Travaux Pratiques de Structures de Données n°6

Cours d'Informatique de Seconde Année

— Licence 2.2 —

Arbre AVL v2

L'objectif de ce TP est d'implémenter les fonctions d'ajout et de suppression d'un élément d'un AVL.

Pour l'implémentation de ce type d'arbre, on utilisera la structure suivante :

```
typedef struct noeud
{
int valeur;
int hauteur;
int occurence;
struct noeud *fg, *fd;
} Noeud, *AVL;
```

▸ **Exercice 1.**

- *Écrire une fonction* `int Hauteur(AVL avl)` *qui retourne la hauteur d'un arbre AVL (on suppose que les champs hauteur possèdent des valeurs correct).*
- *Implémenter une fonction* `int recherche(AVL arbre,int a)` *qui retourne le nombre d'occurrence de l'entier* `a` *dans* `racine`.

▸ **Exercice 2.**

- *Écrire la fonction* `void RotationG(AVL* avl)` *qui permet d'effectuer la rotation gauche de l'arbre AVL. Il ne faut pas oublier de mettre à jour la hauteur des noeuds affectés.*
- *Écrire la fonction* `void RotationD(AVL* avl)` *qui permet d'effectuer la rotation droite de l'arbre AVL. Il ne faut pas oublier de mettre à jour la hauteur des noeuds affectés.*

▸ **Exercice 3.** *Implémenter une fonction* `mise_a_jour(AVL* racine)` *qui suppose que les deux sous arbres de* `racine` *sont des AVL et fait les rotations nécessaires pour rendre l'arbre AVL (on suppose que la différence de hauteur entre les deux sous arbres est au plus 2).*

▸ **Exercice 4.** *Implémenter une fonction* `void Ajouter(AVL* arbre,int a)` *qui ajoute un élément* `a` *à l'arbre AVL.*

▸ **Exercice 5.** *Implémenter une fonction* `int Supprimer(AVL* arbre,int a)` *qui supprime un élément* `a` *de l'arbre et retourne le nombre d'occurrences de* `a` *restant dans l'arbre.*