CPI1 2009-2010

Algorithmique fonctionnelle

Nga Nguyen – Stefan Bornhofen – Peio Loubière

TD8: Arbres binaires de recherche

1 Insertion et suppression

1. Insérer successivement dans un arbre binaire de recherche (ABR) vide: 4,7,2,6,12,3,5,1.

2. Y supprimer successivement: 12, 4, 5, 7, 3.

2 Réflexions

1. Quel est le critère d'une série d'insertions dans un arbre vide pour que l'ABR résultant soit filiforme?

2. Supposons une branche d'ABR qui va de la racine jusqu'à une feuille donnée. On note respectivement A, B et C l'ensemble des valeurs situées à gauche, le long et à droite de cette branche. Si $a \in A, b \in B, c \in C$, est-ce que nous avons nécessairement a<b<c?

3. Montrer (par récurrence) que la connaissance du parcours préfixe d'un ABR suffit pour reconstituer l'arbre initial. Est-ce vrai pour le parcours postfixe?

4. Montrer que la connaissance de la donnée du parcours infixe ne suffit pas pour reconstituer l'arbre initial.

5. Les séquences suivantes peuvent-elles correspondre à une suite d'éléments dans un ABR visités lors de la recherche de la valeur 2048 ?

   (a) 100, 140, 4096, 150, 3000, 2048
   (b) 100, 150, 4096, 140, 3000 ,2048
   (c) 4096, 100, 140, 150, 3000 ,2048
   (d) 100, 150, 140, 4096, 3000 ,2048
   (e) 3000, 4096, 140, 150, 100 ,2048

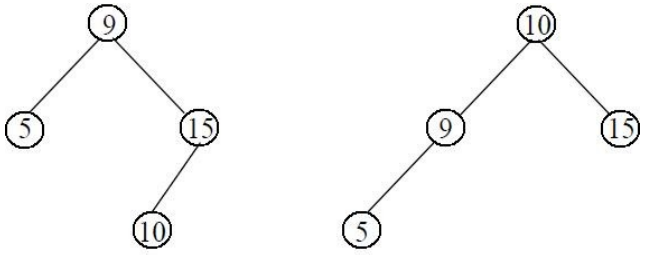
3 Définir en pseudo-code les fonctions

1. **estABR** qui teste si un arbre binaire donné est un arbre binaire de recherche. *Attention*: l'information comme quoi le fils gauche est inférieur au noeud et le fils droit est supérieur au noeud est insuffisante (cherchez un contre-exemple)!

2. **rechercher** qui recherche un élément donné dans un ABR et renvoie l'ABR dont la racine contient cet élément;

3. **rechercheValide** qui reçoit une liste d'éléments et détermine si cette liste est une recherche valide dans un ABR (voir l'exercice 2.5);

4. **équivalent** qui teste l'équivalence de deux ABR. Deux ABR sont dits équivalents lorsqu'ils comportent les mêmes valeurs, par exemple:



4 Insertion à la racine

Dans cette méthode d'insertion, on ajoute un nouvel élément non pas en tant que feuille, mais à la racine de l'arbre. Ceci permet de retrouver plus rapidement les éléments récemment insérés. Le principe consiste à construire un nouvel arbre dont

- la racine est x
- le sous-arbre gauche est une copie de l'arbre A où l'on a taillé tous les éléments > x
- le sous-arbre droit est une copie de l'arbre A où l'on a taillé tous les éléments < x

Ecrire les fonctions

- **taillerSup (A,x)** qui renvoie une copie de l'arbre A où l'on a taillé tous les éléments > x
- **taillerInf (A,x)** qui renvoie une copie de l'arbre A où l'on a taillé tous les éléments < x
- **insérerRacine(A,x)** qui compose les deux arbres précédents et l'élément x à insérer.

Exemple: Insertion de « f » à la racine de l'arbre A.