Exercice 3.9.4 « Arbres AVL »

1. Démontrer que la hauteur d'un AVL est en $\theta(log(n))$.
2. Proposer un algorithme d'insertion d'une clé dans un AVL en $O(log(n))$ dans le pire des cas.
3. Proposer un algorithme de suppression d'une clé dans un AVL en $O(log(n))$ dans le pire des cas.
4. Quelle la complexité finale de chacune des 7 opérations sur un AVL dans le pire des cas ?

La preuve de la hauteur d'un AVL est donnée dans le cours dans la Proposition 9.

Pour l'insertion et la suppression, nous allons exploiter la solution donnée dans la référence :

http://gallium.inria.fr/~maranget/X/421/poly/poly.pdf

```
class Avl {
    int contenu;
    int hauteur;
    Avl filsG, filsD;

    Avl(Avl g, int c, Avl d) {
        filsG = g;
        contenu = c;
        filsD = d;
        hauteur = 1 + Math.max(H(g), H(d));
    }

    static int H(Avl a) {
        return (a == null) ? -1 : a.hauteur;
    }

    static void calculerHauteur(Avl a) {
        a.hauteur = 1 + Math.max(H(a.filsG), H(a.filsD));
    }

    static Avl rotationG(Avl a) {
        Avl b = a.filsD;
        Avl c = new Avl(a.filsG, a.contenu, b.filsG);
        return new Avl(c, b.contenu, b.filsD);
    }


    static Avl rotationD(Avl a) {
        Avl b = a.filsG;
        Avl c = new Avl(b.filsD, a.contenu, a.filsD);
        return new Avl(b.filsG, b.contenu, c);
    }
```

```
static Avl equilibrer(Avl a) {
    a.hauteur = 1 + Math.max(H(a.filsG), H(a.filsD));
    if (H(a.filsG) - H(a.filsD) == 2) {
        if (H(a.filsG.filsG) < H(a.filsG.filsD))
            a.filsG = rotationG(a.filsG);
        return rotationD(a);
    } // else version symétrique
    if (H(a.filsG) - H(a.filsD) == -2) {
        if (H(a.filsD.filsD) < H(a.filsD.filsG))
            a.filsD = rotationD(a.filsD);
        return rotationG(a);
    }
    return a;
}
```

```
static Avl inserer(int x, Avl a) {
    if (a == null)
        return new Avl(null, x, null);
    if (x < a.contenu)
        a.filsG = inserer(x, a.filsG);
    else if (x > a.contenu)
        a.filsD = inserer(x, a.filsD);
    return equilibrer(a); // seul changement
}
```

```
    static Avl supprimer(int x, Avl a) {
        if (a == null)
            return a;
        if (x == a.contenu)
            return supprimerRacine(a);
        if (x < a.contenu)
            a.filsG = supprimer(x, a.filsG);
        else
            a.filsD = supprimer(x, a.filsD);
        return equilibrer(a); // seul changement
    }

    static Avl supprimerRacine(Avl a) {
        if (a.filsG == null && a.filsD == null)
            return null;
        if (a.filsG == null)
            return equilibrer(a.filsD);
        if (a.filsD == null)
            return equilibrer(a.filsG);
        Avl b = min(a.filsD);
        a.contenu = b.contenu;
        a.filsD = supprimer(a.contenu, a.filsD);
        return equilibrer(a); // seul changement
    }

    static Avl min(Avl a) // min
    {
        if (a.filsG == null)
            return a;
        return min(a.filsD);
    }

}
```

Illustrations :
https://courses.cs.washington.edu/courses/cse326/01au/lectures/AVLTrees.ppt