Lille iut Université de Lille

# TD 6 : Arbres binaires de recherche (ABR)

Structures de données
DUT info 2ème année
2019–2020

## Exercice 1

Énumérez tous les arbres binaires de recherche contenant les clés 1, 2, 3, 4.

## Exercice 2

Le professeur Košmrlj pense avoir découvert une propriété remarquable des arbres binaires de recherche. Supposez que la recherche d'une clef $k$ dans un arbre binaire de recherche se termine sur une feuille. On considère trois ensembles :

— $A$, les clés situées à gauche du chemin de recherche ;
— $B$, les clés situées sur le chemin de recherche ;
— $C$, les clés situées à droite du chemin de recherche.

Le professeur Košmrlj affirme que, étant données trois clés, $a \in A$, $b \in B$ et $c \in C$, quelconques, elles doivent satisfaire $a \leq b \leq c$. Donnez un contre exemple qui soit le plus petit possible.

## Exercice 3

a) Écrivez une classe `BST` (pour *Binary Search Tree*) avec au minimum une clé de type $T$, ses fils gauche et droit et ses constructeurs `BST<>()` et `BST<>(T key)` associés.

b) Écrivez une méthode récursive `BST<T> bstMin()` qui retourne le sous-arbre de clé minimum de l'arbre, puis aussi la méthode `T min()`

c) Quel est l'emplacement du successeur d'un nœud dans un ABR ? Décrivez tous les cas possibles.

d) Montrez que, si un nœud d'un ABR (dont tous les nœuds sont distincts) a deux enfants, alors son successeur n'a pas de fils gauche et son prédécesseur n'a pas de fils droit.

## Exercice 4

a) A partir d'un arbre binaire vide, ajoutez successivement les valeurs suivantes : 22, 09, 41, 39, 27, 06, 11, 45, 59, 18, 36.

b) Écrivez une méthode `boolean add(T key)`, version récursive de l'algorithme d'ajout d'un nœud de clé `key` dans un ABR.

## Exercice 5

a) A partir de l'arbre final de l'exercice précédent, supprimez successivement les valeurs 59 et 11.

b) Supprimez maintenant les valeurs 22 et 41. Quelles valeurs choisissez-vous pour les remplacer ?

c) Écrivez une méthode `T deleteMin()` qui supprime le nœud de clé minimum de l'ABR.

d) Écrivez une méthode `delete(T key)` version récursive de l'algorithme de suppression d'un nœud dans un ABR de racine $x$. Conseil : utilisez les méthodes `min` et `deleteMin` précédemment écrites.

## Exercice 6

**Question 1 :** Expliquez où se situe le prédécesseur d'un noeud dans un arbre binaire de recherche. Détaillez votre réponse en fonction des différentes configurations possibles.

**Question 2 :** À partir de l'arbre binaire de recherche vide, ajoutez successivement les valeurs suivantes : 22, 7, 71, 39, 27, 2, 11, 42, 61, 18, 17. **Détaillez toutes les étapes !**

**Question 3 :** Puis supprimez la valeur 22 en détaillant aussi votre démarche.

**Question 4 :** Votre classe d'arbre binaire de recherche est déclarée ainsi :

```
public class BST<E> {
    E key;
    BST<E> left;
    BST<E> right;
}
```

Redéfinissez la fonction `equals` adaptée à votre classe d'arbre de telle façon qu'elle retourne `true` si et seulement si les deux arbres ont même structure et contiennent les mêmes valeurs.