El algoritmo de Euclides

Euclides fue un matemático griego que vivió entre los siglos IV y III antes de Cristo. Su obra "Elementos", en 13 volúmenes, es probablemente el mayor *best-seller* de la historia de las matemáticas.

En esta obra describió por primera vez el algoritmo (conocido hoy con su nombre) que permite calcular, de modo sencillo, el máximo común divisor (MCD) de dos números naturales. Se trata, sin duda, de uno de los algoritmos más antiguos que, aun hoy en día, se sigue utilizando (se usa, por ejemplo, en la simplificación de fracciones y en algoritmos relacionados con la criptografía).

Este algoritmo se basa en el hecho de que el MCD de dos números no varía si se reemplaza el mayor de ellos por su diferencia con el otro. Repitiendo este proceso, se reduce en cada paso el mayor de ambos números, de modo que, en algún momento, necesariamente, uno llegará a ser *0*; en ese momento, el otro es el MCD de los dos números originales.

Utilice directamente la versión original de Euclides para implementar, de modo recursivo, en OCaml una función ***mcd: int -> int > int*** tal que *mcd x y* coincida con el MCD de *x* e *y* (siempre que al menos uno de ellos sea mayor que *0)*. Intente, como de costumbre, que la definición sea lo más sencilla posible; pero, como se ha dicho antes, utilice directamente la versión del algoritmo que acabamos de describir.

Si la definición de mcd es correcta, deberían obtenerse resultados como los siguientes

```
# mcd 1 1;;
- : int = 1
# mcd 12 4;;
- : int = 4
# mcd 12 20;;
- : int = 4
# mcd 1716 105;;
- : int = 3
# mcd 31 30;;
- : int = 1
# mcd 6589923 167745;;
- : int = 3
# mcd 101 101;;
- : int = 101
# mcd 0 7;;
- : int = 7
```

Esta primera versión del algoritmo puede requerir muchos pasos para llegar al resultado. Por ejemplo, el cálculo de mcd 1 200_000_000 llevaría 200 millones de pasos. A pesar de la enorme velocidad de los procesadores actuales que pueden realizar miles de millones de operaciones básicas por segundo, en mi ordenador, este cálculo lleva unos 2 segundos. Compruebe cuánto tiempo consume ese mismo cálculo en su equipo.

A esa velocidad, un cálculo extremo como *mcd 1 max_int* (con ints de 63 bits) llevaría ¡más de 1.000 años!

Una versión mucho más eficiente del algoritmo consistiría en reemplazar, en cada paso, el mayor de ambos números por el resto de su división por el menor. Esto equivaldría a restarle al mayor tantas veces el menor como sea posible, pero en un solo paso.

En 1844 Gabriel Lamé publicó un teorema[1] donde demostraba que, de esta manera, el número necesario de pasos para completar el algoritmo de Euclides es menor que 5 veces el número de cifras (en decimal) del menor de ambos números. Según esto, el MCD de 1 y cualquier otro número se terminaría inmediatamente (lo cual es obvio). Un algoritmo implementado siguiendo esta mejora, si trabajamos con un tipo int de 63 bits, terminaría, por tanto, siempre en menos de 95 pasos (ya que *max_int,* con ints de 63 bits, tiene 19 cifras decimales); lo cual quiere decir que, a nuestra vista, el resultado sería prácticamente instantáneo.

Defina en OCaml una función ***mcd' : int -> int -> int*** aprovechando esta mejora del algoritmo y compruebe, luego, que ahora todas sus aplicaciones son prácticamente instantáneas.

Las definiciones de *mcd* y *mcd'* deben incluirse en una archivo con nombre *mcd.ml.*

Opcional: Defina en un archivo ***mcd_pasos.ml*** una función ***mcd_pasos : int -> int -> int * int*** que calcule el MCD según el algoritmo de Euclides optimizado y devuelva un par de enteros: la primera componente el valor del MCD y la segunda el número de pasos necesarios para calcularlo (esto es, el número de veces que se aplicó la función).

Esta función debería devolver (paso arriba, paso abajo) valores como los siguientes:

```
# mcd_pasos 1 1;;
- : int * int = (1, 2)
# mcd_pasos 12 4;;
- : int * int = (4, 2)
# mcd_pasos 12 20;;
- : int * int = (4, 4)
# mcd_pasos 1716 105;;
- : int * int = (3, 5)
# mcd_pasos 1 max_int;;
- : int * int = (1, 2)
# mcd_pasos 927 573;;
- : int * int = (3, 11)
# mcd_pasos 902685 557890;;
- : int * int = (5, 23)
# mcd_pasos 99012591 61695194;;
- : int * int = (1, 32)
# mcd_pasos 64251823723465 37728299599179;;
- : int * int = (7, 48)
# mcd_pasos 10610209857723 17167680177565;;
- : int * int = (1, 64)
# mcd_pasos 1779979416004714189 2880067194370816120;;
- : int * int = (1, 89)
```

[1] Esta demostración establece el comienzo del estudio de la Teoría de la Complejidad Computacional