ARBRES BINAIRES

Sauf indication contraire, les arbres sont binaires et homogènes de type :

`type 'a arbre = Vide | S of ('a * 'a arbre * 'a arbre).`

La taille d'un arbre binaire est le nombre de ses sommets.

1)

a) Écrivez des fonctions qui calculent la taille d'un arbre, sa hauteur.

b) Écrivez des fonctions qui calculent les listes des feuilles ou des nœuds d'un arbre (dans un ordre quelconque).

2) Si a est un arbre, on appelle profondeur d'un sommet le nombre d'arêtes du chemin menant de l'origine à ce sommet. La hauteur de l'arbre est donc la profondeur maximale d'un sommet.

a) Écrivez une fonction qui teste si un objet appartient à un arbre : si oui, on retourne la profondeur de l'objet dans l'arbre, si non, on retourne -1.

b) Écrivez une fonction qui donne la liste des sommets de profondeur donnée d'un arbre : si le paramètre de profondeur n'est pas compatible avec l'arbre, on retourne la liste vide.

c) Un arbre binaire est dit parfait si toutes ses feuilles ont la même profondeur. Écrivez une fonction qui vérifie si un arbre est parfait.

3) Si a est un arbre et x un objet, écrivez une fonction qui donne le chemin menant de la racine vers x, c'est-à-dire la liste des étiquettes des sommets à parcourir pour aller de la racine jusque l'objet x.

4) On appelle équilibre d'un sommet x d'un arbre la différence $eq(x) = h_g(x) - h_d(x)$, où $h_g(x)$ (resp. $h_d(x)$) est la hauteur de son fils gauche (resp. droit). Un sommet est dit équilibré quand $|eq(x)| \leqslant 1$. Un arbre est dit équilibré quand tous ses sommets sont équilibrés.

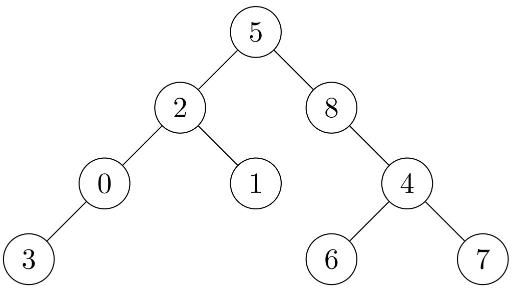
a) Écrivez une fonction qui vérifie si un arbre est équilibré ou non.

b) On appelle déséquilibre d'un arbre le maximum des $|eq(x)|$ pour tous ses sommets x. Écrive une fonction qui calcule le déséquilibre d'un arbre.

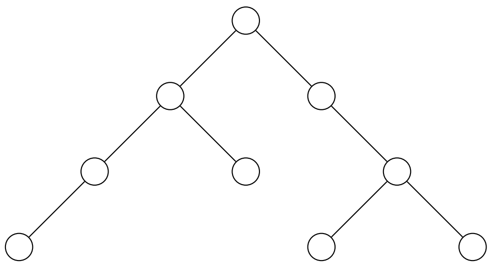
5) On appelle squelette d'un arbre binaire l'arbre binaire auquel on a retiré tous les contenus : seul subsiste la géométrie de l'arbre.

Par exemple,

le squelette de l'arbre binaire



est



On peut définir en CAML le type squelette : `type squelette = Rien | Sq of squelette * squelette.`

a) Dessinez tous les squelettes d'arbres de taille comprise entre 0 et 4.

b) On note s_n le nombre de squelettes des arbres de taille n. Montrez que $s_n = \sum_{k=0}^{n-1} s_k s_{n-1-k}$.

c) Écrivez une fonction CAML qui calcule s_n en fonction de n. Combien y-a-t-il de squelettes de taille 5 ? 6 ? 20 ?

d) Écrivez une fonction CAML qui calcule le squelette d'un arbre.

e) Écrivez une fonction CAML qui compare au sens de l'inclusion deux squelettes d'arbres (définition évidente).

6) Dans cet exercice, on suppose que les sommets des arbres sont des entiers.

Un arbre est un arbre binaire de recherche (ABR en abgrégé) si et seulement si pour tout sommet x de l'arbre, les sommets de son fils gauche sont tous inférieurs ou égaux à x et ceux de son fils droit sont tous strictement supérieurs à x.

a) Écrivez une fonction qui vérifie si un arbre binaire est un ABR.

b) Écrivez une fonction qui calcule le maximum d'un ABR. Quelle est sa complexité en fonction de h, hauteur de l'arbre ? en fonction de n, taille de l'arbre ?

c) Écrivez une fonction qui vérifie si un entier appartient à un ABR. Quelle est sa complexité en fonction de h ? de n ?

d) Écrivez une fonction qui ajoute un entier à un ABR : l'arbre construit devra être aussi un ABR.

e) Plus difficile : écrivez une fonction qui supprime un entier d'un ABR et qui retourne un ABR.

f) Écrivez une fonction qui coupe un ABR a en deux ABR b, c en fonction d'un entier x : l'ABR b contient tous les éléments de l'ABR a qui sont inférieurs ou égaux à x et c les éléments qui sont strictement supérieurs à x.

g) Écrivez une fonction qui fusionne deux ABR en un seul ABR.

7) [Linéarisation d'un arbre]

À tout arbre binaire homogène, on associe la liste de ses éléments parcourus dans l'ordre préfixe, appelée liste préfixe de l'arbre.

a) Écrivez une fonction `liste_prefixe a` qui calcule la liste préfixe des éléments d'un arbre `a`. Donnez un exemple de deux arbres distincts qui ont même liste préfixe.

Lorsqu'on crée la liste préfixe des éléments, on perd la structure de l'arbre.

On peut faire mieux : au lieu de ranger dans la liste uniquement la valeur du sommet, on va aussi enregistrer les sommets vides. On crée pour cela un type CAML : `type 'a contenu = Nil | Cont of 'a`

a) Modifiez la fonction précédente pour qu'elle enregistre l'information supplémentaire dans la liste : la liste est appelée liste préfixe complète de l'arbre.

b) Montrez que la fonction suivante de paramètre une liste préfixe complète reconstitue l'arbre associé à la liste :

```
let rec constr l =
 match l with
 | [] -> failwith "liste incorrecte"
 | Nil :: q -> Vide,q
 | C(x) :: q ->
    let g,q' = constr q in
    let d,q'' = constr q' in
    S(x,g,d), q'';;
let recons l =
  let a, q = constr l in
  if q = [] then a else failwith "liste incorrecte";;
```