Arbres génériques

OPTION INFORMATIQUE - TP nº 2.4 - Olivier Reynet

On se dote d'un type pour représenter les arbres génériques :

```
type 'a gtree =
| Empty
| Node of 'a * 'a gtree list;;
```

C'est ainsi que l'on peut représenter en machine l'arbre générique exposé dans le cours :

```
let gt = Node('A', [Node('B', [Node('C', []); Node('D', []); Node('E',[])]);
                    Node('F', [Node('G', []); Node('H',[])]);
                    Node('I', [Node('J', [Node('K', []); Node('L', []); Node('M',[]);
                        Node('N',[])])]);
                    Node('O', [])])
```

---

1. Écrire une fonction de signature `hauteur : 'a gtree -> int` qui renvoie la hauteur d'un arbre générique. On pourra proposer deux solutions : une mutuellement récursive et une utilisant `List.filst_left` et `max`
2. Même question pour la taille d'un arbre générique.
3. Écrire une fonction de signature `bfs : 'a gtree -> 'a list` qui renvoie le parcours par niveaux d'un arbre générique sous la forme d'une liste. On s'appuiera sur le module `Queue` d'OCaml (cf. documentation en ligne) pour implémenter une file d'attente.
4. Écrire une fonction de signature `dfs : 'a gtree -> 'a list` génère le parcours de l'arbre générique en profondeur dans l'ordre préfixe sous la forme d'une liste.

   On se dote d'un type pour représenter les arbres binaires :

   ```
   type 'a btree =
               | BEmpty
               | BNode of 'a * 'a btree * 'a btree;;
   ```

   ---

5. Écrire une fonction de signature `convert_to_btree : 'a gtree -> 'a btree` qui convertit un arbre générique en arbre binaire. On choisit la convention de coder le fils d'un nœud à gauche et ses frères à droite.