Feuille d'exercices no. 7

Révisions
L'ordre des exercices est au choix, selon chacun d'entre vous

Exercice 1.

1. Construire l'arbre AVL obtenu par insertions successives des entiers suivants : 13, 26, 31, 18, 12, 20, 11, 10.
2. Ensuite, construire l'AVL obtenu après les suppressions successives des entiers 31, 13, 11, 10.

Exercice 2. Construire l'arbre rouge et noir obtenu par insertions successives des entiers suivants : 36, 40, 24, 10, 50, 22, 26, 9, 12, 13, 25. Donner tous les détails de la construction pas à pas.

Rappels : Lors de l'insertion d'un nouveau sommet dans un arbre rouge et noir, deux cas problématiques peuvent apparaître, impliquant le sommet inséré, son père, son grand-père et son oncle. Dans l'un des cas, une rotation et un recoloriage résolvent le problème. Dans l'autre, un recoloriage résout le problème ou le remonte vers la racine.

Exercice 3. Dans cet exercice, on considère qu'une expression arithmétique est donnée comme une suite de caractères, dont les *lettres* sont des variables et les *opérations* sont les quatre opérations arithmétiques de base : $+, -, *, /$. De plus, on suppose que l'expression est correcte arithmétiquement et qu'elle ne comporte pas de parenthèses ; les opérations s'effectuent donc selon leurs priorités habituelles. On dit qu'un arbre binaire T *représente* une expression E si :

- chaque feuille de T contient une lettre de E (si une lettre apparait plusieurs fois dans E alors elle apparait autant de fois dans T).
- chaque noeud interne de T (y compris la racine) contient une opération de E (si une opération apparait plusieurs fois dans E alors elle apparait autant de fois dans T).
- le parcours symétrique de T affiche l'expression E.

Aussi, on dit qu'un arbre binaire T *calcule* une expression E si T représente E et si, de plus, T permet d'évaluer correctement E, quelles que soient les valeurs des variables, en calculant pour tout noeud interne (y compris la racine) :

- la valeur de son sous-arbre gauche,
- la valeur de son sous-arbre droit, et
- le résultat de l'opération indiquée dans le noeud, utilisant les deux valeurs des sous-arbres gauche et droit.

Dans ce cas, l'évaluation correcte de E est le résultat obtenu à la racine.

On demande :

1. Soit l'expression
$$E = a/b/c * d - f - a - g + a * g - a/b * f * g/c.$$
Dessiner un arbre T_1 qui représente E mais qui ne calcule pas E, ainsi qu'un arbre T_2 qui calcule E.
2. Ecrire en pseudo-code un algorithme $Representer(E)$, qui prend en entrée une expression E et qui construit un arbre T_1 qui représente E (sans se soucier de savoir si T_1 calcule E ou non).
3. Exécuter l'algorithme $Representer(E)$ sur l'expression E du point 1.
4. Quelle est la complexité de l'algorithme $Representer(E)$? Justifier.
5. Ecrire en pseudo-code l'algorithme $Calculer(E)$, qui prend en entrée une expression E et qui construit un arbre T_2 qui calcule E.
6. Exécuter l'algorithme $Calculer(E)$ sur l'expression E du point 1.
7. Quelle est la complexité de l'algorithme $Calculer(E)$? Justifier.
8. Supposons maintenant qu'un arbre T qui calcule E a été construit, et que l'expression E a été perdue. Supposons également que nous disposons d'un tableau Val dont les indices sont les lettres de E et tel que $Val(x)$ représente, pour toute lettre x de E, la valeur de la variable x. Ecrire un algorithme $Evaluer(T, Val)$ qui évalue l'expression E à l'aide seulement de T lorsque ses variables ont les valeurs indiquées par le tableau Val.
9. Quelle est la complexité de l'algorithme $Evaluer(T, Val)$? Justifier.