Travaux Pratiques n°3

Cours d'Informatique de Deuxième Année

—Licence MI L2.2—

Arbres Binaires de Recherche (ABR)

Cette feuille d'exercice a pour but de vous faire manipuler la structure de donnée des ABRs. La définition d'un ABR est un arbre binaire tel que la valeur stockée dans chaque sommet est supérieure à celles stockées dans son sous-arbre gauche et inferieure à celles de son sous-arbre droit. Durant tous le td nous nous contenterons de stoquer des entiers dans chaque noeud.

▸ **Exercice 1.** ***Structure de l'arbre binaire***

Déterminer la structure de l'ABR.

Écrire les fonctions :

- `Noeud * creerNoeud(int valeur)` *qui alloue un noeud de l'arbre et initialise ses champs ;*
- `int libereNoeud(Noeud * n)` *qui libére l'espace réservé pour* `n`*.*

▸ **Exercice 2.** ***Insertion, recherche dans un ABR***

En respectant la définition d'un ABR, écrire les fonctions suivantes :

- `int insertionABR(ABR * abr, int valeur)` *qui insère la* `valeur` *dans l'ABR* `abr` *;*
- `int recherche(ABR abr, int valeur)` *qui retourne 1 si la valeur est trouvée ;*

Puis déterminer qu'elle est la compléxité de ces deux algorithmes.

▸ **Exercice 3.** ***Parcours d'ABR***

Réaliser la fonction qui affiche les valeurs contenues dans l'arbre dans l'ordre croissant.

▸ **Exercice 4.** ***Suppression dans un ABR***

Réaliser l'algorithme de suppression dans un ABR. Attention l'arbre après suppression doit toujours respecter les propriétés des ABRs.

▸ **Exercice 5.** ***Algorithme de tris***

Utiliser l'ABR pour réaliser un algorithme de tris. En réfléchisant sur le compléxité de l'algorithme d'insertion et de suppression, déterminer la compléxitée de cette algorithme.