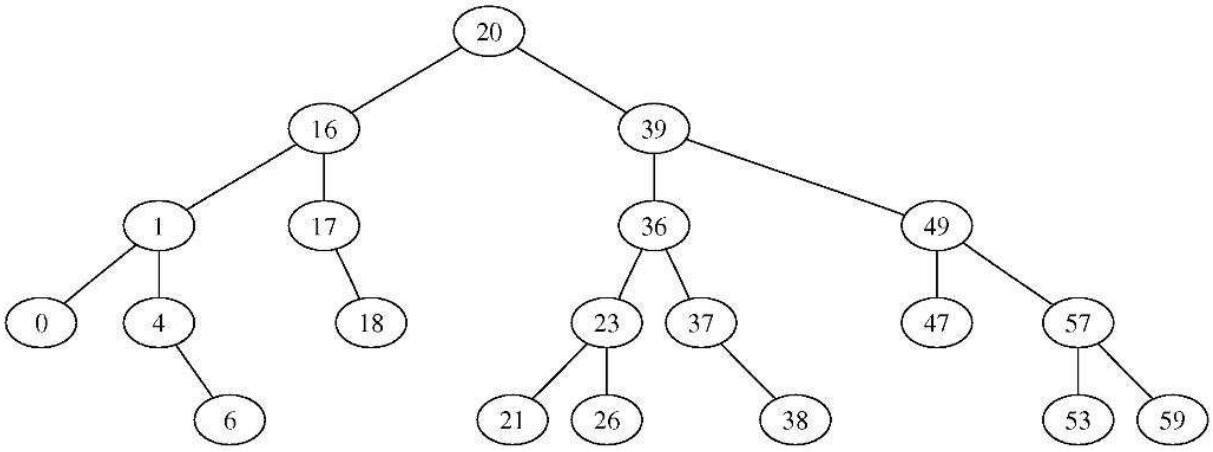
Algorithmique et programmation procédurale

TD No 7

Arbres AVL

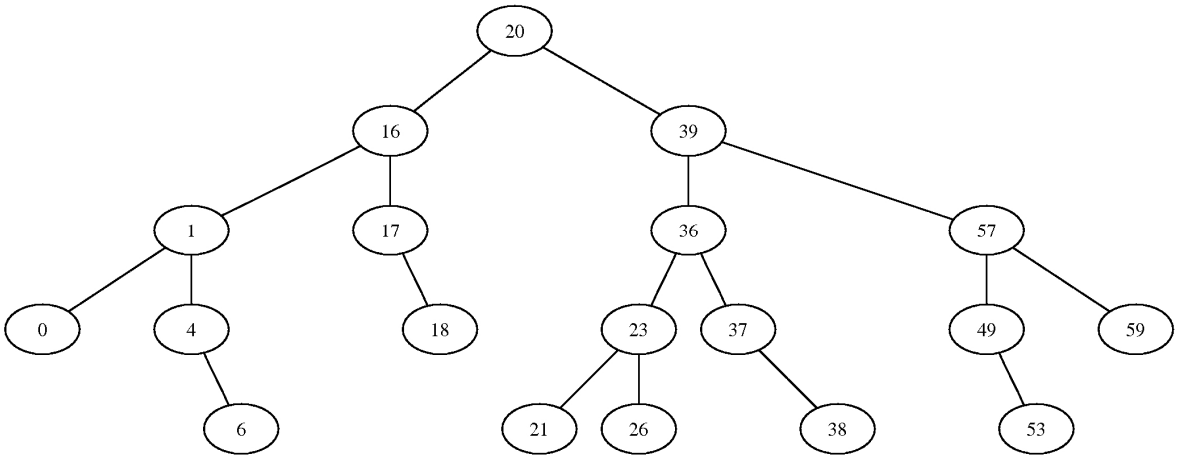
Exercice 1 : Soit l'arbre



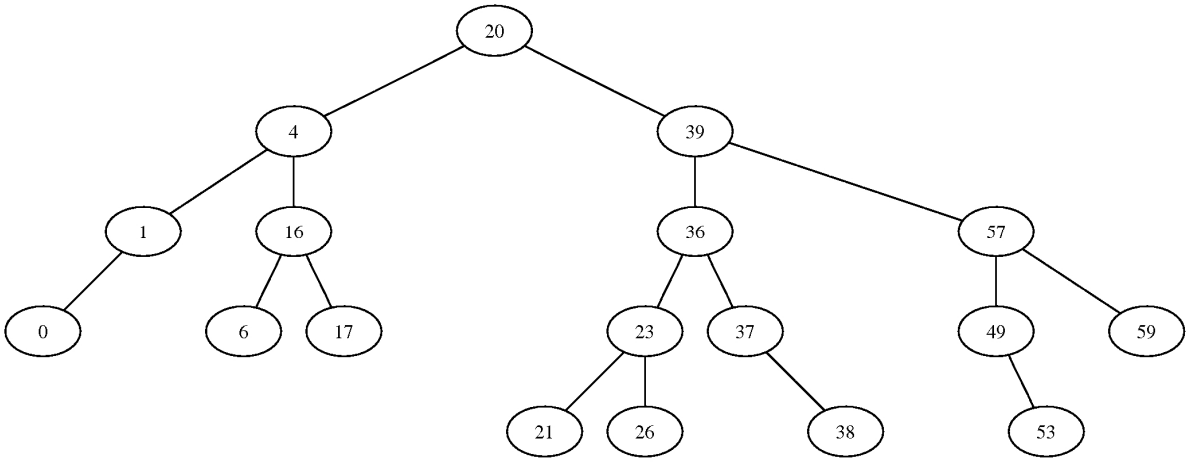
a) Vérifier que cet arbre est de type AVL.
b) Expliquer comment se fait la suppression de l'élément 47, puis la suppression de l'élément 18, puis l'ajout de 51. Dessiner l'arbre obtenu après chaque opération.

Corrigé :

La suppression de 47 déclenche une rotation gauche en 49 et la mise à jour de l'arbre s'arrête à ce niveau.



La suppression de 18 déclenche une rotation gauche droite en 16 et la mise à jour de l'arbre s'arrête à ce niveau.



Exercice 2 :

1) Il existe combien d'architectures d'arbre AVL à 7 sommets ?
2) Dessiner un arbre AVL de hauteur 5 avec un nombre de nœuds minimal.

Corrigé

1) 17 arbres (1 arbre complet + 4 x 4 arbres dont la différence de hauteur = 1)
2) Arbre à 20 nœuds (fils gauche de hauteur 4, fils droite de hauteur 3, fils gauche de fils gauche de hauteur 3, fils droit de fils gauche de hauteur 2, …).

   Question générale : quel est le nombre minimal de nœuds que peut avoir un arbre AVL de hauteur *h*.

   Nous avons :

$$NB_0^{\min} = 0,\ NB_1^{\min} = 1,\ NB_2^{\min} = 2$$
$$NB_h^{\min} = 1 + NB_{h-1}^{\min} + NB_{h-2}^{\min} \ \text{pour}\ h \geq 2$$

   Cette récurrence a pour solution le nombre de Fibonacci au rang *h+2* moins 1. Soit :

$$NB_h^{\min} = F_{h+2} - 1$$
$$NB_h^{\min} = 1/\sqrt{5} * (((1+\sqrt{5})/2)^{h+2} - ((1+\sqrt{5})/2)^{h+2}) - 1$$
$$NB_h^{\min} \approx 1/\sqrt{5} * (((1+\sqrt{5})/2)^{h+2} - 1$$

   ⇨ Pour hauteur 5, $F_7 = 21$ => nombre de nœuds = 20.

Exercice 3 : Écrire en pseudo-code l'algorithme qui permet de supprimer un sommet dans un arbre AVL.

Corrigé

Fonction supprimer(x: **T**, A: **AVL**): **AVL**
Début

```
        Si estVide(A) Alors
                retourner arbreVide
        SinonSi x > racine(A)
                retourner equilibrer(cons(racine(A), fG(A), supprimer(x,fD(A))))
        SinonSi x < racine(A)
                retourner equilibrer(cons(racine(A), supprimer(x,fG(A)),fD(A)))
        SinonSi estVide(fG(A))
                retourner fD(A)
        SinonSi estVide(fD(A))
                retourner fG(A)
        Sinon
                retourner equilibrer(cons(max(fG(A)), supprimerMax(fG(A)), fD(A)))
        FinSi
Fin

Fonction supprimerMax(A : AVL): AVL
Début
        Si estVide(fD(A)) Alors
                retourner fG(A)
        Sinon
                retourner equilibrer(cons(racine(A),fG(A),supprimerMax(fD(A)))
        FinSi
Fin

Fonction max(A : AVL): T
Début
        Si estVide(fD(A)) Alors
                Retourner racine(A)
        Sinon
                Retourner max(fD(A))
        FinSi
Fin
```

Exercice 4 :

Écrire un algorithme qui réalise la fusion de deux AVL.

```
Fonction fusion(a1 : AVL, a2 : AVL): AVL
Début
        Si estVide(a1) Alors
                retourner a2
        Sinon Si estVide(a2)
                retourner a1
        Sinon
                retourner fusion(fG(a1), fusion(fD(a1), inserer(racine(a1),a2)))
        FinSi
Fin
```

Exercice 5 :

Écrire un algorithme qui réalise « la scission d'un AVL en x », c'est-à-dire qu'il crée deux AVL a1 et a2 où a1 contient les valeurs inférieures ou égales à x, et a2 contient les valeurs strictement supérieures à x.

```
Procédure scission(a : AVL, x : Entier, S a1 : AVL, S a2 : AVL)
Début
    Si estVide(a) Alors
        a1 <- arbreVide
        a2 <- arbreVide
    Sinon Si racine(a) <= x
        scission(fD(a), x, a1, a2)
        a1 <- fusion(inserer(racine(a), fG(a)), a1)
    Sinon
        scission(fG(a), x, a1, a2)
        a2 <- fusion(inserer(racine(a), fD(a)), a2)
    FinSi
Fin
```