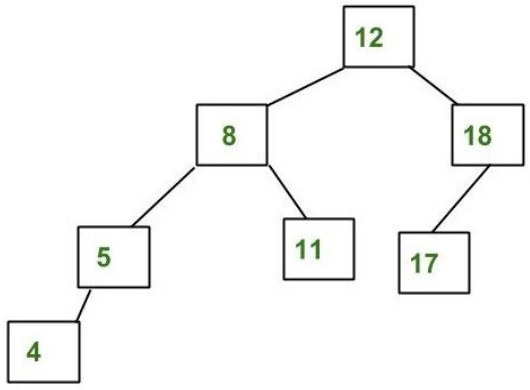
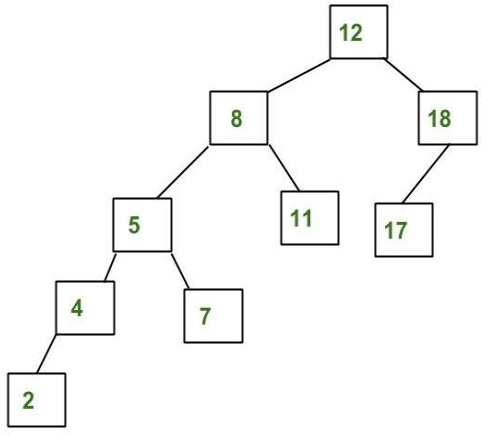
Introduction aux arbres AVL

[1]L'arbre AVL est un arbre de recherche binaire (BST) auto-équilibré où la différence entre les hauteurs des sous-arbres gauche et droit ne peut pas être plus d'un pour tous les nœuds.

Un exemple d'arbre qui est un arbre AVL

	L'arbre suivant est AVL car les différences entre les hauteurs des sous-arbres gauche et droit pour chaque nœud sont inférieures ou égales à 1.

Un exemple d'arbre qui n'est PAS un arbre AVL

	L'arbre suivant n'est pas AVL car les différences entre les hauteurs des sous-arbres gauche et droit pour 8 et 12 sont supérieures à 1.

Pourquoi les arbres AVL ?

La plupart des opérations qui requièrent un parcours en profondeur d'un arbre (par exemple, recherche, max, min, insertion, suppression, etc.) prennent un temps $O(h)$ où h est la hauteur d'un ABR. Le coût de ces opérations peut devenir $O(n)$ pour un

[1] https://www.geeksforgeeks.org/avl-tree-set-1-insertion/

arbre binaire asymétrique. Si nous nous assurons que la hauteur de l'arbre reste $O(log\ n)$ après chaque insertion et suppression, alors nous pouvons garantir une limite supérieure de $O(log\ n)$ pour toutes ces opérations. La hauteur d'un arbre AVL est toujours $O(log\ n)$, où n est le nombre de nœuds dans l'arbre (voir cette vidéo pour preuve).

Insertion dans un AVL

Pour nous assurer qu'un ABR reste AVL après chaque insertion, nous devons adapter l'opération d'insertion standard dans un ABR pour effectuer un rééquilibrage. Voici deux opérations de base qui peuvent être effectuées pour rééquilibrer un ABR sans violer la propriété d'un ABR.

1) Rotation gauche

2) Rotation droite

```
T1, T2 et T3 sont des sous-arbres de l'arbre
enraciné avec y (sur le côté gauche) ou x (sur
Le côté droit)
     yx
    / \ Rotation à droite / \
   x T3 - - - - - - -> T1 y
  / \ <- - - - - - - / \
 T1 T2 Rotation gauche T2 T3
Les clés des deux arbres ci-dessus suivent
ordre suivant
 touches (T1) <touche (x) <touches (T2) <touche (y) <touches (T3)
La propriété BST n'est donc violée nulle part.
```

Étapes à suivre pour l'insertion

Considérons le nœud nouvellement inséré comme étant w.

1) Effectuer une insertion standard pour w.

2) En partant de w, remonter et trouver le premier nœud déséquilibré. Soit z le premier nœud déséquilibré, y l'enfant de z qui vient sur le chemin de w à z et x le petit-enfant de z qui vient sur le chemin de w à z.

3) Rééquilibrer l'arbre en effectuant les rotations appropriées sur le sous-arbre enraciné avec z. Il peut y avoir 4 cas possibles qui doivent être traités car x, y et z peuvent être arrangés de 4 manières. Voici les 4 arrangements possibles:

a) y est l'enfant gauche de z et x est l'enfant gauche de y (cas gauche gauche)

b) y est l'enfant gauche de z et x est l'enfant droit de y (cas gauche droite)

c) y est l'enfant droit de z et x est l'enfant droit de y (cas droite droite)

d) y est l'enfant droit de z et x est l'enfant gauche de y (cas droite gauche)

Voici les opérations à effectuer dans les 4 cas mentionnés ci-dessus. Dans tous les cas, il suffit de rééquilibrer le sous-arbre enraciné avec z et l'arbre complet devient équilibré lorsque la hauteur du sous-arbre (après des rotations appropriées) enracinée avec z devient la même qu'avant l'insertion. (Voir cette vidéo pour preuve)

a) Cas gauche gauche

```
T1, T2, T3 et T4 sont des sous-arbres.
         zy
        / \ / \
       y T4 Rotation à droite (z) xz
      / \ - - - - - - - - -> / \ / \
     x T3 T1 T2 T3 T4
    / \
  T1 T2
```

b) Cas gauche droite

```
    zzx
    / \ / \ / \
   y T4 Rotation gauche (y) x T4 Rotation droite (z) yz
  / \ - - - - - - - - -> / \ - - - - - - - -> / \ / \
T1 xy T3 T1 T2 T3 T4
    / \ / \
  T2 T3 T1 T2
```

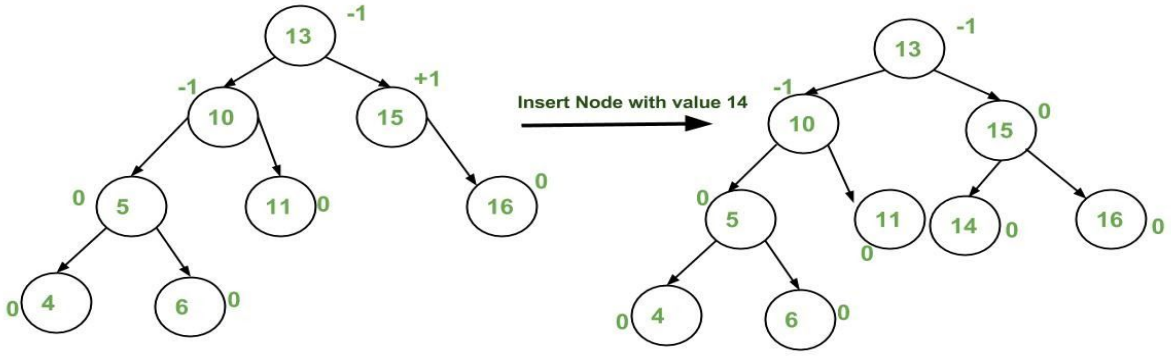
c) Cas droite droite

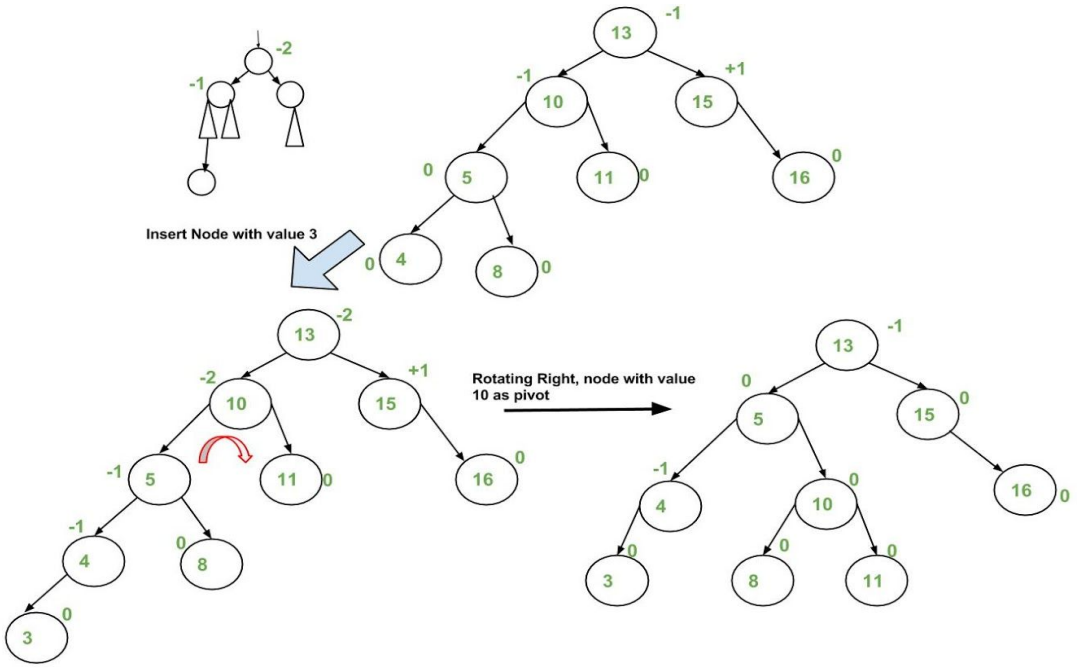
```
 zy
 / \ / \
T1 y Rotation gauche (z) zx
    / \ - - - - - - - -> / \ / \
   T2 x T1 T2 T3 T4
      / \
     T3 T4
```

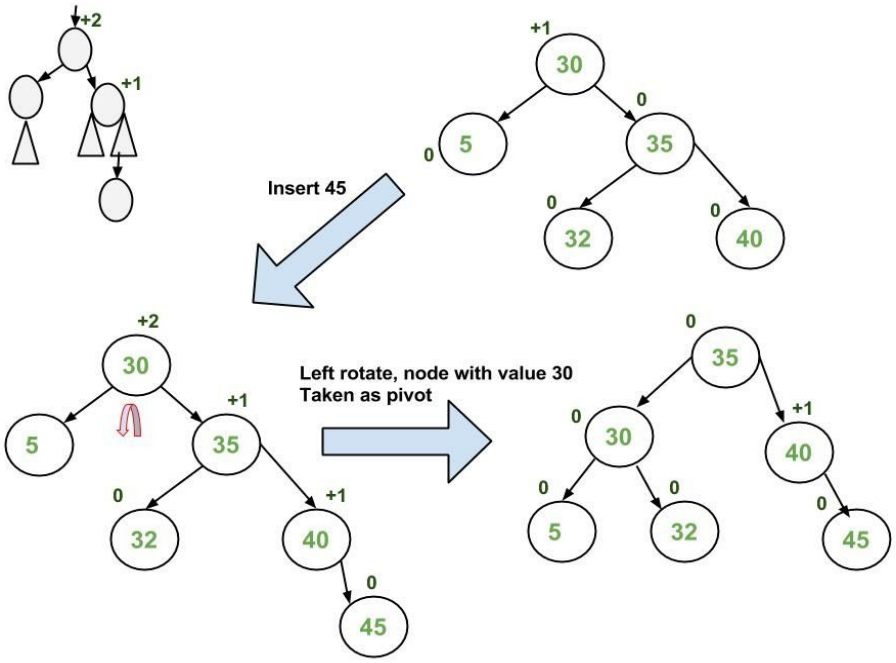
d) Cas droite gauche

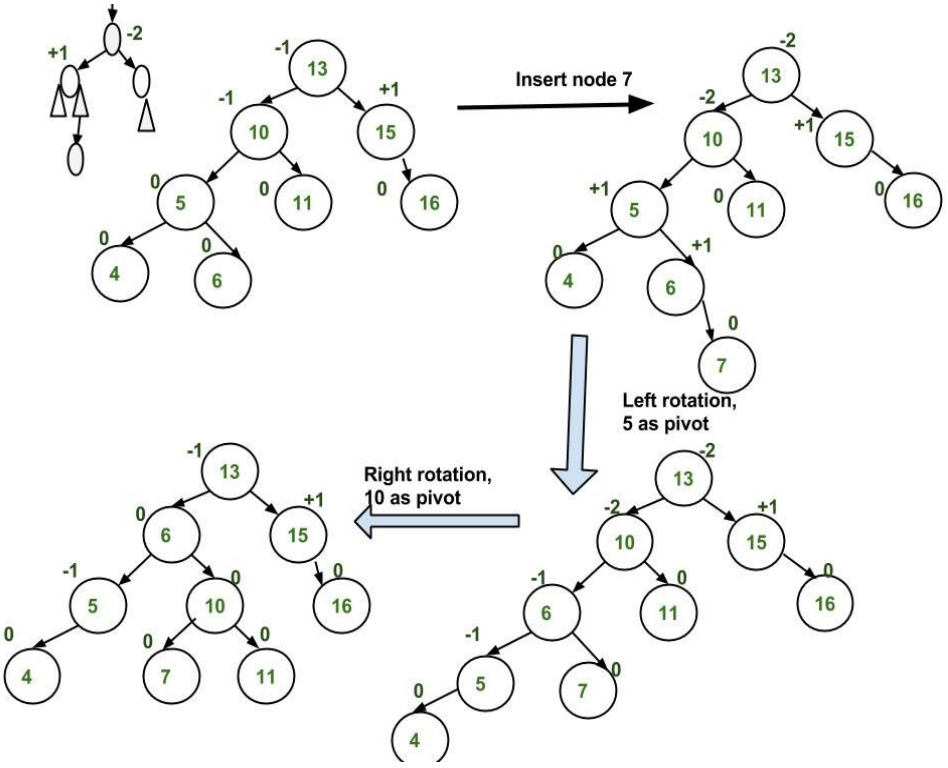
```
  zzx
  / \ / \ / \
T1 y Rotation à droite (y) T1 x Rotation à gauche (z) zy
    / \ - - - - - - - - -> / \ - - - - - - - -> / \ / \
   x T4 T2 et T1 T2 T3 T4
  / \ / \
T2 T3 T3 T4
```

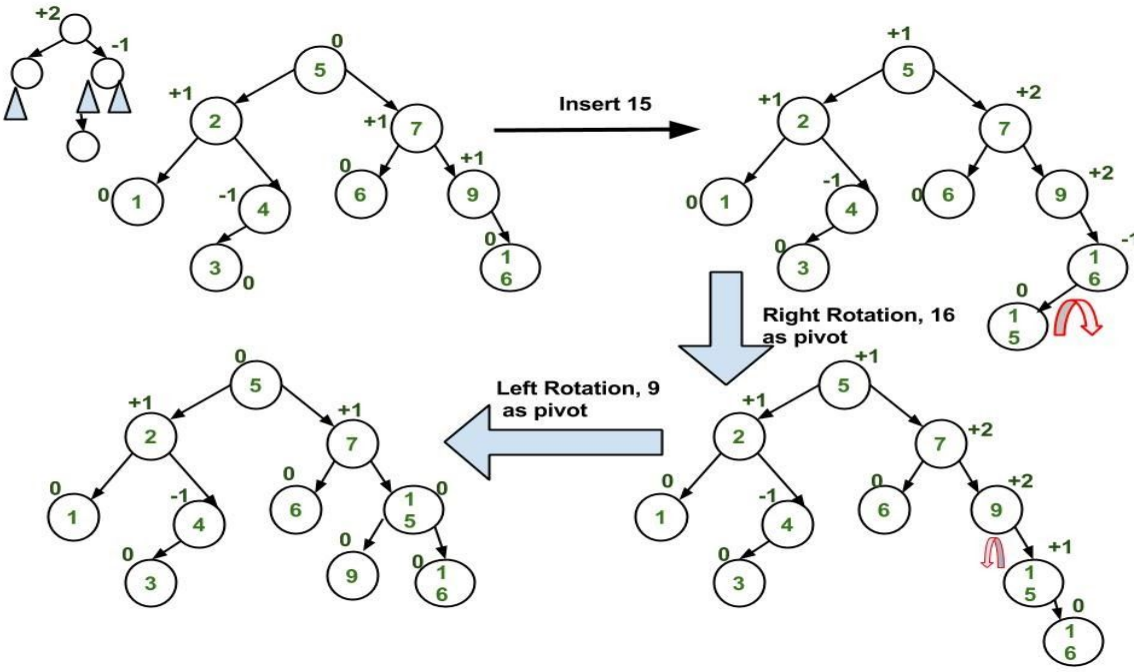
Exemples d'insertion:






+2
+1
Insert 45
+1
30
0
5
35
0
0
32
0
40
+2
30
Left rotate, node with value 30
Taken as pivot
5
+1
35
0
32
+1
40
0
45
0
35
0
30
+1
40
0
0
5
32
0
45


+1
-2
-1
13
-1
10
+1
15
0
5
0
11
0
16
0
4
0
6
Insert node 7
-2
13
-2
10
+1
15
+1
5
0
11
0
16
0
4
+1
6
0
7
Left rotation,
5 as pivot
-1
13
0
6
+1
15
-1
5
0
10
0
16
0
4
0
7
0
11
Right rotation,
10 as pivot
-2
13
-2
10
+1
15
-1
6
0
11
0
16
-1
5
0
7
0
4



Implémentation

L'implémentation peut utiliser la version récursive de l'insertion dans un ABR pour insérer un nouveau nœud et . Après l'insertion, nous obtenons des pointeurs vers tous les ancêtres un par un de manière ascendante. Nous n'avons donc pas besoin du pointeur parent pour remonter. Le code récursif lui-même parcourt et visite tous les ancêtres du nœud nouvellement inséré.

1) Effectuer l'insertion normale dans un ABR.

2) Le nœud actuel doit être l'un des ancêtres du nœud nouvellement inséré. Mettre à jour la hauteur du nœud actuel.

3) Calculer le facteur d'équilibre (hauteur du sous-arbre gauche - hauteur du sous-arbre droit) du nœud actuel.

4) Si le facteur d'équilibre est supérieur à 1, alors le nœud actuel est déséquilibré et nous sommes soit dans le cas **gauche gauche** ou **gauche droite**. Pour vérifier si c'est le cas **gauche gauche** ou non, comparer la donnée nouvellement insérée avec celle dans la racine du sous-arbre gauche.

5) Si le **facteur d'équilibre est inférieur à -1**, alors le nœud actuel est **déséquilibré** et nous sommes soit dans le cas **droite droite** ou **droite gauche**. Pour vérifier s'il s'agit du cas **droite droite** ou non, comparez la donnée nouvellement insérée avec la donnée dans la racine du sous-arbre droit.

Tâches à Faire

En vous basant sur vos propres recherches et sur les ressources du cours, proposez un programmes avec les fonctions suivantes:

- insertion dans un arbre AVL,
- suppression dans un arbre AVL,
- recherche dans un arbre AVL,
- transformation d'un arbre binaire de recherche (ABR) en un arbre AVL,
- menu de choix pour effectuer les différentes opérations dans un AVL.

Afin de mieux tester l'utilité des arbres AVL,

1. créer un ABR non AVL;
2. faire une fonction itérant sur l'insertion successive de 1000 à 10000 valeurs aléatoires dans l'ABR, avec une intervalle de 1000, c'est-à-dire, vous testez d'abord sur 1000, ensuite 2000, ensuite 3000, …, jusqu'à 10000 (remarque: vous pouvez aller aussi au delà, rien ne vous empêche de tester avec 100000 noeuds);
3. pour chaque test à la question précédente, appeler une fonction de recherche dans cet ABR, et estimer la durée de calcul;
4. comparer les durées en temps de calcul les différentes cardinalités (1000, 2000, 3000, 4000, 5000, …, 10000);
5. Qu'observez-vous ?
6. Refaire les étapes 2 à 4 cette fois ci avec un arbre AVL.
7. Qu'observez-vous ?
8. Comparer les durées obtenues avec les ABR et celles obtenues avec les AVL.
9. Quelles conclusions tirez-vous de ces étapes et comparaisons ?