63-ARITM-num-div-euclides

December 3, 2017

Ejercicio 1

```
In [2]: def num_div(a,b):
            a0,b0 = a,b
            contador = 0
            if a < b:
                return num_div(b,a)
            while a%b != 0:
                a,b = b,a%b
                contador += 1
            return contador,a0,b0,b,
```

```
In [2]: %time num_div(134768,67)

CPU times: user 0 ns, sys: 0 ns, total: 0 ns
Wall time: 18.8 µs


Out[2]: (3, 134768, 67, 1)
```

```
In [3]: %time L1 = [num_div(a,b) for b in xrange(1,1000) for a in xrange(b+1,10000)]

CPU times: user 27.6 s, sys: 725 ms, total: 28.4 s
Wall time: 28.1 s


In [4]: max([item[0] for item in L1])

Out[4]: 14
```

```
In [5]: %time L2 = [num_div(a,b) for b in xrange(11,99) for a in xrange(b+1,1000)]

CPU times: user 187 ms, sys: 8.21 ms, total: 195 ms
Wall time: 193 ms


In [6]: max([item[0] for item in L2])

Out[6]: 9
```

Definimos una función que hace lo mismo, pero ahorra RAM ya que no define listas grandes:

```
In [7]: def max_div(N0,N1,N2):
            max = 1;
            for b in xrange(N0,N1):
                for a in xrange(b,N2):
                    if num_div(a,b)[0] > max:
                        max = num_div(a,b)[0]
            return max
```

```
In [8]: %time max_div(1,999,5000)
```

```
CPU times: user 12.9 s, sys: 165 ms, total: 13.1 s
Wall time: 12.9 s
```

```
Out[8]: 14
```

```
In [5]: %time print max([num_div(a,b)[0] for b in xrange(100,999) for a in xrange(b+1,5000)])
```

```
14
CPU times: user 12.7 s, sys: 128 ms, total: 12.8 s
Wall time: 12.8 s
```

Los dos métodos tardan lo mismo, en este rango de valores, pero el segundo debe utilizar menos RAM. Modificamos la función para guardar resultados intermedios.

```
In [9]: def max_div2(N0,N1,N2):
            max = 1;
            L = []
            for b in xrange(N0,N1):
                for a in xrange(b,N2):
                    if num_div(a,b)[0] == max:
                        L.append((num_div(a,b)[1],num_div(a,b)[2]))
                    elif num_div(a,b)[0] > max:
                        L = [(num_div(a,b)[1],num_div(a,b)[2])]
                        max = num_div(a,b)[0]
            return max,L
```

```
In [10]: %time max_div2(10,99,1000)
```

```
CPU times: user 454 ms, sys: 90.2 ms, total: 545 ms
Wall time: 428 ms
```

```
Out[10]: (9,
          [(144, 89),
           (233, 89),
           (322, 89),
```

```
            (411, 89),
            (500, 89),
            (589, 89),
            (678, 89),
            (767, 89),
            (856, 89),
            (945, 89)])
```

```
In [11]: num_div(233,89)
```

```
Out[11]: (9, 233, 89, 1)
```

```
In [12]: num_div(322,89)
```

```
Out[12]: (9, 322, 89, 1)
```

```
In [13]: def mcd_i2(a,b):
             if a < b:
                 return mcd_i2(b,a)
             while a%b != 0:
                 print a,b,a//b,a%b
                 a,b = b,a%b
             return b
```

```
In [14]: mcd_i2(144,89)
```

```
144 89 1 55
89 55 1 34
55 34 1 21
34 21 1 13
21 13 1 8
13 8 1 5
8 5 1 3
5 3 1 2
3 2 1 1
```

```
Out[14]: 1
```

```
In [15]: mcd_i2(233,89)
```

```
233 89 2 55
89 55 1 34
55 34 1 21
34 21 1 13
21 13 1 8
13 8 1 5
8 5 1 3
5 3 1 2
3 2 1 1
```

```
Out[15]: 1
```

```
In [16]: mcd_i2(322,89)
```

```
322 89 3 55
89 55 1 34
55 34 1 21
34 21 1 13
21 13 1 8
13 8 1 5
8 5 1 3
5 3 1 2
3 2 1 1
```

```
Out[16]: 1
```

£Cuál es nuestra conclusión?