Programmation et Structures de Données 1995-96
Projet 2: Arbres binaires de recherche

Frédéric Vivien, bureau 306, e-mail: Frederic.Vivien

24 octobre 1995

1 Documentation sur le graphisme en *Caml Light*

Une documentation complète est disponible à l'URL suivant (à ouvrir avec `mosaic` ou `netscape`):

`http://www/~fvivien/Enseignement/caml-graph.html`

2 Affichage graphique d'un arbre

Écrire une procédure d'affichage graphique d'un arbre binaire.

3 Parcours d'arbres

Écrire deux fonctions de parcours d'arbres binaires: l'une en profondeur préfixe (parcours du sous-arbre gauche, puis du du sous-arbre droit); l'autre en largeur préfixe (le fils gauche, puis le fils droit, puis les petit-fils, etc.). Faire varier la couleur du nœud visité pour rendre lisible le parcours sur l'affichage graphique.

4 Recherche d'une clé

Écrire une fonction de recherche d'une clé dans un arbre binaire de recherche.
Affichage en rouge du nœud contenant la clé, et en vert des nœuds visités.
Compter le nombre de nœuds visités.

5 Définition des AVL

Les AVL sont des arbres binaires de recherches particuliers.
Soit $t = (\text{clé}, t_1, t_2)$ un arbre binaire.
Définissons δ la fonction qui à tout arbre associe sa fonction de déséquilibre à la racine:

$$\delta(\text{clé}, t_1, t_2) = \text{hauteur}(t_1) - \text{hauteur}(t_2)$$

Un arbre binaire de recherche est un arbre **AVL** si pour chacun de ses sous-arbres la fonction δ vaut au plus un en valeur absolue.
La figure 1 montre un arbre AVL.
Dans notre cas, chaque nœud d'un AVL contiendra les champs suivants:

- clé
- couleur

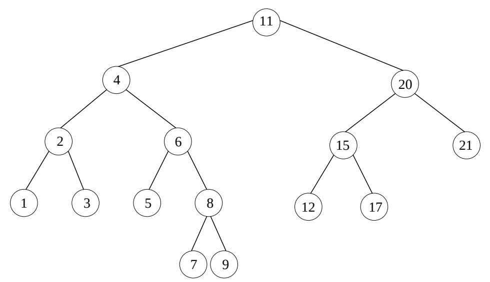

FIG. 1 - *Exemple d'arbre AVL*

- balance
- fils_gauche
- fils_droit

6 Insertion d'une clé dans un AVL

Algorithme: Soit $t = (clé, t_1, t_2)$ un arbre AVL.
Soit *nouvelle*, la clé à insérer. On suppose que *nouvelle* n'est pas déjà présente dans t.
Si *nouvelle* $<$ *clé*, l'ajout s'effectuera dans le sous-arbre t_1 ce qui fournira un arbre ${t'}_1$. Plusieurs cas de figures se présentent:

- Si $\text{hauteur}({t'}_1) = \text{hauteur}(t_1)$, le résultat de l'ajout sera $(clé, {t'}_1, t_2)$
- Si $\text{hauteur}({t'}_1) = \text{hauteur}(t_1) + 1$, et si $\delta(t) = 0$ ou $\delta(t) = -1$, le résultat de l'ajout sera encore $(clé, {t'}_1, t_2)$.Il faudra toutefois positionner correctement l'information d'équilibre à la racine de l'arbre résultat.
- Si $\text{hauteur}({t'}_1) = \text{hauteur}(t_1) + 1$, et si $\delta(t) = 1$, l'arbre $t' = (clé, {t'}_1, t_2)$ n'est pas un AVL car $\delta(t') = 2$. Il convient de le rééquilibrer en utilisant les rotations. Ici, il faut considérer l'équilibre de l'arbre ${t'}_1$. Cet arbre ne peut pas être équilibré. S'il l'était, il aurait la même hauteur que t_1. Il n'y a donc que deux cas possibles:
  - $\delta({t'}_1) = 1$. t' peut être rééquilibré par une rotation droite. Le résultat de l'ajout sera donc $rd((clé, {t'}_1, t_2))$.
  - $\delta({t'}_1) = -1$. t' peut être rééquilibré par une rotation gauche sur ${t'}_1$ suivie d'une rotation droite sur l'arbre tout entier . Le résultat de l'ajout sera donc $rd((clé, rg({t'}_1), t_2))$.
- Les autres cas sont symétriques des précédents.

Remarque: Un ajout donne lieu à au plus une opération de rotation (simple ou double).
Les rotations gauches et droites sont définies sur la figure 2.
Écrire une fonction d'insertion d'une clé dans un AVL. Affichage en couleur de la partie de l'arbre modifiée.

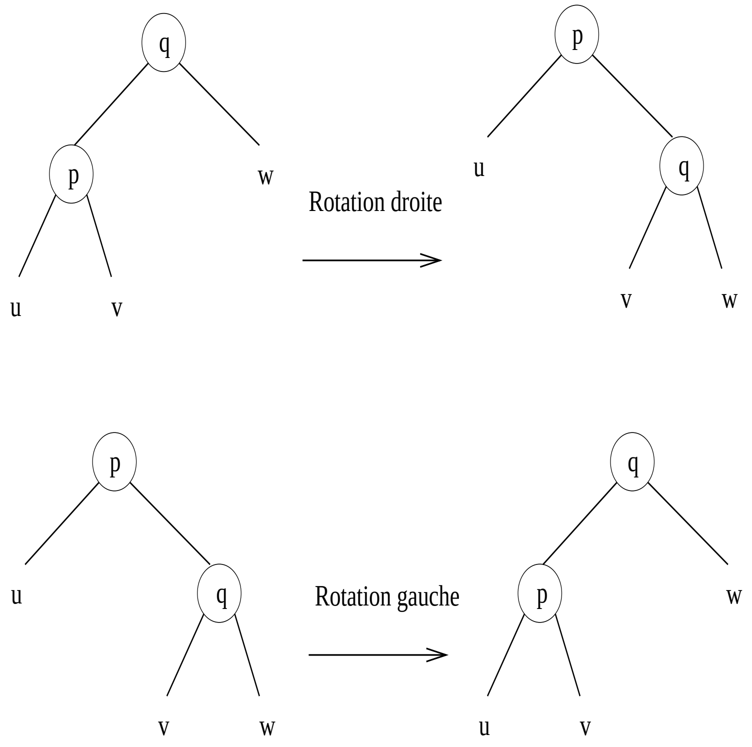

FIG. 2 - *Rotations sur les AVL*

7 Suppression d'une clé d'un AVL

L'opération de retrait dans un arbre AVL repose, comme le retrait dans les arbres binaires de recherche, sur le remplacement de l'élément à éliminer par le plus grand élément du fils gauche du nœud où se trouve cet élément.

L'élimination du plus grand élément d'un arbre est susceptible de réduire la hauteur du sous-arbre droit de cet arbre et de modifier l'équilibre de l'arbre tout entier.

La fonction d'élimination du plus grand élément devra donc retourner également une information supplémentaire indiquant si la taille de l'arbre a décru ou non.

La fonction principale d'élimination recherche l'élément *usagé* à éliminer dans l'arbre, et, lorsqu'il est trouvé, appelle la fonction d'élimination du plus grand élément sur le sous-arbre $(usagé, t_1, t_2)$. Cet appel renvoie un arbre $(plus\_grand\_élément\_de\_t\_1, {t'}_1, t_2)$ et une information sur la variation de hauteur entre t_1 et ${t'}_1$. Cette dernière information est utilisée ensuite pour équilibrer le cas échéant l'arbre $(plus\_grand\_élément\_de\_t\_1, {t'}_1, t_2)$.

De plus le rééquilibrage de l'arbre $(plus\_grand\_élément\_de\_t\_1, {t'}_1, t_2)$ peut produire un arbre de taille inférieure à $(usagé, t_1, t_2)$ qu'il remplace et on doit donc répercuter cette information de façon à procéder à d'autres rééquilibrages en remontant vers la racine.

8 Les dates et le travail demandé

Il vous est demandé de rédiger les programmes, de m'en donner les listings et d'être en mesure d'en faire une démonstration pour le TD du 10 novembre.

Ce projet peut être réalisé seul ou en binôme[1]

Si vous avez le moindre problème, mes coordonnées se trouvent en en-tête du sujet...

1. Un unique trinôme sera autorisé si le nombre des élèves est impair et si aucun élève ne travaille seul !