Más Hanoi

Para jugar a “Torres de Hanoi” con *n* discos se utilizan *3* postes y *n* discos, todos ellos de diferentes diámetros, agujereados por el centro, de modo que pueden ser insertados en los postes. Los postes tienen altura suficiente para que puedan ser insertados todos los discos en cualquiera de ellos. Para distinguir los *3* postes, utilizaremos los términos *izquierdo*, *central* y *derecho*. Los discos se suponen numerados de *1* a *n* en orden de diámetro creciente. Un *estado estable* es aquel en el que todos los discos están insertados en los postes de modo que no hay ninguno que descanse sobre otro de diámetro menor.

Un *movimiento legal* es aquel que pasa de un estado estable a otro, desplazando, únicamente, el disco en la cima de uno de los postes sobre la cima de otro.

El juego clásico consiste en encontrar una secuencia de movimientos legales, de modo que, partiendo de un estado estable en el que todos los discos están insertados en el mismo poste, se llegue a otro estado estable en el que todos los discos están insertados en otro poste.

Como hay gente que es capaz incluso de realizar los movimientos a ciegas, hoy usaremos una variante del juego en la que buscaremos la secuencia mínima de movimientos legales que nos lleve de un estado estable cualquiera dado a otro estado cualquiera dado.

Para representar un estado utilizaremos una terna *(i,c,d)* donde *i, c* y *d* son, respectivamente, las listas que enumeran (de la cima a la base) los discos insertados, respectivamente, en los postes izquierdo, central y derecho.

Para distinguir los movimientos que llevan un disco de la cima de un poste a la cima de otro, utilizaremos los valores del tipo *move*, definido en OCaml a continuación (y cuya interpretación debería resultar bastante obvia; *CtoL* representaría el movimiento que pasa un disco del poste central al izquierdo)

type move = LtoC | LtoR | CtoL | CtoR | RtoL | RtoC

Se pide definir en OCaml una función

hanoi : int -> int list * int list * int list -> int list * int list * int list -> move list

de modo que *hanoi n inicial final* devuelva la secuencia mínima de movimientos para resolver esta variante de “Torres de Hanoi” con *n* discos, partiendo del estado *ini* con el objetivo de dejar los discos en el estado fin. Si *inicial* o *final* no representan estados estables para el juego “Torres de Hanoi” con *n* discos, la función debe provocar una excepción *Invalid_argument*.

Así por ejemplo debería resultar

```
# hanoi 4 ([1;3], [2; 4], []) ([1;2;4], [3], []);;
- : mov list = [LtoC; LtoR; CtoL; CtoR; LtoR; CtoL; RtoC; RtoL; CtoL; RtoC]
```

La definición de esta función *hanoi* debe incluirse en un archivo ***hanoi_plus.ml***, acompañada de la definición proporcionada para el tipo *move*. El archivo debe compilar correctamente con la orden *ocamlc -c hanoi_plus.mli hanoi_plus.ml* (con el archivo *hanoi.mli* proporcionado)

Nota sobre el rendimiento esperado: Se espera que la función hanoi sea capaz de resolver los juegos de 24 piezas en un tiempo similar al que necesitaría la siguiente definición para calcular el término 45 de la sucesión de Fibonacci (compiladas ambas a bytecode y ejecutadas en una máquina con al menos 2 GB de memoria disponibles)[1]

let rec fib n = if n <= 2 then n else fib (n-1) + fib (n-2)

[1] En código nativo este tiempo podría estar más próximo al del término 47.
Probablemente haya bastante margen para mejorar este rendimiento.