# 5

# RELACIONES DE RECURRENCIA

*¿Ya me vas a decir?*
*¿Decirte qué?*
*Qué es lo que quieres saber.*
*Sabes, es gracioso. Estás tratando de saber aquello que debo averiguar para tu padre y yo estoy tratando de descubrir por qué lo quieres saber.*
*Podrías continuar así por siempre, ¿no?*

—The Big Sleep



Este capítulo ofrece una introducción a las relaciones de recurrencia, las cuales son útiles en ciertos problemas de conteo. Una relación de recurrencia relaciona el $n$-ésimo elemento de una sucesión con sus predecesores. Debido a que las relaciones de recurrencia están íntimamente relacionadas con los algoritmos recursivos, dichas relaciones surgen de manera natural en el análisis de este tipo de algoritmos.

## 5.1 *INTRODUCCIÓN*

Consideremos las siguientes instrucciones para generar una sucesión:

1. Comenzar con 5.
2. Dado cualquier término, sumarle 3 para obtener el siguiente término.

Si enumeramos los términos de la sucesión, obtenemos

$$5, \quad 8, \quad 11, \quad 14, \quad 17, \quad \ldots . \tag{5.1.1}$$

El primero término es igual a 5 debido a la primera instrucción. El segundo término es 8 debido a que la instrucción 2 dice que debemos sumar 3 a 5 para obtener el siguiente término, 8. El tercer término es 11 debido a que la instrucción 2 dice que debemos sumar 3 a 8 para obtener el siguiente término, 11. Si seguimos las instrucciones 1 y 2, podemos calcular cualquier término de la sucesión. Las instrucciones 1 y 2 no proporcionan una fórmula explícita para el $n$-ésimo término de la sucesión, en el sentido de proporcionar una fórmula en que podamos "sustituir $n$" para obtener el valor del $n$-ésimo término, sino que al ir calculando término a término podemos obtener cualquier término de la sucesión.





Si denotamos la sucesión (5.1.1) como $a_1, a_2, \ldots$, podemos enunciar la instrucción 1 como

$$a_1 = 5 \tag{5.1.2}$$

y la instrucción 2 como

$$a_n = a_{n-1} + 3, \quad n \geq 2. \tag{5.1.3}$$

Si hacemos $n = 2$ en (5.1.3), obtenemos

$$a_2 = a_1 + 3.$$

Por (5.1.2), $a_1 = 5$; así,

$$a_2 = a_1 + 3 = 5 + 3 = 8.$$

Si hacemos $n = 3$ en (5.1.3), obtenemos

$$a_3 = a_2 + 3.$$

Como $a_2 = 8$,

$$a_3 = a_2 + 3 = 8 + 3 = 11.$$

(5.1.2) y (5.1.3) permiten calcular cualquier término de la sucesión, de la misma manera que lo hicimos utilizando las instrucciones 1 y 2. Vemos que (5.1.2) y (5.1.3) son equivalentes a las instrucciones 1 y 2.

La ecuación (5.1.3) proporciona un ejemplo de **relación de recurrencia**. Una relación de recurrencia define una sucesión dando el $n$-ésimo valor en términos de algunos de sus predecesores. En (5.1.3), el $n$-ésimo valor está dado en términos del valor inmediato anterior. Para que una relación de recurrencia como (5.1.3) defina una sucesión, hay que dar cierto valor o valores "de arranque", como (5.1.2). Estos valores de arranque son llamados **condiciones iniciales**. A continuación damos las definiciones formales.

**DEFINICIÓN 5.1.1**

Una *relación de recurrencia* para la sucesión $a_0, a_1, \ldots$ es una ecuación que relaciona $a_n$ con algunos de sus predecesores $a_0, a_1, \ldots, a_{n-1}$.

Las *condiciones iniciales* para la sucesión $a_0, a_1, \ldots$ son valores dados en forma explícita para un número finito de términos de la sucesión.

Hemos visto que es posible definir una sucesión mediante una relación de recurrencia, junto con ciertas condiciones iniciales. Daremos varios ejemplos de relaciones de recurrencia.

**EJEMPLO 5.1.2**

La sucesión de Fibonacci (véase el análisis después del algoritmo 3.4.7) se define mediante la relación de recurrencia

$$f_n = f_{n-1} + f_{n-2}, \quad n \geq 3$$

y las condiciones iniciales

$$f_1 = 1, \quad f_2 = 2.$$

□

**EJEMPLO 5.1.3**

Una persona invierte \$1000 a 12% compuesto anualmente. Si $A_n$ representa la cantidad al final de $n$ años, determinar una relación de recurrencia y condiciones iniciales que definan la sucesión $\{A_n\}$.

Al final de $n - 1$ años, la cantidad es $A_{n-1}$. Después de un año más, tendremos la cantidad $A_{n-1}$ más los intereses. Así,

$$A_n = A_{n-1} + (0.12)A_{n-1} = (1.12)A_{n-1}, \quad n \geq 1. \tag{5.1.4}$$

Para aplicar esta relación de recurrencia a $n = 1$, necesitamos saber el valor de $A_0$. Como $A_0$ es la cantidad del principio, tenemos la condición inicial

$$A_0 = 1000. \tag{5.1.5}$$

□

La condición inicial (5.1.5) y la relación de recurrencia (5.1.4) nos permiten calcular el valor de $A_n$ para cualquier $n$. Por ejemplo,

$$\begin{aligned} A_3 &= (1.12)A_2 = (1.12)(1.12)A_1 \\ &= (1.12)(1.12)(1.12)A_0 = (1.12)^3(1000) = 1404.93. \end{aligned} \tag{5.1.6}$$

Así, al final del tercer año, la cantidad es \$1404.93.

El cálculo (5.1.6) se puede realizar para un valor arbitrario de $n$ para obtener

$$\begin{aligned} A_n &= (1.12)A_{n-1} \\ &\vdots \\ &= (1.12)^n(1000). \end{aligned}$$

Vemos que en ciertas ocasiones podemos deducir una fórmula explícita a partir de una relación de recurrencia y las condiciones iniciales. La determinación de fórmulas explícitas a partir de las relaciones de recurrencia es el tema de la sección 5.2.

Aunque es fácil obtener una fórmula explícita a partir de la relación de recurrencia y la condición inicial para la sucesión del ejemplo 5.1.3, no es tan inmediata la forma de obtener una fórmula explícita para la sucesión de Fibonacci. En la sección 5.2 daremos un método que proporcionará una fórmula explícita para la sucesión de Fibonacci.

Las relaciones de recurrencia, los algoritmos recursivos y la inducción matemática tienen una relación muy estrecha. En las tres, se suponen conocidos casos anteriores del caso en cuestión. Una relación de recurrencia utiliza valores anteriores en una sucesión para calcular el valor actual. Un algoritmo recursivo utiliza instancias menores de la entrada actual para calcular ésta. El paso inductivo en una demostración por inducción matemática supone la verdad de instancias anteriores del enunciado, para demostrar la verdad del enunciado en cuestión.

Una relación de recurrencia que define una sucesión se puede convertir de manera directa en un algoritmo para el cálculo de la sucesión. Por ejemplo, el algoritmo 5.1.4, deducido de la relación de recurrencia (5.1.4) y la condición inicial (5.1.5), calcula la sucesión del ejemplo 5.1.3.



**ALGORITMO 5.1.4** ***Cálculo del interés compuesto***

Este algoritmo recursivo calcula la cantidad de dinero al final de $n$ años, suponiendo un capital inicial de \$1000 y una tasa de interés de 12% compuesto anualmente.

Entrada: $n$, el número de años

Salida: La cantidad de dinero al cabo de $n$ años

```
1. procedure compound_interest(n)
2.   if n = 0 then
3.     return(1000)
4.   return(1.12 * compound_interest(n − 1))
5. end compound_interest
```

El algoritmo 5.1.4 es una traducción directa de las ecuaciones (5.1.4) y (5.1.5) que definen a la sucesión $A_0, A_1, \ldots$. Las líneas 2 y 3 corresponden a la condición inicial (5.1.5) y la línea 4 corresponde a la relación de recurrencia (5.1.4).

**EJEMPLO 5.1.5**

Sea $S_n$ el número de subconjuntos de un conjunto con $n$ elementos. Como el paso de un conjunto con $(n - 1)$ elementos a un conjunto con $n$ elementos duplica el número de subconjuntos (véase el teorema 2.1.4), obtenemos la relación de recurrencia

$$S_n = 2S_{n-1}.$$

La condición inicial es

$$S_0 = 1.$$

□

Una de las principales razones para el uso de las relaciones de recurrencia es que a veces es más fácil determinar el $n$-ésimo término de una sucesión a partir de sus predecesores que determinar una fórmula explícita para el $n$-ésimo término en términos de $n$. Los siguientes ejemplos pretenden ilustrar esta tesis.

**EJEMPLO 5.1.6**

Sea $S_n$ el número de cadenas de $n$ bits que no contienen el patrón 111. Desarrollar una relación de recurrencia para $S_1, S_2, \ldots$, y las condiciones iniciales que definen la sucesión $S$.

Contaremos 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 los conjuntos 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 las cantidades de cadenas de los tipos (a), (b) y (c). Supongamos que una cadena de $n$ bits comienza con 0 y que no contiene al patrón 111. Entonces, la cadena de $(n-1)$ bits que va después del 0 inicial no contiene al patrón 111. Como después del 0 inicial puede aparecer cualquier cadena de $(n-1)$ bits que no contenga 111, existen $S_{n-1}$ cadenas de tipo (a). Si una cadena de $n$ bits comienza con 10 y no contiene al patrón 111, entonces la cadena de $(n-2)$ bits posterior al 10 inicial no puede contener al patrón 111; por tanto, existen $S_{n-2}$ cadenas de tipo (b). Si una cadena de $n$ bits comienza con 11 y no contiene al patrón 111, entonces el tercer bit debe ser 0. La cadena de $(n-3)$ bits posterior al 110 inicial no puede contener el patrón 111; por tanto, existen $S_{n-3}$ cadenas de tipo (c). Así,

$$S_n = S_{n-1} + S_{n-2} + S_{n-3}, \quad n \geq 4.$$

Encontramos las condiciones iniciales por inspección:

$$S_1 = 2, \quad S_2 = 4, \quad S_3 = 7.$$

□

**EJEMPLO 5.1.7**

El lector debe recordar (véase el ejemplo 4.2.23) que el número de Catalan $C_n$ es igual al número de rutas que van de la esquina inferior izquierda de una retícula de $n \times n$ a la esquina superior derecha si sólo podemos recorrerla hacia la derecha o hacia arriba y si sólo se permite tocar pero no rebasar una recta diagonal, de la esquina inferior izquierda a la esquina superior derecha. Una ruta de este tipo será una *ruta buena*. Daremos una relación de recurrencia para los números de Catalan.

Separaremos las rutas buenas en clases, con base en el punto donde la ruta toca la diagonal después de salir de la esquina inferior izquierda. Por ejemplo, la ruta de la figura 5.1.1 corta por vez primera la diagonal en el punto (3, 3). Consideraremos que las rutas que tocan por vez primera la diagonal en $(k, k)$ se construyen mediante un proceso de dos pasos: En primer lugar, se construye la parte de (0, 0) a $(k, k)$. En segundo lugar se construye

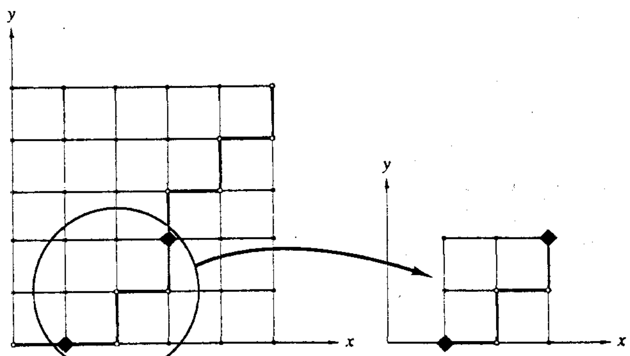



FIGURA 5.1.1 Descomposición de una ruta buena.





la parte de $(k, k)$ a $(n, n)$. Una ruta buena siempre sale de (0, 0) yendo 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 $(k, k-1)$ proporcionan una ruta buena en la retícula $(k-1) \times (k-1)$ con esquinas en (1, 0), $(1, k-1)$, $(k, k-1)$ y $(k, 0)$. [En la figura 5.1.1, hemos marcado los puntos (1, 0) y $(k, k-1)$, $k = 3$, con diamantes, y hemos separado la subretícula $(k-1) \times (k-1)$.] Así, existen $C_{k-1}$ rutas de (0, 0) a $(k, k)$ que cortan por vez primera la diagonal en $(k, k)$. La parte de $(k, k)$ a $(n, n)$ es una ruta buena en la retícula $(n-k) \times (n-k)$ con esquinas en $(k, k)$, $(k, n)$, $(n, n)$ y $(n, k)$ (véase la figura 5.1.1). Existen $C_{n-k}$ de estas rutas. Por el principio de multiplicación, existen $C_{k-1}C_{n-k}$ rutas buenas en una retícula $n \times n$ que cortan por vez primera la diagonal en $(k, k)$. Las rutas buenas que cortan por vez primera la diagonal en $(k, k)$ son distintas de aquellas que cortan por vez primera la diagonal en $(k', k')$, si $k \neq k'$. Así, podemos utilizar el principio de la suma para obtener una relación de recurrencia para la cantidad total de rutas buenas en una retícula de $n \times n$:

$$C_n = \sum_{k=1}^{n} C_{k-1}C_{n-k}.$$

□

**EJEMPLO 5.1.8** ***Torre de Hanoi***

La Torre de Hanoi es un juego que consta de tres postes montados sobre un tablero y $n$ discos de diversos tamaños con agujeros en sus centros (véase la figura 5.1.2). Se supone que si un disco está en algún poste, sólo se puede colocar sobre tal disco otro con diámetro menor. Dados todos los discos apilados en un poste, como en la figura 5.1.2, el problema consiste en transferir los discos a otro poste, moviendo un disco a la vez.

Daremos una solución y luego determinaremos una relación de recurrencia y una condición inicial para la sucesión $c_1, c_2, \ldots,$ donde $c_n$ denota el número de movimientos necesarios en nuestra solución del problema con $n$ discos. Luego mostraremos que nuestra solución es óptima; es decir, mostraremos que ninguna otra solución utiliza menos movimientos.

Daremos un algoritmo recursivo. Si sólo existe un disco, basta moverlo al poste deseado. Si tenemos $n > 1$ discos en el poste 1, como en la figura 5.1.2, primero llamamos de manera recursiva a nuestro algoritmo, para mover los $n-1$ discos superiores al poste 2 (véase la figura 5.1.3). Durante estos movimientos, el disco inferior en el poste 1 permanece fijo. A continuación movemos el disco restante del poste 1 al poste 3. Por último, de nuevo llamamos de manera recursiva a nuestro algoritmo para mover los $n-1$ discos del poste 2 al poste 3. Con esto hemos podido mover $n$ discos del poste 1 al poste 3.

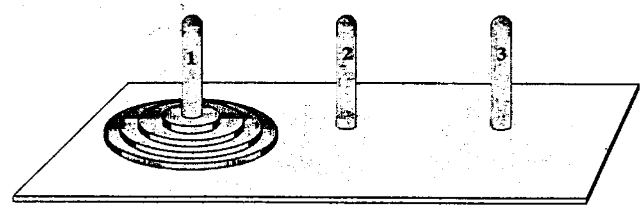



FIGURA 5.1.2 Torre de Hanoi.

Si $n > 1$, resolvemos dos veces el problema con $(n - 1)$ discos y movemos de manera explícita un disco. Por tanto,

$$c_n = 2c_{n-1} + 1, \quad n > 1.$$

La condición inicial es

$$c_1 = 1.$$

En la sección 5.2 mostraremos que $c_n = 2^n - 1$.

A continuación mostraremos que nuestra solución es óptima. Sea $d_n$ el número de movimientos necesarios por una solución óptima. Utilizaremos la inducción matemática para mostrar que

$$c_n = d_n, \quad n \geq 1. \tag{5.1.7}$$

**PASO BASE ($n = 1$).** Por inspección,

$$c_1 = 1 = d_1;$$

de modo que (5.1.7) es verdadera para $n = 1$.

**PASO INDUCTIVO.** Supongamos que (5.1.7) es verdadera para $n - 1$. Consideremos el momento de una solución óptima para el problema con $n$ discos en que el disco de mayor tamaño se mueve por vez primera. Este disco debe estar sólo en un poste (para que pueda moverse) y otro poste debe estar vacío (de modo que este poste pueda recibir el disco de mayor tamaño). Así, los $n - 1$ discos menores deben estar apilados en un tercer poste (véase la figura 5.1.3). En otras palabras, debe haberse resuelto el problema con $n - 1$ discos, lo cual requería al menos $d_{n-1}$ movimientos. Luego se mueve el disco mayor, para lo cual se requiere un movimiento adicional. Por último, en algún momento, los $n - 1$ discos se colocan sobre el disco mayor, lo que requiere al menos $d_{n-1}$ movimientos adicionales. Esto implica que

$$d_n \geq 2d_{n-1} + 1.$$

Por la hipótesis de inducción, $c_{n-1} = d_{n-1}$. Así,

$$d_n \geq 2d_{n-1} + 1 = 2c_{n-1} + 1 = c_n. \tag{5.1.8}$$

La última igualdad es consecuencia de la relación de recurrencia para la sucesión $c_1, c_2, \ldots$. Por definición, ninguna solución puede realizarse en menos movimientos que la solución óptima, de modo que

$$c_n \geq d_n. \tag{5.1.9}$$

Combinamos las desigualdades (5.1.8) y (5.1.9) para obtener

$$c_n = d_n.$$

Con esto concluye el paso inductivo. Por tanto, nuestra solución es óptima. □

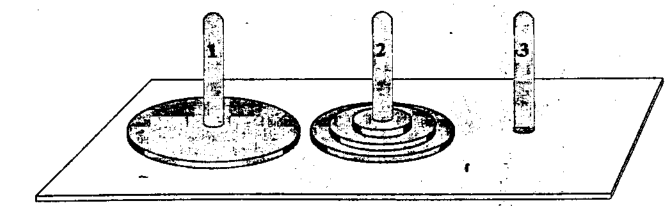



FIGURA 5.1.3 Estado de la Torre de Hanoi después de mover de manera recursiva los $n - 1$ discos superiores del poste 1 al poste 2.



El juego de la Torre de Hanoi fue ideado por el matemático francés Édouard Lucas a finales del siglo XIX. (Lucas fue la primera persona que llamó a la sucesión 1, 2, 3, 5, . . . sucesión de Fibonacci.) Se creó la siguiente leyenda para acompañar al juego (y, suponemos, para apoyar su comercialización). Se decía que el juego se había deducido de una mítica torre de oro con 64 discos. Los 64 discos debían ser trasladados por monjes, de acuerdo con las reglas ya establecidas. Se decía que antes de que los monjes terminasen de mover la torre, ésta caería y el mundo llegaría a su fin. Como al menos se necesitan $2^{64} - 1 = 18{,}446{,}744{,}073{,}709{,}551{,}615$ movimientos para resolver el juego de la Torre de Hanoi con 64 discos, podemos estar seguros de que algo ocurriría a la torre antes de moverla por completo.

**EJEMPLO 5.1.9** ***La telaraña en la economía***

Supongamos un modelo económico en el cual la oferta y la demanda están dadas mediante ecuaciones lineales (véase la figura 5.1.4). En forma específica, la demanda está dada por la ecuación

$$p = a - bq,$$

donde $p$ es el precio, $q$ es la cantidad y $a$ y $b$ son parámetros positivos. La idea es que si el precio aumenta, los consumidores demandan menos producto. La oferta está dada por la ecuación

$$p = kp,$$

donde $p$ es el precio, $q$ es la cantidad y $k$ es un parámetro positivo. La idea es que si el precio se incrementa, el fabricante está dispuesto a ofrecer mayores cantidades del producto.

También suponemos que existe un cierto retraso en la reacción de la oferta a los cambios. (Por ejemplo, se necesita cierto tiempo para fabricar bienes o para que crezcan las cosechas.) Denotamos los intervalos de tiempo discretos como $n = 0, 1, \ldots$. Suponemos que la demanda está dada por la ecuación

$$p_n = a - bq_n;$$

es decir, en el instante $n$, la cantidad $q_n$ del producto se venderá al precio $p_n$. Suponemos que la oferta está dada por la ecuación

$$p_n = kq_{n+1}; \tag{5.1.10}$$

es decir, se necesita una unidad de tiempo para que el fabricante ajuste la cantidad $q_{n+1}$, en el instante $n + 1$, al precio $p_n$, en el instante anterior $n$.

Si despejamos $q_{n+1}$ en (5.1.10) y sustituimos el valor obtenido en la ecuación de la demanda para el instante $n + 1$,

$$p_{n+1} = a - bq_{n+1},$$

obtenemos la relación de recurrencia

$$p_{n+1} = a - \frac{b}{k} p_n$$

para el precio. En la sección 5.2 resolveremos esta relación de recurrencia.

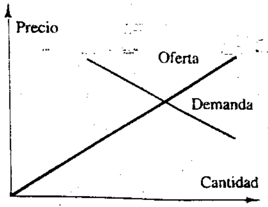



FIGURA 5.1.4 Un modelo económico.

```
1.  procedure insertion_sort(s, n)
2.    if n = 1 then
3.      return
4.    insertion_sort(s, n − 1)
5.    i := n − 1
6.    temp := s_n
7.    while i ≥ 1 and s_i > temp do
8.      begin
9.      s_{i+1} := s_i
10.     i := i − 1
11.     end
12.   s_{i+1} := temp
13. end insertion_sort
```

Sea $b_n$ el número de veces que se realiza, en el peor de los casos, la comparación $s_i > temp$ en la línea 7. Suponga que si $i < 1$, la comparación $s_i > temp$ no se realiza.

39. Explique por qué el algoritmo 5.3.14 ordena la sucesión.
40. ¿Cuál entrada produce el comportamiento en el peor de los casos para el algoritmo 5.3.14?
41. Determine $b_1, b_2$ y $b_3$.
42. Determine una relación de recurrencia para la sucesión $\{b_n\}$.
43. Resuelva la relación de recurrencia del ejercicio 42.

Los ejercicios 44-46 se refieren al algoritmo 5.3.15.

**ALGORITMO 5.3.15**

Entrada: $s_1, \ldots, s_n, n$
Salida: $s_1, \ldots, s_n$

```
procedure algor(s, n)
  i := n
  while i ≥ 1 do
    begin
    s_i := s_i + 1
    i := ⌊i/2⌋
    end
  n := ⌊n/2⌋
  if n ≥ 1 then
    algor(s, n)
end algor
```

Sea $b_n$ el número de veces que se ejecuta el enunciado $s_i := s_i + 1$.

44. Determine una relación de recurrencia para la sucesión $\{b_n\}$ y calcule $b_1, b_2$ y $b_3$.
45. Resuelva la relación de recurrencia del ejercicio 44 cuando $n$ sea una potencia de 2.
46. Demuestre que $b_n = \Theta((\lg n)^2)$.
**47.** Resuelva la relación de recurrencia

$$a_n = 3a_{\lfloor n/2 \rfloor} + n, \qquad n > 1,$$

cuando $n$ sea una potencia de 2. Suponga que $a_1 = 1$.



48. Muestre que $a_n = \Theta(n^{\lg 3})$, donde $a_n$ es como en el ejercicio 47.

Los ejercicios 49-56 se refieren a un algoritmo que acepta como entrada la sucesión

$$s_i, \ldots, s_j.$$

Si $j > i$, los subproblemas

$$s_i, \ldots, s_{\lfloor (i+j)/2 \rfloor} \quad \text{y} \quad s_{\lfloor (i+j)/2+1 \rfloor}, \ldots, s_j$$

se resuelven de manera recursiva. Las soluciones de los subproblemas de tamaños $m$ y $k$ se pueden combinar en un tiempo $c_{m,k}$ para resolver el problema original. Sea $b_n$ el tiempo necesario para que el algoritmo procese una entrada de tamaño $n$.

**49.** Escriba una relación de recurrencia para $b_n$, suponiendo que $c_{m,k} = 3$.
50. Escriba una relación de recurrencia para $b_n$, suponiendo que $c_{m,k} = m + k$.
51. Resuelva la relación de recurrencia del ejercicio 49 para el caso en que $n$ sea una potencia de 2, suponiendo que $b_1 = 0$.
**52.** Resuelva la relación de recurrencia del ejercicio 49 para el caso en que $n$ sea una potencia de 2, suponiendo que $b_1 = 1$.
53. Resuelva la relación de recurrencia del ejercicio 50 para el caso en que $n$ sea una potencia de 2, suponiendo que $b_1 = 0$.
54. Resuelva la relación de recurrencia del ejercicio 50 para el caso en que $n$ sea una potencia de 2, suponiendo que $b_1 = 1$.
☆ **55.** Suponga que si $m_1 \geq m_2$ y $k_1 \geq k_2$, entonces $c_{m_1,k_1} \geq c_{m_2,k_2}$. Muestre que la sucesión $b_1, b_2, \ldots$ es creciente.
★ 56. Suponga que $c_{m,k} = m + k$ y $b_1 = 0$, y muestre que $b_n \leq 4n \lg n$.

Los ejercicios 57-62 se refieren a la siguiente situación. Sea $P_n$ un problema particular de tamaño $n$. Si $P_n$ se divide en subproblemas de tamaños $i$ y $j$, existe un algoritmo que combina las soluciones de estos dos subproblemas en una solución de $P_n$ en un tiempo a lo más de $2 + \lg(ij)$. Suponga que ya se ha resuelto un problema de tamaño 1.

**57.** Escriba un algoritmo recursivo para resolver $P_n$, similar al algoritmo 5.3.8.
58. Sea $a_n$ el tiempo, en el peor de los casos, para resolver $P_n$ mediante el algoritmo del ejercicio 57. Muestre que

$$a_n \leq a_{\lfloor n/2 \rfloor} + a_{\lfloor (n+1)/2 \rfloor} + 2 \lg n.$$

59. Sea $b_n$ la relación de recurrencia obtenida a partir del ejercicio 58 reemplazando "$\leq$" por "$=$". Suponga que $b_1 = a_1 = 0$. Muestre que si $n$ es una potencia de 2,

$$b_n = 4n - 2 \lg n - 4.$$

**60.** Muestre que $a_n \leq b_n$ para $n = 1, 2, 3, \ldots$.
61. Muestre que $b_n \leq b_{n+1}$ para $n = 1, 2, 3, \ldots$.
62. Muestre que $a_n \leq 8n$ para $n = 1, 2, 3, \ldots$.
**63.** Suponga que $\{a_n\}$ es una sucesión creciente y que siempre que $m$ divida a $n$,

$$a_n = a_{n/m} + d,$$

donde $d$ es un número real positivo y $m$ es un entero que satisface $m > 1$. Muestre que $a_n = \Theta(\lg n)$.
★ 64. Suponga que $\{a_n\}$ es una sucesión creciente y que siempre que $m$ divida a $n$,

$$a_n = ca_{n/m} + d,$$

donde $c$ y $d$ son números reales positivos que satisfacen $c > 1$ y $d > 0$, y $m$ es un entero que satisface $m > 1$. Muestre que $a_n = \Theta(n^{\log_m c})$.
65. [Proyecto] Investigue otros algoritmos de ordenamiento. Considere de manera específica la complejidad, los análisis empíricos y las características particulares de los algoritmos (véase [Knuth, 1973, vol. 3]).

## NOTAS

Las relaciones de recurrencia se analizan con más detalle en [Liu, 1985; Roberts; y Tucker]. [Cormen] presenta varias aplicaciones al análisis de algoritmos.

[Cull] proporciona algoritmos para resolver ciertos problemas del tipo de la Torre de Hanoi con un mínimo de complejidad en el espacio y el tiempo. [Hinz] es un amplio análisis de la Torre de Hanoi con 50 referencias.

La telaraña de la economía apareció por vez primera en [Ezekiel].

Todos los libros relativos a las estructuras de datos y los algoritmos contienen amplios análisis de la búsqueda y el ordenamiento (véase, por ejemplo, [Brassard; Cormen; Knuth, 1973, vol. 3; Kruse; y Nyhoff]).

Las relaciones de recurrencia también se llaman ecuaciones en diferencias. [Goldberg] contiene un análisis de las ecuaciones en diferencias y sus aplicaciones.

## CONCEPTOS BÁSICOS DEL CAPÍTULO

*Sección 5.1*

Relación de recurrencia

Condición inicial

Interés compuesto

Torre de Hanoi

Telaraña en la economía

Función de Ackermann

*Sección 5.2*

Solución de una relación de recurrencia por iteración

Relación de recurrencia, lineal y homogénea de orden $n$ con coeficientes constantes y la forma de resolver una relación de recurrencia de segundo orden

Crecimiento de poblaciones

*Sección 5.3*

Cómo determinar una relación de recurrencia que describa el tiempo necesario para ejecutar un algoritmo recursivo

Ordenamiento por selección

Búsqueda binaria

Fusión de sucesiones

Ordenamiento por fusión

## AUTOEVALUACIÓN DEL CAPÍTULO

*Sección 5.1*

**1.** Responda las partes (a)-(c) para la sucesión definida mediante las reglas:
1. El primer término es 3.
2. El $n$-ésimo término es $n$ más el término anterior.
   (a) Escriba los cuatro primeros términos de la sucesión.
   (b) Determine una condición inicial para la sucesión.
   (c) Determine una relación de recurrencia para la sucesión.

**2.** Suponga que una persona invierte $4000 a 17% compuesto anualmente. Sea $A_n$ la cantidad al final de $n$ años. Determine una relación de recurrencia y una condición inicial para la sucesión A0, A1, . . . .

**3.** Sea $P_n$ el número de particiones de un conjunto con $n$ elementos. Muestre que la sucesión $P_0, P_1, \ldots$ satisface la relación de recurrencia

$$P_n = \sum_{k=0}^{n-1} C(n-1, k)\, P_k.$$



**4.** Suponga que tenemos un tablero rectangular $2 \times n$ dividido en $2n$ cuadrados. Sea $a_n$ el número de formas de cubrir exactamente este tablero con dominóes $1 \times 2$. Muestre que la sucesión $\{a_n\}$ satisface la relación de recurrencia

$$a_n = a_{n-1} + a_{n-2}.$$

Muestre que $a_n = f_n$, donde $\{f_n\}$ es la sucesión de Fibonacci.

*Sección 5.2*

**5.** ¿Es la relación de recurrencia

$$a_n = a_{n-1} + a_{n-3}$$

una relación de recurrencia lineal homogénea con coeficientes constantes?

En los ejercicios 6 y 7, resuelva la relación de recurrencia sujeta a las condiciones iniciales.

**6.** $a_n = -4a_{n-1} - 4a_{n-2}$; $a_0 = 2$, $a_1 = 4$

**7.** $a_n = 3a_{n-1} + 10a_{n-2}$; $a_0 = 4$, $a_1 = 13$

**8.** Sea $c_n$ el número de cadenas sobre $\{0, 1, 2\}$ de longitud $n$ que tienen una cantidad 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 5.3*

Los ejercicios 9-12 se refieren al siguiente algoritmo.

**ALGORITMO** ***Evaluación de un polinomio***

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: $p(t)$

```
procedure poly(c,n, t)
  if n = 0 then
    return(c_0)
  return(t · poly(c, n − 1, t) + c_n)
end poly
```

Sea $b_n$ el número de multiplicaciones necesarias para calcular $p(t)$.

**9.** Determine 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 obtener $c_k t^{n-k}$. ¿Cuántas multiplicaciones se necesitarían para calcular $p(t)$? ¿Preferiría usted este método o el algoritmo anterior? Explique.

```
1.  procedure merge_sort (s, i, j)
2.  // caso base: i = j
3.  if i = j then
4.    return
5.  // dividir la sucesión y ordenar
6.  m := ⌊(i + j)/2⌋
7.  call merge_sort (s, i, m)
8.  call merge_sort (s, m + 1, j)
9.  // fusión
10. call merge (s, i, m, j, c)
11. // copiar c, la salida de la fusión, en s
12. for k := i to j do
13.   s_k := c_k
14. end merge_sort
```

**EJEMPLO 5.3.9**

La figura 5.3.2 muestra la forma en que el algoritmo 5.3.8 ordena la sucesión

$$12,\quad 30,\quad 21,\quad 8,\quad 6,\quad 9,\quad 1,\quad 7.$$

□

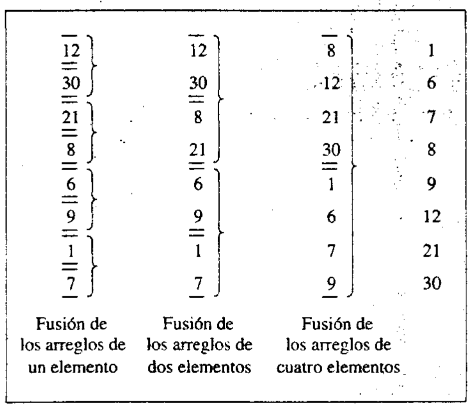



**FIGURA 5.3.2** Ordenamiento por fusión.

Concluimos esta sección mostrando que el ordenamiento por fusión (algoritmo 5.3.8) es $\Theta(n \lg n)$ en el peor de los casos. El método de demostración es igual al utilizado para mostrar que la búsqueda binaria es $\Theta(\lg n)$ en el peor de los casos.

**TEOREMA 5.3.10**

*El ordenamiento por fusión (algoritmo 5.3.8) es $\Theta(n \lg n)$ en el peor de los casos.*



**Demostración.** Sea $a_n$ el número de comparaciones necesarias para que el algoritmo 5.3.8 ordene $n$ elementos en el peor de los casos. Entonces $a_1 = 0$. Si $n > 1$, $a_n$ es a lo más la suma de los números de comparaciones en el peor de los casos resultante de las llamadas recursivas en las líneas 7 y 8, y el número de comparaciones en el peor de los casos necesarias para la fusión en la línea 10. Es decir,

$$a_n \le a_{\lfloor n/2 \rfloor} + a_{\lfloor (n+1)/2 \rfloor} + n - 1.$$

De hecho, esta cota superior se puede alcanzar (véase el ejercicio 11), de modo que

$$a_n = a_{\lfloor n/2 \rfloor} + a_{\lfloor (n+1)/2 \rfloor} + n - 1.$$

Primero resolvemos la anterior relación de recurrencia cuando $n$ es una potencia de 2, digamos, $n = 2^k$. La ecuación se convierte en

$$a_{2^k} = 2a_{2^{k-1}} + 2^k - 1.$$

Podemos resolver esta última ecuación mediante el método de iteración (véase la sección 5.2):

$$\begin{aligned}
a_{2^k} &= 2a_{2^{k-1}} + 2^k - 1 \\
&= 2[2a_{2^{k-2}} + 2^{k-1} - 1] + 2^k - 1 \\
&= 2^2 a_{2^{k-2}} + 2 \cdot 2^k - 1 - 2 \\
&= 2^2[2a_{2^{k-3}} + 2^{k-2} - 1] + 2 \cdot 2^k - 1 - 2 \\
&= 2^3 a_{2^{k-3}} + 3 \cdot 2^k - 1 - 2 - 2^2 \\
&\ \vdots \\
&= 2^k a_{2^0} + k \cdot 2^k - 1 - 2 - 2^2 - \cdots - 2^{k-1} \\
&= k \cdot 2^k - (2^k - 1) \\
&= (k-1)2^k + 1. \qquad (5.3.8)
\end{aligned}$$

Un valor arbitrario de $n$ está entre dos potencias de 2, digamos,

$$2^{k-1} < n \le 2^k. \qquad (5.3.9)$$

Como la sucesión $a$ es creciente (véase el ejercicio 14),

$$a_{2^{k-1}} \le a_n \le a_{2^k}. \qquad (5.3.10)$$

Observe que (5.3.9) implica

$$k - 1 < \lg n \le k. \qquad (5.3.11)$$

(5.3.8), (5.3.10) y (5.3.11) implican que

$$\begin{aligned}
\Omega(n \lg n) = (-2 + \lg n)\frac{n}{2} &< (k-2)2^{k-1} + 1 = a_{2^{k-1}} \\
&\le a_n \le a_{2^k} \le k2^k + 1 \le (1 + \lg n)2n + 1 = O(n \lg n).
\end{aligned}$$

Por tanto, $a_n = \Theta(n \lg n)$, de modo que el ordenamiento por fusión es $\Theta(n \lg n)$ en el peor de los casos. ■

Como ya hemos observado, en la sección 7.7 mostraremos que cualquier algoritmo de ordenamiento basado en comparaciones es $\Omega(n \lg n)$ en el peor de los casos. Este resultado implica, en particular, que el ordenamiento por fusión es $\Omega(n \lg n)$ en el peor de los casos. Si ya hubiéramos demostrado este resultado, para probar que el ordenamiento por fusión es $\Theta(n \lg n)$ en el peor de los casos, sería suficiente demostrar que el ordenamiento por fusión es $O(n \lg n)$ en el peor de los casos.

Aunque el ordenamiento por fusión (algoritmo 5.3.8) es óptimo, podría no ser el algoritmo a elegir para un problema de ordenamiento particular. Habría que tomar en cuenta algunos factores, como el tiempo en el caso promedio, el número de elementos por ordenar, la memoria disponible, las estructuras de datos por utilizar, el hecho de que los elementos por ordenar estén en la memoria o en dispositivos de almacenamiento periférico como discos o cintas, el hecho de que los elementos por ordenar estén "casi" ordenados, o el hardware por utilizar.

## Ejercicios

Los ejercicios 1-4 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 la forma en que el algoritmo 5.3.2 se ejecuta cuando $key$ = '$G$'.

2. Muestre la forma en que el algoritmo 5.3.2 se ejecuta cuando $key$ = '$P$'.

3. Muestre la forma en que el algoritmo 5.3.2 se ejecuta cuando $key$ = '$C$'.

**4.** Muestre la forma en que el algoritmo 5.3.2 se ejecuta cuando $key$ = '$Z$'.

5. Sea $a_n$ el tiempo de la búsqueda binaria (algoritmo 5.3.2) en el peor de los casos. Demuestre que $a_n \le a_{n+1}$ para $n \ge 1$.

6. Demuestre que si $a_n$ es el número de veces que se llama, en el peor de los casos, al algoritmo de búsqueda binaria (algoritmo 5.3.2) para una sucesión con $n$ elementos, entonces

$$a_n = 2 + \lfloor \lg n \rfloor$$

para cada entero positivo $n$.

**7.** Supongamos que el algoritmo $A$ necesita $\lceil n \lg n \rceil$ comparaciones para ordenar $n$ elementos y el algoritmo $B$ necesita $\lceil n^2/4 \rceil$ comparaciones para ordenar $n$ elementos. ¿Para cuáles $n$ el algoritmo $B$ es mejor que el algoritmo $A$?

8. Muestre la forma en que el ordenamiento por fusión (algoritmo 5.3.8) ordena la sucesión 1, 9, 7, 3.

9. Muestre la forma en que el ordenamiento por fusión (algoritmo 5.3.8) ordena la sucesión 2, 3, 7, 2, 8, 9, 7, 5, 4.

**10.** Suponga que tenemos dos sucesiones, cada una de tamaño $n$, ordenadas en forma creciente.

(a) ¿Bajo qué condiciones se alcanza el número máximo de comparaciones en el algoritmo 5.3.5?

(b) ¿Bajo qué condiciones se alcanza el número mínimo de comparaciones en el algoritmo 5.3.5?

11. Sea $a_n$ como en la demostración del teorema 5.3.10. Describa la entrada para la cual

$$a_n = a_{\lfloor n/2 \rfloor} + a_{\lfloor (n-1)/2 \rfloor} + n - 1.$$

12. ¿Cuál es el número mínimo de comparaciones necesarias para que el algoritmo 5.3.8 ordene un arreglo de tamaño 6?

**13.** ¿Cuál es el número máximo de comparaciones necesarias para que el algoritmo 5.3.8 ordene un arreglo de tamaño 6?



14. Sea $a_n$ como en la demostración del teorema 5.3.10. Demuestre que $a_n \le a_{n+1}$ para toda $n \ge 1$.

15. Sea $a_n$ el número de comparaciones necesarias para el ordenamiento por fusión en el peor de los casos. Muestre que $a_n \le 3n \lg n$ para $n = 1, 2, 3, \ldots$.

16. Muestre que en el mejor de los casos, el ordenamiento por fusión necesita $\Theta(n \lg n)$ comparaciones.

Los ejercicios 17-21 se refieren al algoritmo 5.3.11.

**ALGORITMO 5.3.11** ***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: $a$ (un número real), $n$ (un entero positivo)

Salida: $a^n$

```
1. procedure exp1(a, n)
2.   if n = 1 then
3.     return(a)
4.   m := ⌊n/2⌋
5.   return(exp1(a, m) · exp1(a, n − m))
6. end exp1
```

Sea $b_n$ el número de multiplicaciones (línea 5) necesarias para calcular $a^n$.

**17.** Explique por qué el algoritmo 5.3.11 calcula $a^n$.

**18.** Determine una relación de recurrencia y condiciones iniciales para la sucesión $\{b_n\}$.

**19.** Calcule $b_2$, $b_3$ y $b_4$.

**20.** Resuelva la relación de recurrencia del ejercicio 18, cuando $n$ es una potencia de 2.

**21.** Demuestre que $b_n = n - 1$ para cada entero positivo $n$.

Los ejercicios 22-27 se refieren al algoritmo 5.3.12.

**ALGORITMO 5.3.12** ***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: $a$ (un número real), $n$ (un entero positivo)

Salida: $a^n$

```
procedure exp2(a, n)
  if n = 1 then
    return(a)
  m := ⌊n/2⌋
  power := exp2(a, m)
  power := power · power
  if n is even then
    return(power)
  else
    return(power · a)
end exp2
```

Sea $b_n$ el número de multiplicaciones (líneas 6 y 10) necesarias para calcular $a^n$.

22. Explique por qué el algoritmo 5.3.12 calcula $a^n$.
23. Muestre que

$$b_n = \begin{cases} b_{(n-1)/2} + 2, & \text{si } n \text{ es impar;} \\ b_{n/2} + 1, & \text{si } n \text{ es par.} \end{cases}$$

24. Determine $b_1, b_2, b_3$ y $b_4$.
25. Resuelva la relación de recurrencia del ejercicio 23 cuando $n$ es una potencia de 2.
26. Muestre, mediante un ejemplo, que $b$ no es creciente.
☆ 27. Demuestre que $b_n = \Theta(\lg n)$.

Los ejercicios 28-33 se refieren al algoritmo 5.3.13.

**ALGORITMO 5.3.13** ***Determinación de los elementos máximo y mínimo de una sucesión***

Este algoritmo recursivo determina los elementos máximo y mínimo de una sucesión.

Entrada: $si, \ldots, sj$, $i$ y $j$

Salida: *large* (el elemento máximo de una sucesión), *small* (el elemento mínimo de una sucesión)

```
1.  procedure large_small(s, i, j, large, small)
2.    if i = j then
3.      begin
4.      large := s_i
5.      small := s_i
6.      return
7.      end
8.    m := ⌊(i + j)/2⌋
9.    large_small(s, i, m, large_left, small_left)
10.   large_small(s, m + 1, j, large_right, small_right)
11.   if large_left > large_right then
12.     large := large_left
13.   else
14.     large := large_right
15.   if small_left > small_right then
16.     small := small_right
17.   else
18.     small := small_left
19. end large_small
```

Sea $b_n$ el número de comparaciones (líneas 11 y 15) necesarias para una entrada de tamaño $n$.

28. Explique por qué el algoritmo 5.3.13 determina los elementos máximo y mínimo.
29. Muestre que $b_1 = 0$ y $b_2 = 2$.
30. Determine $b_3$.
31. Establezca la relación de recurrencia

$$b_n = b_{\lfloor n/2 \rfloor} + b_{\lfloor (n+1)/2 \rfloor} + 2 \tag{5.3.12}$$

para $n > 1$.



32. Resuelva la relación de recurrencia (5.3.12) cuando $n$ es una potencia de 2, para obtener

$$b_n = 2n - 2, \quad n = 1, 2, 4, \ldots$$

33. Utilice inducción para mostrar que

$$b_n = 2n - 2$$

para cada entero positivo $n$.

Los ejercicios 34-37 se refieren al algoritmo 5.3.13, con las líneas siguientes introducidas después de la línea 7.

```
7a. if j = i + 1 then
7b.   begin
7c.   if s_i > s_j then
7d.     begin
7e.     large := s_i
7f.     small := s_j
7g.     end
7h.   else
7i.     begin
7j.     small := s_i
7k.     large := s_j
7l.     end
7m.   return
7n.   end
```

Sea $b_n$ el número de comparaciones (líneas 7c, 11 y 15) para una entrada de tamaño $n$.

**34.** Muestre que $b_1 = 0$ y $b_2 = 1$.
**35.** Calcule $b_3$ y $b_4$.
**36.** Muestre que la relación de recurrencia (5.3.12) es válida para $n > 2$.
**37.** Resuelva la relación de recurrencia (5.3.12) cuando $n$ es una potencia de 2 para obtener

$$b_n = \frac{3n}{2} - 2, \quad n = 2, 4, 8, \ldots$$

★ **38.** Modifique el algoritmo 5.3.13 insertando las líneas anteriores al ejercicio 34 después de la línea 7 y reemplazando la línea 8 con lo siguiente.

```
8a. if j − i is odd and (1 + j − i)/2 is odd then
8b.   m := ⌊(i + j)/2⌋ − 1
8c. else
8d.   m := ⌊(i + j)/2⌋
```

Muestre que en el peor de los casos, este algoritmo modificado necesita a lo más $\lceil (3n/2) - 2 \rceil$ comparaciones para determinar los elementos máximo y mínimo en un arreglo de tamaño $n$.

Los ejercicios 39-41 se refieren al algoritmo 5.3.14.

**ALGORITMO 5.3.14** ***Ordenamiento por inserción***

Este algoritmo ordena la sucesión

$$s_1, s_2, \ldots, s_n$$

en orden creciente, ordenando de manera recursiva los primeros $n - 1$ elementos y luego insertando $s_n$ en la posición correcta.

Entrada: $s_1, s_2, \ldots, s_n$ y la longitud $n$ de la sucesión

Salida: $s_1, s_2, \ldots, s_n$ ordenada de manera creciente en orden creciente, seleccionando primero al elemento máximo para colocarlo al final, y luego ordenar de manera recursiva los demás elementos.

Entrada: $s_1, s_2, \ldots, s_n$ y la longitud $n$ de la sucesión

Salida: $s_1, s_2, \ldots, s_n$, en orden creciente

```
1.  procedure selection_sort(s, n)
2.    // caso base
3.    if n = 1 then
4.      return
5.    // se encuentra el máximo
6.    max_index := 1   // se supone al inicio que s_1 es el máximo
7.    for i := 2 to n do
8.      if s_i > s_max_index then   // se encontró uno mayor, así que debe actualizarse
9.        max_index := i
10.   // se mueve el máximo al final
11.   swap(s_n, s_max_index)
12.   call selection_sort(s, n − 1)
13. end selection_sort
```

Para medir el tiempo necesario para este algoritmo, contaremos el número de comparaciones $b_n$ en la línea 8 necesarias para ordenar $n$ elementos. (Observe que los tiempos en el mejor de los casos, en el caso promedio y en el peor de los casos son todos iguales para este algoritmo.) De inmediato obtenemos la condición inicial

$$b_1 = 0.$$

Para obtener una relación de recurrencia para la sucesión $b_1, b_2, \ldots$, simulamos la ejecución del algoritmo para una entrada arbitraria de tamaño $n > 1$. Contamos el número de comparaciones en cada línea y luego sumamos estos números para obtener la cantidad total de comparaciones $b_n$. En las líneas 1-7, no hay comparaciones (del tipo que estamos contando). En la línea 8, existen $n - 1$ comparaciones (pues la línea 7 hace que la línea 8 se ejecute $n - 1$ veces). No hay comparaciones en las líneas 9-11. La llamada recursiva aparece en la línea 12, donde llamamos a este algoritmo con una entrada de tamaño $n - 1$. Pero por definición, este algoritmo requiere $b_{n-1}$ comparaciones para una entrada de tamaño $n - 1$. Así, existen $b_{n-1}$ comparaciones en la línea 12. Por tanto, la cantidad total de comparaciones es

$$b_n = n - 1 + b_{n-1},$$

lo cual da la relación de recurrencia deseada.

Nuestra relación de recurrencia se puede resolver por iteración:

$$\begin{aligned}
b_n &= b_{n-1} + n - 1 \\
&= (b_{n-2} + n - 2) + (n-1) \\
&= (b_{n-3} + n - 3) + (n-2) + (n-1) \\
&\vdots \\
&= b_1 + 1 + 2 + \cdots + (n-2) + (n-1) \\
&= 0 + 1 + 2 + \cdots + (n-1) = \frac{(n-1)n}{2} = \Theta(n^2).
\end{aligned}$$

Así, el tiempo necesario para el algoritmo 5.3.1 es $\Theta(n^2)$.



Nuestro siguiente algoritmo (algoritmo 5.3.2) es la **búsqueda binaria**. La búsqueda binaria busca un valor en una sucesión *ordenada* y regresa el índice del valor si lo encuentra, o 0, en caso contrario. El algoritmo utiliza el enfoque divide y vencerás. La sucesión se divide en dos partes casi iguales (línea 4). Si el elemento se encuentra en el punto de división (línea 5), el algoritmo termina. Si el elemento no se encuentra, como la sucesión está ordenada, una comparación adicional (línea 7) localizará la mitad de la secuencia en la que el elemento aparecerá si está presente. Luego llamamos de manera recursiva a la búsqueda binaria (línea 11) para continuar la búsqueda.

**ALGORITMO 5.3.2** ***Búsqueda binaria***

Este algoritmo busca un valor en una sucesión creciente y regresa el índice del valor si se encuentra, o 0 en caso contrario.

Entrada: Una sucesión $s_i, s_i + 1, \ldots, s_j$, $i \geq 1$, en orden creciente, un valor *key* (elemento que se busca), $i$ y $j$.

Salida: La salida es un índice $k$ para el cual $s_k = key$, o si *key* no está en la sucesión, la salida es el valor 0.

```
1.  procedure binary_search (s, i, j, key)
2.    if i > j then   // no encontrado
3.      return (0)
4.    k := ⌊(i + j) / 2⌋
5.    if key = s_k then   // encontrado
6.      return (k)
7.    if key < s_k then   // busca en la mitad izquierda
8.      j := k − 1
9.    else   // busca en la mitad derecha
10.     i := k + 1
11.   return (binary_search (s, i, j, key))
12. end binary_search
```

**EJEMPLO 5.3.3**

Ilustramos el algoritmo 5.3.2 para la entrada

$$s_1 = \text{'}B\text{'}, \quad s_2 = \text{'}D\text{'}, \quad s_3 = \text{'}F\text{'}, \quad s_4 = \text{'}S\text{'},$$

y *key* = '*S*'. En la línea 2, como $i > j$ $(1 > 4)$ es falso, pasamos a la línea 4, donde hacemos $k$ igual a 2. En la línea 5, como *key* ('*S*') no es igual a $s_2$ ('*D*'), pasamos a la línea 7. En la línea 7, $key < s_k$ ('*S*' < '*D*') es falsa, de modo que en la línea 10, hacemos $i$ igual a 3. Luego llamamos a este algoritmo con $i = 3$, $j = 4$ para buscar *key* en

$$s_3 = \text{'}F\text{'}, \quad s_4 = \text{'}S\text{'}.$$

En la línea 2, como $i > j$ $(3 > 4)$ es falso, pasamos a la línea 4, donde hacemos $k$ igual a 3. En la línea 5, como *key* ('*S*') no es igual a $s_3$ ('*F*'), pasamos a la línea 7. En la línea 7,

$key < s_k$ ('S' < 'F') es falso, de modo que en la línea 10 hacemos $i$ igual a 4. Luego llamamos de nuevo a este algoritmo con $i = j = 4$ para buscar $key$ en

$$s_4 = \text{'S'}.$$

En la línea 2, como $i > j$ ($4 > 4$) es falso, pasamos a la línea 4, donde hacemos $k$ igual a 4. En la línea 5, como $key$ ('S') es igual a $s_4$ ('S'), regresamos 4, el índice de $key$ en la sucesión $s$. □

Ahora analizaremos el peor de los casos de la búsqueda binaria. Definimos el tiempo necesario para la búsqueda binaria en el peor de los casos como el número de veces que se llama al algoritmo en el peor de los casos para una sucesión que contiene $n$ elementos. Sea $a_n$ el tiempo en el peor de los casos.

Supongamos que $n$ es 1; es decir, supongamos que la sucesión consta de un elemento $s_i$ e $i = j$. En el peor de los casos, el elemento no está en la línea 5, de modo que el algoritmo será llamado una segunda vez en la línea 11. Sin embargo, al llamarlo por segunda vez, tendremos $i > j$ y el algoritmo terminará sin éxito en la línea 3. Hemos mostrado que si $n$ es 1, el algoritmo se llama dos veces. Obtenemos la condición inicial

$$a_1 = 2. \tag{5.3.1}$$

Ahora, supongamos que $n > 1$. En este caso, $i < j$, de modo que la condición en la línea 2 es falsa. En el peor de los casos, el elemento no se encontrará en la línea 5, de modo que el algoritmo será llamado en la línea 11. Por definición, la llamada en la línea 11 requerirá $m$ total de $a_m$ llamadas, donde $m$ es el tamaño de la secuencia que se introduce en 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 de los casos ocurre con la sucesión de mayor tamaño, el número total de llamadas en la línea 11, será $a_{\lfloor n/2 \rfloor}$. La llamada original, junto con las llamadas en la línea 11 proporcionan el total de llamadas; así, obtenemos la relación de recurrencia

$$a_n = 1 + a_{\lfloor n/2 \rfloor}. \tag{5.3.2}$$

La relación de recurrencia (5.3.2) es típica de aquellas que son resultado de algoritmos del tipo divide y vencerás. Por lo general tales relaciones de recurrencia no se resuelven tan fácilmente en forma explícita (sin embargo, véase el ejercicio 6). En vez de esto, uno estima el crecimiento de la sucesión correspondiente mediante la notación theta. Nuestro método para deducir una notación theta para la sucesión definida mediante (5.3.1) y (5.3.2) ilustra un método general de manejo de tales relaciones de recurrencia. Primero *resolvemos de manera explícita* (5.3.2) en el cado de que $n$ sea una potencia de 2. Cuando $n$ no es una potencia de 2, $n$ está entre dos potencias de 2, digamos que $2^{k-1}$ y $2^k$ y $a_n$ está entre $a_{2^{k-1}}$ y $a_{2^k}$. Como se conocen fórmulas explícitas para $a_{2^{k-1}}$ y $a_{2^k}$, podemos *estimar* $a_n$ y de este modo deducir una notación theta para $a_n$.

En primer lugar, resolvemos la relación de recurrencia (5.3.2) en caso de que $n$ sea una potencia de 2. Si $n = 2^k$, (5.3.2) se escribe

$$a_{2^k} = 1 + a_{2^{k-1}}, \quad k = 1, 2, \ldots.$$

Si hacemos $b_k = a_{2^k}$, obtenemos la relación de recurrencia

$$b_k = 1 + b_{k-1}, \quad k = 1, 2, \ldots, \tag{5.3.3}$$

y la condición inicial

$$b_0 = 2.$$



La relación de recurrencia (5.3.3) se puede resolver mediante el método iterativo:

$$b_k = 1 + b_{k-1} = 2 + b_{k-2} = \cdots = k + b_0 = k + 2.$$

Así, si $n = 2^k$,

$$a_n = 2 + \lg n. \tag{5.3.4}$$

Un valor arbitrario de $n$ está entre dos potencias de 2, digamos

$$2^{k-1} < n \le 2^k. \tag{5.3.5}$$

Como la sucesión $a$ es creciente (hecho que se puede *demostrar* por inducción, véase el ejercicio 5),

$$a_{2^{k-1}} \le a_n \le a_{2^k}. \tag{5.3.6}$$

Observe que (5.3.5) implica

$$k - 1 < \lg n \le k. \tag{5.3.7}$$

De (5.3.4), (5.3.6) y (5.3.7), deducimos que

$$\lg n < 1 + k = a_{2^{k-1}} \le a_n \le a_{2^k} = 2 + k < 3 + \lg n = O(\lg n).$$

Por tanto, $a_n = \Theta(\lg n)$, de modo que la búsqueda binaria es $\Theta(\lg n)$ en el peor de los casos. Este resultado es lo bastante importante como para resaltarse como teorema.

**TEOREMA 5.3.4**

*El tiempo en el peor de los casos de la búsqueda binaria para una entrada de tamaño n es $\Theta(\lg n)$.*

**Demostración**. La demostración está antes del enunciado del teorema. ■

Como último ejemplo, presentamos y analizamos otro algoritmo de ordenamiento conocido como el **ordenamiento por fusión** (algoritmo 5.3.8). Mostraremos que el ordenamiento por fusión tiene un tiempo de ejecución en el peor de los casos de $\Theta(n \lg n)$, de modo que para entradas de gran tamaño, el ordenamiento por fusión es mucho más rápido que el ordenamiento por selección (algoritmo 5.3.1), el cual tiene un tiempo de ejecución en el peor de los casos $\Theta(n^2)$. En la sección 7.7 mostraremos que *cualquier* algoritmo de ordenamiento que compare los elementos y, con base en el resultado de una comparación, mueva los elementos dentro de un arreglo, tiene un tiempo de ejecución $\Omega(n \lg n)$ en el peor de los casos; así, el ordenamiento por fusión es óptimo dentro de esta clase de algoritmos de ordenamiento.

En el ordenamiento por fusión, la sucesión por ordenar,

$$s_i, \ldots, s_j,$$

se divide en dos sucesiones casi iguales,

$$s_i, \ldots, s_m, \quad s_{m+1}, \ldots, s_j,$$

donde $m = \lfloor (i + j)/2 \rfloor$. Cada una de estas sucesiones se ordena de manera recursiva, después de lo cual éstas se combinan para producir un arreglo ordenado de la sucesión original. El proceso de combinación de dos sucesiones ordenadas es una **fusión**.

**ALGORITMO 5.3.5** ***Fusión de dos sucesiones***

Este algoritmo combina dos sucesiones crecientes en una única sucesión creciente.

Entrada: Dos sucesiones crecientes $s_i, \ldots, s_m$ y $s_{m+1}, \ldots, s_j$, y los índices $i$, $m$ y $j$

Salida: La sucesión $c_i, \ldots, c_j$ que consta de los elementos $s_i, \ldots, s_m$ y $s_{m+1}, \ldots, s_j$ combinados en una sucesión creciente

```
1.  procedure merge(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.    // copiar el menor entre s_p y s_q
9.    while p ≤ m and q ≤ j do
10.     begin
11.     if s_p < s_q then
12.       begin
13.       c_r := s_p
14.       p := p + 1
15.       end
16.     else
17.       begin
18.       c_r := sq
19.       q := q + 1
20.       end
21.     r := r + 1
22.     end
23.   // copiar el resto de la primera sucesión
24.   while p ≤ m do
25.     begin
26.     c_r := s_p
27.     p := p + 1
28.     r := r + 1
29.     end
30.   // copiar el resto de la segunda sucesión
31.   while q ≤ j do
32.     begin
33.     c_r := s_q
34.     q := q + 1
35.     r := r + 1
36.     end
37. end merge
```



**EJEMPLO 5.3.6**

La figura 5.3.1 muestra la forma en que el algoritmo 5.3.5 fusiona las sucesiones

$$1, 3, 4 \quad 2, 4, 5, 6.$$

□

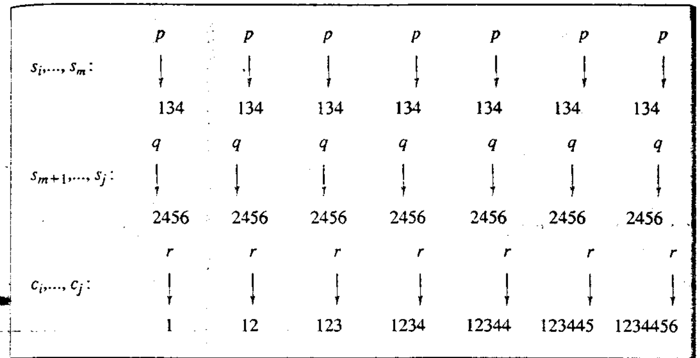



**FIGURA 5.3.1** Fusión de $s_i, \ldots, s_m$ y $s_{m+1}, \ldots, s_j$. El resultado es $c_i, \ldots, c_j$.

El teorema 5.3.7 muestra que en el peor de los casos, se necesitan $n - 1$ comparaciones para fusionar dos sucesiones tales que la suma de sus longitudes es $n$.

**TEOREMA 5.3.7**

*En el peor de los casos, el algoritmo 5.3.5 requiere $j - i$ comparaciones. En particular, en el peor de los casos, se necesitan $n - 1$ comparaciones para fusionar dos sucesiones tales que la suma de sus longitudes es $n$.*

**Demostración.** En el algoritmo 5.3.5, la comparación de los elementos de la sucesión ocurre en el ciclo while de la línea 11. El ciclo while se ejecutará mientras $p \leq m$ y $q \leq j$. Así, en el peor de los casos, el algoritmo 5.3.5 necesita $j - i$ comparaciones. ■

A continuación utilizamos el algoritmo 5.3.5 (fusión) para construir el ordenamiento por fusión.

**ALGORITMO 5.3.8** ***Ordenamiento por fusión***

Este algoritmo recursivo ordena una sucesión de manera creciente utilizando el algoritmo 5.3.5, el cual fusiona dos sucesiones crecientes.

Entrada: $s_i, \ldots, s_j$, $i$ y $j$

Salida: $s_i, \ldots, s_j$, ordenados de manera creciente

☆ 31. $A(n, m) = 1 + A(n-1, m-1) + A(n-1, m); \quad n-1 \ge m \ge 1, \quad n \ge 2;$
$A(n, 0) = A(n, n) = 1, \quad n \ge 0$

**32.** Muestre que

$$f_n \ge \left(\frac{1+\sqrt{5}}{2}\right)^{n-1}, \qquad n \ge 1,$$

donde $f$ denota la sucesión de Fibonacci.

33. La ecuación

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + f(n) \tag{5.2.20}$$

es una relación de **recurrencia lineal no homogénea de segundo orden con coeficientes constantes.**

Sea $g(n)$ una solución de (5.2.20). Muestre que cualquier solución $U$ de (5.2.20) es de la forma

$$U_n = V_n + g(n), \tag{5.2.21}$$

donde $V$ es una solución de la ecuación homogénea (5.2.13).

Si $f(n) = C$ en (5.2.20), se puede mostrar que $g(n) = C'$ en (5.2.21). Además, si $f(n) = Cn$, $g(n) = C_1'n + C_0'$; si $f(n) = Cn^2$, $g(n) = C_2'n^2 + C_1'n + C_0'$; y si $f(n) = C^n$, $g(n) = C'C^n$. Utilice estos hechos y el ejercicio 33 para determinar las soluciones generales de las relaciones de recurrencia de los ejercicios 34-39.

**34.** $a_n = 6a_{n-1} - 8a_{n-2} + 3$
35. $a_n = 7a_{n-1} - 10a_{n-2} + 16n$
36. $a_n = 2a_{n-1} + 8a_{n-2} + 81n^2$
**37.** $2a_n = 7a_{n-1} - 3a_{n-2} + 2^n$
38. $a_n = -8a_{n-1} - 16a_{n-2} + 3n$
39. $9a_n = 6a_{n-1} - a_{n-2} + 5n^2$
**40.** La ecuación

$$a_n = f(n)a_{n-1} + g(n)a_{n-2} \tag{5.2.22}$$

es una **relación de recurrencia lineal homogénea de segundo orden**. Los coeficientes $f(n)$ y $g(n)$ no necesariamente son constantes. Muestre que si $S$ y $T$ son soluciones de (5.2.22), entonces $bS + dT$ también es solución de (5.2.22).

41. Suponga que ambas raíces de

$$t^2 - c_1 t - c^2 = 0$$

son iguales a $r$ y supongamos que $a_n$ satisface

$$a_n = c_1 a_{n-1} + c_2 a_{n-2}, \quad a_0 = C_0, \quad a_1 = C_1.$$



Muestre que existen constantes $b$ y $d$ tales que

$$a_n = br^n + dnr^n, \quad n = 0, 1, \ldots,$$

lo cual completa la demostración del teorema 5.2.14.

42. Sea $a_n$ el número mínimo de enlaces necesarios para resolver el problema de comunicación de $n$ nodos (véase el ejercicio 48 de la sección 5.1). Utilice la iteración para mostrar que $a_n \le 2n - 4$, $n \ge 4$.

El juego de la Torre de Hanoi con cuatro postes y $n$ discos tiene las mismas reglas que el juego con tres postes; la única diferencia es que existe un poste más. Los ejercicios 43-46 se refieren al siguiente algoritmo para resolver el juego de la Torre de Hanoi con cuatro postes y $n$ discos.

Suponga que los postes están numerados 1, 2, 3, 4 y que el problema consiste en mover los discos, que inicialmente están apilados en el poste 1, hasta el poste 4. Si $n = 1$, se mueve el disco al poste 4 y se concluye. Si $n > 1$, sea $k_n$ el máximo entero que satisface

$$\sum_{i=1}^{k_n} i \le n.$$

Fijemos $k_n$ en la parte inferior del poste 1. Se llama este algoritmo en forma recursiva para mover los $n - k_n$ discos en la parte superior del poste 1 al poste 2. Durante esta parte del algoritmo, los $k_n$ discos inferiores en el poste 1 permanecen fijos. A continuación, se mueven los $k_n$ discos del poste 1 al poste 4 llamando al algoritmo óptimo del caso de tres postes (véase el ejemplo 5.1.8) utilizando sólo los postes 1, 3 y 4. Por último, de nuevo se llama recursivamente a este algoritmo para mover los $n - k_n$ discos del poste 2 al poste 4. Durante esta parte del algoritmo, los $k_n$ discos del poste 4 permanecen fijos. Sea $T(n)$ el número de movimientos necesarios para este algoritmo.

Este algoritmo, aunque se sabe que no es óptimo, utiliza la menor cantidad de movimientos entre todos los algoritmos propuestos para el problema de las cuatro espigas.

43. Deduzca la relación de recurrencia

$$T(n) = 2T(n - k_n) + 2^{k_n} - 1.$$

44. Calcule $T(n)$ para $n = 1, \ldots, 10$. Compare estos valores con el número óptimo de movimientos necesarios para resolver el problema de los tres postes.

☆ 45. Sea

$$r_n = n - \frac{k_n(k_n+1)}{2}.$$

Utilice inducción o algún otro método para demostrar que

$$T(n) = (k_n + r_n - 1)2^{k_n} + 1.$$

★ **46.** Muestre que $T(n) = O(4^{\sqrt{n}})$.

# RINCÓN DE SOLUCIÓN DE PROBLEMAS: RELACIONES DE RECURRENCIA

**Problema**

(a) En una comida, $n$ personas dejan su abrigo en el guardarropa. Al salir, los abrigos se regresan al azar; por desgracia, nadie recibe el abrigo correcto. Sea $D_n$ el número de formas en que $n$ personas pueden recibir todas los abrigos equivocados. Muestre 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 de la parte (a) realizando la sustitución $C_n = D_n - nD_{n-1}$.

**Para enfrentar el problema**

Antes de enfrentar el problema, veamos qué es lo necesario en la parte (a). Para demostrar la relación de recurrencia, debemos reducir el problema de los abrigos incorrectos para $n$ personas a los problemas de los abrigos incorrectos para $(n-1)$ y $(n-2)$ personas (pues en la fórmula, $D_n$ está dado en términos de $D_{n-1}$ y $D_{n-2}$). Así, al revisar de manera sistemática algunos ejemplos, debemos intentar ver la forma en que el caso para $n$ personas se relaciona con los casos para $n-1$ y $n-2$ personas. La situación es similar a la de la inducción matemática y los algoritmos recursivos, en donde una instancia dada de un problema se relaciona con instancias menores del mismo problema.

Ahora veamos algunos ejemplos. El mínimo caso es $n = 1$. Una única persona debe recibir el abrigo correcto, de modo que $D_1 = 0$. Para $n = 2$, sólo hay una forma en que todas reciban el abrigo equivocado: la persona 1 recibe el abrigo 2 y la persona 2 recibe el abrigo 1. Así, $D_2 = 1$. Antes de continuar, desarrollaremos algo de notación para la distribución de abrigos. Una notación elegida en forma cuidadosa puede ayudar a resolver un problema.

Escribiremos

$$c_1, \quad c_2, \ldots, \quad c_n$$

para indicar que la persona 1 recibió el abrigo $c_1$, que la persona 2 recibió el abrigo $c_2$, y así sucesivamente. La única forma de que dos personas obtengan los abrigos incorrectos se denota 2, 1.

Si $n = 3$, la persona 1 puede recibir el abrigo 2 o 3, de modo que las posibilidades son 2, ?, ? y 3, ?, ?. Veamos ahora el caso de los números faltantes. Supongamos que la persona 1 recibe el abrigo 2. La persona 2 no puede recibir el abrigo 1 (pues entonces la persona 3 recibiría el abrigo correcto); así, la persona 2 recibe el abrigo 3. Esto deja el abrigo 1 para la persona 3. Así, si la persona 1 recibe el abrigo 2, la única posibilidad es

$$2, 3, 1.$$

Supongamos que la persona 1 recibe el abrigo 3. La persona 3 no puede recibir el abrigo 1 (ya que entonces la persona 2 recibiría el abrigo correcto); así, la persona 3 recibe el abrigo 2. Esto deja el abrigo 1 para la persona 2. Así, si la persona 1 recibe el abrigo 3, la única posibilidad es

$$3, 1, 2.$$

Así, $D_3 = 2$.



Verifiquemos si la relación de recurrencia es válida para $n = 3$:

$$D_3 = 2 = 2(1 + 0) = (3-1)(D_2 + D_1).$$

Si $n = 4$, la persona 1 recibe el abrigo 2, 3 o 4, de modo que las posibilidades son 2, ?, ?, ?; 3, ?, ?, ?; y 4, ?, ?, ?. (Si hay $n$ personas, la persona 1 recibe el abrigo 2 o 3 o … o $n-1$. Estas $n-1$ posibilidades implican la presencia del factor $n-1$ en la relación de recurrencia.) Veamos cuáles son los números faltantes. Supongamos que la persona 1 recibe el abrigo 2. Si la persona 2 recibe el abrigo 1, entonces la persona 3 recibe el abrigo 4 y la persona 4 recibe el abrigo 3, lo cual da 2, 1, 4, 3. Si la persona 2 no recibe el abrigo 1, las posibilidades son 2, 3, 4, 1 y 2, 4, 1, 3. Así, si la persona 1 recibe el abrigo 2, existen tres posibilidades. De manera análoga, si la persona 1 recibe el abrigo 3, existen tres posibilidades, y si la persona 1 recibe el abrigo 4, existen tres posibilidades. Usted debe enumerar las posibilidades para confirmar esta última afirmación. Así, $D_4 = 9$.

Verifiquemos que la relación de recurrencia también es válida para $n = 4$:

$$D_4 = 9 = 3(2 + 1) = (4-1)(D_3 + D_2).$$

Antes de continuar, el lector debe analizar el caso $n = 5$. Enumere sólo las posibilidades cuando la persona 1 reciba el abrigo 2. (Existen demasiadas posibilidades como para enumerarlas.) También verifique la relación de recurrencia para $n = 5$.

Observe que si la persona 1 recibe el abrigo 2 y la persona 2 recibe el abrigo 1 (es decir, si las personas 1 y 2 intercambian sus abrigos), el número de formas en que las demás personas reciben el abrigo equivocado es $D_{n-2}$ (las restantes $n-2$ personas deben recibir abrigos equivocados). Por esto aparece $D_{n-2}$ en la relación de recurrencia. Tendremos una solución, siempre que el término $D_{n-1}$ aparezca en el caso restante: cuando la persona 1 recibe el abrigo 2, pero la persona 2 no recibe el abrigo 1 (es decir, si las personas 1 y 2 no intercambian sus abrigos).

**Determinación de una solución**

Supongamos que existen $n$ personas. Resumiremos el argumento que hemos desarrollado a través de los ejemplos. La persona 1 puede recibir el abrigo 2, o 3, …, o $n$; así, existen $n-1$ formas posibles de que la persona 1 reciba el abrigo equivocado. Supongamos que la persona 1 recibe el abrigo 2. Existen dos posibilidades: la persona 2 recibe el abrigo 1 o no recibe el abrigo 1. Si la persona 2 recibe el abrigo 1, el número de formas en que las demás personas reciben el abrigo equivocado es $D_{n-2}$. El caso restante es que la persona 2 no reciba el abrigo 1.

Escribamos con cuidado lo que debemos contar. Las personas $2, 3, \ldots, n$ tienen entre ellas los abrigos $1, 3, 4, \ldots, n$ (falta el abrigo 2 pues la persona 1 lo tiene). Queremos determinar el número de formas en que las personas $2, 3, \ldots, n$ recibirán el abrigo equivocado y que la persona 2 no reciba el abrigo 1. Éste es casi el problema de que $n-1$ personas no reciban los abrigos correctos. Podemos transformarlo en este problema si decimos a las personas $2, 3, \ldots, n$ que el abrigo 1 es el abrigo 2. (¡Basta coserle una etiqueta temporal!) Ahora, la persona 2 no tendrá el abrigo 1, pues la persona 2 piensa que es el abrigo 2. Como existen $n-1$ personas, existen $D_{n-1}$ formas en que las personas $2, 3, \ldots, n$ recibirán el abrigo equivocado y la persona 2 no recibirá el abrigo 1. Esto implica que existen $D_{n-1} + D_{n-2}$ formas en que la persona 1 recibirá el abrigo 2 de modo que las demás personas tengan los abrigos equivocados. Como existen $n-1$ formas posibles en que la persona 1 obtenga el abrigo equivocado, obtenemos la relación de recurrencia deseada.

La relación de recurrencia define a $D_n$ en términos de $D_{n-1}$ y $D_{n-2}$, de modo que no se puede resolver mediante iteración. Además, la relación de recurrencia no tiene coeficientes constantes (aunque es lineal), de modo que no puede resolverse mediante el teorema 5.2.11 o 5.2.14. Esto explica la necesidad de realizar la sustitución en la parte (b). Es claro que después de hacer la sustitución, podemos resolver la relación de recurrencia en términos de $C_n$ mediante los métodos de la sección 5.2.

Si desarrollamos

$$D_n = (n-1)(D_{n-1} + D_{n-2}),$$

obtenemos

$$D_n = nD_{n-1} - D_{n-1} + (n-1)D_{n-2}.$$

Si entonces pasamos $nD_{n-1}$ al lado izquierdo de la ecuación (para obtener una expresión igual a $C_n$), obtenemos

$$D_n - nD_{n-1} = -D_{n-1} + (n-1)D_{n-2}.$$

Ahora, el lado izquierdo de la ecuación es igual a $C_n$ y el lado derecho es igual a $-C_{n-1}$. Así, obtenemos la relación de recurrencia

$$C_n = -C_{n-1}.$$

Esta ecuación se puede resolver mediante iteración.

**Solución formal**

Parte (a): Supongamos que las $n$ personas tienen los abrigos equivocados. Consideremos el abrigo que tiene una persona $p$. Supongamos que $p$ tiene el abrigo de $q$. Consideremos dos casos: $q$ tiene el abrigo de $p$ y $q$ no tiene el abrigo de $p$.

Existen $D_{n-2}$ distribuciones en las que $q$ tiene el abrigo de $p$ dado que los $n-2$ abrigos restantes están en posesión de las restantes $n-2$ personas, pero cada una tiene el abrigo equivocado.

Demostramos que hay $D_{n-1}$ distribuciones en las cuales $q$ no tiene el abrigo de $p$. Observe que el conjunto de abrigos $C$ que tienen las $n-1$ personas (excluyendo a $p$) incluye a todos los abrigos menos el de $q$ (pues $p$ lo tiene). Asignemos por un momento la propiedad del abrigo de $p$ a $q$. Entonces, cualquier distribución de $C$ entre las $n-1$ personas en las que nadie tiene su propio abrigo proporciona una distribución en la cual $q$ no tiene el abrigo que realmente es de $p$. Como existen $D_{n-1}$ de tales distribuciones, existen $D_{n-1}$ distribuciones en las que $q$ no tiene el abrigo de $p$.

Esto implica que existen $D_{n-1} + D_{n-2}$ distribuciones en las que $p$ tiene el abrigo de $q$. Como $p$ puede tener cualquiera de $n-1$ abrigos, obtenemos la relación de recurrencia.

Parte (b): Al hacer la sustitución dada, obtenemos

$$C_n = -C_{n-1}.$$

Al utilizar iteración obtenemos

$$\begin{aligned} C_n &= (-1)^1 C_{n-1} = (-1)^2 C_{n-2} = \cdots \\ &= (-1)^{n-2} C_2 = (-1)^n C_2 = (-1)^n (D_2 - 2D_1) = (-1)^n. \end{aligned}$$



Por tanto,

$$D_n - nD_{n-1} = (-1)^n.$$

Al resolver esta última relación de recurrencia mediante iteración, obtenemos

$$\begin{aligned} D_n &= (-1)^n + nD_{n-1} \\ &= (-1)^n + n[(-1)^{n-1} + (n-1)D_{n-2}] \\ &= (-1)^n + n(-1)^{n-1} + n(n-1)[(-1)^{n-2} + (n-2)D_{n-3}] \\ &\vdots \\ &= (-1)^n + n(-1)^{n-1} + n(n-1)(-1)^{n-2} + \cdots \\ &\quad - [n(n-1)\cdots 4] + [n(n-1)\cdots 3]. \end{aligned}$$

**Resumen de técnicas para resolver problemas**

- Cuando los ejemplos tienen un lenguaje complicado, desarrolle una notación para describirlos en forma concisa. La elección cuidadosa de una notación puede ayudar en gran medida a resolver un problema.
- Al analizar los ejemplos, intente ver la forma en que el problema en cuestión se relaciona con instancias menores del mismo problema.
- Con frecuencia es útil escribir con cuidado aquello que se debe contar.
- A veces es posible convertir una relación de recurrencia que no es una ecuación lineal homogénea con coeficientes constantes en una ecuación lineal homogénea con coeficientes constantes. Dicha relación de recurrencia se puede resolver entonces mediante los métodos de la sección 5.2.

**Comentario**

El nombre técnico de una permutación en la que ningún elemento queda en su posición original es un **desordenamiento**.

## 5.3 *APLICACIONES AL ANÁLISIS DE ALGORITMOS*

En esta sección utilizamos las relaciones de recurrencia para analizar el tiempo necesario utilizado por los algoritmos. La técnica consiste en desarrollar una relación de recurrencia y las condiciones iniciales que definan una sucesión $a_1, a_2, \ldots$, donde $a_n$ es el tiempo (en el mejor de los casos, en el caso promedio o en el peor de los casos) necesario para que un algoritmo procese una entrada de tamaño $n$. Al resolver la relación de recurrencia, podemos determinar el tiempo necesario utilizado por el algoritmo.

Nuestro primer algoritmo es una versión del algoritmo de ordenamiento por selección. Este algoritmo selecciona el elemento máximo y lo coloca al final, para luego repetir este proceso de manera recursiva.

**ALGORITMO 5.3.1** *Ordenamiento por selección*

Este algoritmo ordena la sucesión

$$s_1, \quad s_2, \quad \ldots, \quad s_n$$

entonces $U$ es una solución de (5.2.9).

Para satisfacer las condiciones iniciales (5.2.10), debemos tener

$$7 = U_0 = b2^0 + d3^0 = b + d, \qquad 16 = U_1 = b2^1 + d3^1 = 2b + 3d.$$

Al resolver estas ecuaciones en términos de $b$ y $d$, obtenemos

$$b = 5, \qquad d = 2.$$

Por tanto, la sucesión $U$ definida como

$$U_n = 5 \cdot 2^n + 2 \cdot 3^n$$

satisface la relación de recurrencia (5.2.9) y las condiciones iniciales (5.2.10). Concluimos que

$$a_n = U_n = 5 \cdot 2^n + 2 \cdot 3^n, \qquad \text{para } n = 0, 1, \ldots.$$

Ahora haremos un resumen y justificaremos las técnicas utilizadas para resolver la relación de recurrencia anterior.

**TEOREMA 5.2.11**

*Sea*

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} \tag{5.2.13}$$

*una relación de recurrencia lineal homogénea de segundo orden con coeficientes constantes.*

*Si $S$ y $T$ son soluciones de* (5.2.13), *entonces $U = bS + dT$ también es solución de* (5.2.13).

*Si $r$ es una raíz de*

$$t^2 - c_1 t - c_2 = 0, \tag{5.2.14}$$

*entonces la sucesión $r^n$, $n = 0, 1, \ldots$, es una solución de* (5.2.13).

*Si $a$ es la sucesión definida mediante* (5.2.13),

$$a_0 = C_0, \qquad a_1 = C_1, \tag{5.2.15}$$

*y $r_1$ y $r_2$ son raíces de (5.2.14) 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 (5.2.13),

$$S_n = c_1 S_{n-1} + c_2 S_{n-2}, \qquad T_n = c_1 T_{n-1} + c_2 T_{n-2}.$$

Si multiplicamos la primera ecuación por $b$ y la segunda por $d$ y sumamos, obtenemos

$$\begin{aligned} U_n = bS_n + dT_n &= c_1(bS_{n-1} + dT_{n-1}) + c_2(bS_{n-2} + dT_{n-2}) \\ &= c_1 U_{n-1} + c_2 U_{n-2}. \end{aligned}$$

Por tanto, $U$ es una solución de (5.2.13).



Como $r$ es una raíz de (5.2.14),

$$r^2 = c_1 r + c_2.$$

Ahora,

$$c_1 r^{n-1} + c_2 r^{n-2} = r^{n-2}(c_1 r + c_2) = r^{n-2} r^2 = r^n;$$

de modo que la sucesión $r^n$, $n = 0, 1, \ldots$, es una solución de (5.2.13).

Si hacemos $U_n = br_1^n + dr_2^n$, entonces $U$ es una solución de (5.2.13). Para cumplir con las condiciones iniciales (5.2.15), debemos tener

$$U_0 = b + d = C_0, \qquad U_1 = br_1 + dr_2 = C_1.$$

Si multiplicamos la primera ecuación por $r_1$ y restamos, obtenemos

$$d(r_1 - r_2) = r_1 C_0 - C_1.$$

Como $r_1 - r_2 \neq 0$, podemos despejar $d$. De manera análoga, podemos determinar $b$. Con estas elecciones de $b$ y $d$, tenemos

$$U_0 = C_0, \qquad U_1 = C_1.$$

Sea $a$ la sucesión definida mediante (5.2.13) y (5.2.15). Como $U$ también satisface (5.2.13) y (5.2.15), esto implica que $U_n = a_n$, $n = 0, 1, \ldots$. ■

**EJEMPLO 5.2.12** ***Más sobre crecimiento de poblaciones***

Suponga que la población de venados de Rustic County es de 200 en el instante $n = 0$ y de 220 en el instante $n = 1$ y que el incremento del instante $n - 1$ al instante $n$ es el doble del incremento del instante $n - 2$ al instante $n - 1$. Escribir una relación de recurrencia y una condición inicial que definan la población de venados en el instante $n$ y luego resolver la relación de recurrencia.

Sea $d_n$ la población de venados en el instante $n$. Tenemos las condiciones iniciales

$$d_0 = 200, \qquad d_1 = 220.$$

El incremento del instante $n - 1$ al instante $n$ es $d_n - d_{n-1}$ y el incremento del instante $n - 2$ al instante $n - 1$ es $d_{n-1} - d_{n-2}$. Así, obtenemos la relación de recurrencia

$$d_n - d_{n-1} = 2(d_{n-1} - d_{n-2}),$$

la cual se puede escribir como

$$d_n = 3d_{n-1} - 2d_{n-2}.$$

Para resolver esta relación de recurrencia, primero resolvemos 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 con las condiciones iniciales, debemos tener

$$200 = d_0 = b + c, \qquad 220 = d_1 = b + 2c.$$

Al despejar $b$ y $c$, tenemos que $b = 180$ y $c = 20$. Así, $d_n$ está dada por

$$d_n = 180 + 20 \cdot 2^n.$$

Como en el ejemplo 5.2.3, el crecimiento es exponencial. □

**EJEMPLO 5.2.13**

Determine una fórmula explícita para la sucesión de Fibonacci.

La sucesión de Fibonacci se define mediante la relación de recurrencia lineal homogénea de segundo orden

$$f_n - f_{n-1} - f_{n-2} = 0, \quad n \geq 3,$$

y las condiciones iniciales

$$f_1 = 1, \quad f_2 = 2.$$

Primero utilizamos la fórmula cuadrática para resolver

$$t^2 - t - 1 = 0.$$

Las soluciones son

$$t = \frac{1 \pm \sqrt{5}}{2}.$$

Así, 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, debemos tener

$$b\left(\frac{1+\sqrt{5}}{2}\right) + d\left(\frac{1-\sqrt{5}}{2}\right) = 1$$

$$b\left(\frac{1+\sqrt{5}}{2}\right)^2 + d\left(\frac{1-\sqrt{5}}{2}\right)^2 = 2.$$

Al resolver estas ecuaciones en términos de $b$ y $d$, obtenemos

$$b = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right), \quad d = -\frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right).$$

Por tanto, una fórmula explícita para la sucesión de Fibonacci es

$$f_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^{n+1}$$

Es sorprendente que aunque $f_n$ sea un entero, la fórmula anterior implica el uso del número irracional $\sqrt{5}$. □

El teorema 5.2.11 establece que cualquier solución de (5.2.13) puede darse en términos de dos soluciones básicas $r_1^n$ y $r_2^n$. Sin embargo, si (5.2.14) tiene dos raíces iguales $r$, sólo obtenemos una solución básica $r^n$. El siguiente teorema muestra que en este caso, $nr^n$ proporciona la otra solución básica.



**TEOREMA 5.2.14**

*Sea*

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} \tag{5.2.16}$$

*una relación de recurrencia lineal homogénea de segundo orden con coeficientes constantes.*

*Sea a la sucesión que satisface* (5.2.16) *y*

$$a_0 = C_0, \quad a_1 = C_1.$$

*Si ambas raíces de*

$$t^2 - c_1 t - c_2 = 0 \tag{5.2.17}$$

*son iguales a r, entonces existen constantes b y d tales que*

$$a_n = br^n + dnr^n, \quad n = 0, 1, \ldots.$$

**Demostración.** La demostración del teorema 5.2.11 muestra que la sucesión $r^n$, $n = 0, 1, \ldots$, es una solución de (5.2.16). Mostraremos que la sucesión $nr^n$, $n = 0, 1, \ldots$, también es una solución de (5.2.16).

Como $r$ es la única solución de (5.2.17), debemos tener

$$t^2 - c_1 t - c_2 = (t - r)^2.$$

Esto implica que

$$c_1 = 2r, \quad c_2 = -r^2.$$

Ahora,

$$\begin{aligned} c_1[(n-1)r^{n-1}] + c_2[(n-2)r^{n-2}] &= 2r(n-1)r^{n-1} - r^2(n-2)r^{n-2} \\ &= r^n[2(n-1) - (n-2)] = nr^n. \end{aligned}$$

Por tanto, la sucesión $nr^n$, $n = 0, 1, \ldots$, es una solución de (5.2.16).

Por el teorema 5.2.11, la sucesión $U$ definida como $U_n = br^n + dnr^n$ es una solución de (5.2.16).

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 5.2.11 y se deja como ejercicio (ejercicio 41). Esto implica que $U_n = a_n$, $n = 0, 1, \ldots$. ■

**EJEMPLO 5.2.15**

Resuelva la relación de recurrencia

$$d_n = 4(d_{n-1} - d_{n-2}) \tag{5.2.18}$$

sujeta a las condiciones iniciales

$$d_0 = 1 = d_1.$$

De acuerdo con el teorema 5.2.11, $S_n = r^n$ es una solución de (5.2.18), donde $r$ es una solución de

$$t^2 - 4t + 4 = 0. \tag{5.2.19}$$

Así, obtenemos la solución

$$S_n = 2n$$

de (5.2.18). Como 2 es la única solución de (5.2.19), por el teorema 5.2.14,

$$T_n = n2^n$$

también es una solución de (5.2.18). Así, la solución general de (5.2.18) es de la forma

$$U = aS + bT.$$

Debemos tener

$$U_0 = 1 = U_1.$$

Estas últimas ecuaciones se convierten en

$$aS_0 + bT_0 = a + 0b = 1, \quad aS_1 + bT_1 = 2a + 2b = 1.$$

Al despejar $a$ y $b$, obtenemos

$$a = 1, \quad b = -\tfrac{1}{2}.$$

Por tanto, la solución de (5.2.18) es

$$d_n = 2^n - n2^{n-1}.$$

Para la relación de recurrencia general lineal homogénea de orden $k$ con coeficientes constantes (5.2.5), si $r$ es una raíz de

$$t^k - c_1t^{k-1} - c_2t^{k-2} - \cdots - c_k = 0$$

de multiplicidad $m$, se puede mostrar que

$$r^n, \quad nr^n, \quad \ldots, \quad n^{m-1}r^n$$

son soluciones de (5.2.5). Este hecho se puede utilizar, al igual que en los ejemplos anteriores para las relaciones de recurrencia de orden 2, para resolver una relación de recurrencia lineal homogénea de orden $k$ con coeficientes constantes. Para un enunciado preciso y una demostración del resultado general, véase [Brualdi].

## Ejercicios

Indique si cada relación de recurrencia en los ejercicios 1-10 es una relación de recurrencia lineal homogénea con coeficientes constantes. Proporcione el orden de cada relación de recurrencia lineal homogénea con coeficientes constantes.

**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 11-25, resuelva la relación de recurrencia dada para las condiciones iniciales dadas.

**11.** Ejercicio 1; $a_0 = 2$
12. Ejercicio 2; $a_0 = 1$
13. Ejercicio 4; $a_0 = 0$
**14.** $a_n = 6a_{n-1} - 8a_{n-2}$; $\quad a_0 = 1, \quad a_1 = 0$
15. $a_n = 7a_{n-1} - 10a_{n-2}$; $\quad a_0 = 5, \quad a_1 = 16$
16. $a_n = 2a_{n-1} + 8a_{n-2}$; $\quad a_0 = 4, \quad a_1 = 10$
**17.** $2a_n = 7a_{n-1} - 3a_{n-2}$; $\quad a_0 = a_1 = 1$
18. Ejercicio 6; $a_0 = 0$
19. Ejercicio 8; $a_0 = a_1 = 1$
**20.** $a_n = -8a_{n-1} - 16a_{n-2}$; $\quad a_0 = 2, \quad a_1 = -20$
21. $9a_n = 6a_{n-1} - a_{n-2}$; $\quad a_0 = 6, \quad a_1 = 5$
22. La sucesión de Lucas:

$$L_n = L_{n-1} + L_{n-2}, \quad n \geq 3; \quad L_1 = 1, \quad L_2 = 3$$

**23.** Ejercicio 50, sección 5.1
24. Ejercicio 52, sección 5.1
25. La relación de recurrencia anterior al ejercicio 53, sección 5.1
**26.** Suponga que la población de venados de Rustic County es 0 en el instante $n = 0$. Supongamos que en el instante $n$, se llevan $100n$ venados a Rustic County y que la población crece 20% cada año. Escriba una relación de recurrencia y una condición inicial que defina la población de venados en el instante $n$ y luego resuelva la relación de recurrencia. La siguiente fórmula puede ser útil:

$$\sum_{i=1}^{n-1} ix^{i-1} = \frac{(n-1)x^n - nx^{n-1} + 1}{(x-1)^2}.$$

En ciertas ocasiones, una relación de recurrencia que no es una ecuación lineal homogénea con coeficientes constantes se puede transformar en una ecuación lineal homogénea con coeficientes constantes. En los ejercicios 27 y 28, realice la sustitución dada y resuelva la relación de recurrencia resultante, y luego determine la solución de la relación de recurrencia original.

27. Resuelva la relación de recurrencia

$$\sqrt{a_n} = \sqrt{a_{n-1}} + 2\sqrt{a_{n-2}}$$

con condiciones iniciales $a_0 = a_1 = 1$ realizando la sustitución $b_n = \sqrt{a_n}$.

28. Resuelva la relación de recurrencia

$$a_n = \sqrt{\frac{a_{n-2}}{a_{n-1}}}$$

con condiciones iniciales $a_0 = 8$, $a_1 = 1/(2\sqrt{2})$ calculando el logaritmo de ambos lados y realizando la sustitución $b_n = \lg a_n$.

En los ejercicios 29-31, resuelva la relación de recurrencia para las condiciones iniciales dadas.

**29.** $a_n = -2na_{n-1} + 3n(n-1)a_{n-2}$; $\quad a_0 = 1, \quad a_1 = 2$

30. $c_n = 2 + \Sigma_{i=1}^{n-1} c_i, \quad n \geq 2; \quad c_1 = 1$

☆ 63. Muestre 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 (véase el ejercicio 75 de la sección 4.2).

**64.** Suponga que una persona invierte una suma de dinero a $r\%$ compuesto anualmente. Explique la siguiente regla: Para estimar el tiempo necesario para duplicar la inversión, divida 70 entre $r$.

65. Deduzca una relación de recurrencia para el número de multiplicaciones necesarias para evaluar un determinante $n \times n$ mediante el método de los cofactores.

Una *permutación sube/baja* es una permutación $p$ de $1, 2, \ldots, n$ que satisface

$$p(i) < p(i+1) \quad \text{para } i = 1, 3, 5, \ldots$$

y

$$p(i) > p(i+1) \quad \text{para } i = 2, 4, 6, \ldots$$

Por ejemplo, existen cinco permutaciones sube/baja de 1, 2, 3, 4:

$$1, 3, 2, 4; \quad 1, 4, 2, 3; \quad 2, 3, 1, 4; \quad 2, 4, 1, 3; \quad 3, 4, 1, 2.$$

Sea $E_n$ el número de permutaciones sube/baja de $1, 2, \ldots, n$. (Defina $E_0 = 1$.) Los números $E_0, E_1, E_2, \ldots$ son los *números de Euler.*

**66.** Enumere todas las permutaciones sube/baja de 1, 2, 3. ¿Cuál es el valor de $E_3$?

67. Enumere todas las permutaciones sube/baja de 1, 2, 3, 4, 5. ¿Cuál es el valor de $E_5$?

68. Muestre que en una permutación sube/baja de $1, 2, \ldots, n$, $n$ debe aparecer en la posición $2i$, para alguna $i$.

★ **69.** Utilice el ejercicio 68 para deducir la relación de recurrencia

$$E_n = \sum_{j=1}^{\lfloor n/2 \rfloor} C(n-1, 2j-1)E_{2j-1}E_{n-2j}.$$

☆ 70. Analice el lugar donde debe aparecer el 1 en una permutación sube/baja y deduzca la relación de recurrencia

$$E_n = \sum_{j=0}^{\lfloor (n-1)/2 \rfloor} C(n-1, 2j)E_{2j}E_{n-2j-1}.$$

☆ 71. Demuestre que

$$E_n = \frac{1}{2}\sum_{j=1}^{n-1} C(n-1, j)E_jE_{n-j-1}.$$

## 5.2 SOLUCIÓN DE RELACIONES DE RECURRENCIA

Resolver una relación de recurrencia asociada a la sucesión $a_0, a_1, \ldots$ consiste en determinar una fórmula explícita para el término general $a_n$. En esta sección analizaremos dos métodos para resolver relaciones de recurrencia: la **iteración** y un método especial que se aplica a las **relaciones de recurrencia lineales homogéneas con coeficientes constantes**. Para otros métodos más poderosos, como aquellos que utilizan funciones generatrices, el lector puede consultar [Brualdi].

Para resolver una relación de recurrencia asociada a la sucesión $a_0, a_1, \ldots$ por iteración, utilizamos la relación de recurrencia para escribir el $n$-ésimo término $a_n$ en términos de algunos de sus predecesores $a_{n-1}, \ldots, a_0$. Luego utilizamos de manera sucesiva la relación de recurrencia para reemplazar cada uno de los términos $a_{n-1}, \ldots$ por algunos de sus predecesores. Continuamos hasta obtener una fórmula explícita. Utilizamos el método iterativo para resolver la relación de recurrencia del ejemplo 5.1.3.



**EJEMPLO 5.2.1**

Podemos resolver la relación de recurrencia

$$a_n = a_{n-1} + 3, \tag{5.2.1}$$

sujeta a la condición inicial

$$a_1 = 2,$$

por iteración. Al reemplazar $n$ con $n-1$ en (5.2.1), obtenemos

$$a_{n-1} = a_{n-2} + 3.$$

Si sustituimos esta expresión para $a_{n-1}$ en (5.2.1), obtenemos

$$\begin{aligned} a_n &= \boxed{a_{n-1}} + 3 \\ &\quad\ \downarrow \\ &= \boxed{a_{n-2} + 3} + 3 \\ &= a_{n-2} + 2 \cdot 3. \end{aligned} \tag{5.2.2}$$

Al reemplazar $n$ con $n-2$ en (5.2.1), obtenemos

$$a_{n-2} = a_{n-3} + 3.$$

Si sustituimos esta expresión para $a_{n-2}$ en (5.2.2), obtenemos

$$\begin{aligned} a_n &= \boxed{a_{n-2}} + 2 \cdot 3 \\ &\quad\ \downarrow \\ &= \boxed{a_{n-3} + 3} + 2 \cdot 3 \\ &= a_{n-3} + 3 \cdot 3. \end{aligned}$$

En general, tenemos

$$a_n = a_{n-k} + k \cdot 3.$$

Si hacemos $k = n - 1$ en esta última expresión, tenemos

$$a_n = a_1 + (n-1) \cdot 3.$$

Como $a_1 = 2$, obtenemos la fórmula explícita

$$a_n = 2 + 3(n-1)$$

para la sucesión $a$. □

**EJEMPLO 5.2.2**

Podemos resolver la relación de recurrencia

$$S_n = 2S_{n-1}$$

del ejemplo 5.1.5, sujeta a la condición inicial

$$S_0 = 1,$$

por iteración:

$$S_n = 2S_{n-1} = 2(2S_{n-2}) = \cdots = 2^n S_0 = 2^n.$$

□

**EJEMPLO 5.2.3** *Crecimiento de poblaciones*

Suponga que la población de venados en Rustic County es 1000 en el instante $n = 0$ y que el incremento desde el instante $n - 1$ hasta el instante $n$ es 10% del tamaño en el instante $n - 1$. Escriba una relación de recurrencia y una condición inicial que defina la población de venados en el instante $n$, y luego resuelva la relación de recurrencia.

Sea $d_n$ la población de venados en el instante $n$. Tenemos la condición inicial

$$d_0 = 1000.$$

El incremento del instante $n - 1$ al instante $n$ es $d_n - d_{n-1}$. Como este incremento es igual a 10% del tamaño en el instante $n - 1$, obtenemos la relación de recurrencia

$$d_n - d_{n-1} = 0.1d_{n-1},$$

que se puede escribir como

$$d_n = 1.1d_{n-1}.$$

La relación de recurrencia se puede resolver por iteración:

$$\begin{aligned} d_n &= 1.1d_{n-1} = 1.1(1.1d_{n-2}) = (1.1)^2(d_{n-2}) \\ &= \cdots = (1.1)^n d_0 = (1.1)^n 1000. \end{aligned}$$

La hipótesis implica un crecimiento exponencial de la población.

□

**EJEMPLO 5.2.4**

Determine una fórmula explícita para $c_n$, el número mínimo de movimientos en que puede resolverse el juego de la Torre de Hanoi con $n$ discos (véase el ejemplo 5.1.8).

En el ejemplo 5.1.8 obtuvimos la relación de recurrencia

$$c_n = 2c_{n-1} + 1 \tag{5.2.3}$$

y la condición inicial

$$c_1 = 1.$$



Al aplicar el método iterativo a (5.2.3), obtenemos

$$\begin{aligned} c_n &= 2c_{n-1} + 1 \\ &= 2(2c_{n-2} + 1) + 1 \\ &= 2^2c_{n-2} + 2 + 1 \\ &= 2^2(2c_{n-3} + 1) + 2 + 1 \\ &= 2^3c_{n-3} + 2^2 + 2 + 1 \\ &\vdots \\ &= 2^{n-1}c_1 + 2^{n-2} + 2^{n-3} + \cdots + 2 + 1 \\ &= 2^{n-1} + 2^{n-2} + 2^{n-3} + \cdots + 2 + 1 \\ &= 2^n - 1. \end{aligned}$$

El último paso surge de la fórmula para la suma geométrica (véase el ejemplo 1.6.2). □

**EJEMPLO 5.2.5**

Podemos 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 5.1.9 por iteración. Para que la notación sea más sencilla, hacemos $s = -b/k$.

$$\begin{aligned} p_n &= a + sp_{n-1} \\ &= a + s(a + sp_{n-2}) \\ &= a + as + s^2 p_{n-2} \\ &= a + as + s^2(a + sp_{n-3}) \\ &= a + as + as^2 + s^3 p_{n-3} \\ &\vdots \\ &= a + as + as^2 + \cdots + as^{n-1} + s^n p_0 \\ &= \frac{a - as^n}{1-s} + s^n p_0 \\ &= s^n\left(\frac{-a}{1-s} + p_0\right) + \frac{a}{1-s} \\ &= \left(-\frac{b}{k}\right)^n \left(\frac{-ak}{k+b} + p_0\right) + \frac{ak}{k+b}. \end{aligned} \tag{5.2.4}$$

Vemos que si $b/k < 1$, el término

$$\left(-\frac{b}{k}\right)^n \left(\frac{-ak}{k+b} + p_0\right)$$

es cada vez más pequeño conforme n crece, de modo que el precio tiende a estabilizarse en aproximadamente $ak/(k + b)$. Si $b/k = 1$, (5.2.4) muestra que $p_n$ oscila entre $p_0$ y $p_1$. Si $b/k > 1$, (5.2.4) muestra que las diferencias entre los precios sucesivos aumentan. Anteriormente habíamos observado estas propiedades de manera gráfica (véase el ejemplo 5.1.9). □

Ahora veremos una clase particular de relaciones de recurrencia.

**DEFINICIÓN 5.2.6**

*Una relación de recurrencia lineal homogénea de orden k con coeficientes constantes* es una relación de recurrencia de la forma

$$a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ka_{n-k}, \qquad c_k \neq 0. \tag{5.2.5}$$

Observe que una relación de recurrencia lineal homogénea de orden $k$ con coeficientes constantes (5.2.5), junto con las $k$ condiciones iniciales

$$a_0 = C_0, \quad a_1 = C_1, \ldots, \quad a_{k-1} = C_{k-1},$$

definen de manera única una sucesión $a_0, a_1, \ldots$.

**EJEMPLO 5.2.7**

Las relaciones de recurrencia

$$S_n = 2S_{n-1} \tag{5.2.6}$$

del ejemplo 5.2.2 y

$$f_n = f_{n-1} + f_{n-2}, \tag{5.2.7}$$

que define la sucesión de Fibonacci, son ambas relaciones de recurrencia lineales homogéneas con coeficientes constantes. La relación de recurrencia (5.2.6) es de orden 1 y (5.2.7) es de orden 2. □

**EJEMPLO 5.2.8**

La relación de recurrencia

$$a_n = 3a_{n-1}a_{n-2} \tag{5.2.8}$$

no es una relación de recurrencia lineal homogénea con coeficientes constantes. En una relación de recurrencia lineal homogénea con coeficientes constantes, cada término es de la forma $ca_k$. Los términos como $a_{n-1}a_{n-2}$ no están permitidos. Las relaciones de recurrencia como (5.2.8) son *no lineales*. □

**EJEMPLO 5.2.9**

La relación de recurrencia

$$a_n - a_{n-1} = 2n$$

no es una relación de recurrencia lineal homogénea con coeficientes constantes, debido a que la expresión del lado derecho de la ecuación no es igual a cero. (Tal ecuación es *no homogénea*. Las relaciones de recurrencia lineales no homogéneas con coeficientes constantes se analizan en los ejercicios 33-39.) □



**EJEMPLO 5.2.10**

La relación de recurrencia

$$a_n = 3na_{n-1}$$

no es una relación de recurrencia lineal homogénea con coeficientes constantes debido a que el coeficiente $3n$ no es constante. Es una relación de recurrencia lineal homogénea con coeficientes no constantes. □

Ilustraremos el método general de resolución de las relaciones de recurrencia lineales homogéneas con coeficientes constantes determinando una fórmula explícita para la sucesión definida mediante la relación de recurrencia

$$a_n = 5a_{n-1} - 6a_{n-2} \tag{5.2.9}$$

y condiciones iniciales

$$a_0 = 7, \qquad a_1 = 16. \tag{5.2.10}$$

En matemáticas, al intentar resolver una instancia más difícil de algún problema, con frecuencia comenzamos con una expresión que resuelve una versión más sencilla. Para la relación de recurrencia de primer orden (5.2.6), vimos en el ejemplo 5.2.2 que la solución era de la forma

$$S_n = t^n;$$

así, para nuestro primer intento de determinar una solución de la relación de recurrencia de segundo orden (5.2.9), buscaremos una solución de la forma $V_n = t^n$.

Si $V_n = t^n$ fuese solución de (5.2.9), debemos tener

$$V_n = 5V_{n-1} - 6V_{n-2}$$

o

$$t^n = 5t^{n-1} - 6t^{n-2}$$

o

$$t^n - 5t^{n-1} + 6t^{n-2} = 0.$$

Al dividir entre $t^{n-2}$, obtenemos la ecuación equivalente

$$t^2 - 5t + 6 = 0. \tag{5.2.11}$$

Al resolver (5.2.11), tenemos las soluciones

$$t = 2, \qquad t = 3.$$

En este momento, tenemos dos soluciones $S$ y $T$ de (5.2.9), dadas por

$$S_n = 2^n, \qquad T_n = 3^n. \tag{5.2.12}$$

Podemos verificar (véase el teorema 5.2.11) que si $S$ y $T$ son soluciones de (5.2.9), entonces $bS + dT$, donde $b$ y $d$ son números arbitrarios, también es una solución de (5.2.9). En nuestro caso, si definimos la sucesión $U$ mediante la ecuación

$$\begin{aligned} U_n &= bS_n + dT_n \\ &= b2^n + d3^n, \end{aligned}$$

Los cambios del precio con respecto del tiempo se pueden observar en una gráfica. Si el precio inicial es $p_0$, el fabricante estará dispuesto a ofrecer la cantidad $q_1$, en el instante $n = 1$. Localizamos esta cantidad moviéndonos de manera horizontal hasta la curva de oferta (véase la figura 5.1.5). Sin embargo, las fuerzas del mercado obligan al precio a bajar hasta $p_1$, como podemos ver al movernos de manera vertical hasta la curva de demanda. En el precio $p_1$, el fabricante estará dispuesto a ofrecer la cantidad $q_2$ en el instante $n = 2$, como podemos ver moviéndonos en forma horizontal hasta la curva de oferta. Ahora, las fuerzas del mercado obligan al precio a subir hasta $p_2$, como podemos ver al movernos en forma vertical hasta la curva de demanda. Al continuar este proceso, obtenemos la "telaraña" que aparece en la figura 5.1.5.

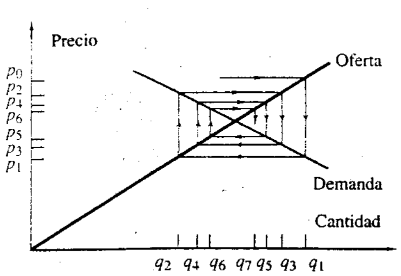



FIGURA 5.1.5 Una telaraña con un precio estabilizante.

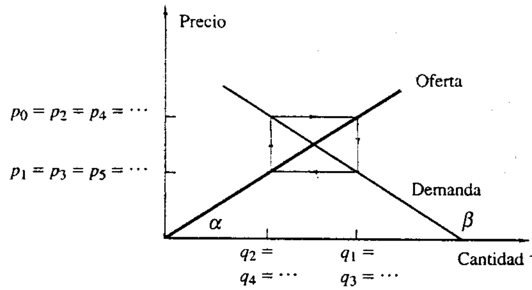



FIGURA 5.1.6 Una telaraña con un precio fluctuante.

Para las funciones de oferta y de demanda de la figura 5.1.5, el precio tiende al precio dado por la intersección de las curvas de oferta y de demanda. Sin embargo, esto no siempre ocurre. Por ejemplo, en la figura 5.1.6, el precio fluctúa entre $p_0$ y $p_1$, mientras que en la figura 5.1.7, el precio comienza a oscilar en forma cada vez más pronunciada. El comportamiento queda determinado por las pendientes de las rectas de oferta y de demanda. Para producir el comportamiento fluctuante de la figura 5.1.6, los ángulos $\alpha$ y $\beta$ deben sumar 180°. Las pendientes de las curvas de oferta y de demanda son tan $\alpha$ y tan $\beta$, respectivamente, de modo que en la figura 5.1.6 tenemos

$$k = \tan \alpha = -\tan \beta = b.$$

Hemos mostrado que el precio fluctúa entre dos valores cuando $k = b$. Un análisis similar muestra que el precio tiende al dado por la intersección de las curvas de oferta y de demanda (figura 5.1.5) cuando $b < k$ y el caso del precio oscilante y creciente (figura 5.1.7)

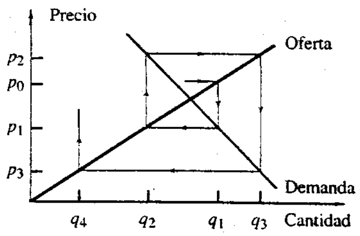



FIGURA 5.1.7 Una telaraña con oscilaciones crecientes del precio.



ocurre cuando $b > k$ (véanse los ejercicios 35 y 36). En la sección 5.2 analizaremos el comportamiento del precio con respecto del tiempo, mediante una fórmula explícita para el precio $p_n$. □

La definición de relación de recurrencia se puede ampliar para incluir funciones con índices dados por $n$-adas de enteros positivos. Nuestro último ejemplo es de esta forma.

**EJEMPLO 5.1.10** ***Función de Ackermann***

La función de Ackermann se puede definir mediante las relaciones de recurrencia

$$A(m, 0) = A(m-1, 1), \qquad m = 1, 2, \ldots, \tag{5.1.11}$$

$$A(m, n) = A(m-1, A(m, n-1)), \qquad m = 1, 2, \ldots, \quad n = 1, 2, \ldots, \tag{5.1.12}$$

y las condiciones iniciales

$$A(0, n) = n + 1, \qquad n = 0, 1, \ldots. \tag{5.1.13}$$

La función de Ackermann tiene importancia teórica debido a 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 unión y búsqueda (véase [Tarjan, páginas 22-29]).

El cálculo

$$\begin{aligned} A(1, 1) &= A(0, A(1, 0)) && \text{por (5.1.12)} \\ &= A(0, A(0, 1)) && \text{por (5.1.11)} \\ &= A(0, 2) && \text{por (5.1.13)} \\ &= 3 && \text{por (5.1.13)} \end{aligned}$$

ilustra el uso de las ecuaciones (5.1.11)-(5.1.13). □

## Ejercicios

En los ejercicios 1-3, determine una relación de recurrencia y las condiciones iniciales que generan 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 4-8, suponga que una persona invierte $2000 al 14% compuesto anualmente. Sea $A_n$ la cantidad al final de $n$ años.

4. Determine una relación de recurrencia para la sucesión $A_0, A_1, \ldots$.
5. Determine una condición inicial para la sucesión $A_0, A_1, \ldots$.
6. Determine $A_1, A_2$ y $A_3$.
7. Determine una fórmula explícita para $A_n$.
8. ¿Cuánto tiempo tardará una persona en duplicar la inversión inicial?

Si una persona invierte en una anualidad protegida contra impuestos, la cantidad invertida, al igual que los intereses devengados, no están sujetos a impuestos hasta ser retirados de la cuenta. En los ejercicios 9-12, suponga que una persona invierte $2000 cada año en una anualidad protegida contra impuestos, a 10% compuesto anualmente. Sea $A_n$ la cantidad al final de $n$ años.

9. Determine una relación de recurrencia para la sucesión $A_0, A_1, \ldots$.
10. Determine una condición inicial para la sucesión $A_0, A_1, \ldots$.
11. Determine $A_1, A_2$ y $A_3$.
12. Determine una fórmula explícita para $A_n$.

En los ejercicios 13-17, suponga que una persona invierte $3000 a 12% de interés anual compuesto en forma trimestral. Sea $A_n$ la cantidad al final de $n$ años.

13. Determine una relación de recurrencia para la sucesión $A_0, A_1, \ldots$.
14. Determine una condición inicial para la sucesión $A_0, A_1, \ldots$.
15. Determine $A_1, A_2$ y $A_3$.
16. Determine una fórmula explícita para $A_n$.
17. ¿Cuánto tiempo tardará una persona en duplicar la inversión inicial?
**18.** Sea $S_n$ el número de cadenas de $n$ bits que no contienen al patrón 000. Determine una relación de recurrencia y condiciones iniciales para la sucesión $\{S_n\}$.

Los ejercicios 19-21 se refieren a la sucesión $S$, donde $S_n$ denota el número de cadenas de $n$ bits que no contienen al patrón 00.

**19.** Determine una relación de recurrencia y condiciones iniciales para la sucesión $\{S_n\}$.
20. Muestre que $S_n = f_{n+1}$, $n = 1, 2, \ldots$, donde $f$ denota la sucesión de Fibonacci.
21. Considere el número de cadenas de $n$ bits con exactamente $i$ ceros y el ejercicio 20 para mostrar que

$$f_{n+1} = \sum_{i=0}^{\lfloor (n+1)/2 \rfloor} C(n+1-i,\, i), \qquad n = 1, 2, \ldots,$$

donde $f$ denota la sucesión de Fibonacci.

Los ejercicios 22-24 se refieren a la sucesión $S_1, S_2, \ldots$, donde $S_n$ denota el número de cadenas de $n$ bits que no contienen al patrón 010.

**22.** Calcule $S_1, S_2, S_3$ y $S_4$.
23. Considere el número de cadenas de $n$ bits que no contienen el patrón 010 y que no comienzan con 0; aquellas que comienzan con un único 0 (es decir, que comienzan con 01), aquellas que comienzan con 0, y así sucesivamente, para deducir la relación de recurrencia

$$S_n = S_{n-1} + S_{n-3} + S_{n-4} + S_{n-5} + \cdots + S_1 + 3. \tag{5.1.14}$$

24. Reemplace $n$ con $n-1$ en (5.1.14) y 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 deducir la relación de recurrencia

$$S_n = 2S_{n-1} - S_{n-2} + S_{n-3}.$$

En los ejercicios 25-30, $C_0, C_1, C_2, \ldots$ 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 $C_5$ utilizando la relación de recurrencia del ejemplo 5.1.7.



**26.** Muestre que los números de Catalan están dados por la relación de recurrencia

$$(n+2)\, C_{n+1} = (4n+2)\, C_n, \quad n \geq 0$$

y la condición inicial $C_0 = 1$.
27. Demuestre que

$$C_n \geq \frac{4^{n-1}}{n^2} \qquad \text{para toda } n \geq 1.$$

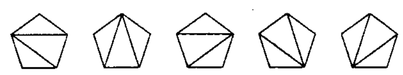
**28.** Deduzca una relación de recurrencia y una condición inicial para el número de formas de colocar paréntesis en el producto

$$a_1 * a_2 * \cdots * a_n, \quad n \geq 2.$$

*Ejemplos:* Existe una forma de colocar paréntesis en $a_1 * a_2$; a saber, $(a_1 * a_2)$. Existen dos formas de colocar paréntesis a $a_1 * a_2 * a_3$, a saber, $((a_1 * a_2) * a_3)$ y $(a_1 * (a_2 * a_3))$. Deduzca que el número de formas de colocar paréntesis en el producto de $n$ elementos es $C_{n-1}$, $n \geq 2$.

★ **29.** Éste es el problema analizado originalmente por Catalan.

Deduzca una relación de recurrencia y una condición inicial para el número de formas de dividir un polígono convexo de $(n+2)$ lados, $n \geq 1$, en triángulos, trazando $n-1$ líneas a través de los vértices y de modo que no se intersequen en el interior del polígono. (Un polígono es convexo si cualquier recta que una dos puntos del polígono está completamente dentro del polígono.) Por ejemplo, existen cinco formas de dividir un pentágono convexo en triángulos, trazando dos rectas ajenas a través de los vértices:



Deduzca que el número de formas de dividir un polígono convexo de $(n+2)$ lados en triángulos trazando $n-1$ rectas ajenas a través de los vértices es $C_n$, $n \geq 1$.
30. Considere las rutas desde la esquina inferior izquierda hasta la esquina superior derecha en una retícula $(n+1) \times (n+1)$, donde sólo se puede ir hacia la derecha o hacia arriba; separe dichas rutas en clases con base en el momento en que, después de salir de la esquina inferior izquierda, la ruta toca por vez primera la diagonal que va de la esquina inferior izquierda a la esquina superior derecha y deduzca 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 hasta la esquina superior derecha en una retícula $n \times n$, donde sólo se puede ir hacia la derecha, hacia arriba o en diagonal hacia el noreste [es decir, de $(i, j)$ hasta $(i+1, j+1)$] y en donde se permite tocar pero no rebasar la diagonal que va de la esquina inferior izquierda a la esquina superior derecha. Los números $S_0, S_1, \ldots$ se llaman *números de Schröder.*

31. Muestre que $S_0 = 1$, $S_1 = 2$, $S_2 = 6$ y $S_3 = 22$.
32. Deduzca una relación de recurrencia para la sucesión de números de Schröder.
**33.** Escriba las soluciones explícitas del juego de la Torre de Hanoi para $n = 3, 4$.
34. ¿A qué valores tienden el precio y la cantidad en el ejemplo 5.1.9 cuando $b < k$?
**35.** Muestre que cuando $b < k$ en el ejemplo 5.1.9, el precio tiende al dado por la intersección de las curvas de oferta y de demanda.
36. Muestre que cuando $b > k$ en el ejemplo 5.1.9, las diferencias entre los precios sucesivos aumentan.

Los ejercicios 37-43 se refieren a la función de Ackermann $A(m, n)$.

**37.** Calcule $A(2, 2)$ y $A(2, 3)$.
38. Utilice la inducción para mostrar que

$$A(1, n) = n + 2, \quad n = 0, 1, \ldots.$$

39. Utilice la inducción para mostrar que

$$A(2, n) = 3 + 2n, \quad n = 0, 1 \ldots.$$

**40.** Haga una conjetura acerca de una fórmula para $A(3, n)$ y demuéstrela por inducción.
☆ 41. Demuestre que $A(m, n) > n$ para toda $m \geq 0$, $n \geq 0$, por inducción sobre $m$. El paso inductivo utilizará inducción sobre $n$.
42. Utilice el ejercicio 41 o algún otro recurso, para demostrar que $A(m, n) > 1$ para toda $m \geq 1, n \geq 0$.
**43.** Utilice el ejercicio 41 o algún otro recurso, para demostrar que $A(m, n) < A(m, n + 1)$ para toda $m \geq 0, n \geq 0$.

Lo que hemos llamado función de Ackermann en realidad se deduce de la función original de Ackermann dada por

$$\begin{aligned}
AO(0, y, z) &= z + 1, & y, z &\geq 0\\
AO(1, y, z) &= y + z, & y, z &\geq 0\\
AO(2, y, z) &= yz, & y, z &\geq 0\\
AO(x + 3, y, 0) &= 1, & x, y &\geq 0\\
AO(x + 3, y, z + 1) &= AO(x + 2, y, AO(x + 3, y, z)), & x, y, z &\geq 0.
\end{aligned}$$

Los ejercicios 44-47 se refieren a la función $AO$ y a la función de Ackermann $A$.

**44.** Muestre que $A(x, y) = AO(x, 2, y + 3) - 3$ para $y \geq 0$ y $x = 0, 1, 2$.
45. Muestre que $AO(x, 2, 1) = 2$ para $x \geq 2$.
46. Muestre que $AO(x, 2, 2) = 4$ para $x \geq 2$.
★ **47.** Muestre que $A(x, y) = AO(x, 2, y + 3) - 3$ para $x, y \geq 0$.
48. Una red consta de $n$ nodos. Cada nodo tiene cierta capacidad de comunicación y de almacenamiento local. En forma periódica, hay que compartir todos los archivos. Un *enlace* consta de dos nodos que comparten archivos. En forma específica, al enlazar los nodos $A$ y $B$, $A$ transmite todos sus archivos a $B$ y viceversa. Sólo existe un enlace a la vez, y después de establecer un enlace y compartir los archivos, el enlace se elimina. Sea $a_n$ el número mínimo de enlaces necesarios para $n$ nodos, de modo que todos los archivos sean conocidos por todos los nodos.
(a) Muestre que $a_2 = 1$, $a_3 \leq 3$, $a_4 \leq 4$.
(b) Muestre que $a_n \leq a_{n-1} + 2$, $n \geq 3$.
49. Si $P_n$ denota el número de permutaciones de $n$ objetos distintos, determine una relación de recurrencia y una condición inicial para la sucesión $P_1, P_2, \ldots$.
**50.** Suponga que tenemos $n$ dólares y que cada día compramos jugo de naranja ($1), leche ($2), o cerveza ($2). Si $R_n$ es el número de formas de gastar el dinero, muestre que

$$R_n = R_{n-1} + 2R_{n-2}.$$

El orden se toma en cuenta. Por ejemplo, existen 12 formas de gastar cuatro dólares: *LC, CL, JJL, JJC, JLJ, JCJ, LJJ, CJJ, JJJJ, LL, CC*.
51. Suponga que tenemos $n$ dólares y que cada día compramos cintas ($1), papel ($1), plumas ($2), lápices ($2) o carpetas ($3). Si $R_n$ es el número de formas de gastar todo el dinero, deduzca una relación de recurrencia para la sucesión $R_1, R_2, \ldots$.
52. Sea $R_n$ el número de regiones en que queda dividido el plano mediante $n$ rectas. Suponga que cada par de rectas se interseca en un punto, pero que no existen tres rectas que se intersequen en un punto. Deduzca 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 como

$$S_1 = 0, \quad S_2 = 1, \quad S_n = \frac{S_{n-1} + S_{n-2}}{2}, \quad n = 3, 4, \ldots.$$

53. Calcule $S_3$ y $S_4$.
☆ 54. Haga una conjetura acerca de una fórmula para $S_n$ y muestre que es correcta utilizando inducción.
★ **55.** Sea $F_n$ el número de funciones $f$ de $X = \{1, \ldots, n\}$ en $X$ con la propiedad de que si $i$ está en el rango de $f$, entonces $1, 2, \ldots, i - 1$ también están en el rango de $f$. (Haga $F_0 = 1$.) Muestre 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 $\alpha$ es una cadena de bits, sea $C(\alpha)$ el máximo número de ceros consecutivos en $\alpha$. [*Ejemplos*: $C(10010) = 2$, $C(00110001) = 3$.] Sea $S_n$ el número de cadenas de $n$ bits $\alpha$ con $C(\alpha) \leq 2$. Desarrolle una relación de recurrencia para $S_1, S_2, \ldots$.
57. Deduzca una relación de recurrencia para $C(n, k)$, el número de subconjuntos con $k$ elementos de un conjunto con $n$ elementos. Específicamente, escriba $C(n + 1, k)$ en términos de $C(n, i)$ para $i$ adecuada.
**58.** Deduzca una relación de recurrencia para $S(k, n)$, el número de formas de elegir $k$ elementos, permitiendo repeticiones, de $n$ tipos disponibles. Específicamente, escriba $S(k, n)$ en términos de $S(k - 1, i)$ para $i$ adecuada.
59. Sea $S(n, k)$ el número de funciones de $\{1, \ldots, n\}$ sobre $\{1, \ldots, k\}$. Muestre que $S(n, k)$ satisface la relación de recurrencia

$$S(n,k) = k^n - \sum_{i=1}^{k-1} C(k,i)S(n,i).$$

60. La *sucesión de Lucas* $L_1, L_2, \ldots$ (la cual recibe el nombre de Édouard Lucas, el inventor del juego de la Torre de Hanoi) se define mediante la relación de recurrencia

$$L_n = L_{n-1} + L_{n-2}, \quad n \geq 3,$$

y las condiciones iniciales

$$L_1 = 1, \qquad L_2 = 3.$$

(a) Determine los valores de $L_3$, $L_4$ y $L_5$.
(b) Muestre que

$$L_{n+2} = f_n + f_{n+2}, \quad n \geq 1,$$

donde $f_1, f_2, \ldots$ denota la sucesión de Fibonacci.
**61.** Establezca la relación de recurrencia

$$s_{n+1,k} = s_{n,k-1} + ns_{n,k}$$

para los números de Stirling del primer tipo (véase el ejercicio 74 de la sección 4.2).
62. Establezca la relación de recurrencia

$$S_{n+1,k} = S_{n,k-1} + kS_{n,k}$$

para los números de Stirling del segundo tipo (véase el ejercicio 75 de la sección 4.2).