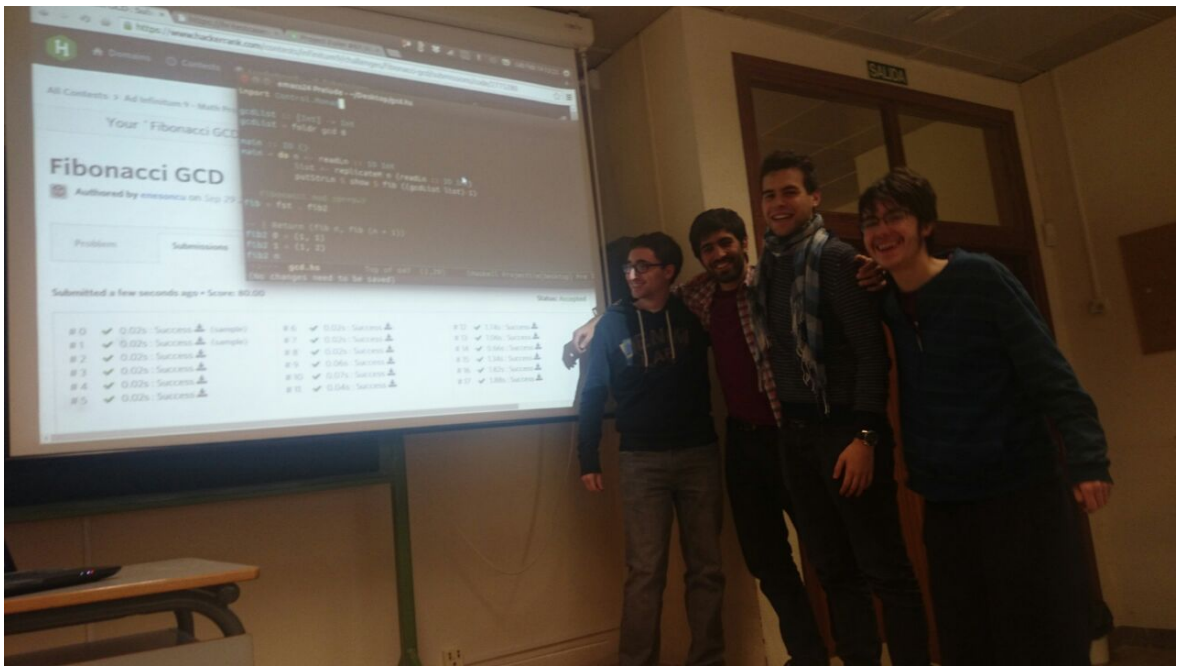
Seminarios del Doble Grado
Problemas de Hackerrank: Fibonacci GCD



Andrés Herrera Poyatos, Óscar Bermúdez Garrido

15 de febrero de 2015

Enunciado del Problema

El problema a resolver se encuentra en Hackerrank. Se muestra continuación una versión traducida del mismo.

Los números de Fibbonacci tienen la siguiente forma:

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_2 &= 1 \\ F_3 &= 2 \\ &\vdots \\ F_n &= F_{n-2} + F_{n-1} \end{aligned}$$

Tenemos un array $a_1, a_2, \ldots, a_n$ que contiene n elementos. Se pretende calcular $\gcd(F_{a_1}, F_{a_2}, \ldots, F_{a_n})$.

El formato de entrada es:

- La primera línea contiene el tamaño del array, n.
- En las siguientes n líneas hay un número, la i-ésima línea contiene a_i.

El formato de salida consiste en imprimir una sólo número entero. Este es el resto de la división entera del máximo común divisor pedido entre $10^9 + 7$.

Las restricciones son las siguientes:

- $1 \leq n \leq 2 \times 10^5$
- $1 \leq a_i \leq 10^{12}$

Solución

La primera observación a realizar es que, dado el tamaño que pueden tomar los subíndices a_i, no se puede pretender calcular tal cual los números de Fibonacci. Debemos obtener propiedades sobre el máximo común divisor de estos que nos permitan calcularlo de forma indirecta. La siguiente serie de proposiciones busca este objetivo.

Proposición 1. *Sean $n, k \in \mathbb{N}$. Se tiene que $F_{n+k} = F_{k-1}F_n + F_kF_{n+1}$.*

Demostración. La prueba se realiza por inducción sobre k para un n arbitrario en cada paso. Para $k = 1$ es trivial denotando $F_0 = 0$. Supongamos el resultado cierto para $k \in \mathbb{N}$.

$$F_{n+k+1} = F_{k-1}F_{n+1} + F_kF_{n+2} = F_{k-1}F_{n+1} + F_k(F_n + F_{n+1}) = F_kF_n + F_{k+1}F_{n+1}$$

donde se ha utilizado en primer lugar la hipótesis de inducción para k y posteriormente la definición de la sucesión dos veces. □

Proposición 2. *Sean $n, k \in \mathbb{N}$. Se tiene que $\gcd(F_n, F_{k+n}) = gcd(F_n, F_k)$.*

Demostración. En primer lugar:

$$\gcd(F_n, F_{n+1}) = \gcd(F_n, F_n + F_{n-1}) = \gcd(F_n, F_{n-1})$$

donde se ha utilizado que $\gcd(a, b) = \gcd(a, b - qb)$ para cualquier q. Por inducción se llega a que

$$\gcd(F_n, F_{n+1}) = \gcd(F_1, F_2) = \gcd(1, 1) = 1$$

Luego los términos consecutivos de la sucesión de Fibonacci son primos relativos entre sí.

Ahora, para $k > 1$ usamos la proposición anterior:

$$\gcd(F_n, F_{n+k}) = \gcd(F_n, F_{k-1}F_n + F_kF_{n+1}) = \gcd(F_n, F_kF_{n+1})$$

Como F_n y F_{n+1} son primos relativos:

$$\gcd(F_n, F_{n+k}) = \gcd(F_n, F_kF_{n+1}) = \gcd(F_n, F_k)$$

□

Proposición 3. *Sean $a, b \in \mathbb{N}$. Se tiene que $\gcd(F_a, F_b) = F_{gcd(a,b)}$.*

Demostración. Si $a = b$ es trivial. Supongamos que $a < b$ sin pérdida de generalidad. Tenemos que $\gcd(F_a, F_b) = gcd(F_a, F_{b-a})$. Podemos repetir el proceso hasta que aparezca un 0 en los índices. Estamos haciendo en definitiva el Algoritmo de Euclides sobre los índices y por ser el mismo proceso tenemos garantizado que el índice final no nulo es el máximo común divisor. Esto es:

$$\gcd(F_a, F_b) = \gcd(F_0, F_{\gcd(a,b)}) = \gcd(0, F_{\gcd(a,b)}) = F_{\gcd(a,b)}$$

□

Proposición 4. *Sean $n \in \mathbb{N}$ y $a_1, \dots a_n \in \mathbb{N}$. Se tiene que $\gcd(F_{a_1}, \dots, F_{a_n}) = F_{gcd(a_1,\dots,a_n)}$.*

Demostración. Usaremos que $\gcd(b_1, \dots, b_n) = \gcd(\gcd(b_1, b_2), b_3, \dots, b_n)$:

$$\gcd(F_{a_1}, \dots, F_{a_n}) = \gcd(\gcd(F_{a_1}, F_{a_2}), F_{a_3}, \dots, F_{a_n}) = \gcd(F_{\gcd(a_1,a_2)}, F_{a_3}, \dots, F_{a_n})$$

Repetimos el proceso $n - 1$ veces y usando la definición del máximo común divisor de n elementos sobre los a_i obtenemos el resultado. □

De la proposición anterior se tiene automáticamente el Algoritmo 1 que resuelve el problema.

Algoritmo 1 Solución del problema GCD Fibonacci

Dados $a_1, \dots, a_n \in \{1, \dots, 10^{12}\}$ con $n \in \{1, \dots, 2 \times 10^5\}$. Realizamos el siguiente algoritmo:

1. Calculamos $m = \gcd(a_1, \dots, a_n)$.
2. Calculamos $F_m \mod 10^9 + 7$

La única cuestión restante consiste en la implementación del algoritmo anterior. Para calcular el máximo común divisor de la lista utilizamos el Algoritmo de Euclides para los dos primeros elementos, que quedan sustituidos por el resultado obtenido. Repetimos el proceso hasta obtener un único número final que es el máximo común divisor buscado. Este algoritmo tiene eficiencia n veces la del propio Algoritmo de Euclides.

Para calcular $F_m \mod 10^9 + 7$ tenemos que tener en cuenta que m puede ser hasta 10^{12}. Es fácil realizar un algoritmo lineal para este propósito calculando todos los términos progresivamente y guardando solo los dos últimos en cada iteración. Sin embargo, esto no alcanza los objetivos en el peor caso. Se propone la siguiente idea para obtener una solución logarítmica sobre m:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_{k+2} & F_{k+1} \\ F_{k+1} & F_k \end{pmatrix} = \begin{pmatrix} F_{k+3} & F_{k+2} \\ F_{k+2} & F_{k+1} \end{pmatrix} \tag{1}$$

Consecuentemente, por inducción:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^k \begin{pmatrix} F_2 & F_1 \\ F_1 & F_0 \end{pmatrix} = \begin{pmatrix} F_{k+3} & F_{k+2} \\ F_{k+2} & F_{k+1} \end{pmatrix} \tag{2}$$

En nuestro caso:

$$\begin{pmatrix} F_2 & F_1 \\ F_1 & F_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \tag{3}$$

Por consiguiente:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^k = \begin{pmatrix} F_{k+2} & F_{k+1} \\ F_{k+1} & F_k \end{pmatrix} \tag{4}$$

El problema se ha reducido a la exponenciación de la matriz $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$. La exponenciación de matrices se puede conseguir logarítmica de forma análoga a la exponenciación (con exponente natural) de un número. El Algoritmo 2 muestra un pseudo-código recursivo para este propósito.

Algoritmo 2 Exponenciación de matrices. Calcula A^n en $\theta(\log n)$.

```
function POTENCIA(A, n)
    if n = 1 then return A
    else if n mód 2 = 0 then
        B ← Potencia(A, n/2)
        return B^2
    else
        B ← Potencia(A, n/2)
        return AB^2
    end if
end function
```

Puesto que m puede ser 10^{12}, el valor obtenido por el Algoritmo 2 en este caso no cabría en memoria. De todas formas, debemos imprimir el resultado módulo $10^9 + 7$. Puesto que el módulo de una suma o producto es el módulo de la suma o producto de los módulos, podemos aplicar el módulo a cada operación realizada manteniendo el funcionamiento. De esta forma los números con los que trabajará el algoritmo serán menores que $10^9 + 7$, por lo que puede ejecutarse sin ningún problema.