El Algoritmo de Euclides para calcular el máximo común divisor

Pablo L. De Nápoli

Departamento de Matemática
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Álgebra I - Segundo cuatrimestre de 2020

Parte I

El algoritmo de Euclides

El Máximo Común Divisor

El algoritmo de Euclides es un algoritmo para el cálculo del máximo común divisor desarrollado por el matemático griego Euclides (aprox. 325-265 a. C.).

Definición (4.5.1 en el apunte de la profesora Kirck)

Sean $a, b \in \mathbb{Z}$. El máximo común divisor entre a y b, que se nota $(a : b)$ o $mcd(a, b)$, es el mayor de los divisores comunes de a y b . Es decir: $(a : b)|a, (a : b)|b$ y si $d|a$ y $d|b$, entonces $d \leq (a : b)$.

Nota: en el apunte dice que a y b deben ser no nulos, pero la definición tiene sentido si uno de los dos es nulo, de hecho

$$(a : 0) = |a| \quad \forall\ a \in \mathbb{Z}$$

Además

$$(a : b) = (|a| : |b|) \quad \forall\ a, b \in \mathbb{Z}$$

por lo que podemos suponer que $a, b \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

El Algoritmo de Euclides

Para calcular $(a : b)$ podemos suponer $a \geq b$ (sino intercambiamos los roles pues $(a : b) = (b : a)$
Efectuamos entonces la división entera de a por b obteniendo un primer cociente q_1 y un primer resto r_1

$$a = q_1 \cdot b + r_1 \quad 0 \leq r_1 < b$$

Si $r_1 = 0$ el algorimo termina. Sino, dividimos a b por r_1 obteniendo un segundo cociente q_2 y un segundo resto r_2

$$b = q_2 \cdot r_1 + r_2 \quad 0 \leq r_2 < r_1$$

Si $r_2 = 0$ el algorimo termina. Sino, dividimos a r_1 por r_2 obteniendo un tercer cociente q_3 y un tercer resto r_3

$$r_1 = q_3 \cdot r_2 + r_3 \quad 0 \leq r_3 < r_2$$

El algoritmo de Euclides (continuación)

Repitiendo (inductivamente) el procedimiento, supongamos que ya calculamos r_ℓ. Si $r_\ell = 0$ el algoritmo termina. Si no dividimos a $r_{\ell-1}$ por r_ℓ, obteniendo que

$$r_{\ell-1} = q_{\ell+1} \cdot r_\ell + r_{\ell+1} \quad 0 \leq r_{\ell+1} < r_\ell$$

Notamos que por construcción la sucesión de restos

$$r_1 > r_2 > r_3 > \ldots$$

es una sucesión de enteros no negativos estrictamente decreciente. Por el principio del mínimo entero, se deduce que esta sucesión no puede continuar indefinidamente. En consecuencia, siempre existe un $N \in \mathbb{N}$ tal que

$$r_N = 0$$

es decir que el algoritmo de Euclides siempre termina.

El algoritmo de Euclides (continuación)

Teorema (4.5.3 en el apunte de la profesora Kirck)

El algoritmo de Euclides siempre termina y el último resto no nulo r_{N-1} que se obtiene es el máximo común divisor entre a y b.

Ejemplo: Calculemos el máximo común divisor entre 32 y 17 por el algoritmo de Euclides:

$$32 = 1 * 17 + 15$$
$$17 = 1 * 15 + 2$$
$$15 = 7 * 2 + 1$$
$$2 = 2 * 1 + 0$$

El algoritmo de Euclides termina: El mcd entre 32 y 17 es 1.

El invariante del Algoritmo de Euclides

Para demostrar el teorema, usamos el siguiente lema que expresa un invariante del algoritmo (una propiedad que se mantiene a lo largo de las iteraciones del algoritmo).

Propisición (4.5.2 en el apunte de la profesora Kirck)

Si efectuamos la división entera de a por b, obteniendo un cociente q y un resto r, entonces

$$(a : b) = (b : r)$$

La prueba del lema es inmediata: si d es un divisor común entre a y b, entonces d también divide a $r = a - q \cdot b$, en consecuencia d es un divisor común entre b y r.

Recíprocamente si d divide a b y r, también dividirá a a pues $a = q \cdot b + r$. Se deduce que b y r tienen los mismos divisores comunes que los que tenían a y b, y en consecuencia tienen el mismo máximo común divisor.

Prueba de la correctitud del Algoritmo de Euclides

Usando el lema vemos que la sucesión de restos construida por el algoritmo de Euclides, cumple que

$$(a : b) = (b : r_1) = (r_1 : r_2) = (r_2 : r_3) = \dots$$

$$\dots = (r_{N-2} : r_{N-1}) = (r_{N-1} : 0) = r_{N-1}$$

(pues para cualquier entero $c \in \mathbb{N}_0$, $\text{mcd}(c, 0) = c$)

Es decir que hemos demostrado que el algoritmo de Euclides calcula correctamente el máximo común divisor.

Bonus track: ¡lo progamamos!

Programita recursivo (en Python 3) para el Algorimo de Euclides

```
def mcd(a,b):
  if b>a:
    return mcd(b,a)
  if b==0:
    print ("El algoritmo de Euclides termina!")
    print ("¡El máximo común divisor es",a)
    return a
  else:
    q, r= divmod(a,b)
    print(a,"=",q,"*",b,"+",r)
    print ("mcd(",a,",",b,")=mcd(",b,",",r,")")
    return mcd(b,r)
```

La salida del programa

Ejemplo de una salida del programa

```
Calculamos el máximo común divisor entre  360  y  46
360 = 7 * 46 + 38
mcd( 360 , 46 )=mcd( 46 , 38 )
46 = 1 * 38 + 8
mcd( 46 , 38 )=mcd( 38 , 8 )
38 = 4 * 8 + 6
mcd( 38 , 8 )=mcd( 8 , 6 )
8 = 1 * 6 + 2
mcd( 8 , 6 )=mcd( 6 , 2 )
6 = 3 * 2 + 0
mcd( 6 , 2 )=mcd( 2 , 0 )
¡El algoritmo de Euclides termina!
El máximo común divisor es 2
```

Parte II

Variantes del algoritmo de Euclides

El algoritmo de Euclides, según Euclides

Euclid's Elements

Book VII

Proposition 2

To find the greatest common measure of two given numbers not relatively prime.

Let *AB* and *CD* be the two given numbers not relatively prime.

It is required to find the greatest common measure of *AB* and *CD.*

If now *CD* measures *AB,* since it also measures itself, then *CD* is a common measure of *CD* and *AB.* And it is clear that it is also the greatest, for no greater number than *CD* measures *CD.*

But, if *CD* does not measure *AB,* then, when the less of the numbers *AB* and *CD* being continually subtracted from the greater, some number is left which measures the one before it.

For a unit is not left, otherwise *AB* and *CD* would be relatively prime, which is contrary to the hypothesis. VII.Def.12 VII.1

Therefore some number is left which measures the one before it.

Now let *CD,* measuring *BE,* leave *EA* less than itself, let *EA,* measuring *DF,* leave *FC* less than itself, and let *CF* measure *AE.*

Since then, *CF* measures *AE,* and *AE* measures *DF,* therefore *CF* also measures *DF.* But it measures itself, therefore it also measures the whole *CD.*

But *CD* measures *BE,* therefore *CF* also measures *BE.* And it also measures *EA,* therefore it measures the whole *BA.*

But it also measures *CD,* therefore *CF* measures *AB* and *CD.* Therefore *CF* is a common measure of *AB* and *CD.*

I say next that it is also the greatest.

If *CF* is not the greatest common measure of *AB* and *CD,* then some number *G,* which is greater than *CF,* measures the numbers *AB* and *CD.*

Now, since *G* measures *CD,* and *CD* measures *BE,* therefore *G* also measures *BE.* But it also measures the whole *BA,* therefore it measures the remainder *AE.*

But *AE* measures *DF,* therefore *G* also measures *DF.* And it measures the whole *DC,* therefore it also measures the remainder *CF,* that is, the greater measures the less, which is impossible.

Therefore no number which is greater than *CF* measures the numbers *AB* and *CD.* Therefore *CF* is the greatest common measure of *AB* and *CD.*

Q.E.D.

http://aleph0.clarku.edu/~djoyce/java/elements/bookVII/propVII2.html // Invariante: $(a : b) = (b : a - b)$

El algoritmo de Euclides binario

La siguiente es una variante del algoritmo de Euclides que sólo utiliza divisiones por 2, lo que resulta ventajoso si se opera con números escritos en el sistema binario (como sucede en una computadora), ya las divisiones se pueden efectuar mediante operaciones de shift (corrimiento de los dígitos hacia la derecha).

El algoritmo puede describirse recursivamente de la siguiente manera:

$$\operatorname{mcd}(u,v) := \begin{cases} u & \text{si } v = 0 \\ 2\operatorname{mcd}\left(\frac{u}{2}, \frac{v}{2}\right) & \text{si u es par y v par} \\ \operatorname{mcd}\left(\frac{u}{2}, v\right) & \text{si u es par y v impar} \\ \operatorname{mcd}\left(u, \frac{v}{2}\right) & \text{si u es impar y v par} \\ \operatorname{mcd}\left(v, \frac{u-v}{2}\right) & \text{si u es impar y v impar} \end{cases}$$

Ejercicio: Programarlo y justificar porqué funciona.

Parte III

Una consecuencia del algoritmo de Euclides: El máximo común divisor se escribe como combinación lineal

El MCD como combinación lineal (identidad de Bézout)

Una consecuencia muy importante del algoritmo de Euclides es el siguiente teorema:

Teorema (4.4.5 en el apunte de la profesora Kirck)

El máximo común divisor entre a y b se puede escribir como una combinación lineal de ellos: es decir, existen enteros $s = s(a, b)$ y $t = t(a, b)$ tales que

$$s \cdot a + t \cdot b = (a : b)$$

El algoritmo de Euclides nos permite dar una prueba constructiva de este teorema. Les voy a presentar una versión matricial siguiendo un artículo de mi colega Antonio Cafure. Esta forma de presentarlo es diferente de las quen están en el apunte de la profesora Kirch, y también en el mio. Pero tiene la ventaja que es más fácil recordar cuál es el algoritmo.

Matrices y sistemas de ecuaciones lineales de 2x2

Consideremos un sistema lineal de 2 ecuaciones con 2 incógnitas

$$\begin{cases} m_{11} \cdot x + m_{12} \cdot y & = & m_{13} \\ m_{21} \cdot x + m_{22} \cdot y & = & m_{23} \end{cases}$$

donde los m_{ij} son números dados. le podemos asociar su matriz ampliada de tamaño 2×3:

$$\left(\begin{array}{cc|c} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{array} \right)$$

Por ejemplo al sistema:

$$\begin{cases} 2 \cdot x + 3 \cdot y & = & 13 \\ 8 \cdot x - 5 \cdot y & = & 1 \end{cases}$$

le corresponde la matriz

$$\left(\begin{array}{rr|r} 2 & 3 & 13 \\ 8 & -5 & 1 \end{array} \right)$$

Es una forma compacta de organizar la información sobre el sistema.

Operaciones elementales por filas

Si a una ecuación del sistema/fila de la matriz le sumamos o restamos un múltiplo (no nulo) de otra fila obtenmos un sistema equivalente. Es decir que tendrá las mismas soluciones.

Por consideramos sistema:

$$\begin{cases} 2 \cdot x + 3 \cdot y & = & 13 \\ 8 \cdot x - 5 \cdot y & = & 1 \end{cases} \qquad \begin{pmatrix} 2 & 3 & 13 \\ 8 & -5 & 1 \end{pmatrix}$$

(cuya única solución es $x = 2, y = 3$) y le hacemos la operación $\text{Fila}_1 \leftarrow \text{Fila}_1 - 2 \cdot \text{Fila}_2$ obtenemos el sistema/matriz equivalente:

$$\begin{cases} -14 \cdot x + 13 \cdot y & = & 11 \\ 8 \cdot x - 5 \cdot y & = & 1 \end{cases} \qquad \left(\begin{array}{rr|r} -14 & 13 & 11 \\ 8 & -5 & 1 \end{array}\right)$$

con la misma solución.

Idea del algoritmo

Ahora volvamos a nuestro problema: dados dos números a y b queremos expresar a su máximo común divisor como una combinación lineal $s \cdot a + t \cdot b = d$ de ellos.
La idea para hacerlo será considerar el sistema

$$\begin{cases} 1 \cdot x + 0 \cdot y &= a \\ 0 \cdot x + 1 \cdot y &= b \end{cases} \qquad \left(\begin{array}{cc|c} 1 & 0 & a \\ 0 & 1 & b \end{array} \right)$$

cuya única solución es evidentemente $x = a$, $y = b$ y usar el algoritmo de Euclides y operaciones elementales por fila para obtener sucesivamente sistemas/matrices equivalentes en las que aparecen los restos r_k en la última columna. Hasta que enventualmente obtenemos una matriz con un cero en la última columna.

Cuando ello ocurra, podremos leer en la fila correspondiente quienes son el mcd y los coeficientes s, t de la combinación lineal

Veamos un ejemplo

Como antes apliquemos el algoritmo de Euclides para hallar el máximo común divisor entre $a = 360$ y $b = 46$.

- Paso 0: Empezamos por la matriz

$$\left(\begin{array}{cc|c} 1 & 0 & 360 \\ 0 & 1 & 46 \end{array}\right)$$

- Paso 1: Dividimos a 360 por 46. $360 = 7 \times 46 + 38$.
  Hacemos entonces la operación $\text{Fila}_1 \leftarrow \text{Fila}_1 - 7 \cdot \text{Fila}_2$ a la matriz.
  Notamos que en la tercer columna aparece el primer resto del algoritmo de Euclides $r_1 = 38$.

$$\Rightarrow \quad \left(\begin{array}{cc|c} 1 & -7 & 38 \\ 0 & 1 & 46 \end{array}\right)$$

Veamos un ejemplo (2)

- Paso 2: Ahora dividimos a $b = 46$ por $r_1 = 38$. Obtenemos $46 = 1 \times 38 + 8$. Hacemos entonces la operación $\text{Fila}_2 \leftarrow \text{Fila}_2 - 1 \cdot \text{Fila}_1$

$$\Rightarrow \quad \left(\begin{array}{rr|r} 1 & -7 & 38 \\ 0 & 1 & 46 \end{array}\right) \quad \Rightarrow \quad \left(\begin{array}{rr|r} 1 & -7 & 38 \\ -1 & 8 & 8 \end{array}\right)$$

- Paso 3: Continuando, $38 = 4 \times 8 + 6$. Hacemos $\text{Fila}_1 \leftarrow \text{Fila}_1 - 4 \cdot \text{Fila}_2$

$$\Rightarrow \quad \left(\begin{array}{rr|r} 5 & -39 & 6 \\ -1 & 8 & 8 \end{array}\right)$$

- Paso 4: Ahora $8 = 1 \times 6 + 2$. Hacemos $\text{Fila}_2 \leftarrow \text{Fila}_2 - 1 \cdot \text{Fila}_1$

$$\Rightarrow \quad \left(\begin{array}{rr|r} 5 & -39 & 6 \\ -6 & 47 & 2 \end{array}\right)$$

Veamos un ejemplo (3): Finalmente ...

- Paso 5: Finalmente $6 = 3 \times 2 + 0$. Acá como ya sabemos, el algoritmo de Euclides termina. El último resto no nulo 2 es el máximo común divisor. $\text{Fila}_1 \leftarrow \text{Fila}_1 - 3 \cdot \text{Fila}_2$ Nos queda la matriz

$$\left(\begin{array}{rr|r} 5 & -39 & 6 \\ -6 & 47 & 2 \end{array} \right) \Rightarrow \left(\begin{array}{rr|r} 23 & -180 & 0 \\ -6 & 47 & 2 \end{array} \right)$$

Es decir que el sistema original es quivalente a

$$\begin{cases} 23 \cdot x - 180 \cdot y & = & 0 \\ -6 \cdot x + 47 \cdot y & = & 2 \end{cases}$$

Pero sabemos que $x = 360$ e $y = 46$ es la solución de este sistema (porque lo era del original) luego

$$(-6) \cdot 360 + 47 \cdot 46 = 2$$

Esto muestra que el máximo común divisor se escribe como combinación lineal de 360 y 46.

Demostración del teorema en general

En general, empezamos con la matriz

$$\left(\begin{array}{cc|c} 1 & 0 & a \\ 0 & 1 & b \end{array} \right)$$

correspondiente a un sistema de ecuaciones cuya única solución es $x = a$, $y = b$ Aplicando el algoritmo de Euclides obtenemos una sucesión de cocientes (q_ℓ) y una sucesión de restos (r_ℓ)

$$r_1 > r_2 > r_3 > \ldots > r_{N-1} > r_N = 0$$

Haciendo alertenadamente las operaciones elementales por filas

$$\text{Fila}_1 \leftarrow \text{Fila}_1 - q_\ell \cdot \text{Fila}_2 \quad \ell \text{ impar}$$

$$\text{Fila}_2 \leftarrow \text{Fila}_2 - q_\ell \cdot \text{Fila}_1 \quad \ell \text{ par}$$

obtendremos sucesivamente matrices equivalentes por filas a la orginal, donde los restos (r_k) aparecen en la última columna.

Demostración del teorema en general

Es decir, matrices de la forma:

$$\left(\begin{array}{cc|r} \cdot & \cdot & r_\ell \\ \cdot & \cdot & r_{\ell-1} \end{array} \right) \quad \text{o} \quad \left(\begin{array}{cc|r} \cdot & \cdot & r_{\ell-1} \\ \cdot & \cdot & r_\ell \end{array} \right)$$

Como vimos antes, el algoritmo de Euclides siempre termina (con $r_N = 0$) y obtenemos entonces una matriz de la forma

$$\left(\begin{array}{cc|r} \cdot & \cdot & 0 \\ s & t & r_{N-1} \end{array} \right) \quad \text{o} \quad \left(\begin{array}{cc|r} s & t & r_{N-1} \\ \cdot & \cdot & 0 \end{array} \right)$$

En esta situación, ya sabemos que $r_{N-1} = (a, b)$. Y como $x = a$, $y = b$ es la solución del sistema asociado

$$s \cdot a + t \cdot b = r_{N-1} = (a : b)$$

Es decir que el máximo común divisor entre a y b se escribe como una combinación lineal de ellos.

Bonus track: ¡lo progamamos! (en Python 3/Sagemath)

Programita para el Algorimo de Euclides Extendido

```
def Euclides_extendido(a,b):
        paso =0
        m=matrix([[1,0,a],[0,1,b]])
        while True:
                paso=paso+1
                print("paso ",paso,"\n",m)
                if m[0,2]==0:
                        mcd=m[1,2]
                        s = m[1,0]
                        t = m[1,1]
                        break
                elif m[1,2]==0:
                        mcd=m[0,2]
                        s = m[0,0]
                        t = m[0,1]
                        break
```

Bonus track: ¡lo progamamos! (2)

Programita (en Python 3) para el Algorimo de Euclides Extendido

```
        elif m[0,2] >= m[1,2]:
            q,r=divmod(m[0,2],m[1,2])
# Restamos a la primer fila q veces la segunda
            m[0,0] = m[0,0] - q*m[1,0]
            m[0,1] = m[0,1] - q*m[1,1]
            m[0,2] = r
        else:
            q,r=divmod(m[1,2],m[0,2])
# Restamos a la segunda fila q veces la primera
            m[1,0] = m[1,0] - q*m[0,0]
            m[1,1] = m[1,1] - q*m[0,1]
            m[1,2] = r
        return (s,t,mcd)
```

Otro ejemplo del algoritmo de Euclides Extendido

Calculamos el máximo común divisor entre $a = 120$ y $b = 37$.

Ejemplo de corrida del programa

```
sage: Euclides_extendido(120,37)
paso  1
[  1   0 120]
[  0   1  37]
paso  2
[ 1 -3  9]
[ 0  1 37]
paso  3
[ 1 -3  9]
[-4 13  1]
paso  4
[  37 -120    0]
[  -4   13    1]
(-4, 13, 1)
```

Parte IV

Bonus track: ¿cuánto tarda el algoritmo de Euclides?

Los números de Fibonacci

Recordamos que los números de Fibonacci se definen recursivamente por

$$F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2} \quad (n \geq 2)$$

Es fácil escribir un programa para generar los números de Fibonacci

Calculamos los números de Fibonacci

```
def fibo(n):
        if n==0:
                return 0
        elif n==1:
                return 1
        else:
                return fibo(n-1)+fibo(n-2)

for k in range(0,16):
        print ("fibo(",k,")=",fibo(k))
```

Tabla de los números de Fibonacci

```
fibo( 0 )= 0
fibo( 1 )= 1
fibo( 2 )= 1
fibo( 3 )= 2
fibo( 4 )= 3
fibo( 5 )= 5
fibo( 6 )= 8
fibo( 7 )= 13
fibo( 8 )= 21
fibo( 9 )= 34
fibo( 10 )= 55
fibo( 11 )= 89
fibo( 12 )= 144
fibo( 13 )= 233
fibo( 14 )= 377
fibo( 15 )= 610
```

¿Qué pasa si aplicamos el algoritmo de Euclides a dos números de Fibonacci consecutivos?

Calculemos $\operatorname{mcd}(F_k, F_{k+1})$ para distintos valores de k usando nuestros programitas:
Por ejemplo si $k = 11$:

```
a=fibo( 12 )= 144
b=fibo( 11 )= 89
144 = 1 * 89 + 55
89 = 1 * 55 + 34
55 = 1 * 34 + 21
34 = 1 * 21 + 13
21 = 1 * 13 + 8
13 = 1 * 8 + 5
8 = 1 * 5 + 3
5 = 1 * 3 + 2
3 = 1 * 2 + 1
2 = 2 * 1 + 0
```

Generalizando....

Notamos que:

- Se requieren exactamente $k-1$ pasos para calcular $\text{mcd}(F_k, F_{k+1})$ usando el algoritmo de Euclides.
- Todos los cocientes son 1 (salvo el último) y los restos son los números de Fibonacci.
- Se puede ver que este es el peor caso del algoritmo de Euclides, o sea: que si $b \leq a \leq F_{k+1}$ el algoritmo va a tardar menos. (Porque los cocientes son q_j por lo menos 1, y si son más grandes, el algoritmo tarda menos)

La complejidad del algoritmo de Euclides

- Pero

$$F_{k+1} \simeq \frac{1}{\sqrt{5}}\, \Phi^{k+1}$$

si k es grande [por la proposición 2.5.5 del apunte de la profesora Kirck], donde

$$\Phi = \frac{1+\sqrt{5}}{2} \approx 1,61803$$

es el número de oro . Se deduce que si a es grande y $b \leq a$, el número de divisiones que requiere el algoritmo de Euclides para calcular $\operatorname{mcd}(a,b)$ es aproximadamente (en el peor caso)

$$\frac{\log\ a}{\log\ \Phi}$$

(¡Esto es muy bueno!, tiene complejidad lineal como función del número de dígitos de a)