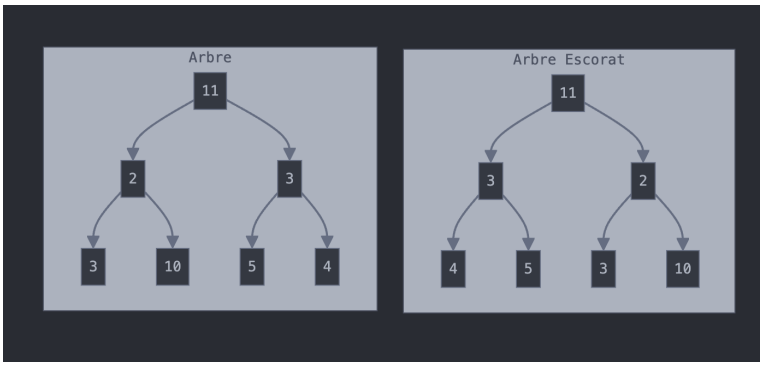
## Haskell — Arbres escorats Y60421_ca

Donat un arbre binari de naturals, el seu *pes* és la suma dels valors en els seus nodes i la seva *mida* és el nombre de nodes que té.
*Escorar* un arbre és transformar-lo de manera que, per a tots els seus nodes, el fill esquerre sempre pesi menys que el fill dret. Així, si el pes del fill esquerre és superior al pes del fill dret, l'arbre tindrà els seus dos fills intercanviats. En cas que els dos fills pesin el mateix,cal intercanviar-los si la mida del fill esquerre és més gran que la mida del fill dret.
Noteu que a l'arbre hi haurà els mateixos nodes abans i després d'escorar, però col·locats de manera diferent, i també hi ha els mateixos camins abans i després però disposats de manera diferent. Aquí teniu un arbre abans i després d'escorar:




Escriviu un programa que llegeixi una seqüència d'arbres binaris (donats per un recorregut en preordre amb -1 per a arbres buits) i els escori. Per a fer-ho, descarregueu-vos el fitxer `code.hs` que conté l'esquelet del programa.

El arbres binaris de naturals venen donats per aquest tipus:

**data** *Arbin* = *Buit* | *Node* **Int** *Arbin Arbin* **deriving Show**

La funció *escorar* té tipus *Arbin* → (*Arbin*, **Int**, **Int**). Fixeu-vos que no només ha de retornar l'arbre, sinó que, a més, ha de retornar el seu pes i la seva mida. Així la implementació resulta molt més senzilla i eficient.
Un exemple:

```
escorar (Node 2 (Node 3 Buit Buit) Buit)
```

ha de retornar

```
(Node 2 Buit (Node 3 Buit Buit), 5, 2)
```

Un altre exemple:

```
escorar (Node 11
              (Node 2 (Node 3 Buit Buit) (Node 10 Buit Buit))
              (Node 3 (Node 5 Buit Buit) (Node 4 Buit Buit)))
```

ha de retornar

```
(Node 11
     (Node 3 (Node 4 Buit Buit) (Node 5 Buit Buit))
     (Node 2 (Node 3 Buit Buit) (Node 10 Buit Buit)), 38, 7)
```

La funció *convertirEnArbin* és de tipus [**Int**] → *Arbin*. Per exemple,

```
convertirEnArbin [2, 3, -1, -1, -1]
```
ha de retornar
```
Node 2 (Node 3 Buit Buit) Buit
```
Un altre exemple:
```
convertirEnArbin [11, 2, 3, -1, -1, 10, -1, -1, 3, 5, -1, -1, 4, -1,
-1]
```
ha de retornar
```
Node 11
     (Node 2 (Node 3 Buit Buit) (Node 10 Buit Buit))
     (Node 3 (Node 5 Buit Buit) (Node 4 Buit Buit))
```

## Entrada

L'entrada és una seqüència d'arbres binaris, donats pel seu recorregut en preordre i amb la marca $-1$ per indicar els arbres buits. Hi ha un arbre per línia.

## Sortida

Per cada arbre de l'entrada, el programa ha d'escriure, en una línia, l'arbre escorat corresponent.

## Observacions

La correcció valorarà l'eficiència, l'elegància, la senzillesa i la qualitat general de la solució lliurada.

## Exemple d'entrada

```
11 2 3 -1 -1 10 -1 -1 3 5 -1 -1 4 -1 -1
5 2 10 -1 -1 8 -1 -1 20 -1 -1
5 2 -1 -1 -1
10 2 3 -1 -1 5 -1 -1 2 3 -1 -1 5 -1 -1
10 -1 -1
```

## Exemple de sortida

```
Node 11 (Node 3 (Node 4 Buit Buit) (Node 5 Buit Buit)) (Node 2 (Node 3 Buit Buit) (Node 10 Buit
Node 5 (Node 20 Buit Buit) (Node 2 (Node 8 Buit Buit) (Node 10 Buit Buit))
Node 5 Buit (Node 2 Buit Buit)
Node 10 (Node 2 (Node 3 Buit Buit) (Node 5 Buit Buit)) (Node 2 (Node 3 Buit Buit) (Node 5 Buit B
Node 10 Buit Buit
```

## Informació del problema

Autor : Jordi Delgado, Jordi Petit
Generació : 2025-04-01 07:15:25