# Cours 12 : Arbres AVL

Algorithmique et programmation procédurale

# Arbres AVL

- Créé par Adelson-Velskii et Landis (1962)
- Définition
  - On appelle **arbre AVL** tout ABR tel que, pour tout sommet, la différence des hauteurs des sous-arbres gauche et droit est en valeur absolue inférieure ou égale à 1.

# Arbres AVL

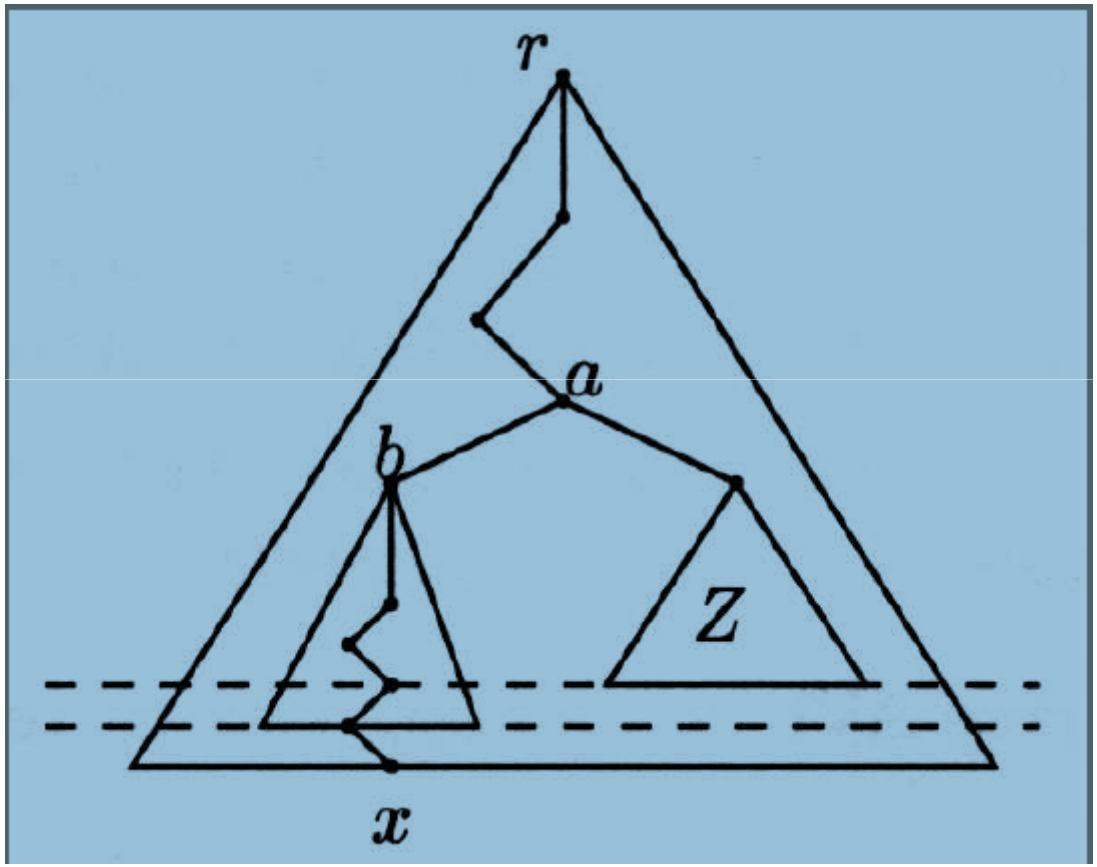
- Théorème
  - Soit A un arbre AVL ayant n sommets, on a :

$$\log_2 (n+1)-1 \leq \text{hauteur}(A) \leq \Phi \log_2 (n+2)-1$$

où $\Phi$ est le nombre d'or = $(1+\sqrt{5})/2 \approx 1{,}44$

- Opérations
  - Ajout d'un élément
  - Suppression d'un élément

# Ajout d'un élément

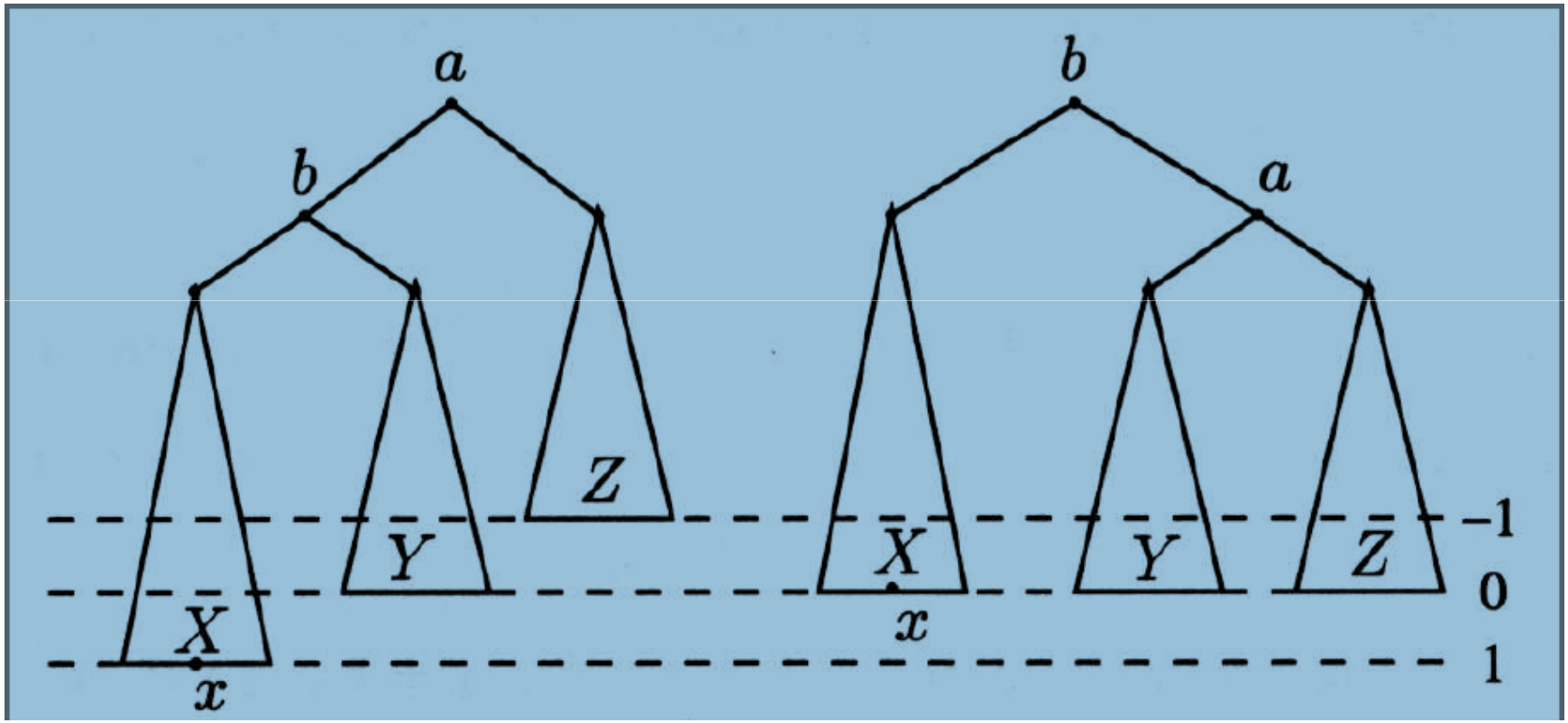



# Ajout d'un élément : Algorithme

- Phase 1 : insérer l'élément x comme une nouvelle feuille dans un ABR
- Phase 2 : remonter la branche qui relie x à la racine :
  - Si tous les arbres enracinés sur cette branche sont des arbres AVL => rien à faire
  - S'il existe un sommet sur cette branche où la condition AVL n'est pas respectée :
    - Noter a le **premier sommet** sur la branche (en partant de la feuille) où le déséquilibre se produit
    - Supposer que l'insertion s'est faite dans le sous-arbre gauche de a (manière analogue pour sous-arbre droit)
    - On distingue 2 cas :

# Ajout d'un élément : Algorithme (suite)

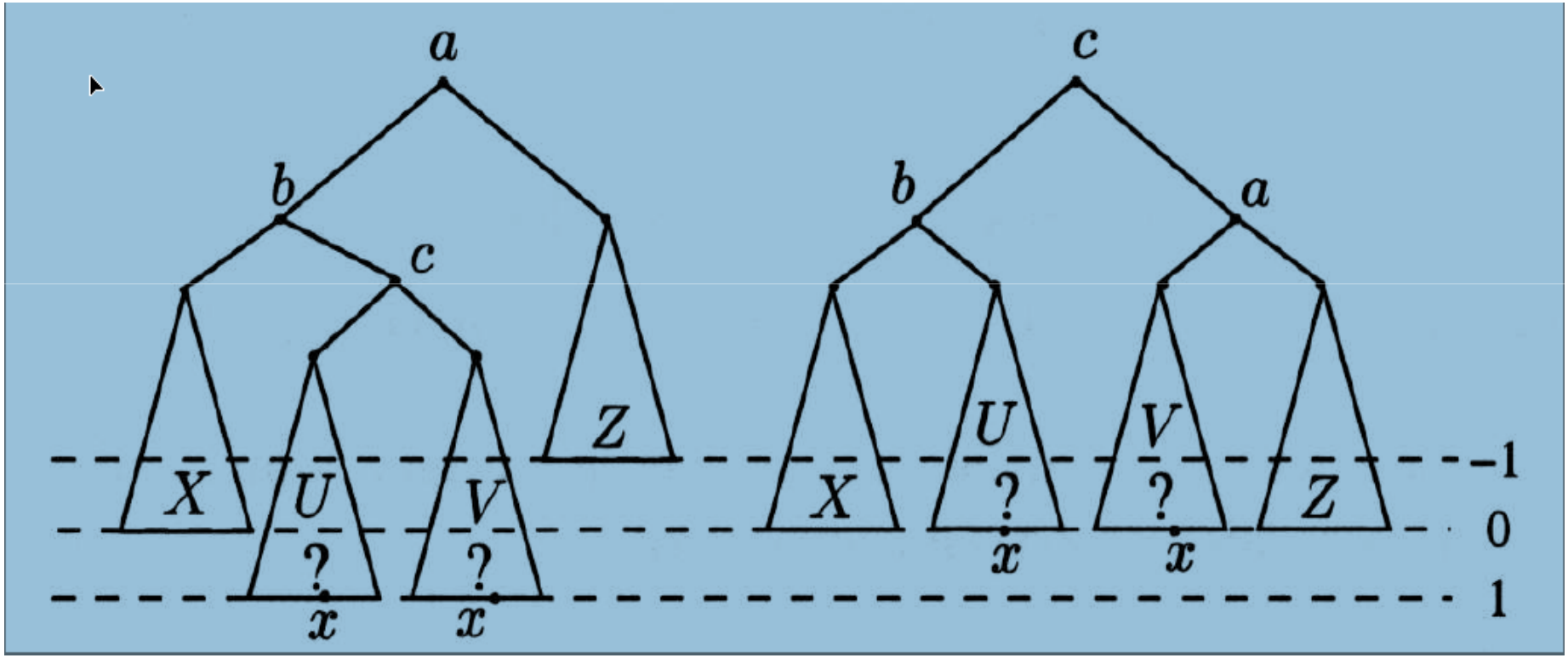
- Cas 1 : x est ajouté dans le sous-arbre gauche X




- *une rotation droite en a suffit pour rééquilibrer l'arbre original*

# Ajout d'un élément : Algorithme (suite)

- Cas 2 : x est ajouté dans le sous-arbre droit Y




- *une rotation gauche-droite en a suffit pour rééquilibrer l'arbre original*

# Ajout d'un élément

- Complexité
  - L'ajout d'un élément dans un arbre AVL ayant n éléments se fait en temps O(log(n)) et ne nécessite qu'une rotation (simple ou double).

# Exemple (exercice)

- Créer un arbre AVL en ajoutant successivement les nombres 12, 3, 2, 5, 4, 7, 9, 11, 14 et 10.

# Corrigé

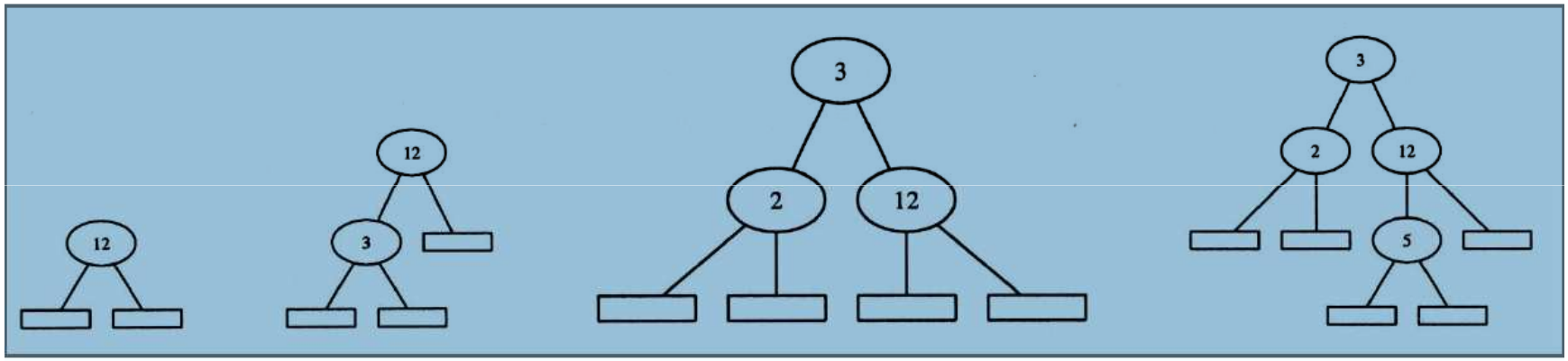



FIG.: Insertion de 12, 3, 2 (rotation droite) et 5.

# Corrigé

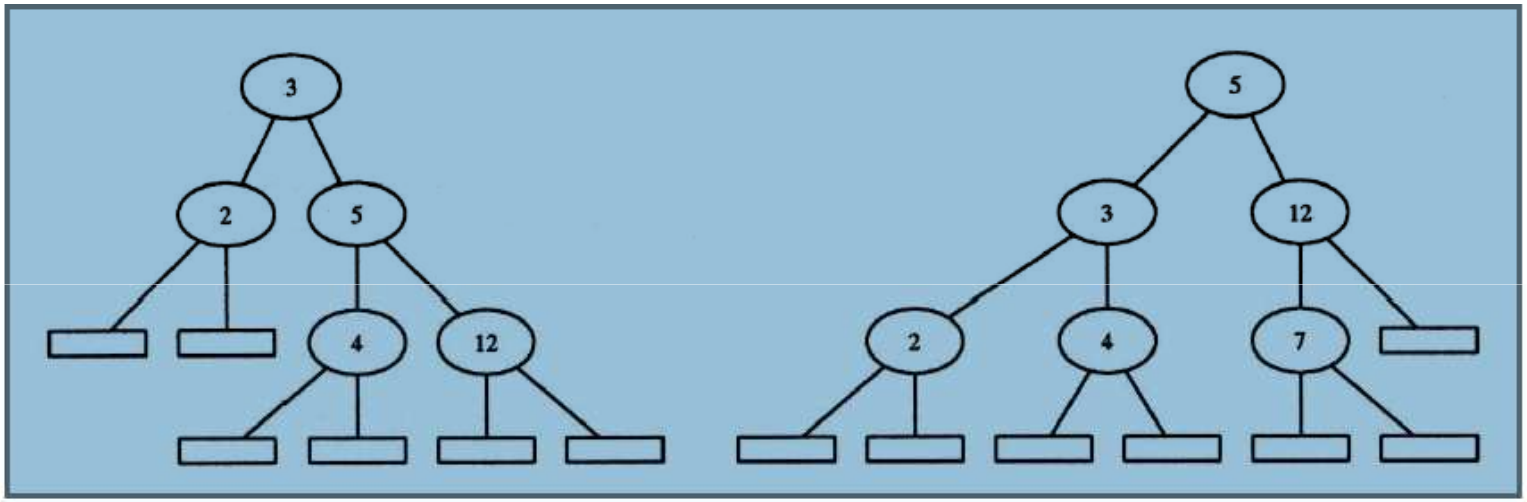



FIG.: Insertion de 4 (rotation droite) et 7 (rotation gauche).

# Corrigé

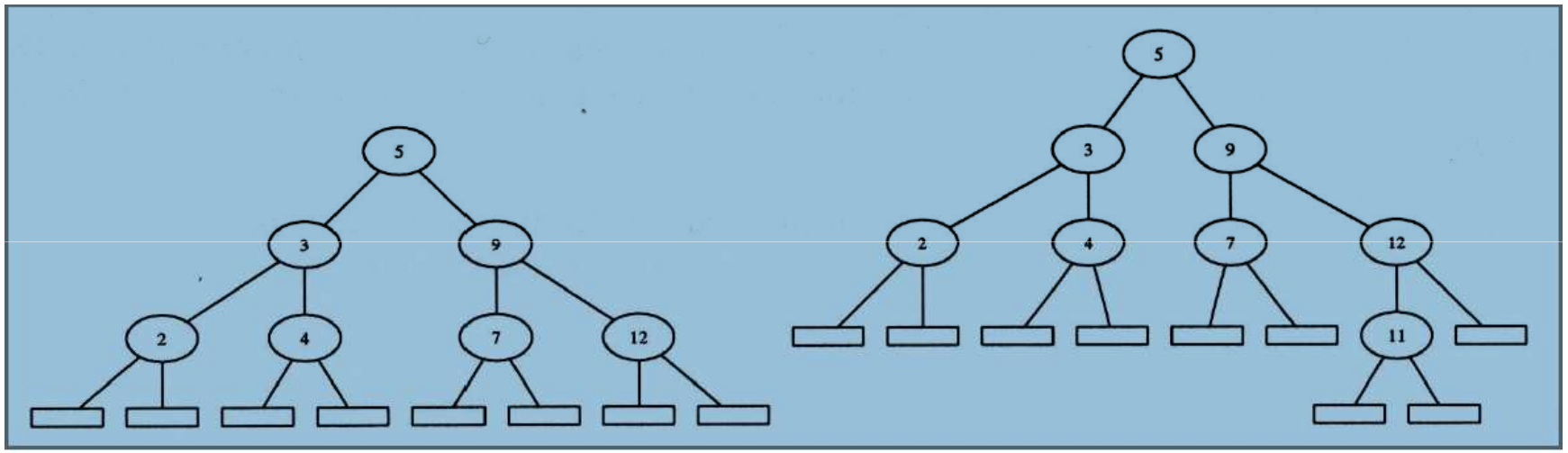



FIG.: Insertion de 9 (rotation gauche-droite) et 11.

# Corrigé

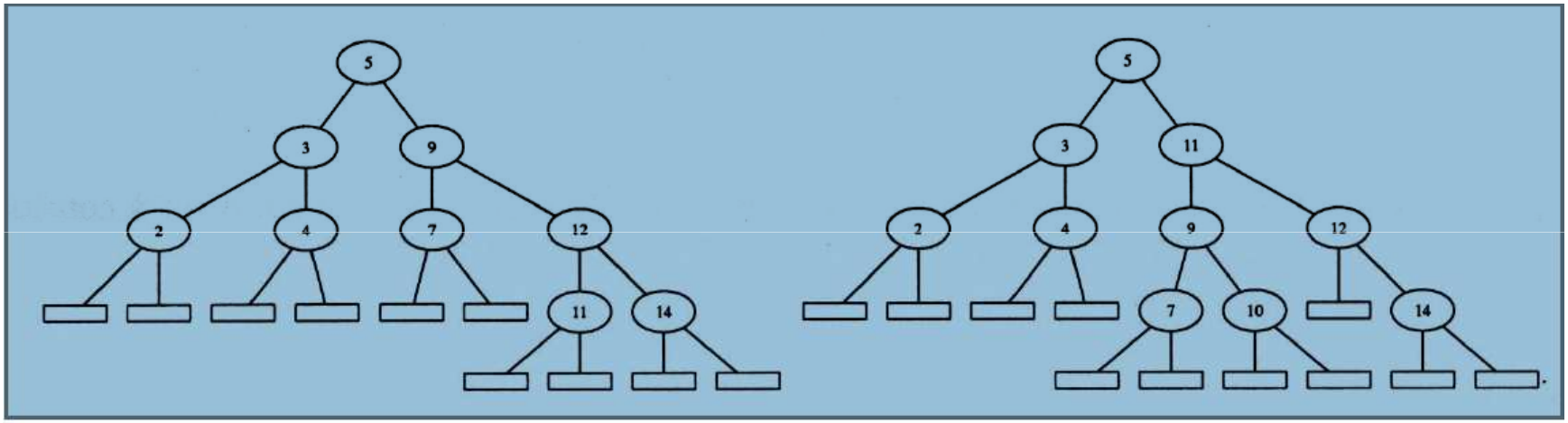



FIG.: Insertion de 14 et 10 (rotation droite-gauche).

# Pseudo-code : fonction ajouter

```
Fonction ajouter(x: T, A: AVL): AVL
Début
  Si estVide(A) Alors
       Retourner cons(x,arbreVide,arbreVide)
  SinonSi x = r(A)
       Retourner A
  SinonSi x > r(A)
       Retourner equilibrer(cons(r(A),fG(A),ajouter(x,fD(A))))
  Sinon
       Retourner equilibrer(cons(r(A),ajouter(x,fG(A)),fD(A)))
  FinSi
Fin
```

# Pseudo-code : fonction écart

Ecrire une fonction qui renvoie la différence de hauteur entre le fils gauche et le fils droit d'un arbre binaire.

# Pseudo-code : fonction écart

```
Fonction ecart(A : ABR): Entier
Début
        Si estVide(A) Alors
                Retourner 0
        Sinon
                Retourner hauteur(fG(A)) – hauteur(fD(A))
        FinSi
Fin
```

# Pseudo-code : fonction équilibrer

Ecrire la fonction qui rééquilibre un ABR.

```
Fonction équilibrer(A: ABR): AVL
Début
  Si |ecart(A)| <= 1 Alors          // l'arbre est déjà AVL
       Retourner A
  SinonSi ecart(A) = 2
       ...                          // l'insertion s'est faite dans sous-arbre gauche
  Sinon { ecart = -2 }
       ...                          // l'insertion s'est faite dans sous-arbre droit
  FinSi
Fin
```

# Pseudo-code : fonction équilibrer

```
Fonction équilibrer(A: ABR): AVL
Début
   Si |ecart(A)| <= 1 Alors                    // arbre AVL
          Retourner A
   SinonSi ecart(A) = 2                        // insertion s'est faite dans sous-arbre gauche
         Si ecart(fG(A)) = 1 Alors             // cas 1 : ajout dans sous-arbre gauche de fG
                  Retourner rotDroite(A)
         Sinon  { ecart(fG(A)) = -1 }          // cas 2 : ajout dans sous-arbre droit de fG
                  Retourner rotGaucheDroite(A)
         FinSi
   Sinon  {ecart(A) = 2 }                      // insertion s'est faite dans sous-arbre droit
         Si ecart(fD(A)) = -1 Alors            // traiter de manière analogue
                  Retourner rotGauche(A)
         Sinon  { ecart(fD(A)) = -1 }
                  Retourner rotDroiteGauche(A)
         FinSi
   FinSi
Fin
```

# Suppression d'un élément : Algorithme

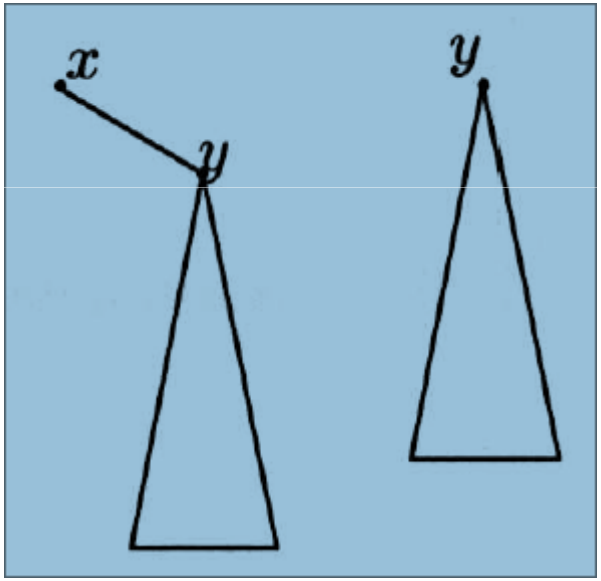
- Phase 1 : rechercher l'élément à supprimer x
  - a) Si x est une feuille => supprimer cette feuille

# Suppression d'un élément : Algorithme

- Phase 1 : rechercher l'élément à supprimer x
  - b) Si x est un sommet interne, 3 cas à envisager :

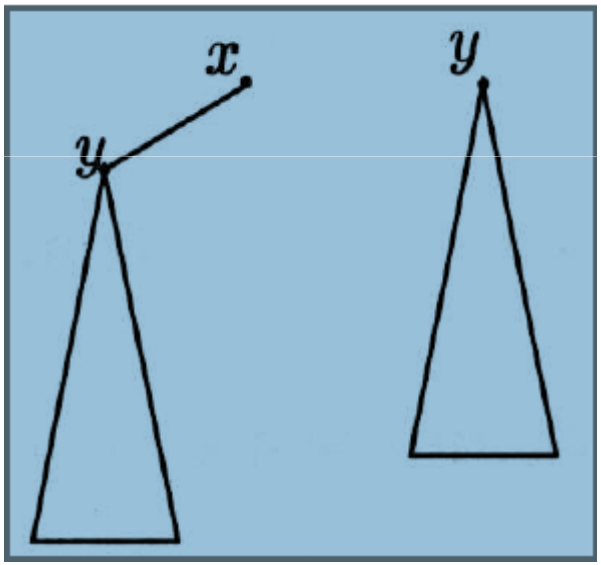
# Suppression d'un élément : Algorithme (suite)

- Cas 1 : Si x n'a pas de fils gauche, remplacer x par son fils droit y



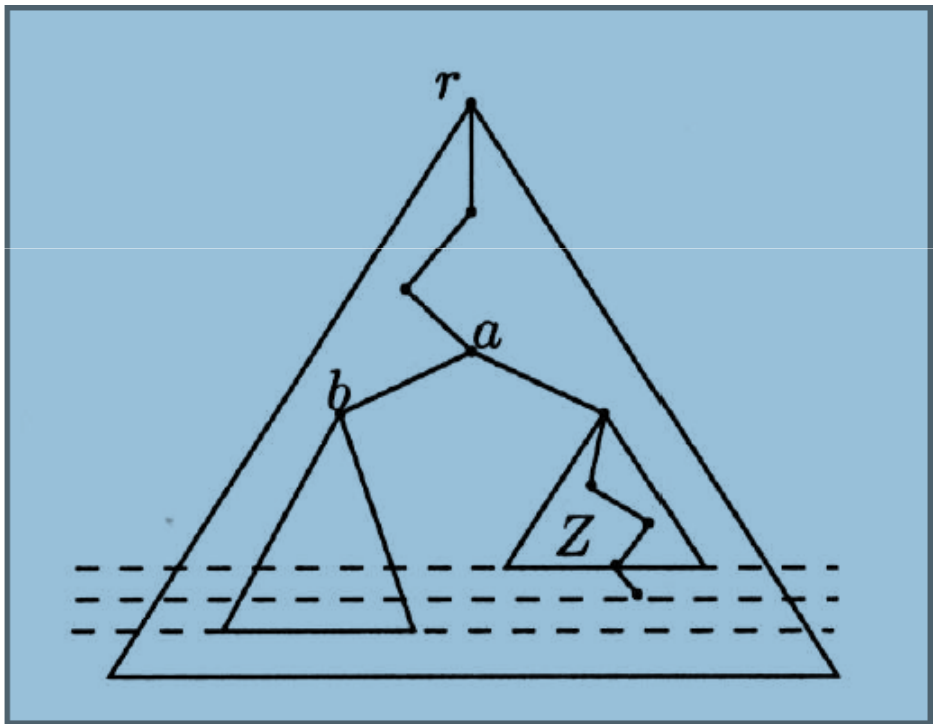

# Suppression d'un élément : Algorithme (suite)

- Cas 2 : Si x n'a pas de fils droit, remplacer x par son fils gauche y




# Suppression d'un élément : Algorithme (suite)

- Cas 3 : x a deux fils, remplacer x par son prédécesseur dans l'arbre (ou par son successeur)



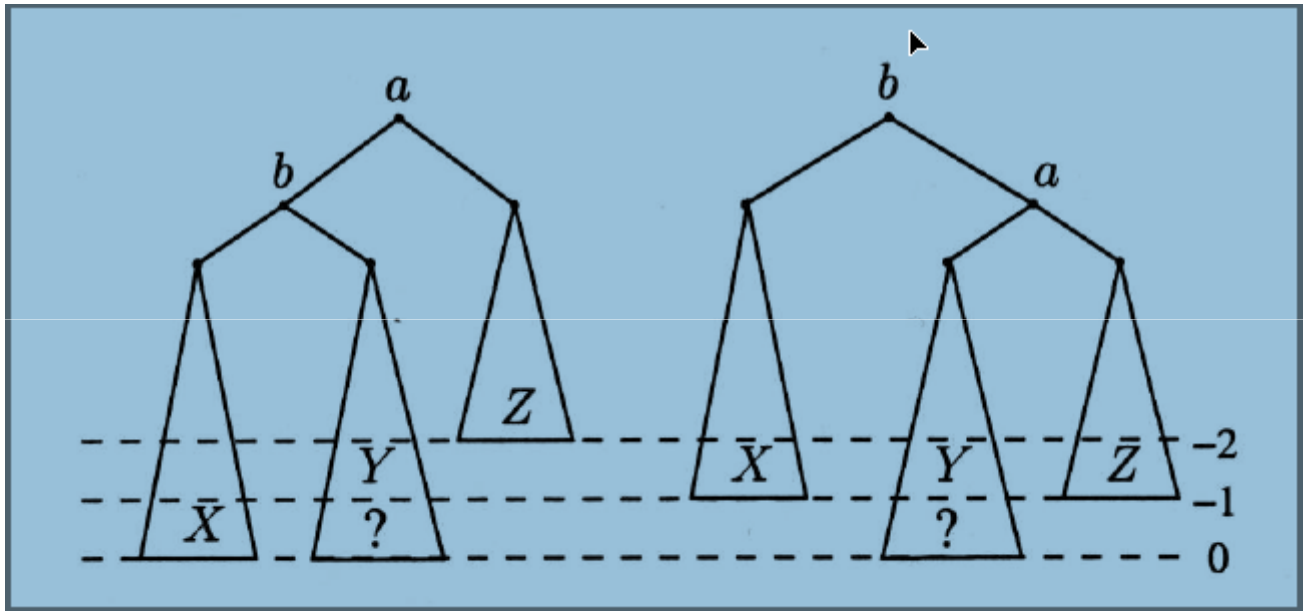

- *À la fin, on risque dans tous les cas de déséquilibrer l'arbre!*

# Suppression d'un élément : Algorithme (suite)

- Phase 2 : remonter la branche qui relie x à la racine
  - Si tous les arbres enracinés sur cette branche sont des arbres AVL => rien à faire
  - Sinon :
    - Noter a le premier sommet sur la branche (en partant de la feuille) où la condition AVL n'est pas respectée
    - Supposer que la suppression se soit produite dans le sous-arbre droit de a
    - 2 cas à envisager :

# Suppression d'un élément : Algorithme (suite)

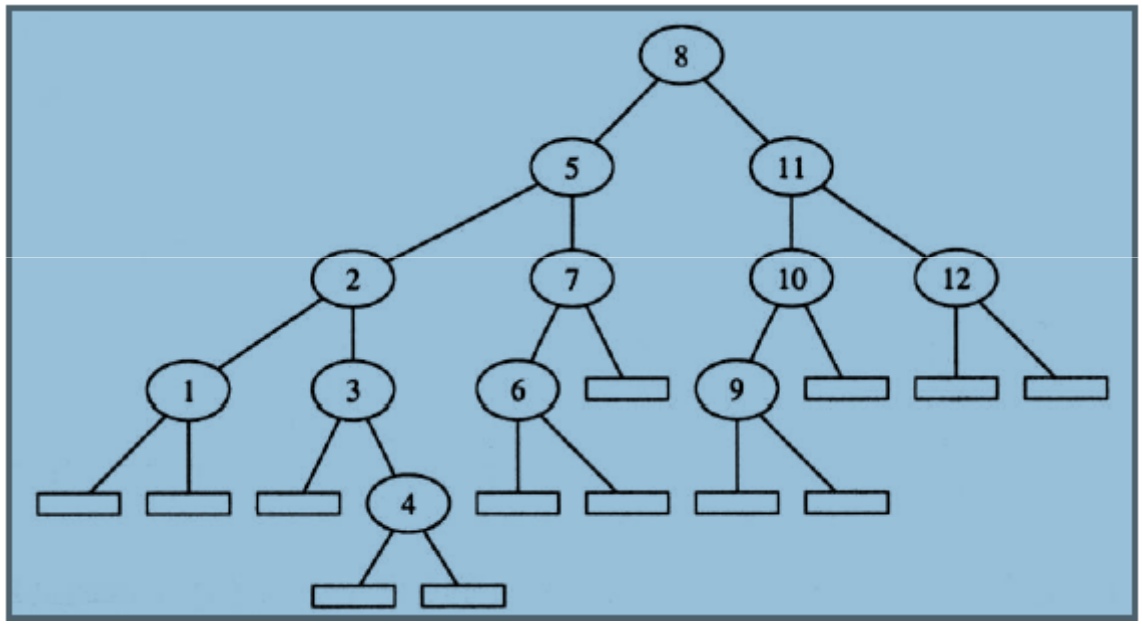
- Cas 1 : le déséquilibre est produit par le fils gauche de b




- Une rotation droite en a rétablit la condition AVL mais le nouveau sous-arbre en b n'a pas nécessairement la même hauteur que l'arbre original enraciné en a => continuer le procédé en remontant vers la racine

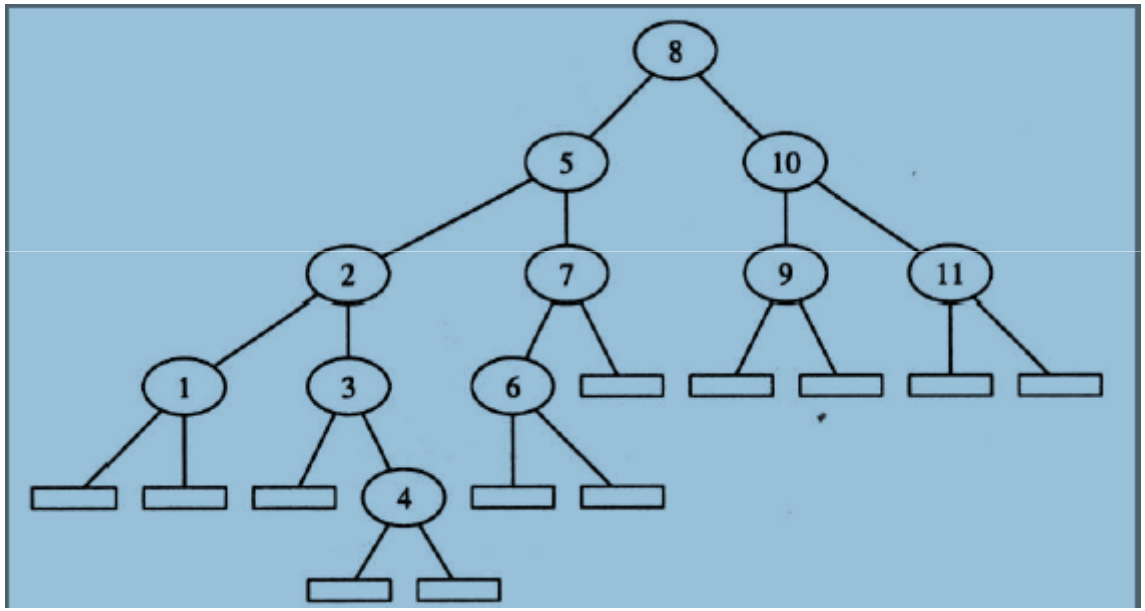
# Exemple

- Suppression de 12 dans l'arbre suivant



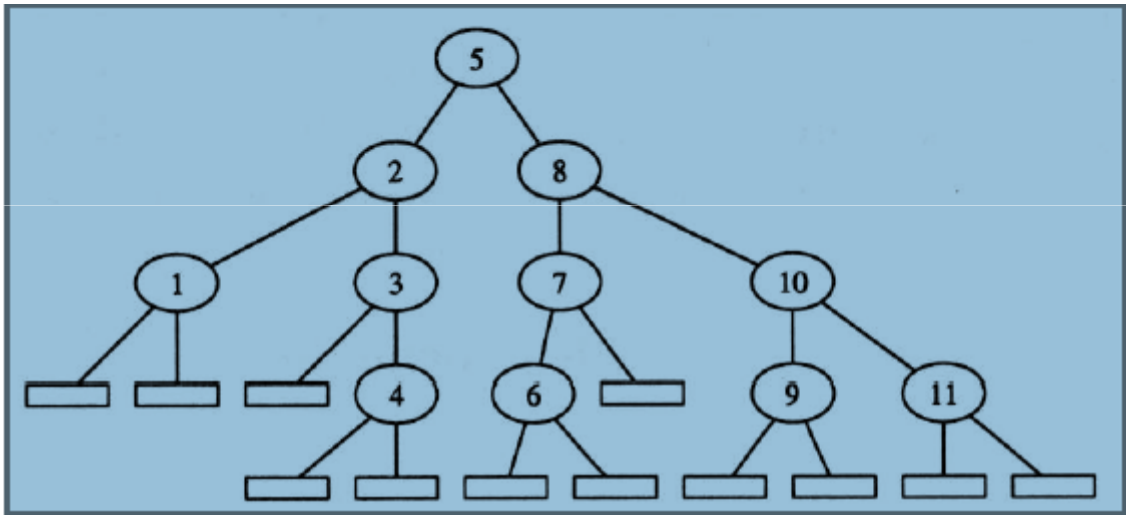

# Exemple (suite)

- Rotation droite en 11 ne suffit pas pour rééquilibrer arbre



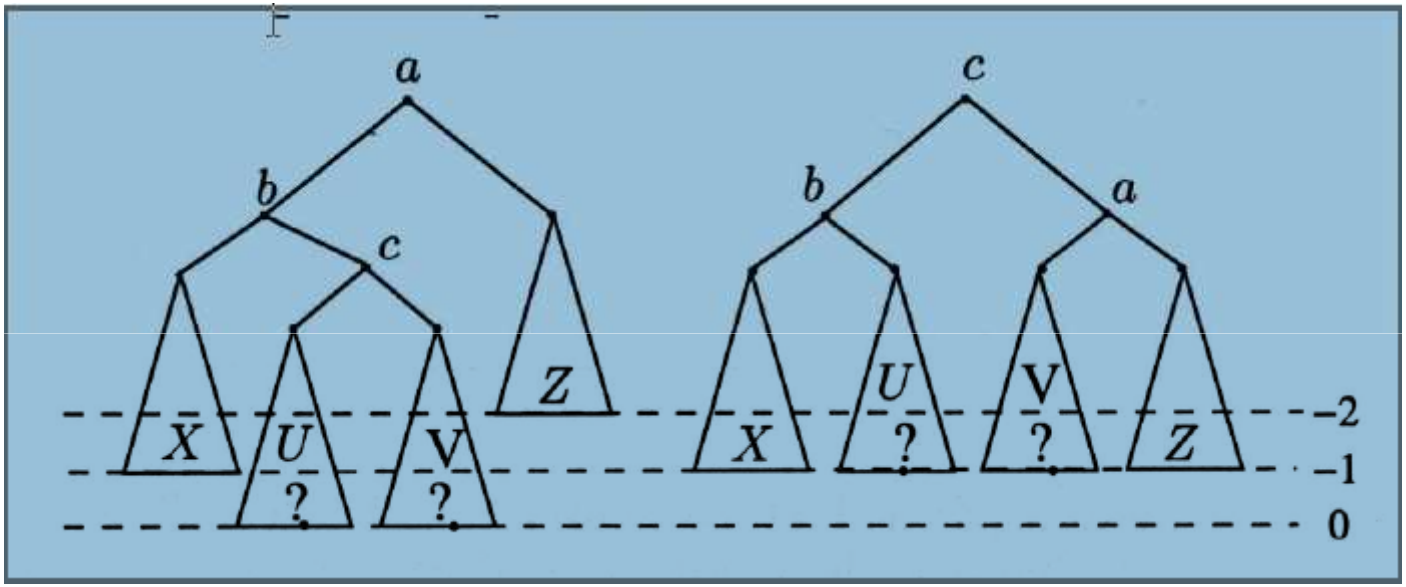

# Exemple (suite)

- Il faut continuer et faire une rotation droite à la racine




# Suppression d'un élément : Algorithme (suite)

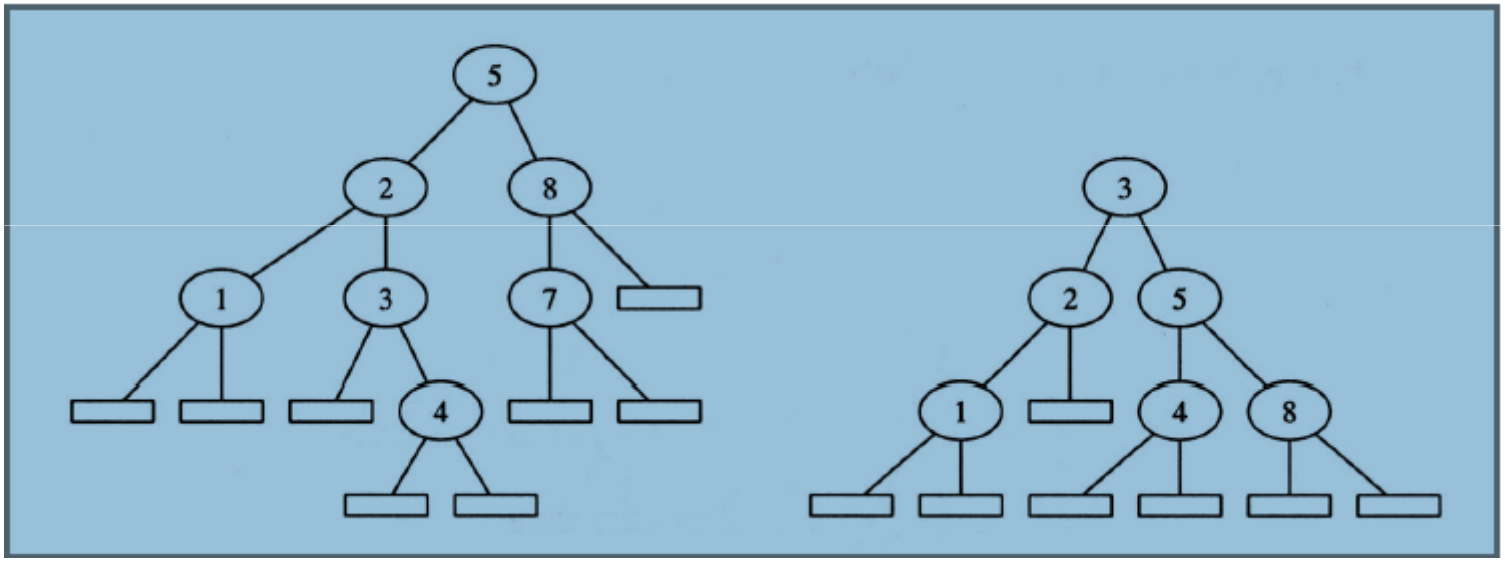
- Cas 2 : le déséquilibre est produit par le fils droit de b




  - Une rotation gauche-droite en a rétablit la condition AVL mais le nouveau sous-arbre en c est de hauteur inférieure de 1 à la hauteur de l'arbre original => continuer le procédé en remontant vers la racine

# Exemple

- Suppression de 7 dans l'arbre ci-dessous conduit à faire une rotation gauche-droite à la racine




# Suppression d'un élément

- Complexité
  - La suppression d'un élément dans un arbre AVL ayant n sommets se réalise en temps O(log(n)) mais peut exiger un nombre non constant de rotations.