## 5.3 ➜ El algoritmo euclidiano

En la sección 5.1 se estudiaron algunos métodos para calcular el máximo común divisor de dos enteros que resultaron ineficientes. El **algoritmo euclidiano** es un algoritmo antiguo, conocido y *eficiente* para encontrar el máximo común divisor de dos enteros.

El algoritmo euclidiano se basa en el hecho de que si $r = a \bmod b$, entonces

$$\text{mcd}(a, b) = \text{mcd}(b, r). \qquad \textbf{(5.3.1)}$$

Antes de probar (5.3.1), se ilustra cómo usa esta ecuación el algoritmo euclidiano para encontrar el máximo común divisor.

**Ejemplo 5.3.1 ▶**

Como 105 mod 30 = 15, por (5.3.1)

$$\text{mcd}(105, 30) = \text{mcd}(30,15).$$

Como 30 mod 15 = 0, por (5.3.1)

$$\text{mcd}(30, 15) = \text{mcd}(15, 0).$$

Por inspección, mcd(15, 0) = 15. Por lo tanto,

$$\text{mcd}(105, 30) = \text{mcd}(30, 15) = \text{mcd}(15, 0) = 15.$$ ◀

Ahora se demostrará la ecuación (5.3.1).

**Teorema 5.3.2**

*Si a es un entero no negativo, b es un entero positivo y* $r = a \bmod b$*, entonces*

$$\text{mcd}(a, b) = \text{mcd}(b, r).$$

***Demostración*** Por el teorema del cociente-residuo, existen $q$ y $r$ que satisfacen

$$a = bq + r, \qquad \text{o} \le r < b.$$

Se demuestra que el conjunto de divisores comunes de $a$ y $b$ es igual al conjunto de divisores comunes de $b$ y $r$, lo que prueba el teorema.

Sea $c$ un divisor común de $a$ y $b$. Por el Teorema 5.1.3(c), $c \mid bq$. Como $c \mid a$ y $c \mid bq$, por el Teorema 5.1.3(b), $c \mid a - bq$ $(= r)$. Entonces $c$ es un divisor común de $b$ y $r$. Inversamente, si $c$ es un divisor común de $b$ y $r$, entonces, $c \mid bq$ y $c \mid bq + r$ $(= a)$ y $c$ es un divisor común de $a$ y $b$. Así, el conjunto de divisores comunes de $a$ y $b$ es igual al conjunto de divisores comunes de $b$ y $r$. Por lo tanto,

$$\text{mcd}(a, b) = \text{mcd}(b, r).$$

Ahora se establecerá de manera formal el algoritmo euclidiano como el algoritmo 5.3.3.

**Algoritmo 5.3.3**

**Algoritmo euclidiano**

Este algoritmo encuentra el máximo común divisor de los enteros no negativos $a$ y $b$, donde no son cero $a$ y $b$.

```
        Entrada:    a y b (enteros no negativos, ambos diferentes de cero)
         Salida:    máximo común divisor de a y b
1.      mcd(a, b) {
2.          // sea a el mayor
3.          if (a < b)
4.              intercambia(a, b)
5.          while (b ¬=0) {
6.              r = a mod b
7.              a = b
8.              b = r
9.          }
10.         return a
11.     }
```

Se observa que el ciclo "while" en el algoritmo euclidiano (líneas 5 al 9) siempre termina al final del ciclo (líneas 7 y 8), los valores de $a$ y $b$ se actualizan con valores *más pequeños*. Como los enteros no negativos no pueden decrecer indefinidamente, en algún momento $b$ se convierte en cero y el ciclo termina.

Sea $G = \text{mcd}(a, b)$, donde $a$ y $b$ son los valores de entrada al algoritmo 5.3.3. Se demuestra que el algoritmo 5.3.3 es correcto verificando que $G = \text{mcd}(a, b)$ es un invariante de ciclo, donde ahora $a$ y $b$ denotan la variables en el pseudocódigo.

Por definición, la invariante de ciclo es verdadera la primera vez que se llega a la línea 5. Suponga que $G = \text{mcd}(a, b)$ es verdadera antes de la siguiente iteración del ciclo y que $b \neq 0$. El Teorema 5.3.2 nos dice que después de ejecutar la línea 6,

$$\text{mcd}(a, b) = \text{mcd}(b, r).$$

En las líneas 7 y 8, $a$ se convierte en $b$ y $b$ se convierte en $r$. Por lo tanto, $G = \text{mcd}(a, b)$ es verdadera para los nuevos valores de $a$ y $b$. Se concluye que $G = \text{mcd}(a, b)$ es una invariante del ciclo. El ciclo "while" termina cuando $b$ se hace 0. En este punto, la invariante del ciclo es $G = \text{mcd}(a, 0)$. El algoritmo regresa $a$ [$= \text{mcd}(a, 0)$]. Entonces el valor que regresa el algoritmo es $G$, que por definición es el máximo común divisor de los valores de entrada. Por lo tanto, el algoritmo 5.3.3 es correcto.

El algoritmo 5.3.3 encuentra correctamente el máximo común divisor si se omiten las líneas 3 y 4 (vea el ejercicio 13). Estas líneas se incluyen porque simplifican el análisis del algoritmo 5.3.3 en el siguiente apartado.

**Ejemplo 5.3.4 ▶**

Se mostrará cómo el algoritmo 5.3.3 encuentra el mcd(504, 396).

Sean $a = 504$ y $b = 396$. Como $a > b$, pasamos a la línea 5. Como $b \neq 0$, se procede a la línea 6, donde se hace $r$ igual a

$$a \bmod b = 504 \bmod 396 = 108.$$

Después pasamos a las líneas 7 y 8, donde $a$ se hace igual a 396 y $b$ igual a 108. Después regresamos a la línea 5.

Como $b \neq 0$, se procede a la línea 6, donde se hace $r$ igual a

$$a \bmod b = 396 \bmod 108 = 72.$$

Después nos movemos a las líneas 7 y 8, donde $a$ se hace igual a 108 y $b$ igual a 72. Se regresa a la línea 5.

Como $b \neq 0$, se procede a la línea 6, donde $r$ se iguala a

$$a \bmod b = 108 \bmod 72 = 36.$$

Ahora vamos a las líneas 7 y 8, donde $a$ se hace igual a 72 y $b$ a 36. Después regresamos a la línea 5.

Como $b \neq 0$, se procede a la línea 6, donde $r$ se hace igual a

$$a \bmod b = 72 \bmod 36 = 0.$$

Pasamos de nuevo a las líneas 7 y 8, donde $a$ es 36 y $b$ es 0. Regresamos a la línea 5.

Ahora $b = 0$, por lo que vamos a la línea 10, donde la salida es $a$ (36), el máximo común divisor de 396 y 504. ◀

## Análisis del algoritmo euclidiano

Se analiza el desempeño del algoritmo 5.3.3 en el peor caso. Se define el tiempo requerido como el número de operaciones de módulo que se ejecutan en la línea 6. La tabla 5.3.1 lista el número de operaciones del módulo requeridas para algunos valores de entrada pequeños.

El peor caso para el algoritmo euclidiano ocurre cuando el número de operaciones de módulo es tan grande como sea posible. En referencia a la tabla 5.3.1, se puede determinar el par de entrada $a$, $b$, $a > b$, con $a$ tan pequeña como sea posible, que requiere $n$ operaciones de módulo para $n = 0, \ldots, 5$. Los resultados se presentan en la tabla 5.3.2.

Recuerde que la sucesión de Fibonacci $\{f_n\}$ (vea la sección 4.4) se define por las ecuaciones

$$f_1 = 1, \qquad f_2 = 1, \qquad f_n = f_{n-1} + f_{n-2}, \qquad n \geq 3.$$

La sucesión de Fibonacci comienza

$$1, \quad 1, \quad 2, \quad 3, \quad 5, \quad 8, \ldots.$$

Un patrón sorprendente se desarrolla en la tabla 5.3.2: la columna $a$ es la sucesión de Fibonacci comenzando con $f_2$ y, excepto por el primer valor, la columna $b$ también es la suce-

**TABLA 5.3.1** ■ Número de operaciones de módulo requeridas por el algoritmo euclidiano para diferentes valores de entrada.

| *a* \ *b* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 1 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| 3 | 0 | 1 | 2 | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 |
| 4 | 0 | 1 | 1 | 2 | 1 | 2 | 2 | 3 | 1 | 2 | 2 | 3 | 1 | 2 |
| 5 | 0 | 1 | 2 | 3 | 2 | 1 | 2 | 3 | 4 | 3 | 1 | 2 | 3 | 4 |
| 6 | 0 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 2 | 2 | 3 | 3 | 1 | 2 |
| 7 | 0 | 1 | 2 | 2 | 3 | 3 | 2 | 1 | 2 | 3 | 3 | 4 | 4 | 3 |
| 8 | 0 | 1 | 1 | 3 | 1 | 4 | 2 | 2 | 1 | 2 | 2 | 4 | 2 | 5 |
| 9 | 0 | 1 | 2 | 1 | 2 | 3 | 2 | 3 | 2 | 1 | 2 | 3 | 2 | 3 |
| 10 | 0 | 1 | 1 | 2 | 2 | 1 | 3 | 3 | 2 | 2 | 1 | 2 | 2 | 3 |
| 11 | 0 | 1 | 2 | 3 | 3 | 2 | 3 | 4 | 4 | 3 | 2 | 1 | 2 | 3 |
| 12 | 0 | 1 | 1 | 1 | 1 | 3 | 1 | 4 | 2 | 2 | 2 | 2 | 1 | 2 |
| 13 | 0 | 1 | 2 | 2 | 2 | 4 | 2 | 3 | 5 | 3 | 3 | 3 | 2 | 1 |

**TABLA 5.3.2** ■ Par más pequeño de entrada que requiere *n* operaciones de módulo en el algoritmo euclidiano.

| *a* | *b* | *n* (= *número de operaciones de módulo*) |
|---|---|---|
| 1 | 0 | 0 |
| 2 | 1 | 1 |
| 3 | 2 | 2 |
| 5 | 3 | 3 |
| 8 | 5 | 4 |
| 13 | 8 | 5 |

| | |
|---|---|
| $34 = 91 \bmod 57$ | (1 operación de módulo) |
| 57, 34 requiere 4 operaciones de módulo | (con un total de 5) |
| $57 \geq f_6$ y $34 \geq f_5$ | (por la suposición inductiva) |
| $\therefore\ 91 = 57 \cdot 1 + 34 \geq 57 + 34 \geq f_6 + f_5 = f_7$ | |

**Figura 5.3.1** Prueba del Teorema 5.3.5. El par 91, 57, que requiere $n + 1 = 5$ operaciones de módulo, es la entrada al algoritmo euclidiano.

sión de Fibonacci ¡comenzando con $f_2$! Se llega a la conjetura de que si el par $a, b, a > b$, cuando se introduce al algoritmo euclidiano requiere $n \geq 1$ operaciones de módulo, entonces $a \geq f_{n+2}$ y $b \geq f_{n+1}$. Como evidencia adicional de la conjetura, si se calcula el par de entrada más pequeño que requiere 6 operaciones de módulo, se obtiene $a = 21$ y $b = 13$. El siguiente Teorema confirma que la conjetura es correcta. La prueba del Teorema se ilustra en la figura 5.3.1.

**Teorema 5.3.5**

*Suponga que el par a, b, $a > b$, requiere $n \geq 1$ operaciones de módulo cuando se introduce al algoritmo. Entonces $a \geq f_{n+2}$ y $b \geq f_{n+1}$, donde $\{f_n\}$ denota la sucesión de Fibonacci.*

***Demostración*** La prueba es por inducción sobre *n*.

**Paso base ($n = 1$)** Se ha observado que el teorema es cierto para $n = 1$.

**Paso inductivo** Suponga que el Teorema es cierto para $n \geq 1$. Debe demostrarse que el teorema es cierto para $n + 1$.

Suponga que el par $a, b, a > b$, requiere $n + 1$ operaciones de módulo cuando se introduce al algoritmo euclidiano. En la línea 6, se calcula $r = a \bmod b$. Entonces

$$a = bq + r, \qquad 0 \leq r < b. \tag{5.3.2}$$

Luego, el algoritmo repite esto usando los valores $b$ y $r$, $b > r$. Estos valores requieren $n$ operaciones de módulo adicionales. Por la suposición inductiva,

$$b \geq f_{n+2} \qquad \text{y} \qquad r \geq f_{n+1}. \tag{5.3.3}$$

Combinando (5.3.2) y (5.3.3), se obtiene

$$a = bq + r \geq b + r \geq f_{n+2} + f_{n+1} = f_{n+3}. \tag{5.3.4}$$

[La primera desigualdad en (5.3.4) se cumple porque $q > 0$; $q$ no puede ser igual que 0 porque $a > b$.] Las desigualdades (5.3.3) y (5.3.4) dan

$$a \geq f_{n+3} \qquad \text{y} \qquad b \geq f_{n+2}$$

El paso inductivo está terminado y la prueba queda completa.

El teorema 5.3.5 resulta útil para analizar el desempeño del algoritmo euclidiano en el peor caso.

**Teorema 5.3.6**

*Si los enteros en un intervalo de* 0 *a* $m$, $m \geq 8$, *ambos diferentes de cero, se introducen al algoritmo euclidiano, entonces se requieren cuando mucho*

$$\log_{3/2} \frac{2m}{3}$$

*operaciones de módulo*.

***Demostración*** Sea $n$ el número máximo de operaciones de módulo requeridas por el algoritmo euclidiano para enteros en el intervalo de 0 a $m$, $m \geq 8$. Sea $a$, $b$ un par de entrada en el intervalo de 0 a $m$ que requiere $n$ operaciones de módulo. La tabla 5.3.1 muestra que $n \geq 4$ y $a \neq b$. Se puede suponer que $a > b$. (Al intercambiar los valores de $a$ y $b$ no se altera el número de operaciones de módulo requeridas). Por el Teorema 5.3.5, $a \geq f_{n+2}$. Entonces

$$f_{n+2} \leq m.$$

Por el ejercicio 27, en la sección 4.4, como $n + 2 \geq 6$,

$$\left(\frac{3}{2}\right)^{n+1} < f_{n+2}.$$

Combinando estas dos desigualdades, se obtiene

$$\left(\frac{3}{2}\right)^{n+1} < m.$$

Tomando logaritmos con base $3/2$, se obtiene

$$n + 1 < \log_{3/2} m.$$

Por lo tanto,

$$n < (\log_{3/2} m) - 1 = \log_{3/2} m - \log_{3/2} \frac{3}{2} = \log_{3/2} \frac{2m}{3}.$$

Como la función logaritmo crece con lentitud, el Teorema 5.3.6 nos dice que el algoritmo euclidiano es bastante eficiente, aun para valores grandes de los datos de entrada. Por ejemplo, dado que

$$\log_{3/2} \frac{2(1{,}000{,}000)}{3} = 33.07\ldots,$$

el algoritmo euclidiano requiere cuando mucho 33 operaciones de módulo para calcular el máximo común divisor de cualquier par de enteros, ambos diferentes de cero, en el intervalo de 0 a 1,000,000.

### Un resultado especial

El siguiente resultado especial se usará para calcular el inverso del módulo de un entero (vea el siguiente apartado). Estos inversos se usan en los sistemas criptográficos RSA(sección 5.4). Sin embargo, este resultado especial también es útil de otras maneras (vea los ejercicios 24 y 26 y el rincón de solución de problemas que sigue).

**Teorema 5.3.7**

*Si $a$ y $b$ son enteros no negativos, ambos diferentes de cero, existen enteros $s$ y $t$ tales que*

$$\text{mcd}(a, b) = sa + tb.$$

El método del algoritmo euclidiano se puede usar para probar el teorema 5.3.7 y calcular $s$ y $t$. Antes de demostrar el Teorema, se ilustrará la prueba con un ejemplo específico.

**Ejemplo 5.3.8 ▶**

Considere la forma en que el algoritmo euclidiano calcula el mcd(273, 110). Se comienza con $a = 273$ y $b = 110$. El algoritmo euclidiano calcula primero

$$r = 273 \bmod 110 = 53. \tag{5.3.5}$$

Después establece $a = 110$ y $b = 53$.

Luego el algoritmo calcula

$$r = 110 \bmod 53 = 4. \tag{5.3.6}$$

Después hace $a = 53$ y $b = 4$.

El algoritmo euclidiano calcula entonces

$$r = 53 \bmod 4 = 1 \tag{5.3.7}$$

Luego establece $a = 4$ y $b = 1$.

Ahora el algoritmo calcula

$$r = 4 \bmod 1 = 0.$$

Como $r = 0$, el algoritmo termina por encontrar el máximo común divisor de 273 y 110 como 1.

Para encontrar $s$ y $t$, se trabaja hacia atrás, comenzando con la última ecuación [(5.3.7)] donde $r \neq 0$. La ecuación (5.3.7) se rescribe como

$$1 = 53 - 4 \cdot 13 \tag{5.3.8}$$

ya que el cociente cuando 53 se divide entre 4 es 13.

La ecuación (5.3.6) se rescribe como

$$4 = 110 - 53 \cdot 2.$$

Después se sustituye esta fórmula para 4 en la ecuación (5.3.8) para obtener

$$1 = 53 - 4 \cdot 13 = 53 - (110 - 53 \cdot 2)13 = 27 \cdot 53 - 13 \cdot 110. \tag{5.3.9}$$

La ecuación (5.3.5) se rescribe como

$$53 = 273 - 110 \cdot 2.$$

Se sustituye esta fórmula para 53 en la ecuación (5.3.9) y se obtiene

$$1 = 27 \cdot 53 - 13 \cdot 110 = 27(273 - 110 \cdot 2) - 13 \cdot 110 = 27 \cdot 273 - 67 \cdot 110.$$

Así, si $s = 27$ y $t = -67$, se obtiene

$$\text{mcd}(273, 110) = 1 = s \cdot 273 + t \cdot 110.$$ ◀

***Demostración del Teorema 5.3.7*** Dados $a > b \geq 0$, sea $r_0 = a$, $r_1 = b$ y $r_i$ igual al valor de $r$ después de la ($i$–1)ésima vez que se ejecuta el ciclo en el algoritmo 5.3.3 (por ejemplo, $r_2 = a \bmod b$). Suponga que $r_n$ es el primer valor de $r$ que es cero, de manera que $\text{mcd}(a, b) = r_{n-1}$.

En general,

$$r_i = r_{i+1}q_{i+2} + r_{i+2}. \tag{5.3.10}$$

Haciendo $i = n - 3$ en (5.3.10), se obtiene

$$r_{n-3} = r_{n-2}q_{n-1} + r_{n-1},$$

que se rescribe como

$$r_{n-1} = -q_{n-1}r_{n-2} + 1 \cdot r_{n-3}.$$

Se puede hacer $t_{n-3} = -q_{n-1}$ y $s_{n-3} = 1$ para obtener

$$r_{n-1} = t_{n-3}r_{n-2} + s_{n-3}r_{n-3}. \tag{5.3.11}$$

Haciendo $i = n - 4$ en (5.3.10) se obtiene

$$r_{n-4} = r_{n-3}q_{n-2} + r_{n-2}$$

o sea

$$r_{n-2} = -q_{n-2}r_{n-3} + r_{n-4}. \tag{5.3.12}$$

Al sustituir (5.3.12) en (5.3.11) se tiene

$$\begin{aligned} r_{n-1} &= t_{n-3}[-q_{n-2}r_{n-3} + r_{n-4}] + s_{n-3}r_{n-3} \\ &= [-t_{n-3}q_{n-2} + s_{n-3}]r_{n-3} + t_{n-3}r_{n-4}. \end{aligned}$$

Estableciendo $t_{n-4} = -t_{n-3}q_{n-2} + s_{n-3}$ y $s_{n-4} = t_{n-3}$, se obtiene

$$r_{n-1} = t_{n-4}r_{n-3} + s_{n-4}r_{n-4}.$$

Si se continúa de esta manera, al final se obtiene

$$\text{mcd}(r_0, r_1) = r_{n-1} = t_0r_1 + s_0r_0 = t_0\, b = s_0a.$$

Y si se hace $s = s_0$ y $t = t_0$, se llega a

$$\text{mcd}(r_0, r_1) = sa + tb. \qquad \blacktriangleleft$$

## Cálculo del inverso del módulo de un entero

Suponga que se tienen dos enteros $n > 0$ y $\phi > 1$ tal que $\text{mcd}(n, \phi) = 1$. Se mostrará cómo calcular de manera eficiente un entero $s$, $0 < s < \phi$ tal que $ns \bmod \phi = 1$. Llamamos a $s$ el **inverso de *n* mod** $\phi$. El sistema criptográfico RSA en la sección 5.4 requiere calcular con eficiencia este inverso.

Como el $\text{mcd}(n, \phi) = 1$, se usa el algoritmo euclidiano, como se explicó antes, para encontrar números $s'$ y $t'$ tales que $s'n + t'\phi = 1$. Entonces $ns' = -t'\phi + 1$ y, como $\phi > 1$, el residuo es 1. Entonces

$$ns' \bmod \phi = 1. \tag{5.3.13}$$

Advierta que $s'$ es casi el valor deseado; el problema es que quizá $s'$ no satisfaga $0 < s' < \phi$. Sin embargo, es posible convertir $s'$ en el valor adecuado haciendo

$$s = s' \bmod \phi.$$

Ahora $0 \leq s < \phi$. De hecho, $s \neq 0$ puesto que si $s = 0$, entonces $\phi \mid s'$, lo que contradice (5.3.13). Como $s = s' \bmod \phi$, existe $q$ tal que

$$s' = q\phi + s.$$

Combinando las ecuaciones anteriores, se tiene

$$ns = ns' - \phi nq = -t'\phi + 1 - \phi nq = \phi(-t' - nq) + 1.$$

Por lo tanto,

$$ns \bmod \phi = 1. \tag{5.3.14}$$

**Ejemplo 5.3.9 ▶**

Sean $n = 110$ y $\phi = 273$. En el ejemplo 5.3.8, se demostró que el $\text{mcd}(n, \phi) = 1$ y que

$$s'n + t'\phi = 1,$$

donde $s' = -67$ y $t' = 27$. Así,

$$110(-67) \bmod 273 = ns' \bmod \phi = 1.$$

Aquí $s = s' \bmod \phi = -67 \bmod 273 = 206$. Por lo tanto, el inverso de 110 módulo 273 es 206. ◀

Se concluye probando que el número $s$ en la ecuación (5.3.14) es único. Suponga que

$$ns \bmod \phi = 1 = ns' \bmod \phi, \qquad 0 < s < \phi, \qquad 0 < s' < \phi.$$

Debe demostrarse que $s = s'$. Ahora

$$s' = (s' \bmod \phi)(ns \bmod \phi) = s'ns \bmod \phi = (s'n \bmod \phi)(s \bmod \phi) = s.$$

Por lo tanto, el número $s$ en la ecuación (5.3.14) es único.

## Sugerencias para resolver problemas

El algoritmo euclidiano para calcular el máximo común divisor de enteros no negativos $a$ y $b$, ambos diferentes de cero, se basa en la ecuación

$$\text{mcd}(a, b) = \text{mcd}(b, r),$$

donde $r = a \bmod b$. Se sustituye el problema original, calcular el $\text{mcd}(a, b)$, por el problema, calcular el $\text{mcd}(b, r)$. Después se remplaza $a$ por $b$ y $b$ por $r$, y se repite. A la larga, $r = 0$, de manera que la solución es $\text{mcd}(b, 0) = b$.

El algoritmo euclidiano es bastante eficiente. Si dos enteros en el intervalo de 0 a $m$, $m \geq 8$, ambos diferentes de cero, se dan como datos al algoritmo euclidiano, entonces se requieren cuando mucho

$$\log_{3/2} \frac{2m}{3}$$

operaciones de módulo.

Si $a$ y $b$ son enteros no negativos, ambos diferentes de cero, existen enteros $s$ y $t$ tales que

$$\text{mcd}(a, b) = sa + tb.$$

Para calcular $s$ y $t$, se usa el algoritmo euclidiano. En un problema que implica el máximo común divisor, la ecuación anterior resultará útil. (Intente realizar los ejercicios 24 y 26).

Suponga que se tienen dos enteros $n > 0$ y $\phi > 1$ tales que $\text{mcd}(n, \phi) = 1$. Para calcular de manera eficiente un entero $s$, $0 < s < \phi$ tal que $ns \bmod \phi = 1$, primero se calculan $s'$ y $t'$ que satisfacen

$$\text{mcd}(n, \phi) = s'n + t'\phi$$

(vea el apartado "Cálculo del inverso del módulo de un entero"). Después se hace $s = s' \bmod \phi$.

## Sección de ejercicios de repaso

1. Enuncie el algoritmo euclidiano.
2. ¿Qué teorema clave es la base para el algoritmo euclidiano?
3. Si el par $a, b, a > b$, requiere $n \geq 1$ operaciones de módulo cuando se alimenta al algoritmo euclidiano, ¿cómo se relacionan $a$ y $b$ con la sucesión de Fibonacci?
4. Dos enteros en el intervalo de 0 a $m$, $m \geq 8$, ambos diferentes de cero, se alimentan al algoritmo euclidiano. Dé una cota superior para el número de operaciones de módulo requeridas.
5. El Teorema 5.3.7 establece que existen enteros $s$ y $t$ tales que $\text{mcd}(a, b) = sa + tb$. Explique cómo utilizar el algoritmo euclidiano para calcular $s$ y $t$.
6. Explique qué significa que $s$ sea el inverso de $n$ módulo $\phi$.
7. Suponga que $\text{mcd}(n, \phi) = 1$. Explique cómo calcular el inverso de $n$ módulo $\phi$.

## Ejercicios

*Use el algoritmo euclidiano para encontrar el máximo común divisor de cada par de enteros en los ejercicios 1 al 10.*

1. 60, 90
2. 110, 273
3. 220, 1400
4. 315, 825
5. 20, 40
6. 331, 993
7. 2091, 4807
8. 2475, 32670
9. 67942, 4209
10. 490256, 337
11. Para cada par de números $a, b$ en los ejercicios 1 al 10, encuentre enteros $s$ y $t$ tales que $sa + tb = \text{mcd}(a, b)$.
12. Encuentre dos enteros $a$ y $b$, cada uno menor que 100, que maximicen el número de iteraciones del ciclo "while" del algoritmo 5.3.3.
13. Demuestre que el algoritmo 5.3.3 encuentra correctamente el mcd$(a, b)$ aun cuando se eliminen las líneas 3 y 4.
14. Escriba una versión recursiva del algoritmo euclidiano. Pruebe que su algoritmo es correcto.
15. Si $a$ y $b$ son enteros positivos, demuestre que $\text{mcd}(a, b) = \text{mcd}(a, a + b)$.
16. Demuestre que si $a > b \geq 0$, entonces

$$\text{mcd}(a, b) = \text{mcd}(a - b, b).$$

17. Con base en el ejercicio 16, escriba un algoritmo que calcule el máximo común divisor de dos enteros no negativos $a$ y $b$, ambos diferentes de cero, que use la resta pero no las operaciones de módulo.
18. ¿Cuántas restas requiere el algoritmo del ejercicio 17 en el peor caso para números en el intervalo de 0 a $m$?
19. Amplíe las tablas 5.3.1 y 5.3.2 al intervalo de 0 a 21.
20. ¿Exactamente cuántas operaciones de módulo requiere el algoritmo euclidiano en el peor caso para números entre 0 y 1,000,000?
21. Pruebe que cuando el par $f_{n+2}$, $f_{n+1}$ se introduce en el algoritmo euclidiano, $n \geq 1$, se requieren exactamente $n$ operaciones de módulo.
22. Demuestre que para cualquier entero $k > 1$, el número de operaciones de módulo requeridas por el algoritmo euclidiano para calcular el mcd$(a, b)$ es el mismo que el número de operaciones de módulo requeridas para calcular mcd$(ka, kb)$.
23. Demuestre que el $\text{mcd}(f_n, f_{n+1}) = 1$, $n \geq 1$.
24. ★ Demuestre que si $p$ es un número primo, $a$ y $b$ son enteros positivos y $p \mid ab$, entonces $p \mid a$ o $p \mid b$.
25. Dé un ejemplo de enteros positivos $p$, $a$ y $b$ donde $p \mid ab$, $p \nmid a$ y $p \nmid b$.
26. Sean $m$ y $n$ enteros positivos. Sea $f$ la función de

$$X = \{0,1, \ldots, m - 1\}$$

en $X$ definida por

$$f(x) = nx \bmod m.$$

Pruebe que $f$ es uno a uno y sobre si y sólo si $\text{mcd}(m, n) = 1$.

*Los ejercicios 27 al 31 muestran otra manera de probar que si $a$ y $b$ son enteros no negativos, ambos diferentes de cero, existen enteros $s$ y $t$ tales que*

$$\text{mcd}(a, b) = sa + tb.$$

*Sin embargo, a diferencia del algoritmo euclidiano, esta demostración no conduce a una técnica para calcular $s$ y $t$.*

27. Sea

$$X = \{sa + tb \mid sa + tb > 0 \text{ y } s \text{ y } t \text{ son enteros}\}.$$

Demuestre que $X$ es no vacío.

28. Demuestre que $X$ tiene un elemento menor. Denote por $g$ el elemento menor.
29. Demuestre que si $c$ es un divisor común de $a$ y $b$, entonces $c$ divide a $g$.
30. Demuestre que $g$ es un divisor común de $a$ y $b$. *Sugerencia:* Suponga que $g$ no divide a $a$. Entonces $a = qg + r$, $0 < r < g$. Obtenga una contradicción demostrando que $r \in X$.
31. Demuestre que $g$ es el máximo común divisor de $a$ y $b$.

*En los ejercicios 32 al 38, demuestre que el* $\text{mcd}(n, \phi) = 1$ *y encuentre el inverso $s$ de $n$ módulo $\phi$ que satisface* $0 < s < \phi$.

32. $n = 2, \phi = 3$
33. $n = 1, \phi = 47$
34. $n = 7, \phi = 20$
35. $n = 11, \phi = 47$
36. $n = 50, \phi = 231$
37. $n = 100, \phi = 231$
38. $n = 100, \phi = 243$
39. Demuestre que 6 no tiene inverso módulo 15. ¿Contradice esto el resultado que precede al ejemplo 5.3.9? Explique su respuesta.
40. Demuestre que $n > 0$ tiene un inverso módulo $\phi > 1$ si y sólo si $\text{mcd}(n, \phi) = 1$.