MA-53G Intr. a la Criptografía **Semestre Otoño, 1998**

Cátedra No. 3 : 23 de Marzo

Transcriptor: Felipe Saint-jean[1]

1 Divisibilidad y el Algoritmo de Euclides

En esta sección recordaremos algunas definiciones y repasaremos algunos resultados que conciernen a los números enteros.

Dados dos enteros a y b decimos que *a divide a b*, o *b es divisible por a*, y escribimos $a|b$ si existe un entero k tal que $b = ak$. En este caso decimos que a es un divisor de b. Si a no es un divisor de b escribimos $a \not| b$.

Se tienen las siguientes propiedades:

- Si $a|b$ y c es un entero, entonces $a|b\,c$.
- Si $a|b$ y $b|c$, entonces $a|c$.
- Si $a|b$ y $a|c$, entonces $a|(b \pm c)$.

Notemos que esta relación así definida, no es relación de orden, pero de esta manera es valida para cualquier Cada entero b mayor que 1 tiene al menos dos divisores: 1 y b. Un divisor no trivial de b es un divisor de b distinto de 1 y de b. Un número *primo* es, por definición, un número mayor que 1 que no tiene divisores no triviales. Un número es *compuesto* si es mayor que 1 y tiene al menos un divisor no trivial.

Teorema 1 *El Teorema Fundamental de la Aritmética nos dice que cualquier número entero positivo se puede expresar de manera única (salvo el orden de los factores) como un producto de primos. Sea m entero:*

$$m = \prod_{i=1}^{n} p_i^{e_i}$$

donde p_i son los factores primos, y e_i son las veces que se repite el i-ésimo factor.

Dos consecuencias de este importante teorema son:

- Si p es primo y $p|(ab)$, entonces $p|a$ o $p|b$.
- Si $m|a$ y $n|a$, y m y n no tienen divisores comunes mayores que 1, entonces $(mn)|a$.

[1]Estoso apuntes están basados en la transcripción de Marcos Kiwi (7/8/96

El *maximo común divisor* de dos enteros a y b no ambos nulos, denotado por $\mathrm{mcd}(a,b)$, es el mayor entero d tal que $d|a$ y $d|b$. Equivalentemente, $d = \mathrm{mcd}(a,b)$ si cualquiera sea el entero positivo c tal que $c|a$ y $c|b$ se tiene que $c|d$. Es facil ver que el máximo común divisor es único, ya que de haber dos, uno será mayor que el otro. También notemos que $\mathrm{mcd}(a,b)$=$\mathrm{mcd}(b,a)$.

El *minimo común multiplo* de dos enteros no nulos a y b, denotado $\mathrm{mcm}(a,b)$, es el menor entero positivo divisible por a y b. No es dificil probar que $\mathrm{mcm}(a,b) = |a\,b|/\mathrm{mcd}(a,b)$.

Dados dos enteros a y b, si conocemos los factores primos de ambos, podemos calcular el *maximo común divisor* entre ambos números de la siguiente forma. Si

$$a = \prod_{p \epsilon B} p^{\alpha_p}, b = \prod_{q \epsilon A} q^{\alpha_q}$$

entonces

$$\mathrm{mcd}(a,b) = \prod_{p \epsilon A \cap B} p_i^{min(\alpha_p, \beta_p)}$$

Donde A es el conjunto de factores de a, B es el conjunto de factres de b , α_p el número de veces que se repite el factor p en la factorización de a, y β_p el número de veces que p esta en la factorización de b. El problema de este método es que no se sabe cómo obtener los factores primos de un número de manera eficientemente. Es decir el mejor algoritmo conocido toma un tiempo exponencial, que en general, no nos sirve. Este inconveniente da origen alsiguiente algoritmo.

1.1 Algoritmo de Euclides

Sean u y v dos enteros no ambos nulos, definimos la función recursiva

```
Euclid(u,v)
   if (v = 0) then
      return (|u|)
   else return (Euclid(b,u mod b))
end:
```

El algoritmo Euclid retorna el $\mathrm{mcd}(u,v)$, y para eso se basa en dos hechos fundamentales:

- $d|u$ y $d|v$ si y sólo si $d|v$ y $d|(u \bmod v)$, que es lo que se conoce como el invariante de la iteración.
- $\mathrm{mcd}(a,0) = a$, que es la condición de salida de la iteración.

Es claro que este algoritmo termina, ya que $(a \bmod b) < b$, es decir en cada iteración el argumento de la función *Euclid* disminuye. Si suponemos $b > a$ (sin perder generalidad, ya que siempre podemos suponer que un número es mayor que el otro y al mayor lo llamamos b) el algoritmo termina en $O(b)$ iteraciones. Como entragamos al algoritmo la representacion binaria de a y b, el tamaño de la entrada es $O(log(b))$, entonces el tiempo que demora el algoritmo ,que es $O(b)$, es exponencial en el tamaño de la entrada. Queda claro que esta primera aproximación no nos sirve y nesesitamos algo mejor. Veamos una cota mas fina para el algoritmo.

2 Análisis del peor caso para el algoritmo de Euclides

Proposición 1 *Sean u,v con $u > v > 0$ tal que Euclid(u,v) termina en n iteraciones, se tiene que $u \geq F_{n+2}$, $v \geq F_{n+1}$. Donde F_n es la suceción de Fibonacci con $F_0 = 0$ y $F_1 = 1$.*

Dem: Lo demostraremos por inducción sobre n:
Caso base, n=1.
El algoritmo itera solo una vez, es decir $u = kv$ con k entero. Como el algoritmo iteró una vez (es decir no salió en la primera condición $v = 0$) se tiene que $v \geq 1$, entonces, como $u > v$, se tiene$u \geq 2$. Como $F_2 = 1$ y $F_3 = 2$, tenemos lo que buscabamos.
Paso inductivo:
Lo que haremos es suponer que se comple de la segunda iteración hasta la ń$-esima$ y concluiremos que se cumple en la primera también. Suponemos cierto para $n-1$, entonces sabemos que $v > F_{n+1}$ y $(u \bmod v) > F_n$, como $u = \lfloor \frac{u}{v} \rfloor v + (u \bmod v) \geq v + (u \bmod v)$, desigualdad que se comple ya que $u > v$ lo que implica $\frac{u}{v} > 1$. Podemos ver que $u > F_{n+1} + F_n = F_{n+2}$, compliendose lo que buscabamos. □

Ejercicio 1 *Pruebe que*

$$F_n = \frac{1}{\sqrt{5}}\left(\phi^n - \varphi^n\right), \qquad \text{donde } \phi = \frac{1}{\sqrt{2}}(1+\sqrt{5}), \quad \varphi = \frac{1}{\sqrt{2}}(1-\sqrt{5}).$$

Luego, para n suficientemente grande F_n es el entero más cercano a $\phi^n/\sqrt{5}$.

Lema 1 *Sean u, v con $u > v > 0$, y sea $E(u,v)$ el número de iteraciones que toma $Euclid(u,v)$ entonces $E(u,v) < c_1 \log u + c_2 + c_3 u^{-1} - 2$, donde $c_1 = 1/\phi \approx 2.08$, $c_2 = \log\sqrt{5}/\log\phi \approx 1.67$, y $c_3 = (1/\sqrt{5})\log\phi \approx 0.93$. También $E(u,v) < c_1 \log v + c_2 - 2 + c_3 v^{-1}$.*

Dem: Sea $u \geq F_{n+2}$, entonces

$$u \geq \frac{\phi^{n+2} - \varphi^{n+2}}{\sqrt{5}} \geq \frac{\phi^{n+2} - 1}{\sqrt{5}}$$

entonces, usando que $\log x + 1 = \log x + \log(1 + 1/x) \geq \log x + 1/x$ tenemos que

$$\log\sqrt{5}u + \frac{1}{\sqrt{5}u} \geq \log\sqrt{5}u + 1$$

obteniendo

$$\log\sqrt{5}u + \frac{1}{\sqrt{5}u} \geq (n+2)\log\phi$$

despejando n obtenemos que

$$n \geq \frac{\log(\sqrt{5}u)}{\log(\phi)} + \frac{1}{\sqrt{5}\log(\phi)u} - 2$$

concluyendo, luego de agrupar las constanes, que $n \geq c_1 \log u + c_2 - 2 + c_3 u^{-1}$
la otra desigualdad se demuestra de manera similar.

□

Para ver la eficiencia podemos, como primera aproximación, contar el número de multiplicaciones, pero eso no nos sirve, ya que la eficiencia de la multiplicación depende del tamaño de los multiplicandos, entonces, si los números van creciendo, las multiplicaciones del final serán mucho mas costosas que las primeras. Es decir el costo de multiplicar no es constante. Para evitar ese inconveniente, contaremos las operaciones binarias del algoritmo, que si son a tiempo constante. Ya que operaremos con números muy grandes, queremos multiplicar eficientemente. Veremos (en la catedra 4) que calcular uv requiere $O(\log u \log v)$ operaciones binarias.

Proposición 2 *El algoritmo de Euclides, en la entrada (u, v), requiere $O(\log u logv)$ operaciones binarias.*

Dem: Primero definamos una función que mide el largo de un numero o el número de bits que requiere representarlo (sin incluir el bit del signo):

$$tam(n) = \begin{cases} 1 & \text{si } n = 0 \\ (\lfloor log_2 |n| \rfloor) + 1 & \text{si } n > 0 \end{cases}$$

Veamos que podemos reescribir el algoritmo de euclides de la siguiente manera: $u_0 = u$, $u_1 = v$

$$\begin{array}{ll}
u_0 = a_0 u_1 + u_2 & a_0 = \lfloor u_0/u_1 \rfloor \\
u_1 = a_1 u_2 + u_3 & a_1 = \lfloor u_1/u_2 \rfloor \\
\ldots & \\
u_{i-2} = a_{i-2} u_{i-1} + u_i & a_{i-2} = \lfloor u_{i-2}/u_{i-1} \rfloor \\
\ldots & \\
 & \\
u_{n-1} = a_{n-1} u_n & a_0 = \lfloor u_{n-1}/u_n \rfloor
\end{array}$$

Para ejecutar el algoritmo así expresado, en cada iteración debemos calcular a_i, lo que nos cuesta $O(tam(a_i)tam(u_{i+1}))$ operaciones binarias,entonces tenemos un costo total:

$$O\left(\sum_{0 \leq i \leq n-1} tam(a_i) tam(u_{i+1})\right)$$

por otro lado $v = u_1 \geq u_2 \geq u_3 \geq ... \geq u_n$ debido a que $u_{i-2} = a_{i-2}u_{i-1} + u_i > u_i$, y patiendo del caso base $u_0 > u_1$ inductivamente vemos que los u_i decrecen. Usando esto obtenemos que:

$$O\left(\sum_{0\leq i\leq n-1} tam(u_{i+1})tam(a_i)\right) \leq O\left(tam(v)\sum_{0\leq\ i\leq n-1} tam(a_i)\right)$$

Además, por la definición de la función tam se tiene que $tam(a_i) \geq 1 + \log a_i$, con lo que vemos que:

$$O\left(tam(v)\sum_{0\leq\ i\leq n-1} tam(a_i)\right) \leq O\left(tam(v)(n + \log\prod_{0\leq i\leq n-1} a_i)\right)$$

pero $a_i = \lfloor u_i/u_{i+1}\rfloor$, entonces:

$$\prod_{0\leq i\leq n-1}\lfloor\frac{u_i}{u_{i+1}}\rfloor \leq \prod_{0\leq i\leq n-1}\frac{u_i}{u_{i+1}} = \frac{u_0}{u_n} = u$$

entonces

$$O\left(tam(v)(n + \log\prod_{0\leq i\leq n-1} a_i)\right) = O\left(tam(v)(n + \log u)\right)$$

vimos que el numero de iteraciones n es $O(\log u)$ obteniento así, que el algoritmo de Euclides en la entrada u, v realiza

$$O(tam(v)tam(u))$$

operaciones binarias.

□

Proposición 3 *Sean u,v,c enteros,* $au + bv = c$ *tiene solución entera si y sólo si* $mcd(u, v)|c$

Dem: Si la ecuación $au + bv = c$ tiene solución entera para a y b, significa que al dividir $au + bv$ por mcd(u, v) nos resulta un entero, entonces es claro que $mcd(u, v)|c$.

Para la otra implicancia, usaremos el algoritmo de Euclides,

$$u_{i-2} = a_{i-2}u_{i-1} + u_i$$

reescribiendo

$$u_i = -a_{i-2}u_{i-1} + u_{i-2}$$

$$u_{i-1} = u_{i-1}$$

podemos escribir estas ecuaciones de forma matricial

$$\begin{bmatrix} u_i \\ u_{i-1} \end{bmatrix} = \begin{bmatrix} -a_{i-2} & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u_{i-1} \\ u_{i-2} \end{bmatrix}$$

Así expuesto, comenzamos con el vector

$$\begin{bmatrix} u_0 \\ u1 \end{bmatrix}$$

que lo vamos multiplicando en cada paso por una matriz para obtener el nuevo paso de la iteración

$$\begin{bmatrix} u_2 \\ u_1 \end{bmatrix} = \begin{bmatrix} -a_0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_0 \end{bmatrix}$$

Al aplicar iterativamente podemos ver

$$\begin{bmatrix} u_i \\ u_{i-1} \end{bmatrix} = \begin{bmatrix} -a_{i-2} & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} -a_{i-3} & 1 \\ 1 & 0 \end{bmatrix} \dots \begin{bmatrix} -a_0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_0 \end{bmatrix}$$

como sabemos que al terminar el algoritmo $u_n = 0$ y $u_{n-1} = mcd(u, v) = d$,al final del producto de matrices nos queda:

$$\begin{bmatrix} 0 \\ d \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

donde la matriz $M = m_{ij}$ es la multiplicación de las matrices obtenidas en cada iteración, y todos sus elementos son enteros.

Entonces, $d = m_{11}u + m_{12}v$ es decir, existen a y b que cumplen $d = au + bv$, y $d = \text{mcd}(u, v)$.

□

Notemos que esta demostración nos da un método para calcular los coeficientes a y b.

Decimos que dos enteros a y b son *primos relativos* si $\text{mcd}(a, b) = 1$, i.e. si no tienen divisores comunes mayores que 1.

Corolario 1 *Si a y b son primos relativos, entonces existen enteros u y v tales que $1 = u\,a + v\,b$.*

Basandonos en la demostración anterior , veremos un nuevo método:

3 El algoritmo de Euclides extendido

El algoritmo es el siguiente:

$EucludEx(u, v)$
  $M \leftarrow \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
  $n \leftarrow 0$
  while (v$\neq$0) do
    $q \leftarrow \lfloor \frac{u}{v} \rfloor$
    $M \leftarrow M \begin{bmatrix} q & 1 \\ 1 & 0 \end{bmatrix}$
    $(u, v) \leftarrow (v, u - qv)$
    $n \leftarrow n + 1$
  return($d = u, a = (-1)^n M_{22}, b = (-1)^{n+1} M_{12}$)
end:

Este algoritmo, además de retornar el *máximo común divisor* de u y v, nos retorna las constantea a y b que cumplen con la ecuación $au + bv = mcd(u, v)$.