TD8 - Arbre binaire de recherche (ABR)

Exercice 1 - Rappels sur les ABR

Étudier les algorithmes *successeur*, *predecesseur*, *recherche*, *insertion* et *suppression*.

Exercice 2 - Final Printemps 2003

Soit A un ABR et a une valeur. On veut déterminer la plus petite valeur b contenue dans A qui est supérieure ou égale à a.

Par exemple, pour A contenant les valeurs 1, 3, 5, 7, 9 : Si $a = 4$ alors $b = 5$. Si $a = 9$ alors $a = b = 9$. Si $a = 10$ alors il n'existe pas de valeur b. Un noeud de l'arbre A correspond à une structure de type *struct noeud* comportant trois champs :

— une valeur entière

— un pointeur vers le sous-arbre gauche

— un pointeur vers le sous-arbre droit

Le type *arbre* correspond à un pointeur sur la structure *struct noeud*.

1. Définir la structure *struct noeud* et le type *arbre* en C.
2. Écrire une fonction *int max(arbre A)* en C qui retourne la clé de valeur maximale contenue dans un ABR non vide, ainsi qu'une fonction *int min(arbre A)* qui retourne la clé de valeur minimale.
3. Écrire un algorithme récursif *search_geq(A : ABR, a : entier)* qui prend en entrée un ABR A et une valeur a et qui retourne le noeud de l'arbre A contenant la plus petite valeur supérieure ou égale à a (ou NIL si un tel noeud n'existe pas).
4. Écrire une fonction *search_geq(arbre A, int a)* en C qui correspond à l'algorithme de la question précédente.

Exercice 3 - Intervalles et modélisation avec un ABR

On s'intéresse à la gestion informatique des réservations sur l'année d'une salle de conférence. La salle ne peut être réservée que par journées. Une réservation peut être assimilée à un intervalle de jours $I = [min(I), max(I)]$, où $min(I)$ est le jour d'arrivée des participants et $max(I)$ le dernier jour de la conférence. Pour un intervalle I, on a trois opérations de base : *ajouter(I)*, *supprimer(I)* et *intersecte(I)* (qui renvoie VRAI lorsque la salle ne peut pas être réservée pendant l'intervalle I).

1. Définir les propriétés d'un ABR pour représenter les différentes réservations.
2. Ajouter graphiquement les réservations successives [37, 39], [3, 7], [100, 120], [50, 68], [10, 30], [1, 2], [8, 9].
3. Supprimer graphiquement la réservation [3, 7].
4. Écrire l'algorithme *intersecte* et le prouver. Analyser sa complexité.
5. Déterminer la complexité des trois opérations de base lorsque les réservations sont gérées avec un ABR, une liste chaînée triée et une liste chaînée quelconque. Comparer.