TD6 : Arbres binaires de recherche et équilibrage

Exercice 1 – Arbres binaires de recherche

Dans cet exercice, on considère des arbres binaires de recherche (ABR) contenant des entiers, et donc on propose de travailler avec la structure suivante :

```
1  typedef struct noeud {
2     int valeur;
3     struct noeud * fg;
4     struct noeud * fd;
5  } Noeud;
6
7  typedef Noeud* ABR;
```

Q 1.1 Écrire une fonction `Noeud* rechercherValeur(ABR ab, int val);` qui recherche si un élément `val` est dans l'arbre, puis retourne un pointeur sur l'élément, ou NULL sinon.

Q 1.2 Écrire une fonction `void insererElem(ABR* ab, int val);` qui insère un élément de valeur `val` dans un ABR.

Q 1.3 Écrire une fonction `void supprimer(ABR* ab, int val);` qui supprime un élément dans un ABR.

Exercice 2 – Arbres binaires de recherche équilibrés

Q 2.1 Donner l'arbre binaire de recherche obtenu en ajoutant itérativement les entiers de 1 à 6 dans un arbre binaire de recherche vide (sans équilibrage). Rappeler pourquoi il est important d'avoir des arbres équilibrés.

Q 2.2 Pour équilibrer un arbre binaire efficacement, il faut parvenir à connaître rapidement la hauteur de chaque noeud. Expliquer comment modifier la structure précédente pour pouvoir faire cela.

Q 2.3 Écrire une fonction `AB_hauteur` qui étant donné un arbre binaire, retourne sa hauteur (on considère que la hauteur est donnée en nombre d'arrêtes, et que la hauteur d'un arbre vide est -1). Écrire une fonction `void majHauteur(ABR ab);` qui met à jour la hauteur d'un ABR si au moins l'un de ses deux fils a été modifié.

Q 2.4 Écrire une fonction `void rotationDroite(ABR* ab);` qui permet de réaliser une rotation droite d'un arbre binaire (la figure 1 présente une telle rotation).

Q 2.5 Écrire une fonction `AB_rotationGauche` qui permet de réaliser une rotation gauche d'un arbre binaire.

Q 2.6 Écrire une fonction `void insererElem_avec_eq(ABR * ab, int v);` qui étant donnés un ABR équilibré et une valeur, retourne l'arbre équilibré obtenu en ajoutant la valeur à l'arbre. Notons A l'arbre obtenu après insertion sans équilibrage, et G et D ses sous-arbres gauche et droite respectivement. On rééquilibre A de la manière suivante :

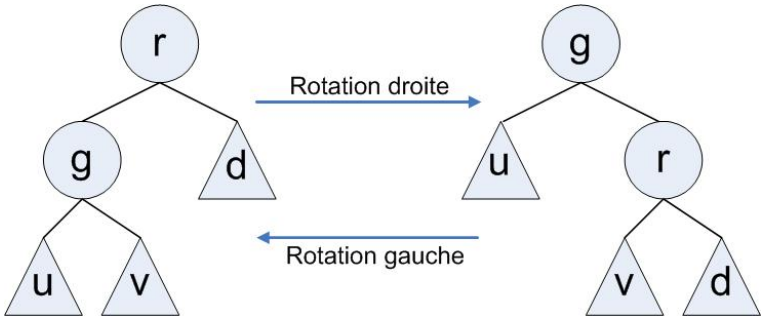

FIGURE 1 – Rotation droite et rotation gauche dans un arbre binaire

— Si $|h(G) - h(D)| < 2$, aucune réorganisation n'est nécessaire.
— Si $h(G) - h(D) = 2$, alors notons g et d les sous-arbres gauche et droite de G respectivement. Si $h(g) < h(d)$, alors on effectue une rotation gauche de G. Dans tous les cas, on effectue ensuite une rotation droite de A.
— Si $h(G) - h(D) = -2$ alors notons g et d les sous-arbres gauche et droite de D respectivement. Si $h(d) < h(g)$, alors on effectue une rotation droite de D. Dans tous les cas, on effectue ensuite une rotation gauche de A.

Q 2.7 Donner l'arbre binaire de recherche obtenu quand on ajoute les nombres entiers de 1 à 6 dans un arbre binaire vide, avec équilibrage.