Algoritmo de Euclides

Melanie Sclar

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

AED III

Ejercicio 2.8 de la práctica

Ejercicio 2.8

a. Escribir el algoritmo de Euclides para calcular el máximo común divisor entre 2 números b y c en forma recursiva y no recursiva. Mostrar que su complejidad es $O(min\{b, c\})$. Puede construir un ejemplo (un peor caso) donde esta complejidad efectivamente se alcance? ¿Puede hacerlo para b y c tan grandes como se desee?

Máximo Común Divisor

Definición del máximo común divisor o *mcd*

El máximo común divisor entre a y b o $mcd(a, b)$ ($a, b \in \mathbb{N}_0$) es el mayor número natural que los divide a ambos sin dejar resto.

Por ejemplo, $mcd(2, 5) = 1$, $mcd(10, 30) = 10$, $mcd(55, 77) = 11$, $mcd(15, 1) = 1$, $mcd(17, 0) = 17$.

Notar que si $mcd(a, b) = k$ entonces $mcd(\frac{a}{k}, \frac{b}{k}) = 1$ pues quitamos todos los factores primos que eran comunes a ambos números.

Algoritmos recursivos

Definición

Un algoritmo se dice recursivo si calcula instancias de un problema en función de otras instancias del mismo problema hasta llegar a un caso base, que suele ser una instancia pequeña del problema, cuya respuesta generalmente está dada en el algoritmo y no es necesario calcularla.

Para calcular el *mcd* de forma recursiva, necesitamos una propiedad del problema que cumpla esto: es decir, que escriba $mcd(a, b)$ en función de instancias menores que ella hasta llegar a un caso base.

Algoritmos recursivos

Definición

Un algoritmo se dice recursivo si calcula instancias de un problema en función de otras instancias del mismo problema hasta llegar a un caso base, que suele ser una instancia pequeña del problema, cuya respuesta generalmente está dada en el algoritmo y no es necesario calcularla.

Para calcular el *mcd* de forma recursiva, necesitamos una propiedad del problema que cumpla esto: es decir, que escriba $mcd(a, b)$ en función de instancias menores que ella hasta llegar a un caso base.

Algoritmos recursivos

Lema

Sea $a \leq b$ $(a, b \in \mathbb{N}_0)$, entonces $mcd(a, b) = k$, $mcd(a, b - a) = k$.

Algoritmos recursivos

Lema

Sea $a \leq b$ $(a, b \in \mathbb{N}_0)$, entonces $mcd(a, b) = k$, $mcd(a, b - a) = k$.

Demostración: $mcd(a, b) = k$, luego $a = ka'$, $b = kb'$ con $mcd(a', b') = 1$.

Veamos que $mcd(a, b - a) = k$:
$mcd(a, b-a) = mcd(ka', kb' - ka') = mcd(ka', k(b' - a')) = k \cdot mcd(a', b' - a')$

$\Rightarrow mcd(a, b - a) \geq k \cdot 1 = k$. Entonces seguro que $mcd(a, b - a) \geq k$, pero ¿puede suceder que $mcd(a, b - a) > k$?

Supongamos que $mcd(a, b - a) = q > k$. Entonces tenemos que:

$$\begin{cases} a \equiv 0 \pmod{q} \\ b - a \equiv 0 \pmod{q} \end{cases}$$

Luego concluimos que $a \equiv b \equiv 0 \pmod{q}$.

Como $a \equiv b \equiv 0 \pmod{q}$, deducimos que a y b son los dos múltiplos de q (y recordemos que por hipótesis $q > k$). Pero luego $mcd(a, b) \geq q > k$.

¡Absurdo pues $mcd(a, b) = k$ por hipótesis del ejercicio! Provino de suponer que $mcd(a, b - a) > k$. Luego, como $mcd(a, b - a) \leq k$ y $mcd(a, b - a) \geq k$ deducimos que $mcd(a, b - a) = k$. ■

Lema 1

Sea $a \leq b$ $(a, b \in \mathbb{N}_0)$, entonces $mcd(a, b) = k$, $mcd(a, b - a) = k$.

Este lema es válido como función recursiva pero puede ser mejorado. Notemos que si $b - a \geq a$, entonces al aplicar de nuevo el Lema obtendremos $mcd(a, b - 2a) = k$, y así sucesivamente hasta que $b - xa < a$. Es decir que podemos mejorar el lema 1 aplicándolo varias veces para obtener el lema 2.

Lema 1

Sea $a \leq b$ $(a, b \in \mathbb{N}_0)$, entonces $mcd(a, b) = k$, $mcd(a, b - a) = k$.

Este lema es válido como función recursiva pero puede ser mejorado. Notemos que si $b - a \geq a$, entonces al aplicar de nuevo el Lema obtendremos $mcd(a, b - 2a) = k$, y así sucesivamente hasta que $b - xa < a$. Es decir que podemos mejorar el lema 1 aplicándolo varias veces para obtener el lema 2.

Lema 2

Sea $0 < a \leq b$ $(a, b \in \mathbb{N}_0)$, entonces $mcd(a, b) = k$, $mcd(a, b \pmod{a}) = k$.

Lema 2

Sea $0 < a \leq b$ ($a, b \in \mathbb{N}_0$), entonces $mcd(a, b) = k$, $mcd(a, b \bmod a) = k$.

Lema 2

Sea $0 < a \leq b$ ($a, b \in \mathbb{N}_0$), entonces $mcd(a, b) = k$, $mcd(a, b \bmod a) = k$.

Demostración:

Como $a \leq b$, b se puede escribir como $b = ra + s$ con $r, s \in \mathbb{N}_0$ y $0 \leq s < a$.

Aplicando sucesivamente el Lema 1, por inducción podemos ver que $mcd(a, b - ra) = k$, pues $b - (r-1)a \geq a$ y por ende vale aplicar el lema por r-ésima vez. Ahora bien, $b - ra = s$, y s es justamente el resto de b módulo a. Es decir, $s = b \bmod a$.
En conclusión, queda que $mcd(a, s) = k$, o en otras palabras $mcd(a, b \bmod a) = k$. ■

Entonces ya tenemos una propiedad recursiva que nos permitirá hallar $mcd(a, b)$. Pero nos falta saber cuándo parar, es decir, los **casos base**.

Entonces ya tenemos una propiedad recursiva que nos permitirá hallar $mcd(a, b)$. Pero nos falta saber cuándo parar, es decir, los **casos base**.

$mcd(a, 0) = a$
$mcd(0, a) = a$

Notemos además que si no fueran casos base incurriríamos en un error, pues $a \mod 0$ no se puede efectuar.

Pseudocódigo de solución recursiva

```
function mcd (natural a, natural b) {
      if (a == 0)
            devolver b
      if (b == 0)
            devolver a
      if (a <= b)
            devolver mcd(a, b mod a)
      devolver mcd(a mod b, b)
}
```

Pseudocódigo de solución iterativa

```
function mcd (natural a, natural b) {
        if (a > b)
                intercambiar a y b
        // siempre mantendremos el invariante a <= b

        while (a > 0) {
                tmp = b mod a
                b = a
                a = tmp
                // notar que b mod a <= a,
                // por eso los pusimos en ese orden
        }
        devolver b
}
```

Complejidad (esbozo)

Es fácil ver que nuestras implementaciones son $O(min(b, c))$ pues en cada paso el número más chico entre los dos involucrados decrece en al menos 1. Como cada ejecución es $O(1)$ y paramos cuando el mínimo llega a 0, la ejecución completa sera $O(min(b, c))$. Después veremos que esta cota se puede mejorar mucho más.

Peores casos - Fibonacci

Los peores casos del algoritmo ocurren cuando a y b son Fibonacci's consecutivos. Esto es así pues sabemos que en cada paso se efectúa al menos una resta (hacer $b \mod a$ implica restarle al menos a a b) y el peor caso ocurre cuando en cada paso se efectúa exactamente una resta.

$$mcd(F_k, F_{k+1}) = mcd(F_{k-1}, F_k) = \cdots = mcd(F_0, F_1) = mcd(0, 1) = 1$$

Más adelante retomaremos esta observación para demostrar la complejidad real de nuestros algoritmos.

Ejercicio 2.8 de la práctica

Ejercicio 2.8

b. Analizar el siguiente algoritmo para determinar el máximo común divisor entre dos números b y c, y mostrar que su complejidad también es $O(min(b, c))$.

$g \leftarrow min(b, c)$
`mientras` $g > 1$ `hacer`
    `si` $\frac{b}{g}$ `y` $\frac{c}{g}$ `son enteros, informar` $mcd = g$ `y parar.`
    `poner` $g = g - 1$
`informar` $mcd = 1$ `y parar`

Notemos que el algoritmo es correcto pues recorre todos los posibles divisores de b y c y se queda con el mayor de ellos (que por como recorremos, es el primero que cumple la propiedad).

$mcd(b, c) \leq min(b, c)$, y recorremos a lo sumo una vez el cuerpo por cada uno de los g. Dicho cuerpo es $O(1)$, y se ejecuta $O(min(b, c))$ veces, por lo tanto el algoritmo completo es $O(min(b, c))$.

Ejercicio 2.8 de la práctica

Ejercicio 2.8

c. Las complejidades calculadas para los algoritmos de las partes a. y b. son iguales. ¿Cuál de los dos algoritmos elegirá? Justificar y comentar.

c. Probar que se puede mejorar la complejidad calculada en a. demostrando que el algoritmo de Euclides es en realidad $O(log_2(min\{b, c\}))$.

Demostración de complejidad de las implementaciones del ítem a

Recordemos que en cada paso tomamos módulo del número más grande sobre el más chico. Así, en cada paso obtenemos un par de números. Como el *mcd* es conmutativo, ponemos primero el menor número y luego el mayor. Como $a \leq b$, en dos pasos de aplicación del lema 2 se tiene que:

$$mcd(a, b) = mcd(b \mod a, a) = mcd(a \mod (b \mod a), b \mod a)$$

Queremos probar que el algoritmo es $O(\lg a)$, pues $a = min(a, b)$, y para ello demostraremos que cada 2 pasos del algoritmo, a se reduce a la mitad.

Demostración de complejidad de las implementaciones del ítem a (cont.)

$$mcd(a, b) = mcd(b \bmod a, a) = mcd(a \bmod (b \bmod a), b \bmod a)$$

Queremos probar que el algoritmo es $O(\lg a)$ (pues $a = min(a, b)$) y para ello demostraremos que cada 2 pasos del algoritmo, el mínimo del par actual se reduce a la mitad. Si probamos eso, como cada paso es $O(1)$ y habrá $O(2 \lg a) = O(\lg a)$ pasos, el algoritmo será $O(\lg a)$.

$mcd(a, b) = mcd(b \mod a, a) = mcd(a \mod (b \mod a), b \mod a)$

1. Si $b \mod a \leq \frac{a}{2}$ entonces el mínimo número del par se redujo a la mitad en tan solo un paso. Como en cada paso se reduce cada número o queda igual, luego de 2 pasos el mínimo número del par original se habrá reducido a la mitad.

2. Si $b \mod a > \frac{a}{2}$, veamos que $a \mod (b \mod a)$ será pequeño:
   $a = (b \mod a)r + s$, pues $b \mod a < a$ (por definición de módulo).
   Ahora bien, como $b \mod a > \frac{a}{2}$ deducimos que $r = 1$. En otras palabras, $a = (b \mod a) + s \Rightarrow a - (b \mod a) = s < a - \frac{a}{2}$, por lo que $s < \frac{a}{2}$. Como s es por definición $a \mod (b \mod a)$, acabamos de ver que el menor número del par se redujo a la mitad de su valor original en dos pasos.

Tarea

- `http://acm.timus.ru/problem.aspx?space=1&num=1139`
  - Un lindo problema que sale con MCD.