61-ARITM-euclides

December 3, 2017

Algoritmo de Euclides

```
In [1]: def mcd_i(a,b):
            if a < b:
                return mcd_i(b,a)
            while a%b != 0:
                a,b = b,a%b
            return b
```

```
In [2]: mcd_i(2*7*11*31,9*44*7)
```

```
Out[2]: 154
```

```
In [3]: def mcd_r(a,b):
            if a < b:
                return mcd_r(b,a)
            elif b == 0:
                return a
            else:
                return mcd_r(b,a%b)
```

```
In [4]: mcd_r(2*7*11*31,9*44*7)
```

```
Out[4]: 154
```

Teorema de Bezout

```
In [5]: def bezout_r(a,b):
            if a < b:
                y,x = bezout_r(b,a)
                return x,y
            elif b == 0:
                return 1,-1
            else:
                c = a//b
                r = a%b
                x,y = bezout_r(b,r)
                return y,x-c*y
```

In [6]: bezout_r(3*5,2*5)

Out[6]: (3, -4)

In [7]: 3*5*3-4*5*2

Out[7]: 5

In [8]: %time bezout_r(1234567899678,345678998765432)

```
CPU times: user 0 ns, sys: 0 ns, total: 0 ns
Wall time: 36 ţs
```

Out[8]: (-180629316861549, 645104727607)

In [9]:
```python
def bezout_i(a,b):
    if a < b:
        y,x = bezout_i(b,a)
        return (x,y)
    L = [(a,b,a//b,a%b)]
    while a%b != 0:
        a,b = b,a%b
        L.append((a,b,a//b,a%b))
        if a%b == 0:
            L.append((b,0,b,0))
    ##print L
    return procesar(L)
```

In [10]:
```python
def procesar(L):
    x,y = 1,-1
    while len(L) > 0:
        x,y = y,x-(L[-1:][0][2])*y
        L = L[:-1]
    return (x,y)
```

In [11]: bezout_i(3*5,2*5)

Out[11]: (-13, 19)

In [12]: -13*3*5+19*2*5

Out[12]: -5

In [13]: %time bezout_i(1234567899678,345678998765432)

```
CPU times: user 0 ns, sys: 0 ns, total: 0 ns
Wall time: 111 ţs
```

Out[13]: (526308315626981, -1879672627285)

En esta primera solución del "bezout iterativo" copiamos el procedimiento que realizamos a mano y que se explica en las notas. Hay una solución iterativa mucho más elegante que utiliza matrices 2×2. La base del método, como en las otras soluciones, es la observación siguiente:

Si $a = b \cdot c_0 + r_0$ y sabemos que $mcd(a, b) = d$ y $d = x \cdot a + y \cdot b$ entonces

$$d = x \cdot a + y \cdot b = x \cdot (b \cdot c_0 + r_0) + y \cdot b = (x \cdot c_0 + y)b + x \cdot r_0$$

de forma que se pueden obtener los coeficientes de b y r_0 en la expresión que da d multiplicando una matriz por los coeficientes de a y b:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} c_0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

```
In [34]: def bezout_i2(a,b):
             M = matrix(ZZ,[[1,0],[0,1]]) #matriz identidad para inicializar M
             while a%b != 0:
                 M = matrix(ZZ,[[a//b,1],[1,0]])*M
                 a,b = b,a%b
             return M.inverse()*matrix(ZZ,[[0,1]]).transpose()
```

```
In [35]: bezout_i2(385,66)
```

```
Out[35]: [-1]
         [ 6]
```

```
In [36]: -385+6*66 == mcd_i(385,66)
```

```
Out[36]: True
```

```
In [37]: xgcd(385,66)
```

```
Out[37]: (11, -1, 6)
```

```
In [23]: A = matrix(SR,[[x,1],[1,0]])
         B = A.inverse()
         show(B)
```

```
[ 0  1]
[ 1 -x]
```

```
In [ ]:
```