TD3: arbres binaires

Exercice 1. Parcours d'arbres

a. Dessiner l'arbre binaire complet dont l'affichage des valeurs par un parcours en largeur donne la séquence suivante :

$$15, 5, 9, 12, 2, 4, 11, 3, 7, 8, 1, 6, 10, 13, 14$$

b. Donner l'ordre des valeurs que l'on obtient quand on fait un parcours préfixe, infixe et postfixe.

c. Existe-t-il un arbre à 8 nœuds dont le parcours préfixe est 1,2,3,4,5,6,7,8 et le parcours postfixe est 5,3,2,4,7,6,8,1 ? Si oui le dessiner, si non le justifier.

d. Même question mais avec le parcours postfixe 4,3,5,2,7,8,6,1.

e. Écrire un algo qui prend en entrée deux tableaux `t_pre` et `t_post` de même taille n et qui contiennent chacun toutes les valeurs de 1 à n (comme dans les 2 questions précédentes). Cet algorithme retourne `vrai` si ces deux tableaux peuvent être les parcours préfixe (pour `t_pre`) et postfixe (pour `t_post`) d'un même arbre binaire, et `faux` sinon.

Exercice 2. Opérations sur les arbres binaires

Vous disposez des 4 opérations suivantes sur les arbres binaires :

```
arbre_vide(a: Arbre):booleen, racine(a: Arbre):entier,
gauche(a: Arbre): Arbre, droite(a: Arbre):Arbre
```

Écrire les algorithmes suivants en utilisant ces opérations et donner leur complexité en fonction de la taille de l'arbre :

a. Retourner la hauteur de l'arbre. Convention utilisée en cours : hauteur de l'arbre vide est 0 et feuille est 1.

b. Retourner la valeur maximale contenue dans l'arbre (supposé non vide).

c. Retourner le nombre de sommets ayant exactement un enfant.

d. Retourner vrai si l'arbre est un arbre complet, faux sinon.

Exercice 3. Calculette

a. Représenter l'expression suivante à l'aide d'un arbre binaire :

$$\frac{(3a-1)}{2(b+c+d)} - 5(e+4)$$

b. Donner l'ordre des valeurs que l'on obtient en faisant un parcours préfixe, infixe et postfixe. Quel avantage a l'expression obtenue par le parcours postfixe ou préfixe par rapport à celle obtenue par la parcours infixe ?

c. Écrire un algorithme qui prend en entrée une expression arithmétique donnée sous forme d'arbre et qui renvoie sa valeur.

d. Le parcours postfixe donne l'écriture de l'expression en notation polonaise inversée. En utilisant uniquement une pile, retourner la valeur de l'expression qui est donnée par sa notation polonaise inversée.

Exercice 4. Codage d'Huffman

Le codage d'Huffmann a pour but de coder un texte en binaire en respectant deux propriétés :

1. Coder chaque lettre par un mot sur 0,1 (toujours le même pour une lettre).
2. Ce code est préfixe (aucun code de lettre n'est le préfixe d'un autre code).

Sous certaines hypothèses, ce codage a été démontré comme étant optimal.

Le codage d'Huffman est un algorithme qui calcule un code pour chaque caractère. Pour chaque caractère, on connait sa fréquence dans le texte à coder. L'algorithme est le suivant :

— Pour débuter, chaque lettre est un arbre ramené à un sommet étiqueté par la fréquence de cette lettre dans le texte.
— Tant qu'il y a au moins 2 arbres faire
  — considérer a_1 et a_2 les deux arbres dont les racines portent les plus petites étiquettes e_1 et e_2 ;
  — construire un nouvel arbre a dont la racine a pour sous-arbres a_1 et a_2 ;
  — la racine de a est étiquetée par $e_1 + e_2$.

Dans cet arbre, toute arête dirigée vers le sous-arbre gauche (resp. droit) est étiquetée 0 (resp. 1).

a. Construire l'arbre obtenu pour le tableau de fréquences suivant.

Lettre	A	C	E	I	N	O	R	S	T
Prop.	18	12	24	6	7	2	11	17	3

b. Coder le mot "CREATION". Expliquer comment décoder.

c. Quel est l'intérêt que le code utilisé soit préfixe ?

d. Ecrire l'algorithme `est_un_code(t, a)` qui vérifie si le tableau de bits t est un code selon l'arbre construit a.

Exercice 5. Arbre binaire de recherche

a. Enumérez tous les arbres binaires de recherche (ABR) qui contiennent les valeurs 1, 2, 3, 4.

b. Combien y a-t'il d'arbres binaires de recherche qui contiennent tous les entiers de 1 à 10 (*indication : trouver une formule récursive qui détermine le nombre d'ABR contenant tous les entiers de 1 à n*) ?

c. Construire l'ABR par ajout des valeurs successives dans l'ordre de la liste ci-dessous : 25, 60, 35, 10, 5, 20, 65, 45, 70, 40, 50, 55, 30, 15

d. Donner l'arbre obtenu par la suppression de la valeur 45.

e. On considère maintenant l'ABR dont le parcours préfixe est 8,5,2,1,4,3,7,12,10,15. Dessiner un arbre compatible avec ce parcours. Est-il unique ?

f. Ecrire un algorithme qui teste si un arbre binaire est un ABR, et donner sa complexité.

g. On suppose que les entiers compris entre 1 et 1000 sont disposés dans un arbre binaire de recherche, et on souhaite retrouver le nombre 363. Parmi les séquences suivantes, lesquelles pourraient et ne pourraient pas être la séquence de nœuds parcourus ?

1. 2, 252, 401, 398, 330, 344, 397, 363.
2. 924, 220, 911, 244, 898, 258, 362, 363.
3. 925, 202, 911, 240, 912, 245, 363.

Exercice 6. Médiane d'un arbre binaire de recherche

Dans cet exercice, on considère des arbres binaires de recherche (ABR) dont les nœuds contiennent des valeurs entières qu'on suppose toutes différentes deux à deux.

a. Rappeler comment trouver la plus petite valeur d'un ABR.

b. Écrire un algorithme qui retourne la deuxième plus petite valeur contenue dans un ABR qu'on suppose contenir au moins deux nœuds.

c. Donner un algorithme qui retourne la médiane d'un ARB, et calculer sa complexité.

Exercice 7. Arbres équilibrés

On appelle ici *arbre équilibré* un arbre binaire tel que, en tout nœud, la hauteur des sous-arbres gauches et droits diffère d'au plus 1.

a. Donner plusieurs exemples possibles d'arbres équilibrés de 10 nœuds au total.

b. Quelle est la taille minimale et la taille maximale d'un arbre équilibré de hauteur h ?

c. En déduire les hauteurs minimale et maximale d'un arbre équilibré de 20 nœuds au total.

d. Ecrire un algorithme qui vérifie qu'un arbre binaire quelconque est bien un arbre équilibré. Donner sa complexité.

Exercice 8. Arbres AVL

a. Dessiner tous les arbres AVL contenant les entiers de 1 à 5.

b. Ecrire un algorithme qui teste si un arbres est un AVL. Donner sa complexité.

c. Construire l'arbre AVL par ajout des valeurs successives dans l'ordre de la liste ci-dessous (l'arbre doit être un AVL à chaque insertion d'une valeur, et vous ferez au plus une opération de rotation pour maintenir l'équilibre) : 25, 60, 35, 10, 5, 20, 65, 45, 70, 40, 50, 55, 30, 15

d. Écrire l'algorithme d'insertion dans un AVL.

e. Écrire l'algorithme de suppression dans un AVL.