TD10 - Arbre binaire de recherche et arbre équilibré

Exercice 1 - Équilibre d'une arbre binaire de recherche

Soit A un ABR et x un sommet de A. On appelle équilibre de x la quantité $eq(x) = h(A_g[x]) - h(A_d[x])$ (différence entre la hauteur du sous-arbre gauche et la hauteur du sous-arbre droit). On note :

— y le fils gauche (respectivement droit) de x lors d'une rotation droite (respectivement gauche) ;

— eq_x et eq_y les équilibres de x et y avant rotation ;

— eq_x^r et eq_y^r les équilibres de x et y après rotation.

Après rotation sur x, les relations suivantes sont satisfaites :

Rotation droite

$$eq_x^r = eq_x - 1 - max(0, eq_y)$$

$$eq_y^r = \begin{cases} eq_y - 1 \text{ si } eq_x^r \geqslant 0 \\ eq_x - 2 + min(0, eq_y) \text{ sinon} \end{cases}$$

$$\Delta = \begin{cases} -1 & \iff eq_y^r \geqslant 0 \\ 1 & \iff eq_x \leqslant 0 \\ 0 & \text{sinon} \end{cases}$$

Rotation gauche

$$eq_x^r = eq_x + 1 - min(0, eq_y)$$

$$eq_y^r = \begin{cases} eq_y + 1 \text{ si } eq_x^r \leqslant 0 \\ eq_x + 2 + max(0, eq_y) \text{ sinon} \end{cases}$$

$$\Delta = \begin{cases} 1 & \iff eq_x \geqslant 0 \\ -1 & \iff eq_y^r \leqslant 0 \\ 0 & \text{sinon} \end{cases}$$

1. Démontrer la formule de eq_x^r pour la rotation droite.
2. Démontrer la formule de Δ pour la rotation droite.
3. En déduire une fonction *Rotd(x : Noeud ; Δ : entier)* qui réalise une rotation droite autour de x, met à jour l'équilibre des sommets, renseigne Δ et renvoie l'arbre modifié.
4. Soit v un noeud de A. Exprimer en fonction de eq_v la variation de hauteur de $A(v)$ selon que $A_d[v]$ ou $A_g[v]$ augmente ou diminue d'une unité.
5. Écrire une fonction *Rotgd(x : Noeud ; Δ : entier)* qui réalise une rotation gauche-droite. On suppose qu'on dispose des fonctions *Rotd(x : Noeud ; Δ : entier)* et *Rotg(x : Noeud ; Δ : entier)*.

Soit un AVL A de racine r. Après insertion d'un nouveau noeud n, on note x le premier noeud parent de n tel que $|eq_x| = 2$. Tous les noeuds z descendants de x sont tels que $|eq_z| \leqslant 1$. On note y la racine du sous-arbre de x où n a été inséré.

6. Montrer que $eq_y \neq 0$
7. Montrer que les rotations suivantes sur x permettent de rééquilibrer le sous-arbre de racine x :
   (a) Rotation droite si $eq_x = 2$ et $eq_y = 1$;
   (b) Rotation gauche si $eq_x = -2$ et $eq_y = -1$;
   (c) Rotation gauche-droite si $eq_x = 2$ et $eq_y = -1$;
   (d) Rotation droite-gauche si $eq_x = -2$ et $eq_y = 1$.

   Rapporter à chaque fois la valeur de Δ, variation de hauteur du sous-arbre (initialement de racine x) suite à la rotation réalisée.
8. Écrire une fonction *reequilibre_avl(x : Noeud ; Δ : entier)* qui équilibre l'arbre de racine x suite à l'insertion d'un élément n dans un AVL. x est le premier noeud parent de n tel que $|eq_x| = 2$. L'algorithme renvoie le noeud racine de l'arbre équilibré et renseigne Δ, variation de hauteur de l'arbre.
9. Une fois cette opération réalisée, est-il nécessaire d'équilibrer d'autres sommets de l'arbre ?
10. Écrire un algorithme *insere_avl(x : Noeud ; n : Noeud ; Δ : entier)* qui insère le noeud n dans l'AVL de racine x et équilibre l'arbre. L'algorithme renvoie l'AVL après insertion et sa variation de hauteur Δ.
11. Analyser la complexité de cet algorithme.