Subárboles (Bintree)

Aunque el concepto de *subgrafo* es coherente de una fuente a otra, no sucede lo mismo con el concepto de *subárbol*.

Dado que un árbol[1] no es más que un determinado tipo de grafo (i.e. un grafo que satisface ciertas propiedades bien conocidas), nosotros entenderemos que un *subárbol* de un árbol dado, *t*, es cualquier subgrafo de *t* que sea (por sí mismo) un árbol. Pensado de un modo más gráfico o intuitivo, sería cualquier árbol que pueda obtenerse simplemente borrando, si acaso, algunos de los nodos (y las aristas correspondiente) del árbol *t*. También puede pensarse que un *subárbol de t* es cualquier subgrafo conexo de *t*.

Esta es la definición que utilizaremos en todos los ejercicios de esta práctica.

Distinguiremos también los siguientes tipos (o clases) de subárboles:

- ***Subárbol raíz***. Diremos que un subárbol ***s*** de un árbol ***t*** es un ***subárbol raíz*** de ***t*** si, y solo si, ***s*** no es vacío y el nodo raíz de ***s*** es el nodo raíz de ***t***.
- ***Subárbol completo***. Diremos que un subárbol ***s*** de un árbol ***t*** es un ***subárbol completo*** *de* ***t*** si, y solo si, los descendientes en ***s*** del nodo raíz de ***s*** son exactamente los nodos descendientes de ese mismo nodo en ***t***. (Esto es lo que muchas fuentes denominan simplemente ***subárbol***)
- ***Subárbol terminal***. Diremos que un subárbol ***s*** de un árbol ***t*** es un ***subárbol terminal*** *de* ***t*** si, y solo si, ***s*** es no vacío y sus nodos hoja son también nodos hoja en ***t***.

[1] En este contexto entenderemos que estamos hablando siempre de árboles con raíz (dirigidos) y que el número de nodos es finito.

- ***Subárbol interior***. Diremos que un subárbol ***s*** de un árbol ***t*** es un ***subárbol interior*** *de* ***t*** si, y solo si, ***s*** es no vacío, su raíz no es la raíz de ***t*** y ningún nodo hoja de ***s*** es nodo hoja en ***t***.

Siguiendo estas definiciones, se pide que implemente un módulo ***BinSubTrees*** (basado en el módulo ***BinTree*** de la práctica anterior) que contenga la definición de una serie de funciones para calcular los subárboles de tipo ***'a BinTree.t***, de cada uno de estas clases, que tendría cualquier árbol dado de tipo ***'a BinTree.t***.

Las funciones incluídas en la interfaz de estos módulos deben devolver siempre una lista con todos los subárboles de cada una de las clases descritas anteriormente. Tenga en cuenta que, para un árbol dado, un determinado subárbol podría aparecer varias veces en la lista (si aparece en distintas partes del árbol original; es decir, si puede superponerse sobre este de varias maneras). Pero si los valores de todos los nodos son diferentes en un árbol, entonces, necesariamente, cada subárbol aparecerá sólo una vez en la lista. El orden en que deben aparecer los subárboles en la lista no está especificado.

Recuerde que, en los árboles binarios del módulo ***BinTree*** un árbol con una única rama se distingue de otro árbol que tenga la misma raíz y una única rama idéntica a la del anterior, si en uno de ellos es rama izquierda y en el otro es rama derecha.

El módulo ***BinSubTrees*** debe compilar correctamente (estando presentes los archivos compilados del módulo ***BinTree***: ***binTree.cmi*** y ***binTree.cmo***) con la orden:

```
ocamlc -c binSubTrees.mli binSubTrees.ml
```

Ejemplo de uso:

```
~ % ocaml
OCaml version 5.2.0
Enter #help;; for help.

# #load_rec "binSubTrees.cmo";;
# open BinTree;;
# let comb x l r =
    left_replacement (right_replacement (leaftree x) r) l;;
val comb : 'a -> 'a BinTree.t -> 'a BinTree.t -> 'a BinTree.t =
<fun>
# let complete_tree n =
      let rec aux i =
          if i > n then empty
          else comb i (aux (2*i)) (aux (2*i+1))
   in aux 1;;
val complete_tree : int -> int BinTree.t = <fun>
# let t = complete_tree 15;;
val t : int BinTree.t = <abstr>
# open BinSubTrees;;
# List.length (root_subtrees t);;
- : int = 676
# List.length (subtrees t);;
- : int = 751
# List.length (complete_subtrees t);;
- : int = 15
# List.length (terminal_subtrees t);;
- : int = 305
# List.length (inner_subtrees t);;
- : int = 12
# #quit;;
~ %
```