercicio 32, sección 1.7. Se enlosan los dos subtableros de 6×4 usando el resultado del ejercicio 31, sección 1.7. Se enlosa el subtablero de 5×5 con un cuadro faltante en la esquina usando el resultado del ejercicio 29, sección 1.7. Con esto queda enlosado el tablero de 11×11.



Paso inductivo Suponga que $n > 11$ y suponga que si $k < n$, k es impar, $k > 5$, y 3 divide a $k^2 - 1$, entonces un tablero deficiente de $k \times k$ se puede enlosar con trominos.

Considere un tablero deficiente de $n \times n$. Encierre el cuadro faltante en una esquina del subtablero de $(n - 6) \times (n - 6)$. Por la suposición inductiva, este tablero se puede enlosar con trominos. Enlose los dos subtableros de $6 \times (n-7)$ usando el resultado del ejercicio 31, sección 1.7. Enlose el tablero deficiente de 7×7 usando el resultado del ejercicio 32, sección 1.7. El tablero de $n \times n$ queda enlosado, y el paso inductivo queda completo.



27. Sea X un conjunto no vacío de enteros no negativos. Se debe probar que X tiene un elemento menor. Usando inducción matemática, se demuestra que para toda $n \geq 0$, si X contiene un elemento menor o igual que n, entonces X tiene un elemento menor. Observe que esto prueba que X tiene un elemento menor. (Como X es no vacío, X contiene un entero n. Ahora bien, X contiene un elemento menor o igual que n; se deduce que X tiene un elemento menor).

Paso base ($n = 0$) Si X contiene un elemento menor o igual que 0, entonces X contiene al 0 puesto que X consta de enteros no negativos. En este caso, 0 es el elemento menor en X.

Paso inductivo Ahora se supone que si X contiene un elemento menor o igual que n, entonces X tiene un elemento menor. Debe demostrarse que si X contiene un elemento menor o igual que $n + 1$, entonces X tiene un elemento menor.

Suponga que X contiene un elemento menor o igual que $n + 1$. Se consideran dos casos: X contiene un elemento menor o igual que n, y X no contiene un elemento menor o igual que n. Si X contiene un elemento menor o igual que n, por la suposición inductiva, X tiene un elemento menor. Si X no contiene un elemento menor o igual que n, como X contiene un elemento menor o igual que $n + 1$, X debe contener a $n + 1$, que es el elemento menor en X. El paso inductivo queda completo.

Capítulo 1 Autoevaluación

1. Falsa

2.

p	q	r	$\neg(p \wedge q) \vee (p \vee \neg r)$
V	V	V	V
V	V	F	V
V	F	V	V
V	F	F	V
F	V	V	V
F	V	F	V
F	F	V	V
F	F	F	V

3. Tomo el curso de administración de hoteles y ya sea que no tome supervisión recreativa o tome cultura popular.

4. $p \vee (q \wedge \neg r)$

5. Si Luis obtiene 10 en matemáticas discretas, entonces Luis estudia duro.

6. Recíproca: Si Luis estudia duro, entonces obtiene 10 en matemáticas discretas. Contrapositiva: Si Luis no estudia duro, entonces no obtiene 10 en matemáticas discretas.

7. Verdadera

8. $(\neg r \vee q) \rightarrow \neg q$

9. La afirmación no es una proposición. El valor de verdad no se puede determinar sin saber a qué se refiere "el equipo".

10. La afirmación es una función proposicional. Cuando se sustituye un equipo en particular en la variable "equipo", la afirmación se convierte en una proposición.

11. Para todos los enteros positivos n, n y $n+2$ son primos. La proposición es falsa. Un contraejemplo es $n = 7$.

12. Para algún entero positivo n, n y $n+2$ son primos. La proposición es verdadera. Por ejemplo, si $n = 5$, n y $n+2$ son primos.

13. $\exists x \forall y \neg K(x, y)$

14. $\forall x \exists y K(x, y)$; todos conocen a alguien.

15. La afirmación es verdadera. Para toda x, existe y, a saber la *raíz cúbica* de x, tal que $x = y^3$. En palabras: Todo número real tiene una raíz cúbica.

16.
$$\begin{aligned}\neg(\forall x \exists y \forall z P(x, y, z)) &\equiv \exists x \neg(\exists y \forall z P(x, y, z)) \\ &\equiv \exists x \forall y \neg(\forall z P(x, y, z)) \\ &\equiv \exists x \forall y \exists z \neg P(x, y, z)\end{aligned}$$

17. Suponga que si cuatro equipos juegan siete juegos, ningún par de equipos juega al menos dos veces; o, de manera equivalente, si cuatro equipos juegan siete juegos, cada par de equipos juega cuando mucho una vez. Si los equipos son A, B, C y D, y cada par de equipos juega cuando mucho una vez, el número mayor de juegos que se pueden jugar es:

A y B; A y C; A y D; B y C; B y D; C y D.

Entonces se pueden jugar cuando mucho seis juegos. Ésta es una contradicción. Por lo tanto, si cuatro equipos juegan siete juegos, algún par de equipos juega al menos dos veces.

18. Los axiomas son afirmaciones que se suponen verdaderas. Las definiciones se usan para crear nuevos conceptos en términos de los existentes.

19. En una prueba directa, no se supone la negación de la conclusión, mientras que en una prueba por contradicción, se supone la negación de la conclusión.

20. El argumento es inválido. Si p y r son verdaderas y q es falsa, las hipótesis son verdaderas, pero la conclusión es falsa.

21.
$$\begin{aligned}(p \vee q) \to r &\equiv \neg(p \vee q) \vee r \\ &\equiv \neg p \neg q \vee r \\ &\equiv (\neg p \vee r)(\neg q \vee r)\end{aligned}$$

22.
$$\begin{aligned}(p \vee \neg q) \to \neg r s &\equiv \neg(p \vee \neg q) \vee \neg r s \\ &\equiv \neg p q \vee \neg r s \\ &\equiv (\neg p \vee \neg r)(\neg p \vee s)(q \vee \neg r)(q \vee s)\end{aligned}$$

23.
1. $\neg p \vee q$
2. $\neg q \vee \neg r$
3. $p \vee \neg r$
4. $\neg p \vee \neg r$ de 1 y 2
5. $\neg r$ de 3 y 4

24.
1. $\neg p \vee q$
2. $\neg q \vee \neg r$
3. $p \vee \neg r$
4. r negación de la conclusión
5. $\neg p \vee \neg r$ de 1 y 2
6. $\neg r$ de 3 y 5

Ahora 4 y 6 dan una contradicción.

En los ejercicios 25 al 28, sólo se da el paso inductivo.

25. $2 + 4 + \cdots + 2n + 2(n+1) = n(n+1) + 2(n+1) = (n+1)(n+2)$

26.
$$\begin{aligned}2^2 + 4^2 + \cdots + (2n)^2 + [2(n+1)]^2 &= \frac{2n(n+1)(2n+1)}{3} \\ + [2(n+1)]^2 &= \frac{2(n+1)(n+2)[2(n+1)+1]}{3}\end{aligned}$$

27.
$$\begin{aligned}&\frac{1}{2!} + \frac{2}{3!} + \cdots + \frac{n}{(n+1)!} + \frac{n+1}{(n+2)!} \\ &\quad = 1 - \frac{1}{(n+1)!} + \frac{n+1}{(n+2)!} = 1 - \frac{1}{(n+2)!}\end{aligned}$$

28.
$$\begin{aligned}2^{n+2} &= 2 \cdot 2^{n+1} < 2[1 + (n+1)2^n] = 2 + (n+1)2^{n+1} \\ &= 1 + [1 + (n+1)2^{n+1}] \\ &< 1 + [2^{n+1} + (n+1)2^{n+1}] \\ &= 1 + (n+2)2^{n+1}\end{aligned}$$

29. $q = 9, r = 2$

30. $c_2 = 2, c_3 = 3, c_4 = 8, c_5 = 9$

31. **Paso base** $(n = 1)$ $c_1 = 0 \leq 0 = 1 \lg 1$

Paso inductivo
$$\begin{aligned}c_n &= 2c_{\lfloor n/2 \rfloor} + n \\ &\leq 2\lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor + n \\ &\leq 2(n/2) \lg(n/2) + n \\ &= n(\lg n - 1) + n = n \lg n\end{aligned}$$

32. Sea X un conjunto no vacío de enteros no negativos que tiene una cota superior. Debe demostrarse que X contiene un elemento mayor.

Sea Y el conjunto de cotas superiores enteras para X. Por suposición, Y es no vacío. Como X consiste en enteros no negativos, Y también consiste en enteros no negativos. Por la propiedad del buen orden, Y tiene al menos un elemento, digamos n. Como Y consiste en cotas superiores para X, $k > n$ para toda k en X. Suponga, a manera de contradicción, que n no está en X. Entonces $k \leq n - 1$ para toda k en X. Así, $n - 1$ es una cota superior de X, lo cual es una contradicción. Por lo tanto, n está en X. Como $k \leq n$ para toda k en X, n es el elemento mayor en X.

Sección 2.1 Repaso

1. Un conjunto es una colección de objetos.

2. Un conjunto se puede definir listando los elementos en él. Por ejemplo $\{1, 2, 3, 4\}$ es el conjunto que consiste en los enteros 1, 2, 3, 4. Un conjunto también se puede definir dando las propiedades necesarias para sus integrantes. Por ejemplo,

$$\{x \mid x \text{ es número real, positivo}\}$$

define el conjunto que consiste en los números reales positivos.

3. El número de elementos en X.

4. $x \in X$ **5.** $x \notin X$ **6.** $\emptyset$

7. $X = Y$ si X y Y tienen los mismos elementos.

8. $X \subseteq Y$ si todo elemento de X es un elemento de Y.

9. X es un subconjunto propio de Y, denotado $X \subset Y$, si $X \subseteq Y$ y $X \neq Y$.

10. El conjunto potencia de X es la colección de todos los subconjuntos de X. Se denota por $\mathcal{P}(X)$.

11. 2^n

12. *X unión Y* es el conjunto de elementos que pertenecen a X o a Y o a ambos. Se denota por $X \cup Y$.

13. La unión de $\mathcal{S}$ es el conjunto de elementos que pertenecen al menos a un conjunto en $\mathcal{S}$. Se denota por $\cup \mathcal{S}$.

14. *X intersección Y* es el conjunto de elementos que pertenecen a X y a Y. Se denota por $X \cap Y$.

15. La intersección de $\mathcal{S}$ es el conjunto de elementos que pertenecen a todos los conjuntos en $\mathcal{S}$. Se denota $\cap \mathcal{S}$.

16. $X \cap Y = \emptyset$

17. Una colección de conjuntos $\mathcal{S}$ es ajena por pares si siempre que X y Y son conjuntos distintos en $\mathcal{S}$, X y Y son ajenos.

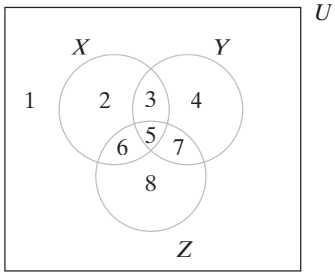
18. La diferencia de X y Y es el conjunto de elementos que están en X pero no en Y. Se denota por $X - Y$.

19. Un conjunto universal es un conjunto que contiene todos los conjuntos bajo estudio.

20. El complemento de X es $U - X$, donde U es el conjunto universal. El complemento de X se denota por $\overline{X}$

21. Un diagrama de Venn proporciona un panorama pictórico de los conjuntos. En un diagrama de Venn, un rectángulo describe un conjunto universal y los subconjuntos del conjunto universal se dibujan como círculos. El interior de un círculo representa los miembros del conjunto.

22.



La región 1 representa elementos que no están en X, Y o Z. La región 2 representa elementos en X, pero no en Y o Z. La región 3 representa elementos en X y Y pero no en Z. La región 4 representa elementos en Y, pero no en X o Z. La región 5 representa elementos en X, Y y Z. La región 6 representa elementos en X y Z, pero no en Y. La región 7 representa elementos en Y y Z, pero no en X. La región 8 representa elementos en Z, pero no en X o Y.

23. $(A \cup B) \cup C = A \cup (B \cup C),\ (A \cap B) \cap C = A \cap (B \cap C)$

24. $A \cup B = B \cup A,\ A \cap B = B \cap A$

25. $A \cap (B \cup C) = (A \cap B) \cup (A \cap C), A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

26. $A \cup \emptyset = A,\ A \cap U = A$

27. $A \cup \overline{A} = U,\ A \cap \overline{A} = \emptyset$

28. $A \cup A = A,\ A \cap A = A$

29. $A \cup U = U,\ A \cap \emptyset = \emptyset$

30. $A \cup (A \cap B) = A,\ A \cap (A \cup B) = A$

31. $\overline{\overline{A}} = A$ **32.** $\overline{\emptyset} = U,\ \overline{U} = \emptyset$

33. $\overline{(A \cup B)} = \overline{A} \cap \overline{B}, \overline{(A \cap B)} = \overline{A} \cup \overline{B}$

34. Una colección $\mathcal{S}$ de subconjuntos no vacíos de X es una partición de X si cada elemento de X pertenece exactamente a un miembro de $\mathcal{S}$.

35. El producto cartesiano de X y Y es el conjunto de todos los pares ordenados (x, y) donde $x \in X$ y $y \in Y$. Se denota por $X \times Y$.

36. El producto cartesiano de $X_1, X_2, \ldots, X_n$ es el conjunto de todas las n-eadas $(x_1, x_2, \ldots, x_n)$ donde $x_i \in X_i$ para $i = 1, \ldots, n$. Se denota por $X_1 \times X_2 \times \cdots \times X_n$.

Sección 2.1

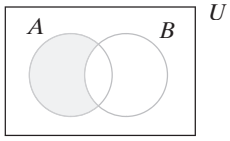
1. $\{1, 2, 3, 4, 5, 7, 10\}$ **4.** $\{2, 3, 5\}$

7. $\emptyset$ **10.** U

13. $\{6, 8\}$

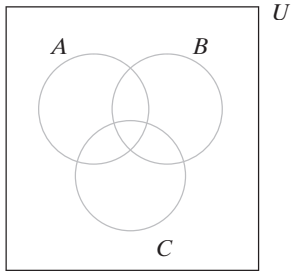
16. $\{1, 2, 3, 4, 5, 7, 10\}$

17.



20. Igual que en el ejercicio 17.

23.



25. 10 **28.** 54 **30.** 4

32. $\{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c)\}$

35. $\{(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (c, a), (c, b), (c, c)\}$

36. $\{(1, a, \alpha), (1, a, \beta), (2, a, \alpha), (2, a, \beta)\}$

39. $\{(a, 1, a, \alpha), (a, 2, a, \alpha), (a, 1, a, \beta), (a, 2, a, \beta)\}$

40. $\{\{1\}\}$

43. $\{\{a, b, c, d\}\}, \{\{a, b, c\}, \{d\}\},$
$\{\{a, b, d\}, \{c\}\}, \{\{a, c, d\}, \{b\}\}, \{\{b, c, d\}, \{a\}\},$
$\{\{a, b\}, \{c\}, \{d\}\}, \{\{a, c\}, \{b\}, \{d\}\},$
$\{\{a, d\}, \{b\}, \{c\}\},$
$\{\{b, c\}, \{a\}, \{d\}\}, \{\{b, d\}, \{a\}, \{c\}\}, \{\{c, d\}, \{a\}, \{b\}\},$
$\{\{a, b\}, \{c, d\}\}, \{\{a, c\}, \{b, d\}\}, \{\{a, d\}, \{b, c\}\},$
$\{\{a\}, \{b\}, \{c\}, \{d\}\}$

44. Verdadero **47.** Verdadero

48. Igual **51.** Igual

53. $\emptyset, \{a\}, \{b\}, \{a, b\}$. Todos menos $\{a, b\}$ son subconjuntos propios.

56. $2^n - 1$

58. Falso. $X = \{1, 2\}, Y = \{2, 3\}$.

61. Falso. $X = \{1, 2, 3\}, Y = \{2\}, Z = \{3\}$.

64. Falso. $X = \{1\}, Y = \{1, 2\}$.

67. Verdadero **70.** Verdadero

71. $A \subseteq B$ **74.** $B \subseteq A$

75. $\{1, 4, 5\}$

79. $|A| + |B|$ cuenta los elementos en A y B pero cuenta dos veces los elementos en $A \cap B$.

82. P es el conjunto de primos.

96. Se demuestra la aseveración usando inducción sobre n. El paso base es $n = 1$. En este caso, existe un subconjunto de $\{1\}$ con un número par de elementos, a saber, $\emptyset$. Como $2^{n-1} = 2^0 = 1$, la aseveración es verdadera cuando $n = 1$.

Suponga que el número de subconjuntos de $\{1, \ldots, n\}$ que contienen un número par de elementos es 2^{n-1}. Debe probarse que el número de subconjuntos de $\{1, \ldots, n+1\}$ que contienen un número par de elementos es 2^n.

Sean $E_1, \ldots, E_{2^{n-1}}$ los subconjuntos de $\{1, 2, \ldots, n\}$ que contienen un número par de elementos. Como hay 2^n subconjuntos de $\{1, 2, \ldots, n\}$ en total y 2^{n-1} contienen un número par de elementos, existen $2^n - 2^{n-1} = 2^{n-1}$ subconjuntos de $\{1, \ldots, n\}$ que contienen un número impar de elementos. Denote éstos por $O_1, \ldots, O_{2^{n-1}}$. Ahora $E_1, \ldots, E_{2^{n-1}}$ son subconjuntos de $\{1, \ldots, n+1\}$ que contienen un número par de elementos que no contienen $n+1$, y

$$O_1 \cup \{n+1\}, \ldots, O_{2^{n-1}} \cup \{n+1\}$$

son subconjuntos de $\{1, \ldots, n+1\}$ que contienen un número par de elementos que contienen $n+1$. Entonces hay $2^{n-1} + 2^{n-1} = 2^n$ subconjuntos de $\{1, \ldots, n+1\}$ que contienen un número par de elementos. El paso inductivo queda completo.

Sección 2.2 Repaso

1. Sean X y Y conjuntos. Una *función* f de X en Y es un subconjunto del producto cartesiano $X \times Y$ que tiene la propiedad de que para cada $x \in X$, existe exactamente una $y \in Y$ tal que $(x, y) \in f$.
2. En un diagrama de flechas de la función f, hay una flecha de i a j si $(i, j) \in f$.
3. La gráfica de una función f, cuyo dominio y rango son subconjuntos de los números reales, consiste en los puntos en el plano que corresponden a los elementos en f.
4. Un conjunto S de puntos en el plano define una función cuando cada línea vertical intersecta cuando mucho un puntos de S.
5. El residuo cuando x se divide entre y.
6. Una función de dispersión (*hashing*) toma un dato que debe almacenarse o recuperarse y calcula la primera opción de localización para el dato.
7. Una colisión ocurre para una función de dispersión H si $H(x) = H(y)$ pero $x \neq y$.
8. Cuando ocurre una colisión, la política de resolución de colisiones determina una localización alternativa para uno de los datos.
9. Los números seudoaleatorios son números que parecen aleatorios aunque están generados por un programa.
10. Un generador de números aleatorios congruencial lineal usa una fórmula del tipo

$$x_n = (ax_{n-1} + c) \bmod m.$$

Dado el número seudoaleatorio x_{n-1}, el siguiente número seudoaleatorio x_n está dado por la fórmula. Una "semilla" se usa como el primer número seudoaleatorio en la sucesión. Como ejemplo, la fórmula

$$x_n = (7x_{n-1} + 5) \bmod 11$$

con semilla 3 da una sucesión que comienza 3, 4, 0, 5, ...

11. El piso de x es el entero más grande menor o igual que x. Se denota por $\lfloor x \rfloor$.
12. El techo de x es el entero más pequeño mayor o igual que x. Se denota por $\lceil x \rceil$.
13. Se dice que una función f de X a Y es uno a uno si para cada $y \in Y$ existe cuando mucho una $x \in X$ tal que $f(x) = y$. La función $\{(a, 1), (b, 3), (c, 0)\}$ es uno a uno. Si una función de X a Y es uno a uno, cada elemento de Y en su diagrama de flechas tendrá a lo más una flecha que le apunta.
14. Se dice que una función f de X a Y es sobre Y si el rango de f es Y. La función $\{(a, 1), (b, 3), (c, 0)\}$ es sobre $\{0, 1, 3\}$. Si una función de X a Y es sobre Y, cada elemento de Y en su diagrama de flechas tendrá al menos una flecha que le apunta.
15. Una biyección es una función que es uno a uno y sobre. La función de los ejercicios 13 y 14 es uno a uno y sobre $\{0, 1, 3\}$.
16. Si f es una función uno a uno sobre de X a Y, la función inversa es

$$f^{-1} = \{(y, x) \mid (x, y) \in f\}.$$

Si f es la función de los ejercicios 13 y 14, se tiene

$$f^{-1} = \{(1, a), (3, b), (0, c)\}.$$

Dado el diagrama de flechas para una función f uno a uno sobre de X a Y, se puede obtener el diagrama de flechas para f^{-1} invirtiendo la dirección de cada flecha.

17. Suponga que g es una función de X a Y y f es una función de Y a Z. La composición de funciones de X a Z se define como

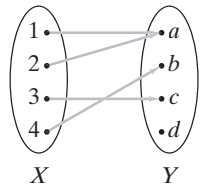
$$f \circ g = \{(x, z) \mid (x, y) \in g \quad \text{y} \quad (y, z) \in f \text{ para alguna } y \in Y\}.$$

Si $g = \{(1, 2), (2, 2)\}$ y $f = \{(2, a)\}$, $f \circ g = \{(1, a), (2, a)\}$. Dados los diagramas de flechas para las funciones g de X a Y y f de Y a Z, se puede obtener el diagrama de flechas de $f \circ g$ dibujando una flecha de $x \in X$ a $z \in Z$ siempre que existen flechas de x a alguna $y \in Y$ y de y a z.

18. Un operador binario sobre X es una función de $X \times X$ a X. El operador suma $+$ es un operador binario en el conjunto de enteros.
19. Un operador unitario en X es una función de X a X. El operador menos $-$ es un operador unitario en el conjunto de enteros.

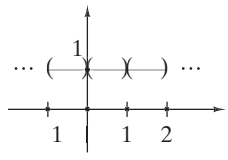
Sección 2.2

1. Es una función de X a Y; dominio $= X$, rango $= \{a, b, c\}$; no es uno a uno ni sobre. Su diagrama de flechas es

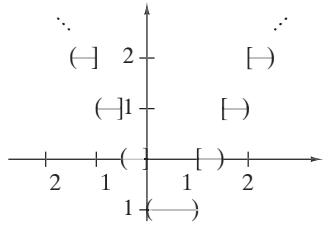


4. No es una función (de X a Y).

6.



9.



10. f es uno a uno y sobre.
13. f es uno a uno y sobre.

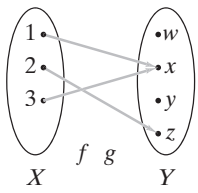
16. Defina una función f de $\{1, 2, 3, 4\}$ a $\{a, b, c, d, e\}$ como

$$f = \{(1, a), (2, c), (3, b), (4, d)\}.$$

Entonces f es uno a uno, pero no sobre.

19. $f^{-1}(y) = (y - 2)/4$ **22.** $f^{-1}(y) = 1/(y - 3)$

25. $f \circ g = \{(1, x), (2, z), (3, x)\}$



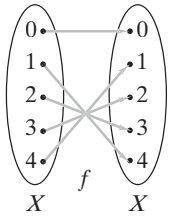
28. $(f \circ f)(x) = 2\lfloor 2x \rfloor$, $(g \circ g)(x) = x^4$, $(f \circ g)(x) = \lfloor 2x^2 \rfloor$, $(g \circ f)(x) = \lfloor 2x \rfloor^2$

29. Sea $g(x) = \log_2 x$ y $h(x) = x^2 + 2$. Entonces $f(x) = (g \circ h)(x)$.

32. Sea $g(x) = 2x$ y $h(x) = \operatorname{sen} x$. Entonces $f(x) = (g \circ h)(x)$.

35. $f = \{(-5, 25), (-4, 16), (-3, 9), (-2, 4), (-1, 1), (0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25)\}$. f no es uno a uno ni sobre. Se omite el diagrama de flechas de f.

38. $f = \{(0, 0), (1, 4), (2, 3), (3, 2), (4, 1)\}$; f es uno a uno y sobre. El diagrama de flechas de f es



41. 4

En la solución de los ejercicios 42 y 45, a : b significa "guarda el dato a en la celda b".

42. 53 : 9, 13 : 2, 281 : 6, 743 : 7, 377 : 3, 20 : 10, 10 : 0, 796 : 4

45. 714 : 0, 631 : 6, 26 : 5, 373 : 1, 775 : 8, 906 : 13, 509 : 2, 2032 : 7, 42 : 4, 4 : 3, 136 : 9, 1028 : 10

48. Durante una búsqueda, si nos detenemos en una celda vacía, tal vez no se encuentre el objeto o dato aun cuando esté presente. La celda puede estar vacía porque se eliminó un dato. Una solución es marcar las celdas eliminadas y considerarlas no vacías durante una búsqueda.

49. Falso. Tome $g = \{(1, a), (2, b)\}$ y $f = \{(a, z), (b, z)\}$.

52. Verdadero. Sea $z \in Z$. Como f es uno a uno, existe $y \in Y$ tal que $f(y) = z$. Dado que g es sobre, existe $x \in X$ tal que $g(x) = y$. Ahora $f(g(x)) = f(y) = z$. Por lo tanto $f \circ g$ es sobre.

55. Verdadero. Suponga que $g(x_1) = g(x_2)$. Entonces $f(g(x_1)) = = f(g(x_2))$. Como $f \circ g$ es uno a uno, $x_1 = x_2$. Por lo tanto, g es uno a uno.

57. $g(S) = \{a\}$, $g(T) = \{a, c\}$, $g^{-1}(U) = \{1\}$, $g^{-1}(V) = \{1, 2, 3\}$.

62. Si $x \in X \cap Y$, $C_{X \cap Y}(x) = 1 = 1 \cdot 1 = C_X(x)C_Y(x)$. Si $x \notin X \cap Y$, entonces $C_{X \cap Y}(x) = 0$. Como ya sea $x \notin X$ o $y \notin Y$, ocurre $C_X(x) = 0$ o bien $C_Y(x) = 0$. Entonces $C_X(x)C_Y(x) = 0 = C_{X \cap Y}(x)$..

65. Si $x \in X$, entonces

$$C_{X-Y}(x) = 1 = 1 \cdot [1 - 0] = C_X(x)[1 - C_Y(x)].$$

Si $x \notin X - Y$, entonces ya sea $x \notin X$ o $y \in Y$. En el caso $x \notin X$,

$$C_{X-Y}(x) = 0 = 0 \cdot [1 - C_Y(x)] = C_X(x)[1 - C_Y(x)].$$

En el caso $x \in Y$,

$$C_{X-Y}(x) = 0 = C_X(x)[1 - 1] = C_X(x)[1 - C_Y(x)].$$

Entonces la ecuación se cumple para toda $x \in U$.

68. f es sobre por definición. Suponga que $f(X) = f(Y)$. Entonces $C_X(x) = C_Y(x)$, para toda $x \in U$. Suponga que $x \in X$. Entonces $C_X(x) = 1$. Así, $C_Y(x) = 1$. Por lo tanto, $x \in Y$. Este argumento muestra que $X \subseteq Y$. De manera similar, $Y \subseteq X$. Por lo tanto, $X = Y$ y f es uno a uno.

70. f es un operador binario, conmutativo.

73. f no es un operador binario ya que $f(x, 0)$ no está definido.

75. $g(x) = -x$

78. La afirmación es verdadera. El entero más pequeño mayor o igual que x es el único entero k que satisface

$$k - 1 < x \leq k.$$

Ahora bien

$$k + 2 < x + 3 \leq k + 3.$$

Entonces, $k + 3$ es el entero más pequeño mayor o igual que $x + 3$. Por lo tanto, $k + 3 = \lceil x + 3 \rceil$. Como $k = \lceil x \rceil$, se tiene

$$\lceil x + 3 \rceil = k + 3 = \lceil x \rceil + 3.$$

81. Si n es un entero impar, $n = 2k + 1$ para algún entero k. Ahora

$$\frac{n^2}{4} = \frac{(2k + 1)^2}{4} = \frac{4k^2 + 4k + 1}{4} = k^2 + k + \frac{1}{4}.$$

Como $k^2 + k$ es un entero,

$$\left\lfloor \frac{n^2}{4} \right\rfloor = k^2 + k.$$

El resultado se deduce ahora porque

$$\left(\frac{n-1}{2}\right)\left(\frac{n+1}{2}\right) = \left[\frac{(2k+1)-1}{2}\right]\left[\frac{(2k+1)+1}{2}\right]$$
$$= \frac{2k(2k+2)}{4}$$
$$= \frac{4k^2 + 4k}{4} = k^2 + k.$$

84. Sea $k = \lceil x \rceil$. Entonces $k - 1 < x \leq k$ y $2x \leq 2k$.. Así, $\lceil 2x \rceil \leq 2k = 2\lceil x \rceil$. Ahora bien, $\lceil x \rceil = k < x + 1$. Por lo tanto, $2\lceil x \rceil < 2x + 2 \leq \lceil 2x \rceil + 2$, así que $2\lceil x \rceil - 2 < \lceil 2x \rceil$. Por lo tanto, $2\lceil x \rceil - 1 \leq \lceil 2x \rceil$.

85. Abril, julio.

Sección 2.3 Repaso

1. Una sucesión es una función en la que el dominio consiste en un conjunto de enteros consecutivos.

2. Si s_n denota el elemento n de una sucesión, n es llama el índice de la sucesión.

3. Una sucesión s es creciente si $s_n < s_{n+1}$ para toda n.

4. Una sucesión s es decreciente si $s_n > s_{n+1}$ para toda n.

5. Una sucesión s es no creciente si $s_n \geq s_{n+1}$ para toda n.

6. Una sucesión s es no decreciente si $s_n \leq s_{n+1}$ para toda n.

7. Sea $\{s_n\}$ una sucesión definida para $n = m, m + 1, \ldots$, y sea $n_1, n_2, \ldots$ una sucesión creciente cuyos valores están en el conjunto $\{m, m + 1, \ldots\}$. La sucesión $\{s_{n_k}\}$ se llama una subsucesión de $\{s_n\}$.

8. $a_m + a_{m+1} + \cdots + a_n$ **9.** $a_m a_{m+1} \cdots a_n$

10. Una cadena en X es una sucesión finita de elementos de X.

11. La cadena nula es la cadena sin elementos.

12. X^* es el conjunto de todas las cadenas en X.

13. X^+ es el conjunto de todas las cadenas no nulas en X.

14. La longitud de una cadena α es el número de elementos en α. Se denota por $|\alpha|$.

15. La concatenación de cadenas α y β es la cadena que consiste en α seguida de β. Se denota por $\alpha\beta$.

16. Una cadena β es una subcadena de la cadena α si existen cadenas γ y δ tales que $\alpha = \gamma\beta\delta$.

Sección 2.3

1. c **2.** c

3. $cddcdc$ **25.** 52

26. 52 **27.** No

28. No **29.** No

30. Sí **39.** 12

40. 23 **41.** 7

42. 46 **43.** 1

44. 3 **45.** 3

46. 21 **47.** No

48. No **49.** No

50. Sí **67.** 15

68. 155 **69.** $2n + 3(n-1)n/2$

70. Sí **71.** No

72. No **73.** Sí

83. 1, 3, 5, 7, 9, 11, 13 **84.** 1, 5, 9, 13, 17, 21, 25,

85. $n_k = 2k - 1$ **86.** $s_{n_k} = 4k - 3$

91. 88 **92.** 1140

93. 48 **94.** 3168

111. $b_1 = 1, b_2 = 2, b_3 = 3, b_4 = 4, b_5 = 5, b_6 = 126$

114. Sea $s_0 = 0$. Entonces

$$\begin{aligned}\sum_{k=1}^{n} a_k b_k &= \sum_{k=1}^{n} (s_k - s_{k-1}) b_k \\ &= \sum_{k=1}^{n} s_k b_k - \sum_{k=1}^{n} s_{k-1} b_k \\ &= \sum_{k=1}^{n} s_k b_k - \sum_{k=1}^{n} s_k b_{k+1} + s_n b_{n+1} \\ &= \sum_{k=1}^{n} s_k (b_k - b_{k+1}) + s_n b_{n+1}.\end{aligned}$$

117. 00, 01, 10, 11

120. 000, 010, 001, 011, 100, 110, 101, 111, 00, 01, 11, 10, 0, 1, λ

124. Sea $\alpha = \lambda$. Entonces $\alpha \in L$ y la primera regla establece que $ab = a\alpha b \in L$. Ahora $\beta = ab \in L$ y la primera regla establece que $aabb = a\beta b \in L$. Ahora $\gamma = aabb \in L$ y la primera regla establece que $aaabbb = a\gamma b \in L$.

127. Se usa la inducción fuerte sobre la longitud n de α para demostrar que si $\alpha \in L$, α tiene un número igual de símbolos a y b.

El paso base es $n = 0$. En este caso, α es la cadena nula, que tienen un número igual de símbolos a y b.

Ahora se ve el paso inductivo. Se supone que cualquier cadena en L de longitud $k < n$ tiene un número igual de símbolos a y b. Debe demostrarse que cualquier cadena en L de longitud n tiene un número igual de símbolos a y b. Sea $\alpha \in L$ y suponga que $|\alpha| = n > 0$. Ahora bien, α está en L por la regla 1 o a la regla 2.

Suponga que α está en L por la regla 1. En este caso, $\alpha = a\beta b$ o $\alpha = b\beta a$, donde $\beta \in L$. Como $|\beta| < n$, por la hipótesis inductiva β tiene un número igual de símbolos a y b. Como $\alpha = a\beta b$ o $\alpha = b\beta a$, α también tiene igual número de símbolos a y b.

Suponga que α está en L por la regla 2. En este caso, $\alpha = \beta\gamma$, donde $\beta \in L$ y $\gamma \in L$. Como $|\beta| < n$ y $|\gamma| < n$, por la hipótesis inductiva β y γ tienen cada uno igual número de símbolos a y b. Como $\alpha = \beta\gamma$, α también tiene igual número de símbolos a y b. La prueba de inducción queda completa.

Capítulo 2 Autoevaluación

1. $\emptyset$

2. 256, 255

3. $A \subseteq B$

4. Sí

5. f no es uno a uno. f es sobre.

6. $x = y = 2.3$

7. Defina f de $X = \{1, 2\}$ a $\{3\}$ por $f(1) = f(2) = 3$. Defina g de $\{1\}$ a X por $g(1) = 1$.

8. ($a : b$ significa "almacena artículo a en celda b".) 1 : 1, 784 : 4, 18 5, 329 : 6, 43 : 7, 281 : 8, 620 : 9, 1141 : 10, 31 : 11, 684 : 12

9. *a*) 14

b) 18

c) 192

d) $a_{n_k} = 4k$

10. $\displaystyle\sum_{k=-1}^{n-2} (n - k - 2) r^{k+2}$

11. *a*) $b_5 = 35,\ b_{10} = 120$

b) $(n+1)^2 - 1$

c) Sí

d) No

12. *a*) $ccddccccdd$

b) $cccddccddc$

c) 5

d) 20

Sección 3.1 Repaso

1. Una relación binaria de un conjunto X a un conjunto Y es un subconjunto del producto cartesiano $X \times Y$.

2. $\{x \in X \mid (x, y) \in R$ para alguna $y \in Y\}$

3. $\{y \in Y \mid (x, y) \in R$ para alguna $x \in X\}$

4. En una digráfica de una relación en X, los vértices representan los elementos de X y las aristas dirigidas de x a y representan los elementos (x, y) en la relación.

5. Una relación R en un conjunto X es reflexiva si $(x, x) \in R$ para toda $x \in X$. La relación $\{(1, 1), (2, 2)\}$ es una relación reflexiva en $\{1, 2\}$. La relación $\{(1, 1)\}$ no es reflexiva en $\{1, 2\}$.

6. Una relación R en un conjunto X es simétrica si para toda $x, y \in X$, si $(x, y) \in R$, entonces $(y, x) \in R$. La relación $\{(1, 2), (2, 1)\}$ es una relación simétrica en $\{1, 2\}$. La relación $\{(1, 2)\}$ no es una relación simétrica en $\{1, 2\}$.

7. Una relación R en un conjunto X es antisimétrica para toda $x, y \in X$, si $(x, y) \in R$ y $x \neq y$, entonces $(y, x) \notin R$. La relación $\{(1, 2)\}$ es una relación antisimétrica en $\{1, 2\}$. La relación $\{(1, 2), (2, 1)\}$ no es una relación antisimétrica en $\{1, 2\}$.

8. Una relación R en un conjunto X es transitiva si para toda $x, y, z \in X$, si (x, y) y $(y, z) \in R$, entonces $(x, z) \in R$. La relación $\{(1, 2), (2, 3), (1, 3)\}$ es una relación transitiva en $\{1, 2, 3\}$. La relación $\{(1, 2), (2, 1)\}$ no es una relación transitiva en $\{1, 2\}$.

9. Una relación R en un conjunto X es un orden parcial si R es reflexiva, antisimétrica y transitiva. La relación

$$\{(1, 1), (2, 2), (3, 3), (1, 2), (2, 3), (1, 3)\}$$

es un orden parcial en $\{1, 2, 3\}$.

10. Si R es una relación de X a Y, el inverso de R es la relación de Y a X:

$$R^{-1} = \{(y, x) \mid (x, y) \in R\}.$$

El inverso de la relación $\{(1, 2), (1, 3)\}$ es $\{(2,1), (3,1)\}$.

11. Sea R_1 una relación de X a Y y sea R_2 una relación de Y a Z. La composición de R_1 y R_2 es la relación de X a Z

$$R_2 \circ R_1 = \{(x, z) \mid (x, y) \in R_1 \text{ y } (y, z) \in R_2 \text{ para alguna } y \in Y\}.$$

La composición de las relaciones

$$R_1 = \{(1, 2), (1, 3), (2, 2)\}$$

y

$$R_2 = \{(2, 1), (2, 3), (1, 4)\}$$

es

$$R_2 \circ R_1 = \{(1, 1), (1, 3), (2, 1), (2, 3)\}.$$

Sección 3.1

1. {(8840, martillo), (9921, alicates), (452, pintura), (2207, alfombra)}

4. $\{(a, a), (b, b)\}$

5.

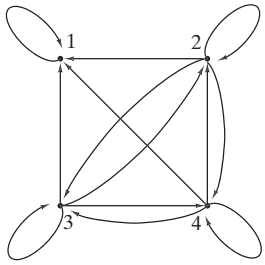
a	6
b	2
a	1
c	1

8.

Mercurio	1
Venus	2
Tierra	3
Marte	4
Júpiter	5
Saturno	6
Urano	7
Neptuno	8
Plutón	9

9.

12.



13. $\{(a, b), (a, c), (b, a), (b, d), (c, c), (c, d)\}$

16. $\{(b, c), (c, b), (d, d)\}$

17. (Para el ejercicio 1) dominio = {8840, 9921, 452, 2207}, rango = {martillo, alicates, pintura, alfombra}

19. $\{(1, 1), (1, 4), (2, 2), (2, 5), (3, 3), (4, 1), (4, 4), (5, 2), (5, 5)\}$

22. $\{1, 2, 3, 4, 5\}$

25. $R = R^{-1} = \{(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (2, 1), (2, 2), (2, 3), (2, 4), (3, 1), (3, 2), (3, 3), (4, 1), (4, 2), (5, 1)\}$

$$\text{dominio } R = \text{rango } R = \text{dominio } R^{-1} = \text{rango } R^{-1} = \{1, 2, 3, 4, 5\}$$

28. Antisimétrica

29. Antisimétrica

32. Reflexiva, simétrica, antisimétrica, transitiva, orden parcial

35. Reflexiva, antisimétrica, transitiva, orden parcial

38. $R_1 \circ R_2 = \{(1, 1), (1, 2), (2, 1), (2, 2), (3, 1), (3, 2), (4, 2)\}$
$R_2 \circ R_1 = \{(1, 1), (1, 2), (3, 4), (4, 1), (4, 2)\}$

39. $\{(1, 1), (2, 2), (3, 3), (4, 4), (1, 2), (2, 3), (2, 1), (3, 2)\}$

42. $\{(1, 1), (1, 2), (2, 1), (2, 2)\}$

44. Falsa. Sea $R = \{(1, 2)\}$, $S = \{(2, 3)\}$

47. Verdadera. Sea $(x, y), (y, z) \in R^{-1}$. Entonces $(z, y), (y, x) \in R$. Como R es transitiva, $(z, x) \in R$. Entonces $(x, z) \in R^{-1}$. Por lo tanto, R^{-1} es transitiva.

50. Verdadera. Debe demostrarse que $(x, x) \in R \circ S$ para toda $x \in X$. Sea $x \in X$. Como R y S son reflexivas, $(x, x) \in R$ y $(x, x) \in S$. Por lo tanto, $(x, x) \in R \circ S$ y $R \circ S$ es reflexiva.

53. Verdadera. Sea $(x, y) \in R \cap S$. Entonces $(x, y) \in S$ y $(x, y) \in S$. Como R y S son simétricas, $(y, x) \in R$ y $(y, x) \in S$. Por lo tanto, $(y, x) \in R \cap S$ y $R \cap S$ es simétrica.

56. Falsa. Sea $R = \{(1, 2)\}$, $S = \{(2, 1)\}$.

59. Verdadera. Suponga que $(x, y) \in R^{-1}$ y $x \neq y$. Entonces $(y, x) \in R$ y $y \neq x$. Como R es antisimétrica, $(x, y) \notin R$. Entonces $(y, x) \notin R^{-1}$ y R^{-1} es antisimétrica.

60. R es reflexiva y simétrica. R no es antisimétrica, ni transitiva, ni un orden parcial.

Sección 3.2 Repaso

1. Una relación de equivalencia es una relación reflexiva, simétrica y transitiva. La relación

$$\{(1, 1), (2, 2), (3, 3), (1, 2), (2, 1)\}$$

es una relación de equivalencia en $\{1, 2, 3\}$. La relación

$$\{(1, 1), (3, 3), (1, 2), (2, 1)\}$$

no es una relación de equivalencia en $\{1, 2, 3\}$.

2. Sea R una relación de equivalencia en X. Las clases de equivalencia

de X dadas por R son los conjuntos de la forma

$$\{x \in X \mid xRa\},$$

donde $a \in X$.

3. Si R es una relación de equivalencia en X, las clases de equivalencia forman una partición de X. Inversamente, si S es una partición de X y se define xRy tal que para alguna $S \in \mathcal{S}$ tanto x como y pertenecen a S, entonces R es una relación de equivalencia.

Sección 3.2

1. Relación de equivalencia: $[1] = [3] = \{1,3\}$, $[2] = \{2\}$, $[4] = \{4\}$, $[5] = \{5\}$

4. Relación de equivalencia: $[1] = [3] = [5] = \{1, 3, 5\}$, $[2] = \{2\}$, $[4] = \{4\}$

7. No es una relación de equivalencia (ni transitiva ni reflexiva).

9. La relación es de equivalencia.

12. La relación no es una de equivalencia. No es reflexiva ni simétrica.

15. $\{(1, 1), (1, 2), (2, 1), (2, 2), (3, 3), (3, 4), (4, 3), (4, 4)\}$,
$[1] = [2] = \{1, 2\}$, $[3] = [4] = \{3, 4\}$

18. $\{(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3), (4, 4)\}$,
$[1] = [2] = [3] = \{1, 2, 3\}$, $[4] = \{4\}$

22. $\{1\}, \{1, 3\}, \{1, 4\}, \{1, 3, 4\}$

24. [Inciso *b)*]

{San Francisco, San Diego, Los Ángeles}, {Pittsburg, Filadelfia}, {Chicago}

27. $R = \{(x, x) \mid x \in X\}$

30. [Inciso *b)*]
$(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (1, 10), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1), (10, 1)$

33. *a)* Se demuestra sólo la simetría. Sea $(x, y) \in R_1 \cap R_2$, entonces $(x, y) \in R_1$ y $(x, y) \in R_2$. Como R_1 y R_2 son simétricas, $(y, x) \in R_1$ y $(y, x) \in R_2$. Entonces $(y, x) \in R_1 \cap R_2$ y, por lo tanto, $R_1 \cap R_2$ es simétrica.

b) A es una clase de equivalencia de $R_1 \cap R_2$ si y sólo si existen clases de equivalencia A_1 de R_1 y A_2 de R_2 tales que $A = A_1 \cap A_2$.

36. [Inciso *b)*] Parecido a una dona.

39. Si $x \in X$, entonces $x \in f^{-1}(f(\{x\}))$. Así, $\cup\{S \mid S \in \mathcal{S}\} = X$. Suponga que

$$a \in f^{-1}(\{y\}) \cap f^{-1}(\{z\})$$

para alguna $y, z \in Y$. Entonces $f(a) = y$ y $f(a) = z$. Entonces $y = z$. Por lo tanto, $\mathcal{S}$ es una partición de X. La relación de equivalencia que genera esta partición está dada en el ejercicio 37.

42. *b)* La sucesiones son equivalentes si sus dominios son del mismo tamaño y sus primeros valores del rango coinciden, sus segundos valores del rango coinciden, etcétera.

c) Para que dos sucesiones sean iguales, sus dominios deben ser *iguales* y sus primeros valores del rango deben coincidir, sus segundos valores del rango, coincidir, etcétera. Las sucesiones equivalentes pueden tener dominios diferentes.

43. $\rho(R_1) = \{(1, 1), (2, 2), (3, 3), (4, 4), (1, 2), (3, 4), (4, 2)\}$
$\sigma(R_1) = \{(1, 1), (2, 1), (1, 2), (3, 4), (4, 3), (4, 2), (2, 4)\}$
$\tau(R_1) = \{(1, 1), (1, 2), (3, 4), (4, 2), (3, 2)\}$
$\tau(\sigma(\rho(R_1))) = \{(x, y) \mid x, y \in \{1, 2, 3, 4\}\}$

46. Sea $(x, y), (x, z) \in \tau(R)$. Entonces $(x, y) \in R^m$ y $(y, z) \in R^n$. Así, $(x, z) \in R^{m+n}$. Por lo tanto, $(x, z) \in \tau(R)$ y $\tau(R)$ es transitiva.

49. Suponga que R es transitiva. Si $(x, y) \in \tau(R) = \cup\{R^n\}$, entonces existe $x = x_0, \ldots, x_n = y \in X$ tal que $(x_{i-1}, x_i) \in R$ para $i = 1, \ldots, n$. Como R es transitiva, se deduce que $(x, y) \in R$. Entonces $R \supseteq \tau(R)$. Como siempre se tiene $R \subseteq \tau(R)$, se deduce que $R = \tau(R)$.

Suponga que $\tau(R) = R$. Por el ejercicio 46, $\tau(R)$ es transitiva. Por lo tanto, R es transitiva.

50. Verdadera. Sea $D = \{(x, x) \mid x \in X\}$. Entonces, por definición, $\rho(R) = R \cup D$, donde R es cualquier relación en X. Ahora

$$\begin{aligned}\rho(R_1 \cup R_2) &= (R_1 \cup R_2) \cup D = (R_1 \cup D) \cup (R_2 \cup D) \\ &= \rho(R_1) \cup \rho(R_2).\end{aligned}$$

53. Falsa. Sea $R_1 = \{(1, 2), (2, 3)\}$, $R_2 = \{(1, 3), (3, 4)\}$.

56. Verdadera. Usando la notación y la sugerencia para el ejercicio 50,

$$\rho(\tau(R_1)) = \tau(R_1) \cup D$$

y

$$\tau(\rho(R_1)) = \tau(R_1 \cup D).$$

De manera que debe demostrarse que $\tau(R_1) \cup D = \tau(R_1 \cup D)$.

Primero se observa que si $A \subseteq B$, entonces $\tau(A) \subseteq \tau(B)$. Ahora $R_1 \subseteq R_1 \cup D$. Por lo tanto, $\tau(R_1) \subseteq \tau(R_1 \cup D)$. Además $D \subseteq R_1 \cup D$. Por lo tanto, $D = \tau(D) \subseteq \tau(R_1 \cup D)$. Se deduce que $\tau(R_1) \cup D \subseteq \tau(R_1 \cup D)$.

Como $R_1 \subseteq \tau(R_1)$, $R_1 \cup D \subseteq \tau(R_1) \cup D$. Por la nota en el párrafo anterior, se tiene $\tau(R_1 \cup D) \subseteq \tau(\tau(R_1) \cup D)$. Como $\tau(R_1) \cup D$ es transitiva, $\tau(\tau(R_1) \cup D) = \tau(R_1) \cup D$ (ejercicio 49). Por lo tanto, $\tau(R_1 \cup D) \subseteq \tau(R_1) \cup D$.

57. Un conjunto es equivalente a sí mismo mediante la función de identidad.

Si X es equivalente a Y, existe una función f uno a uno y sobre de X a Y. Ahora f^{-1} es una función uno a uno y sobre de X a Y.

Si X es equivalente a Y, existe una función f uno a uno y sobre de X a Y. Si Y es equivalente a Z, existe una función g uno a uno y sobre de Y a Z. Ahora $g \circ f$ es una función uno a uno y sobre de X a Z.

60. Suponga que X es equivalente a $\mathcal{P}(X)$. Entonces existe una función f uno a uno y sobre de X a $\mathcal{P}(X)$. Sea

$$Y = \{x \in X \mid x \notin f(x)\}.$$

Entonces $f(y) = Y$ para alguna $y \in X$. Considere las posibilidades $y \in Y$ y $y \notin Y$.

Sección 3.3 Repaso

1. Para obtener la matriz de una relación de X a Y, se etiquetan los renglones con los elementos de X y las columnas con los elementos de Y. Después, el elemento en el renglón x y la columna y se hace igual a 1 si xRy y a 0 de otra manera.

2. Una relación es reflexiva si y sólo si su matriz tiene unos en la diagonal principal.

3. Una relación es simétrica si y sólo si su matriz A satisface lo siguiente: Para toda i y j, el elemento ij de A es igual al elemento ji de A.

4. Vea el párrafo que sigue a la demostración del teorema 3.3.6.

5. La matriz de la relación $R_2 \circ R_1$ se obtiene sustituyendo cada término diferente de cero en A_1A_2 por 1.

Sección 3.3

1.
$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} \begin{array}{cccc} \alpha & \beta & \Sigma & \delta \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

4.
$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

8. $R = \{(a, w), (a, y), (c, y), (d, w), (d, x), (d, y), (d, z)\}$

11. La prueba es, siempre que el elemento *ij* sea 1, $i \neq j$, entonces el elemento *ji no* es 1.

14. (Para el ejercicio 8)
$$\begin{array}{c} \\ w \\ x \\ y \\ z \end{array}\begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

16. *a)* $A_1 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \end{pmatrix}$

b) $A_2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$

c) $A_1A_2 = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$

d) Se cambia cada elemento diferente de cero en el inciso *c)* a 1 para obtener
$$A_1A_2 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

e) $\{(1, b), (1, a), (1, c), (2, b), (3, b)\}$

19. Cada columna que contiene 1 en el renglón x corresponde a un elemento de la clase de equivalencia que contiene a x.

21. Suponga que el elemento *ij* de A es 1. Entonces el elemento *ij* de A_1 o de A_2 es 1. Así, ya sea $(i, j) \in R_1$ o $(i, j) \in R_2$. Por lo tanto, $(i, j) \in R_1 \cup R_2$. Ahora suponga que $(i, j) \in R_1 \cup R_2$. Entonces el elemento *ij* de A_1 o de A_2 es 1. Por lo tanto, el elemento *ij* de A es 1. Se deduce que A es la matriz de $R_1 \cup R_2$.

25. Cada renglón debe contener exactamente un 1 para que la relación sea una función.

Sección 3.4 Repaso

1. Una relación *n*-aria es un conjunto de *n*-eadas.

2. Un sistema de administración de base de datos es un programa que ayuda a los usuarios a tener acceso a la información de esa base de datos.

3. Una base de datos relacional representa los datos como tablas y proporciona maneras de manipular las tablas.

4. Un solo atributo o combinación de atributos para una relación es una clave si los valores de los atributos definen una *n*-eada de manera única.

5. Una consulta es una solicitud de información de una base de datos.

6. El operador selección elige ciertas *n*-eadas de una relación. Las elecciones se hacen dando condiciones a los atributos. Para ilustrar esto, vea el ejemplo 3.4.3.

7. El operador proyección elige columnas específicas de una relación. Además, se eliminan los duplicados. Para ilustrar esto, vea el ejemplo 3.4.4.

8. La operación unión en las relaciones R_1 y R_2 comienza examinando todos los pares de *n*-eadas, una de R_1 y una de R_2. Si la condición conjunta se satisface, las *n*-eadas se combinan para formar una nueva *n*-eada. La condición conjunta especifica una relación entre un atributo en R_1 y un atributo en R_2. Para ilustrar esto, vea el ejemplo 3.4.5.

Sección 3.4

1. {(1089, Suzuki, Zamora), (5620, Kaminski, Jones), (9354, Jones, Yu), (9551, Ryan, Washington), (3600, Beaulieu, Yu), (0285, Schmidt, Jones), (6684, Manacotti, Jones)}

5. EMPLEADO [Nombre]
Suzuki, Kaminski, Jones, Ryan, Beaulieu,
Schmidt, Manacotti

8. COMPRADOR [Nombre]
United Supplies, ABC Unlimited, JCN Electronics,
Danny's, Underhanded Sales, DePaul University

11. TEMP: = COMPRADOR [parte núm = 20A8]
TEMP [Nombre]
Underhanded Sales, Danny's, ABC Unlimited

14. TEMP1:= COMPRADOR [Nombre = Danny′s]
TEMP2 := TEMP1 [parte núm = parte núm] PROVEEDOR
TEMP2 [dept]
04, 96

17. TEMP1 := COMPRADOR [nombre = JNC Electronics]
TEMP2 := TEMP1 [parte núm = parte núm] PROVEEDOR
TEMP3 := TEMP2 [dept = dept] DEPARTAMENTO
TEMP4 := TEMP3 [admin = admin] EMPLEADO
TEMP4 [nombre]
Kaminski, Schmidt, Manicotti

22. Sean R_1 y R_2 dos relaciones *n*-arias. Suponga que el conjunto de elementos en la columna *i* de R_1 y el conjunto de elementos en la columna *j* de R_2 vienen de un domino común para $i = 1, \ldots, n$. La *unión* de R_1 y R_2 es la relación *n*-aria $R_1 \cup R_2$.
TEMP1 := DEPARTAMENTO [dept = 23]
TEMP2 := DEPARTAMENTO [dept = 96]
TEMP3 := TEMP1 *unión* TEMP2
TEMP4 := TEMP3 [admin = admin] EMPLEADO
TEMP4 [nombre]
Kaminski, Schmidt, Manacotti, Suzuki

Capítulo 3 Autoevaluación

1. Reflexiva, simétrica, transitiva

2. Simétrica

3. $R = \{(1, 1), (2, 2), (3, 3), (4, 4), (1, 2), (2, 1),$

4. Las relaciones de contraejemplo están dadas en $\{1, 2, 3\}$.
a) Falsa. $R = \{(1, 1)\}$
b) Verdadera
c) Verdadera
d) Falsa. $R = \{(1, 1)\}$

5. Sí. Es reflexiva, simétrica y transitiva.

6. $[3] = \{3, 4\}$. Hay dos clases de equivalencia.

7. $\{(a, a), (b, b), (b, d), (b, e), (d, b), (d, d), (d, e), (e, b), (e, d), (e, e), (c, c)\}$

8. *a)* R es reflexiva porque cualquier cadena de 8 bits tiene el mismo número de ceros que sí misma.

R es simétrica porque si s_1 y s_2 tienen el mismo número de ceros, entonces s_2 y s_1 tienen el mismo número de ceros.

Para ver que R es transitiva, suponga que s_1 y s_2 tienen el mismo número de ceros y que s_2 y s_3 tienen el mismo número de ceros. Entonces, s_1 y s_3 tienen el mismo número de ceros. Por lo tanto, R es una relación de equivalencia.

b) Hay nueve clases de equivalencia.

c) 11111111, 01111111, 00111111, 00011111, 00001111, 00000111, 00000011, 00000001, 00000000

9. $\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$ **10.** $\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$

11. $\begin{pmatrix} 1 & 1 & 0 \\ 2 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$ **12.** $\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$

13. ASIGNACIÓN [equipo]

Blue Sox, Mutts, Jackalopes

14. JUGADOR [nombre, edad]

Johnsonbaugh, 22; Glover, 24; Battey, 18; Cage, 30; Homer, 37; Score, 22; Johnsonbaugh, 30; Singleton, 31

15. TEMP1 := JUGADOR [posición]

TEMP2 := TEMP1 [núm ident = IDP] ASIGNACIÓN

TEMP2 [equipo]

Mutts, Jackalopes

16. TEMP1 : JUGADOR [edad $\geq$ 30]

TEMP2 : TEMP1 [núm ident = IDP] ASIGNACIÓN

TEMP2 : [equipo]

Blue Sox, Mutts

Sección 4.1 Repaso

1. Un algoritmo es un método paso a paso para resolver un problema.

2. Entrada: El algoritmo recibe una entrada. Salida: El algoritmo produce una salida. Precisión: Los pasos se enuncian con precisión. Determinismo: Los resultados intermedios de cada paso de la ejecución son únicos y están determinados sólo por las entradas y los resultados de los pasos anteriores. Carácter finito: El algoritmo termina, es decir, se detiene después de ejecutar un número finito de instrucciones. Corrección: La salida producida por el algoritmo es correcta; esto es, el algoritmo resuelve el problema correctamente. Generalidad: El algoritmo se aplica a un conjunto de entradas.

3. Un seguimiento o rastreo de un algoritmo es una simulación de su ejecución.

4. Las ventajas del seudocódigo en comparación con el texto normal son que el seudocódigo tiene más precisión, estructura y universalidad. Con frecuencia se convierte de manera directa en el código de computadora.

5. Un algoritmo se forma con una o más funciones de seudocódigo.

Sección 4.1

2. El algoritmo no recibe una entrada (pero, lógicamente, no la necesita). Si un número par mayor que 2 no es la suma *de dos* números primos, el algoritmo se detiene y produce "no". Si todo número par mayor que 2 es la suma de dos números primos, las líneas 2 y 3 se convierten en un ciclo infinito; en este caso, el algoritmo no termina y, por lo tanto, no produce una salida. El algoritmo carece de precisión; para ejecutar la línea 2 se necesita saber cómo verificar si n es la suma de dos primos. El algoritmo no tiene la propiedad del determinismo. El algoritmo *puede* no tener la propiedad de carácter finito. Ya se hizo notar que si cada número par mayor que 2 es la suma de dos números primos (que por el momento no se ha establecido), el algoritmo no termina. El algoritmo no es general; es decir, no se aplica a un conjunto de entradas. Más bien, se aplica a *un solo* conjunto de entradas, a saber, el conjunto vacío.

5. Entrada: *s, n*

Salida: *chico*, el valor menor en la sucesión *s*

```
mín(s, n) {
    chico = s_1
    for i = 2 to n
        if (s_i < chico) //menor valor encontrado
            chico = s_i
    return chico
}
```

8. Entrada: *s*, *n*

Salida: *chico* (menor), *grande* (mayor)

```
chico_grande(s, n, chico, grande) {
    chico = grande = s_1
    for i = 2 to n {
        if (s_i < chico)
            chico = s_i
        if (s_i > grande)
            grande = s_i
    }
}
```

11. Entrada: *s*, *n*

Salida: *suma*

```
suma_suc(s, n) {
sum = 0
  for i = 1 to n
    suma = suma + S_i
  return suma
}
```

14. Entrada: *s*, *n*

Salida: *s*(en reversa)

```
reversa(s, n) {
    i = 1
    j = n
    while (i < j) {
        cambia(s_i, s_j)
        i = i + 1
        j = j - 1
    }
}
```

17. Entrada: *A* (matriz de $n \times n$ de una relación R), *n*

Salida: verdadero, si R es reflexiva; falso, si no lo es.

```
es_reflexiva(A, n) {
    for i = 1 to n
        if (A_ii == 0)
            return falso
    return veradero
}
```

20. Entrada: A (matriz de $n \times n$ de una relación R), n
Salida: verdadero, si R es antismétrica; falso, si R no es antisimétrica

```
es_reflexiva(A, n) {
    for i = 1 to n − 1
        for j = i + 1 to n
            if (A_ij == 1 ∧ A_ji == 1)
                return falso
    return verdadero
}
```

23. Entrada: A (matriz de $m \times k$ de una relación R_1), B (matriz de $k \times n$ de una relación R_2), m, k, n
Salida: C (matriz de $m \times n$ de una relación $R_1 \circ R_2$)

```
rel_comp(A, B, m, k, n, C) {
    //primero se calcula el producto de matrices AB
    for i = 1 to m
        for j = 1 to n {
            C_ij = 0
            for t = 1 to k
                C_ij = C_ij + A_it B_tj
        }
    //se sustituye cada elemento diferente de 0 en C por 1
    for i = 1 to m
        for j = 1 to n
            if (C_ij > 0)
                C_ij = 1
}
```

Sección 4.2 Repaso

1. Encontrar páginas en Internet que contengan las palabras clave es un problema de búsqueda. Los programas que realizan la búsqueda se llaman *máquinas de búsqueda*. Encontrar registros médicos en un hospital es otro problema de búsqueda. Este tipo de búsqueda puede ser realizada por personas o por una computadora.

2. Se tiene el texto t y se desea encontrar la primera ocurrencia del patrón p en t o determinar que p no ocurre en t.

3. Se usa la notación en la solución del ejercicio 2. Determine si p está en t comenzando en el índice 1 en t. Si así es, se detiene. De otra manera, determine si p está en t comenzando en el índice 2 en t. Si así es, se detiene. Continúe de esta manera hasta encontrar p en t o determinar que p no puede estar en t. En el último caso, la búsqueda puede terminar cuando el índice en t es tan grande que no hay suficientes caracteres restantes en t para que ocurra p.

4. Ordenar una sucesión s significa arreglar los datos de manera que s esté en orden (orden no creciente u orden no decreciente).

5. Los elementos en el índice de un libro se disponen en orden creciente para facilitar la localización de un elemento.

6. Para ordenar $s_1, \ldots, s_n$ usando inserción por orden, primero hay que insertar s_2 en s_1 de manera que s_1, s_2 esté ordenado. Después se inserta s_3 en s_1, s_2, de forma que s_1, s_2, s_3 esté ordenado. Se continúa hasta insertar s_n en $s_1, \ldots, s_{n-1}$ de manera que la sucesión $s_1, \ldots, s_n$ esté ordenada.

7. El tiempo requerido por un algoritmo es el número de pasos hasta la terminación. El espacio requerido por un algoritmo es la cantidad de almacenamiento requerido por la entrada, las variables locales, etcétera.

8. Conocer o poder estimar el tiempo y espacio requeridos por un algoritmo da una indicación de cómo será su desempeño para entradas de diferentes tamaños cuando se corre en una computadora. Conocer o poder estimar el tiempo y espacio requeridos por dos o más algoritmos que resuelven el mismo problema hace posible su comparación.

9. Muchos problemas prácticos son muy difíciles para resolverse con eficiencia y comprometen la generalidad o bien la corrección.

10. Cuando un algoritmo aleatorizado se ejecuta, en algún punto toma decisiones aleatorias.

11. Se viola el requerimiento de que los resultados intermedios en cada paso de la ejecución estén definidos de manera única y dependan sólo de las entradas y de los resultados de los pasos anteriores.

12. Para barajar o desordenar $s_1, \ldots, s_n$, primero se intercambian s_1 y un elemento elegido al azar en $s_1, \ldots, s_n$. Después se intercambian s_2 y un elemento aleatorio en $s_2, \ldots, s_n$. El proceso continúa hasta intercambiar s_{n-1} y un elemento aleatorio en s_{n-1}, s_n.

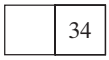
13. Se pueden generar arreglos aleatorios de sucesiones para usarlos como entrada para pruebas o para tomar tiempo a un programa que ordena.

Sección 4.2

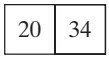
1. Primero i y j se establecen en 1. El ciclo "while" compara $t_1 \ldots t_4$ = "bala" con p = "bala". Como la comparación ocurre, el algoritmo regresa $i = 1$ para indicar que se encontró p en t al comenzar en el índice 1 en t.

4. Primero 20 se inserta en

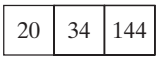
Como 20 < 34, 34 debe moverse una posición a la derecha



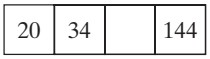
Ahora se inserta 20



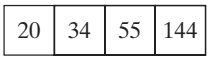
Como 144 > 34, se inserta justo a la derecha de 34



Como 55 < 144, 144 debe moverse una posición a la derecha



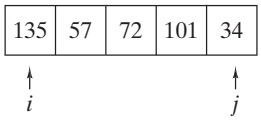
Como 55 > 34, se inserta 55



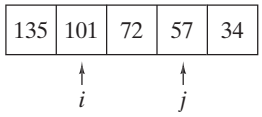
Ahora la sucesión está ordenada.

7. Como cada elemento es mayor o igual que el elemento a su izquierda, el elemento siempre se inserta en su posición original.

8. Primero se intercambian a_i y a_j, donde $i = 1$ y $j = rand(1, 5) = 5$. Después del intercambio se tiene

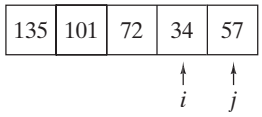


El siguiente paso intercambia a_i y a_j, donde $i = 2$ y $j = rand(2, 5) = 4$. Después del intercambio se tiene



Después se intercambian a_i y a_j, donde $i = 3$ y $j = rand(3, 5) = 3$. La sucesión no cambia.

Después se intercambian a_i y a_j, donde $i = 4$ y $j = rand(4, 5) = 5$. Después del intercambio se tiene



11. El ciclo "while" prueba si p ocurre en el índice i en t. Si p ocurre en el índice i en t, t_{i+j-1} será igual a p_j para toda $j = 1, \ldots, m$. Entonces j se convierte en $m + 1$ y el algoritmo regresa i. Si p no ocurre en el índice i en t, t_{i+j-1} no será igual a p_j para alguna j. En este caso, el ciclo "while" termina (sin ejecutar return i).

Ahora suponga que p ocurre en t y su primera ocurrencia es en el índice i en t. Como se observó en el párrafo anterior, el algoritmo correctamente regresa i, el índice menor en t donde aparece p.

Si p no ocurre en t, entonces el ciclo "while" termina para toda i e incrementos de i en el ciclo "for". Por lo tanto, el ciclo "for" corre hasta terminar, y el algoritmo correctamente regresa 0 para indicar que p no se encontró en t.

14. Entrada: s (la sucesión $s_1, \ldots, s_n$), n y *clave*
Salida: i (el índice de la última ocurrencia de *clave* en s, o 0 si *clave* no está en s

```
búsq_lineal−inver(s, n, clave) {
    i = n
    while (i ≥ 1) {
        if (s_i == clave)
            return i
        i = i − 1
    }
    return 0
}
```

17. Se mide el tiempo del algoritmo contando el número de comparaciones ($t_{i+j-1} == p_j$) en el ciclo "while".

No se hacen comparaciones si $n - m + 1 \leq 0$. En el resto de esta solución, se supone que $n - m + 1 > 0$.

Si p está en t, deben realizarse m comparaciones para verificar que p está, de hecho, en t. Se puede garantizar que se realizan exactamente m comparaciones si p está en el índice 1 en t.

Si p no está en t, debe realizarse al menos una comparación por cada i. Se puede garantizar que se realiza exactamente una comparación por cada i si el primer carácter en p no ocurre en t. En este caso se hacen $n - m + 1$ comparaciones.

Si $m < n - m + 1$, el mejor caso es que p esté en el índice 1 en t. Si $n - m + 1 < m$, el mejor caso es que el primer carácter en p no ocurra en t. Si $m = n - m + 1$, cualquier situación corresponde al mejor caso.

20. Entrada: s (la sucesión $s_1, \ldots, s_n$) y n
Salida: s (ordenada en orden no decreciente)

```
orden_selección(s, n) {
    for i = 1 to n − 1 {
        //encuentra el más pequeño en s_i, . . . , s_n
        índice_menor = i
        for j = i + 1 to n
            if (s_j < s_índice_menor)
                índice_menor = j
        cambia(s_i, s_índice_menor)
    }
}
```

Sección 4.3 Repaso

1. Análisis de algoritmos se refiere al proceso de derivar estimaciones para el tiempo y el espacio necesarios para ejecutarlos.

2. El tiempo en el peor caso para una entrada de tamaño n de un algoritmo es el tiempo máximo necesario para ejecutar el algoritmo entre todas las entradas de tamaño n.

3. El tiempo en el mejor caso para una entrada de tamaño n de un algoritmo es el tiempo promedio necesario para ejecutar el algoritmo entre todas las entradas de tamaño n.

4. El tiempo en el caso promedio para una entrada de tamaño n de un algoritmo es el tiempo promedio necesario para ejecutar el algoritmo para un conjunto finito de entradas todas de tamaño n.

5. $f(n) = O(g(n))$ si existe una constante positiva C_1 tal que $|f(n)| \leq C_1|g(n)|$ para todos menos un número finito de enteros positivos n. Esta notación se llama notación O mayúscula.

6. Excepto por las constantes y un número finito de excepciones, f tiene una cota superior g.

7. $f(n) = \Omega(g(n))$ si existe una constante positiva C_2 tal que $|f(n)| \geq C_2|g(n)|$ para todos menos un número finito de enteros positivos n. Ésta es la notación omega.

8. Excepto por las constantes y un número finito de excepciones, f tiene una cota inferior g.

9. $f(n) = \Theta(g(n))$ si $f(n) = O(g(n))$ y $f(n) = \Omega(g(n))$. Ésta es la notación theta.

10. Excepto por las constantes y un número finito de excepciones, f está acotada arriba y abajo por g.

Sección 4.3

1. $\Theta(n)$ **4.** $\Theta(n^2)$
7. $\Theta(n^2)$ **10.** $\Theta(n)$
13. $\Theta(n^2)$ **16.** $\Theta(n)$
19. $\Theta(n^2)$ **22.** $\Theta(n^3)$
25. $\Theta(n \lg n)$ **28.** $\Theta(1)$

31. Cuando $n = 1$, se obtiene

$$1 = A + B + C.$$

Cuando $n = 2$, se obtiene

$$3 = 4A + 2B + C.$$

Cuando $n = 3$, se obtiene

$$6 = 9A + 3B + C.$$

Al obtener A, B, C, de este sistema de ecuaciones, se tiene

$$A = B = \frac{1}{2}, \qquad C = 0.$$

Se obtiene la fórmula

$$1 + 2 + \cdots + n = \frac{n^2}{2} + \frac{n}{2} + 0 = \frac{n(n+1)}{2},$$

que se puede probar usando inducción matemática (sección 1.7).

33. $n! = n(n-1)\cdots 2 \cdot 1 \leq n \cdot n \cdots n = n^n$

36. Como $n = 2^{\lg n}$, $n^{n+1} = (2^{\lg n})^{n+1} = 2^{(n+1)\lg n}$. Entonces, es suficiente demostrar que $(n+1)\lg n \le n^2$ para toda $n \ge 1$. Una prueba por inducción muestra que $n \le 2^{n-1}$ para toda $n \ge 1$. Entonces, $\lg n \le n-1$ para toda $n \ge 1$. Por lo tanto,

$(n+1)\lg n \le (n+1)(n-1) = n^2 - 1 < n^2$ para toda $n \ge 1$.

39. Dado que $f(n) = O(g(n))$, existen constantes $C' > 0$ y N tales que

$$f(n) \le C'g(n) \text{ para toda } n \ge N.$$

Sea

$$C = \text{máx}\{C', f(1)/g(1), f(2)/g(2), \ldots, f(N)/g(N)\}$$

Para $n \le N$,

$$f(n)/g(n) \le \text{máx}\{f(1)/g(1), f(2)/g(2), \ldots, f(N)/g(N)\} \le C.$$

Para $n \ge N$,

$$f(n) \le C'g(n) \le Cg(n).$$

Por lo tanto, $f(n) \le Cg(n)$ para toda n.

42. Falsa. Si la afirmación fuera cierta, se tendría $n^n \le C2^n$ para alguna constante C y para toda n suficientemente grande. La desigualdad anterior se puede escribir como

$$\left(\frac{n}{2}\right)^n \le C$$

para alguna constante C y para toda n suficientemente grande. Como $(n/2)^n$ crece arbitrariamente cuando n es grande, no se puede tener $n^n \le C2^n$ para alguna constante C y para toda n suficientemente grande.

44. Verdadera

46. Falsa. Un contraejemplo es $f(n) = n$ y $g(n) = 2n$.

49. Verdadera

52. Falsa. Un contraejemplo es $f(n) = 1$ y $g(n) = 1/n$.

53. $f(n) \ne O(g(n))$ significa que para toda constante positiva C, $|f(n)| > C|g(n)|$ para un número infinito de enteros positivos n.

56. Primero se encuentran las funciones positivas no decrecientes f_0 y g_0 tales que para un número infinito de n, n, $f_0(n) = n^2$ y $g_0(n) = n$. Esto implica que $f_0(n) \ne O(g_0(n))$. Nuestras funciones también satisfacen $f_0(n) = n$ y $g_0(n) = n^2$ para un número infinito de n [obviamente n diferentes de aquellas para las que $f_0(n) = n^2$ y $g_0(n) = n$]. Esto implica que $g_0(n) \ne O(f_0(n))$. Si luego se hace $f(n) = f_0(n) + n$ y $g(n) = g_0(n) + n$, se obtienen funciones positivas crecientes para las cuales $f(n) \ne O(g(n))$ y $g(n) \ne O(f(n))$.

Se comienza con $f_0(2) - 2$ y $g_0(2) = 2^n$. Entonces

$$f_0(n) = n, \; g_0(n) = n^2, \quad \text{si } n = 2.$$

Como g_0 es no decreciente, la menor n para la que se puede tener $g_0(n) = n$ es $n = 2^2$. De manera que se definen $f_0(2^2) = 2^4$ y $g_0(2^2) = 2^2$. Entonces

$$f_0(n) = n^2, \quad g_0(n) = n, \text{ si } n = 2^2.$$

El análisis anterior motiva la definición

$$f_0(2^{2^k}) = \begin{cases} 2^{2^k} & \text{si } k \text{ es par} \\ 2^{2^{k+1}} & \text{si } k \text{ es impar} \end{cases}$$

$$g_0(2^{2^k}) = \begin{cases} 2^{2^{k+1}} & \text{si } k \text{ es par} \\ 2^{2^k} & \text{si } k \text{ es impar.} \end{cases}$$

Suponga que $n = 2^{2^k}$. Si k es impar, $f_0(n) = n^2$ y $g_0(n) = n$; si k es par, $f_0(n) = n$ y $g_0(n) = n^2$. Ahora bien f_0 y g_0 están definidas sólo para $n = 2^{2^k}$, pero son no decrecientes en este dominio. Para extender sus dominios al conjunto de enteros positivos, simplemente se define $f_0(1) = g_0(1) = 1$ y se hacen constantes en los conjuntos de la forma $\{i \mid 2^{2^k} \le i < 2^{2^{k+1}}\}$.

60. No

62. *a)* La suma de las áreas de los rectángulos abajo de la curva es igual a

$$\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}.$$

Esta área es menor que el área bajo la curva, que es igual a

$$\int_1^n \frac{1}{x}dx = \log_e n.$$

La desigualdad dada se deriva directamente de aquí.

b) La suma de las áreas de los rectángulos cuyas bases están en el eje x y cuyas partes superiores están más arriba de la curva es igual a

$$1 + \frac{1}{2} + \cdots + \frac{1}{n-1}.$$

Como esta área es mayor que el área bajo la curva, la desigualdad dada se deduce de inmediato

c) El inciso *a)* demostró que

$$1 + \frac{1}{2} + \cdots + \frac{1}{n} = O(\log_e n).$$

Como $\log_e n = \Theta(\lg n)$ (vea el ejemplo 4.3.6),

$$1 + \frac{1}{2} + \cdots + \frac{1}{n} = O(\lg n).$$

De manera similar, se puede concluir a partir del inciso *b)* que

$$1 + \frac{1}{2} + \cdots + \frac{1}{n} = \Omega(\lg n).$$

Por lo tanto,

$$1 + \frac{1}{2} + \cdots + \frac{1}{n} = \Theta(\lg n).$$

64. Al sustituir a por b en la suma, se obtiene

$$\frac{b^{n+1} - a^{n+1}}{b-a} = \sum_{i=0}^{n} a^i b^{n-i} < \sum_{i=0}^{n} b^i b^{n-i} = \sum_{i=0}^{n} b^n = (n+1)b^n.$$

67. Por el ejercicio 65, la sucesión $\{(1 + 1/n)^n\}_{n=1}^{\infty}$ es creciente. Por lo tanto

$$2 = \left(1 + \frac{1}{1}\right)^1 \le \left(1 + \frac{1}{n}\right)^n$$

para todo entero positivo n. El ejercicio 66 demuestra que

$$\left(1 + \frac{1}{n}\right)^n < 4$$

para todo entero positivo n. Tomando logaritmos base 2, se obtiene

$$1 = \lg 2 \le \lg\left(1 + \frac{1}{n}\right)^n < \lg 4 = 2.$$

Puesto que

$$\lg\left(1 + \frac{1}{n}\right)^n = n\lg\left(1 + \frac{1}{n}\right) = n\lg\left(\frac{n+1}{n}\right) = n[\lg(n+1) - \lg n],$$

se tiene

$$1 \le n[\lg(n+1) - \lg n] < 2.$$

Al dividir entre n se obtiene la desigualdad deseada.

70. Verdadera. Como $\lim_{n\to\infty} f(n)/g(n) = 0$, tomando $\varepsilon = 1$, existe N tal que

$$\left|\frac{f(n)}{g(n)}\right| < 1, \qquad \text{para toda } n \ge N.$$

Por lo tanto, para toda $n \ge N$, $|f(n)| < |g(n)|$ y $f(n) = O(g(n))$.

73. Verdadera. Sea $d = |c|$. Como $\lim_{n\to\infty} |f(n)|/|g(n)| = d > 0$, tomando $\varepsilon = d/2$, existe N tal que

$$\left|\frac{|f(n)|}{|g(n)|} - d\right| < d/2, \quad \text{para toda } n \ge N.$$

Esta desigualdad se puede escribir

$$-\frac{d}{2} < \frac{|f(n)|}{|g(n)|} - d < \frac{d}{2}, \quad \text{para toda } n \ge N,$$

o

$$\frac{d}{2} < \frac{|f(n)|}{|g(n)|} < \frac{3d}{2}, \quad \text{para toda } n \ge N,$$

o

$$\frac{d}{2}|g(n)| < |f(n)| < \frac{3d}{2}|g(n)|, \quad \text{para toda } n \ge N.$$

Por lo tanto, $f(n) = \Theta(g(n))$.

78. Se multiplican ambos lados de la desigualdad en el ejercicio 77 por lg e y se usa la fórmula del cambio de base para logaritmos.

Sección 4.4 Repaso

1. Un algoritmo que contiene una función recursiva.

2. Una función que se invoca a sí misma.

3.

```
factorial(n) {
    if (n == 0)
        return 1
    return n * factorial(n − 1)
}
```

4. El problema original se divide en dos subproblemas o más. Se encuentran entonces las soluciones para los subproblemas (por lo general, subdividiendo más). Estas soluciones se combinan a fin de obtener una solución para el problema original.

5. En un caso base, una solución se obtiene directamente, es decir, sin una llamada recursiva.

6. Si una función recursiva no tiene caso base, continuará llamándose a sí misma sin terminar.

7. $f_1 = 1$, $f_2 = 1$, $f_n = f_{n-1} + f_{n-2}$ for $n \ge 3$

8. $f_1 = 1$, $f_2 = 1$, $f_3 = 2$, $f_4 = 3$

Sección 4.4

1. *a)* En la línea 2, como $4 \neq 0$, se procede a la línea 4. El algoritmo se invoca con entrada 3.

b) En la línea 2, como $3 \neq 0$, se procede a la línea 4. El algoritmo se invoca con entrada 2.

c) En la línea 2, como $2 \neq 0$, se procede a la línea 4. El algoritmo se invoca con entrada 1.

d) En la línea 2, como $1 \neq 0$, se procede a la línea 4. El algoritmo se invoca con entrada 0.

e) En las líneas 2 y 3, como $0 = 0$, se regresa 1.

La ejecución se reanuda en la parte *d)* en la línea 4 después de calcular 0! (= 1). Se regresa $0! \cdot 1 = 1$.

La ejecución se reanuda en la parte *c)* en la línea 4 después de calcular 1! (= 1). Se regresa $1! \cdot 2 = 2$.

La ejecución se reanuda en la parte *b)* en la línea 4 después de calcular 2! (= 2). Se regresa $2! \cdot 3 = 6$.

La ejecución se reanuda en la parte *a)* en la línea 4 después de calcular 3! (= 6). Se regresa $3! \cdot 4 = 24$.

4. Se usa inducción sobre i, donde $n = 2^i$. El paso base es $i = 1$. En este caso, el tablero es un tromino T. El algoritmo enlosa correctamente el tablero con T y regresa. Entonces el algoritmo es correcto para $i - 1$.

Ahora suponga que si $n = 2^i$, el algoritmo es correcto. Sea $n = 2^{i+1}$. El algoritmo divide el tablero en cuatro subtableros de $(n/2) \times (n/2)$. Después coloca un tromino derecho en el centro como en la figura 1.7.5. Considera cada uno de los cuadrados cubiertos por el tromino central como faltantes. Después enlosa los cuatro subtableros y, por la suposición inductiva, estos subtableros están enlosados correctamente. Por lo tanto, el tablero de $n \times n$ se enlosa correctamente. El paso inductivo queda completo. El algoritmo es correcto.

7. La demostración es por inducción fuerte sobre n. Los pasos base ($n = 1, 2$) se verifican sin problema.

Suponga que el algoritmo es correcto para toda $k < n$. Debe demostrarse que el algoritmo es correcto para $n > 2$. Como $n > 2$, el algoritmo ejecuta la instrucción de regresar.

return *caminar*$(n - 1)$ + caminar$(n - 2)$

Por la suposición inductiva, el algoritmo calcula correctamente los valores de *caminar*$(n - 1)$ y *caminar*$(n - 2)$. Como

caminar(n) = *caminar*$(n - 1)$ + *caminar*$(n - 2)$,

el algoritmo regresa el valor correcto de *caminar*(n).

10. *a)* Entrada: n

Salida: $2 + 4 + \cdots + 2n$

```
1.  suma(s, n) {
2.      if (n == 1)
3.          return 2
4.      return suma(n − 1) + 2n
5.  }
```

b) **Paso base** ($n = 1$) Si n es igual a 1, se regresa 2 correctamente.

Paso inductivo Suponga que el algoritmo calcula correctamente la suma cuando la entrada es $n - 1$. Ahora suponga que la entrada a este algoritmo es $n > 1$. En la línea 2, como $n \neq 1$, se procede a la línea 4, donde se invoca este algoritmo con entrada $n - 1$. Por la suposición inductiva, el valor regresado, *suma*$(n - 1)$, es igual a

$$2 + \cdots + 2(n-1).$$

En la línea 4, se regresa entonces

$$\textit{suma}(n-1) + 2n = 2 + \cdots + 2(n-1) + 2n,$$

que es el valor correcto.

13. Entrada: La sucesión $s_1, \ldots, s_n$ y la longitud n de la sucesión
Salida: El valor máximo en la sucesión

```
encuentra_máx(s, n) {
    if(n == 1)
        return s_1
    x = encuentra_máx(s, n − 1)
    if(x > s_n)
        return x
    else
        return s_n
}
```

Se demuestra que el algoritmo es correcto usando inducción sobre n. El caso base es $n = 1$. Si $n = 1$, el único elemento en la sucesión es s_1 y el algoritmo lo regresa correctamente.

Suponga que el algoritmo calcula el máximo para una entrada de tamaño $n - 1$, y suponga que el algoritmo recibe una entrada de tamaño n. Por suposición, la llamada recursiva

$x = encuentra\_máx(s, n - 1)$

calcula correctamente x como el valor máximo en la sucesión $s_1, \ldots, s_{n-1}$. Si x es mayor que s_n, el valor máximo en la sucesión $s_1, \ldots, s_n$ es x, el valor regresado por el algoritmo. Si x no es mayor que s_n, el valor máximo en la sucesión $s_1, \ldots, s_n$ es s_n, de nuevo el valor que regresa el algoritmo. En cualquier caso, el algoritmo calcula correctamente el valor máximo en la sucesión. El paso inductivo queda completo y se probó que el algoritmo es correcto.

16. Para listar todas las maneras en que un robot puede caminar n metros, se hace s igual a la cadena nula y se invoca el algoritmo.

Entrada: n, s (una cadena)
Salida: Todas las maneras en que el robot puede caminar n metros. Cada método para caminar n metros incluye la cadena adicional s en la lista.

```
lista_caminar1(n, s)
    if (n == 1) {
        println(s + "da un paso de longitud 1")
        return
    }
    if (n == 2) {
        println(s + "da dos pasos de longitud 1")
        println(s + "da un paso de longitud 2")
        return
    }
    s' = s + "da un paso de longitud 2" //concatenación
    lista_camina1(n − 2, s')
    s' = s + "da un paso de longitud 1" //concatenación
    lista_camina1(n − 1, s')
}
```

18. Después de un mes, todavía hay sólo un par porque un par no se convierte en productivo sino hasta un mes después. Por lo tanto, $a_1 = 1$. Después de dos meses, el par vivo al inicio se convierte en productivo y agrega una par adicional. Por lo tanto, $a_2 = 2$. El incremento en pares de conejos $a_n - a_{n-1}$ del mes $n - 1$ al mes n se debe a cada par vivo en el mes $n - 2$ que produce un par adicional. Esto es, $a_n - a_{n-1} = a_{n-2}$. Como $\{a_n\}$ satisface la misma relación de recurrencia que $\{f_n\}$, $a_1 = f_2$, y $a_2 = f_3$, $a_n = f_{n+1}$, $n \geq 1$.

21. **Paso base** $(n = 2)$

$$f_2^2 = 1 = 1 \cdot 2 - 1 = f_1 f_3 + (-1)^3$$

Paso inductivo

$$\begin{aligned} f_n f_{n+2} + (-1)^{n+2} &= f_n(f_{n+1} + f_n) + (-1)^{n+2} \\ &= f_n f_{n+1} + f_n^2 + (-1)^{n+2} \\ &= f_n f_{n+1} + f_{n-1} f_{n+1} + (-1)^{n+1} + (-1)^{n+2} \\ &= f_{n+1}(f_n + f_{n-1}) = f_{n+1}^2 \end{aligned}$$

24. **Paso base** $(n = 1)$ $f_1^2 = 1^2 = 1 = 1 \cdot 1 = f_1 f_2$

Paso inductivo $\displaystyle\sum_{k=1} f_k^2 = \sum_{k=1} f_k^2 + f_{n+1}^2 = f_n f_{n+1} + f_{n+1}^2$
$= f_{n+1}(f_n + f_{n+1}) = f_{n+1} f_{n+2}$

27. Se usa inducción fuerte.

Paso base $(n = 6, 7)$ $(n = 6, 7)$ $f_6 = 8 > 7.59 = (3/2)^5$ $f_7 = 13 > 11.39 = (3/2)^6$

Paso inductivo

$$f_n = f_{n-1} + f_{n-2} > \left(\frac{3}{2}\right)^{n-2} + \left(\frac{3}{2}\right)^{n-3}$$

$$= \left(\frac{3}{2}\right)^{n-1}\left[\left(\frac{3}{2}\right)^{-1} + \left(\frac{3}{2}\right)^{-2}\right] = \left(\frac{3}{2}\right)^{n-1}\left[\frac{16}{9}\right] > \left(\frac{3}{2}\right)^{n-1}$$

30. Se usa inducción fuerte sobre n.

Paso base $(n = 1)$ $1 = f_1$

Paso inductivo Suponga que $n > 2$ y que cada entero positivo menor que n se puede expresar como la suma de números de Fibonacci distintos, sin que haya dos consecutivos. Sea f_{k_1} el número de Fibonacci más grande que satisface $n \geq f_{k_1}$. Si $n = f_{k_1}$, entonces, de manera trivial, n es la suma de números de Fibonacci distintos, sin que haya dos consecutivos. Suponga que $n > f_{k_1}$. Por la suposición inductiva, $n - f_{k_1}$ se puede expresar como la suma de números de Fibonacci distintos $f_{k_2} > f_{k_3} > \cdots > f_{k_m}$, sin que haya dos consecutivos.

$$n - f_{k_1} = \sum_{i=2}^{m} f_{k_i}.$$

Ahora n se expresa como la suma de números de Fibonacci:

$$n = \sum_{i=1}^{m} f_{k_i}. \qquad (*)$$

A continuación se demuestra que $f_{k_1} > f_{k_2}$, de manera que, en particular, n es la suma de números de Fibonacci *distintos*.

Observe que $f_{k_2} < n$. Como f_{k_1} es el número de Fibonacci más grande que satisface $n \geq f_{k_1}$, $f_{k_2} \leq f_{k_1}$. Si $f_{k_2} = f_{k_1}$,

$$n \geq f_{k_1} + f_{k_2} > f_{k_1} + f_{k_1 - 1} = f_{k_1 + 1}.$$

Esta última desigualdad contradice la elección de f_{k_1} como el número de Fibonacci más grande que satisface $n \geq f_{k_1}$. Por lo tanto, $f_{k_2} > f_{k_1}$.

Los únicos números de Fibonacci en la suma (*) que pueden ser consecutivos son f_{k_1} y f_{k_2}. Si son consecutivos, (*) también se puede escribir como

$$\begin{aligned} n &= \sum_{i=1}^{m} f_{k_i} \\ &= f_{k_1} + f_{k_2} + \sum_{i=3}^{m} f_{k_i} \\ &= f_{k_1} + f_{k_1 - 1} + \sum_{i=3}^{m} f_{k_i} \\ &= f_{k_1 + 1} + \sum_{i=3}^{m} f_{k_i}. \end{aligned}$$

Ahora $f_{k_1+1} \leq n$ y $f_{k_1+1} > f_{k_1}$. Esto contradice la elección de f_{k_1} como el número de Fibonacci más grande que satisface $n \geq f_{k_1}$. El paso inductivo queda completo.

33. **Paso base** ($n = 1$)

$$\frac{dx}{dx} = 1 = 1x^{1-1}$$

Paso inductivo

$$\frac{dx^{n+1}}{dx} = \frac{d(x \cdot x^n)}{dx} = x\frac{dx^n}{dx} + x^n\frac{dx}{dx}$$
$$= xnx^{n-1} + x^n \cdot 1 = (n+1)x^n$$

Capítulo 4 Autoevaluación

1. En la línea 2, se hace *grande* igual a 12. En la línea 3, como $b >$ *grande* (3 > 12) es falso, se pasa a la línea 5. En la línea 5, como $c >$ *grande* (0 > 12) es falso, se pasa a la línea 7, donde se regresa *grande* (12), el máximo de los valores dados.

2.
```
ordena(a, b, c, x, y, z) {
    x = a
    y = b
    z = c
    if (y < x)
        intercambia(x, y)
    if (z < x)
        intercambia(x, z)
    if (z < y)
        intercambia(y, z)
}
```

3.
```
prueba_dif(a, b, c) {
    if (a == b ∨ a == c ∨ b = c)
        return falso
    return verdadero
}
```

4. Si el conjunto S es un conjunto infinito, el algoritmo no terminará, de manera que le faltan las propiedades de carácter finito y salida. La línea 1 no está establecida con precisión ya que no se especifica *cómo* listar todos los subconjuntos de S y sus sumas; entonces el algoritmo no tiene la propiedad de precisión. El orden de los subconjuntos listados en la línea 1 depende del método usado para generarlos, de modo que el algoritmo carece de la propiedad de determinismo. Como la línea 2 depende del orden de los subconjuntos generados en la línea 1, la propiedad de determinismo también falta.

5. El ciclo "while" prueba primero si "110" ocurre en t en el índice 1. Como "110" no ocurre en t en el índice 1, el algoritmo prueba a continuación si "110" ocurre en t en el índice 2. Como "110" no ocurre en t en el índice 2, el algoritmo regresa el valor 2.

6. Primero se inserta 64

44

Como 64 > 44, de inmediato se inserta a la derecha de 44

44	64

Después se inserta 77. Como 77 > 64, se inserta justo a la derecha de 64

44	64	77

Después se inserta 15. Como 15 < 77, 77 se mueve una posición a la derecha

44	64		77

Como 15 < 64, 64 debe moverse una posición a la derecha

44		64	77

Como 15 < 44, 44 debe moverse una posición a la derecha

	44	64	77

Ahora se inserta 15

15	44	64	77

Por último se inserta 3. Como 3 < 77, 77 se mueve una posición a la derecha

15	44	64		77

Como 3 < 64, 64 se mueve una posición a la derecha

15	44		64	77

Como 3 < 44, 44 se mueve una posición a la derecha

15		44	64	77

Como 3 < 15, 15 se mueve una posición a la derecha

	15	44	64	77

Ahora se inserta 3

3	15	44	64	77

La sucesión está ordenada.

7. Primero se intercambian a_i y a_j, donde $i = 1$ y $j = rand(1, 5) = 1$. La sucesión no cambia.

Después se intercambian a_i y a_j, donde $i = 2$ y $j = rand(2, 5) = 3$. Después del intercambio se tiene

5	2	51	44	96
	i	j		

Luego se intercambian a_i y a_j, donde $i = 3$ y $j = rand(3, 5) = 5$. Después del intercambio se tiene

5	2	96	44	51
		i		j

Luego se intercambian a_i y a_j, donde $i = 4$ y $j = rand(4, 5) = 5$. Después del intercambio se tiene

5	2	96	51	44
			↑ i	↑ j

8.
```
repetidos(s, n) {
    i = 1
    while (i < n){
        if (s_i == s_{i+1})
            println(s_i)
        //salta al siguiente elemento diferente de s_i
        j = i
        while (i < n ∧ s_i == s_j)
            i = i + 1
    }
}
```

9. $\Theta(n^3)$ **10.** $\Theta(n^4)$ **11.** $\Theta(n^2)$

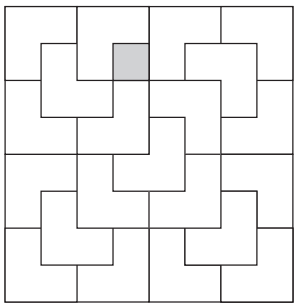
12. Entrada: A y B (matrices de $n \times n$) y n

Salida: verdadero (si $A = B$); falso (si $A \neq B$)

```
matrices_iguales(A, B, n) {
    for i = 1 to n
        for j = 1 to n
            if (A_ij ¬ = B_ij)
                return falso
    return verdadero
}
```

El tiempo en el peor caso es $\Theta(n^2)$.

13. Como $n \neq 2$, se procede de inmediato a la línea 6, donde se divide el tablero en cuatro tableros de 4×4. En la línea 7, se gira el tablero de manera que el cuadro faltante esté en el cuadrante superior izquierdo. En la línea 8 se coloca un tromino en el centro. Luego se procede a las líneas 9 a la 12, donde se llama al algoritmo para enlosar los subtableros. Se obtiene el enlosado siguiente:



14. $t_4 = 3, t_5 = 5$

15. Entrada: n, un entero mayor o igual que 1
Salida: t_n

```
tribonacci(n) {
1.      if (n == 1 ∨ n == 2 ∨ n == 3)
2.          return 1
3.      return tribonacci(n − 1) + tribonacci(n − 2)
            + tribonacci(n − 3)
}
```

16. **Pasos base** ($n = 1, 2, 3$) si $n = 1, 2, 3$, en las líneas 1 y 2 se regresa el valor correcto, 1. Por lo tanto, el algoritmo es correcto en estos casos.

Paso inductivo Suponga que $n > 3$ y que el algoritmo calcula correctamente t_k, si $k > n$. Como $n > 3$, se procede a la línea 3. Después se llama este algoritmo para calcular $t_{n-1} + t_{n-2} + t_{n-3}$. Por la suposición inductiva, los valores calculados son correctos. El algoritmo calcula entonces $t_{n-1} + t_{n-2} + t_{n-3}$. Pero la fórmula muestra que este valor es igual a t_n. Por lo tanto, el algoritmo regresa el valor correcto para t_n.

Sección 5.1 Repaso

1. Se dice que d divide a n si existe un entero q que satisface $n = dq$.

2. Si d divide a n, se dice que d es un divisor de n.

3. Si d divide a n, $n = dq$, q se llama cociente.

4. Un entero mayor que 1 cuyos únicos divisores positivos son 1 y él mismo se llama primo.

5. Un entero mayor que 1 que no es primo se llama compuesto.

6. Si n es compuesto, debe tener un divisor d que satisface $2 \leq d \leq \lfloor \sqrt{n} \rfloor$ (vea el teorema 5.1.7).

7. El algoritmo 5.1.8 no corre en tiempo polinomial respecto al *tamaño* de la entrada.

8. Cualquier entero mayor que 1 se puede escribir como un producto de primos. Más aún, si los primos se escriben en orden no decreciente, la factorización es única.

9. Vea la prueba del teorema 5.1.12.

10. Un divisor común de m y n, ambos diferentes de cero, es un entero que divide tanto a m como a n.

11. El máximo común divisor de m y n, ambos diferentes de cero, es el máximo divisor que m y n tienen en común.

12. Vea el teorema 5.1.17.

13. Un múltiplo común de m y n es un entero que es divisible tanto entre m como entre n.

14. El mínimo común múltiplo de m y n es el mínimo múltiplo positivo que m y n tienen en común.

15. Vea el teorema 5.1.22. **16.** $\text{mcd}(m, n) \cdot \text{mcm}(m, n) = mn$

Sección 5.1

1. Primero d se establece en 2. Como $n \bmod d = 9 \bmod 2 = 1$ no es igual a 0, d se incrementa y se convierte en 3.

Ahora $n \bmod d = 9 \bmod 3$ es igual a 0, de manera que el algoritmo regresa $d = 3$ para indicar que $n = 9$ es compuesto y 3 es un divisor de 9.

4. Cuando d se hace 2, . . . , 6, $n \bmod d$ no es igual a cero. Sin embargo, cuando d se convierte en 7, $n \bmod d = 637 \bmod 7$ es igual a 0, por lo que el algoritmo regresa $d = 7$ para indicar que $n = 637$ es compuesto y 7 es un divisor de 637.

7. Primero d se hace 2, Como $n \bmod d = 3738 \bmod 2$ es igual a 0, el algoritmo regresa $d = 2$ para indicar que $n = 3728$ es compuesto y 2 es un divisor de 3738.

9. 47 **12.** 17

15. 1 **18.** 20

21. 13 **24.** $3^2 \cdot 7^3 \cdot 11$

25. (Para el ejercicio 13) 25

28. Como d divide a m, existe q tal que $m = dq$. Multiplicar por n da $mn = d(qn)$. Por lo tanto, d divide a mn (con cociente qn).

31. Como a divide a b, existe q_1 tal que $b = aq_1$. Como b divide a c,

existe q_2 tal que $c = bq_2$. Ahora

$$c = bq_2 = (aq_1)q_2 = a(q_1q_2).$$

Por lo tanto, a divide a c (con cociente q_1q_2).

Sección 5.2 Repaso

1. $\sum_{i=0}^{n} d_i 10^i$ **2.** $\sum_{i=0}^{n} b_i 2^i$

3. $\sum_{i=0}^{n} h_i 16^i$ **4.** $\lfloor 1 + \lg n \rfloor$

5. Realice el cálculo $\sum_{i=0}^{n} b_i 2^i$ en decimal.

6. Divida el número que va a convertir a binario entre 2. El residuo da el bit de 1. Divida el cociente entre 2. El residuo da el bit de 2. Continúe.

7. Realice el cálculo $\sum_{i=0}^{n} h_i 16^i$ en decimal.

8. Divida el número que va a convertir a hexadecimal entre 16. El residuo da el número de unos. Divida el cociente entre 16. El residuo da el número de números 16. Continúe.

9. Use el algoritmo común que suma números decimales para sumar números binarios (pero sustituya la tabla de suma decimal por la tabla de suma binaria).

10. Utilice el algoritmo común que suma números decimales para sumar números hexadecimales (pero sustituya la tabla de suma decimal por la tabla de suma hexadecimal).

11. Sea

$$n = \sum_{i=0}^{m} b_i 2^i$$

la expansión binaria de n. Eleve al cuadrado repetidas veces para calcular $a^1, a^2, a^4, a^8, \ldots, a^{b_m}$. Después

$$a^n = a^{\sum_{i=0}^{m} b_i 2^i} = \prod_{i=0}^{m} a^{b_i 2^i}.$$

12. Proceda como se describe en la solución del ejercicio 11, pero use la fórmula

$$ab \bmod z = [(a \bmod z)(b \bmod z)] \bmod z.$$

Sección 5.2

1. 6 **4.** 7 **7.** 1585

8. 9 **11.** 32 **14.** 100010

17. 110010000 **20.** 11000 **23.** 1001000

26. 58 **29.** 2563 **32.** (Para el ejercicio 8) 9

35. FE **38.** 3DBF9

40. 2010 no puede representar un número binario porque 2 es un símbolo no permitido en binario. 2010 puede representar un número en decimal o en hexadecimal.

42. 51 **45.** 4570

48. (Para el ejercicio 8) 11 **51.** (Para el ejercicio 42) 33

54. 9450 no puede representar un número en binario porque 9, 4 y 5 no son símbolos permitidos en binario. 9450 no puede representar un número en octal porque este último no permite el 9. 9450 representa un número en decimal o en hexadecimal.

56. El algoritmo comienza por hacer el *resultado* igual a 1 y x igual a a. Como $n = 16 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 no es igual a 1, el *resultado* no se modifica. x se convierte en a^2 y n ahora es 8.

Como $n = 8 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 no es igual a 1, el *resultado* no se modifica. x se convierte en a^4 y n ahora es 4.

Como $n = 4 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 no es igual a 1, el *resultado* no se modifica. x se convierte en a^8 y n ahora es 2.

Como $n = 2 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 no es igual a 1, el *resultado* no se modifica. x se convierte en a^{16} y n ahora es 1.

Como $n = 1 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 es igual a 1, el *resultado* se convierte en *resultado* $* x = 1 * a^{16} = a^{16}$. x se convierte en a^{32} y n ahora es 0.

Como $n = 0$ no es mayor que 0, el ciclo "while" termina. El algoritmo regresa *resultado*, que es igual a a^{16}.

59. El algoritmo comienza por hacer el *resultado* igual a 1 y x igual a a mod $z = 5$ mod $21 = 5$. Como $n = 10 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 no es igual a 1, el *resultado* no se modifica. x se hace igual a $x * x$ mod $z = 25$ mod $21 = 4$ y n es igual a 5.

Como $n = 5 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 es igual a 1, el *resultado* se hace igual a (*resultado* $* x$) mod $z = 4$ mod $21 = 4$. x se hace igual a $x * x$ mod $z = 16$ mod $21 = 16$ y n es igual a 2.

Como $n = 2 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 no es igual a 1, el *resultado* no se modifica. x se hace igual a $x * x$ mod $z = 256$ mod $21 = 4$ y n es igual a 1.

Como $n = 1 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 1 es igual a 1, el *resultado* se hace igual a (*resultado* $* x$) mod $z = 16$ mod $21 = 16$. x se hace igual a $x * x$ mod $z = 16$ mod $21 = 16$ y n es igual a 0.

Como $n = 0$ no es mayor que 0, el ciclo "while" termina. El algoritmo regresa *resultado*, que es igual a a^n mod $z = 5^{10}$ mod 21 $= 16$.

62. Si m_k es la potencia más alta de 2 que divide a m, entonces $m = 2^{m_k} p$, donde p es impar. De manera similar, si n_k es la potencia más alta de 2 que divide a n, entonces $n = 2^{n_k} q$, donde q es impar. Ahora $mn = 2^{m_k + n_k} pq$. Como pq es impar, $m_k + n_k$ es la potencia más alta de 2 que divide a mn, y el resultado es directo.

Sección 5.3 Repaso

1. Vea el algoritmo 5.3.3.

2. Si a es un entero no negativo, b es un entero positivo y $r = a$ mod b, entonces mcd$(a, b) =$ mcd(b, r).

3. $a \geq f_{n+2}$ y $b \geq f_{n+1}$

4. $\log_{3/2} 2m/3$

5. Escriba los residuos diferentes de cero según los encuentra el algoritmo euclidiano en la forma

$$r = n - dq$$

en el orden en que los calcula el algoritmo. Sustituya la fórmula para el penúltimo residuo en la última ecuación. Llame E_1 a la ecuación resultante. Sustituya la fórmula del antepenúltimo residuo en E_1. Llame E_2 a la ecuación resultante. Sustituya la fórmula del tercer residuo antes del último en E_2. Continúe hasta sustituir la fórmula del primer residuo en la última ecuación E_k.

6. s es el inverso de n mod z si ns mod $z = 1$.

7. Encuentre números s' y t' tales que $s'n + t'\emptyset = 1$. Haga $s = s'$ mod $\emptyset$.

Sección 5.3

1. 90 mod 60 $= 30$; 60 mod 30 $= 0$; así, mcd$(60, 90) = 30$.

4. 825 mod 315 = 195; 315 mod 195 = 120; 195 mod 120 = 75; 120 mod 75 = 45; 75 mod 45 = 30; 45 mod 30 = 15; 30 mod 15 = 0; así mcd (825, 315) = 15.

7. 4807 mod 2091 = 625; 2091 mod 625 = 216; 625 mod 216 = 193; 216 mod 193 = 23; 193 mod 23 = 9; 23 mod 9 = 5; 9 mod 5 = 4; 5 mod 4 = 1; 4 mod 1 = 0; de manera que mcd(2091, 4807) = 1.

10. 490256 mod 337 = 258; 337 mod 258 = 79; 258 mod 79 = 21; 79 mod 21 = 16; 21 mod 16 = 5; 16 mod 5 = 1; 5 mod 1 = 0; de manera que mcd(490256, 337) = 1.

11. (Para el ejercicio 10) Los residuos diferentes de cero en el orden en que los calcula el algoritmo euclidiano son

$$\begin{aligned} 490256 \bmod 337 &= 258 \\ 337 \bmod 258 &= 79 \\ 258 \bmod 79 &= 21 \\ 79 \bmod 21 &= 16 \\ 21 \bmod 16 &= 5 \\ 16 \bmod 5 &= 1. \end{aligned}$$

Al escribir estas ecuaciones en la forma $r = n - dq$, donde r es el residuo y q el cociente, se llega a

$$\begin{aligned} 258 &= 490256 - 337 \cdot 1454 \\ 79 &= 337 - 258 \cdot 1 \\ 21 &= 258 - 79 \cdot 3 \\ 16 &= 79 - 21 \cdot 3 \\ 5 &= 21 - 16 \cdot 1 \\ 1 &= 16 - 5 \cdot 3. \end{aligned}$$

Sustituyendo la penúltima fórmula para 5 en la última ecuación se tiene

$$1 = 16 - (21 - 16 \cdot 1) \cdot 3 = 16 \cdot 4 - 21 \cdot 3.$$

Sustituyendo la antepenúltima fórmula para 16 en la ecuación anterior se tiene

$$1 = (79 - 21 \cdot 3)4 - 21 \cdot 3 = 79 \cdot 4 - 21 \cdot 15.$$

Al sustituir la tercera fórmula antes de la última para 21 en la ecuación anterior se tiene

$$1 = 79 \cdot 4 - (258 - 79 \cdot 3)15 = 79 \cdot 49 - 258 \cdot 15.$$

Sustituyendo la segunda fórmula para 79 en la ecuación anterior se llega a

$$1 = (337 - 258)49 - 258 \cdot 15 = 337 \cdot 49 - 258 \cdot 64.$$

Por último, si se sustituye la primera fórmula para 258 en la ecuación anterior se tiene

$$\begin{aligned} 1 &= 337 \cdot 49 - (490256 - 337 \cdot 1454)64 \\ &= 337 \cdot 93105 - 490256 \cdot 64. \end{aligned}$$

Entonces si se hace $s = -64$ y $t = 93105$,

$$s \cdot 490256 + t \cdot 337 = \gcd(490256, 337) = 1.$$

14.
```
mcd_recurs(a, b) {
    //sea a el mayor
    if (a < b)
        intercambia(a, b)
    if (b == 0)
        return a
    r = a mod b
    return mcd_recurs(b, r)
}
```

Para probar que este algoritmo es correcto, se usa inducción fuerte sobre mín$\{a, b\}$.

Paso base ($n = 0$) Suponga que mín$\{a, b\} = 0$. Si $a < b$, a y b se intercambian, de manera que se supone que $a \geq b$. Entonces $b = 0$. En este caso, el algoritmo regresa a. Como mcd$(a, 0) = a$, el algoritmo es correcto.

Paso inductivo Suponga que $n =$ mín$\{a, b\} > 0$ y para todo $k > n$, si $k =$ mín$\{a', b'\}$ el algoritmo regresa el valor mcd(a', b').

Suponga que a y b son entrada al algoritmo y que el mín$\{a, b\} > 0$. Como antes, si $a < b$, a y b se intercambian, de manera que se puede suponer que $a \geq b$. Como $b \neq 0$, el algoritmo calcula $r = a \bmod b$. Entonces $r < b$. Ahora

$$\text{mín}\ \{a, b\} = b > r = \text{mín}\ \{b, r\}.$$

Por la suposición inductiva, el algoritmo calcula correctamente mcd(b, r). Entonces el algoritmo regresa el valor mcd(b, r). Por el teorema 5.3.2,

$$\text{mcd}(a, b) = \text{mcd}(b, r).$$

Por lo tanto, el algoritmo regresa correctamente el valor de mcd(a, b). El paso inductivo queda completo.

17.
```
mcd_resta(a, b) {
    while (verdadero) {
        //sea a el mayor
        if (a < b)
            intercambia(a, b)
        if (b == 0)
            return a
        a = a - b
    }
}
```

20. Por el teorema 5.3.5, un par a, b, $a > b$, requerirá n operaciones de módulo cuando se alimenta al algoritmo euclidiano sólo si $a \geq f_{n+2}$ y $b \geq f_{n+1}$. Ahora $f_{29} = 514229, f_{30} = 832040$ y $f_{31} = 1346269$. Entonces, ningún par puede requerir más de 28 operaciones de módulo en el peor caso porque 29 operaciones de módulo requerirían que un miembro del par fuera más grande que 1000000. El par 514229, 832040 en sí requiere 28 operaciones de módulo.

23. Se prueba la afirmación por inducción sobre n.

Paso base ($n = 1$) $\text{mcd}(f_1, f_2) = \text{mcd}(1, 1) = 1$

Paso inductivo Suponga que $\text{mcd}(f_n, f_{n+1}) = 1$. Ahora

$$\text{mcd}(f_{n+1}, f_{n+2}) = \text{mcd}(f_{n+1}, f_{n+1} + f_n) = \text{mcd}(f_{n+1}, f_n) = 1$$

Se usa el ejercicio 16 con $a = f_{n+1} + f_n$ y $b = f_{n+1}$ para justificar la segunda igualdad.

26. Si $m = 1$, el resultado es inmediato, así, se supone que $m > 1$.

Suponga que f es uno a uno y sobre. Como $m > 1$, existe x tal que $f(x) = nx \bmod m = 1$. Entonces existe q tal que

$$nx = mq + 1.$$

Sea g el máximo común divisor de m y n. Entonces g divide a ambos, m y n, y $nx - mq = 1$. Por lo tanto, $g = 1$.

Ahora suponga que mcd$(m, n) = 1$. Por el teorema 5.3.7, existen s y t tales que

$$1 = sm + tn.$$

Sea $k \in X$. Entonces

$$k = msk + ntk.$$

Por lo tanto,

$$(ntk) \bmod m = (k - msk) \bmod m = k \bmod m = k.$$

Se puede argumentar, igual que en el apartado dedicado al cálculo del inverso del módulo de un entero, que si se hace $x = tk \bmod m$, entonces $f(x) = (ntk) \bmod m$. Por lo tanto, f es sobre. Como f es una función de X a X, f también es uno a uno.

27. Si $a \neq 0$, $a = 1 \cdot a + 0 \cdot b > 0$. En este caso, $a \in X$. De manera similar, si $b \neq 0$, $b \in X$.

30. Suponga que g no divide a a. Entonces $a = qg + r$, $0 < r < g$. Como $g \in X$, existen s y t tales que $g = sa + tb$. Ahora

$$r = a - qg = a - q(sa + tb) = (1 - qs)a + (-qt)b.$$

Por lo tanto, $r \in X$. Como g es el elemento menor en X y $0 < r < g$, se tiene una contradicción. Por lo tanto, g divide a a. De manera similar, g divide a b.

32. $\text{mcd}(3, 2) = \text{mcd}(2, 1) = \text{mcd}(1, 0) = 1$, $s = 2$

35. $\text{mcd}(47, 11) = \text{mcd}(11, 3) = \text{mcd}(3, 2) = \text{mcd}(2, 1) = \text{mcd}(1, 0) = 1$, $s = 30$

38. $\text{mcd}(243, 100) = \text{mcd}(100, 43) = \text{mcd}(43, 14) = \text{mcd}(14, 1) = \text{mcd}(1, 0) = 1$, $s = 226$

39. Se argumenta por contradicción. Suponga que 6 tiene un módulo inverso 15; es decir, suponga que existe s tal que $6s \bmod 15 = 1$. Entonces existe q tal que

$$15 - 6sq = 1.$$

Como 3 divide a 15 y 3 divide a $6sq$, 3 divide a 1. Ésta es la contradicción deseada. Entonces, 6 no tiene un módulo inverso 15.

El hecho de que 6 no tenga un módulo inverso 15 no contradice el resultado que precede al ejemplo 5.3.9. Para garantizar que n tiene un módulo inverso ϕ, el resultado que precede al ejemplo 5.3.9 requiere que $\text{mcd}(n, \phi) = 1$. En este ejercicio $\text{mcd}(6, 15) = 3$.

Sección 5.4 Repaso

1. Criptología es el estudio de sistemas para comunicaciones seguras.

2. Un sistema criptográfico es un sistema para comunicaciones seguras.

3. Encriptar o cifrar un mensaje significa transformarlo de manera que sólo el receptor autorizado lo pueda reconstruir.

4. Desencriptar o descifrar un mensaje es transformar un mensaje encriptado para que se pueda leer.

5. Calcule $c = a^n \bmod z$ y envíe c.

6. Calcule $c^s \bmod z$. z se elige como el producto de los primos p y q. s satisface $ns \bmod (p - 1)(q - 1) = 1$.

7. La seguridad del criptosistema RSA se basa principalmente en el hecho de que, por ahora, no se conoce un algoritmo eficiente para factorizar enteros.

Sección 5.4

1. OGICAEQAQISK

4. CUIDA EL AGUA

7. $z = pq = 17 \cdot 23 = 391$

10. $c = a^n \bmod z = 101^{31} \bmod 391 = 186$

12. $z = pq = 59 \cdot 101 = 5959$

15. $c = a^n \bmod z = 584^{41} \bmod 5959 = 3237$

Capítulo 5 Autoevaluación

1. Para $d = 2, \ldots, 6$, 539 mod d no es igual a cero, de manera que d se incrementa. Cuando $d = 7$, 539 mod d es igual a cero, de forma que el algoritmo regresa $d = 7$ para indicar que 539 es compuesto y 7 es un divisor de 539.

2. $539 = 7^2 \cdot 11$

3. $7^2 \cdot 13^2$

4. $2 \cdot 5^2 \cdot 7^4 \cdot 13^4 \cdot 17$

5. 150

6. 110101110, 1AE

7. El algoritmo comienza por hacer el *resultado* igual a 1 y x igual a a. Como $n = 30 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 no es igual a 1, el *resultado* no se modifica. x se convierte en a^2 y n es 15.

Como $n = 15 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 es igual a 1, el *resultado* es ahora *resultado* $* x = 1 * a^2 = a^2$. x se convierte en a^4 y n es 7.

Como $n = 7 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 es igual a 1, el *resultado* es ahora *resultado* $* x = a^2 * a^4 = a^6$. x se convierte en a^8 y n es 3.

Como $n = 3 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 es igual a 1, el *resultado* es ahora *resultado* $* x = a^6 * a^8 = a^{14}$. x se convierte en a^{16} y n es 1.

Como $n = 1 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 es igual a 1, el *resultado* es ahora *resultado* $* x = a^{14} * a^{16} = a^{30}$. x se convierte en a^{32} y n es 0.

Como $n = 0$ no es mayor que 0, el ciclo "while" termina. El algoritmo regresa *resultado*, que es igual a a^{30}.

8. El algoritmo comienza por hacer el *resultado* igual a 1 y x igual a $a \bmod z = 50 \bmod 11 = 6$. Como $n = 30 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 no es igual a 1, el *resultado* no se modifica. x se convierte en $x * x \bmod z = 36 \bmod 11 = 3$ y n es 15.

Como $n = 15 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 es igual a 1, el *resultado* es ahora (*resultado* $* x$) mod $z = 3 \bmod 11 = 3$. x se convierte en $x * x \bmod z = 9 \bmod 11 = 9$ y n es 7.

Como $n = 7 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 es igual a 1, el *resultado* es ahora (*resultado* $* x$) mod $z = 27 \bmod 11 = 5$. x se convierte en $x * x \bmod z = 81 \bmod 11 = 4$ y n es 3.

Como $n = 3 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 es igual a 1, el *resultado* es ahora (*resultado* $* x$) mod $z = 20 \bmod 11 = 9$. x se convierte en $x * x \bmod z = 16 \bmod 11 = 5$ y n es 1.

Como $n = 1 > 0$, el cuerpo del ciclo "while" se ejecuta. Como n mod 2 es igual a 1, el *resultado* es ahora (*resultado* $* x$) mod $z = 45 \bmod 11 = 1$. x se convierte en $x * x \bmod z = 25 \bmod 11 = 3$ y n es 0.

Como $n = 0$ no es más grande que 0, el ciclo "while" termina. El algoritmo regresa *resultado*, que es igual a $a^n \bmod z = 50^{30} \bmod 11 = 1$.

9. $\text{mcd}(480, 396) = \text{mcd}(396, 84) = \text{mcd}(84, 60) = \text{mcd}(60, 24) = \text{mcd}(24, 12) = \text{mcd}(12, 0) = 12$

10. Dado que

$$\begin{aligned}\log_{3/2} \frac{2(100{,}000{,}000)}{3} &= \log_{3/2} 100^4 + \log_{3/2} \frac{2}{3} \\ &= (4 \log_{3/2} 100) - 1 \\ &= 4(11.357747) - 1 = 44.430988,\end{aligned}$$

una cota superior para el número de operaciones de módulo requeridas por el algoritmo euclidiano para enteros en el intervalo de 0 a 100,000,000 es 44.

11. Los residuos diferentes de cero en el orden en que los calcula el

algoritmo euclidiano son

$$480 \bmod 396 = 84$$
$$396 \bmod 84 = 60$$
$$84 \bmod 60 = 24$$
$$60 \bmod 24 = 12.$$

Al escribir estas ecuaciones en la forma $r = n - dq$, donde r es el residuo y q el cociente, se llega a

$$84 = 480 - 396 \cdot 1$$
$$60 = 396 - 84 \cdot 4$$
$$24 = 84 - 60 \cdot 1$$
$$12 = 60 - 24 \cdot 2.$$

Sustituyendo la penúltima fórmula para 24 en la última ecuación se tiene

$$12 = 60 - 24 \cdot 2 = 60 - (84 - 60) \cdot 2 = 3 \cdot 60 - 2 \cdot 84.$$

Sustituyendo la segunda fórmula para 60 en la ecuación anterior se tiene

$$12 = 3 \cdot (396 - 84 \cdot 4) - 2 \cdot 84 = 3 \cdot 396 - 14 \cdot 84.$$

Por último, sustituyendo la primera fórmula para 84 en la ecuación anterior se tiene

$$12 = 3 \cdot 396 - 14 \cdot (480 - 396) = 17 \cdot 396 - 14 \cdot 480.$$

Entonces, si se establece que $s = 17$ y $t = -14$,

$$s \cdot 396 + t \cdot 480 = \text{mcd}(396, 480) = 12.$$

12. Los residuos diferentes de cero en el orden en que los calcula el algoritmo euclidiano son

$$425 \bmod 196 = 33$$
$$196 \bmod 33 = 31$$
$$33 \bmod 31 = 2$$
$$31 \bmod 2 = 1.$$

Al escribir estas ecuaciones en la forma $r = n - dq$, donde r es el residuo y q el cociente, se llega a

$$33 = 425 - 196 \cdot 2$$
$$31 = 196 - 33 \cdot 5$$
$$2 = 33 - 31 \cdot 1$$
$$1 = 31 - 2 \cdot 15.$$

Sustituyendo la penúltima fórmula para 2 en la última ecuación se tiene

$$1 = 31 - (33 - 31) \cdot 15 = 16 \cdot 31 - 15 \cdot 33.$$

Sustituyendo la segunda fórmula para 31 en la ecuación anterior se tiene

$$1 = 16 \cdot (196 - 33 \cdot 5) - 15 \cdot 33 = 16 \cdot 196 - 95 \cdot 33.$$

Por último, sustituyendo la primera fórmula para 33 en la ecuación anterior se tiene

$$1 = 16 \cdot 196 - 95 \cdot (425 - 196 \cdot 2) = 206 \cdot 196 - 95 \cdot 425.$$

Entonces, si se establece que $s' = 206$ y $t' = -95$,

$$s' \cdot 196 + t' \cdot 425 = \text{mcd}(196, 425) = 1.$$

Por lo tanto, $s = s' \bmod 425 = 206 \bmod 425 = 206$.

13. $z = pq = 13 \cdot 17 = 221,\ \phi = (p-1)(q-1) = 12 \cdot 16 = 192$

14. $s = 91$

15. $c = a^n \bmod z = 144^{19} \bmod 221 = 53$

16. $a = c^s \bmod z = 28^{91} \bmod 221 = 63$

Sección 6.1 Repaso

1. Si una actividad se puede construir en t pasos sucesivos y el paso 1 se puede realizar de n_1 maneras, el paso 2 se puede realizar de n_2 maneras, . . . , y el paso t se puede realizar de n_t maneras, entonces el número de actividades diferentes posibles es $n_1 \cdot n_2 \cdot \cdots n_t$. Como ejemplo, si se tienen dos opciones de aperitivo y cuatro para el plato principal, el número total de cenas es $2 \cdot 4 = 8$.

2. Suponga que $X_1, \ldots, X_t$ son conjuntos y que el i-ésimo conjunto X_i tiene n_i elementos. Si $\{X_1, \ldots, X_t\}$ es una familia de conjuntos ajenos por pares, el número de elementos posibles que se pueden seleccionar de X_1 o X_2 o . . . o X_t es $n_1 + n_2 + \cdots + n_t$. Como ejemplo suponga que dentro de un conjunto de cadenas, dos comienzan con a y cuatro comienzan con b. Entonces $2 + 4 = 6$ comienzan con a o b.

Sección 6.1

1. $2 \cdot 4$ **4** $8 \cdot 4 \cdot 5$ **8.** 6^2

11. $6 + 2$ **14.** 11 **17.** $10 \cdot 5$

20. $26^3 10^2$, $26 \cdot 25 \cdot 24 \cdot 10 \cdot 9$

23. $3 \cdot 2^6$ **26.** $2^8 - 1$ **28.** $5 \cdot 4 \cdot 3$

31. $3 \cdot 4 \cdot 3$ **34.** 5^3 **37.** $4 \cdot 3$

40. $5^3 - 4^3$ **42.** $200 - 5 + 1$ **45.** 40

48. Un número de un dígito contiene a 7. Los números distintos de dos dígitos que contienen a 7 son 17, 27, . . . , 97 y 70, 71, . . . , 76, 78, 79. Hay 18 de estos números. Los números distintos de tres dígitos que contiene a 7 son 107 y 1xy, donde xy es uno de los números de dos dígitos listados antes. La respuesta es $1 + 18 + 19$.

51. $5 + (8 + 7 + \cdots + 1) + (7 + 6 + \cdots + 1)$

54. 10! **57.** (3!)(5!)(2!)(3!) **61.** 2^{10}

64. Se cuenta el número de relaciones antisimétricas en $\{1, 2, \ldots, n\}$ calculando el número de maneras de construir una matriz de una relación antisimétrica.

Cada elemento de la diagonal puede ser 0 o 1. Entonces hay 2^n maneras de asignar valores a la diagonal.

Para i y j que satisfacen $1 \le i < j \le n$, se pueden asignar los elementos en el renglón i y la columna j, y en el renglón j y la columna i de tres maneras:

Renglón i, columna j	*Renglón j, columna i*
0	0
1	0
0	1

Como hay $(n^2 - n)/2$ valores de i y j que satisfacen $1 \le i < j \le n$, se pueden asignar los valores fuera de la diagonal de $3^{(n^2-n)/2}$ maneras. Por lo tanto, hay

$$2^n \cdot 3^{(n^2-n)/2}$$

relaciones antisimétricas en un conjunto de n elementos.

66. $2^5 + 2^7 - 2^4$ (Según el ejercicio 65, el número total de posibilidades = número de cadenas que comienzan con 100 + número de cadenas cuyo cuarto bit es 1 − número de cadenas que comienzan con 100 y cuyo cuarto bit es 1.)

69. $5 \cdot 4 + 3 \cdot 5 \cdot 4 - 2 \cdot 4$ (Según el ejercicio 65, el número total de posibilidades = número en que Consuelo puede ser presidente

+ número en que Alicia puede tener un puesto − número en que Consuelo es presidente y Alicia tiene un puesto).

Sección 6.2 Repaso

1. Un ordenamiento de $x_1, \ldots, x_n$
2. Existen $n!$ permutaciones de un conjunto de n elementos. Hay n maneras de elegir el primer elemento, $n-1$ maneras de elegir el segundo, etcétera. Por lo tanto, el número total de permutaciones es
$$n(n-1)\cdots 2\cdot 1 = n!.$$
3. Un ordenamiento de r elementos seleccionados entre $x_1, \ldots, x_n$.
4. Hay $n(n-1)\cdots(n-r+1)$ permutaciones r de un conjunto de n elementos. Hay n maneras de elegir el primer elemento, $n-1$ maneras de elegir el segundo . . . , y $n-r+1$ maneras de elegir el elemento r. Por lo tanto, el número total de permutaciones r es
$$n(n-1)\cdots(n-r+1).$$
5. $P(n, r)$
6. Un subconjunto de r elementos de $\{x_1, \ldots, x_n\}$
7. Existen
$$\frac{n!}{(n-r)!r!}$$
combinaciones r de un conjunto de n elementos.

   Hay $P(n, r)$ maneras de seleccionar una permutación r de un conjunto de n elementos. Esta permutación r también se puede construir eligiendo primero una combinación r [$C(n, r)$ maneras] y después ordenándola de [$r!$ maneras]. Por lo tanto, $P(n, r) = C(n, r)r!$. Entonces
$$C(n,r) = \frac{P(n,r)}{r!} = \frac{n(n-1)\cdots(n-r+1)}{r!} = \frac{n!}{(n-r)!r!}.$$
8. $C(n, r)$

Sección 6.2

1. $4! = 24$
4. *abc*, *acb*, *bac*, *bca*, *cab*, *cba*, *abd*, *adb*, *bad*, *bda*, *dab*, *dba*, *acd*, *adc*, *cad*, *cda*, *dac*, *dca*, *bcd*, *bdc*, *cbd*, *cdb*, *dbc*, *dcb*
7. $P(11, 3) = 11\cdot 10\cdot 9$ **10.** 3!
13. 4! contiene la subcadena *AE* y 4! contiene la subcadena *EA*; por lo tanto, el número total es $2\cdot 4!$.
16. Primero se cuenta el número N de cadenas que contienen la subcadena *AB* o la subcadena *BE*. La respuesta al ejercicio será: Número total de cadenas $-N$ o $5! - N$.

    Según el ejercicio 65, sección 6.1, el número de cadenas que contienen *AB* o *BE* = número de cadenas que contienen *AB* + número de cadenas que contienen *BE* − número de cadenas que contienen *AB* y *BE*. Una cadena contiene *AB* y *BE* si y sólo si contiene *ABE* y el número de estas cadenas es 3!. El número de cadenas que contienen *AB* = número de cadenas que contienen *BE* = 4!. Entonces, el número de cadenas que contienen *AB* o *BE* es $4! + 4! - 3!$. La solución al ejercicio es
$$5! - (2\cdot 4! - 3!).$$
19. $8!P(9, 5) = 8!(9\cdot 8\cdot 7\cdot 6\cdot 5)$
21. 10!
24. Se fija un asiento para un venusino. Hay 7! arreglos para los venusinos restantes. Por cada uno de estos arreglos, se pueden colocar los marcianos en cinco de las ocho posiciones intermedias, esto se puede hacer de $P(8, 5)$ maneras. Así que hay $7!P(8, 5)$ arreglos.
25. $C(4, 3) = 4$ **28.** $C(11, 3)$ **31.** $C(13, 5)$
34. Un comité que tiene cuando mucho un hombre tiene exactamente un hombre o ninguno. Hay $C(6, 1)C(7, 3)$ comités con exactamente un hombre. Hay $C(7, 4)$ comités sin hombres. Entonces la respuesta es $C(6, 1)C(7, 3) + C(7, 4)$.
37. $C(10, 4)C(12, 3)C(4, 2)$
40. Primero, se cuenta el número de cadenas de ocho bits sin dos ceros seguidos. El problema se divide en contar el número de tales cadenas con exactamente ocho unos, con exactamente siete unos, etcétera.

    Hay una cadena de 8 bits sin dos ceros seguidos que tiene exactamente ocho unos. Suponga que una cadena de 8 bits sin dos ceros seguidos tiene exactamente siete unos. El 0 se puede colocar en cualquiera de las ocho posiciones; así que hay ocho de esas cadenas. Suponga que una cadena de 8 bits sin dos ceros seguidos tiene exactamente seis unos. Los dos ceros deben colocarse en dos de los espacios mostrados:
    _1_1_1_1_1_1_·
    Entonces los dos ceros se pueden colocar de $C(7, 2)$ maneras. Entonces hay $C(7, 2)$ cadenas de éstas. De manera similar, hay $C(6, 3)$ cadenas de 8 bits sin dos ceros seguidos que tienen exactamente cinco unos y hay $C(5, 4)$ cadenas de 8 bits sin dos ceros seguidos que tienen exactamente cuatro unos seguidos. Si una cadena tiene menos de cuatro unos, tendrá dos ceros seguidos. Por lo tanto, el número de cadenas de 8 bits sin dos ceros seguidos es
$$1 + 8 + C(7, 2) + C(6, 3) + C(5, 4).$$
    Como hay 2^8 cadenas de 8 bits, se tienen
$$2^8 - [1 + 8 + C(7, 2) + C(6, 3) + C(5, 4)]$$
    cadenas de 8 bits que contienen al menos dos ceros seguidos.
41. $1\cdot 48$ (Los cuatro ases se pueden elegir de una manera y la quinta carta se puede elegir de 48 maneras).
44. Primero se cuenta el número de manos que contienen cartas de espadas y corazones. Como hay 26 espadas y corazones, hay $C(26, 5)$ maneras de seleccionar cinco cartas entre estas 26. Sin embargo, $C(13, 5)$ contienen sólo espadas y $C(13, 5)$ contienen sólo corazones. Por lo tanto, hay
$$C(26, 5) - 2C(13, 5)$$
    maneras de seleccionar cinco cartas que contienen espadas y corazones.

    Como hay $C(4, 2)$ maneras de seleccionar los dos palos, el número de manos que contienen cartas de exactamente dos palos es
$$C(4, 2)[C(26, 5) - 2C(13, 5)].$$
47. Hay nueve patrones consecutivos: A2345, 23456, 34567 45678, 56789, 6789T, 789TJ, 89TJQ, 9TJQK. Correspondientes a los cuatro palos posibles, hay cuatro maneras para que ocurra cada patrón. Entonces hay $9\cdot 4$ manos que son consecutivas y del mismo palo.
50. $C(52, 13)$
53. $1\cdot C(48, 9)$ (Se seleccionan los ases, después se seleccionan las nueve cartas restantes).
56. Hay $C(13, 4)C(13, 4)C(13, 4)C(13, 1)$ manos que contienen cuatro espadas, cuatro corazones, cuatro diamantes y un trébol. Como hay cuatro maneras de seleccionar los tres palos para que tengan cuatro cartas cada uno, hay $C(13, 4)^3C(13, 1)$ manos que contienen cuatro cartas de tres palos y una carta del cuarto palo.
58. 2^{10} **61.** 2^9 **63.** $C(50, 4)$

66. $C(50, 4) - C(46, 4)$ (Número total: número sin defectuosos).

70. Se ordenan los $2n$ objetos. El primer objeto se puede aparear de $2n - 1$ maneras. El siguiente objeto (todavía no seleccionado) puede aparearse de $2n - 3$ maneras, y así sucesivamente.

71. Una lista de votos en la que Ramírez nunca está detrás de Uribe y cada uno recibe r votos es una cadena de símbolos rW y rU donde, al leer la cadena de izquierda a derecha, el número de W siempre es mayor que el número de U. Esta cadena también se puede considerar una trayectoria del tipo descrito en el ejemplo 6.2.23, donde W es un movimiento a la derecha y U es un movimiento hacia arriba. El ejemplo 6.2.23 demostró que hay C_r de estas trayectorias. Por lo tanto, el número de maneras para contar los votos en los que Ramírez nunca está detrás de Uribe es C_r.

74. Por el ejercicio 73, pueden ocurrir k pasos verticales de $C(k, \lceil k/2 \rceil)$ maneras puesto que, en cualquier momento, el número de pasos hacia arriba es mayor o igual que el número pasos hacia abajo. Entonces, se pueden insertar $n - k$ pasos horizontales entre los k pasos verticales de $C(n, k)$ maneras. Como cada paso horizontal puede ocurrir de dos maneras, el número de trayectorias que contienen exactamente k pasos verticales que nunca van estrictamente abajo del eje x es

$$C(k, \lceil k/2 \rceil)C(n, k)2^{n-k}.$$

Sumando sobre k, se encuentra que el número total de trayectorias es

$$\sum_{k=0}^{n} C(k, \lceil k/2 \rceil)C(n, k)2^{n-k}.$$

80. La solución cuenta manos *ordenadas*.

83. Use los teoremas 3.2.1 y 3.2.8.

Sección 6.3 Repaso

1. Sean $\alpha = s_1, \ldots, s_p$ y $\beta = t_1, \ldots, t_q$ cadenas en $\{1, 2, \ldots, n\}$. Entonces α es menor que β en el orden lexicográfico si $p < q$ y $s_i = t_i$ para toda $i = 1, \ldots, p$, o bien para alguna i, $s_i \neq t_i$ y para la i más pequeña se tiene $s_i < t_i$.

2. Dada la cadena $s_1 \ldots s_r$, que representa la combinación r $\{s_1, \ldots, s_r\}$, para encontrar la siguiente cadena $t_1 \ldots t_r$, se encuentra el elemento de la extrema derecha s_m que no tiene su valor máximo. (El valor máximo de s_r es n, el valor máximo de s_{r-1} es $n - 1$, etcétera). Entonces se hace $t_i = s_i$ para $i = 1, \ldots, m - 1$; se hace $t_m = s_m + 1$; y se hace $t_{m+1} \cdots t_r = (s_m + 2)(s_m + 3) \cdots$. Se comienza con la cadena $12 \cdots r$.

3. Dada una cadena s, que representa una permutación, para encontrar la siguiente cadena, se encuentra el dígito de la derecha d de s cuyo vecino a la derecha exceda a d. Se encuentra el elemento más a la derecha r que satisface $d < r$. Se intercambian d y r. Por último, se invierte la subcadena a la derecha de la posición original de d. Se comienza con una cadena $12 \cdots n$.

Sección 6.3

1. 1357

4. 12435

7. (Para el ejercicio 1) En las líneas 8 a la 12, hasta la derecha se encuentra s_m no en su valor máximo. En este caso, $m = 4$. En la línea 14, se incrementa s_m. Esto hace que el último dígito sea 7. Como m es la posición de la extrema derecha, en las líneas 16 y 17, no se hace nada. La siguiente combinación es 1357.

9. 123, 124, 125, 126, 134, 135, 136, 145, 146, 156, 234, 235, 236, 245, 246, 256, 345, 346, 356, 456

12. 12, 21

14. Entrada: r, n

Salida: Lista de todas la combinaciones r de $\{1, 2, \ldots, n\}$ en orden lexicográfico creciente

```
comb_r(r, n)
  s_0 = -1
 for i = 1 to r
   s_i = i
 println(s_1, . . . , s_n)
  while (verdadero) {
  m = r
  máx_val = n
  while (s_m == máx_val) {
    m = m − 1
    máx_val = máx_val − 1
  }
  if (m == 0)
    return
  s_m = s_m + 1
  for j = m + 1 to r
    s_j = s_{j−1} + 1
  println(s_1, . . . , s_n)
 }
}
```

17. Entrada: $s_1, \ldots, s_r$ (una combinación r de $\{1, \ldots, n\}$, r y n

Salida: $s_1, \ldots, s_r$, la siguiente combinación r (La primera combinación r sigue a la última combinación r).

```
comb_sig(s, r, n) {
    s_0 = n + 1 //valor ficticio
    m = r
    máx_val = n
    //la prueba del ciclo siempre falla si m = 0
    while(s_n == máx_val) {
        //encuentra elemento hasta la derecha no en su valor
        máximo
        m = m − 1
        máx_val = máx_val − 1
    }
    if(m == 0) //última combinación r detectada
        s_1 = 0
        m = 1
    }
    //se incrementa el elemento más a la derecha
    s_m = s_m + 1
    //demás elementos son sucesores de s_m
    for j = m + 1 to r
        s_j = s_{j−1} + 1
}
```

19. Entrada: $s_1, \ldots, s_r$ (una combinación r de $\{1, \ldots, n\}$, r y n

Salida: $s_1, \ldots, s_r$, la combinación r anterior (La última combinación r precede a la primera combinación r.)

```
comb_anterior(s, r, n) {
    s_0 = n //valor ficticio
    //encuentra el elemento más a la derecha
    //al menos 2 más grande que su vecino a la izquierda
    m = r
    //la prueba del ciclo siempre falla si m = 1
    while (s_m − s_{m−1} == 1)
        m = m − 1
    s_m = s_m − 1
    if(m == 1 ∧ s_1 == 0)
        m = 0
    //los elementos a la derecha del índice m se igualan al valor máximo
    for j = m + 1 to r
        s_j = n + j − r
}
```

21. Entrada: r, s_k, s_{k+1}, . . . , s_n, una cadena α, k y n
Salida: Lista de todas las combinaciones r de $\{s_k, s_{k+1}, \ldots, s_n\}$ cada una con α como prefijo [Para listar todas las combinaciones de $\{s_1, s_2, \ldots, s_n\}$ se invoca esta función como *comb_r2*(r, s, 1, n, λ), donde λ es la cadena nula.]

```
comb_r2(r, s, k, n, α) {
    if (r == 0) {
        println(α)
        return
    }
    if (r == n){
        println(α, s_k, s_{k+1}, . . . , s_n)
        return
    }
    //imprime las combinaciones r que contienen a s_k
    β = α + "" + s_k //concatenación
    comb_r2(r − 1, s, k + 1, n − 1, β)
    //imprime las combinaciones r que no contienen a s_k
    comb_r2(r, s, k + 1, n − 1, α)
}
```

Sección 6.4 Repaso

1. Un experimento es un proceso que lleva a un resultado.

2. Un evento es un resultado o combinación de resultados de un experimento.

3. El espacio muestra es el evento que consiste en todos los resultados posibles.

4. El número de resultados en el evento dividido entre el número de resultados en el espacio muestra.

Sección 6.4

1. (H, 1), (H, 2), (H, 3), (H, 4), (H, 5), (H, 6), (T, 1), (T, 2), (T, 3), (T, 4), (T, 5), (T, 6)

4. (H, 1), (H, 2), (H, 3)

5. (1, 1), (1, 3), (1, 5), (2, 2), (2, 4), (2, 6), (3, 1), (3, 3), (3, 5), (4, 2), (4, 4), (4, 6), (5, 1), (5, 3), (5, 5), (6, 2), (6, 4), (6, 6)

8. Se lanzan tres dados.

11. 1/6

14. 1/52

17. 4/36

20. $C(90, 4)/C(100, 4)$

23. $1/10^3$

26. $1/[C(50, 5) \cdot 36]$

28. $$\frac{4 \cdot C(13, 5) \cdot 3 \cdot C(13, 4)C(13, 2)^2}{C(52, 13)}$$

30. $1/2^{10}$

33. $C(10, 5)/2^{10}$

34. $2^{10}/3^{10}$

37. 1/5!

38. $10/C(12, 3)$

41. 18/38

44. 2/38

45. 1/3

48. 1/4

51. Las posibilidades son: A (correcto), B (incorrecto), C (incorrecto); y A (incorrecto), B (incorrecto), C (incorrecto). En el primer caso, si el estudiante se queda con A, la respuesta será correcta; pero si el estudiante cambia a B, la respuesta será incorrecta. En el segundo caso, si el estudiante se queda con A, la respuesta será incorrecta; pero si cambia a B, la respuesta será correcta. Así que la probabilidad de una repuesta correcta es de 1/2.

Sección 6.5 Repaso

1. Una función de probabilidad P asigna a cada resultado x, en un espacio muestra S, un número $P(x)$ de manera que

$$0 \le P(x) \le 1 \qquad \text{para toda } x \in S$$

y

$$\sum_{x \in S} P(x) = 1.$$

2. $P(x) = 1/n$, donde n es el tamaño del espacio muestra.

3. La probabilidad de E es

$$P(E) = \sum_{x \in E} P(x).$$

4. $P(E) + P(\overline{E}) = 1$

5. E_1 o E_2 (o ambos)

6. E_1 y E_2

7. $P(E_1 \cup E_2) = P(E_1) + P(E_2) - P(E_1 \cap E_2)$. $P(E_1) + P(E_2)$ es igual a $P(x)$ para toda $x \in E_1$ más $P(x)$ para toda $x \in E_2$, que es igual a $P(E_1) + P(E_2)$, excepto que $P(x)$, para $x \in E_1 \cap E_2$ está contado dos veces. De aquí se deduce la fórmula.

8. Los eventos E_1 y E_2 son mutuamente excluyentes si $E_1 \cap E_2 = \emptyset$.

9. Si se lanzan dos dados, los eventos "dobles" y "la suma es impar" son mutuamente excluyentes.

10. $P(E_1 \cup E_2) = P(E_1) + P(E_2)$. Esta fórmula se deduce de la fórmula del ejercicio 7 porque $P(E_1 \cap E_2) = 0$.

11. *E dado F* es el evento *E,* dado que ocurrió el evento *F*.

12. $E \mid F$

13. $P(E \mid F) = P(E \cap F)/P(F)$

14. Los eventos E y F son independientes si $P(E \cap F) = P(E)P(F)$.

15. Si se lanzan dos dados, los eventos "sale un número impar en el primer dado" y "sale un número par en el segundo dado" son independientes.

16. El reconocimiento de patrones coloca los artículos en varias clases basadas en las características de los artículos.

17. Suponga que las clases posibles son $C_1, \ldots, C_n$. Suponga aún más que cada par de clases es mutuamente excluyente y cada artículo que se debe clasificar pertenece a una de las clases. Para un conjunto de características F, se tiene

$$P(C_j \mid F) = \frac{P(F \mid C_j)P(C_j)}{\sum_{i=1}^{n} P(F \mid C_i)P(C_i)}.$$

La ecuación

$$P(C_j \mid F) = \frac{P(C_j \cap F)}{P(F)} = \frac{P(F \mid C_j)P(C_j)}{P(F)}$$

se deduce a partir de la definición de probabilidad condicional. La demostración queda completa si se demuestra que

$$P(F) = \sum_{i=1}^{n} P(F \mid C_i)P(C_i),$$

que se deduce del hecho de que cada par de clases es mutuamente excluyente y cada artículo que se debe clasificar pertenece a una de las clases.

Sección 6.5

1. 1/8

4. $P(2) = P(4) = P(6) = 1/12$. $P(1) = P(3) = P(5) = 3/12$

7. $1 - (1/4)$

8. $3(1/12)^2 + 3(3/12)^2$

11. Sea E el evento "la suma es 6" y sea F el evento "al menos un dado muestra 2". entonces

$$P(E \cap F) = P((2,4)) + P((4,2)) = 2\left(\frac{1}{12}\right)^2 = \frac{2}{144},$$

y

$$\begin{aligned} P(F) &= P((1,2)) + P((2,1)) + P((2,2)) + P((2,3)) \\ &\quad + P((2,4)) + P((2,5)) + P((2,6)) + P((3,2)) \\ &\quad + P((4,2)) + P((5,2)) + P((6,2)) \\ &= \left(\frac{3}{12}\right)\left(\frac{1}{12}\right) + \left(\frac{1}{12}\right)\left(\frac{3}{12}\right) + \left(\frac{1}{12}\right)^2 \\ &\quad + \left(\frac{1}{12}\right)\left(\frac{3}{12}\right) + \left(\frac{1}{12}\right)^2 + \left(\frac{1}{12}\right)\left(\frac{3}{12}\right) \\ &\quad + \left(\frac{1}{12}\right)^2 + \left(\frac{3}{12}\right)\left(\frac{1}{12}\right) + \left(\frac{1}{12}\right)^2 \\ &\quad + \left(\frac{3}{12}\right)\left(\frac{1}{12}\right) + \left(\frac{1}{12}\right)^2 = \frac{23}{144}. \end{aligned}$$

Por lo tanto,

$$P(E \mid F) = \frac{P(E \cap F)}{P(F)} = \frac{\frac{2}{144}}{\frac{23}{144}} = \frac{2}{23}.$$

14. (T,1), (T,2), (T,3), (T,4), (T,5), (T,6), (H,3)

17. Sí **19.** $C(90,6)/C(100,6)$

22. $1/2^4$

25. $\dfrac{\frac{1}{2^4}}{\frac{2^4-1}{2^4}} = \dfrac{1}{15}$

28. Sea E_1 el evento "niños de uno y otro sexo" y sea E_2 el evento "cuando mucho un niño". Entonces

$$P(E_1) = \frac{14}{16}, \quad P(E_2) = \frac{5}{16}, \quad \text{y} \quad P(E_1 \cap E_2) = \frac{4}{16}.$$

Ahora

$$P(E_1 \cap E_2) = \frac{1}{4} \neq \frac{35}{128} = P(E_1)P(E_2).$$

Por lo tanto, los eventos E_1 y E_2 no son independientes.

31. $1/2^{10}$ **34.** $1 - (1/2^{10})$

37. Sea E el evento "cuatro de cinco o seis caras" y sea F el evento "al menos una cara". Entonces

$$\begin{aligned} P(E \cap F) &= \frac{C(10,4)}{2^{10}} + \frac{C(10,5)}{2^{10}} + \frac{C(10,6)}{2^{10}} \\ &= \frac{210 + 252 + 210}{2^{10}} = 0.65625. \end{aligned}$$

Como $P(F) = 1 - (1/2^{10}) = 0.999023437$,

$$P(E \mid F) = \frac{P(E \cap F)}{P(F)} = \frac{0.65625}{0.999023437} = 0.656891495.$$

40. Sea E_1 el evento "más de 350 libras" y sea E_2 el evento "chico malo". Entonces

$$\begin{aligned} P(E_1 \cup E_2) &= P(E_1) + P(E_2) - P(E_1 \cap E_2) \\ &= \frac{35}{90} + \frac{20}{90} - \frac{15}{90} = \frac{40}{90}. \end{aligned}$$

42. $P(A) = 0.55$, $P(D) = 0.10$, $P(N) = 0.35$

45. $$\begin{aligned} P(B) &= P(B \mid A)P(A) + P(B \mid D)P(D) + P(B \mid N)P(N) \\ &= (0.10)(0.55) + (0.30)(0.10) + (0.30)(0.35) = 0.19 \end{aligned}$$

46. Se requiere que

$$P(H \mid Pos) = 0.5 = \frac{(0.95)P(H)}{(0.95)P(H) + (0.02)(1 - P(H))}.$$

Al despejar $P(H)$ se obtiene $P(H) = .0206$.

49. Sí. Suponga que E y F son independientes, es decir, $P(E)P(F) = P(E \cap F)$. Ahora

$$\begin{aligned} P(\overline{E})P(\overline{F}) &= (1 - P(E))(1 - P(F)) \\ &= 1 - P(E) - P(F) + P(E)P(F) \\ &= 1 - P(E) - P(F) + P(E \cap F). \end{aligned}$$

Por las leyes de De Morgan para conjuntos,

$$\overline{E} \cap \overline{F} = \overline{E \cup F};$$

entonces,

$$\begin{aligned} P(\overline{E} \cap \overline{F}) &= P(\overline{E \cup F}) \\ &= 1 - P(E \cup F) \\ &= 1 - [P(E) + P(F) - P(E \cap F)] \\ &= 1 - P(E) - P(F) + P(E \cap F). \end{aligned}$$

Por lo tanto,

$$P(\overline{E})P(\overline{F}) = P(\overline{E} \cap \overline{F}),$$

y $\overline{E}$ y $\overline{F}$ son independientes.

52. Sea E_i el evento "el atleta termina la maratón en el intento i". El error en el razonamiento es suponer que $P(E_2) = 1/3 = P(E_3)$. De hecho, $P(E_2) \neq 1/3 \neq P(E_3)$ porque, si el atleta termina la maratón, no vuelve a hacer la carrera. Aunque $P(E_1) = 1/3$,

$$\begin{aligned} P(E_2) &= P(\text{fallar intento 1 y completar intento 2}) \\ &= P(\text{fallar intento 1})P(\text{completar intento 2}) \\ &= \frac{2}{3} \cdot \frac{1}{3} = \frac{2}{9} \end{aligned}$$

De manera similar,

$$P(E_3) = \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{1}{3} = \frac{4}{27}.$$

Entonces, la probabilidad de completar la maratón es

$$\begin{aligned} P(E_1 \cup E_2 \cup E_3) &= P(E_1) + P(E_2) + P(E_3) \\ &= \frac{1}{3} + \frac{2}{9} + \frac{4}{27} = \frac{19}{27} = 0.704, \end{aligned}$$

lo que significa que existe una oportunidad cercana al 70% de que el atleta termine la maratón (¡no exactamente una certidumbre virtual!).

Sección 6.6 Repaso

1. $n!/(n_1! \cdots n_t!)$.. La fórmula se obtiene del principio de la multiplicación. Primero se asignan posiciones a los n_1 artículos del tipo 1, lo que se puede hacer de $C(n, n_1)$ maneras. Una vez hechas estas asignaciones, se asignan posiciones a los n_2 artículos de tipo 2, lo que se hace de $C(n - n_1, n_2)$ maneras, etcétera. El número de ordenamientos es entonces

$$C(n, n_1)C(n - n_1, n_2) \cdots C(n - n_1 - \cdots - n_{t-1}, n_t),$$

lo cual, después de aplicar la fórmula para $C(n, k)$ y simplificar, da $n!/(n_1! \cdots n_t!)$.

2. $C(k + t - 1, t - 1)$. La fórmula se obtiene considerando $k + t - 1$ ranuras y $k + t - 1$ símbolos que consisten en k símbolos $\times$ y $t - 1$ símbolos $|$. Cada colocación de estos símbolos en las ranuras determina una selección. El número n_1 de $\times$ entre la primera y la

segunda | representa n_1 copias del primer elemento en el conjunto. El número n_2 de $\times$ entre la segunda y tercera | representa n_2 copias del segundo elemento en el conjunto, y así sucesivamente. Como hay $C(k+t-1, t-1)$ maneras de seleccionar las posiciones de las |, también hay $C(k+t-1, t-1)$ selecciones.

Sección 6.6

1. 5!

4. Se permuta una ficha con cuatro letras *S* y otras fichas con una letra cada una entre *ALEPERON*, lo cual se puede hacer de 9!/2! maneras.

7. $C(6+6-1, 6-1)$

10. Cada ruta se puede diseñar mediante una cadena de *i* letras *X*, *j* letras *Y* y *k* letras *Z*, donde una *X* significa moverse una unidad en la dirección *x*, una *Y* significa moverse una unidad en la dirección *y* y una *Z* quiere decir moverse una unidad en la dirección *z*. Existen

$$\frac{(i+j+k)!}{i!j!k!}$$

cadenas de este tipo.

14. $10!/(5!\cdot 3!\cdot 2!)$

15. $C(10+3-1, 10)$ **18.** $C(9+2-1, 9)$

21. Cuatro, como las posibilidades son (0, 0), (2, 1), (4, 2) y (6, 3), donde el par (r, v) designa *r* pelotas rojas y *v* verdes.

22. $C(15+3-1, 15)$ **25.** $C(13+2-1, 13)$

28. $C(12+4-1, 12)$
$- [C(7+4-1, 7) + C(6+4-1, 6)$
$+ C(3+4-1, 3) + C(2+4-1, 2)$
$- C(1+4-1, 1)]$

32. $52!/(13!)^4$ **35.** $C(20, 5)$ **38.** $C(20, 5)^2$

41. $C(15+6-1, 15)$ **44.** $C(10+12-1, 10)$

47. Se aplica el resultado del ejemplo 6.6.9 a al ciclo interior $k-1$ anidado de ese ejemplo. Luego se escribe el número de iteraciones para $i_1 = 1$; después $i_1 = 2$; etcétera. Por ejemplo 6.6.9, esta suma es igual a $C(k+n-1, k)$.

50. Existen $C(10+3-1, 3-1)$ maneras de distribuir 10 discos compactos entre María, Iván y Juan. Si cada uno recibe al menos dos discos compactos, deben distribuirse los seis restantes, y hay $C(6+3-1, 3-1)$ maneras de hacerlo. Entonces la probabilidad de que cada persona reciba al menos dos discos es

$$\frac{C(6+3-1, 3-1)}{C(10+3-1, 3-1)}.$$

Sección 6.7 Repaso

1. Si *a* y *b* son números reales y *n* es un entero positivo, entonces

$$(a+b)^n = \sum_{k=0}^{n} C(n, k)a^{n-k}b^k.$$

2. En la expansión de

$$(a+b)^n = \underbrace{(a+b)(a+b)\cdots(a+b)}_{n \text{ factores}},$$

el término $a^{n-k}b^k$ surge al elegir *b* entre *k* factores y *a* entre los otros $n-k$ factores, lo que puede hacerse de $C(n, k)$ maneras. Sumando sobre toda *k* se obtiene el teorema binomial.

3. El triángulo de Pascal es un arreglo en forma triangular de los coeficientes binomiales. La orilla consiste en unos y cualquier valor interior es la suma de los dos números arriba de él:

$$\begin{array}{ccccccccccc}
 & & & & & 1 & & & & & \\
 & & & & 1 & & 1 & & & & \\
 & & & 1 & & 2 & & 1 & & & \\
 & & 1 & & 3 & & 3 & & 1 & & \\
 & 1 & & 4 & & 6 & & 4 & & 1 & \\
1 & & 5 & & 10 & & 10 & & 5 & & 1 \\
 & & & & & \vdots & & & & &
\end{array}$$

4. $C(n, 0) = C(n, n) = 1$, para toda $n \geq 0$; y $C(n+1, k) =$, $C(n, k-1) + C(n, k)$, para todo $1 \leq k \leq n$.

Sección 6.7

1. $x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^4$

3. $C(11, 7)x^4y^7$ **6.** 5,987,520

9. $C(7, 3) + C(5, 2)$, ya que

$$\begin{aligned}(a+\sqrt{ax}+x)^2(a+x)^5 &= [(a+x)+\sqrt{ax}]^2(a+x)^5 \\ &= (a+x)^7 + 2\sqrt{ax}(a+x)^6 + ax(a+x)^5.\end{aligned}$$

10. $C(10+3-1, 10)$ **13.** 1 8 28 56 70 56 28 8 1

16. [Sólo el paso inductivo] Suponga que el teorema es cierto para *n*.

$$\begin{aligned}(a+b)^{n+1} &= (a+b)(a+b)^n \\ &= (a+b)\sum_{k=0}^{n} C(n, k)a^{n-k}b^k \\ &= \sum_{k=0}^{n} C(n, k)a^{n+1-k}b^k \\ &\quad + \sum_{k=0}^{n} C(n, k)a^{n-k}b^{k+1} \\ &= \sum_{k=0}^{n} C(n, k)a^{n+1-k}b^k \\ &\quad + \sum_{k=1}^{n+1} C(n, k-1)a^{n+1-k}b^k \\ &= C(n, 0)a^{n+1}b^0 + \sum_{k=1}^{n} C(n, k)a^{n+1-k}b^k \\ &\quad + C(n, n)a^0b^{n+1} \\ &\quad + \sum_{k=1}^{n} C(n, k-1)a^{n+1-k}b^k \\ &= C(n+1, 0)a^{n+1}b^0 \\ &\quad + \sum_{k=1}^{n} [C(n, k) + C(n, k-1)]a^{n+1-k}b^k \\ &\quad + C(n+1, n+1)a^0b^{n+1} \\ &= C(n+1, 0)a^{n+1}b^0 \\ &\quad + \sum_{k=1}^{n} C(n+1, k)a^{n+1-k}b^k \\ &\quad + C(n+1, n+1)a^0b^{n+1} \\ &= \sum_{k=0}^{n+1} C(n+1, k)a^{n+1-k}b^k\end{aligned}$$

19. El número de soluciones en enteros no negativos de

$$x_1 + x_2 + \cdots + x_{k+2} = n - k$$

es $C(k+2+n-k-1, n-k) = C(n+1, k+1)$. El número de soluciones también es el número de soluciones de $C(k+1+n-k-1,$ con $x_{k+2} = 0$ más el número de soluciones de

$$C(k+1+n-k-1-1, n-k-1) = C(n-1, k)$$

con $x_{k+2} = 1$ más $\cdots$ más el número de soluciones de $C(k+1+0-1, 0) = C(k, k)$ con $x_{k+2} = n-k$. El resultado ahora es directo.

22. Se toma $a = 1$ y $b = 2$ en el teorema binomial.

25. $x^3 + 3x^2y + 3x^2z + 3xy^2 + 6xyz + 3xz^2 + y^3 + 3y^2z + 3yz^2 + z^3$

28. Se hace $a = 1$ y $b = x$ y se sustituye n por $n-1$ en el teorema del binomio para obtener

$$(1+x)^{n-1} = \sum_{k=0}^{n-1} C(n-1, k)x^k.$$

Ahora se multiplica por n para obtener

$$\begin{aligned} n(1+x)^{n-1} &= n\sum_{k=0}^{n-1} C(n-1,k)x^k \\ &= n\sum_{k=1}^{n} C(n-1,k-1)x^{k-1} \\ &= \sum_{k=1}^{n} \frac{n(n-1)!}{(n-k)!\,(k-1)!}x^{k-1} \\ &= \sum_{k=1}^{n} \frac{n!}{(n-k)!\,k!}kx^{k-1} \\ &= \sum_{k=1}^{n} C(n,k)kx^{k-1}. \end{aligned}$$

31. La solución es por inducción sobre k. Se omite el paso base. Suponga que la afirmación es cierta para k. Después de k iteraciones, se obtiene la sucesión definida por

$$a'_j = \sum_{i=0}^{k-1} a_{i+j}\frac{B_i}{2^n}.$$

Sea $B'_0, \ldots, B'_k$ el renglón después de $B_0, \ldots, B_{k-1}$ en el triángulo de Pascal. Suavizando a' con c para obtener a'' se tiene

$$\begin{aligned} a''j &= \frac{1}{2}(a'_j + a'_{j+1}) \\ &= \frac{1}{2^{n+1}}\left(\sum_{i=0}^{k-1} a_{i+j}B_i + \sum_{i=0}^{k-2} a_{i+j+1}B_i\right) \\ &= \frac{1}{2^{n+1}}\left(a_jB_0 + \sum_{i=1}^{k-1} a_{i+j}B_i + \sum_{i=0}^{k-2} a_{i+j+1}B_i + a_{k+j}B_{k-1}\right) \\ &= \frac{1}{2^{n+1}}\left(a_jB_0 + \sum_{i=1}^{k-1} a_{i+j}B_i + \sum_{i=1}^{k-1} a_{i+j}B_{i-1} + a_{k+j}B_{k-1}\right) \\ &= \frac{1}{2^{n+1}}\left(a_jB'_0 + \sum_{i=1}^{k-1} a_{i+j}B'_i + a_{k+j}B'_k\right) \\ &= \frac{1}{2^{n+1}}\sum_{i=0}^{k} a_{i+j}B'_i, \end{aligned}$$

y el paso inductivo queda completo.

Sección 6.8 Repaso

1. *Primera forma:* Si n palomas vuelan a k hoyos de un paloma y $k < n$, algún hoyo contiene al menos dos palomas.

Segunda forma: Si f es una función de un conjunto finito X a un conjunto finito Y y $|X| > |Y|$, entonces $f(x_1) = f(x_2)$ para algunas $x_1, x_2 \in X$, $x_1 \neq x_2$.

Tercera forma: Sea f una función de un conjunto finito X a un conjunto finito Y. Suponga que $|X| = n$ y $|Y| = m$. Sea $k = \lceil n/m \rceil$. Entonces hay al menos k valores $a_1, \ldots, a_k \in X$ tales que

$$f(a_1) = f(a_2) = \cdots = f(a_k).$$

2. *Primera forma:* Si 20 personas (palomas) entran en seis habitaciones (hoyos de paloma), entonces alguna habitación contiene al menos dos personas.

Segunda forma: En el ejemplo anterior, sea X el conjunto de personas y Y el conjunto de habitaciones. Si p es una persona, defina la función f de manera que $f(p)$ sea la habitación donde está p. Entonces para algunas personas p_1 y p_2 diferentes, $f(p_1) = f(p_2)$; es decir, las personas distintas p_1 y p_2 están en la misma habitación.

Tercera forma: Sean X, Y y f como en el último ejemplo. Entonces existen al menos $\lceil 20/6 \rceil = 4$ personas p_1, p_2, p_3, p_4 tales que

$$f(p_1) = f(p_2) = f(p_3) = f(p_4);$$

esto es, existen al menos cuatro personas en la misma habitación.

Sección 6.8

1. Hay 12 nombres posibles para 13 personas. Se puede considerar que la asignación de nombres a personas es lo mismo que la asignación de hoyos de paloma a las palomas. Por el principio del palomar, algún nombre se asigna por lo menos a dos personas.

4. Sí. Se conectan los procesadores 1 y 2, 2 y 3, 2 y 4, 3 y 4. El procesador 5 no se conecta. Ahora, sólo los procesadores 3 y 4 están conectados directamente al mismo número de procesadores.

7. Sea a_1 la posición del i-ésimo artículo no disponible. Considere

$$a_1, \ldots, a_{30}; \quad a_1+3, \ldots, a_{30}+3; \quad a_1+6, \ldots, a_{30}+6.$$

Estos 90 números tienen valores entre 1 y 86. Por la segunda forma del principio de palomar, dos de estos números son los mismos. Si $a_i = a_j + 3$, dos están separados por tres artículos. Si $a_i = a_j + 6$, dos están separados por seis. Si $a_i + 3 = a_j + 6$, dos están separados por tres.

11. $n+1$

12. Suponga que $k \leq m/2$. Es claro que $k \geq 1$. Como $m \leq 2n+1$,

$$k \leq \frac{m}{2} \leq n + \frac{1}{2} < n+1.$$

Suponga que $k > m/2$. Entonces

$$m - k < m - \frac{m}{2} = \frac{m}{2} < n+1.$$

Como m es el elemento mayor en X, $k < m$. Entonces $k+1 \leq m$ de manera que $1 \leq m-k$. Por lo tanto, el intervalo de a está contenido en $\{1, \ldots, n\}$.

13. Se aplica la segunda forma del principio del palomar.

14. Suponga que $a_i = a_j$. Entonces $i \leq m/2$ y $j > m/2$ o bien $j \leq m/2$ e $i > m/2$. Se puede suponer que $i \leq m/2$ y $j > m/2$. Ahora

$$i + j = a_i + m - a_j = m.$$

24. Cuando se divide a entre b, los residuos posibles son 0, 1, . . . , $b-1$. Considere los que ocurre después de b divisiones.

28. Se supone que el tablero tiene tres filas y siete columnas. Dos cuadrados en una columna que son del mismo color se llaman un *par colorido*. Por el principio del palomar, cada columna contiene al menos un par colorido. Entonces el tablero contiene siete pares coloridos, uno en cada columna. De nuevo, por el principio del palomar, al menos cuatro de estos siete pares coloridos son del mismo color, digamos rojo. Como hay tres pares de renglones y cuatro pares coloridos rojos, una tercera aplicación del principio del palomar muestra que al menos dos columnas contienen pares coloridos rojos en los mismos renglones. Estos pares coloridos rojos determinan un rectángulo cuyas cuatro esquinas son rojas.

31. Suponga que es posible marcar k cuadros en la submalla de $k \times k$ en la esquina superior izquierda y k cuadros en la submalla de $k \times k$ en la esquina inferior derecha de manera que no haya dos cuadros marcados en la misma fila, columna o diagonal de la malla de $2k \times 2k$. Entonces los $2k$ cuadros marcados están contenidos en $2k - 1$ diagonales. Una diagonal comienza en el cuadro de la esquina superior izquierda y corre hasta la esquina inferior izquierda; $k - 1$ diagonales comienzan en los $k - 1$ cuadros a la derecha del cuadro de la esquina superior izquierda y corren paralelas a la primera diagonal descrita; y $k - 1$ diagonales comienzan en los k -1 cuadros debajo del cuadro superior izquierdo y corren paralelas a las otras descritas. Por la primera forma del principio del palomar, alguna diagonal contiene dos cuadros marcados. Esta contradicción demuestra que es imposible marcar k cuadros en la submalla de $k \times k$ en la esquina superior izquierda y k cuadros en la submalla de $k \times k$ en la esquina inferior derecha de manera que no haya dos cuadros marcados en el misma fila, columna o diagonal de la submalla de $2k \times 2k$.

Capítulo 6 Autoevaluación

1. 2^4

2. $6 \cdot 9 \cdot 7 + 6 \cdot 9 \cdot 4 + 6 \cdot 7 \cdot 4 + 9 \cdot 7 \cdot 4$

3. $2^n - 2$ **4.** $6 \cdot 5 \cdot 4 \cdot 3 + 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2$

5. $6!/(3!\,3!) = 20$

6. Se construyen las cadenas mediante un proceso de tres pasos. Primero se eligen posiciones para A, C y E [$C(6, 3)$ maneras]. Después se colocan A, C y E en estas posiciones. Se puede colocar C de una manera (al último) y se pueden colocar A y E de dos maneras (AE o EA). Por último, se colocan las tres letras restantes (3! maneras). Por lo tanto, el número total de cadenas es $C(6, 3) \cdot 2 \cdot 3!$.

7. Dos palos se pueden elegir de $C(4, 2)$ maneras. Se pueden seleccionar 3 cartas de un palo de $C(13, 3)$ maneras, y 3 cartas de otro palo de $C(13, 3)$ maneras. Por lo tanto, el número total de manos es $C(4, 2)C(13, 3)^2$.

8. Deben seleccionarse ya sea tres o cuatro discos defectuosos. Entonces el número total de selecciones es $C(5, 3)C(95, 1) + C(5, 4)$.

9. 12567 **10.** 234567

11. 6427153 **12.** 631245

13. 1/4 **14.** 5/36

15. $\dfrac{C(7, 5)C(31 - 7, 2)}{C(31, 7)} = \dfrac{21 \cdot 276}{2629575} = 0.002204158$

16. $\dfrac{4 \cdot C(13, 6) \cdot 3 \cdot C(13, 5) \cdot 2 \cdot C(13, 2)}{C(52, 13)}$

17. $P(H) = 5/6$, $P(T) = 1/6$

18. Sea S el evento "niños de uno y otro sexo" y sea G el evento "cuando mucho una niña". Entonces

$$P(S) = \frac{6}{8} = \frac{3}{4}$$
$$P(G) = \frac{4}{8} = \frac{1}{2}$$
$$P(S \cap G) = \frac{3}{8}.$$

Por lo tanto,

$$P(S)P(G) = \frac{3}{4} \cdot \frac{1}{2} = \frac{3}{8} = P(S \cap G),$$

y S y G son independientes.

19. Sea J el evento "Jorge pasa" y sea A el evento "Alicia pasa". Entonces

$$\begin{aligned}
P(\text{Jorge no pase}) &= P(\overline{J}) = 1 - P(J) = 0.25\\
P(\text{ambos pasen}) &= P(J \cap A) = P(J)P(A)\\
&= (0.75)(0.80) = 0.6\\
P(\text{ambos reprueben}) &= P(\overline{J} \cap \overline{A}) = P(\overline{J \cup A})\\
&= 1 - P(J \cup A)\\
&= 1 - [P(J) + P(A) - P(J \cap A)]\\
&= 1 - [0.75 + 0.80 - 0.6] = 0.05\\
P(\text{al menos uno pase}) &= 1 - P(\text{ambos reprueben})\\
&= 1 - 0.05 = 0.95.
\end{aligned}$$

20. Sea B el evento "falla presente" y sean T, R y J los eventos "Tulio (respectivamente, Roberto, José) escribió el programa". Entonces

$$\begin{aligned}
P(J \mid B) &= \frac{P(B \mid J)P(J)}{P(B \mid J)P(J) + P(B \mid T)P(T) + P(B \mid R)P(R)}\\
&= \frac{(0.05)(0.25)}{(0.05)(0.25) + (0.03)(0.30) + (0.02)(0.45)}\\
&= 0.409836065.
\end{aligned}$$

21. $8!/(3!2!)$

22. Se cuenta el número de cadenas en las que no aparece I antes de L y luego se restan del número total de cadenas.

Se construyen cadenas en las que no aparece I antes de L mediante un proceso de dos pasos. Primero, se eligen posiciones para N, O y S; después se colocan las I y L. Se pueden elegir posiciones para N, O y S de $8 \cdot 7 \cdot 6$ maneras. Las I y L se pueden colocar sólo de una forma porque las L deben aparecer primero. Entonces hay $8 \cdot 7 \cdot 6$ cadenas en las que no aparece I antes de L.

El ejercicio 21 demostró que hay $8!/(3!2!)$ cadenas formadas ordenando las letras *ILLINOIS*. Por lo tanto, hay

$$\frac{8!}{3!\,2!} - 8 \cdot 7 \cdot 6$$

cadenas formadas ordenando las letras *ILLINOIS* en las que alguna I aparece antes que alguna L.

23. $12!/(3!)^4$ **24.** $C(11 + 4 - 1, 4 - 1)$

25. $$\begin{aligned}(s - r)^4 &= C(4, 0)s^4 + C(4, 1)s^3(-r) + C(4, 2)s^2(-r)^2\\ &\quad + C(4, 3)s(-r)^3 + C(4, 4)(-r)^4\\ &= s^4 - 4s^3r + 6s^2r^2 - 4sr^3 + r^4\end{aligned}$$

26. $2^3 \cdot 8!/(3!1!4!)$

27. Si se hace $a = 2$ y $b = -1$ en el teorema del binomio, se obtiene

$$1 = 1^n = [2 + (-1)]^n = \sum_{k=0}^{n} C(n, k)2^{n-k}(-1)^k.$$

28. $C(n, 1) = n$

29. Sean los 15 calcetines individuales, las palomas y los 14 tipos de pares, los hoyos en el palomar. Se asigna a cada calcetín (paloma) su tipo (hoyo de paloma). Por el principio del palomar, algún hoyo contendrá al menos dos palomas (los pares de calcetines).

30. Existen $3 \cdot 2 \cdot 3 = 18$ nombres posibles para las 19 personas. Se puede considerar que las asignaciones de nombres a personas son

lo mismo que asignar palomas a los hoyos de paloma. Por el principio del palomar, algún nombre se asigna al menos a dos personas.

31. Sea a_i la posición del i-ésimo artículo disponible. Los 220 números

$$a_1, \ldots, a_{110}; \qquad a_1 + 19, \ldots, a_{110} + 19$$

están entre 1 y 219. Por el principio del palomar, dos son iguales.

32. Cada punto tiene una coordenada x que es par o impar, y una coordenada y que es par o impar. Como hay cuatro posibilidades y cinco puntos, por el principio del palomar, al menos dos puntos, $p_i = (x_i, y_i)$ y $p_j = (x_j, y_j)$ cumplen

- Ambos, x_i y x_j, son pares o ambos son impares

y

- Ambos, y_i y y_j, son pares o ambos son impares.

Por lo tanto, $x_i + x_j$ es par y $y_i + y_j$ es par. En particular, $(x_i + x_j)/2$ y $(y_i + y_j)/2$ son enteros. Entonces el punto medio del par p_i y p_j tiene coordenadas enteras.

Sección 7.1 Repaso

1. Una relación de recurrencia define el n-ésimo término de una sucesión en términos de ciertos predecesores.

2. Una condición inicial para una sucesión es un valor explícito dado para un término en particular de la sucesión.

3. El interés compuesto es interés sobre el interés. Si una persona invierte d dólares a un porcentaje p compuesto anualmente y A_n es la cantidad de dinero ganado después de n años, la relación de recurrencia

$$A_n = \left(1 + \frac{p}{100}\right) A_{n-1}$$

junto con la condición inicial $A_0 = d$ define la sucesión $\{A_n\}$.

4. El juego de las torres de Hanoi consiste en tres estacas montadas en una tabla y discos de varios tamaños con agujeros en sus centros. Sólo un disco de diámetro menor se puede colocar sobre un disco de diámetro mayor. Dados todos los discos apilados en una estaca, el problema es transferirlos a otra estaca moviendo un disco a la vez.

5. Si hay un disco, éste se mueve y el juego termina. Si hay $n > 1$ discos, mueva de manera recursiva $n - 1$ discos a una estaca vacía. Mueva el disco más grande a la estaca que queda vacía. De manera recursiva, mueva $n - 1$ discos encima del disco más grande.

6. Se supone que en el tiempo n, la cantidad q_n vendida al precio p_n está dada por la ecuación $p_n = a - bq_n$, donde a y b son parámetros positivos. También se supone que $p_n = kq_{n+1}$, donde k es otro parámetro positivo. Si se grafica el precio y la cantidad contra el tiempo, la gráfica se parece a una telaraña (vea la figura 7.1.5).

7. La función de Ackermann $A(m, n)$ se define por las relaciones de recurrencia

$$A(m, 0) = A(m - 1, 1), \qquad m \geq 1$$
$$A(m, n) = A(m - 1, A(m, n - 1)), \qquad m \geq 1, n \geq 1$$

y las condiciones iniciales

$$A(0, n) = n + 1, \qquad n \geq 0.$$

Sección 7.1

1. $a_n = a_{n-1} + 4; a_1 = 3$

4. $A_n = (1.14)A_{n-1}$ **5.** $A_0 = 2000$

6. $A_1 = 2280$, $A_2 = 2599.20$, $A_3 = 2963.088$

7. $A_n = (1.14)^n 2000$

8. Debe tenerse $A_n = 4000$ o $(1.14)^n 2000 = 4000$ o $(1.14)^n = 2$. Tomando logaritmos en ambos lados, se tiene $n \log 1.14 = \log 2$. Entonces

$$n = \frac{\log 2}{\log 1.14} = 5.29.$$

18. Se cuenta el número de cadenas de n bits que no contienen el patrón 000.

- Comienza con 1. En este caso, si la cadena de $(n - 1)$ bits restante no contiene 000, tampoco lo tendrá la cadena de n bits. Hay S_{n-1} de estas cadenas de $(n - 1)$.
- Comienza con 0. Deben considerarse dos casos.
  1. Comienza con 01. En este caso, si la cadena de $(n - 2)$ bits restante no contiene 000, tampoco lo tendrá la cadena de n bits. Hay S_{n-2} de estas cadenas de $(n - 2)$ bits.
  2. Comienza con 00. Entonces el tercer bit debe ser 1 y si la cadena de $(n - 3)$ bits restante no contiene 000, tampoco lo tendrá la cadena de n bits. Existen S_{n-3} cadenas de $(n - 3)$ bits de este tipo.

Como los casos son mutuamente excluyentes y cubren todas las cadenas de n bits $(n > 3)$ que no contienen 000, se tiene $S_n = S_{n-1} + S_{n-2} + S_{n-3}$ para $n > 3$. $S_1 = 2$ (hay dos cadenas de 1 bit), $S_2 = 4$ (hay cuatro cadenas de 2 bits) y $S_3 = 7$ (hay ocho cadenas de 3 bits pero una de ellas es 000).

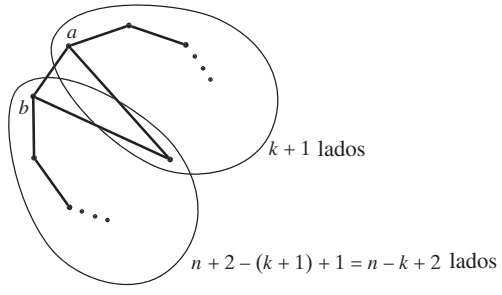
19. Existen S_{n-1} cadenas de n bits que comienzan con 1 y no contienen el patrón 00, y se tienen S_{n-2} cadenas de n bits que comienzan con 0 (ya que el segundo bit debe ser 1) y no contienen el patrón 00. Entonces $S_n = S_{n-1} + S_{n-2}$. Las condiciones iniciales son $S_1 = 2, S_2 = 3$.

22. $S_1 = 2, S_2 = 4, S_3 = 7, S_4 = 12$

25. $C_3 = 5, C_4 = 14, C_5 = 42$

29. Sea P_n el número de maneras de dividir un polígono convexo de $(n + 2)$ lados, $n \geq 1$, en triángulos dibujando $n - 1$ líneas que pasan por los vértices que no se cruzan en el interior del polígono. Se observa que $P_1 = 1$.

Suponga que $n > 1$ y considere un polígono convexo de $(n + 2)$ lados (vea la siguiente figura).



Se elige una arista ab y se construye una partición del polígono con un procedimiento de dos pasos. Primero se selecciona un triángulo al que pertenezca el lado ab. Este triángulo divide al polígono original en dos polígonos: uno con $k + 1$ lados, para alguna k que satisface $1 \leq k \leq n$; el otro con $n - k + 2$ lados (vea la figura anterior). Por definición, se puede hacer una partición del polígono de $(k + 1)$ lados de P_{k-1} maneras y se puede hacer una partición del polígono de $(n - k + 2)$ lados de P_{n-k} maneras. (Para los casos degenerados $k = 1$ y $k = n$, se hace $P_0 = 1$). Por lo tanto, el número total de maneras de hacer particiones del polígono

de $(n + 2)$ lados es

$$P_n = \sum_{k=1}^{n} P_{k-1} P_{n-k}.$$

Como la sucesión $P_1, P_2, \ldots$ satisface la misma relación de recurrencia que la sucesión de Catalan $C_1, C_2, \ldots$ y $P_0 = P_1 = 1 = C_0 = C_1$, se deduce que $P_n = C_n$ para toda $n \geq 1$.

33. [Para $n = 3$]

Paso 1: se mueve el disco 3 de la estaca 1 a la estaca 3.
Paso 2: se mueve el disco 2 de la estaca 1 a la estaca 2.
Paso 3: se mueve el disco 3 de la estaca 3 a la estaca 2.
Paso 4: se mueve el disco 1 de la estaca 1 a la estaca 3.
Paso 5: se mueve el disco 3 de la estaca 2 a la estaca 1.
Paso 6: se mueve el disco 2 de la estaca 2 a la estaca 3.
Paso 7: se mueve el disco 3 de la estaca 1 a la estaca 3.

35. Sean α y β los ángulos mostrados en la figura 7.1.6. La geometría de la situación muestra que el precio tiende a estabilizarse si y sólo si $\alpha + \beta > 180°$. Esta condición se cumple si y sólo si $-\tan \beta < \tan \alpha$. Como $b = -\tan \beta$ y $k = \tan \alpha$, se concluye que el precio se estabiliza si y sólo si $b < k$.

37. $A(2, 2) = 7$, $A(2, 3) = 9$ **40.** $A(3, n) = 2^{n+3} - 3$

43. Si $m = 0$

$$\begin{aligned} A(m, n + 1) &= A(0, n + 1) \\ &= n + 2 > n + 1 \\ &= A(0, n) = A(m, n). \end{aligned}$$

La última desigualdad se obtiene del ejercicio 41.

44. Use los ejercicios 38 y 39.

47. Se prueba la afirmación usando inducción sobre x. A su vez el paso inductivo requerirá inducción sobre y.

El ejercicio 44 muestra que la ecuación es cierta para $x = 0$, 1, 2 y para toda y.

Paso base $(x = 2)$ Vea el ejercicio 44.

Paso inductivo (caso x implica caso $x + 1$) Suponga que $x \geq 2$ y

$$A(x, y) = AO(x, 2, y + 3) - 3 \text{ para toda } y \geq 0.$$

Se debe probar que

$$A(x + 1, y) = AO(x + 1, 2, y + 3) - 3 \text{ para toda } y \geq 0.$$

Se establece esta última ecuación por inducción sobre y.

Paso base $(y = 0)$ Se debe probar que

$$A(x + 1, 0) = AO(x + 1, 2, 3) - 3.$$

Ahora

$$\begin{aligned} &AO(x + 1, 2, 3) - 3 \\ &\quad = AO(x, 2, AO(x + 1, 2, 2)) - 3 && \text{por definición} \\ &\quad = AO(x, 2, 4) - 3 && \text{por el ejercicio 46} \\ &\quad = A(x, 1) && \text{por la suposición de inducción sobre } x \\ &\quad = A(x + 1, 0) && \text{por (7.1.11).} \end{aligned}$$

Paso inductivo (caso y implica caso $y + 1$) Suponga que

$$A(x + 1, y) = AO(x + 1, 2, y + 3) - 3.$$

Debemos probar que

$$A(x + 1, y + 1) = AO(x + 1, 2, y + 4) - 3.$$

Ahora

$$\begin{aligned} &AO(x + 1, 2, y + 4) - 3 \\ &\quad = AO(x, 2, AO(x + 1, 2, y + 3)) - 3 && \text{por definición} \\ &\quad = AO(x, 2, A(x + 1, y) + 3) - 3 && \text{por la suposición inductiva sobre } y \\ &\quad = A(x, A(x + 1, y)) && \text{por la suposición inductiva sobre } x \\ &\quad = A(x + 1, y + 1) && \text{por (7.1.12).} \end{aligned}$$

50. Suponga que se tienen n dólares. Si se compra jugo de naranja el primer día, quedan $n - 1$ dólares, que se pueden gastar de R_{n-1} maneras. De forma similar, si el primer día se compra leche o cerveza, habrá R_{n-2} maneras de gastar los dólares restantes. Como estos casos son ajenos, $R_n = R_{n-1} + 2R_{n-2}$.

53. $S_3 = 1/2$, $S_4 = 3/4$

55. Una función f de $X = \{1, \ldots, n\}$ en X se denotará por $(i_1, i_2, \ldots, i_n)$, que significa que $f(k) = i_k$. El problemas es entonces contar el número de maneras de seleccionar $i_1, \ldots, i_n$ de modo que si i ocurre, también lo hacen $1, 2, \ldots, i - 1$.

Se contará el número de estas funciones que tienen exactamente j unos. Tales funciones se construyen en dos pasos: elegir las posiciones para los j unos y después colocar los otros números. Existen $C(n, j)$ maneras de colocar los unos. El resto de los números deben seleccionarse de modo que si i aparece, también aparezcan $1, \ldots, i - 1$. Hay F_{n-j} maneras de seleccionar los números restantes, ya que se deben seleccionar entre $\{2, \ldots, n\}$. Entonces hay $C(n,j)F_{n-j}$ funciones del tipo deseado que tienen exactamente j unos. Por lo tanto, el número total de funciones de X en X que tienen la propiedad de que si i está en el rango de f, entonces también lo están $1, \ldots, i - 1$, es

$$\begin{aligned} \sum_{j=1}^{n} C(n, j) F_{n-j} &= \sum_{j=1}^{n} C(n, n - j) F_{n-j} \\ &= \sum_{j=0}^{n-1} C(n, j) F_j. \end{aligned}$$

58. $\{u_n\}$ no es una relación de recurrencia porque, si n es impar y mayor que 1, u_n se define en términos del *sucesor* u_{3n+1}.u_i, para $2 \leq i \leq 7$, es igual a 1. Como ejemplos,

$$\begin{aligned} u_2 &= u_1 = 1 \\ u_3 &= u_{10} = u_5 = u_{16} = u_8 = u_4 = u_2 = 1. \end{aligned}$$

61. Utilice la ecuación (7.7.4) para escribir

$$S(k, n) = \sum_{i=1}^{n} S(k - 1, i).$$

64. Se usa la terminología del ejercicio 81, sección 6.2. Se elige una persona de $n + 1$, digamos P. Hay $s_{n,j-1}$ maneras de que P se siente solo. (Las otras n personas se sientan alrededor de las $k - 1$ mesas restantes). Después se cuenta el número de arreglos en los que P no está solo. Se sienta a todos menos P alrededor de las k mesas. Esto se puede hacer de $s_{n,k}$ maneras. Ahora P se puede sentar a la derecha de alguien de n maneras. Entonces existen $ns_{n,k}$ arreglos en los que P no está solo. La relación de recurrencia se obtiene de esto.

67. Sea A_n la cantidad al final de n años y sea i la tasa de interés expresada como decimal. El análisis que sigue al ejemplo 7.1.3 muestra que

$$A_n = (1 + i)^n A_0.$$

El valor de n requerido para duplicar la cantidad satisface

$$2A_0 = (1+i)^n A_0 \text{ o } 2 = (1+i)^n.$$

Si se toman logaritmos naturales (logaritmo base e) en ambos lados de la ecuación, se obtiene

$$\ln 2 = n \ln(1+i).$$

Entonces

$$n = \frac{\ln 2}{\ln(1+i)}.$$

Como $\ln 2 = 0.6931472...$ y $\ln(1+i)$ es aproximadamente igual a i para valores pequeños de i, n es aproximadamente igual a $0.69.../i$, que a su vez es aproximadamente igual a $70/r$.

69. 1, 3, 2; 2, 3, 1; $E_3 = 2$

72. Se cuenta el número de permutaciones sube/baja de $1, \ldots, n$ considerando cuántas tienen a n en la segunda, cuarta, . . . , posiciones.

Suponga que n está en la segunda posición. Como cualquiera de los números restantes es menor que n, cualquiera de ellos se puede colocar en la primera posición. Entonces se puede elegir el número que se coloca en la primera posición de $C(n-1, 1)$ maneras y, después de elegirlo, se puede arreglar de $E_1 = 1$ manera. Las últimas $n-2$ posiciones se pueden llenar de E_{n-2} maneras ya que cualquier permutación sube/baja de los $n-2$ números restantes da una permutación sube/baja de $1, \ldots, n$. Entonces el número de permutaciones sube/baja de $1, \ldots, n$ con n en la segunda posición es $C(n-1, 1)E_1E_{n-2}$.

Suponga que n está en la cuarta posición. Se pueden seleccionar números para colocar en las tres primeras posiciones de $C(n-1, 3)$ maneras. Después de seleccionar los tres elementos, se puede arreglarlos de E_3 maneras. Los últimos $n-4$ números se pueden arreglar de E_{n-4} maneras. Entonces el número de permutaciones sube/baja de $1, \ldots, n$ con n en la cuarta posición es $C(n-1, 3)E_3E_{n-4}$.

En general, el número de permutaciones sube/baja de $1, \ldots, n$ con n en la posición $2j$ es

$$C(n-1, 2j-1)E_{2j-1}E_{n-2j}.$$

Sumando sobre j se obtiene la relación de recurrencia deseada.

Sección 7.2 Repaso

1. Use la relación de recurrencia para escribir el n-ésimo término en términos de ciertos de sus predecesores. Después utilice sucesivamente la relación de recurrencia para sustituir cada uno de los términos resultantes por algunos de sus predecesores. Continúe hasta obtener una fórmula explícita.

2. Una relación de recurrencia homogénea lineal de orden n con coeficientes constantes es una relación de recurrencia de la forma

$$a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ka_{n-k}.$$

3. $a_n = 6a_{n-1} - 8a_{n-2}$

4. Para resolver

$$a_n = c_1a_{n-1} + c_2a_{n-2},$$

primero se despeja t de la ecuación

$$t^2 = c_1t + c_2$$

para t. Suponga que las raíces son t_1 y t_2 y que $t_1 \neq t_2$. Entonces la solución general es de la forma

$$a_n = bt_1^n + dt_2^n,$$

donde b y d son constantes. Los valores de las constantes se obtienen definiendo las condiciones iniciales.

Si $t_1 = t_2 = t$, la solución general es de la forma

$$a_n = bt^n + dnt^n,$$

donde otra vez b y d son constantes. De nuevo, los valores de las constantes se obtienen de las condiciones iniciales.

Sección 7.2

1. Sí; orden 1 **4.** No

7. No **10.** Sí; orden 3

11. $a_n = 2(-3)^n$ **15.** $a_n = 2^{n+1} - 4^n$

18. $a_n = (2^{2-n} + 3^n)/5$ **21.** $a_n = 2(-4)^n + 3n(-4)^n$

24. $R_n = [(-1)^n + 2^{n+1}]/3$

28. Sea d_n la población de venados en el tiempo n. La condición inicial es $d_0 = 0$. La relación de recurrencia es

$$d_n = 100n + 1.2d_{n-1}, \quad n > 0.$$

$$\begin{aligned}
d_n &= 100n + 1.2d_{n-1} = 100n + 1.2[100(n-1) + 1.2d_{n-2}] \\
&= 100n + 1.2 \cdot 100(n-1) + 1.2^2 d_{n-2} \\
&= 100n + 1.2 \cdot 100(n-1) \\
&\quad + 1.2^2[100(n-2) + 1.2d_{n-3}] \\
&= 100n + 1.2 \cdot 100(n-1) \\
&\quad + 1.2^2 \cdot 100(n-2) + 1.2^3 d_{n-3} \\
&\vdots \\
&= \sum_{i=0}^{n-1} 1.2^i \cdot 100(n-i) + 1.2^n d_0 \\
&= \sum_{i=0}^{n-1} 1.2^i \cdot 100(n-i) \\
&= 100n \sum_{i=0}^{n-1} 1.2^i - 1.2 \cdot 100 \sum_{i=1}^{n-1} i \cdot 1.2^{i-1} \\
&= \frac{100n(1.2^n - 1)}{1.2 - 1} \\
&\quad - 120\frac{(n-1)1.2^n - n1.2^{n-1} + 1}{(1.2-1)^2}, \qquad n > 0.
\end{aligned}$$

29. De $p_{n-1} = \frac{1}{2}p_n + \frac{1}{2}p_{n-2}$, se obtiene $p_n = 2p_{n-1} - p_{n-2}$.

32. $p_n = n/(S+T)$

36. Se hace $b_n = a_n/n!$ para obtener $b_n = -2b_{n-1} + 3b_{n-2}$. Al resolver se obtiene $a_n = n!\,b_n = (n!/4)[5 - (-3)^n]$.

39. Se establece la desigualdad usando inducción sobre n.

Los casos base $n = 1$ y $n = 2$ se dejan al lector. Ahora suponga que la desigualdad es verdadera para valores menores que $n+1$. Entonces

$$\begin{aligned}
f_{n+2} &= f_{n+1} + f_n \\
&\geq \left(\frac{1+\sqrt{5}}{2}\right)^{n-1} + \left(\frac{1+\sqrt{5}}{2}\right)^{n-2} \\
&= \left(\frac{1+\sqrt{5}}{2}\right)^{n-2}\left(\frac{1+\sqrt{5}}{2} + 1\right) \\
&= \left(\frac{1+\sqrt{5}}{2}\right)^{n-2}\left(\frac{1+\sqrt{5}}{2}\right)^2 \\
&= \left(\frac{1+\sqrt{5}}{2}\right)^{n},
\end{aligned}$$

y el paso inductivo queda completo.

41. $a_n = b2^n + d4^n + 1$

44. $a_n = b/2^n + d3^n - (4/3)2^n$

47. El argumento es idéntico al del teorema 7.2.11.

50. Invocar de manera recursiva este algoritmo para mover los $n - k_n$ discos hasta arriba de la estaca 1 a la estaca 2 toma $T(n - k_n)$ movimientos. Mover los k_n discos de la estaca 1 a la 4 requiere $2^{k_n} - 1$ movimientos (vea el ejemplo 7.2.4). Invocar de manera recursiva este algoritmo para mover $n - k_n$ discos en la estaca 2 a la estaca 4 de nuevo toma $T(n - k_n)$ movimientos. La relación de recurrencia se obtiene de aquí.

53. De la desigualdad

$$\frac{k_n(k_n + 1)}{2} \leq n,$$

se puede deducir $k_n \leq \sqrt{2n}$. Como

$$n - k_n \leq \frac{k_n(k_n + 1)}{2},$$

se deduce que $r_n \leq k_n$. Por lo tanto,

$$\begin{aligned} T(n) &= (k_n + r_n - 1)2^{k_n} + 1 \\ &< 2k_n 2^{k_n} + 1 \\ &\leq 2\sqrt{2n}\,2^{\sqrt{2n}} + 1 \\ &= O(4^{\sqrt{n}}). \end{aligned}$$

Sección 7.3 Repaso

1. Sea b_n el tiempo requerido por una entrada de tamaño n. Simule la ejecución del algoritmo y cuente el tiempo requerido por los diferentes pasos. Entonces b_n es igual a la suma de los tiempos requeridos por los pasos.

2. El orden de selección elige el elemento más grande, los coloca al último y después de manera recursiva ordena la sucesión que queda.

3. $\Theta(n^2)$

4. La búsqueda binaria examina el elemento a la mitad de la sucesión. Si el término de en medio es el elemento deseado, la búsqueda binaria termina. De otra manera, la búsqueda binaria compara el elemento de en medio con el elemento deseado. Si éste es menor que el elemento de en medio, busca de manera recursiva en la mitad izquierda de la sucesión. Si el elemento deseado es mayor que el de en medio, busca en forma recursiva en la mitad derecha de la sucesión. La entrada debe ordenarse.

5. Si a_n es el tiempo en el peor caso para una entrada de tamaño n, $a_n = 1 + a_{\lfloor n/2 \rfloor}$.

6. $\Theta(\lg n)$

7. La fusión mantiene dos indicadores en elementos de dos sucesiones de entrada. Al inicio, los indicadores señalan los primeros elementos de las sucesiones. La fusión copia el elemento más pequeño a la salida y mueve el indicador al siguiente elemento de la sucesión que contiene el elemento que se copió. Después repite este proceso. Cuando un indicador queda más allá del final de una de las sucesiones, la fusión concluye copiando el resto de la otra sucesión a la salida. Ambas sucesiones de entrada deben estar ordenadas.

8. $\Theta(n)$, donde n es la suma de las longitudes de las sucesiones de entrada.

9. El merge sort primero divide la entrada en dos partes casi iguales. Después ordena de manera recursiva cada mitad y fusiona las mitades para producir la salida.

10. $a_n = a_{\lfloor n/2 \rfloor} + a_{\lfloor (n+1)/2 \rfloor} + n - 1$

11. Si el tamaño de la entrada es una potencia de dos, el tamaño siempre es divisible entre 2 y las cotas inferiores desaparecen.

12. Un tamaño de entrada arbitrario cae entre dos potencias de dos. Como se conoce el tiempo en el peor caso cuando el tamaño de la entrada es una potencia de dos, se puede acotar el tiempo en el peor caso para una entrada de tamaño arbitrario con las cotas de los tiempos del peor caso para entradas cuyos tamaños son potencias de dos.

13. $\Theta(n \lg n)$

Sección 7.3

1. En la línea 2, como $i > j$ (1 > 5) es falso, se procede a la línea 4, donde se establece que k es igual a 3. En la línea 5, como *clave*('G') no es igual a s_3 ('J'), se procede a la línea 7. En la línea 7, *clave* < s_k ('G' < 'J') es verdadero, de manera que en la línea 8 j se hace igual a 2. Después se invoca este algoritmo con $i = 1, j = 2$ para buscar *clave* en

$$s_1 = \text{'C'}, \qquad s_2 = \text{'G'}.$$

En la línea 2, como $i > j$ (1 > 2) es falso, se procede a la línea 4, donde k se hace igual a 1. En la línea 5, como *clave*('G') no es igual a s_1 ('C') se procede a la línea 7. En la línea 7, *clave* < s_k ('G' < 'C') es falso, por lo que en la línea 10 i se hace igual a 2. Después se invoca este algoritmo con $i = j = 2$ para buscar *clave* en

$$s_2 = \text{'G'}.$$

En la línea 2, como $i > j$ (2 > 2) es falso, se procede a la línea 4, donde k se hace igual a 2. En la línea 5, como *clave*('G') es igual a s_2 ('G'), se regresa 2, el índice de la *clave* en la sucesión s.

4. En la línea 2, como $i > j$ (1 > 5) es falso, se procede a la línea 4, donde k se hace igual a 3. En la línea 5, como la *clave*('Z') no es igual a s_3('J'), se procede a la línea 7. En la línea 7, *clave* < s_k ('Z' < 'J') es falso, de manera que en la línea 10 i se hace igual a 4. Después se invoca este algoritmo con $i = 4, j = 5$ para buscar la *clave* en

$$s_4 = \text{'M'}, \qquad s_5 = \text{'X'}.$$

En la línea 2, como $i > j$ (4 > 5) es falso, se procede a la línea 4, donde k se hace igual a 4. En la línea 5, como la *clave*('Z') no es igual a s_4('M'), se procede a la línea 7. En la línea 7, *clave* < s_k ('Z' < 'M') es falso, de manera que en la línea 10 i se hace igual a 5. Después se invoca este algoritmo con $i = j = 5$ para buscar la *clave* en

$$s_5 = \text{'X'}.$$

En la línea 2, como $i > j$ (5 > 5) es falso, se procede a la línea 4, donde k se hace igual a 5. En la línea 5, como *clave*('Z') no es igual a s_5('X'), se procede a la línea 7. En la línea 7, la *clave* < s_k ('Z' < 'X') es falso, de manera que en la línea 10 i se hace igual a 6. Después se invoca este algoritmo con $i = 6, j = 5$.

En la línea 2, como $i > j$ (6 > 5) es verdadero, se regresa 0 para indicar se falló en encontrar la *clave*.

8. El algoritmo B es superior si $2 \leq n \leq 15$. (Para $n = 1$ y $n = 16$, los algoritmos requieren el mismo número de comparaciones).

11. Suponga que las sucesiones son $a_1, \ldots, a_n$ y $b_1, \ldots, b_n$.

a) $a_1 < b_1 < a_2 < b_2 < \cdots$ *b)* $a_n < b_1$

14. 11

18. El algoritmo 7.3.11 calcula a^n usando la fórmula $a^n = a^m a^{n-m}$.

19. $b_n = b_{\lfloor n/2 \rfloor} + b_{\lfloor (n+1)/2 \rfloor} + 1, b_1 = 0$

20. $b_2 = 1, b_3 = 2, b_4 = 3$ **21.** $b_n = n - 1$

22. Se prueba la fórmula por inducción matemática. El paso base, $n = 1$, ya se estableció.

Suponga que $b_k = k - 1$ para toda $k < n$. Se demuestra que $b_n = n - 1$. Ahora

$$\begin{aligned} b_n &= b_{\lfloor n/2 \rfloor} + b_{\lfloor (n+1)/2 \rfloor} + 1 \\ &= \left\lfloor \frac{n}{2} \right\rfloor - 1 + \left\lfloor \frac{n+1}{2} \right\rfloor - 1 + 1 \end{aligned}$$

por la suposición inductiva

$$= \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{n+1}{2} \right\rfloor - 1 = n - 1.$$

35. Si $n = 1$, entonces $i = j$ y se regresa antes de llegar a la línea 6b, 10 o 14. Por lo tanto, $b_1 = 0$. Si $n = 2$, entonces $j = i + 1$. Hay una comparación en la línea 6b y se regresa antes de llegar a la línea 10 o 14. Por lo tanto, $b_2 = 1$.

36. $b_3 = 3, b_4 = 4$

37. Cuando $n > 2$, se requieren $b_{\lfloor (n+1)/2 \rfloor}$ comparaciones para la primera llamada recursiva y $b_{\lfloor n/2 \rfloor}$ comparaciones para la segunda llamada recursiva. Se necesitan dos comparaciones adicionales en las líneas 10 y 14. La relación de recurrencia se deduce de esto.

38. Suponga que $n = 2^k$. Entonces (7.3.12) se convierte en

$$b_{2^k} = 2b_{2^{k-1}} + 2.$$

Ahora

$$\begin{aligned} b_{2^k} &= 2b_{2^{k-1}} + 2 \\ &= 2[2b_{2^{k-2}} + 2] + 2 \\ &= 2^2 b_{2^{k-2}} + 2^2 + 2 = \cdots \\ &= 2^{k-1} b_{2^1} + 2^{k-1} + 2^{k-2} + \cdots + 2 \\ &= 2^{k-1} + 2^{k-1} + \cdots + 2 \\ &= 2^{k-1} + 2^k - 2 \\ &= n - 2 + \frac{n}{2} = \frac{3n}{2} - 2. \end{aligned}$$

39. Se usa el siguiente hecho, que se puede verificar considerando los casos x par y x impar:

$$\left\lceil \frac{3x}{2} - 2 \right\rceil + \left\lceil \frac{3(x+1)}{2} - 2 \right\rceil = 3x - 2 \text{ for } x = 1, 2, \ldots.$$

Sea a_n el número de comparaciones requeridas por el algoritmo en el peor caso. Los casos $n = 1$ y $n = 2$ se pueden verificar directamente. (El caso $n = 2$ es el paso base).

Paso inductivo Suponga que $a_k \le a_k \le \lceil (3k/2) - 2 \rceil$ para $2 \le k < n$. Debe demostrarse que la desigualdad se cumple para $k = n$.

Si n es impar, el algoritmo hace una partición del arreglo en subclases de tamaño $(n-1)/2$ y $(n+1)/2$. Ahora

$$\begin{aligned} a_n &= a_{(n-1)/2} + a_{(n+1)/2} + 2 \\ &\le \left\lceil \frac{(3/2)(n-1)}{2} - 2 \right\rceil \\ &\quad + \left\lceil \frac{(3/2)(n+1)}{2} - 2 \right\rceil + 2 \\ &= \frac{3(n-1)}{2} - 2 + 2 = \frac{3n}{2} - \frac{3}{2} \\ &= \left\lceil \frac{3n}{2} - 2 \right\rceil. \end{aligned}$$

El caso n par se maneja de manera similar.

48. $\Theta(n)$

49. Si $n = 1$, *ordenar* sólo regresa; por lo tanto, todos los ceros preceden a todos los unos. El paso base queda probado.

Suponga que para una entrada de tamaño $n - 1$, después de invocar a *ordenar* todos los ceros preceden a todos los unos. Suponga que se invoca *ordenar* con una entrada de tamaño n. Si el primer elemento es un uno, se intercambia con el último. Luego se llama a *ordenar* de manera recursiva para los primeros $n - 1$ elementos. Por la suposición inductiva, dentro de los primeros $n - 1$ elementos todos los ceros preceden a todos los unos. Como el último elemento es uno, todos los ceros preceden a todos los unos para los n elementos. Si el primer elemento es cero, se llama a *ordenar* de manera recursiva para los últimos $n - 1$ elementos. Por la suposición inductiva, dentro de los últimos $n - 1$ elementos, todos los ceros preceden a todos los unos. Como el primer elemento es cero, todos los ceros preceden a todos los unos para los n elementos. En cualquier caso, *ordenar* produce como salida una versión rearreglada de la sucesión de entrada en la que todos los ceros preceden a todos los unos, y el paso inductivo queda completo.

54. Si $n = 2^k$,

$$a_{2^k} = 3a_{2^{k-1}} + 2^k,$$

de manera que

$$\begin{aligned} a_n = a_{2^k} &= 3a_{2^{k-1}} + 2^k \\ &= 3[3a_{2^{k-2}} + 2^{k-1}] + 2^k \\ &= 3^2 a_{2^{k-2}} + 3 \cdot 2^{k-1} + 2^k \\ &\vdots \\ &= 3^k a_{2^0} + 3^{k-1} \cdot 2^1 + 3^{k-2} \cdot 2^2 + \cdots \\ &\quad + 3 \cdot 2^{k-1} + 2^k \\ &= 3^k + 2(3^k - 2^k) \qquad (*) \\ &= 3 \cdot 3^k - 2 \cdot 2^k \\ &= 3 \cdot 3^{\lg n} - 2n. \end{aligned}$$

La línea (*) se obtiene de la ecuación

$$(a-b)(a^{k-1}b^0 + a^{k-2}b^1 + \cdots + a^1 b^{k-2} + a^0 b^{k-1}) = a^k - b^k$$

con $a = 3$ y $b = 2$.

56. $b_n = b_{\lfloor (1+n)/2 \rfloor} + b_{\lfloor n/2 \rfloor} + 3$

59. $b_n = 4n - 3$

62. Se demostrará que $b_n \le b_{n+1}$, $n = 1, 2, \ldots$. Se tiene la relación de recurrencia

$$b_n = b_{\lfloor (1+n)/2 \rfloor} + b_{\lfloor n/2 \rfloor} + c_{\lfloor (1+n)/2 \rfloor, \lfloor n/2 \rfloor}.$$

Paso base $b_2 = 2b_1 + c_{1,1} \ge 2b_1 \ge b_1$

Paso inductivo Suponga que la afirmación se cumple para $k < n$. Si n es par, se tiene $b_n = 2b_{n/2} + c_{n/2,n/2}$; entonces

$$\begin{aligned} b_{n+1} &= b_{(n+2)/2} + b_{n/2} + c_{(n+2)/2,n/2} \\ &\ge b_{n/2} + b_{n/2} + c_{n/2,n/2} = b_n. \end{aligned}$$

El caso de n impar es similar.

64.
```
ex64(s, i, j) {
    if (i == j)
      return
    m = ⌊(i + j)/2⌋
    ex64(s, i, m)
    ex64(s, m + 1, j)
    combinar(s, i, m, j)
}
```

67. Se prueba la desigualdad usando inducción matemática.

Paso base $a_1 = 0 \le 0 = b_1$

Paso inductivo Suponga que $a_k \le b_k$ para $k < n$. Entonces

$$\begin{aligned} a_n &\le a_{\lfloor n/2 \rfloor} + a_{\lfloor (n+1)/2 \rfloor} + 2 \lg n \\ &\le b_{\lfloor n/2 \rfloor} + b_{\lfloor (n+1)/2 \rfloor} + 2 \lg n = b_n. \end{aligned}$$

70. Sea $c = a_1$. Si n es una potencia de m, digamos $n = m^k$, entonces

$$\begin{aligned} a_n = a_{m^k} &= a_{m^{k-1}} + d \\ &= [a_{m^{k-2}} + d] + d \\ &= a_{m^{k-2}} + 2d \\ &\vdots \\ &= a_{m^0} + kd = c + kd. \end{aligned}$$

Un valor arbitrario de n cae entre dos potencias de m, digamos

$$m^{k-1} < n \le m^k.$$

Esta desigualdad implica que

$$k - 1 < \log_m n \le k.$$

Como la sucesión a es no decreciente,

$$a_{m^{k-1}} \le a_n \le a_{m^k}.$$

Ahora

$$\begin{aligned}\Omega(\log_m n) &= c + (-1 + \log_m n)d \le c + (k-1)d \\ &= a_{m^{k-1}} \le a_n\end{aligned}$$

y

$$\begin{aligned}a_n &\le a_{m^k} = c + kd \\ &\le c + (1 + \log_m n)d = O(\log_m n).\end{aligned}$$

Entonces, $a_n = \Theta(\log_m n)$. Por el ejemplo 4.3.6, $a_n = \Theta(\lg n)$.

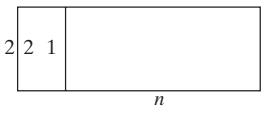
Capítulo 7 Autoevaluación

1. *a)* 3, 5, 8, 12 *b)* $a_1 = 3$ *c)* $a_n = a_{n-1} + n$

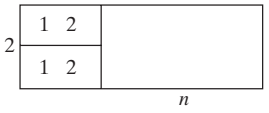
2. $A_n = (1.17)A_{n-1}$, $A_0 = 4000$

3. Sea X un conjunto de n elementos y elija $x \in X$. Sea k un entero fijo, $0 \le k \le n - 1$. Se puede seleccionar un subconjunto de k elementos Y de $X - \{x\}$ de $C(n - 1, k)$ maneras. Habiendo hecho esto, se puede hacer una partición de Y de P_k maneras. Esta partición junto con $X - Y$ son una partición de X. Como todas las particiones de X se pueden generar de esta manera, se obtiene la relación de recurrencia deseada.

4. Si el primer dominó se coloca como se muestra, hay a_{n-1} maneras de cubrir el tablero de $2 \times (n - 1)$ que queda.



Si los primeros dos dominós se colocan como se muestra, hay a_{n-2} maneras de cubrir el tablero de $2 \times (n - 2)$ que queda.



Se deduce que $a_n = a_{n-1} + a_{n-2}$

Por inspección, $a_1 = 1$ y $a_2 = 2$. Como $\{a_n\}$ satisface la misma relación de recurrencia que la sucesión de Fibonacci y $a_1 = f_2$ y $a_2 = f_3$, entonces $a_i = f_{i+1}$ para $i = 1, 2, \ldots$.

5. Sí

6. $a_n = 2(-2)^n - 4n(-2)^n$

7. $a_n = 3 \cdot 5^n + (-2)^n$

8. Considere una cadena de longitud n que contiene un número par de unos y que comienza con 0. La cadena que sigue al 0 puede ser cualquier cadena de longitud $n - 1$ que contiene un número par de unos, y hay c_{n-1} cadenas de éstas. Una cadena de longitud n que contiene un número par de unos que comienza con 2 puede ir seguida de cualquier cadena de longitud $n - 1$ que contiene un número par de unos, y hay c_{n-1} cadenas de este tipo. Una cadena de longitud n que contiene un número par de unos que comienza con 1 puede ir seguida por cualquier cadena de longitud $n - 1$ que contiene un número impar de unos. Como hay 3^{n-1} cadenas en total de longitud $n - 1$ y c_{n-1} de ellas contienen un número par de unos, se tienen $3^{n-1} - c_{n-1}$ cadenas de longitud $n - 1$ que contienen un número impar de unos. Se deduce que

$$c_n = 2c_{n-1} + 3^{n-1} - c_{n-1} = c_{n-1} + 3^{n-1}.$$

Una condición inicial es $c_1 = 2$, ya que hay dos cadenas (0 y 2) que contienen un número par (a saber, cero) de unos.

Se puede resolver la relación de recurrencia iterando:

$$\begin{aligned}c_n &= c_{n-1} + 3^{n-1} = c_{n-2} + 3^{n-2} + 3^{n-1} \\ &\vdots \\ &= c_1 + 3^1 + 3^2 + \cdots + 3^{n-1} \\ &= 2 + \frac{3^n - 3}{3 - 1} = \frac{3^n + 1}{2}.\end{aligned}$$

9. $b_n = b_{n-1} + 1$, $b_0 = 0$

10. $b_1 = 1$, $b_2 = 2$, $b_3 = 3$

11. $b_n = n$

12. $n(n + 1)/2 = O(n^2)$. El algoritmo dado es más rápido que la técnica directa y, por lo tanto, se prefiere.

Sección 8.1 Repaso

1. Una gráfica no dirigida consiste en un conjunto V de vértices y un conjunto E de aristas tal que cada arista $e \in E$ está asociada con un par no ordenado de vértices.

2. La amistad se puede modelar por una gráfica no dirigida, donde los vértices denotan personas y se coloca una arista entre dos personas si son amigos.

3. Una gráfica dirigida consiste en un conjunto V de vértices y un conjunto E de aristas, donde cada arista $e \in E$ está asociada con un par ordenado de vértices.

4. La precedencia se puede modelar por una gráfica dirigida, en la que los vértices denotan las tareas y se coloca una arista dirigida de la tarea t_i a la tarea t_j si t_i debe realizarse antes que t_j.

5. Si la arista e está asociada con los vértices v y w, se dice que e incide en v y w.

6. Si la arista e está asociada con los vértices v y w, se dice que v y w inciden en e.

7. Si la arista e está asociada con los vértices v y w, se dice que v y w son adyacentes.

8. Las aristas paralelas inciden en el mismo par de vértices.

9. Una arista incidente en un sólo vértice se llama lazo.

10. Un vértice que no incide en ninguna arista se llama vértice aislado.

11. Una gráfica simple es una gráfica sin lazos ni aristas paralelas.

12. Una gráfica ponderada es una gráfica con números asignados a las aristas.

13. Un mapa con distancias se puede modelar como una gráfica ponderada. Los vértices son las ciudades; las aristas, son carreteras que las comunican; y los números sobre las aristas, la distancia entre ellas.

14. La longitud de una trayectoria en una gráfica ponderada es la suma de los pesos de sus aristas.

15. Una gráfica de similitud tiene una función de disimilitud s donde $s(v, w)$ mide la disimilitud de los vértices v y w.

16. Un cubo-n tiene 2^n vértices etiquetados $0, 1, \ldots, 2^n - 1$. Una arista conecta dos vértices si la representación binaria de sus etiquetas difiere exactamente en un bit.

17. Una computadora serial ejecuta una instrucción a la vez.

18. Un algoritmo serial ejecuta una instrucción a la vez.

19. Una computadora paralela puede ejecutar varias instrucciones a la vez.

20. Un algoritmo paralelo puede ejecutar varias instrucciones a la vez.

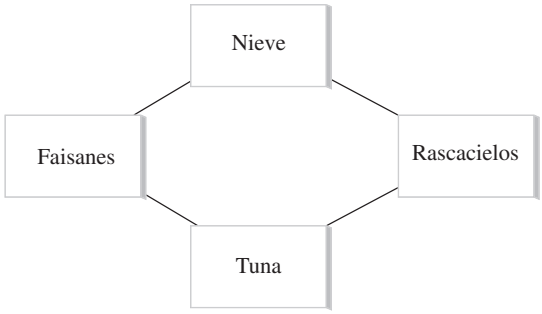
21. La gráfica completa sobre n vértices tiene una arista entre cada par de vértices distinto. Se denota por K_n.

22. Una gráfica $G = (V, E)$ es bipartita si existen subconjuntos V_1 y V_2 (tal vez alguno vacío) de V tales que $V_1 \cap V_2 = \emptyset, V_1 \cup V_2 = V$, y cada arista en E es incidente en un vértice en V_1 y un vértice en V_2.

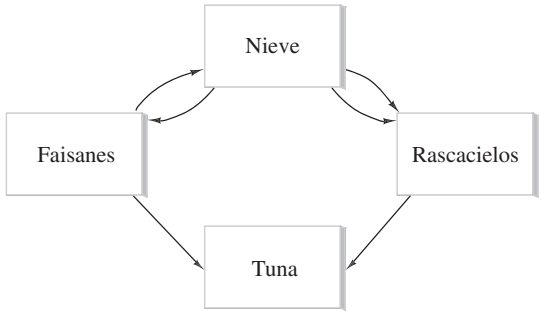
23. La gráfica bipartita completa sobre m y n vértices tiene conjuntos ajenos de vértices V_1 con m vértices y V_2 con n vértices. Hay una arista entre cada par de vértices distinto v_1 y v_2, donde $v_1 \in V_1$ y $v_2 \in V_2$. Se denota por $K_{m,n}$.

Sección 8.1

1. La gráfica es una gráfica simple no dirigida.



4. La gráfica es una gráfica no simple dirigida.

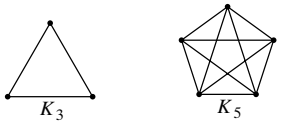


5. Dado que un número impar de aristas toca algunos vértices (c y d), no hay trayectoria de a a a que pase por cada arista exactamente una vez.

8. $(a, c, e, b, c, d, e, f, d, b, a)$

11. $V = \{v_1, v_2, v_3, v_4\}$. $E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$. e_1 y e_6 son aristas paralelas. e_5 es un lazo. No hay vértices aislados. G no es una gráfica simple. e_1 incide en v_1 y v_2.

14.

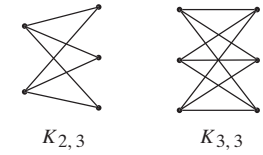


17. Bipartita. $V_1 = \{v_1, v_2, v_5\}$, $V_2 = \{v_3, v_4\}$.

20. No bipartita

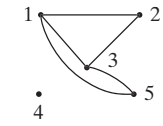
23. Bipartita. $V_1 = \{v_1\}$, $V_2 = \{v_2, v_3\}$.

24.

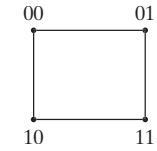


27. (b, c, a, d, e)

32. Dos clases

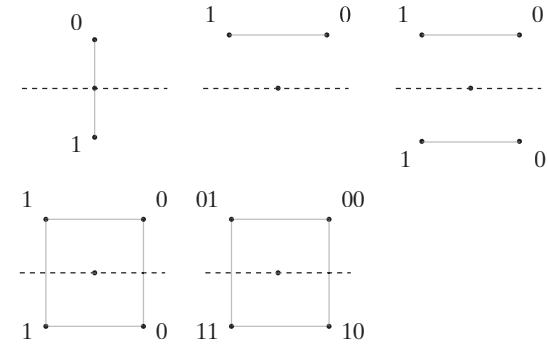


37.

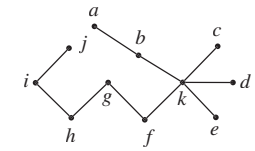


40. n

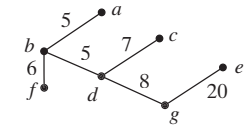
43.



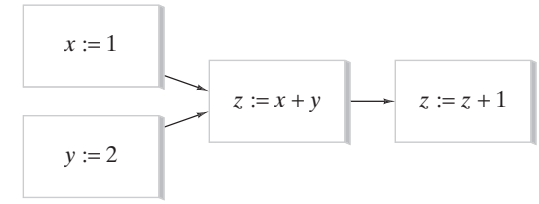
46.



49.



50.



Sección 8.2 Repaso

1. Una trayectoria es una sucesión alternante de vértices y aristas.

$$(v_0, e_1, v_1, e_2, v_2, \ldots, v_{n-1}, e_n, v_n),$$

en la que la arista e_i incide en los vértices v_{i-1} y v_i para $i = 1, \ldots, n$.

2. Una trayectoria simple es una trayectoria sin vértices repetidos.

3. (1, 2, 3, 1)

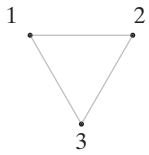
4. Un ciclo es una trayectoria de longitud diferente de cero de v a v sin aristas repetidas.

5. Un ciclo simple es un ciclo del vértice v al vértice v en donde, excepto por el vértice inicial y final que son ambos iguales a v, no hay vértices repetidos.

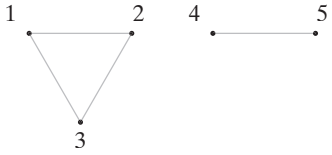
6. (1, 2, 3, 1, 4, 5, 1)

7. Una gráfica es conexa si, dados cualesquiera vértices v y w, existe una trayectoria de v a w.

8.



9.



10. Sea $G = (V, E)$ una gráfica. (V', E') es una subgráfica de G si $V' \subset V$, $E' \subset E$., y para cada arista $e' \in E'$, si e' es incidente en v' y w', entonces v' y $w' \in V'$.

11. La gráfica del ejercicio 8 es una subgráfica de la gráfica del ejercicio 9.

12. Sea G una gráfica y sea v un vértice en G. La subgráfica G' de G consiste en todas las aristas y vértices de G que están contenidas en alguna trayectoria que comienza en v se llama la componente de G que contiene a v.

13. La gráfica del ejercicio 8 es una componente de la gráfica del ejercicio 9.

14. Una.

15. El grado del vértice v es el número de aristas que inciden en v.

16. Un ciclo de Euler en una gráfica G es un ciclo que incluye todas las aristas y todos los vértices de G.

17. Una gráfica G tiene un ciclo de Euler si y sólo si G es conexa y el grado de cada vértice es par.

18. La gráfica del ejercicio 8 tiene un ciclo de Euler (1, 2, 3, 1).

19. La gráfica del ejercicio 9 no tiene un ciclo de Euler porque no es conexa.

20. La suma de los grados de los vértices de una gráfica es igual al doble del número de aristas en la gráfica.

21. Sí.

22. La gráfica es conexa y v y w son los únicos vértices que tienen grado impar.

23. Sí.

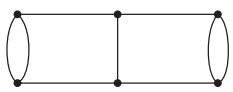
Sección 8.2

1. Ciclo, ciclo simple

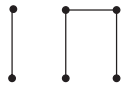
4. Ciclo, ciclo simple

7. Trayectoria simple

10.



13.



16. Suponga que existe tal gráfica con vértices *a, b, c, d, e, f*. Suponga que los grados de a y b son 5. Como la gráfica es simple, los grados de *c, d, e* y *f* son, cada uno, al menos 2; entonces no existe esa gráfica.

19. $(a, a), (b, c, g, b), (b, c, d, f, g, b),$
$(b, c, d, e, f, g, b), (c, g, f, d, c),$
$(c, g, f, e, d, c), (d, f, e, d)$

22. Todo vértice tiene grado 4.

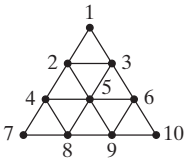
24. $G_1 = (\{v_1\}, \emptyset)$
$G_2 = (\{v_2\}, \emptyset)$
$G_3 = (\{v_1, v_2\}, \emptyset)$
$G_4 = (\{v_1, v_2\}, \{e_1\})$

27. Hay 17 subgráficas.

28. No tiene ciclo de Euler.

31. No tiene ciclo de Euler.

34. Para



un ciclo de Euler es (10, 9, 6, 5, 9, 8, 5, 4, 8, 7, 4, 2, 5, 3, 2, 1, 3, 6, 10). El método generaliza.

37. $m = n = 2$ or $m = n = 1$

39. d y e son los únicos vértices de grado impar.

42. El argumento es similar al de la prueba del teorema 8.2.23.

45. Verdadera. En la trayectoria, para todas las a repetidas,

$$(\ldots, a, \ldots, b, a, \ldots)$$

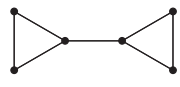
se elimina $a, \ldots, b$.

47. Suponga que $e = (v, w)$ es un ciclo. Entonces hay una trayectoria P de v a w que no incluye e. Sean x y y vértices en $G - \{e\}$. Como G es conexa, existe una trayectoria P' en G de v a w. Sustituya cualquier ocurrencia de e en P' por P. La trayectoria obtenida de v a w está en $G - \{e\}$. Por lo tanto, $G - \{e\}$ es conexa.

50. La unión de todas las subgráficas conexas que contienen a G' es una componente.

53. Sea G una gráfica simple, no conexa con n vértices que tiene el máximo número de aristas. Demuestre que G tiene dos componentes. Si una componente tiene i vértices, demuestre que las componentes son K_i y K_{n-i}. Utilice el ejercicio 11 de la sección 8.1 para encontrar una fórmula para el número de aristas en G como función de i. Demuestre que el máximo ocurre cuando $i = 1$.

55.



58. Modifique las demostraciones de los teoremas 8.2.17 y 8.2.18.

61. Utilice los ejercicios 58 y 60.

64. Primero se cuenta el número de trayectorias

$$(v_0, v_1, \ldots, v_k)$$

de longitud $k \geq 1$. El primer vértice v_0 puede elegirse de n maneras.

Cada vértice subsiguiente puede elegirse de $n-1$ maneras (ya que debe ser diferente de su predecesor). Entonces el número de trayectorias de longitud k es $n(n-1)^k$.

El número de trayectorias de longitud k, $1 \le k \le n$, es

$$\sum_{k=1}^{n} n(n-1)^k = n(n-1)\frac{(n-1)^k - 1}{(n-1)-1}$$
$$= \frac{n(n-1)[(n-1)^k - 1]}{n-2}.$$

68. Si v es un vértice en V, la trayectoria que consiste en v y ninguna arista es una trayectoria de v a v; entonces $v\ R\ v$ para todo vértice v en V. Por lo tanto, R es reflexiva.

Suponga que $v\ R\ w$. Entonces existe una trayectoria $(v_0, \ldots, v_n)$, donde $v_0 = v$ y $v_n = w$. Ahora $(v_0, \ldots, v_n)$ es una trayectoria de w a v, y así $w\ R\ v$. Por lo tanto, R es simétrica.

Suponga que $v\ R\ w$ y $w\ R\ x$. Entonces existe una trayectoria P_1 de v a w y una trayectoria de P_2 de w a x. Ahora P_1 seguida por P_2 es una trayectoria de v a x, y entonces $v\ R\ x$. Por lo tanto, R es transitiva.

Dado que R es reflexiva, simétrica y transitiva en V, R es una relación de equivalencia en V.

70. 2.

73. Sea s_n el número de trayectorias de longitud n de v_1 a v_1. Se demuestra que las sucesiones $s_1, s_2, \ldots$ y $f_1, f_2, \ldots$ satisfacen la misma relación de recurrencia, $s_1 = f_2$ y $s_2 = f_3$, de donde se deduce que $s_n = f_{n+1}$ para $n \ge 1$.

Si $n = 1$, existe una trayectoria de longitud 1 de v_1 a v_1, a saber, el lazo en v_1; entonces $s_1 = f_2$.

Si $n = 2$, hay dos trayectorias de longitud 2 de v_1 a v_1: (v_1, v_1, v_1) y (v_1, v_2, v_1); entonces $s_2 = f_3$.

Suponga que $n > 2$. Considere una trayectoria de longitud n de v_1 a v_1. La trayectoria debe comenzar con el lazo (v_1, v_1) o la arista (v_1, v_2).

Si la trayectoria comienza con el lazo, el resto de ella debe ser una trayectoria de longitud $n-1$ de v_1 a v_1. Como hay s_{n-1} trayectorias de éstas, hay s_{n-1} trayectorias de longitud n de v_1 a v_1 que comienzan $(v_1, v_1, \ldots)$.

Si la trayectoria comienza con la arista (v_1, v_2), la siguiente arista en la trayectoria debe ser (v_2, v_1). El resto de la trayectoria debe ser de longitud $n-2$ de v_1 a v_1. Como hay s_{n-2} de estas trayectorias, hay s_{n-2} trayectorias de longitud n de v_1 a v_1 que comienzan $(v_1, v_2, v_1, \ldots)$.

Dado que cualquier trayectoria de longitud $n > 2$ de v_1 a v_1 comienza con el lazo (v_1, v_1) o la arista $(v_1\ v_2)$, se deduce que

$$s_n = s_{n-1} + s_{n-2}.$$

Como las sucesiones $s_1, s_2, \ldots$ y $f_1, f_2, \ldots$ satisfacen la misma relación de recurrencia, $s_1 = f_2$ y $s_2 = f_3$, sigue que $s_n = f_{n+1}$ para $n \ge 1$.

75. Suponga que todo vértice tiene un arista que sale. Elija un vértice v_0. Siga una arista hacia afuera de v_0 a un vértice v_i. (Por suposición, esa arista existe). Continúe por una arista que sale de v_i a un vértice v_{i+1}. Como hay un número infinito de vértices, en algún momento se regresará a un vértice visitado antes. En este punto, se habrá descubierto un ciclo, lo cual es una contradicción. Por lo tanto, una *gad* tiene al menos un vértice sin aristas que salen.

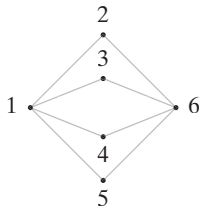
Sección 8.3 Repaso

1. Un ciclo de Hamilton en una gráfica G es un ciclo que contiene cada vértice en G exactamente una vez, excepto por el vértice de inicio y fin que aparece dos veces.

2. La gráfica de la figura 8.3.9 tiene un ciclo de Hamilton y un ciclo de Euler. Los ciclos de Hamilton y de Euler son la misma gráfica.

3. La gráfica de la figura 8.3.2 tiene un ciclo de Hamilton, pero no un ciclo de Euler. El ciclo de Hamilton se muestra en la figura 8.3.3. La gráfica no tiene un ciclo de Euler porque todos los vértices tienen grado impar.

4. La gráfica



tiene un ciclo de Euler porque es conexa y cada vértice tiene grado par. No tiene un ciclo de Hamilton. Para probar esto, se da un argumento por contradicción. Suponga que la gráfica tiene un ciclo de Hamilton. Entonces, como los vértices 2, 3, 4 y 5 tienen grado 2, todas las aristas en la gráfica tendrían que estar incluidas en un ciclo de Hamilton. Como la gráfica en sí no es un ciclo, se tiene una contradicción.

5. La gráfica que consiste en dos vértices y ninguna arista no tiene un ciclo de Hamilton ni un ciclo de Euler porque no es conexa.

6. El problema de agente viajero es: Dada una gráfica ponderada G, encuentre un ciclo de Hamilton de longitud mínima en G. El problema del ciclo de Hamilton sencillamente pide un ciclo de Hamilton: cualquiera sirve. El problema del agente viajero no sólo pide un ciclo de Hamilton, sino también uno de longitud mínima.

7. Un ciclo simple.

8. Un código Gray es una sucesión $s_1, s_2, \ldots, s_{2^n}$, donde cada s_i es una cadena de n bits que satisface lo siguiente:

- Cada cadena de n bits aparece en algún lado de la sucesión.
- s_i y s_{i+1} difieren exactamente en un bit, $i = 1, \ldots, 2^n - 1$,
- s_{2^n} y s_1 difieren exactamente en un bit.

9. Vea el teorema 8.3.6.

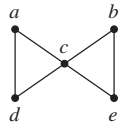
Sección 8.3

1. $(d, a, e, b, c, h, g, f, j, i, d)$

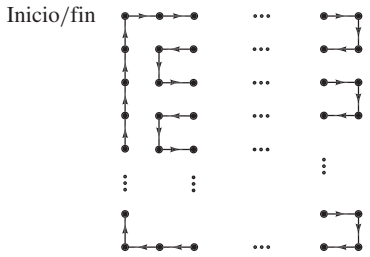
3. Se tendrán que eliminar dos aristas cada una en *b, d, i* y *k*, dejando $19 - 8 = 11$ aristas. Un ciclo de Hamilton tendría 12 aristas.

6. $(a, b, c, j, i, m, k, d, e, f, l, g, h, a)$

9.



12. Si n es par y $m > 1$ o si m es par y $n > 1$, existe un ciclo de Hamilton. El bosquejo muestra la solución en caso de que n sea par.



Si $n = 1$ o si $m = 1$, no hay ciclos y, en particular, no hay un ciclo de Hamilton. Suponga que n y m son ambos impares y que la gráfica tiene un ciclo de Hamilton. Como hay nm vértices, este ciclo tiene nm aristas; por lo tanto, el ciclo de Hamilton contiene un número impar de aristas. Sin embargo, se observa que en un ciclo de Hamilton debe haber tantas aristas que "suben" como aristas que "bajan" y tantas aristas a la "izquierda" como a la "derecha". Entonces un ciclo de Hamilton debe tener un número par de aristas. Esta contradicción demuestra que si n y m son ambos impares, la gráfica no tiene un ciclo de Hamilton.

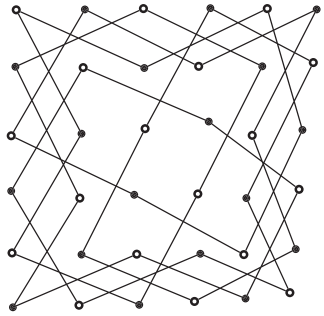
15. Cuando $m = n$ y $n > 1$.

18. Cualquier ciclo C en el cubo-n tiene longitud par ya que los vértices en C alternan entre un número par y un número impar de unos.

Suponga que el cubo-n tiene un ciclo simple de longitud m. Se acaba de ver que m es par. Ahora $m > 0$, por definición. Como el cubo-n es una grafica simple, $m \neq 2$. Por lo tanto, $m \geq 4$.

Ahora suponga que $m \geq 4$ y m es par. Sea G los primeros $m/2$ miembros del código Gray G_{n-1}. Entonces $0G$, $1G^R$ describe un ciclo simple de longitud m en el cubo-n.

21.



25. Sí. Si $(v_1, \ldots, v_{n-1}, v_n)$, $v_1 = v_n$, es un ciclo de Hamilton, $(v_1, \ldots, v_{n-1})$ es una trayectoria de Hamilton.

28. Sí, $(a, b, d, g, m, l, h, i, j, e, f, k, c)$.

31. Sí, $(i, j, g, h, e, d, c, b, a, f)$.

34. Sí, (a, c, d, f, g, e, b).

Sección 8.4 Repaso

1. Se etiqueta con 0 el vértice inicial y con ∞ los demás vértices. Sea T el conjunto de todos los vértices. Se elige $v \in T$ con la etiqueta mínima y se elimina v de T. Para cada $x \in T$ adyacente a v, se etiqueta de nuevo x con el mínimo entre su etiqueta actual y la etiqueta de $v + w(v, x)$, donde $w(v, x)$ es el peso de la arista (v, x). Esto se repite si $z \notin T$.

2. Vea el ejemplo 8.4.2.

3. Vea la prueba del teorema 8.4.3.

Sección 8.4

1. 7; (a, b, c, f)

4. 7; (b, c, f, j)

6. Se puede modelar un algoritmo para el ejemplo 8.4.2.

9. Modifique el algoritmo 8.4.1 de manera que comience por asignar el peso ∞ a cada arista que no existe. Después el algoritmo sigue como está escrito. Al terminar, $L(z)$ será igual a ∞ si no hay una trayectoria de a a z.

Sección 8.5 Repaso

1. Se ordenan los vértices y se etiquetan los renglones y columnas de una matriz con los vértices ordenados. El elemento en el renglón i y la columna j, $i \neq j$, es el número de aristas incidentes en i y j. Si $i = j$, el elemento es el doble del número de lazos que inciden en i. La matriz obtenida es la matriz de adyacencia de la gráfica.

2. Elemento ij en A^n es igual al número de trayectorias de longitud n del vértice i al vértice j.

3. Se ordenan los vértices y las aristas y se etiquetan los renglones de una matriz con los vértices y las columnas con las aristas. El elemento en el renglón v y la columna e es 1 si e incide en v, y 0 de otra manera. La matriz que resulta es la matriz de incidencia de la gráfica.

Sección 8.5

1.

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

4.

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

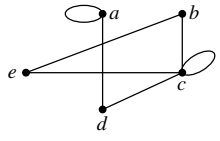
7.

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{cccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

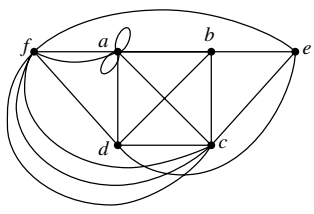
10.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

13.



16.

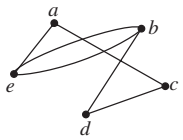


19. [Para K_5]

$$\begin{pmatrix} 4 & 3 & 3 & 3 & 3 \\ 3 & 4 & 3 & 3 & 3 \\ 3 & 3 & 4 & 3 & 3 \\ 3 & 3 & 3 & 4 & 3 \\ 3 & 3 & 3 & 3 & 4 \end{pmatrix}$$

22. La gráfica no es conexa.

24.



27. G no es conexa.

28. A causa de la simetría de la gráfica, si v y w son vértices en K_5, se tiene el mismo número de trayectorias de longitud n de v a v que las que hay de w a w. Entonces todos los elementos en la diagonal de A^n son iguales. De manera similar, todos los elementos fuera de la diagonal de A^n son iguales.

31. Si $n \geq 2$,

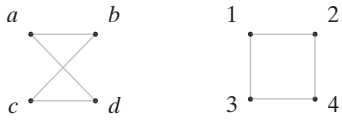
$$\begin{aligned} d_n &= 4a_{n-1} && \text{por el ejercicio 29} \\ &= 4\left(\frac{1}{5}\right)[4^{n-1} + (-1)^n] && \text{por el ejercicio 30.} \end{aligned}$$

La fórmula puede verificarse directamente para $n = 1$.

Sección 8.6 Repaso

1. Las gráficas G_1 y G_2 son isomorfas si existe una función f uno a uno y sobre de los vértices de G_1 a los vértices de G_2 y una función g uno a uno y sobre de las aristas de G_1 a las aristas de G_2, de manera que una arista e incide en v y w en G_1 si y sólo si la arista $g(e)$ incide en $f(v)$ y $f(w)$ en G_2.

2. Las siguientes gráficas



son isomorfas. Un isomorfismo está dado por $f(a) = 1$ $f(b) = 2$, $f(c) = 4$, $f(d) = 3$ y $g(a, b) = (1, 2)$, $g(b, c) = (2, 4)$, $g(c, d) = (4, 3)$, $g(d, a) = (3, 1)$.

3. Las siguientes gráficas

no son isomorfas; la primera gráfica tiene dos vértices y la segunda tiene tres.

4. Una propiedad P es una invariante si, siempre que G_1 y G_2 sean gráficas isomorfas, si G_1 tiene la propiedad P, entonces G_2 también tiene la propiedad P.

5. Para demostrar que dos gráficas no son isomorfas, se encuentra una invariante que tiene una de ellas y la otra no.

6. Dos gráficas son isomorfas si y sólo si para algunos ordenamientos de sus vértices, sus matrices de adyacencia son iguales.

7. Un arreglo rectangular de vértices.

Sección 8.6

1. Las gráficas no son isomorfas, ya que no tienen el mismo número de vértices.

4. Las gráficas no son isomorfas, porque G_2 tiene un vértice (3) de grado 4, pero G_1 no lo tiene.

7. Las gráficas no son isomorfas, ya que G_1 tiene un vértice de grado 2 y G_2 no.

10. Las gráficas son isomorfas: $f(a) = 5$, $f(b) = 3$. Defina $f(c) = 1$, $f(d) = 7$, $f(e) = 4$, $f(f) = 6$, $f(g) = 2$ $f(h) = 8$.

$$g((x, y)) = (f(x), f(y)).$$

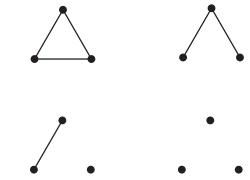
En los ejercicios 12 al 18 se usa la notación de la definición 8.6.1.

12. Si $(v_0, v_1, \ldots, v_k)$ es un ciclo simple de longitud k en G_1, entonces $(f(v_0), f(v_1), \ldots, f(v_k))$ es un ciclo simple de longitud k en G_2. [Los vértices $f(v_i)$, $i = 1, \ldots, k-1$, son distintos, ya que f es uno a uno].

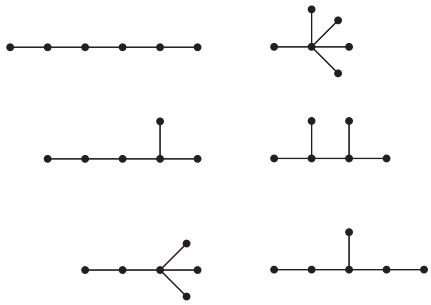
15. En la sugerencia para el ejercicio 12, se muestra que si $C = (v_0, v_1, \ldots, v_k)$, es un ciclo simple de longitud k en G_1, entonces $(f(v_0), f(v_1), \ldots, f(v_k))$, que aquí se denota por $f(C)$, es un ciclo simple de longitud k en G_1. Sean $C_1, C_2, \ldots, C_n$ los n ciclos simples de longitud k en G_1. Entonces $f(C_1), f(C_2), \ldots, f(C_n)$ son n ciclos simples de longitud k en G_2. Más aún, como f es uno a uno, $f(C_1), f(C_2), \ldots, f(C_n)$ son diferentes.

18. La propiedad es una invariante. Si $(v_0, v_1, \ldots, v_n)$ es un ciclo de Euler en G_1, entonces como g es sobre, $(f(v_0), f(v_1), \ldots, f(v_n))$ es un ciclo de Euler en G_2.

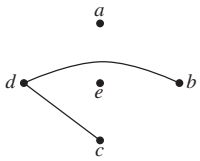
21.



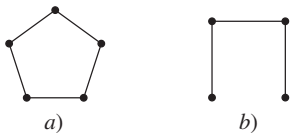
24.



26.



29.



32. Defina $g((v, w)) = (f(v), f(w))$

33. $f(a) = 1$, $f(b) = 2$, $f(c) = 3$, $f(d) = 2$

36. $f(a) = 1$, $f(b) = 2$, $f(c) = 3$, $f(d) = 1$

Sección 8.7 Repaso

1. Una gráfica que se puede dibujar en el plano sin que se crucen sus aristas.

2. Una región contigua. **3.** $f = e - v + 2$.

4. Aristas de la forma (v, v_1) y (v, v_2), donde v tiene grado 2 y $v_1 \neq v_2$.

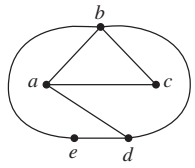
5. Dadas las aristas de la forma (v, v_1) y (v, v_2), donde v tiene grado 2 y $v_1 \neq v_2$, una serie de reducciones elimina el vértice v y sustituye (v, v_1) y (v, v_2) por (v_1, v_2).

6. Dos gráficas son homomorfas si se pueden reducir a gráficas isomorfas realizando una secuencia de reducciones en serie.

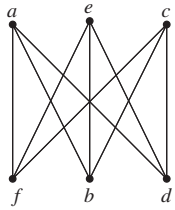
7. Una gráfica es plana si y sólo si no contiene una subgráfica homomorfa a K_5 o $K_{3,3}$.

Sección 8.7

1.

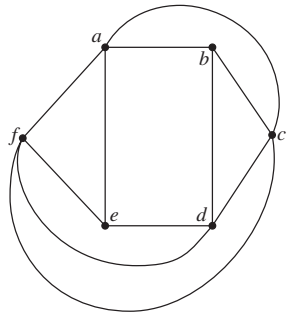


4.



es $K_{3,3}$

6. Plana

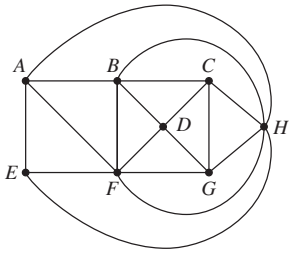


9. $2e = 2 + 2 + 2 + 3 + 3 + 3 + 4 + 4 + 5$, de manera que $e = 14$. $f = e - v + 2 = 14 - 9 + 2 = 7$.

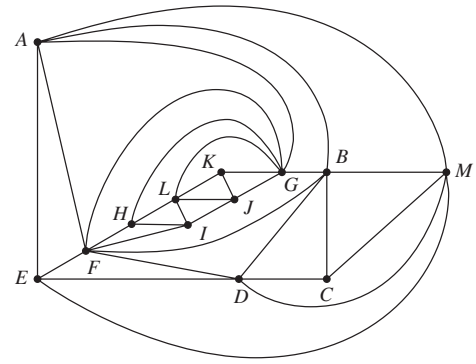
12. Una gráfica con cinco vértices o menos y un vértice de grado 2 es homomorfa a una gráfica con cuatro vértices o menos. Esta gráfica no puede contener una copia homomorfa de $K_{3,3}$ o K_5.

15. Si K_5 es plana, $e \leq 3v - 6$ se convierte en $10 \leq 3 \cdot 5 - 6 = 9$.

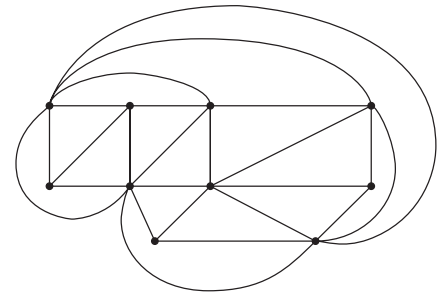
18.



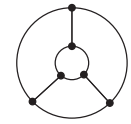
22.



25.



28. Contiene



31. Suponga que G no tiene un vértice de grado 5. Demuestre que $2e \geq 6v$. Ahora use el ejercicio 13 para obtener una contradicción.

Sección 8.8. Repaso

1. El juego de Locura instantánea consiste en cuatro cubos cuyas caras se pintan con uno de cuatro colores, rojo, blanco, azul o verde. El problema consiste en apilar los cubos, uno encima de otro, de manera que al ver la pila desde cualquier lado, frente, atrás, derecha o izquierda, se vean los cuatro colores.

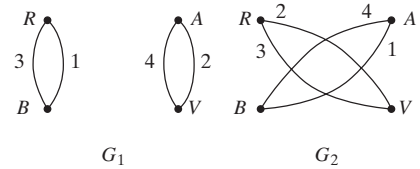
2. Dibuje una gráfica G, donde los vértices representan los cuatro colores y una arista con etiqueta i conecta dos vértices si las caras opuestas del cubo i tienen esos colores. Encuentre dos gráficas en donde

- Cada vértice tiene grado 2.
- Cada cubo representa una arista exactamente una vez en cada gráfica.
- Las gráficas no tienen aristas comunes.

Una gráfica representa la pila por enfrente y atrás, y la otra representa la pila por los lados derecho e izquierdo.

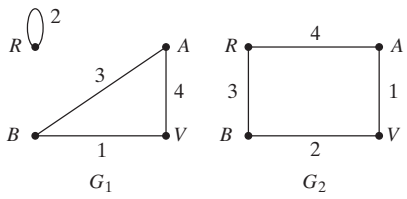
Sección 8.8

1.

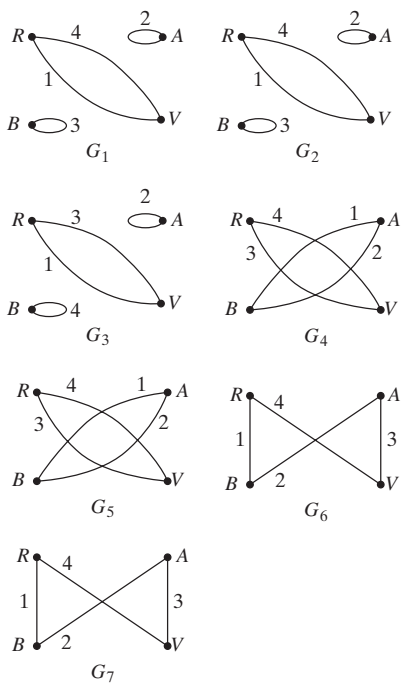


G_1 G_2

4.



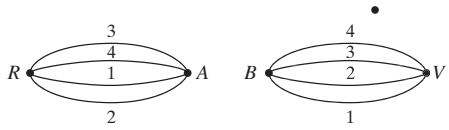
7. *a)*



b) Las soluciones son G_1, G_5; G_1, G_7; G_2, G_4; G_2, G_6 G_3, G_6; y G_3, G_7.

13. Una arista se puede elegir de $C(2 + 4 - 1, 2) = 10$ maneras. Las tres aristas con etiqueta 1 se pueden elegir de $C(3 + 10 - 1, 3) = 220$ maneras. Entonces el número total de gráficas es 220^4.

15.



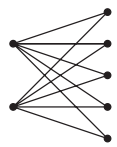
19. Según el ejercicio 14, sin contar lazos, cada vértice debe tener grado al menos 4. En la figura 8.8.5, sin contar lazos, el vértice *B* tiene grado 3 y, por lo tanto, la figura 8.8.5 no tiene una solución para la versión modificada de Locura instantánea. La figura 8.8.3 da una solución a la Locura instantánea original para la figura 8.8.5.

Capítulo 8 Autoevaluación

1. $V = \{v_1, v_2, v_3, v_4\}$. $E = \{e_1, e_2, e_3\}$. e_1 y e_2 son aristas paralelas. No hay lazos. v_1 es un vértice aislado. G no es una gráfica simple. e_3 incide en v_2 y v_4. v_2 incide en e_1, e_2 y e_3.

2. Existen vértices (*a* y *e*) de grado impar.

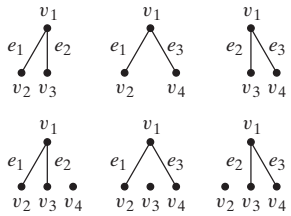
3.



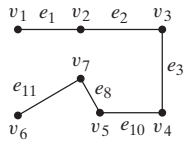
4. Si V_1 denota el conjunto de vértices que contienen un número par de unos y V_2, el conjunto de vértices que contienen un número impar de unos; cada arista incide en un vértice de V_1 y un vértice de V_2. Por lo tanto, el cubo-*n* es bipartita.

5. Es un ciclo.

6.



7.



8. No. Hay vértices de grado impar.

9. $(v_1, v_2, v_3, v_4, v_5, v_7, v_6, v_1)$.

10. (000, 001, 011, 010, 110, 111, 101, 100, 000).

11. Un ciclo de Hamilton tendría siete aristas. Suponga que la gráfica tiene un ciclo de Hamilton. Se tendría que eliminar tres aristas en el vértice *b* y una en el vértice *f*. Esto deja $10 - 4 = 6$ aristas, que no son suficientes para un ciclo de Hamilton. Por lo tanto, la gráfica no tiene un ciclo de Hamilton.

12. En un ciclo de Hamilton de peso mínimo, cada vértice debe tener grado 2. Por lo tanto, las aristas (*a, b*), (*a, j*), (*j, i*), (*i, h*), (*g, f*), (*f, e*) y (*e, d*) deben incluirse. No se puede incluir la arista (*b, h*) porque se completa un ciclo. Esto implica que deben incluirse las aristas (*h, g*) y (*b, c*). Como el vértice *g* ahora tiene grado 2, no se puede incluir la arista (*c, g*) o la (*g, d*). Entonces debe incluirse (*c, d*). Esto es un ciclo de Hamilton y el argumento muestra que es único. Por lo tanto, es mínimo.

13. 9. **14.** 11.

15. *(a, e, f, i, g, z,).* **16.** 12.

17.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	1	0	0	0	1	0
v_2	1	0	1	1	0	1	1
v_3	0	1	0	1	0	0	0
v_4	0	1	1	0	1	0	0
v_5	0	0	0	1	0	1	1
v_6	1	1	0	0	1	0	1
v_7	0	1	0	0	1	1	0

18.

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}
v_1	1	0	0	0	0	0	1	0	0	0	0
v_2	1	1	0	1	1	1	0	0	0	0	0
v_3	0	1	1	0	0	0	0	0	0	0	0
v_4	0	0	1	1	0	0	0	0	0	1	0
v_5	0	0	0	0	0	0	0	1	1	1	0
v_6	0	0	0	0	0	1	1	0	1	0	1
v_7	0	0	0	0	1	0	0	1	0	0	1

19. El número de trayectorias de longitud 3 de v_2 a v_3.

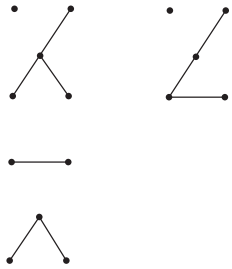
20. No. Cada arista incide al menos en un vértice.

21. Las gráficas son isomorfas. Los ordenamientos v_1, v_2, v_3, v_4, v_5 y w_3, w_1, w_4, w_2, w_5 producen matrices de adyacencia iguales.

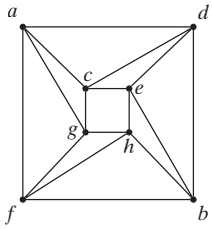
22. Las gráficas son isomorfas. Los ordenamientos $v_1, v_2, v_3, v_4, v_5, v_6$ y $w_3, w_6, w_2, w_5, w_1, w_4$ producen matrices de adyacencia iguales.

23.

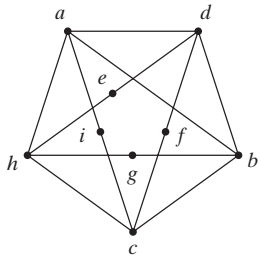
24.



25. La gráfica es plana:

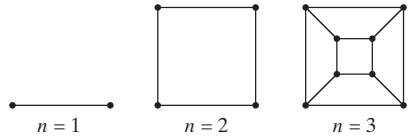


26. La gráfica no es plana; la siguiente subgráfica es homomorfa a K_5:



27. Una gráfica simple, plana, conexa, con aristas e y vértices v satisface $e \leq 3v - 6$ (ejercicio 13, sección 8.7). Si $e = 31$ y $v = 12$, la desigualdad no se satisface, de manera que este tipo de gráfica no puede ser plana.

28. Para $n = 1, 2, 3$, es posible dibujar el cubo-n en el plano sin aristas que se crucen:



Se da un argumento por contradicción para demostrar que el cubo-4 no es plano. Suponga que el cubo-4 es plano. Como todo ciclo tiene al menos cuatro aristas, cada cara está acotada por al menos cuatro aristas. Entonces el número de aristas que acotan las caras es al menos $4f$. En una gráfica plana, cada arista pertenece cuando mucho a dos ciclos frontera. Por lo tanto, $2e \geq 4f$. Usando la fórmula de Euler para gráficas, se encuentra que

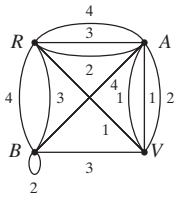
$$2e \geq 4(e - v + 2).$$

Para el cubo-4 se tiene $e = 32$ y $v = 16$, de manera que la fórmula de Euler se convierte en

$$64 = 2 \cdot 32 \geq 4(32 - 16 + 2) = 72,$$

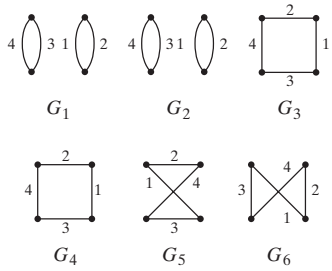
que es una contradicción. Entonces, el cubo-4 no es plano. El cubo-n, para $n > 4$, no es plano puesto que contiene al cubo-4.

29.



30. Vea las sugerencias para los ejercicios 31 y 32.

31.



Las dos aristas incidentes en B y G, etiquetadas con 1 en la gráfica del ejercicio 29, aquí se denotan por 1 e 1′.

32. El juego del ejercicio 29 tiene cuatro soluciones. Usando la notación del ejercicio 31, las soluciones son G_1, G_5; G_2, G_5; G_3, G_6, y G_4, G_6.

Sección 9.1 Repaso

1. Un árbol libre T es una gráfica simple que satisface lo siguiente: Si v y w son vértices en T, existe una trayectoria simple única de v a w.

2. Un árbol con raíz es un árbol en el que un vértice específico está designado como raíz.

3. El nivel de un vértice v es la longitud de la trayectoria simple de la raíz a v.

4. La altura de un árbol con raíz es el número máximo de nivel que ocurre.

5. Vea la figura 9.1.9.

6. En la estructura del árbol con raíz, cada vértice representa un archivo o una carpeta. Directamente abajo de la carpeta f están las carpetas y archivos contenidos en f.

7. Un código de Huffman se puede definir por un árbol con raíz. El código para un carácter en particular se obtiene siguiendo la trayectoria simple de la raíz al carácter. Cada arista se etiqueta con 0 o 1, y la sucesión de bits encontrada sobre la trayectoria simple es el código para ese carácter.

8. Suponga que existen n frecuencias. Si $n = 2$, se construye el árbol mostrado en la figura 9.1.11 y se detiene. De otra manera, sean f_i y f_j las frecuencias más pequeñas, y sustitúyalas en la lista por $f_i + f_j$. Construya de manera recursiva un árbol de código de Huffman óptimo usando la lista modificada. En el árbol obtenido, agregue dos aristas a un vértice etiquetado $f_i + f_j$ y etiquete los vértices agregados como f_i y f_j.

Sección 9.1

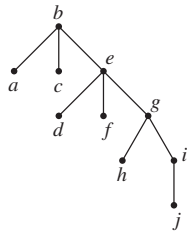
1. La gráfica es un árbol. Para cualesquiera vértices v y w, existe una trayectoria simple única de v a w.

4. La gráfica es un árbol. Para cualesquiera vértices v y w, existe una trayectoria simple única de v a w.

7. $n = 1$.

8. a-1; b-1; c-1; d-1; e-2; f-3; g-3; h-4; i-2; j-3; k-0.

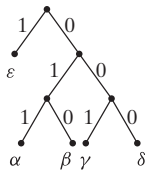
11. Altura $= 4$



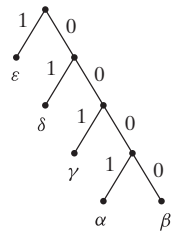
14. PEN. **17.** SALAD.

18. 0111100010. **21.** 01100001001000011111.

24.



27. Se muestra otro árbol en la sugerencia para el ejercicio 24.



32. Sea T un árbol. La raíz T está en un vértice arbitrario. Sea V el conjunto de vértices en niveles pares y sea W el conjunto de vértices en niveles impares. Como cada arista incide en un vértice en V y un vértice en W, T es una gráfica bipartita.

35. e, g.

38. El radio es la excentricidad de un centro. No necesariamente es cierto que $2r = d$ (vea la figura 9.1.5).

Sección 9.2 Repaso

1. Sea $(v_0, \ldots, v_{n-1}, v_n)$ una trayectoria de la raíz v_0 a v_n. Llamamos a v_{n-1} el padre de v_n.

2. Sea $(v_0, \ldots, v_n)$ una trayectoria de la raíz v_0 a v_n. Llamamos a $(v_i, \ldots, v_n)$ los descendientes de v_{i-1}.

3. v y w son hermanos si tienen el mismo padre.

4. Un vértice terminal es uno sin hijos.

5. Si v no es un vértice terminal, es un vértice interno.

6. Una gráfica acíclica es una gráfica sin ciclos.

7. Vea el teorema 9.2.3.

Sección 9.2

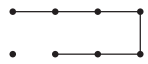
1. Cronos.

4. Apolo, Atenea, Hermes, Heracles.

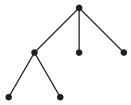
7. *b; d.* **10.** *e, f, g, j; j.*

13. *a, b, c, d, e.* **17.** Son hermanos.

22.



25.



27. Un solo vértice es un ciclo de longitud 0.

30. Cada componente de un bosque es conexa y acíclica y, por lo tanto, un árbol.

33. Suponga que G es conexa. Agregue aristas paralelas hasta que obtenga una gráfica G^* que tenga $n - 1$ aristas. Como G^* es conexa y tiene $n - 1$ aristas, por el teorema 7.2.3, G^* es acíclica. Pero agregar aristas paralelas introduce un ciclo. Contradicción.

36.

Sección 9.3 Repaso

1. Un árbol T es un árbol de expansión de una gráfica G si T es una subgráfica de G que contiene todos los vértices de G.

2. Una gráfica G tiene un árbol de expansión si y sólo si G es conexa.

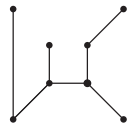
3. Seleccione un ordenamiento de los vértices. Seleccione el primer vértice y etiquételo como la raíz. Sea T este único vértice y ninguna arista. Agregue al árbol todas las aristas que inciden en este vértice que no producen ciclos cuando se agregan. Además, agregue los vértices que inciden en estas aristas. Repita este procedimiento con los vértices en el nivel 1, después los del nivel 2, etcétera.

4. Seleccione un ordenamiento de los vértices. Seleccione el primer vértice y etiquételo como raíz. Agregue al árbol una arista incidente en este vértice y agregue el vértice adicional v que incide en esta arista. Después, agregue una arista incidente en v que no produzca un ciclo al formar parte del árbol y añada el vértice adicional que incide en esta arista. Repita este procedimiento. Si en cualquier punto no se puede agregar una arista incidente en el vértice w, se regresa al padre p de w y se intenta con otra arista incidente en p. Cuando, por último, se regresa hasta la raíz y no se pueden agregar más aristas, la búsqueda a profundidad concluye.

5. Búsqueda a profundidad.

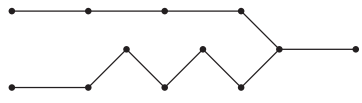
Sección 9.3

1.



4. La trayectoria (*h, f, e, g, b, d, c, a*).

7.



10. Es claro que el problema de las dos reinas no tiene solución. Para el problema de las tres reinas, por simetría, las únicas posiciones posibles en la primera columna son arriba a la izquierda y segunda desde arriba. Si el primer movimiento es primera columna, arriba a la izquierda, el segundo movimiento debe ser hasta abajo de la segunda columna. Ahora ningún movimiento es posible para la tercer columna. Si el primer movimiento es primera columna, segunda desde arriba, no hay movimiento posible en la columna dos. Por lo tanto, no existe solución al problema de las tres reinas.

13. Falso. Considere K_4.

16. Primero demuestre que la gráfica T construida es un árbol. Luego emplee inducción sobre el nivel de T para demostrar que T contiene todos los vértices de G.

19. Suponga que x incide en los vértices a y b. Eliminar x de T produce una gráfica no conexa con dos componentes, U y V. Los vértices a y b pertenecen a diferentes componentes, digamos, $a \in U$ y $b \in V$. Existe una trayectoria P de a a b en T'. Al moverse por P, en algún punto se encuentra una arista $y = (v, w)$ con $v \in U$, $w \in V$. Como agregar y a $T - \{x\}$ produce una gráfica conexa, $(T - \{x\}) \cup \{y\}$ es un árbol de expansión. Es claro que $(T' - \{y\}) \cup \{x\}$ es un árbol de expansión.

22. Suponga que T tiene n vértices. Si se agrega una arista a T, la gráfica T' que se obtiene es conexa. Si T' fuera acíclica, sería un árbol con n aristas y n vértices. Entonces T' contiene un ciclo. Si T' contiene dos o más ciclos, se podría producir una gráfica conexa T'' eliminando dos o más aristas de T'. Pero ahora T'' sería un árbol con n vértices y menos de $n - 1$ aristas, lo cual es imposible.

23.

$$\begin{array}{c} \\ (abca) \\ (acda) \\ (acdb) \\ (bcdeb) \end{array} \begin{array}{c} \begin{array}{cccccccc} e_1 & e_2 & e_6 & e_5 & e_3 & e_4 & e_7 & e_8 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

26. Entrada: una gráfica $G = (V, E)$ con n vértices
Salida: verdadero si G es conexa
falso si G no es conexa

```
es_conexa(V, E ) {
    T = bla(V,E) //búsqueda a lo ancho
    //T = (V', E')es un árbol de expansión que regresa bla
    if(|V'|) == n)
        return verdadero
    else
        return falso
}
```

Sección 9.4 Repaso

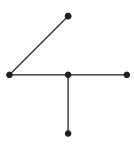
1. Un árbol de mínima expansión es un árbol de expansión con peso mínimo.

2. El algoritmo de Prim construye un árbol de mínima expansión agregando aristas de manera iterativa. El algoritmo comienza con un solo vértice. Después en cada iteración, agrega al árbol actual una arista con peso mínimo que no complete un ciclo.

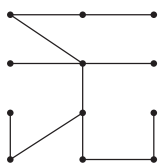
3. El algoritmo ambicioso optimiza la elección en cada iteración.

Sección 9.4

1.



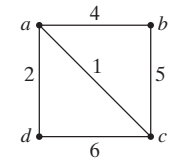
4.



10. Si v es el primer vértice que examina el algoritmo de Prim, la arista estará en el árbol de mínima expansión que construye el algoritmo.

13. Suponga que G tiene dos árboles de mínima expansión T_1 y T_2. Entonces, existe una arista x en T_1 que no está en T_2. Por el ejercicio 19, sección 9.3, existe una arista y en T_2 que no está en T_1 tal que $T_3 = (T_1 - \{x\}) \cup \{y\}$ y $T_4 = (T_2 - \{y\}) \cup \{x\}$ son árboles de expansión. Como x y y tienen pesos diferentes, T_3 o T_4 tiene menor peso que T_1. Ésta es una contradicción.

14. Falsa



16. Falsa. Considere K_5 con el peso de cada arista igual a 1.

20. Entrada: Las aristas E de una gráfica conexa ponderada de n vértices. Si e es una arista, $w(e)$ es igual al peso de e; si e no es una arista, $w(e)$ es igual a ∞ (un valor mayor que cualquier peso real).
Salida: Un árbol de expansión mínima.

```
kruskal(E, w, n) {
    V' = ∅
    E' = ∅
    T' = (V', E')
    while(|E'| < n − 1) {
        entre todas las aristas que, si se agregan a T' no
            completan un ciclo, se elige e = (v_i, v_j) con
            peso mínimo
        E' = E' ∪ {e}
        V' = V' ∪ {v_i, v_j}
        T' = (V', E')
    }
    return T'
}
```

23. Termina el algoritmo de Kruskal después de k iteraciones. Éste agrupa los datos en $n - k$ clases.

27. Se demuestra que $a_1 = 7$ y $a_2 = 3$ proporciona una solución. Se usa inducción sobre n para demostrar que la solución ambiciosa da una solución óptima para $n \geq 1$. Los casos $n = 1, 2, \ldots, 8$ se pueden verificar directamente.

Primero se demuestra que si $n \geq 9$, existe una solución óptima que contiene al menos un 7. Sea S' una solución óptima. Suponga que S' no contiene un 7. Como S' contiene cuando mucho dos unos (ya que S' es óptima), S' contiene al menos tres números 3. Se sustituyen los tres números 3 por un 7 y dos unos para obtener una solución S'. Como $|S| = |S'|$, S es óptima.

Si se elimina un 7 de S, se obtiene una solución S^* al problema de $(n - 7)$. Si S^* no fuera óptima, S no podría ser óptima. Entonces S^* es óptima. Por la suposición de inducción, la solución ambiciosa GS^* para el problema de $(n - 7)$ es óptima, de manera que $|S^*| = |GS^*|$. Observe que 7 junto con GS^* es la solución ambiciosa GS para el problema de n. Como $|GS| = |S|$, GS es óptima.

29. Suponga que el algoritmo ambicioso es óptimo para todas las denominaciones menores que $a_{m-1} + a_m$. Se usa inducción sobre n para demostrar que el algoritmo ambicioso es óptimo para toda n. (Se omite la recíproca.) Se puede suponer que $n \geq a_{m-1} + a_m$.

Considere una solución óptima $\mathcal{S}$ para n. Primero suponga que $\mathcal{S}$ usa al menos una moneda a_m. La solución, $\mathcal{S}$ con una moneda a_m eliminada, es óptima para $n - a_m$. (Si hubiera una solución para $n - a_m$ que ocupara menos monedas, se podría agregar una moneda a_m para obtener una solución para n que empleara menos monedas que $\mathcal{S}$, lo cual es imposible). Por la suposición inductiva, la solución ambiciosa para $n - a_m$ es óptima. Si se agrega una moneda a_m a la solución ambiciosa para $n - a_m$, se obtiene una solución $\mathcal{S}$ para n que usa el mismo número de monedas que $\mathcal{G}$. Por lo tanto, $\mathcal{G}$ es óptima. Pero $\mathcal{G}$ también es ambiciosa porque la solución ambiciosa comienza por eliminar una moneda a_m.

Ahora suponga que $\mathcal{S}$ no usa una moneda a_m. Sea i el índice más grande tal que $\mathcal{S}$ usa una moneda a_i. La solución, $\mathcal{S}$ con una moneda a_i eliminada, es óptima para $n - a_i$. Por la suposición de inducción, la solución ambiciosa para $n - a_i$ es óptima. Ahora

$$n \geq a_{m-1} + a_m \geq a_i + a_m,$$

de manera que $n - a_i \geq a_m$. Por lo tanto, la solución ambiciosa emplea al menos una moneda a_m. Entonces existe una solución óptima para $n - a_i$ que utiliza una moneda a_m. Si se agrega una moneda a_i a esta solución óptima, se obtiene una solución óptima para n que utiliza una moneda a_m. El argumento del párrafo anterior se puede repetir para demostrar que la solución ambiciosa es óptima.

Sección 9.5 Repaso

1. Un árbol binario es un árbol con raíz en el que cada vértice tiene un hijo, dos hijos o ningún hijo.

2. Un hijo izquierdo del vértice v es un hijo designado como "izquierdo".

3. Un hijo derecho del vértice v es un hijo designado como "derecho".

4. Un árbol binario completo es un árbol binario en el que cada vértice tiene dos o cero hijos.

5. $i + 1$.

6. $2i + 1$.

7. Si un árbol binario con altura h tiene t vértices terminales, entonces $\lg t \leq h$.

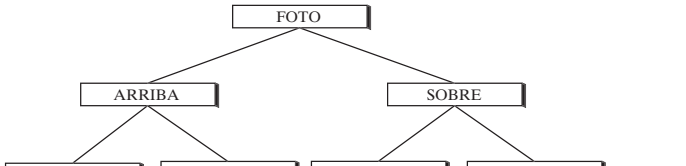
8. Un árbol de búsqueda binaria es un árbol binario T en el que se asocian datos con los vértices. Los datos están arreglados de manera que, para cada vértice v en T, cada dato en el subárbol izquierdo de v es menor que el dato en v, y cada dato en el subárbol derecho de v es mayor que el dato en v.

9. Vea las figuras 9.5.4 y 9.5.5.

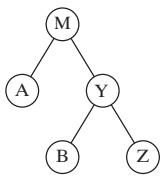
10. Se inserta el primer elemento de datos en un vértice y se etiqueta como raíz. Se inserta el siguiente dato en el árbol de acuerdo con los siguientes pasos. Se inicia en la raíz. Si el dato que se agrega es menor que el dato en el vértice actual, se mueve al hijo izquierdo y se repite; de otra manera, se mueve al hijo derecho y se repite. Si no hay hijos, se crea uno poniendo una arista incidente en él y el último vértice visitado, y se almacena el dato en el vértice agregado.

Sección 9.5

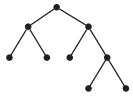
1.



4. Falso. Considere



5.



8. $mi + 1, (m - 1)i + 1$.

11. $t - 1$.

14. Balanceado.

17. Balanceado.

18. Un árbol de altura 0 tiene un vértice, de forma que $N_0 = 1$. En un árbol binario balanceado de altura 1, la raíz debe tener al menos un hijo. Si la raíz tiene exactamente un hijo, el número de vértices se minimiza. Por lo tanto, $N_1 = 2$. En un árbol binario balanceado de altura 2, debe haber una trayectoria de la raíz al vértice terminal de longitud 2. Esto explica tres vértices. Pero para que el árbol esté balanceado, la raíz debe tener dos hijos. Por lo tanto, $N_2 = 4$.

21. Suponga que hay n vértices en un árbol binario balanceado de altura h. Entonces

$$n \geq N_h = f_{h+3} - 1 > \left(\frac{3}{2}\right)^{h+2} - 1,$$

para $h \geq 3$.La igualdad viene del ejercicio 20 y la última desigualdad viene del ejercicio 27, sección 4.4. Por lo tanto,

$$n + 1 > \left(\frac{3}{2}\right)^{h+2}.$$

Tomando logaritmos base $3/2$ en cada lado se obtiene

$$\log_{3/2}(n + 1) > h + 2.$$

Por lo tanto,

$$h < [\log_{3/2}(n + 1)] - 2 = O(\lg n).$$

Sección 9.6 Repaso

1. Un recorrido preorden procesa los vértices de un árbol binario comenzando en la raíz y procesando de manera recursiva el vértice actual, el subárbol izquierdo del vértice y luego el subárbol derecho del vértice.

2. Entrada: *PT*, la raíz de un árbol binario
Salida: Depende de cómo se interprete "procesa"

```
preorden(PT) {
    if (PT está vacío)
        return
    procesa PT
    i = hijo izquierdo de PT
    preorden(i)
    d = hijo derecho de PT
    preorden(d)
}
```

3. El recorrido entreorden procesa el vértice de un árbol binario comenzando en la raíz y procesando de manera recursiva el subárbol izquierdo del vértice, el vértice actual, y luego el subárbol derecho del vértice.

4. Entrada: *PT*, la raíz de un árbol binario
Salida: Depende de cómo se interprete "procesa"

```
entreorden(PT) {
    if (PT está vacío)
        return
    i = hijo izquierdo de PT
    entreorden(i)
    procesa PT
    d = hijo derecho de PT
    entreorden(d)
}
```

5. El recorrido postorden procesa los vértices de un árbol binario comenzando por la raíz y procesando de manera recursiva el subárbol izquierdo del vértice, el subárbol derecho del vértice y luego el vértice actual.

6. Entrada: *PT*, la raíz de un árbol binario
Salida: Depende de cómo se interprete "procesa"

```
postorden(PT) {
    if (PT está vacío)
        return
    i = hijo izquierdo de PT
    postorden(i)
    d = hijo derecho de PT
    postorden(d)
    procesa PT
}
```

7. En la forma de prefijo de una expresión, un operador precede a sus operandos.

8. Notación polaca.

9. En la forma de entrefijo de una expresión, un operador está entre sus operandos.

10. En la forma de posfijo de una expresión, un operador sigue a sus operandos.

11. Notación polaca inversa.

12. No se necesitan paréntesis.

13. En una representación de árbol de una expresión, los vértices internos representan operadores, y estos últimos operan sobre los subárboles.

Sección 9.6

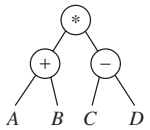
1.

preorden	*entreorden*	*postorden*
ABDCE	*BDAEC*	*DBECA*.

4.

preorden	*entreorden*	*postorden*
ABDCE	*EDCBA*	*EDCBA*.

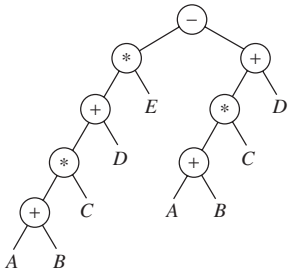
6.



prefijo: $* + AB - CD$

posfijo: $AB + CD - *$

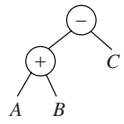
9.



prefijo: $- * + * + ABCDE + * + ABCD$

posfijo: $AB + C * D + E * AB + C * D + -$

11.

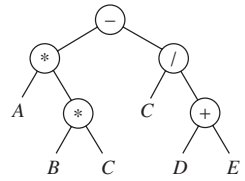


prefijo: $- + ABC$

entrefijo usual: $A + B - C$

entrefijo con paréntesis: $((A + B) - C)$

14.



prefijo: $- * A * BC/C + DE$

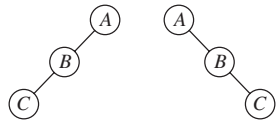
entrefijo usual: $A * B * C - C/(D + E)$

entrefijo con paréntesis: $((A * (B * C)) - (C/(D + E)))$

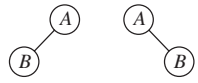
16. −4.

19. 0.

22.



25.



28. Entrada: *PT*, la raíz de un árbol binario
Salida: *PT*, la raíz del árbol binario modificado

```
intercam_hijos(PT) {
    if (PT está vacío)
        return
    intercambia hijos izquierdo y derecho de PT
    i = hijo izquierdo de PT
    intercam_hijos(i)
    d = hijo derecho de PT
    intercam_hijos(d)
}
```

30. Defina un *segmento inicial* de una cadena para que sean los primeros $i \geq 1$ caracteres para alguna i. Defina $r(x) = 1$, para $x = A, B, \ldots, Z$; y $r(x) = -1$, para $x = +, -, *, /$. Si $x_1 \cdots x_n$ es una cadena sobre $\{A, \ldots, Z, +, -, *, /\}$, defina

$$r(x_1 \cdots x_n) = r(x_1) + \cdots + r(x_n).$$

Entonces una cadena s es una cadena posfijo si y sólo si $r(s) = 1$ y $r(s') \geq 1$, para todos los segmentos s' de s.

33. Sea G la gráfica con conjunto de vértices $\{1, 2, \ldots, n\}$ y conjunto de aristas

$$\{(1, i) \mid i = 2, \ldots n\}.$$

El conjunto $\{1\}$ es una cubierta de vértices de G de tamaño 1.

36. Entrada: *PT*, la raíz de un árbol no vacío
Salida: Cada vértice del árbol tiene un campo *en_cubierta* que es verdadero si ese vértice está en la cubierta de vértices o falso si no está ahí.

```
cubierta_árbol(PT) {
    bandera = falso
    ptr = primer hijo de PT
    while(ptr es no vacío) {
        cubierta_árbol(ptr)
        if(en_cubierta de ptr == falso)
            bandera = verdadero
        ptr = siguiente hermano de ptr
    }
    en_cubierta de PT = bandera
}
```

Sección 9.7 Repaso

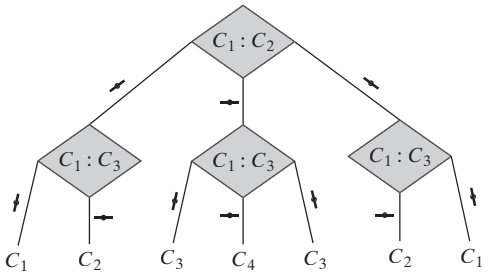
1. Un árbol de decisiones es un árbol binario en el que los vértices internos contienen preguntas con dos respuestas posibles, las aristas se etiquetan con respuestas a las preguntas y los vértices terminales representan decisiones. Si comenzamos en la raíz, respondemos cada pregunta y seguimos las aristas adecuadas, en algún momento llegaremos a un vértice terminal que representa una decisión.

2. El tiempo en el peor caso de un algoritmo es proporcional a la altura del árbol de decisiones que representa el algoritmo.

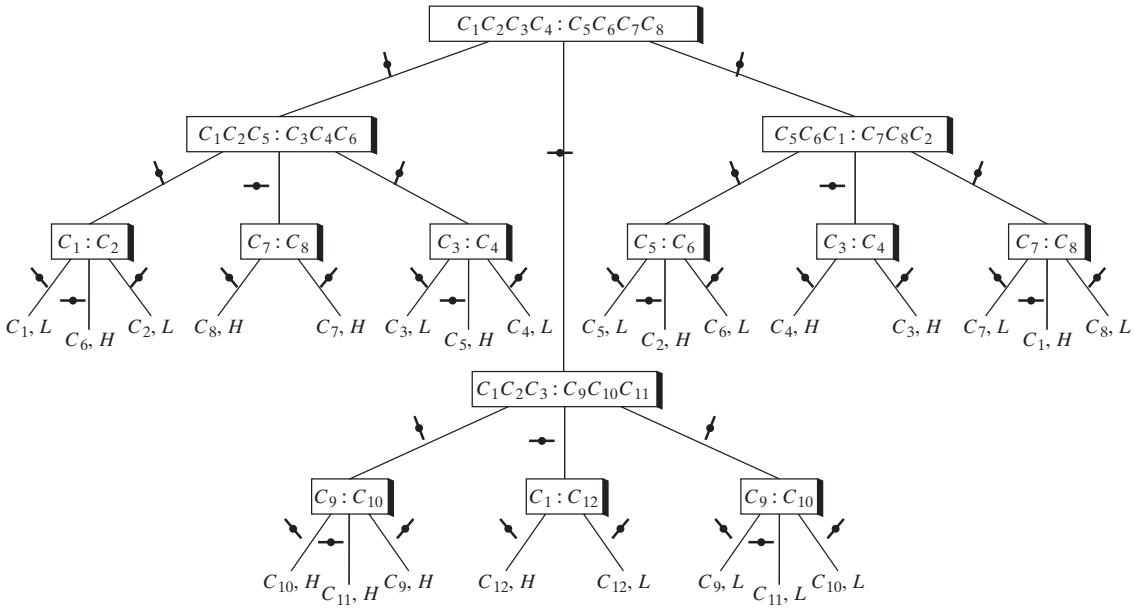
3. Un árbol de decisiones que representa un algoritmo para ordenar tiene $n!$ vértices terminales correspondientes a las $n!$ permutaciones posibles de una entrada de tamaño n. Si h es la altura del árbol, entonces se requieren h comparaciones en el peor caso. Como $\lg n! \leq h$ y $\lg n! = \Theta(n \lg n)$, en el peor caso, ordenar requiere al menos $\Omega(n \lg n)$ comparaciones.

Sección 9.7

1.



4.



7. Existen 28 resultados posibles para el juego de 14 monedas. Un árbol de altura 3 tiene cuando mucho 27 vértices terminales; entonces se requiere pesar al menos cuatro veces en el peor caso. De hecho, existe un algoritmo que usa cuatro pesadas en el peor caso: se comienza por pesar cuatro monedas contra otras cuatro. Si no se balancean, se procede como en la solución dada para el ejercicio 4 (para

el juego de 12 monedas). En este caso, cuando mucho se requiere pesar 3 veces. Si las monedas se balancean, se dejan de lado; el problema entonces es encontrar la moneda mala entre las seis restantes. El juego de seis monedas se resuelve pesando cuando mucho tres veces en el peor caso; si se considera la pesada inicial, se requiere pesar cuatro veces en el peor caso.

10. El análisis de árboles de decisiones muestra que, en el peor caso, se requieren al menos $\lceil \lg 5! \rceil = 7$ comparaciones para ordenar 5 elementos. El siguiente algoritmo ordena 5 elementos usando cuando mucho 7 comparaciones en el peor caso.

Dada la sucesión $a_1, \ldots, a_5$, primero se ordenan a_1, a_2 (una comparación) y después a_3, a_4 (una comparación). [Se supone ahora que $a_1 < a_2$ y $a_3 < a_4$]. Después se comparan a_2 y a_4. Suponga que $a_2 < a_4$. (El caso $a_2 > a_4$ es simétrico por lo que esa parte del algoritmo se omite). En este punto se sabe que

$$a_1 < a_2 < a_4 \quad \text{y} \quad a_3 < a_4 .$$

Después se determina si a_5 debe estar entre a_1, a_2 y a_4 comparando primero a_5 con a_2. Si $a_5 < a_2$, se compara a_5 con a_1; pero si $a_5 > a_2$, se compara a_5 con a_4. En cualquier caso se requieren dos comparaciones adicionales. En este punto, a_1, a_2, a_4, a_5 están ordenados. Por último se inserta a_3 en el lugar apropiado. Si primero se compara a_3 con el segundo elemento más pequeño entre a_1, a_2, a_4, a_5, sólo se requiere una comparación adicional, lo que da un total de 7 comparaciones. Para justificar esta última afirmación, se observa que los siguientes arreglos sean posibles después de insertar a_5 en su posición correcta:

$$a_5 < a_1 < a_2 < a_4$$
$$a_1 < a_5 < a_2 < a_4$$
$$a_1 < a_2 < a_5 < a_4$$
$$a_1 < a_2 < a_4 < a_5.$$

Si a_3 es menor que el segundo elemento, sólo se necesita una comparación más (con el primer elemento) para localizar la posición correcta de a_3. Si a_3 es mayor que el segundo elemento, cuando mucho se requiere una comparación adicional para localizar la posición correcta de a_3. En los tres primeros casos, sólo se debe comparar a_3 con a_2 o con a_5 para encontrar la posición correcta de a_3 puesto que ya se sabe que $a_3 < a_4$. En el cuarto caso, si a_3 es mayor que a_2, se sabe que va entre a_2 y a_4.

12. Se pueden considerar los números como concursantes y los vértices internos como ganadores donde el valor más grande gana.

15. Suponga que se tiene un algoritmo que encuentra el valor más grande entre $x_1, \ldots, x_n$. Sean $x_1, \ldots, x_n$ los vértices de una gráfica. Existe una arista entre x_i y x_j si el algoritmo compara x_i y x_j. La gráfica debe ser conexa. El número menor de aristas que se necesita para conectar n vértices $n - 1$.

18. Por el ejercicio 14, el orden de torneo requiere $2^k - 1$ comparaciones para encontrar el elemento más grande. Por el ejercicio 16, el orden de torneo requiere k comparaciones para encontrar el segundo elemento más grande. De manera similar, el orden de torneo requiere cuando mucho k comparaciones para encontrar el tercero más grande, cuando mucho k comparaciones para encontrar el cuarto más grande, y así sucesivamente. Entonces el número total de comparaciones es cuando mucho

$$\begin{aligned}[2^k - 1] + (2^k - 1)k &\le 2^k + k2^k \\ &\le k2^k + k2^k \\ &= 2 \cdot 2^k k = 2n \lg n.\end{aligned}$$

Sección 9.8 Repaso

1. Los árboles libres T_1 y T_2 son isomorfos si existe una función f uno a uno y sobre del conjunto de vértices de T_1 al conjunto de vértices de T_2, que satisface lo siguiente: Los vértices v_i y v_j son adyacentes en T_1 si y sólo si los vértices $f(v_i)$ y $f(v_j)$ son adyacentes en T_2.

2. Sea T_1 un árbol con raíz r_1 y sea T_2 un árbol con raíz r_2. Entonces T_1 y T_2 son isomorfos si existe una función f uno a uno y sobre del conjunto de vértices de T_1 al conjunto de vértices de T_2 que satisface lo siguiente:

a) v_i y v_j son adyacentes en T_1 si y sólo si $f(v_i)$ y $f(v_j)$ son adyacentes en T_2.

b) $f(r_1) = f(r_2)$.

3. Sea T_1 un árbol binario con raíz r_1 y sea T_2 un árbol binario con raíz r_2. Entonces T_1 y T_2 son isomorfos si existe una función f uno a uno y sobre del conjunto de vértices de T_1 al conjunto de vértices de T_2 que satisface lo siguiente:

a) v_i y v_j son adyacentes en T_1 si y sólo si $f(v_i)$ y $f(v_j)$ son adyacentes en T_2.

b) $f(r_1) = f(r_2)$.

c) v es un hijo izquierdo de w en T_1 si y sólo si $f(v)$ es un hijo izquierdo de $f(w)$ en T_2.

d) v es un hijo derecho de w en T_1 si y sólo si $f(v)$ es un hijo derecho de $f(w)$ en T_2.

4. $C(2n, n)/(n + 1)$

5. Dados los árboles binarios T_1 y T_2, primero se verifica si cualquiera de los dos está vacío (en cuyo caso se determina de inmediato si son o no isomorfos). Si ninguno de los dos está vacío, primero se verifica si los subárboles izquierdos son isomorfos y después si los subárboles derechos son isomorfos. T_1 y T_2 son isomorfos si y sólo si sus subárboles izquierdos y derechos son isomorfos.

Sección 9.8

1. Isomorfos. $f(v_1) = w_1$, $f(v_2) = w_5$, $f(v_3) = w_3$, $f(v_4) = w_4$, $f(v_5) = w_2$, $f(v_6) = w_6$.

4. No isomorfos. T_2 tiene una trayectoria simple de longitud 2 de un vértice de grado 1 a un vértice de grado 2, pero T_1 no.

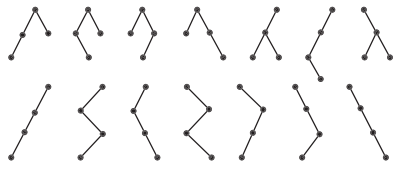
7. Isomorfos como árboles con raíz. $f(v_1) = w_1$, $f(v_2) = w_4$, . $f(v_3) = w_3$, $f(v_4) = w_2$, $f(v_5) = w_6$, $f(v_6) = w_5$, $f(v_7) = w_7$, $f(v_8) = w_8$. También son isomorfos como árboles libres.

10. No isomorfos como árboles binarios. La raíz de T_1 tiene un hijo izquierdo, pero la raíz de T_2 no. Isomorfos como árboles como raíz y como árboles libres.

13.

16.

19.



22. Sea b_n el número de árboles binarios completos de n vértices no isomorfos. Como todo árbol binario completo tiene un número impar de vértices, $b_n = 0$ si n es par. Se demuestra que si $n = 2i + 1$ es impar,

$$b_n = C_i,$$

donde C_i denota el i-ésimo número de Catalan.

La última ecuación se deduce del hecho de que existe una función uno a uno y sobre del conjunto de árboles binarios de i vértices al conjunto de árboles binarios completos de $(2i + 1)$ vértices. Esta función se construye como sigue. Dado un árbol binario de i vértices, en cada vértice terminal se agregan dos hijos. A cada vértice con un hijo se agrega otro hijo. Como el árbol que resulta tiene i vértices internos, hay $2i + 1$ vértices en total (teorema 9.5.4). El árbol construido es un árbol binario completo. Observe que ésta es una función uno a uno. Dado un árbol binario completo T' de $(2i + 1)$ vértices, si se eliminan todos los vértices terminales, se obtiene un árbol binario T de i vértices. La imagen de T es T'. Por lo tanto, la función es sobre.

25. Hay cuatro comparaciones en las líneas 1 y 3. Por el ejercicio 24, la llamada *árbol_bin_isom*(lc_r_1, lc_r_2) require $6(k - 1) + 2$ comparaciones. La llamada *árbol_bin_isom*(rc_r_1, rc_r_2) requiere cuatro comparaciones. Entonces el número total de comparaciones es

Sección 9.9 Repaso

1. En un árbol del juego, cada vértice muestra una posición específica en el juego. En particular, la raíz muestra la configuración inicial del juego. Los hijos de un vértice muestran todas las respuestas posibles para un jugador en la posición que indica el vértice.

2. En el procedimiento minimax, primero se asignan valores a los vértices terminales en el árbol del juego. Después, trabajando de abajo hacia arriba, el valor de un círculo se hace igual al mínimo de los valores de sus hijos y el valor de un cuadro es hace igual al máximo de los valores de sus hijos.

3. Una búsqueda que termina n niveles abajo del vértice dado.

4. Una función de evaluación asigna a cada posición posible del juego el valor de la posición para el primer jugador.

5. El recorte alfa-beta elimina (recorta) partes del árbol del juego y con ello omite la evaluación de algunas partes cuando se aplica el procedimiento minimax. El recorte alfa-beta funciona como sigue. Suponga que se sabe que el valor de un vértice de cuadro v es al menos x. Cuando un nieto w de v tiene el valor cuando mucho x, se elimina el subárbol cuya raíz es el padre de w. De manera similar, suponga que se sabe que el valor de un vértice de círculo v es cuando mucho x. Cuando un nieto w de v tiene un valor de al menos x, se elimina el subárbol cuya raíz es el padre de w.

6. Un valor alfa es una cota inferior para un vértice de cuadro.

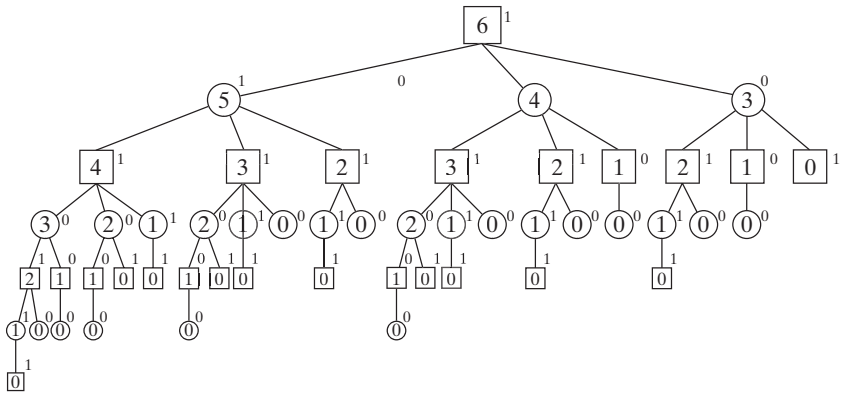
7. Un recorte alfa ocurre en un vértice de cuadro cuando un nieto w de v tiene un valor menor o igual que el valor alfa de v.

8. Un valor beta es una cota superior para un vértice de círculo.

9. Un recorte beta ocurre en un vértice de círculo cuando un nieto w de v tiene un valor mayor o igual que el valor beta de v.

Sección 9.9

1.



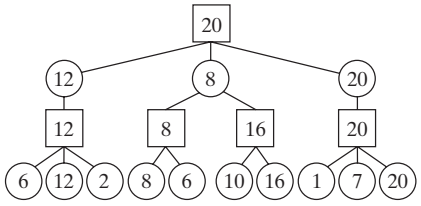
El primer jugador siempre gana. La estrategia ganadora consiste en tomar primero una ficha; después, sin importar qué haga el segundo jugador, se deja una ficha.

4. El segundo jugador siempre gana. Si quedan dos pilas, se dejan pilas con el mismo número de fichas. Si queda una pila, se toma toda.

7. Suponga que el primer jugador puede ganar en nim. El primer jugador siempre puede ganar en nim′ adoptando la siguiente estrategia: Jugar nim′ exactamente como nim a menos que la movida deje un número impar de pilas de una sola ficha y ninguna otra pila. En este caso, se deja un número par de pilas.

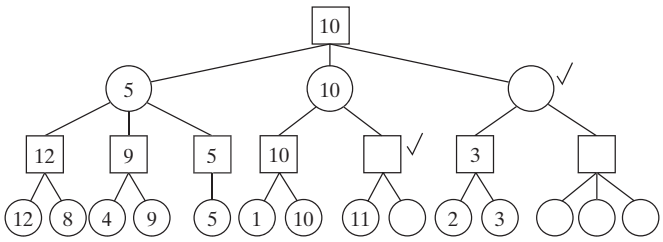
Suponga que el primer jugador siempre puede ganar en nim′. El primer jugador siempre puede ganar nim adoptando la siguiente estrategia: Jugar nim exactamente como nim′ a menos que la movida deje un número par de pilas de una sola ficha y ninguna otra pila. En este caso, se deja un número impar de pilas.

9.



12. El valor de la raíz es 3.

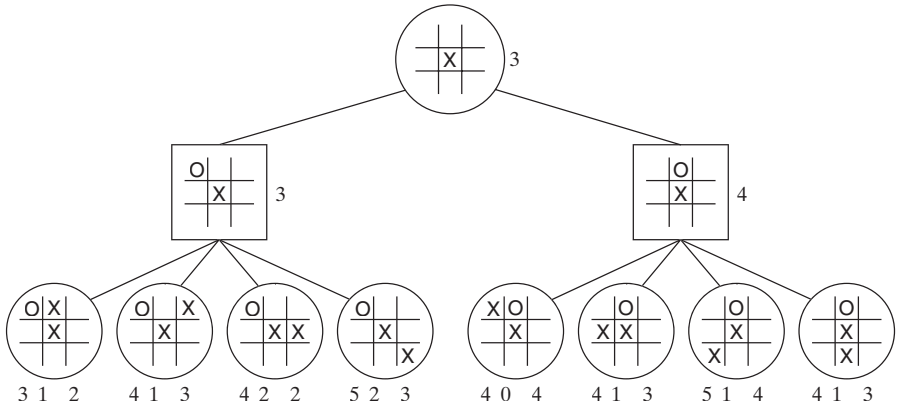
14. (Para el ejercicio 11)



15. $3 - 2 = 1$.

18. $4 - 1 = 3$.

19.



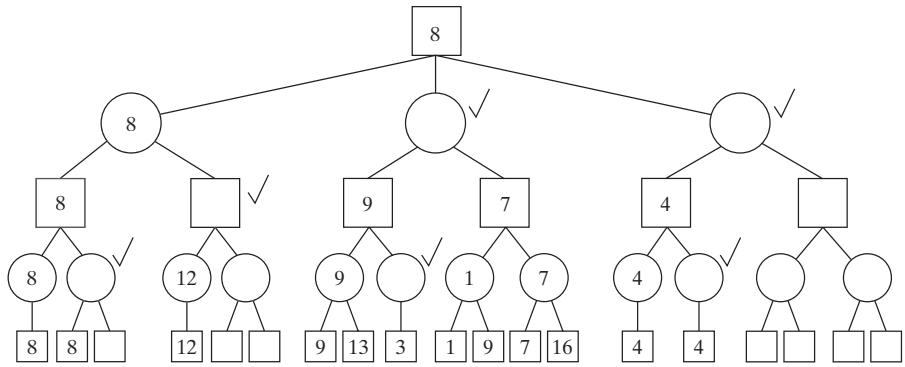
O jugará en una esquina.

22. Entrada: La raíz *PT* del árbol del juego; el tipo, *PT_tipo*, de *PT* (*cuadro* o *círculo*); el nivel *PT_nivel* de *PT*; el nivel máximo *n* al que se debe realizar la búsqueda; una función de evaluación *E*, y un número *val_ab* (el cual es el valor alfa o el valor beta del padre de *PT*). (La llamada inicial hace *val_ab* igual a ∞ si *PT* es un vértice de cuadro o igual a $-\infty$ si *PT* es un vértice de círculo).

Salida: El árbol del juego con la evaluación de *PT*

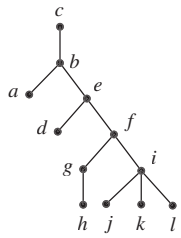
```
recorte_alfa_beta(PT, PT_tipo, PT_nivel, n, E, val_ab) {
    if (PT_nivel == n) {
        contenido(PT) = E(PT)
        return
    }
    if (PT_tipo == cuadro) {
        contenido(PT) = −∞
        para cada hijo C de PT {
            recorte_alfa_beta(C, círculo, PT_nivel + 1, n,
                E, contenido(PT))
            val_c = contenido(C)
            if (val_c ≥ val_ab) {
                contenido(PT) = val_ab
                return
            }
            if (val_c > contenido(PT))
                contenido(PT) = val_c
        }
    }
    else {
        contenido(PT) = ∞
        para cada hijo C de PT {
            recorte_alfa_beta(C, cuadro, PT_nivel + 1, n,
                E, contenido(PT))
            val_c = contenido(C)
            if (val_c ≤ val_ab) {
                contenido(PT) = val_ab
                return
            }
            if (val_c < contenido(PT))
                contenido(PT) = val_c
        }
    }
}
```

23. Primero se obtienen los valores 6, 6, 7 para los hijos de la raíz. Después se ordenan los hijos de la raíz con el hijo en la extrema derecha primero y se usa el procedimiento alfa-beta para obtener



Capítulo 9 Autoevaluación

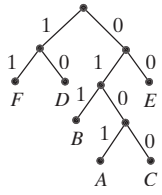
1.



2. *a*-2, *b*-1, *c*-0, *d*-3, *e*-2, *f*-3, *g*-4, *h*-5, *i*-4, *j*-5, *k*-5, *l*-5.

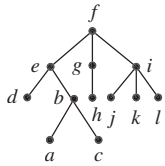
3. 5.

4.



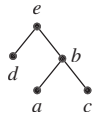
5. *a) b*

b) a, c



c) d, a, c, h, j, k, l

d)

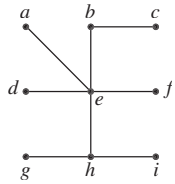


6. Verdadero. Vea el teorema 9.2.3.

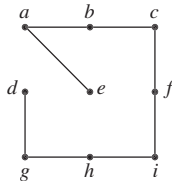
7. Verdadero. Un árbol de altura 6 o mayor debe tener siete vértices o más.

8. Falso.

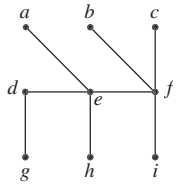
9.



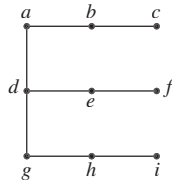
10.



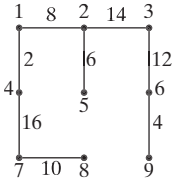
11.



12.



13.



14. (1, 4), (1, 2), (2, 5), (2, 3), (3, 6), (6, 9), (4, 7), (7, 8)

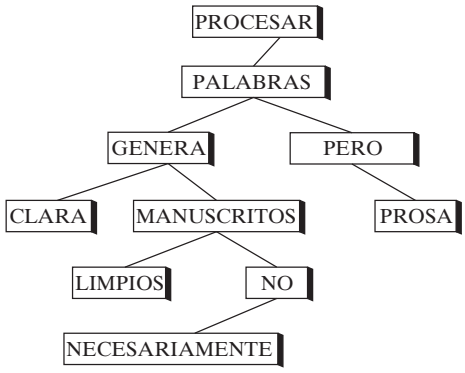
15. (6, 9), (3, 6), (2, 3), (2, 5), (1, 2), (1, 4), (4, 7), (7, 8)

16. Considere un "algoritmo de la ruta más corta" en el que cada paso se selecciona una arista disponible que tiene peso mínimo incidente en el vértice agregado más recientemente (vea la explicación que precede al teorema 9.4.5).

17.

18. 16.

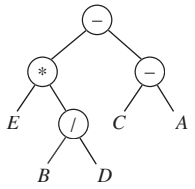
19.



20. Primero se compara MÁS con la palabra PROCESAR en la raíz. Como MÁS es menor que PROCESAR, se pasa al hijo izquierdo. Después se compara MÁS con PALABRAS. Como MÁS es menor que PALABRAS, se pasa al hijo izquierdo. Como MÁS es mayor que GENERA, se pasa el hijo derecho. Como MÁS es mayor que MANUSCRITOS, se pasa al hijo derecho. Como MÁS es menor que NO, se va al hijo izquierdo. Como MÁS es menor que NECESARIAMENTE, se intenta ir al hijo izquierdo. Como no hay hijo izquierdo, se concluye que MÁS no está en el árbol.

21. *ABFGCDE.* **22** *BGFAEDC.* **23.** *GFBEDCA.*

24.

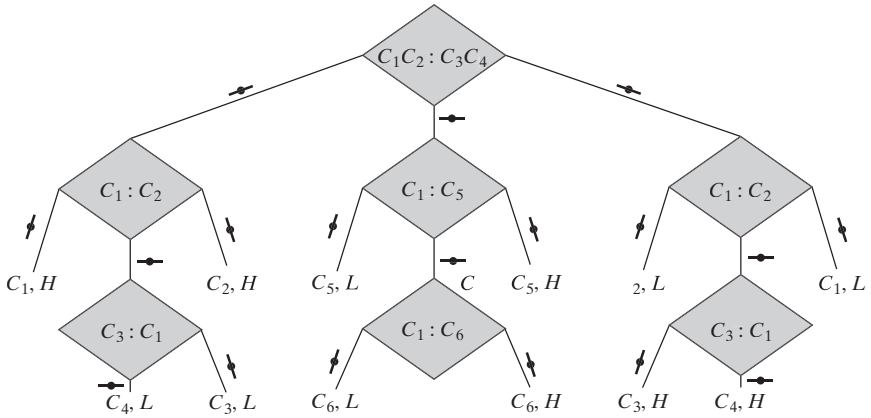


posfijo: $EBD/*CA--$

entrefijo con paréntesis: $((E*(B/D))-(C-A))$

25. Un algoritmo que requiere pesar dos veces se puede representar por un árbol de decisiones cuando mucho de altura 2. Sin embargo, este tipo de árbol tiene a lo sumo 9 vértices terminales. Como hay 12 resultados posibles, no existe tal algoritmo. Por lo tanto, se requiere pesar al menos 3 veces en el peor caso para identificar la moneda mala y determinar si es más o menos pesada.

26.



27. Según el teorema 9.7.3, cualquier algoritmo para ordenar requiere al menos $Cn \lg n$ comparaciones en el peor caso. Como el algoritmo del profesor Sabic usa cuando mucho $100n$ comparaciones, debe tenerse $Cn \lg n \leq 100$ para toda $n \geq 1$. Si se cancela n, se obtiene $C \lg n \leq 100n$ para toda $n \geq 1$, lo cual es falso. Por lo tanto, el profesor no tiene un algoritmo para ordenar que use cuando mucho $100n$ comparaciones en el peor caso para toda $n \geq 1$.

28. En el peor caso, se requieren 3 comparaciones para ordenar 3 artículos usando un orden óptimo (vea el ejemplo 9.7.2).

Si $n = 4$, el orden de inserción binaria ordena tres artículos (3 comparaciones en el peor caso) y luego inserta el cuarto en la lista de tres artículos ordenados (2 comparaciones en el peor caso), se realiza un total de 8 comparaciones en el peor caso.

Si $n = 5$, el orden de inserción binaria ordena cuatro artículos (5 comparaciones en el peor caso) y luego inserta el quinto en la lista de cuatro artículos ordenados (3 comparaciones en el peor caso), se realiza un total de 8 comparaciones en el peor caso.

Si $n = 6$, el orden de inserción binaria ordena cinco artículos (8 comparaciones en el peor caso) y luego inserta el sexto artículo en la lista de cinco artículos ordenados (3 comparaciones en el peor caso), se efectúa un total de 11 comparaciones en el peor caso.

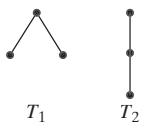
El análisis del árbol de decisiones muestra que cualquier algoritmo requiere al menos cinco comparaciones en el peor caso para ordenar cuatro artículos. Entonces el orden de inserción binaria es óptimo si $n = 4$.

El análisis del árbol de decisiones muestra que cualquier algoritmo requiere al menos siete comparaciones en el peor caso para ordenar cinco artículos. De hecho, es posible ordenar cinco artículos realizando siete comparaciones en el peor caso. Entonces el orden de inserción binaria no es óptimo si $n = 5$.

El análisis del árbol de decisiones muestra que cualquier algoritmo requiere al menos 10 comparaciones en el peor caso para ordenar seis artículos en el peor caso. De hecho, es posible ordenar seis artículos efectuando 10 comparaciones en el peor caso. Entonces el orden de inserción binaria no es óptimo si $n = 6$.

29. Verdadero. Si f es un isomorfismo de T_1 y T_2 como árboles con raíz, f también es un isomorfismo de T_1 y T_2 como árboles libres.

30. Falso.



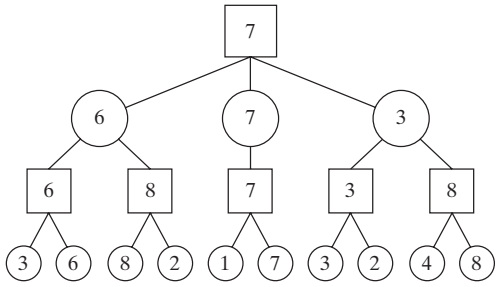
31. Isomorfismo. $f(v_1) = w_6$, $f(v_2) = w_2$, $f(v_3) = w_5$, $f(v_4) = .$ w_7, $f(v_5) = w_4$, $f(v_6) = w_1$, $f(v_7) = w_3$, $f(v_8) = w_8$.

32. No isomorfos. T_1 tiene un vértice (v_3) en el nivel 1 de grado 3, pero T_2 no.

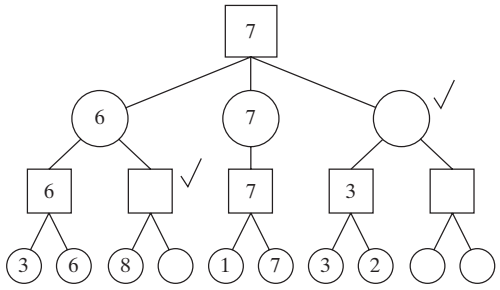
33. $3 - 1 = 2$

34. Cada renglón, columna o diagonal que contiene una *X* y dos blancos cuenta 1. Cada renglón, columna o diagonal que contiene dos *X* y un blanco cuenta 5. Cada renglón, columna o diagonal que contiene tres *X* cuenta 100. Cada renglón, columna o diagonal que contiene un 0 y dos blancos cuenta −1. Cada renglón, columna o diagonal que contiene dos ceros y un blanco cuenta −5. Cada renglón, columna o diagonal que contiene tres ceros cuenta −100. Se suman todos los valores obtenidos.

35.



36.



Sección 10.1 Repaso

1. Una red es una gráfica simple, ponderada y dirigida con un vértice designado que no tiene aristas que llegan, un vértice designado que no tiene aristas que salen y pesos no negativos.

2. Una fuente u origen es un vértice sin aristas que llegan.

3. Un sumidero o destino es un vértice sin aristas que salen.

4. La ponderación de una arista se llama su capacidad.

5. Un flujo asigna a cada arista un número no negativo que no excede a su capacidad, tal que para cada vértice v, que no sea el origen ni el destino, el flujo que entra a v es igual al flujo que sale de v.

6. El flujo en una arista es un número no negativo asignado como en el ejercicio 5.

7. Si F_{ij} es el flujo en la arista (i, j), el flujo que entra a j es $\sum_i F_{ij}$.

8. Si F_{ij} es el flujo en la arista (i, j) el flujo que sale del vértice i es $\sum_j F_{ij}$.

9. La conservación del flujo se refiere a la igualdad del flujo que entra y el que sale de un vértice.

10. Son iguales.

11. Si una red tiene múltiples orígenes, éstos pueden unirse en un solo vértice llamado superorigen.

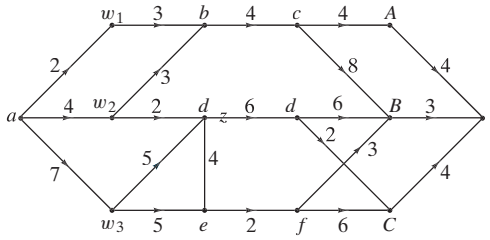
12. Si una red tiene destinos múltiples, éstos pueden unirse en un solo vértice llamado superdestino.

Sección 10.1

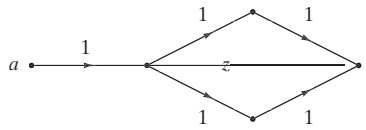
1. (b, c) es 6, 3; (a, d) es 4, 2; (c, e) es 6, 1; (c, z) es 5, 2. El valor del flujo es 5.

4. Se agregan las aristas (a, w_1), (a, w_2), (a, w_3), (A, z), (B, z), y (C, z) cada una con capacidad ∞.

7.



10.



Sección 10.2 Repaso

1. Un flujo máximo es un flujo con valor máximo.

2. Ignorando la dirección de las aristas, sea $P = (v_0, \ldots, v_n)$ una trayectoria del origen al destino. Si una arista en P está dirigida de v_{i-1} a v_i, se dice que tiene la orientación apropiada respecto a P.

3. Ignorando la dirección de las aristas, sea $P = (v_0, \ldots, v_n)$ una trayectoria del origen al destino. Si una arista en P está dirigida de v_i a v_{i-1}, se dice que tiene la orientación inapropiada respecto a P.

4. Se puede incrementar el flujo en una trayectoria cuando cada arista con la orientación apropiada no cubre su capacidad y cada arista con la orientación inapropiada tiene un flujo positivo.

5. Sea $\triangle$ el mínimo de los números $C_{ij} - F_{ij}$, para aristas (i, j) con la orientación apropiada en la trayectoria, y F_{ij} para las aristas (i, j) con la orientación inapropiada. Entonces el flujo se puede aumentar en $\triangle$ sumando $\triangle$ al flujo de cada arista bien orientada y restando $\triangle$ del flujo de cada arista mal orientada.

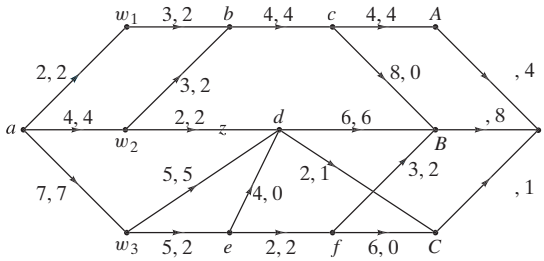
6. Comience con un flujo (es decir, asigne a cada arista un flujo cero). Busque una trayectoria como la descrita en el ejercicio 4. Aumente el flujo en esa trayectoria como se describe en el ejercicio 5.

Sección 10.2

1. 1

4. $(a, w_1) - 6, (a, w_2) - 0, (a, w_3) - 3, (w_1, b) - 6, (w_2, b) - 0,$ $(w_3, d) - 3, (d, c) - 3, (b, c) - 2, (b, A) - 4, (c, A) - 2, (c, B) - 3,$ $(A, z) - 6, (B, z) - 3$

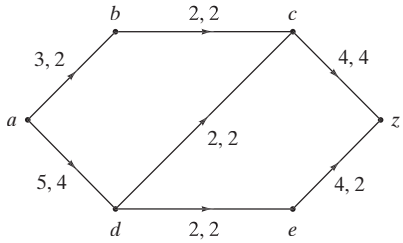
7.



10. $(a, A-7{:}00)-3000, (a, A-7{:}15)-3000, (a, A-7{:}30)-2000,$
$(A-7{:}00, B-7{:}30)-1000, (A-7{:}00, C-7{:}15)-2000,$
$(A-7{:}15, B-7{:}45)-1000, (A-7{:}15, C-7{:}30)-2000,$
$(A-7{:}30, C-7{:}45)-2000, (B-7{:}30, D-7{:}45)-1000,$
$(C-7{:}15, D-7{:}30)-2000, (B-7{:}45, D-8{:}00)-1000,$
$(C-7{:}30, D-7{:}45)-2000, (C-7{:}45, D-8{:}00)-2000,$
$(D-7{:}45, z)-3000, (D-7{:}30, z)-2000, (D-8{:}00, z)-3000.$

El resto de las aristas tiene un flujo igual a 0.

13.



16. El flujo máximo es 9.

19. Suponga que la suma de las capacidades de las aristas que inciden en a es U. Cada iteración del algoritmo 10.2.5 aumenta el flujo en 1. Como el flujo no puede exceder a U, en algún momento el algoritmo debe terminar.

Sección 10.3 Repaso

1. Un corte en una red consiste en un conjunto P de vértices y un complemento $\overline{P}$ de P, donde el origen está en P y el destino en $\overline{P}$.

2. La capacidad de un corte$(P\ \overline{P})$ es el número

$$\sum_{i\in P}\ \sum_{j\in\overline{P}} C_{ij}.$$

3. La capacidad de cualquier corte es mayor o igual que el valor de cualquier flujo.

4. Un corte mínimo es un corte que tiene capacidad mínima.

5. Si el valor de un flujo es igual a la capacidad de un corte, entonces el flujo es máximo y la corte es mínimo. El valor de un flujo F es igual a la capacidad de un corte $(P, \overline{P}$ si y sólo si $F_{ij} = C_{ij}$ para toda $i \in P, j \in \overline{P}$ y $F_{ij} = 0$ para toda $i \in \overline{P}, j \in P$.

6. Sea P el conjunto de vértices etiquetados, y sea $\overline{P}$.el conjunto de vértices no etiquetados al terminar el algoritmo 10.2.4. Se puede demostrar que las condiciones

- $F_{ij} = C_{ik}$ para toda $i \in P,\ j \in \overline{P}$
- $F_{ij} = 0$ para toda $i \in \overline{P},\ j \in P$

del ejercicio 5 se cumplen. Entonces el flujo es máximo.

Sección 10.3

1. 8; mínimo.

4. $P = \{a, b, d\}$.

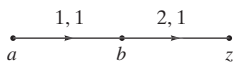
7. $P = \{a, d\}$.

10. $P = \{a, w_1, w_2, w_3, b, d, e\}$.

13. $P = \{a, w_1, w_2, w_3, b, c, d, d', e, f, A, B, C\}$.

16. $P = \{a, b, c, f, g, h, j, k, l, m\}$.

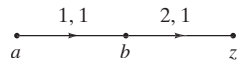
17.



con $C_{ab} = 1, C_{bz} = 2, m_{ab} = 1, m_{bz} = 2$.

20. Altere el algoritmo 10.2.4.

23. Falso. Considere el flujo



y el corte $P = \{a, b\}$.

Sección 10.4 Repaso

En la solución de los ejercicios 1 al 5, G es una gráfica dirigida bipartita con conjuntos ajenos de vértices V y W en los que las aristas están dirigidas de V a W.

1. Un acoplamiento para G es un conjunto de aristas sin vértices en común.

2. Un acoplamiento máximo para G es una asignación que contiene el número máximo de aristas.

3. Un acoplamiento completo para G es una asignación E que tiene la propiedad de que si $v \in V$, entonces $(v, w) \in E$ para alguna $w \in W$.

4. Se agrega un superorigen a y aristas de a a cada vértice en V. Se agrega un superdestino z y aristas de cada vértice en W a z. Se asigna una capacidad de 1 a todas las aristas. La red que se obtiene se llama red de acoplamiento. Entonces un flujo en la red de acoplamiento da una asignación por pares en G [v se asigna a w si y sólo si el flujo en la arista (v, w) es 1]; un flujo máximo corresponde a un acoplamiento máximo; y un flujo cuyo valor es $|V|$ corresponde a un acoplamiento completo.

5. Si $S \subseteq V$, sea

$$R(S) = \{w \in W \mid v \in S \text{ y } (v, w) \text{ es una arista en } G\}.$$

El teorema de Hall del matrimonio establece que existe un acoplamiento completo en G si y sólo si $|S| \leq |R(S)|$ para toda $S \subseteq V$.

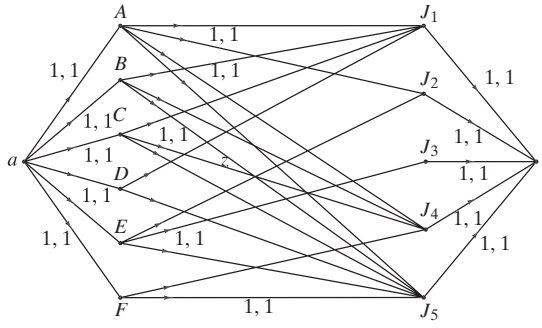
Sección 10.4

1. $P = \{a, A, B, D, J_2, J_5\}$.

3. Encontrar personas calificadas para las tareas.

6. Encontrar personas calificadas para todas las tareas.

9. Todas las aristas no etiquetadas son 1, 2. No hay un acoplamiento completo.



13. Cada renglón y columna tiene cuando mucho una etiqueta.

17. Si $\delta(G) = 0$, entonces $|S| - |R(S)| \leq 0$, para todo $S \subseteq V$. Por el teorema 10.4.7, G tiene un acoplamiento completo.

Si G tiene un acoplamiento completo, entonces $|S| - |R(S)| \leq 0$, para todo $S \subseteq V$, de manera que $\delta(G) \leq 0$. Si $S = \emptyset$, $|S| - |R(S)| = 0$, así, $\delta(G) = 0$.

Capítulo 10 Autoevaluación

1. En cada arista, el flujo es menor o igual a la capacidad y, excepto por el origen y el destino, el flujo que llega a cada vértice v es igual al flujo que sale de v.

2. 3.

3. 3.

4. 3.

5. (a, b, e, f, g, z).

6. Se cambian los flujos a $F_{a,b} = 2$, $F_{e,b} = 1$, $F_{e,f} = 1$, $F_{f,g} = 1$. $F_{g,z} = 1$

7. $F_{a,b} = 3$, $F_{b,c} = 3$, $F_{c,d} = 4$, $F_{d,z} = 4$, $F_{a,e} = 2$, $F_{e,f} = 2$, $F_{f,c} = 2$, $F_{f,g} = 1$, $F_{g,z} = 1$ y los flujos del resto de las aristas son cero.

8. $F_{a,b} = 0$, $F_{b,c} = 5$, $F_{c,d} = 5$, $F_{d,z} = 8$, $F_{e,b} = 3$, $F_{g,d} = 3$, $F_{a,e} = 8$, $F_{e,f} = 3$, $F_{f,g} = 3$, $F_{a,h} = 4$, $F_{e,i} = 2$, $F_{j,z} = 6$ $F_{h,i} = 4$, $F_{i,j} = 6$ y los flujos del resto de las aristas son cero.

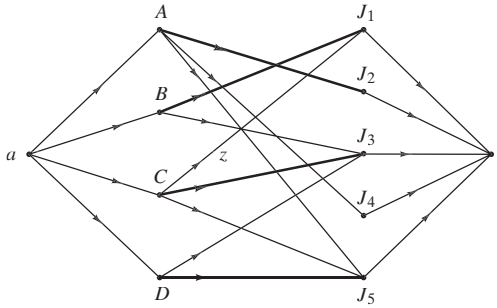
9. *a)* verdadero; *b)* falso; *c*) falso; *d*) verdadero.

10. 6.

11. No. La capacidad de $(P, \overline{P})$ es 6, pero la capacidad de $(P', \overline{P'})$, $P' = \{a, b, c, e, f\}$ es 5.

12. $P = \{a, b, c, e, f, g, h, i\}$.

13.



14. Vea la solución del ejercicio 13.

15. $A - J_2$, $B - J_1$, $C - J_3$, $D - J_5$ es un acoplamiento completo.

16. $P = \{a\}$

Sección 11.1 Repaso

1. Un circuito combinatorio es un circuito en el que la salida se define de manera única para cada combinación de entradas.

2. Un circuito secuencial es un circuito en el que la salida es una función de la entrada y el estado del sistema.

3. Una compuerta AND recibe una entrada x_1 y x_2, donde x_1 y x_2 son bits y produce una salida 1 si x_1 y x_2 son ambos 1, y 0 de otra manera.

4. Una compuerta OR recibe una entrada x_1 y x_2, donde x_1 y x_2 son bits, y produce una salida 0 si x_1 y x_2 son ambos 0, y 1 de otra manera.

5. Una compuerta NOT recibe una entrada x, donde x es un bit, y produce una salida 1 si x es 0, y 0 si x es 1.

6. Un inversor es una compuerta NOT.

7. Una tabla lógica de un circuito combinatorio lista todas las entradas posibles junto con las salidas que resultan.

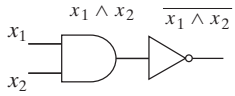
8. Las expresiones booleanas en los símbolos $x_1, \ldots, x_n$ se definen de manera recursiva como sigue. 0, 1, $x_1, \ldots, x_n$ son expresiones booleanas. Si X_1 y X_2 son expresiones booleanas, entonces (X_1), $\overline{X_1}$, $X_1 \vee X_2$ y $X_1 \wedge X_2$ son expresiones booleanas.

9. Una literal es el símbolo x o $\bar{x}$ que aparece en una expresión booleana.

Sección 11.1

1. $\overline{x_1 \wedge x_2}$

x_1	x_2	$\overline{x_1 \wedge x_2}$
1	1	0
1	0	1
0	1	1
0	0	1



4.

x_1	x_2	x_3	$((x_1 \wedge x_2) \quad \overline{(x_1 \wedge x_3)}) \wedge \overline{x}_3$
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	0
0	0	0	1

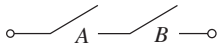
7. Si $x = 1$, la salida y no está determinada: suponga que $x = 1$ y $y = 0$. Entonces la entrada a la compuerta AND es 1, 0. Así, la salida de la compuerta AND es 0. Como entonces se aplica NOT, $y = 1$, lo que es una contradicción. De la misma manera, si $x = 1$ y $y = 1$, se obtiene una contradicción.

10. 0 **13.** 1

16. Es una expresión booleana. x_1, x_2 y x_3 son expresiones booleanas por (11.1.2). $x_2 \vee x_3$ es una expresión booleana por (11.2.3*c*). $(x_2 \vee x_3)$ es una expresión booleana por (11.1.3*a*). $x_1 \wedge (x_2 \vee x_3)$ es una expresión booleana por (11.1.3*d*).

19. No es una expresión booleana.

22.



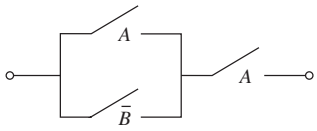
25. $(A \wedge B) \vee (C \wedge \overline{A})$

A	B	C	$(A \wedge B) \vee (C \wedge \overline{A})$
1	1	1	1
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	0

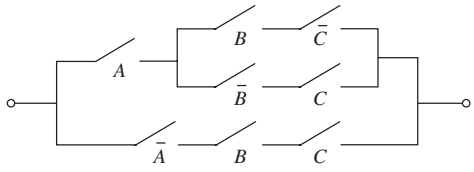
28. $(A \wedge (C \vee (D \wedge C))) \vee (B \wedge (D \vee (C \wedge A) \vee C))$

30.

A	B	$(A \vee \overline{B}) \wedge A$
1	1	1
1	0	1
0	1	0
0	0	0



33.



Sección 11.2 Repaso

1. $(a \vee b) \vee c = a \vee (b \vee c),\ (a \wedge b) \wedge c = a \wedge (b \wedge c)$
2. $a \vee b = b \vee a,\ a \wedge b = b \wedge a$
3. $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c),\ a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$
4. $a \vee 0 = a,\ a \wedge 1 = a$
5. $a \vee \overline{a} = 1,\ a \wedge \overline{a} = 0$
6. Las expresiones booleanas son iguales si tienen los mismos valores para todas las asignaciones posibles de bits a las literales.
7. Los circuitos combinatorios son equivalentes si, siempre que reciben las mismas entradas, producen las mismas salidas.
8. Sean C_1 y C_2 circuitos combinatorios representados, respectivamente, por las expresiones booleanas X_1 y X_2. Entonces C_1 y C_2 son equivalentes si y sólo si $X_1 = X_2$.

Sección 11.2

1.

x_1	x_2	$\overline{x_1 \wedge x_2}$	$\overline{x}_1 \vee \overline{x}_2$
1	1	0	0
1	0	1	1
0	1	1	1
0	0	1	1

4.

x_1	x_2	x_3	$\overline{x}_1 \vee (\overline{x}_2 \vee x_3)$	$\overline{(x_1 \wedge x_2)} \vee x_3$
1	1	1	1	1
1	1	0	0	0
1	0	1	1	1
1	0	0	1	1
0	1	1	1	1
0	1	0	1	1
0	0	1	1	1
0	0	0	1	1

6.

x_1	$x_1 \vee x_1$
1	1
0	0

9.

x_1	x_2	x_3	$x_1 \wedge \overline{(x_2 \wedge x_3)}$	$(x_1 \wedge \overline{x}_2) \vee (x_1 \wedge \overline{x}_3)$
1	1	1	0	0
1	1	0	1	1
1	0	1	1	1
1	0	0	1	1
0	1	1	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	0	0

11.

x	$\overline{\overline{x}}$
1	1
0	0

14. Falso. Tome $x_1 = 1, x_2 = 1, x_3 = 0$.

16.

a	b	c	$a \vee (b \wedge c)$	$(a \vee b) \wedge (a \vee c)$
1	1	1	1	1
1	1	0	1	1
1	0	1	1	1
1	0	0	1	1
0	1	1	1	1
0	1	0	0	0
0	0	1	0	0
0	0	0	0	0

18. Las expresiones booleanas que representan los circuitos son $(A \wedge \overline{B}) \vee \ (A \wedge C)$ y $A \wedge (\overline{B} \vee C)$. Las expresiones son iguales por el teorema 11.2*c*. Por lo tanto, los circuitos de conmutación son equivalentes.

21.

Sección 11.3 Repaso

1. Un álgebra booleana consiste en un conjunto S que contiene elementos distintos 0 y 1, operadores binarios $+$ y $\cdot$ y un operador unitario $'$ en S que satisface las leyes asociativa, conmutativa, distributiva, de identidad y de complementos.
2. $x + x = x, xx = x$
3. $x + 1 = 1, x0 = 0$
4. $x + xy = x, x(x + y) = x$
5. $(x')' = x$
6. $0' = 1, 1' = 0$
7. $(x + y)' = x'y', (xy)' = x' + y'$
8. El dual de una expresión booleana se obtiene sustituyendo 0 con 1, 1 con 0, $+$ con $\cdot$ y $\cdot$ con $+$.
9. El dual de un teorema referente a expresiones booleanas también es un teorema.

Sección 11.3

2. Se puede demostrar que las leyes asociativa y distributiva se cumplen directamente para el mcm y el mcd. Es claro que la ley conmutativa se cumple. Para ver que se cumple la ley de identidad observe que

$$\text{mcm}(x, 1) = x \quad \text{y} \quad \text{mcd}(x, 6) = x.$$

Como

$$\text{mcm}(x, 6/x) = 6 \quad \text{y} \quad \text{mcd}(x, 6/x) = 1,$$

las leyes de complementos se cumplen. Por lo tanto, $(S, +, \cdot, ', 1, 6)$ es un álgebra booleana.

4. Sólo se demuestra que

$$x \cdot (x + z) = (x \cdot y) + (x \cdot z) \text{ para toda } x, y, z \in S_n.$$

Ahora

$$x \cdot (y + z) = \quad = \text{mín}\{x, \text{máx}\{y, z\}\}$$
$$(x \cdot y) + (x \cdot z) = \quad = \text{máx}\{\text{mín}\{x, y\}, \text{mín}\{x, z\}\}$$

Se supone que $y \le z$. (El argumento es similar si $y > z$). Se consideran tres casos: $x < y$; $y \le x \le z$, y $z < x$.

Si $x < y$, se obtiene

$$\begin{aligned} x \cdot (y + z) &= \text{mín}\{x, \text{máx}\{y, z\}\} \\ &= \text{mín}\{x, z\} = x = \text{máx}\{x, x\} \\ &= \text{máx}\{\text{mín}\{x, y\}, \text{mín}\{x, z\}\} \\ &= (x \cdot y) + (x \cdot z). \end{aligned}$$

Si $y \le x \le z$, se obtiene

$$\begin{aligned} x \cdot (y + z) &= \text{mín}\{x, \text{máx}\{y, z\}\} \\ &= \text{mín}\{x, z\} = x = \text{máx}\{y, x\} \\ &= \text{máx}\{\text{mín}\{x, y\}, \text{mín}\{x, z\}\} \\ &= (x \cdot y) + (x \cdot z). \end{aligned}$$

Si $z < x$, se obtiene

$$\begin{aligned} x \cdot (y + z) &= \text{mín}\{x, \text{máx}\{y, z\}\} \\ &= \text{mín}\{x, z\} = z = \text{máx}\{y, z\} \\ &= \text{máx}\{\text{mín}\{x, y\}, \text{mín}\{x, z\}\} \\ &= (x \cdot y) + (x \cdot z). \end{aligned}$$

7. Si $X \cup Y = U$ y $X \cap Y = \emptyset$, entonces $Y = \overline{X}$

8. $xy + x0 = x(x + y)y$

11. $x + y' = 1$ si y sólo si $x + y = x$.

14. $x(x + y0) = x$

15. (Para el ejercicio 12)

$$\begin{aligned} 0 = x + y &= (x + x) + y \\ &= x + (x + y) = x + 0 = x \end{aligned}$$

De manera similar, $y = 0$.

18. [Para el inciso *c)*]

$$\begin{aligned} x(x + y) &= (x + 0)(x + y) \\ &= x + 0y = x + y0 = x + 0 = x. \end{aligned}$$

21. Primero demuestre que si $ba = ca$ y $ba' = ca'$, entonces $b = c$. Ahora tome $a = x, b = x + (y + z)$, y $c = (x + y) + z$ y use este resultado.

23. Si el primo p divide a n, p^2 no divide a n.

Sección 11.4 Repaso

1. El OR-exclusivo de x_1 y x_2 es 0 si $x_1 = x_2$, y 1 de otra manera.

2. Una función booleana es una función de la forma

$$f(x_1, \ldots, x_n) = X(x_1, \ldots, x_n),$$

donde X es un expresión booleana.

3. Un mintérmino es una expresión booleana de la forma

$$y_1 \wedge y_2 \wedge \cdots \wedge y_n,$$

donde cada y_i es ya sea x_1 o $\overline{x_1}$.

4. La forma disyuntiva normal de una función booleana f no idéntica a cero es

$$f(x_1, \ldots, x_n) = m_1 \vee m_2 \vee \cdots \vee m_k,$$

donde cada m_i es un mintérmino.

5. Sean $A_1, \ldots, A_k$ los elementos A_i de Z_2^n para los cuales $f(A_i) = 1$. Para cada $A_i = (a_1, \ldots, a_n)$, se hace $m_i = y_1 \wedge \cdots \wedge y_n$, donde $y_j = x_j$ si $a_j = 1$, y $\overline{x_j}$ si $a_j = 0$. Entonces

$$f(x_1, \ldots, x_n) = m_1 \vee m_2 \vee \cdots \vee m_k.$$

6. Un maxtérmino es una expresión booleana de la forma

$$y_1 \vee y_2 \vee \cdots \vee y_n,$$

donde cada y_i es ya sea x_j o $\overline{x_j}$.

7. La forma conjuntiva normal de una función booleana f no idéntica a 1 es

$$f(x_1, \ldots, x_n) = m_1 \wedge m_2 \wedge \cdots \wedge m_k,$$

donde cada m_i es un maxtérmino.

Sección 11.4

En estas sugerencias, $a \wedge b$ se escribe ab.

1. $xy \vee \overline{x}y \vee \overline{x}\,\overline{y}$

4. $xyz \vee xy\overline{z} \vee x\overline{y}\,\overline{z} \vee \overline{x}yz \vee \overline{x}y\overline{z}$

7. $xyz \vee x\overline{y}\,\overline{z} \vee \overline{x}\,\overline{y}\,\overline{z}$

10. $wx\overline{y}z \vee wx\overline{y}\,\overline{z} \vee w\overline{x}yz \vee w\overline{x}y\overline{z} \vee w\overline{x}\,\overline{y}\,\overline{z}$

11. $\vee\, \overline{w}xy\overline{z} \vee \overline{w}x\overline{y}z \vee \overline{w}x\overline{y}\,\overline{z} \vee \overline{w}\,\overline{x}y\overline{z} \vee \overline{w}\,\overline{x}\,\overline{y}\,\overline{z}$

14. $xy \vee x\overline{y}$

17. $xyz \vee \overline{x}yz \vee xy\overline{z} \vee \overline{x}y\overline{z}$

20. 0

22.

25. (Para el ejercicio 3)

$$(\overline{x} \vee y \vee \overline{z})(x \vee \overline{y} \vee \overline{z})(x \vee \overline{y} \vee z)$$

28. (Para el ejercicio 3)

$$(\overline{x} \vee \overline{y} \vee \overline{z})(\overline{x} \vee \overline{y} \vee z)(\overline{x} \vee y \vee z)(x \vee y \vee \overline{z})(x \vee y \vee z)$$

Sección 11.5 Repaso

1. Una compuerta es una función de Z_2^n en Z_2.

2. Un conjunto de compuertas G es funcionalmente completo si, dado cualquier entero positivo n y una función f de Z_2^n en Z_2, es posible construir una circuito combinatorio que calcule f usando sólo las compuertas en G.

3. {AND, OR, NOT}

4. Una compuerta NAND recibe una entrada x_1 y x_2, donde x_1 y x_2 son bits, y produce 0 si x_1 y x_2 son ambos 1, y 1 de otra manera.

5. Sí.

6. El problema de encontrar el mejor circuito.

7. Componentes pequeñas que por sí mismas son circuitos completos.

8. Vea la figura 11.5.8.

9. Vea la figura 11.5.9.

Sección 11.5

1. AND se puede expresar en términos de OR y NOT: $xy = \overline{\overline{x} \vee \overline{y}}$..

2. Un circuito combinatorio que consiste sólo de compuertas AND siempre da salida 0 cuando todas las entradas son 0.

5. Se usa inducción sobre n para demostrar que no existe un circuito combinatorio de n compuertas que contenga sólo compuertas AND y OR que calcule $f(x) = \overline{x}$.

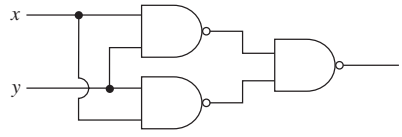
Si $n = 0$, la entrada x es igual a la salida x, de manera que es imposible que un circuito de compuerta 0 calcule f. Se prueba así el paso base.

Suponga que no existe un circuito combinatorio de n compuertas que tenga sólo compuertas AND y OR y que calcule f. Considere un circuito combinatorio de $(n + 1)$ compuertas con sólo compuertas AND y OR. La entrada x llega a una compuerta AND o a una OR. Suponga que x llega primero a una compuerta AND. (El argumento es similar si x llega primero a una compuerta OR y se omite). Como es un circuito combinatorio, la otra entrada a la compuerta AND es x mismo, la constante 1 o la constante 0. Si ambas entradas a la compuerta AND son x, entonces la salida de AND es igual a la entrada. En este caso, el comportamiento del circuito no cambia si se elimina la compuerta AND y se conecta x a lo que era la línea de salida de la compuerta AND. Pero ahora se tiene un circuito equivalente de n compuertas, mismo que, por la hipótesis inductiva, no calcula f. Entonces el circuito de $(n + 1)$ compuertas no puede calcular f.

Si la otra entrada a la compuerta AND es la constante 1, de nuevo la salida de la compuerta AND es igual a la entrada y se puede afirmar, como en el caso anterior, que el circuito de $(n + 1)$ compuertas no puede calcular f.

Si la otra entrada a la compuerta AND es la constante 0, la compuerta AND siempre produce 0 y, de esta manera, cambiar el valor de x no afecta la salida del circuito. En este caso, el circuito no puede calcular f. Esto completa el paso inductivo. Por lo tanto, ningún circuito combinatorio de n compuertas que tenga sólo compuertas AND y OR puede calcular $f(x) = \overline{x}$. Entonces {AND, OR} no es funcionalmente completa.

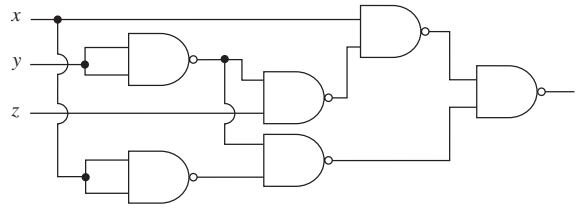
6.



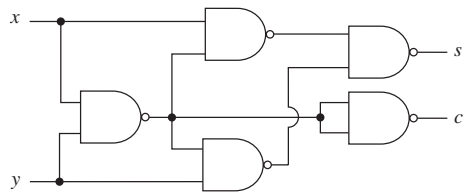
9. $y_1 = x_1x_2 \vee \overline{(x_2 \vee x_3)}$; $y_2 = \overline{x_2 \vee x_3}$

12. (Para el ejercicio 3) La forma disyuntiva normal se puede simplificar a $xy \vee x\overline{z} \vee \overline{x}\,\overline{y}$ y después reescribirla como

$x(y \vee \overline{z}) \vee \overline{x}\,\overline{y} = (x\overline{\overline{y}z}) \vee \overline{x}\,\overline{y} = \overline{\overline{x\overline{\overline{y}z}}\;\overline{\overline{x}\,\overline{y}}}$, que da el circuito



15.



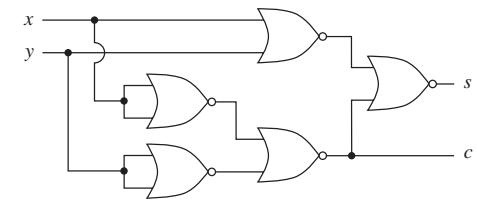
17. $xy = (x \downarrow x) \downarrow (y \downarrow y)$
$x \vee y = (x \downarrow y) \downarrow (x \downarrow y)\ \overline{x} = x \downarrow x$
$x \uparrow y = [(x \downarrow x) \downarrow (y \downarrow y)] \downarrow [(x \downarrow x) \downarrow (y \downarrow y)]$

20. Como

$$\overline{x} = x \downarrow x, \quad x \vee y = (x \downarrow y) \downarrow (x \downarrow y),$$

y {NOT, OR} es funcionalmente completa, {NOR} es funcionalmente completa.

23.



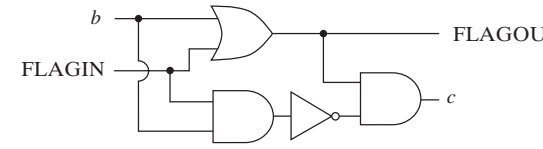
25. La tabla lógica es

x	y	z	*Salida*
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	0

27. La tabla lógica es

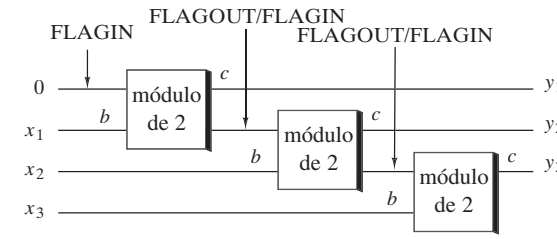
b	FLAGIN	c	FLAGOUT
1	1	0	1
1	0	1	1
0	1	1	1
0	0	0	0

Entonces $c = b \oplus$ FLAGIN y FLAGOUT $= b \vee$ FLAGIN. Se obtiene el circuito



28. 010100

31.



34. Al escribir las tablas de verdad se demuestra que

$$\overline{x} = x \to 0, \qquad x \vee y = (x \to 0) \to y.$$

Por lo tanto, una compuerta NOT se puede reemplazar por un compuerta $\to$ y una compuerta OR se puede reemplazar por dos compuertas $\to$. Como el conjunto {NOT, OR} es funcionalmente completo, se deduce que el conjunto $\{\to\}$ es funcionalmente completo.

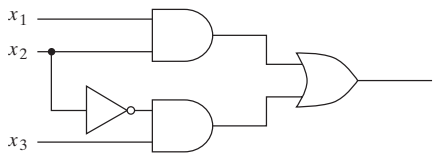
Capítulo 11 Autoevaluación

1.

x	y	z	$\overline{(x \wedge \overline{y})} \vee z$
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	0
0	1	1	1
0	1	0	1
0	0	1	1
0	0	0	1

2. 1.

3.



4. Suponga que x es 1. Entonces la entrada superior a la compuerta OR es 0. Si y es 1, entonces la entrada inferior a la compuerta OR es 0. Como ambas entradas a la compuerta OR son 0, la salida y de la compuerta OR es 0, lo cual es imposible. Si y es 0, entonces la entrada inferior a la compuerta OR es 1. Como la entrada a la compuerta OR es 1, la salida y de la compuerta OR es 1, lo cual es imposible. Por lo tanto, si la entrada al circuito es 1, la salida no está determinada de manera única. Entonces el circuito no es un circuito combinatorio.

5. Los circuitos son equivalentes. La tabla lógica para cualquiera de los circuitos es

x	y	*Salida*
1	1	0
1	0	1
0	1	0
0	0	0

6. Los circuitos no son equivalentes. Si $x = 0$, $y = 1$ y $z = 0$, la salida del circuito *a)* es 1, pero la salida del circuito *b)* es 0.

7. La ecuación es verdadera. La tabla lógica para cualquiera de las expresiones es

x	y	z	*Valor*
1	1	1	1
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	1
0	1	0	1
0	0	1	1
0	0	0	0

8. La ecuación es falsa. Si $x = 1$, $y = 0$ y $z = 1$, entonces

$$(x \wedge y \wedge z) \vee \overline{(x \vee z)} = 0,$$

pero

$$(x \wedge z) \vee (\overline{x} \wedge \overline{z}) = 1.$$

9. Leyes de acotación:

$$X \cup U = U, \quad X \cap \emptyset = \emptyset \text{ para toda } X \in S.$$

Leyes de absorción:

$$X \cup (X \cap Y) = X, \quad X \cap (X \cup Y) = X \text{ para toda } X, Y \in S.$$

10.
$$\begin{aligned}(x(x + y \cdot 0))' &= (x(x + 0))' && \text{(ley de acotación)}\\ &= (x \cdot x)' && \text{(ley de identidad)}\\ &= x' && \text{(ley de idempotencia)}\end{aligned}$$

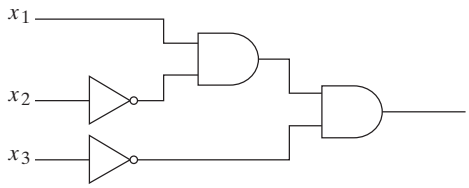
11. Dual: $(x + x(y + 1))' = x'$

$$\begin{aligned}(x + x(y + 1))' &= (x + x \cdot 1)' && \text{(ley de acotación)}\\ &= (x + x)' && \text{(ley de identidad)}\\ &= x' && \text{(ley de idempotencia)}\end{aligned}$$

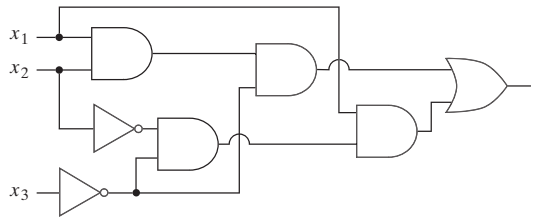
12. $\overline{\ }$ no es un operador unitario en S. Por ejemplo, $\overline{\{1, 2\}} \notin S$. .

En los ejercicios 13 al 16, $a \wedge b$ se escribe ab.

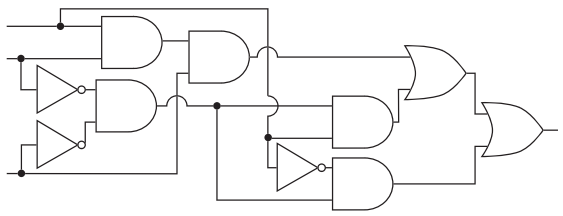
13. $x_1\overline{x}_2\overline{x}_3$



14. $x_1x_2\overline{x}_3 \vee x_1\overline{x}_2\overline{x}_3$



15. $x_1x_2x_3 \vee x_1\overline{x}_2\overline{x}_3 \vee \overline{x}_1\overline{x}_2\overline{x}_3$



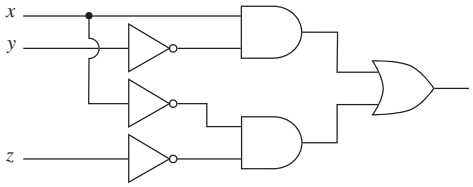
16. $x_1x_2\overline{x}_3 \vee x_1\overline{x}_2\overline{x}_3 \vee \overline{x}_1x_2x_3 \vee \overline{x}_1\overline{x}_2x_3$

17.

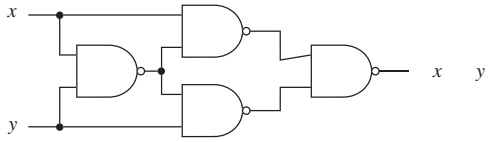
x	y	z	*Salida*
1	1	1	1
1	1	0	0
1	0	1	1
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	1
0	0	0	0

18. Forma disyuntiva normal: $x\overline{y}z \vee x\overline{y}\,\overline{z} \vee \overline{x}y\overline{z} \vee \overline{x}\,\overline{y}\,\overline{z}$

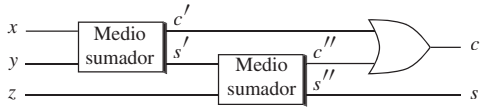
$$(x\overline{y}z \vee x\overline{y}\,\overline{z}) \vee \overline{x}y\overline{z} \vee \overline{x}\,\overline{y}\,\overline{z} = x\overline{y} \vee (\overline{x}y\overline{z} \vee \overline{x}\,\overline{y}\,\overline{z})$$
$$= x\overline{y} \vee \overline{x}\,\overline{z}$$



19.



20.



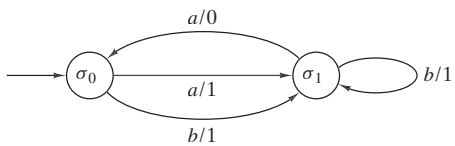
Sección 12.1 Repaso

1. Una unidad de retardo acepta como entrada un bit x_1 en el tiempo t y produce x_{i-1}, el bit recibido como entrada en el tiempo $t - 1$.
2. Un sumador serial alimenta dos números binarios y produce su suma.
3. Una máquina de estado finito consiste en un conjunto finito $\mathcal{I}$ de símbolos de entrada, un conjunto finito $\mathcal{O}$ de símbolos de salida, un conjunto finito $\mathcal{S}$ de estados, una función del siguiente estado f de $\mathcal{S} \times \mathcal{I}$ en $\mathcal{S}$, una función de salida g de $\mathcal{S} \times \mathcal{I}$ en $\mathcal{O}$, y un estado inicial $\sigma \in \mathcal{S}$.
4. Sea $M = (\mathcal{I}, \mathcal{O}, \mathcal{S}, f, g, \sigma)$ una máquina de estado finito. El diagrama de transición de M es una digráfica G cuyos vértices son los estados. Una flecha designa el estado inicial. Una arista dirigida (σ_1, σ_2) existe en G si hay una entrada i con $f(\sigma_1, i) = \sigma_2$. En este caso, si $g(\sigma_1, i) = o$, la arista (σ_1, σ_2) se etiqueta i/o.
5. El flip-flop *SR* se define por la tabla

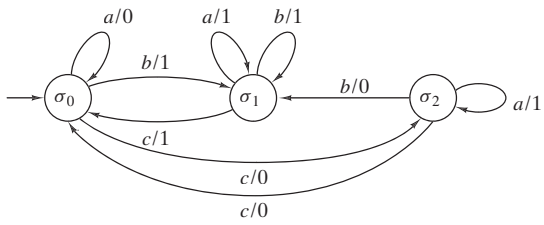
S	R	Q
1	1	No permitido
1	0	1
0	1	0
0	0	1 si S era 1 la última vez 0 si R era 1 la última vez

Sección 12.1

1.



4.



6. $\mathcal{I} = \{a, b\}$; $\mathcal{O} = \{0, 1\}$; $\mathcal{S} = \{\sigma_0, \sigma_1\}$; estado inicial $= \sigma_0$

$\mathcal{S}$ \ $\mathcal{I}$	a	b	a	b
σ_0	σ_1	σ_0	0	1
σ_1	σ_1	σ_1	1	1

9. $\mathcal{I} = \{a, b\}$; $\mathcal{O} = \{0, 1\}$; $\mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}$; estado inicial $= \sigma_0$

$\mathcal{S}$ \ $\mathcal{I}$	a	b	a	b
σ_0	σ_1	σ_2	0	0
σ_1	σ_0	σ_2	1	0
σ_2	σ_3	σ_0	0	1
σ_3	σ_1	σ_3	0	0

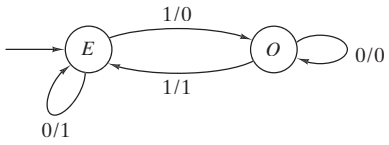
11. 1110.

14. 001110.

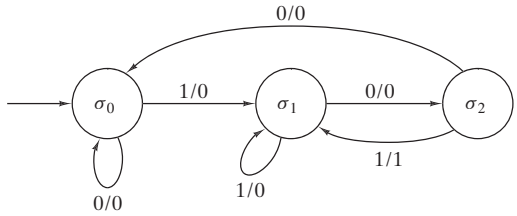
17. 001110001.

20. 020022201020.

21.



24.



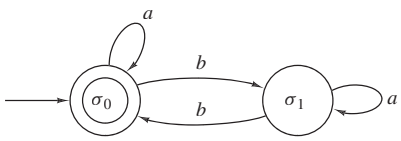
27. Cuando γ se alimenta, la máquina produce x_n, x_{n-1}, . . . hasta que $x_i = 1$. Por lo tanto, produce $\overline{x}_i$. Sin embargo, según el algoritmo 11.5.16, esto es el complemento a 2 de α.

Sección 12.2 Repaso

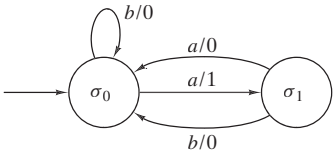
1. Un autómata de estado finito consiste en un conjunto finito $\mathcal{I}$ de símbolos de entrada, un conjunto finito $\mathcal{S}$ de estados, una función f del siguiente estado de $\mathcal{S} \times \mathcal{I}$ en $\mathcal{S}$, un subconjunto $\mathcal{A}$ de $\mathcal{S}$ de estados aceptantes y un estado inicial $\sigma \in \mathcal{S}$.
2. Un autómata de estado finito A acepta una cadena si, cuando se alimenta la cadena en A, el último estado alcanzado es un estado aceptante.
3. Los autómatas de estado finito son equivalentes si aceptan precisamente las mismas cadenas.

Sección 12.2

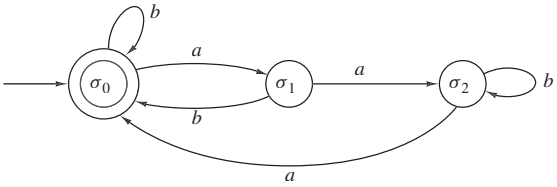
1. Toda las aristas que llegan a σ_0 producen 1 y todas las aristas que llegan a σ_1 producen 0; entonces, la máquina de estado finito es un autómata de estado finito.



4.



7.



10. (Para el ejercicio 1) $\mathcal{I} = \{a, b\}$; $\mathcal{S} = \{\sigma_0, \sigma_1\}$; $\mathcal{A} = \{\sigma_0\}$; ; estado inicial $= \sigma_0$

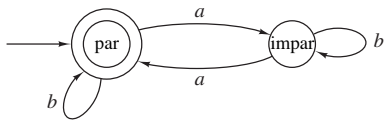
$\mathcal{S}$ \ $\mathcal{I}$	a	b
σ_0	σ_0	σ_1
σ_1	σ_1	σ_0

13. Aceptado.

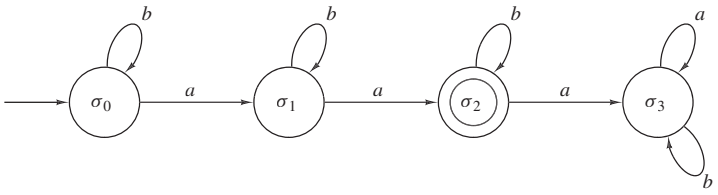
16. Aceptado.

18. No importa en qué estado estemos, después de una *a* nos movemos a un estado aceptante; sin embargo, después de una *b* nos movemos a un estado no aceptante.

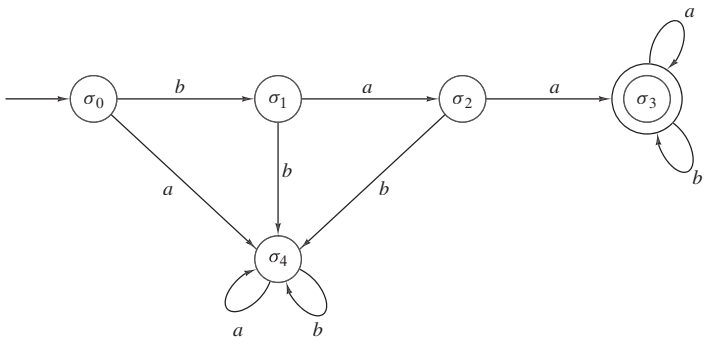
21.



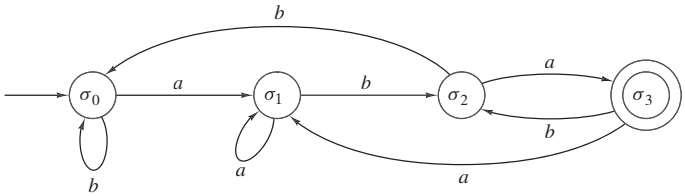
24.



27.



30.



32. (Para el ejercicio 1) Este algoritmo determina si una cadena en {*a, b*} es aceptada por el autómata de estado finito cuyo diagrama de transición está dado en el ejercicio 1.

Entrada: *n*, la longitud de la cadena ($n = 0$ es la cadena nula); $s_1, \ldots, s_n$, la cadena.

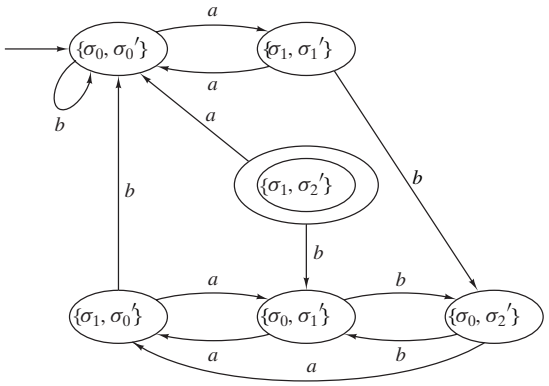
Salida: "Aceptado" si la cadena se acepta
"Rechazado" si la cadena no se acepta.

```
ex32(s, n) {
    estado = 'σ0'
    for i = 1 to n {
        if(estado == 'σ0' ∧ s_i == 'b')
            estado = 'σ1'
        if(estado == 'σ1' ∧ s_i == 'b')
            estado = 'σ0'
    }
```

```
    if (estado == 'σ0')
        return "Aceptado"
    else
        return "Rechazado"
}
```

35. Cada estado aceptante se hace no aceptante y cada estado no aceptante se hace aceptante.

38. A partir de la construcción dada en los ejercicios 36 y 37 se obtiene el siguiente autómata de estado finito que acepta $L_1 \cap L_2$. (Se designan los estados en el ejercicio 5 con primas).



El autómata de estado finito que acepta $L_1 \cup L_2$ es el mismo autómata de estado finito que acepta $L_1 \cap L_2$, excepto que el conjunto de estado aceptante es

$$\{(\sigma_1, \sigma_0'), \quad (\sigma_1, \sigma_1'), \quad (\sigma_1, \sigma_2'), \quad (\sigma_0, \sigma_2')\}.$$

41. Emplee la construcción de los ejercicios 36 y 37.

Sección 12.3 Repaso

1. Un "lenguaje natural" se refiere a las palabras y combinaciones de palabras comunes escritas y habladas. Un "lenguaje formal" es un lenguaje artificial que consiste en un conjunto especificado de cadenas. Los lenguajes formales se usan para modelar los lenguajes naturales y para la comunicación con las computadoras.

2. Una gramática de estructura de frases consiste en un conjunto finito N de símbolos no terminales, un conjunto finito T de símbolos terminales donde $N \cap T = \emptyset$, un subconjunto finito de $[(N \cup T)^* - T^*] \times (N \cup T)^*$ llamado conjunto de producciones y un símbolo de inicio en N.

3. Si $\alpha \rightarrow \beta$ es una producción y $x\,\alpha\,y \in (N \cup T)^*$, se dice que $x\,\beta\,y$ se puede derivar directamente de $x\,\alpha\,y$.

4. Si $\alpha_i \in (N \cup T)^*$ para $i = 1, \ldots, n$ y α_{i+1} se deriva directamente de α_i para $i = 1, \ldots, n-1$, se dice que α_n se deriva directamente de α_1 y se escribe $\alpha_1 \Rightarrow \alpha_n$.

5. $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_n$ se llama una derivación de α_n a partir de α_1.

6. El lenguaje generado por una gramática consiste en todas las cadenas en terminales que se derivan del símbolo de inicio.

7. La forma normal de Backus (BNF) es una manera de escribir las producciones de una gramática. En la BNF los símbolos no terminales suelen comenzar con "(" y terminar con ")". Además, la flecha $\rightarrow$ se sustituye con ::=. Las producciones con el mismo lado izquierdo se combinan usando la barra "|". Un ejemplo es

(entero con signo) ::=
+(enero sin signo) | − (entero sin signo)

8. En una gramática sensible al contexto, toda producción es de la forma $\alpha A \beta \rightarrow \alpha \delta \beta$, , donde $\alpha, \beta \in (N \cup T)^*$, $A \in N$, , y $\delta \in (N \cup T)^* - \{\lambda\}$.

9. En una gramática libre de contexto, toda producción es de la forma $A \rightarrow \delta$, donde $A \in N$ y $\delta \in (N \cup T)^*$.

10. En una gramática normal, toda producción es de la forma $A \rightarrow a$, $A \rightarrow aB$ o $A \rightarrow \lambda$, donde $A, B \in N$ y $a \in T$.

11. Una gramática sensible al contexto.

12. Una gramática libre de contexto.

13. Una gramática normal.

14. Un lenguaje es sensible al contexto si existe una gramática sensible al contexto que lo genere.

15. Un lenguaje es libre de contexto si existe una gramática libre de contexto que lo genere.

16. Un lenguaje es normal si existe una gramática normal que lo genere.

17. Una gramática de Lindenmayer interactiva, libre de contexto consiste en un conjunto finito N de símbolos no terminales; un conjunto finito T de símbolos terminales donde $N \cap T = \emptyset$; un conjunto finito de producciones $A \rightarrow B$, donde $A \in N \cup T$ y $B \in (N \cup T)^*$; y un símbolo de inicio en N.

18. El copo de nieve de von Koch se genera mediante una gramática de Lindenmayer interactiva libre de contexto.

$$N = \{D\}$$
$$T = \{d, +, -\}$$
$$P = \{D \rightarrow D - D + + D - D, D \rightarrow d, + \rightarrow +, - \rightarrow -\}.$$

d significa "dibuja una línea recta en la dirección actual", $+$ significa "gira 60° a la derecha" y $-$ quiere decir "gira 60° a la izquierda".

19. Las curvas fractales se caracterizan porque una parte de la curva tiene la misma forma que toda la curva.

Sección 12.3

1. Normal, libre de contexto, sensible al contexto.

4. Libre de contexto, sensible al contexto.

7. $\sigma \Rightarrow b\sigma \Rightarrow bb\sigma \Rightarrow bbaA \Rightarrow bbabA \Rightarrow bbabbA \Rightarrow bbabbab\sigma \Rightarrow bbabbab$.

10. $\sigma \Rightarrow ABA \Rightarrow ABBA \Rightarrow ABBAA$
$\Rightarrow ABBaAA \Rightarrow abBBaAA \Rightarrow abbBaAA$
$\Rightarrow abbbaAA \Rightarrow abbbaabA \Rightarrow abbbaabab$.

12. (Para el ejercicio 1)

$$<\sigma> ::= b<\sigma> \mid a<A> \mid b$$
$$<A> ::= a<\sigma> \mid b<A> \mid a$$

15. $S \rightarrow aA$, $A \rightarrow aA$, $A \rightarrow bA$, $A \rightarrow a$, $A \rightarrow b$, $S \rightarrow a$.

18. $S \rightarrow aA$, $S \rightarrow bS$, $S \rightarrow \lambda$, $A \rightarrow aA$, $A \rightarrow bB$, $A \rightarrow \lambda$, $B \rightarrow aA$, $B \rightarrow bS$.

21. <número exp> ::= <entero> E <entero> |
<numero flotante> |
<número flotante> E <entero>.

24. $S \rightarrow aSa$, $S \rightarrow bSb$, $S \rightarrow a$, $S \rightarrow b$, $S \rightarrow \lambda$.

25. Si una derivación comienza $S \Rightarrow aSb$, la cadena que resulta comienza con a y termina con b. De manera similar, si la derivación

comienza $S \Rightarrow bSa$, la cadena que se obtiene comienza con b y termina en a. Por lo tanto, la gramática no genera la cadena $abba$.

28. Si una derivación comienza con $S \Rightarrow abS$, la cadena resultante comienza con ab. Si una derivación comienza con $S \Rightarrow baS$, la cadena que resulta comienza con ba. Si una derivación comienza $S \Rightarrow aSb$, la cadena que resulta comienza con a y termina con b. Si una derivación comienza $S \Rightarrow bSa$, la cadena que resulta comienza con b i termina con a. Por lo tanto, la gramática no genera la cadena $aabbabba$.

31. La gramática no genera L, el conjunto de todas las cadenas en $\{a, b\}$ con el mismo número de símbolos a y b.

Cualquier cadena generada por la gramática tiene igual número de letras a y b ya siempre que se use cualquiera de las producciones en una derivación, se agrega el mismo número de símbolos a y b a la cadena.

Para probar el recíproco se considera una cadena arbitraria α en L, y se usa inducción sobre la longitud $|\alpha|$ de α para demostrar que α fue generada por la gramática. El paso base es $|\alpha| = 0$. En este caso, α es la cadena nula y $S \Rightarrow \lambda$ es una derivación de α.

Sea α una cadena no nula y suponga que cualquier cadena en L cuya longitud es menor que $|\alpha|$ está generada por la gramática. Primero se considera el caso de que α comience con a. Entonces α se puede escribir $\alpha = a\alpha_1 b\alpha_2$ donde α_1 y α_2 tienen igual número de letras a y b. Por la hipótesis inductiva, existen derivaciones $S \Rightarrow \alpha_1$ y $S \Rightarrow \alpha_2$ de α_1 y α_2. Pero ahora

$$S \Rightarrow aSbS \Rightarrow a\alpha_1 b\alpha_2$$

es una derivación de α. De manera similar, si α comienza con b, existe una derivación de α. Con esto termina el paso inductivo, y la prueba queda completa.

32. Sustituya cada producción

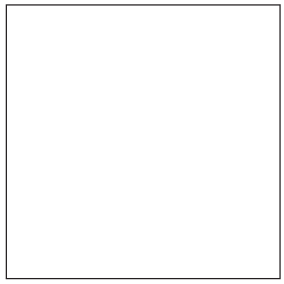
$$A \rightarrow x_1 \cdots x_n B,$$

donde $n > 1$, $x_j \in T$ y $B \in N$, con las producciones

$$\begin{aligned} A &\rightarrow x_1 A_1 \\ A_1 &\rightarrow x_2 A_2 \\ &\vdots \\ A_{n-1} &\rightarrow x_n B, \end{aligned}$$

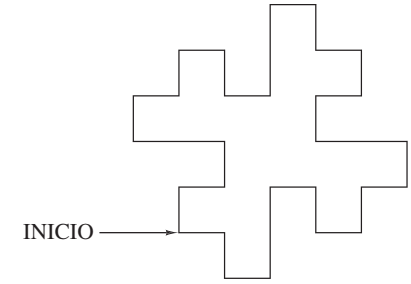
donde $A_1, \ldots, A_{n-1}$ son símbolos no terminales adicionales.

35. $S \Rightarrow D + D + D + D \Rightarrow d + d + d + d$



$$\begin{aligned} S &\Rightarrow D + D + D + D \\ &\Rightarrow D + D - D - DD + D + D - D \\ &\quad + D + D - D - DD + D + D - D \\ &\quad + D + D - D - DD + D + D - D \\ &\quad + D + D - D - DD + D + D - D \\ &\Rightarrow d + d - d - dd + d + d - d \end{aligned}$$

$$\begin{aligned} &+ d + d - d - dd + d + d - d \\ &+ d + d - d - dd + d + d - d \\ &+ d + d - d - dd + d + d - d \end{aligned}$$



Sección 12.4 Repaso

1. Sea σ el estado inicial, T el conjunto de símbolos de entrada y N el conjunto de estados. Sea P el conjunto de producciones $S \rightarrow xS'$, si hay una arista etiquetada x de S a S', y $S \rightarrow \lambda$ si S es un estado aceptante. Sea G la gramática normal (N, T, P, σ). Entonces el conjunto de cadenas aceptadas por A es igual a $L(G)$.

2. Un autómata de estado finito no determinístico consiste en un conjunto finito $\mathcal{I}$ de símbolos de entrada, un conjunto finito $\mathcal{S}$ de estados, una función f del siguiente estado de $\mathcal{S} \times \mathcal{I}$ en $\mathcal{P}(\mathcal{S})$, un subconjunto $\mathcal{A}$ de $\mathcal{S}$ de estados aceptantes y un estado inicial $\sigma \in \mathcal{S}$.

3. Una cadena α es aceptada por un autómata de estado finito no determinístico A si existe alguna trayectoria que represente a α en el diagrama de transición de A que comience en el estado inicial y termine en el estado aceptante.

4. Los autómatas de estado finito no determinísticos son equivalentes si aceptan precisamente las mismas cadenas.

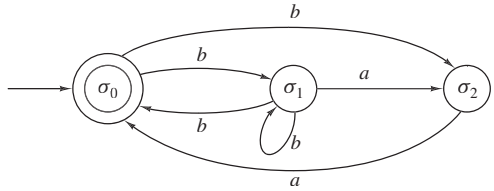
5. Sea $G = (\mathrm{N}, \mathrm{T}, \mathrm{P}, \sigma)$ una gramática normal. El autómata de estado finito A se construye como sigue. El conjunto de símbolos de entrada es T. El conjunto de estados es N junto con un estado adicional $F \notin N \cup T$. La función f del siguiente estado se define como

$$f(S, x) = \{S' \mid S \rightarrow xS' \in P\} \cup \{F \mid S \rightarrow x \in P\}.$$

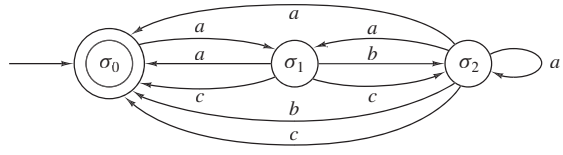
El conjunto de estados aceptantes es F junto con todo S para el cual $S \rightarrow \lambda$ es una producción. Entonces A acepta precisamente las cadenas $L(G)$.

Sección 12.4

1.



4.



6. $\mathcal{I} = \{a, b\}$; $\mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2\}$; $\mathcal{A} = \{\sigma_1, \sigma_2\}$; estado inicial $= \sigma_0$

$\mathcal{S}$ \ $\mathcal{I}$	a	b
σ_0	$\{\sigma_1, \sigma_2\}$	
σ_1	$\{\sigma_1\}$	$\{\sigma_0, \sigma_2\}$
σ_2	$\emptyset$	$\emptyset$

9. $\mathcal{I} = \{a, b\}$; $\mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}$; $\mathcal{A} = \{\sigma_3\}$; estado inicial $= \sigma_0$

$\mathcal{S}$ \ $\mathcal{I}$	a	b
0	$\{\sigma_0\}$	$\{\sigma_0, \sigma_1\}$
1	$\{\sigma_2\}$	
2		$\{\sigma_3\}$
3	$\{\sigma_3\}$	$\{\sigma_3\}$

11. (Para el ejercicio 5) $N = \{\sigma_0, \sigma_1, \sigma_2\}$, $T = \{a, b\}$,

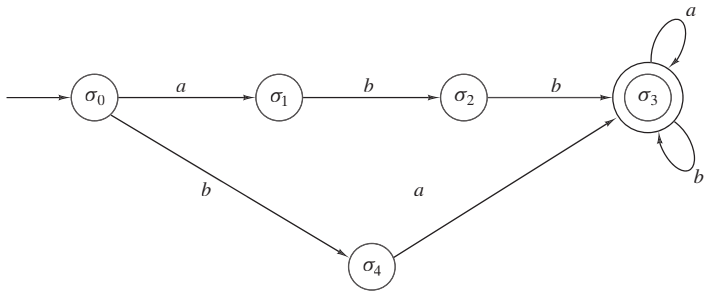
$$\sigma_0 \to a\sigma_1, \quad \sigma_0 \to b\sigma_0, \quad \sigma_1 \to a\sigma_0, \quad \sigma_1 \to b\sigma_2,$$

$$\sigma_2 \to b\sigma_1, \quad \sigma_2 \to a\sigma_0, \quad \sigma_2 \to \lambda.$$

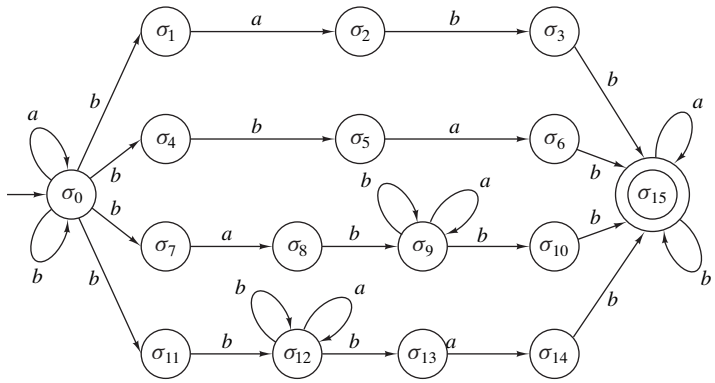
14. No. Para los primeros tres caracteres, *bba*, los movimientos están determinados y se termina en *C*. A partir de *C*, ninguna arista contiene una *a*; por lo tanto, *bbabab* no se acepta.

17. Sí. La trayectoria $(\sigma, \sigma, \sigma, \sigma, C, C)$, que representa la cadena *aaaab*, termina en *C*, que es un estado aceptante.

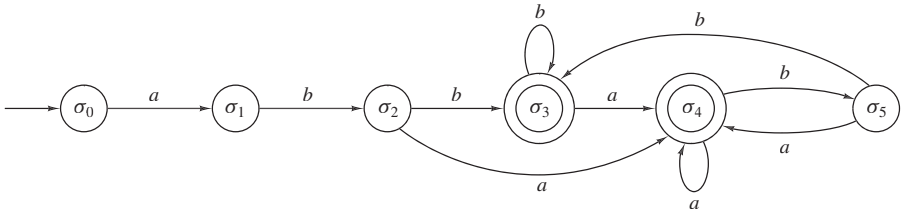
21.



24.



27.



30. (Para el ejercicio 21) $\sigma_0 \to a\sigma_1, \sigma_0 \to b\sigma_4, \sigma_1 \to b\sigma_2$, $\sigma_2 \to b\sigma_3$, $\sigma_3 \to a\sigma_3, \sigma_3 \to b\sigma_3, \sigma_4 \to a\sigma_3, \sigma_3 \to \lambda$

Sección 12.5 Repaso

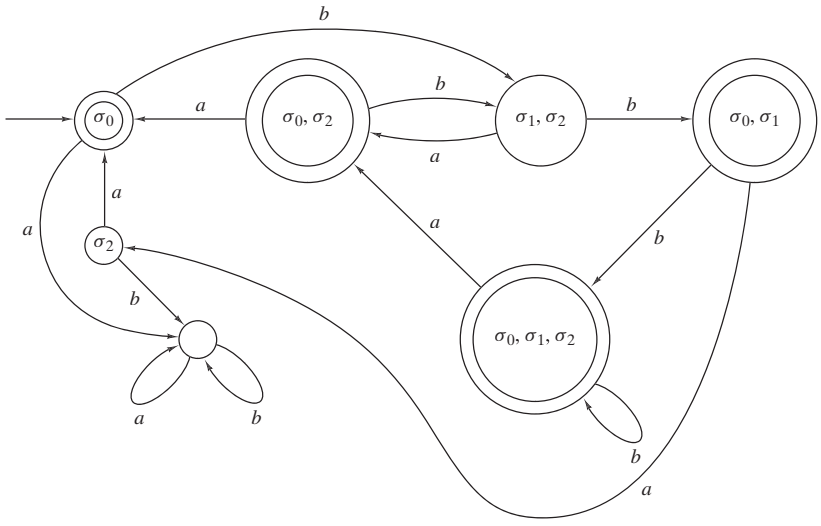
1. Sea $A = (\mathcal{I}, \mathcal{S}, f, \mathcal{A}, \sigma)$ un autómata de estado finito no determinístico. Un autómata de estado finito no determinístico equivalente se puede construir como sigue. El conjunto de estados es el conjunto potencia de $\mathcal{S}$. El conjunto de símbolos de entrada es $\mathcal{I}$ (sin cambios). El símbolo de inicio es $\{\sigma\}$ (en esencia sin cambios). El conjunto de estados aceptantes consiste en todos los subconjuntos de $\mathcal{S}$ que contienen al menos un estado aceptante de A. La función del siguiente estado está definida por la regla

$$f'(X, x) = \begin{cases} \emptyset & \text{si } X = \emptyset \\ \bigcup_{S \in X} f(S, x) & \text{si } X \neq \emptyset. \end{cases}$$

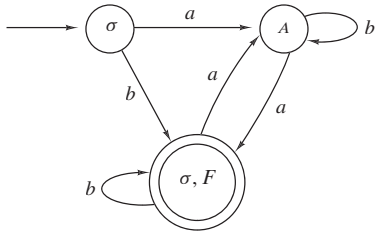
2. Un lenguaje L es normal si y sólo si existe un autómata de estado finito que acepta precisamente las cadenas de L.

Sección 12.5

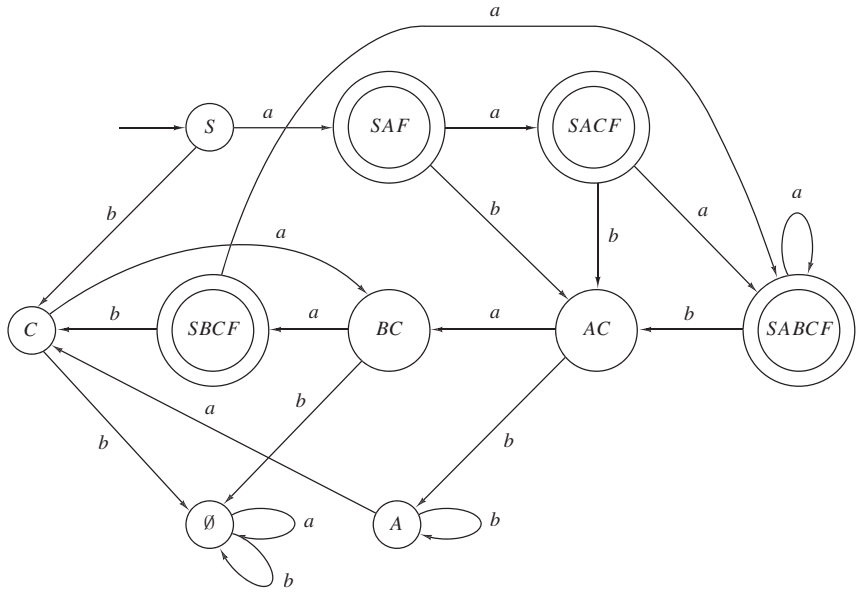
1. (Para el ejercicio 1)



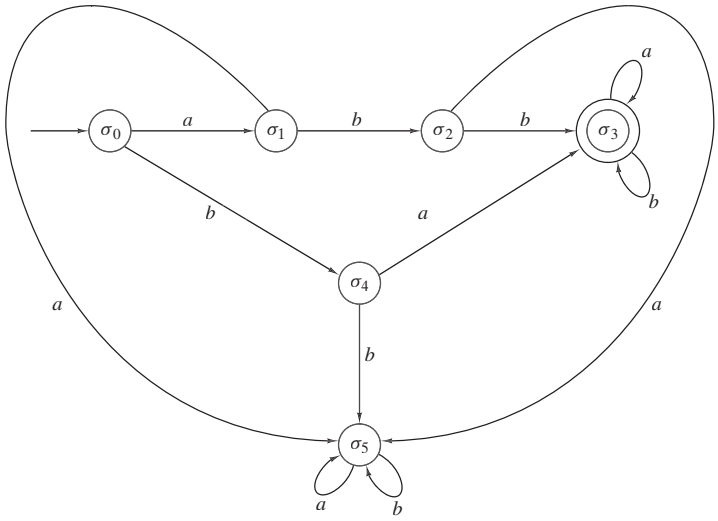
2.



5.

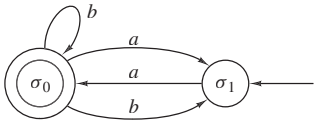


7. (Para el ejercicio 21)

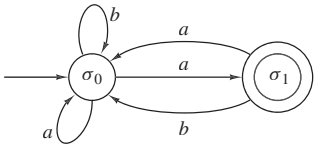


10. La figura 12.5.7 acepta la cadena ba^n, $n > 1$, y las cadenas que terminan con b^2 o aba^n, $n \geq 1$. Mediante el ejemplo 12.5.8, se ve que la figura 12.5.9 acepta la cadena a^nb, $n \geq 1$, y las cadenas que comienzan con b^2 o a^nba, $n \geq 1$.

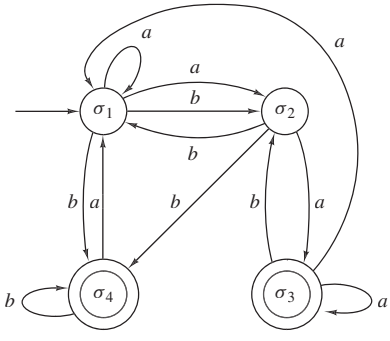
11.



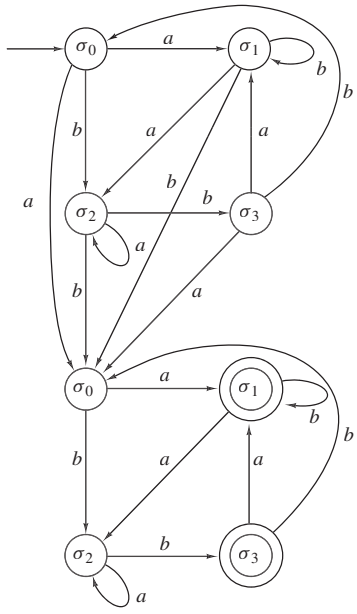
14.



17.



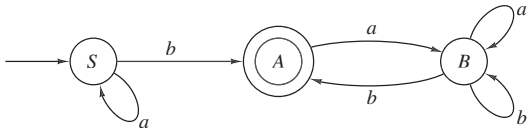
20.



22. $\sigma_0 \to a\sigma_1, \sigma_0 \to b\sigma_2, \sigma_0 \to a, \sigma_1 \to a\sigma_0,$
$\sigma_1 \to a\sigma_2, \sigma_1 \to b\sigma_1, \sigma_1 \to b, \sigma_2 \to b\sigma_0$

25. Suponga que L es normal. Entonces existe un autómata de estado finito A con $L = \text{Ac}(A)$. Suponga que A tiene k estados. Considere la cadena a^kbba^k y argumente como en el ejemplo 12.5.6.

28. La afirmación es falsa. Considere el lenguaje normal $L = \{a^nb \mid n \geq 0\}$, que es aceptado por el autómata de estado finito



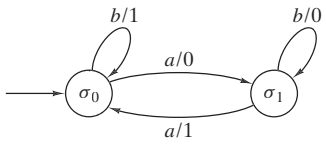
El lenguaje

$$L' = \{u^n \mid u \in L, n \in \{1, 2, \ldots\}\}$$

no es normal. Suponga que L' es normal. Entonces existe un autómata de estado finito A que acepta a L'. En particular, A acepta a $a^n b$ para toda n. Se deduce que para n suficientemente grande, la trayectoria que representa $a^n b$ contiene un ciclo de longitud k. Como A acepta a $a^n b a^n b$, también acepta a $a^{n+k} b a^n b$, lo cual es una contradicción.

Capítulo 12 Autoevaluación

1.

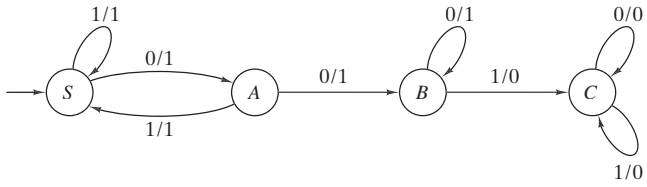


2. $\mathcal{I} = \{a, b\}$; $\mathcal{O} = \{0, 1\}$; $\mathcal{S} = \{S, A, B\}$; estado inicial $= S$

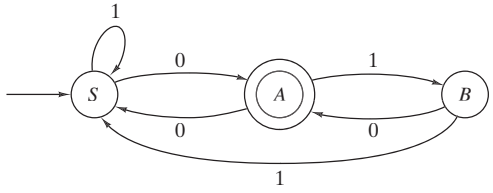
	f		g	
$\mathcal{S}$ \ $\mathcal{I}$	a	b	a	b
S	A	A	0	0
A	S	B	1	1
B	A	B	1	0

3. 1101.

4.

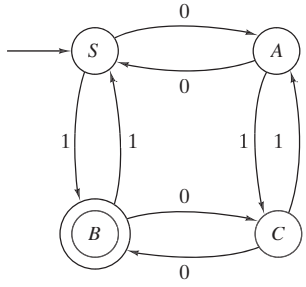


5.



6. Sí.

7.



8. Todo 0 está seguido por un 1.

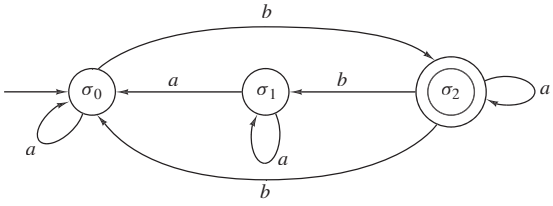
9. Libre de contexto

10. $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaaAbbb \Rightarrow aaaaAbbbb \Rightarrow aaaabbbb$

11. $a^i b^j$, $j \leq 2 + i$, $j \geq 1$, $i \geq 0$

12. $S \rightarrow ASB$, $S \rightarrow AB$, $AB \rightarrow BA$, $BA \rightarrow AB$, $A \rightarrow a$, $B \rightarrow b$

13.



14. $\mathcal{I} = \{a, b\}$; $\mathcal{S} = \{\sigma_0, \sigma_1, \sigma_2\}$; $\mathcal{A} = \{\sigma_0\}$; estado inicial $= \sigma_0$

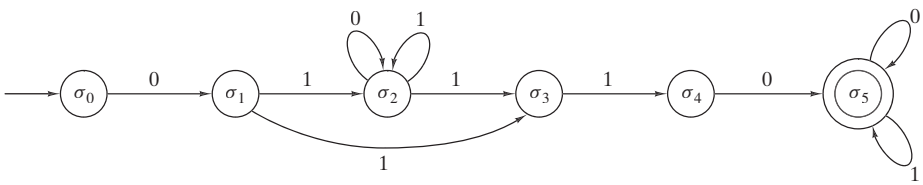
	a	b
$\mathcal{S}$ \ $\mathcal{I}$		
σ_0	$\{\sigma_0, \sigma_1\}$	$\emptyset$
σ_1	$\emptyset$	$\{\sigma_2\}$
σ_2	$\{\sigma_0, \sigma_2\}$	$\{\sigma_2\}$

15. Sí, ya que la trayectoria

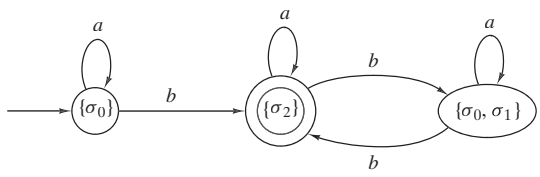
$$(\sigma_0, \sigma_0, \sigma_1, \sigma_2, \sigma_2, \sigma_2, \sigma_2, \sigma_0)$$

representa *aabaaba* y σ_0 es un estado aceptante.

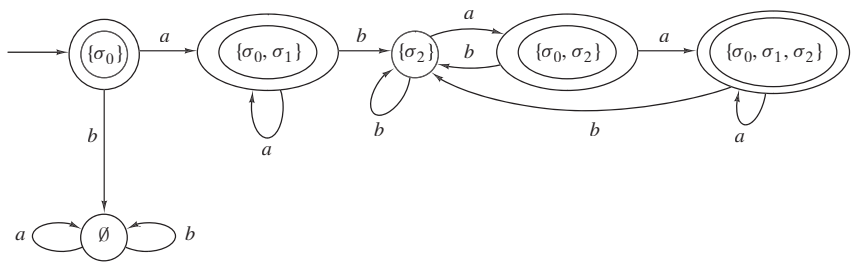
16.



17.



18.



19. Combine el autómata de estado finito no determinístico que acepta L_1 y L_2 de la siguiente manera. Sea S el estado inicial de L_2. Para cada arista de la forma (S_1, S_2) etiquetada a en L_1 donde S_2 es un estado aceptante, se agrega la arista (S_1, S) con etiqueta a. El estado inicial del autómata de estado finito no determinístico es el estado inicial de L_1. Los estados aceptantes del autómata de estado finito no determinístico son los estados aceptantes de L_2.

20. Sea A' un autómata de estado finito no determinístico que acepta un lenguaje normal que no contiene la cadena nula. Se agrega un estado F. Para cada arista, (σ, σ') con etiqueta a en A' donde σ' es aceptante, se agrega una arista (σ, F) con etiqueta a. Se hace F el único estado aceptante. El autómata de estado finito no determinístico A tiene un estado aceptante. Se asegura que $\text{Ac}(A) = \text{Ac}(A')$.

Se demuestra que $\text{Ac}(A) \subseteq \text{Ac}(A')$. [El argumento para $\text{Ac}(A') \subseteq \text{Ac}(A)$ es similar y se omite]. Suponga que $\alpha \in \text{Ac}(A)$. Existe una trayectoria

$$(\sigma_0, \sigma_1, \ldots, \sigma_{n-1}, \sigma_n)$$

que representa α en A, con σ_n como estado aceptante. Como $\alpha \neq \lambda$, hay un último símbolo a en α. Entonces la arista (σ_{n-1}, σ_n) se etiqueta a. Ahora la trayectoria

$$(\sigma_0, \sigma_1, \ldots, \sigma_{n-1}, F)$$

representa a α en A' y termina en un estado aceptante. Por lo tanto, $\alpha \in \text{Ac}(A')$.

Para ver que la afirmación es falsa para un lenguaje normal arbitrario, considere el lenguaje normal

$$L = \{\lambda\} \cup \{0^i \mid i \text{ es impar}\}$$

y un autómata de estado finito no determinístico A con estado inicial S que acepta a L. Como $\lambda \in L$, S es un estado aceptante. Si S tiene un lazo en S con etiqueta 0, entonces A acepta todas las cadenas de 0; por lo tanto, no hay ningún lazo en S con etiqueta 0. Como $0 \in L$ y no hay lazo en S, existe una arista de S a un estado aceptante $S' \neq S$, lo cual es una contradicción. Por lo tanto, A tiene al menos dos estados aceptantes.

Sección 13.1 Repaso

1. La geometría para el cálculo se refiere al diseño y análisis de algoritmos para resolver problemas de geometría.

2. Dados n puntos en el plano, encuentre el par más cercano.

3. Calcule la distancia entre cada par de puntos y seleccione la distancia mínima.

4. Encuentre una línea vertical l que divide los puntos en dos partes casi iguales. Después, de manera recursiva, resuelva el problema para cada parte. Sea δ_L la distancia entre un par más cercano en la parte izquierda y sea δ_R la distancia entre un par más cercano en la parte derecha. Sea $\delta = \text{mín}\{\delta_L, \delta_R\}$. Después examine los puntos que están dentro de una franja vertical de ancho 2δ centrada en l. Ordene los puntos en esta franja en orden creciente de la coordenada y, y examine los puntos en este orden. Calcule la distancia entre cada punto p y los siguientes siete puntos. Siempre que haya un par cuya distancia sea menor que δ, actualice δ. Al terminar, δ es la distancia entre el par más cercano.

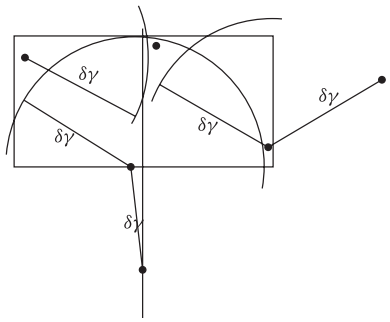
5. El tiempo en el peor caso del algoritmo de fuerza bruta es $\Theta(n^2)$. El tiempo en el peor caso del algoritmo de divide y vencerás es $\Theta(n \lg n)$.

Sección 13.1

1. Los 16 puntos ordenados por la coordenada x son $(1,2)$, $(1,5)$, $(1,9)$, $(3,7)$, $(3,11)$, $(5,4)$, $(5,9)$, $(7,6)$, $(8,4)$, $(8,7)$, $(8,9)$, $(11,3)$, de manera que el punto que divide es (7, 6). Después se encuentra $\delta_L = \sqrt{8}$, la distancia mínima entre los puntos de la izquierda $(1,2)$, $(1,5)$, $(1,9)$, $(3,7)$, $(3,11)$, $(5,4)$, $(5,9)$, $(7,6)$, y $\delta_R = 2$, la distancia mínima entre los puntos de la derecha $(8,4)$, $(8,7)$, $(8,9)$, $(11,3)$, $(11,7)$, $(12,10)$, $(14,7)$, $(17,10)$. Entonces $\delta = \text{mín}\{\delta_L, \delta_R\} = 2$. Los puntos ordenados por la coordenada y en la franja vertical son (8, 4), (7, 6), (8, 7), (8, 9). En este caso se compara cada punto en la franja con todos los puntos que siguen. Las distancias de (8, 4) a (7, 6), (8, 7), (8, 9) nos son menores que 2, así que δ no se actualiza en este punto. La distancia de (7, 6) a (8, 7) es $\sqrt{2}$, entonces δ se actualiza a $\sqrt{2}$. Las distancias de (7, 6) a (8, 9) y de (8, 7) a (8, 9) son mayores que $\sqrt{2}$, de manera que δ se queda en $\sqrt{2}$. Por lo tanto, la distancia entre el par más cercano es $\sqrt{2}$.

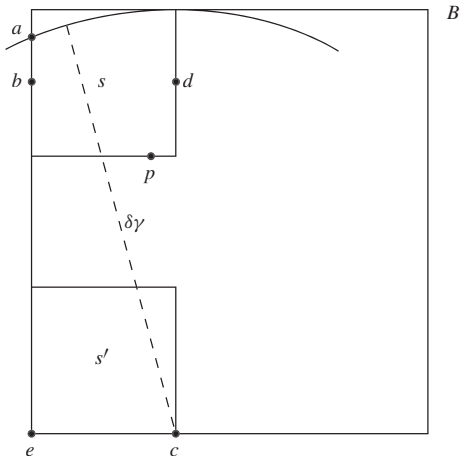
4. Considere el caso extremo cuando todos los puntos están en la línea vertical.

7.



10. Sea B cualquiera de los dos cuadros de $\delta \times \delta$, derecho o izquierdo, que forman el rectángulo de $\delta \times 2\delta$ (vea la figura 13.1.2). Se da un argumento por contradicción y se supone que B contiene cuatro puntos o más. Se hace una partición de B en cuatro cuadros de $\delta/2 \times \delta/2$ como se ilustra en la figura 13.1.3. Entonces, cada uno de los cuatro cuadros contiene cuando mucho un punto y por lo tanto exactamente un punto. En adelante se hará referencia a estos cuatro cuadros como subcuadros de B.

La figura



muestra la siguiente construcción. Se reduce el tamaño de los subcuadros, si es posible, de manera que

- Cada subcuadro contenga un punto.
- Los subcuadros sean del mismo tamaño.
- Los subcuadros sean tan pequeños como se pueda.

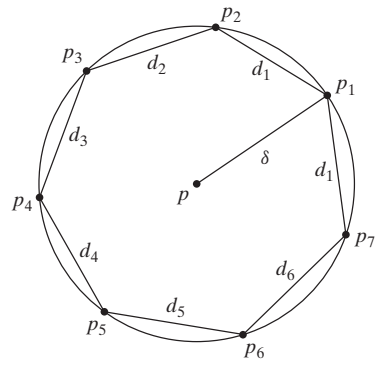
Como al menos un punto no está en una esquina de B, los subcuadros no se colapsan en puntos y entonces al menos un punto está en un lado de un subcuadro s interior a B. Se elige ese punto y se le llama p. Se selecciona un subcuadro s' cercano a p. Se etiquetan los dos puntos de la esquina de s' en el lado más lejano a p, e y c. Se dibuja un círculo de radio δ con centro en c y sea a el punto (no de esquina) donde este círculo cruza el lado de s. Observe que este círculo cruza el lado de s en un punto que no es esquina. Seleccione un punto b en s en el mismo lado de a entre a y e. Sea d el punto correspondiente en el lado opuesto de s. Ahora la longitud del diámetro del rectángulo $R = bdce$ es menor que δ; entonces, R contiene cuando mucho un punto. Esto es una contradicción puesto que R contiene a p y el punto en s'. Por lo tanto, B contiene cuando mucho tres puntos.

13. Además de $p.x$ y $p.y$, se supone que cada punto p tiene otro campo $p.lado$, que se usa para indicar si p está en el lado izquierdo o derecho cuando los puntos se dividen en dos partes casi iguales. El argumento adicional, *etiqueta*, de *rec_encuentra_todo_2δ_una_vez* establece $p.lado$ como *etiqueta* para todos los puntos p.

```
halla_todo_2δ_unavez(p, n) {
    δ = par_cercano(p, n) //procedimiento original
    if(δ > 0) {
        ordena p_1, . . . , p_n por coordenada x
        rec_halla_todo_2δ_unavez(p, 1, n, δ, λ)
            //λ = cadena vacía
    }
}
rec_halla_todo_2δ_unavez(p, i, j, δ, etiqueta)
    if(j − 1 < 3) {
        ordena p_1, . . . , p_j por coordenada y
        directamente halla y produce todos los pares
            distintos a menos de 2δ
        for k = i to j
            p_k.lado = etiqueta
        return
    }
    k = ⌊(i + j)/2⌋
    l = p_k.x
    rec_halla_todo_2δ_unavez(p, i, k, δ, L)
  rec_halla_todo_2δ_unavez(p, k + 1, j, δ, R)
    fusiona p_i, . . . , p_k y p_{k+1}, . . . , p_j por coordenada y
    t = 0
    for k = i to j
        if(p_k.x > l − 2 * δ ∧ p_k.x < l + 2 * δ) {
            t = t + 1
            v_t = p_k
        }
    for k = 1 to t − 1
        for s = k + 1 to mín{t, k + 31}
        if(dist(v_k, v_s) < 2 * δ ∧ v_k.lado¬ = v_s.lado)
            println(v_k + " " + v_s)
    for k = i to j
        p_k.lado = etiqueta
}
```

16. Se demuestra que para cada punto p, hay cuando mucho 6 puntos distintos cuya distancia a p es δ. Se deriva después que el número de pares a distancia δ es menor o igual que $6n$.

Se da un argumento por contradicción. Suponga que algún punto p tiene 7 vecinos diferentes $p_1, \ldots, p_7$ cuya distancia a p es δ. Entonces se tiene la situación



Sea C la circunferencia de este círculo. Como cada d_i está al menos a δ, se tiene

$$2\pi\delta = C > \sum_{i=1}^{7} d_i \geq 7\delta.$$

Por lo tanto, $\pi > 72 = 3.5,$, que es una contradicción. (Una estimación más cuidadosa indica que para cada punto p, existen cuando mucho 5 puntos distintos cuya distancia a p es δ).

Sección 13.2 Repaso

1. Dado un conjunto finito de puntos S en el plano, un punto $p \in S$ es un punto del casco si existe una línea L que pasa por p tal que todos los puntos en S excepto p están en un lado de L.
2. El casco convexo de un conjunto finito de puntos S en el plano es la sucesión $p_1, p_2, \ldots, p_n$ de puntos en el casco de S listados en el siguiente orden. El punto p_1 es el punto con la menor coordenada y. Si varios puntos tienen la misma coordenada y mínima, p_1 es la que tiene la menor coordenada x. Los puntos restantes p_i se dan en orden creciente del ángulo de la horizontal al segmento de recta p_1, p_i.
3. Sea el punto p_i con coordenadas (x_i, y_i). Entonces el producto cruzado de los puntos p_0, p_1, p_2 es

$$\text{cruz}(p_0, p_1, p_2) = (y_2 - y_0)(x_1 - x_0) - (y_1 - y_0)(x_2 - x_0).$$

4. El algoritmo de Graham encuentra primero el punto p_1 con la coordenada y mínima. Si ésta es igual para varios puntos, el punto que se elige es el que tiene la menor coordenada x. Después ordena los puntos restantes p_i según el ángulo de la horizontal al segmento de recta p_1, p_i. Luego examina tercias de puntos sucesivas. Si estos puntos hacen un giro a la izquierda, se conserva el punto medio. Si los puntos hacen un giro a la derecha, el punto medio se descarta. Al terminar el algoritmo, los puntos restantes en orden son el casco convexo.
5. $\Omega(n \lg n)$
6. Cualquier algoritmo de casco convexo se puede usar para ordenar números reales cuyos valores estén entre 0 y 1. Primero se proyectan los puntos sobre el círculo unitario (vea la figura 13.2.11). Después se usa el algoritmo de casco convexo para encontrar el casco convexo. Luego las coordenadas y del casco convexo (en orden) dan el ordenamiento de los puntos originales. Como el tiempo en el peor caso de cualquier algoritmo para ordenar es $\Omega(n \lg n)$, el tiempo en el peor caso de cualquier algoritmo de casco convexo también debe ser $\Omega(n \lg n)$.

Sección 13.2

1. Sea L la línea horizontal que pasa por p_1. Por la elección de p_1, ningún punto de S está abajo de L. Si p_1 es el único punto de S en L, p_1 es un punto del casco. Si otros puntos de S están en L, todos están a la derecha de p_1 (por la elección de p_1). En este caso, si L se gira ligeramente con centro en p_1, en el sentido de las manecillas del reloj, L contendrá sólo a p_1 y los otros puntos de S estarán arriba de L. De nuevo se concluye que p_1 está en el casco.
4. Los puntos [ordenados respecto a (7, 1)] son (7, 1), (10, 1), (16, 4), (12, 3), (14, 5), (16, 10), (13, 8), (10, 5), (10, 9), (10, 13), (7, 7), (7, 13), (6, 10), (3, 13), (4, 8), (1, 8), (4, 4), (2, 2). La siguiente tabla muestra cada tercia que se examina en el ciclo "while", si gira a la izquierda, y la acción tomada respecto a la tercia:

Tercia	*¿Giro a la izquierda?*	*¿Se descarta punto medio?*
(7, 1), (10, 1), (16, 4)	Sí	No
(10, 1), (16, 4), (12, 3)	Sí	No
(16, 4), (12, 3), (14, 5)	No	Sí
(10, 1), (16, 4), (14, 5)	Sí	No
(16, 4), (14, 5), (16, 10)	No	Sí
(10, 1), (16, 4), (16, 10)	Sí	No
(16, 4), (16, 10), (13, 8)	Sí	No
(16, 10), (13, 8), (10, 5)	Sí	No
(13, 8), (10, 5), (10, 9)	No	Sí
(16, 10), (13, 8), (10, 9)	No	Sí
(16, 4), (16, 10), (10, 9)	Sí	No
(16, 10), (10, 9), (10, 13)	No	Sí
(16, 4), (16, 10), (10, 13)	Sí	No
(16, 10), (10, 13), (7, 7)	Sí	No
(10, 13), (7, 7), (7, 13)	No	Sí
(16, 10), (10, 13), (7, 13)	Sí	No
(10, 13), (7, 13), (6, 10)	Sí	No
(7, 13), (6, 10), (3, 13)	No	Sí
(10, 13), (7, 13), (3, 13)	No	Sí
(16, 10), (10, 13), (3, 13)	Sí	No
(10, 13), (3, 13), (4, 8)	Sí	No
(3, 13), (4, 8), (1, 8)	No	Sí
(10, 13), (3, 13), (1, 8)	Sí	No
(3, 13), (1, 8), (4, 4)	Sí	No
(1, 8), (4, 4), (2, 2)	No	Sí
(3, 13), (1, 8), (2, 2)	Sí	No

El casco convexo es (7, 1), (10, 1), (16, 4), (16, 10), (10, 13), (3, 13), (1, 8), (2, 2).

7. Después de encontrar $p_1, \ldots, p_i$, la marcha de Jarvis encuentra el punto p_{i+1} tal que p_{i-1}, p_i, p_{i+1} hacen el giro a la izquierda más pequeño. Se deduce que si la línea L que pasa por p_i, p_{i+1} se gira ligeramente en el sentido de las manecillas del reloj con centro en p_i, L contendrá sólo a p_i y el resto de los puntos de S estarán en un lado de L. Entonces p_i es un punto en el casco. Por construcción, la marcha de Jarvis encuentra todos los puntos del casco. Así que la marcha de Jarvis encuentra el casco convexo.
10. Sí. La marcha de Jarvis es más rápida cuando "la mayoría" de los puntos no está en el casco convexo.

Capítulo 13 Autoevaluación

1. Los 18 puntos ordenados según la coordenada x son (1, 8), (2, 2), (3, 13), (4, 4), (4, 8), (6, 10), (7, 1), (7, 7), (7, 13), (10, 1), (10, 5), (10, 9), (10, 13), (12, 3), (13, 8), (14, 5), (16, 4), (16, 10), de manera que el punto que divide es (7, 13). A continuación se encuentra $\delta_L = \sqrt{8}$, la distancia mínima entre los puntos de la izquierda (1, 8), (2, 2), (3, 13), (4, 4), (4, 8), (6, 10), (7, 1), (7, 7), (7, 13), y $\delta_R = \sqrt{5}$, la distancia mínima entre los puntos de la derecha (10, 1), (10, 5), (10, 9), (10, 13), (12, 3), (13, 8), (14, 5), (16, 4), (16, 10). Entonces $\delta = \text{mín}\{\delta_L, \delta_R\} = \sqrt{5}$. Los puntos, ordenados según la coordenada y en la franja vertical son (7, 1), (7, 7), (6, 10), (7, 13). En este caso se compara cada punto en la franja con todos los puntos que siguen. Como ningún par está más cerca que $\sqrt{5}$, el algoritmo no actualiza δ. Por lo tanto, la distancia entre los puntos del par más cercano es $\sqrt{5}$.
2. Si se reemplaza "tres" por "dos", cuando hay tres puntos, el algoritmo sería llamado de manera recursiva con entradas de tamaños 1 y 2. Pero un conjunto consistente en un punto no tiene par, mucho menos un par más cercano.

3. Cada cuadro de $\delta/2 \times \delta/2$ contiene cuando mucho un punto, entonces hay cuando mucho cuatro puntos en la mitad inferior del rectángulo.

4. $\Theta(n(\lg n)^2)$

5. Sea L la línea vertical que pasa por p. Por la elección de p, ningún punto de S está a la derecha de L. Si p es el único punto de S en L, p es un punto del casco. Si otros puntos de S están en L, todos están abajo de p. En este caso, si se gira L ligeramente en el sentido de las manecillas del reloj con centro en p, L contendrá sólo a p y el resto de los puntos de S estarán a la izquierda de L. De nuevo se concluye que p es un punto en el casco.

6. Sea L el segmento de recta que une p y q. Sea L' una línea que pasa por p perpendicular a L. No puede haber otro punto r de S sobre L' o en el lado opuesto a q de L', ya que si existiera ese punto r, la distancia de r a q sería mayor que la distancia de p a q, lo cual es imposible. Entonces p es un punto del casco. De manera similar, q es un punto del casco.

7. Los puntos [ordenados respecto a (1, 2)] son (1, 2), (11, 3), (8, 4), (14, 7), (5, 4), (11, 7), (17, 10), (7, 6), (8, 7), (12, 10), (8, 9), (5, 9), (3, 7), (3, 11), (1, 5), (1, 9). La siguiente tabla muestra cada tercia examinada en el ciclo "while", si gira a la izquierda, y la acción tomada respecto a la tercia.

Tercia	*¿Giro a la izquierda?*	*¿Se descarta punto medio?*
(1,2), (11,3), (8,4)	Sí	No
(11,3), (8,4), (14,7)	No	Sí
(1,2), (11,3), (14,7)	Sí	No
(11,3), (14,7), (5,4)	Sí	No
(14,7), (5,4), (11,7)	No	Sí
(11,3), (14,7), (11,7)	Sí	No
(14,7), (11,7), (17,10)	No	Sí
(11,3), (14,7), (17,10)	No	Sí
(1,2), (11,3), (17,10)	Sí	No
(11,3), (17,10), (7,6)	Sí	No
(17,10), (7,6), (8,7)	No	Sí
(11,3), (17,10), (8,7)	Sí	No
(17,10), (8,7), (12,10)	No	Sí
(11,3), (17,10), (12,10)	Sí	No
(17,10), (12,10), (8,9)	Sí	No
(12,10), (8,9), (5,9)	No	Sí
(17,10), (12,10), (5,9)	Sí	No
(12,10), (5,9), (3,7)	Sí	No
(5,9), (3,7), (3,11)	No	Sí
(12,10), (5,9), (3,11)	No	Sí
(17,10), (12,10), (3,11)	No	Sí
(11,3), (17,10), (3,11)	Sí	No
(17,10), (3,11), (1,5)	Sí	No
(3,11), (1,5), (1,9)	No	Sí
(17,10), (3,11), (1,9)	Sí	No

El casco convexo es (1, 2), (11, 3), (17, 10), (3, 11), (1, 9).

8. Se corre parte del algoritmo de Graham que sigue al ordenamiento de los puntos restantes.

Apéndice A

1. $\begin{pmatrix} 2+a & 4+b & 1+c \\ 6+d & 9+e & 3+f \\ 1+g & -1+h & 6+i \end{pmatrix}$

2. $\begin{pmatrix} 5 & 7 & 7 \\ -7 & 10 & -1 \end{pmatrix}$

5. $\begin{pmatrix} 3 & 18 & 27 \\ 0 & 12 & -6 \end{pmatrix}$

8. $\begin{pmatrix} -2 & -35 & -56 \\ -7 & -18 & 13 \end{pmatrix}$

9. $\begin{pmatrix} 18 & 10 \\ 14 & -6 \\ 23 & 1 \end{pmatrix}$

12. (-4)

14. *a)* $2 \times 3, 3 \times 3, 3 \times 2$

b) $AB = \begin{pmatrix} 33 & 18 & 47 \\ 8 & 9 & 43 \end{pmatrix}$

$AC = \begin{pmatrix} 16 & 56 \\ 14 & 63 \end{pmatrix}$

$CA = \begin{pmatrix} 4 & 18 & 38 \\ 0 & 0 & 0 \\ 2 & 17 & 75 \end{pmatrix}$

$AB^2 = \begin{pmatrix} 177 & 215 & 531 \\ 80 & 93 & 323 \end{pmatrix}$

$BC = \begin{pmatrix} 18 & 65 \\ 34 & 25 \\ 12 & 54 \end{pmatrix}$

$A = (b_{ij})$, $I_n = (a_{jk})$, $AI_n = (c_{ik})$.

17. Sea $A = (b_{ij})$, $I_n = (a_{jk})$, $AI_n = (c_{ik})$. Entonces

$$c_{ik} = \sum_{j=1}^{n} b_{ij} a_{jk} = b_{ik} a_{kk} = b_{ik}.$$

Por lo tanto, $AI_n = A$. De manera similar $I_n A = A$.

20. La solución es $X = A^{-1}C$.

Apéndice B

1. $-4x$

4. $\dfrac{15x - 3b}{3} = 5x - b$

7. $\dfrac{1}{n} - \dfrac{1}{n+1} = \dfrac{n+1-n}{n(n+1)} = \dfrac{1}{n(n+1)}$

Se puede usar esta ecuación para calcular $\sum_{i=1}^{n} \frac{1}{i(i+1)}$ como sigue:

$$\begin{aligned} &\sum_{i=1}^{n} \frac{1}{i(i+1)} \\ &\quad = \sum_{i=1}^{n} \frac{1}{i} - \frac{1}{i+1} \\ &\quad = \left(1 - \frac{1}{2}\right) + \left(\frac{1}{2} - \frac{1}{3}\right) + \cdots + \left(\frac{1}{n-1} - \frac{1}{n}\right) \\ &\qquad + \left(\frac{1}{n} - \frac{1}{n+1}\right) \\ &\quad = 1 - \frac{1}{n+1} = \frac{n+1-1}{n+1} = \frac{n}{n+1}. \end{aligned}$$

8. 81

11. 1/81

14. *a), c)* y *g)* son iguales, *b)* y *f)* son iguales, *d)* y *e)* son iguales,

16. $x^2 + 8x + 15$

19 $x^2 + 8x + 16$

22. $x^2 - 4$

25. $(x + 5)(x + 1)$

28. $(x - 4)^2$

31. $(2x + 1)(x + 5)$

34. $(2x + 3)(2x - 3)$

37. $(n + 1)! + (n + 1)(n + 1)! = (n + 1)![1 + (n + 1)] = (n + 1)!(n + 2) = (n + 2)!$

40.
$$\begin{aligned}
&7(3 \cdot 2^{n-1} - 4 \cdot 5^{n-1}) - 10(3 \cdot 2^{n-2} - 4 \cdot 5^{n-2}) \\
&\quad = 2^{n-2}(7 \cdot 3 \cdot 2 - 10 \cdot 3) + 5^{n-2}(-7 \cdot 4 \cdot 5 + 10 \cdot 4) \\
&\quad = 2^{n-2} \cdot 12 + 5^{n-2}(-100) \\
&\quad = 2^{n-2}(2^2 \cdot 3) - 5^{n-2}(5^2 \cdot 4) \\
&\quad = 3 \cdot 2^n - 4 \cdot 5^n
\end{aligned}$$

42. La factorización da $(x - 4)(x - 2) = 0$, que tiene soluciones $x = 4, 2$.

45. $2x \leq 6, x \leq 3$

48. $i \leq n$ para $i = 1, \ldots, n$. Al sumar estas desigualdades se obtiene

$$\sum_{i=1}^{n} i \leq n \cdot n = n^2.$$

51. Se multiplica por $(n + 2)n^2(n + 1)^2$ para obtener

$$(2n + 1)(n + 1)^2 > 2(n + 2)n^2$$

o bien

$$2n^3 + 5n^2 + 4n + 1 > 2n^3 + 4n^2$$

o

$$n^2 + 4n + 1 > 0,$$

que es cierto si $n \geq 1$.

54. 6

57. 10

59. 2.584962501

62. −0.736965594

64. 2.392231208

67. 0.480415248

68. 1.489896102

71. Sea $u = \log_b y$ y $v = \log_b x$. Por definición $b^u = y$, y $b^v = x$. Ahora

$$x^{\log_b y} = x^u = (b^v)^u = b^{vu} = (b^u)^v = y^v = y^{\log_b x}.$$

Apéndice C

1. Primero se hace *grande* igual a 2 e *i* igual a 2. Como $i \leq n$ es verdadera, se ejecuta el cuerpo del ciclo "while". Como $s_i > grande$ es verdadera, se hace *grande* igual a 3 e *i* igual a 2; se ejecuta el ciclo "while" de nuevo.

Como $i \leq n$ es verdadera, se ejecuta el cuerpo del ciclo "while". Como $s_i > grande$ es verdadera, el valor de *grande* se hace igual a 8; *i* se hace igual a 4 y se ejecuta de nuevo el ciclo "while".

Como $i \leq n$ es verdadera, se ejecuta el cuerpo del ciclo "while". Como $s_i > grande$ es falsa, el valor de *grande* no cambia; *i* se hace igual a 5 y se ejecuta de nuevo el ciclo "while".

Como $i \leq n$ es falsa, el ciclo "while" termina. El valor de *grande* es 8, el elemento mayor en esta sucesión.

4. Primero *x* se hace 4. Como $b > x$ es falsa, $x = b$ *no* se ejecuta. Como $c > x$ es verdadera, $x = c$ se ejecuta y *x* se hace igual a 5. Entonces *x* es el mayor de los números *a*, *b* y *c*.

7.
```
mín(a, b) {
    if (a < b)
        return a
    else
        return b
}
```

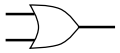
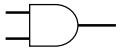
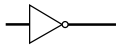
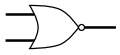
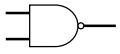
10.
```
impares(n) {
    i = 1
    while (i ≤ n) {
        println(i)
        i = i + 2
    }
}
```

13.
```
producto(s, n) {
    producto_parcial = 1
    for i = 1 to n
        producto_parcial = producto_parcial * s_i
    return producto_parcial
```

ÍNDICE

LISTA DE SÍMBOLOS

ÁLGEBRAS BOOLEANAS Y CIRCUITOS

$x \vee y$	x o y (1 si x o y es 1, 0 de otra manera); p.471
$x \wedge y$	x y y (1 si x y y son 1, 0 de otra manera); p. 471
$x \oplus y$	OR-exclusivo de x y y (0 si $x = y$, 1 de otra manera); p. 488
$\bar{x}$	no x (0 si x es 1, 1 si x es 0); p. 471
$x \downarrow y$	x NOR y, se lee "no x o y" (0 si x o y es 1, 1 de otra manera); p. 500
$x \uparrow y$	x NAND y, se lee "no x y y" (0 si x y y son 1, 1 de otra manera); p. 494
	compuerta OR; p. 471
	compuerta AND; p. 471
	compuerta NOT (inversor); p. 471
	compuerta NOR; p. 500
	compuerta NAND; p. 494

CADENAS, GRAMÁTICAS Y LENGUAJES

λ	cadena nula; p. 109
$\|s\|$	longitud de la cadena s; p. 109
st	concatenación de las cadenas s y t (s seguida de t); p. 109
a^n	$aa \cdot \cdot \cdot a$ (n símbolos a); p. 109
X^*	conjunto de todas las cadenas en X; p. 109
X^+	conjunto de todas las cadenas no nulas en X; p. 109
$\alpha \rightarrow \beta$	producción en una gramática; p. 517
$\alpha \Rightarrow \beta$	β se puede derivar de α; p. 518
$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdot \cdot \cdot \Rightarrow \alpha_n$	derivación de α_n a partir de α_1; p. 518
$L(G)$	lenguaje generado por la gramática G; p. 518
$S ::= T$	forma normal de Bakus (BNF); p. 519
$S ::= T_1 \| T_2$	$S ::= T_1$, $S ::= T_2$; p. 519
Ac(A)	conjunto de cadenas aceptadas por A; p. 514

MATRICES

(a_{ij})	matriz con elementos a_{ij}; p. 556
$A = B$	las matrices A y B son iguales (A y B son del mismo tamaño y sus elementos correspondientes son iguales), p. 556
$A + B$	suma de matrices; p. 557
cA	producto escalar; p. 557
$-A$	$(-1)A$; p. 557
$A - B$	$A + (-\mathrm{B})$; p. 557
AB	producto de matrices; p. 557
A^n	producto de matrices $AA \cdot \cdot \cdot A$ (n matrices A); p. 558

Misceláneo

$\lg x$	logaritmo base 2 de x; p. 159
$\ln x$	logaritmo natural de x (logaritmo base e de x); p. 171
$m \bmod n$	residuo cuando m se divide entre n; p. 89
$\lfloor x \rfloor$	piso de x (mayor entero menor o igual que x); p. 91
$\lceil x \rceil$	techo de x (menor entero mayor o igual que x); p. 91
$b \mid a$	b divide a a; p. 183
$b \nmid a$	b no divide a a; p. 183
$\text{mcd}(a, b)$	máximo común divisor de a y b; p. 188
$\text{mcm}(a, b)$	mínimo común múltiplo de a y b; p. 189
$n!$	n factorial $[n(n-1) \cdot \cdots \cdot 2 \cdot 1]$; p. 55
f_n	n-ésimo número de Fibonacci; p. 177
C_n	n-ésimo número de Catalan; p. 236
$\sum_{i=m}^{n} a_i$	$a_m + a_{m+1} + \cdots + a_n$; p. 107
$\sum_{i \in X} a_i$	la suma de los elementos en el conjunto $\{a_i \mid i \in X\}$; p. 108
$\prod_{i=m}^{n} a_i$	$a_m \cdot a_{m+1} \cdot \cdots a_n$; p. 107
$\prod_{i \in X} a_i$	el producto de los elementos en el conjunto $\{a_i \mid i \in X\}$; p. 108

Notación para algoritmos

$x = y$	asigna el valor y a x; p. 571
{ }	marca el inicio y el final de un bloque de instrucciones; p. 572
if (p) *acción*	si p es verdadera, se ejecuta la *acción*; p. 571

if (p)
  acción1
else
  acción2
	si p es verdadera, se ejecuta la *acción1*; si p es falsa, se ejecuta la *acción2*; p. 572

//	un comentario inicia con // y sigue hasta el final de la línea; p. 573
return	suspende la ejecución y regresa al invocador; p. 575
return x	suspende la ejecución y regresa el valor x al invocador; p. 575
while (p) *acción*	si p es verdadera, ejecuta la *acción* y repite este proceso; p. 573
for *var* = *ini* to *lím* *acción*	ejecuta la *acción* para valores de *var* de *ini* a *lím*; p. 574
fun(p_1,…, p_k)	invoca la función *fun* con argumentos p_1,…, p_k; p. 574
print(s_1,…, s_n)	imprime los valores s_1,…, s_n; p. 575
println(s_1,…, s_n)	imprime los valores s_1,…, s_n y luego agrega una línea nueva; p. 575