Université Paris Diderot – Paris 7
Algorithmique

L3 Informatique
29 octobre 2012

Examen partiel

Documents autorisés : une feuille A4 recto verso manuscripte. Durée 2h

Exercice 1 [Arbres AVL] Question 1. Insérez successivement les valeurs

$$4, 6, 5, 3, 7, 2, 8, 1, 9$$

dans un arbre AVL. Si une rotation a lieu, vous devez dessiner l'arbre avant, l'arbre après, et nommer la rotation, par exemple « rotation gauche entre x et y ».

Toutefois, si une insertion ne fait pas de rotations, vous pouvez juste rajouter le nouveau nœud au dernier arbre dessiné, mais dans une couleur différente (ou en l'entourant deux fois).

Ainsi, la première rotation ayant lieu après l'insertion de 5, le premier arbre dessiné sur votre copie sera celui contenant $4, 6, 5$ avant rotation, et le deuxième aura les 3 même nœuds mais après rotation(s). Vous pouvez rajouter à ce deuxième arbre 3 puis tous les éventuels nœuds jusqu'à obtenir une rotation. Et ainsi de suite.

Question 2. Supprimez maintenant le nœud qui est à la racine de votre arbre. Indiquez tous les arbres intermédiaires à chaque opération modifiant l'arbre et nommez ces opérations (rotations, etc).

Exercice 2 [Fusion de tas] Soit k un entier. Décrire comment réaliser la fusion de deux tas ayant $2^k - 1$ nœuds chacun. Il s'agit de tas-max : la racine est le plus grand élément. Après fusion, on obtient un seul tas contenant les $2^{k+1} - 2$ nœuds.

Vous pouvez choisir d'implémenter les tas
- soit comme des arbres binaires (un noeud a deux pointeurs et une valeur) où le paramètre est la racine
- soit comme des tableaux

Analyser la compexité de votre algorithme. Plus il sera rapide, meilleure sera la note ! Attention l'une des deux implémentations permet un algorithme plus rapide que l'autre.

Exercice 3 [généralisation des ABR : les B-arbres] On se donne un certain k (fixé et constant). Un B-arbre, ou arbre k-aire de recherche, est un arbre où chaque nœud possède
- Au plus k fils $N.Fils[1]...N.Fils[k]$
- Un tableau Val de k-1 valeurs $N.Val[1]...N.Val[k-1]$

Il sert à stocker des valeurs dans les nœuds. Comme un arbre binaire de recherche (ABR) sauf qu'un nœud stocke entre 0 et $k-1$ valeurs, et non 0 ou 1 comme dans un ABR. Les ABR sont en fait exactement les B-arbres pour $k = 2$. Un B-arbre vérifie les propriétés suivantes :

- **propriété 1 :** un nœud N n'a des fils que s'il contient $k-1$ valeurs (aucun $+\infty$ dans $N.Val[\,]$).
- **propriété 2 :** Pour tout nœud N $N.Val$ est trié : pour tout $1 \leq i < k-1$

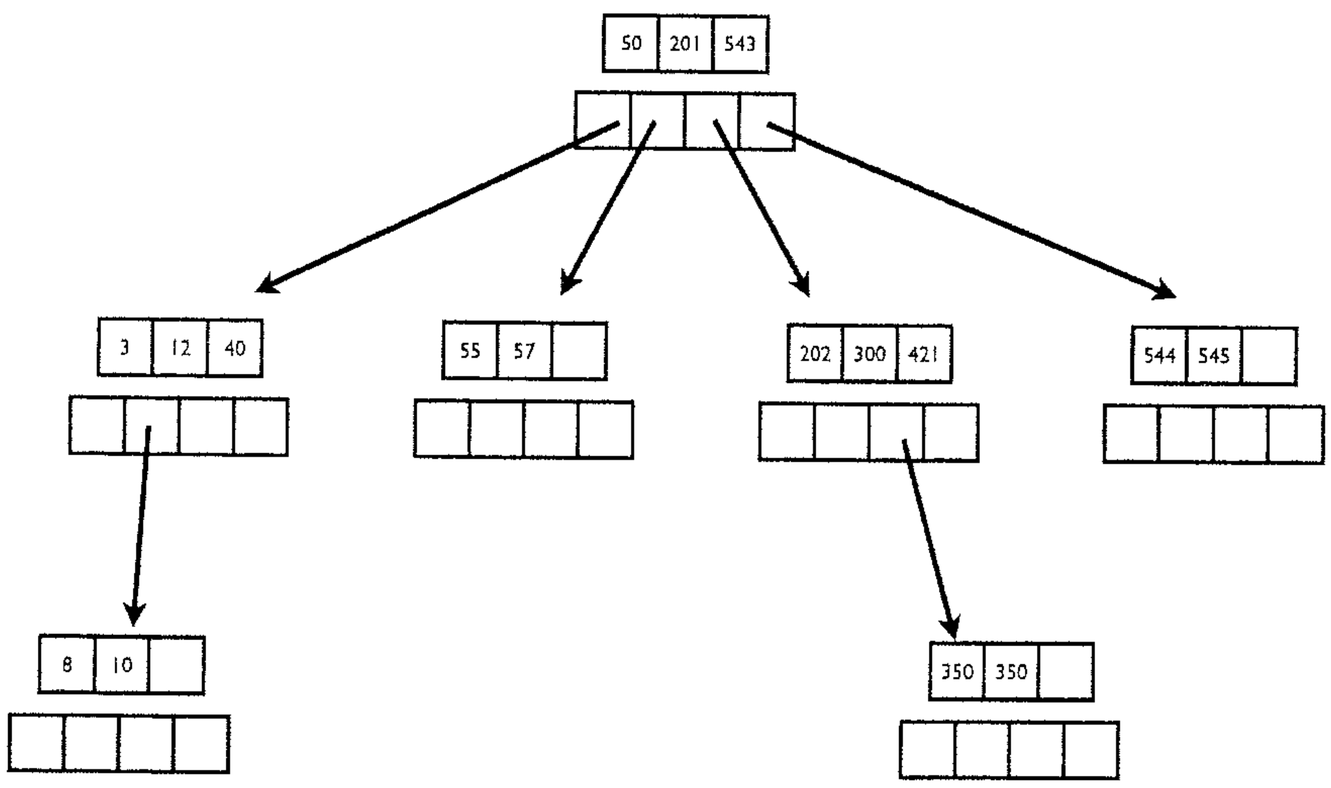
$$N.Val[i] \leq N.Val[i+1]$$

- **propriété 3 :** Pour toute valeur x stockée dans un nœud M, si M descend d'un nœud N par son $i$ème fils (c'est-à-dire si $M = N.Fils[i]$ ou M descend de $N.Fils[i]$) :

$$N.Val[i-1] \leq x \leq N.Val[i]$$

NB : si $i = 1$ on a seulement $x \leq N.Val[1]$ et si $i = k$ on a seulement $N.Val[k-1] \leq x$

Question 1. Insérer les valeurs 9, 600, 700, 202 dans le B-arbre suivant (un seul dessin de B-arbre est à rendre. On a $k = 4$.)



On stocke la valeur $+\infty$ dans une case de $N.Val$ pour signifier que la case est vide (aucune valeur n'y est stockée) et on pose $N.Fils[i]$ =`nil` si le $i$ème fils n'existe pas.

Question 2. Écrire la fonction de recherche d'une valeur dans un B-arbre. Donner sa complexité.

Question 3. Écrire la fonction d'insertion d'une valeur dans un B-arbre. Donner sa complexité. NB : pour maintenir la propriété 1, il faut d'abord remplir le tableau d'un nœud avant de lui donner des fils.