Les boucles sont interdites.

1 Définition

Un arbre A est un arbre binaire de recherche (ABR) ssi

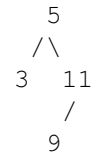
- A est vide

ou
- A.filsG est un ABR, et
- A.filsD est un ABR, et
- toute valeur x dans A.filsG vérifie $x \leq A.val$, et
- toute valeur x dans A.filsD vérifie $x > A.val$

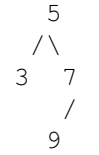
Les ABR sont utilisés comme structure de donnée permettant d'insérer, de supprimer, et de rechercher rapidement des valeurs. En effet, si l'on à n éléments, en utilisant une liste, la recherche peut prendre (dans le pire des cas) de l'ordre de n opérations. D'un autre côté, comme vous le verrez dans une des questions ci-dessous, la recherche dans un ABR coûtera de l'ordre de h opérations (où h est la hauteur de l'arbre), puisque l'on pourra chercher en commençant à la racine, et à chaque étape descendre "toujours du bon côté". L'idée est que, si l'arbre est bien équilibré, on aura $h \approx log(n)$, et on gagnera ainsi beaucoup de temps par rapport aux listes.

Exercice 1. Echauffement

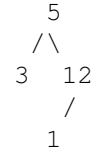
Question 1.1.
Est ce que l'arbre ci-dessous est un ABR ?



Question 1.2.
Est ce que l'arbre ci-dessous est un ABR ?



Question 1.3.
Est ce que l'arbre ci-dessous est un ABR ?



Exercice 2. Recherche

Question 2.1.
Ecrire une fonction `boolean recherche(Arbre a, int x)` suivante. Prérequis : l'arbre a est un ABR. Action : retourne vrai ssi x est dans l'arbre.

Exercice 3. Affichage trié

Question 3.1.
Ecrire une fonction `String toStringTrie(Arbre a)` suivante. Prérequis : l'arbre a est un ABR. Retourne dans une chaîne toutes les valeurs de l'arbre triées par ordre croissant.

Exercice 4. Insertion

Question 4.1.
Ecrire une fonction `Arbre insert(Arbre a, int x)` suivante. Prérequis : l'arbre a est un ABR. Action : retourne un ABR obtenu en inserant x dans a.

Exercice 5. Suppression

Question 5.1.
Ecrire une fonction `Arbre suppr(Arbre a,int x)` suivante. Prérequis : l'arbre a est un ABR. Action : retourne un ABR obtenu en supprimant x de a (si x n'est pas présent on doit retourner a). Indication: le problème délicat sur lequel vous devez tomber est celui où x est contenu dans la racine de a, et où les deux sous arbres de a sont non vides. Dans ce cas, pensez à la stratégie suivante (et faites un dessin pour vous convaincre que l'arbre obtenu est bien un ABR):

- cherchez m, le maximum du sous arbre gauche
- enlever m du sous arbre gauche
- dans la racine, remplacez x par m