Arbres AVL

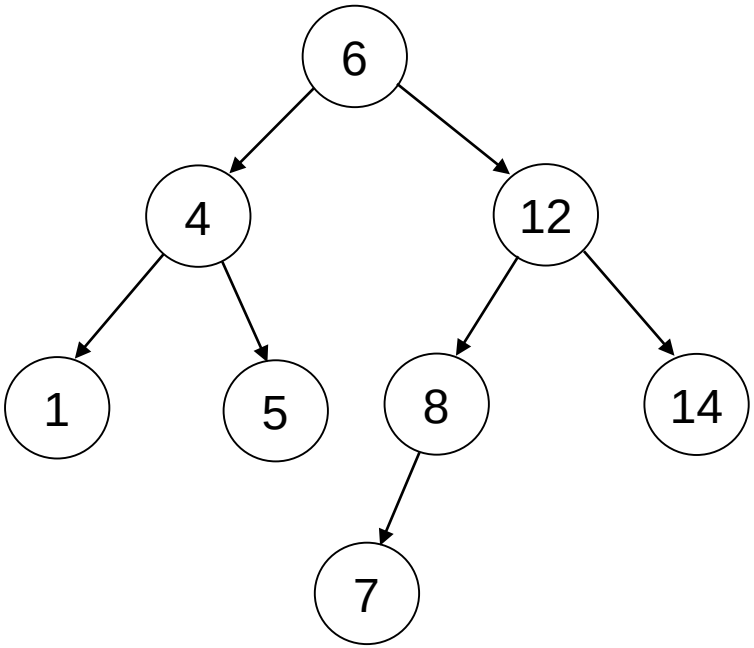
Arbres binaires de recherche équilibrés

Arbre AVL, nommé d'après les initiales de ses inventeurs: Adelson-Velskii and Landis

Arbres binaires de recherche (ABR)

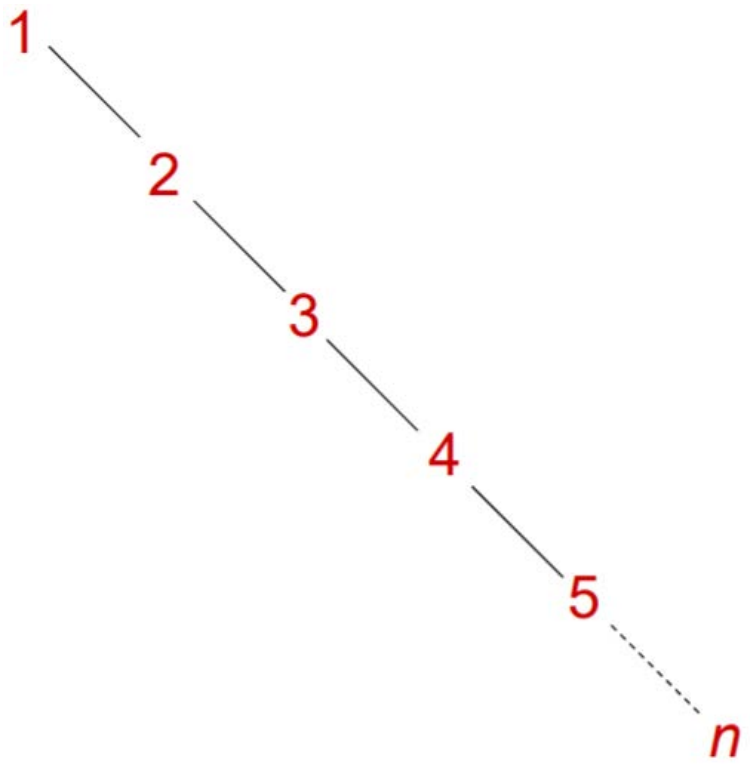
- Arbre ordonné :
  - Tout nœud situé dans un sous-arbre gauche est plus petit que la racine
  - Tout nœud situé dans un sous-arbre droit est plus grand que la racine
- Conçus pour accélérer les recherches d'un élément dans l'arbre
- Objectif dichotomie : on ne parcourt que la moitié de l'arbre

Exemple



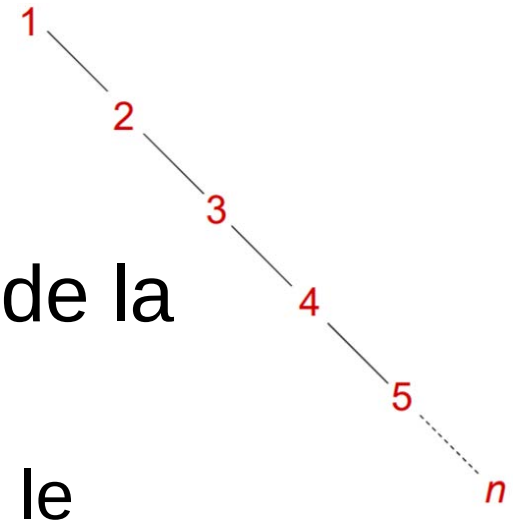
Si on cherche 9, on ne recherche que dans le sous-arbre droit

Problème



Ceci est aussi un ABR…

Problème

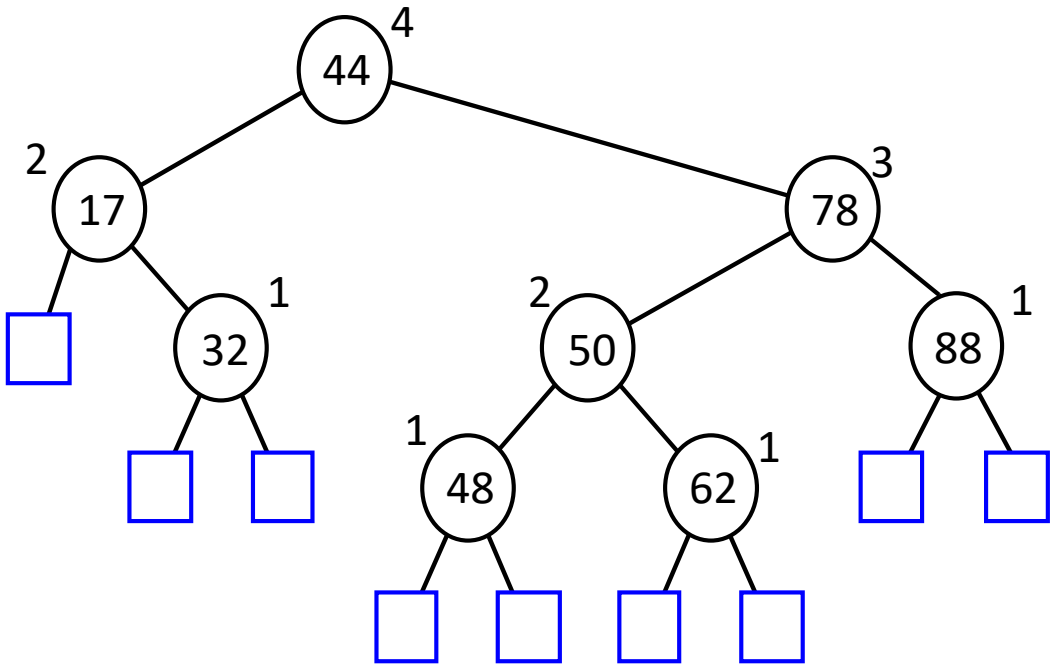


- On ne gagne rien au niveau de la recherche
  - on est obligé de chercher dans le sous-arbre droit
  - coût de l'ordre de n
- Solution :
  - Obliger l'arbre à être relativement équilibré
  - Hauteur du sous-arbre gauche proche de la hauteur du sous-arbre droit

Arbres AVL

- Arbres binaires de recherche équilibrés
- Principe : pour chaque nœud, les hauteurs du sous-arbre gauche et du sous-arbre droit différent au plus de 1
- Modèle proposé par G.M. Adelson-Velsky et E.M. Landis
- Notion de facteur d'équilibre d'un nœud
  - Différence entre les hauteurs des sous-arbre gauche et du sous-arbre droit
  - Un arbre est AVL si tous les nœuds ont un facteur d’équilibre de -1, 0 ou 1

Exemple d'arbre AVL



les hauteurs sont indiquées près des nœuds

Facteur d'équilibre (ou facteur de balancement) :
Un arbre est AVL si

hauteur(sad) – hauteur(sag) $\in$ {-1, 0, 1}

Insertion dans un arbre AVL

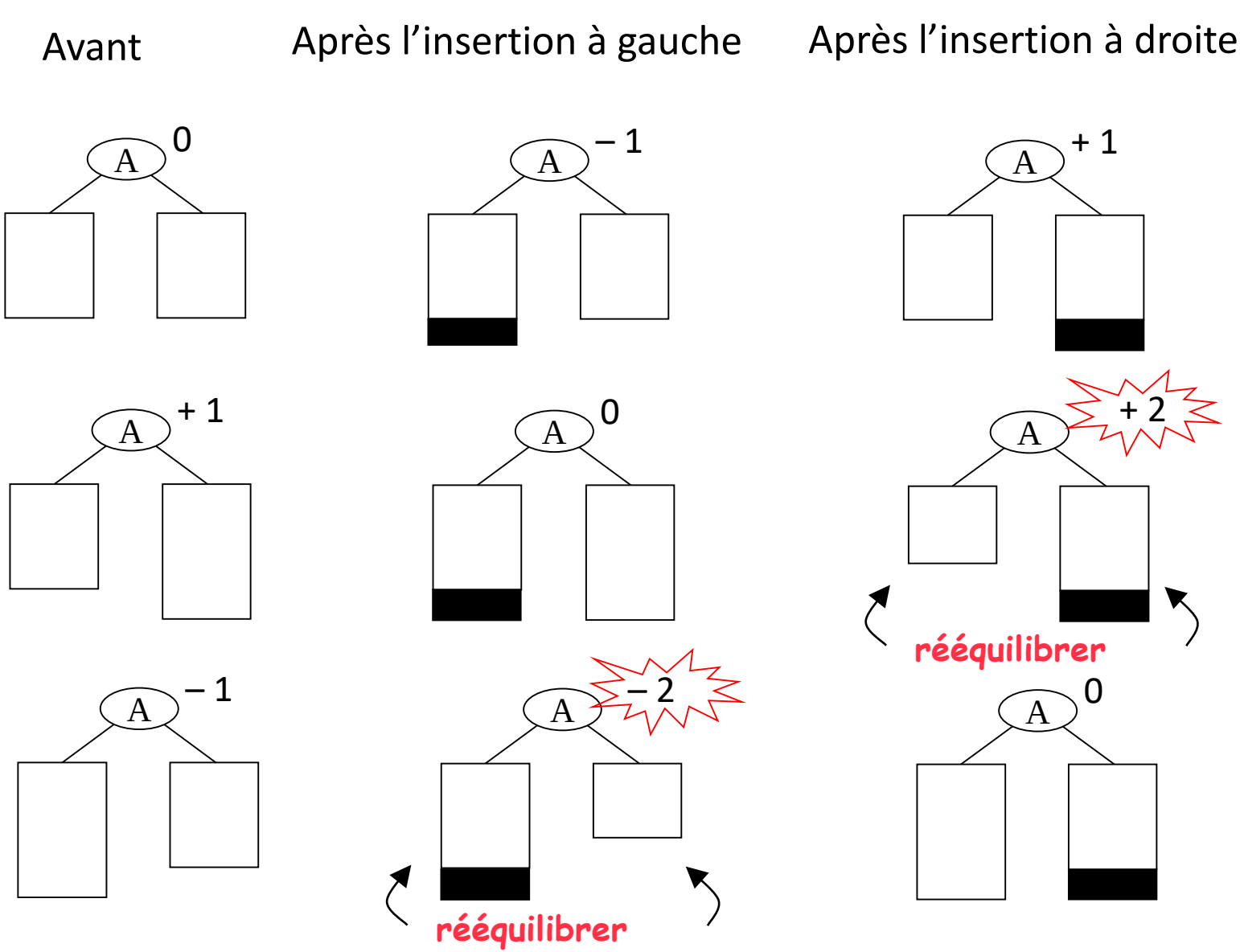
- Un arbre de recherche binaire A est équilibré si, pour chaque nœud *v*, on a:

  $$|\text{hauteur(s.-a. d.)} - \text{hauteur(s.-a. g.)}| \leq 1$$

- L'insertion d'un nœud dans un arbre AVL implique une expansion de l'arbre, ce qui peut changer les hauteurs de quelques-uns des nœuds.

- Si une insertion fait que A devient déséquilibré, nous devons faire une restructuration.

Insertion dans un arbre AVL

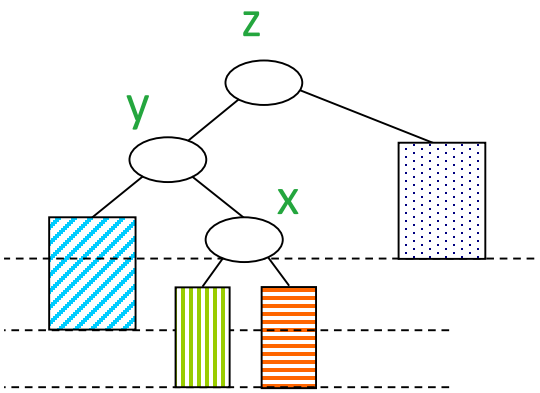


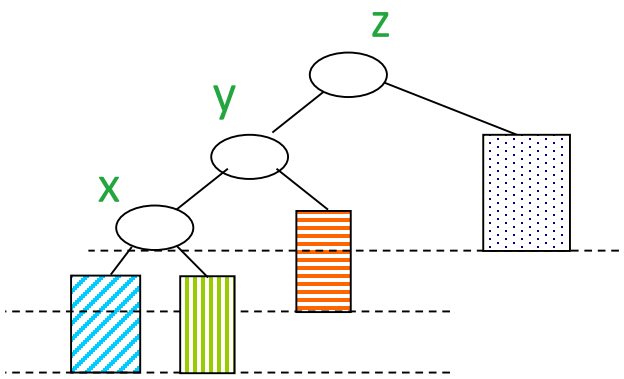
Rééquilibrage après insertion

- Nous allons identifier
  - 3 nœuds qui forment un triplet grand parent, parent, et enfant
  - les 4 sous-arbres attachés

  nous réarrangerons ces éléments pour créer un nouvel arbre équilibré
- Étape 1: on remonte vers la racine de l'arbre à partir du nœud w qui vient d'être inséré, jusqu'à trouver le premier nœud déséquilibré z. Soit y l'enfant de z ayant la plus grande hauteur, et x l'enfant de y ayant la plus grande hauteur.

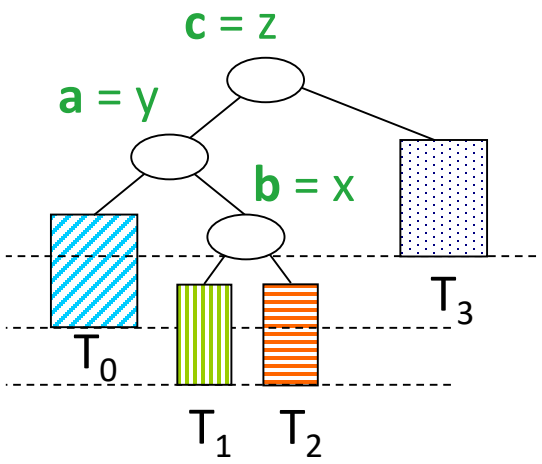
  (x, y, z sont ancêtres du nœud inséré w).

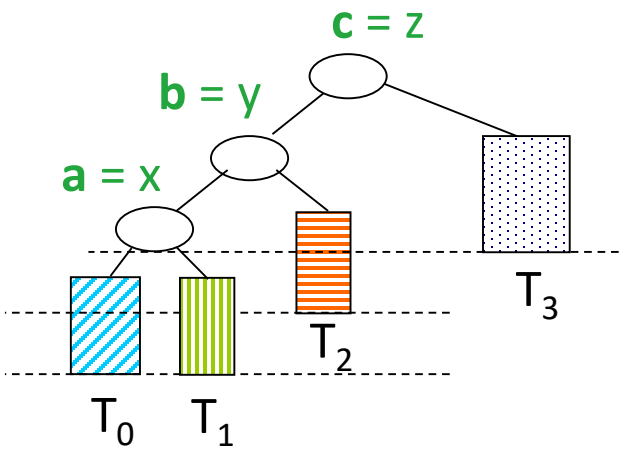




Rééquilibrage après insertion

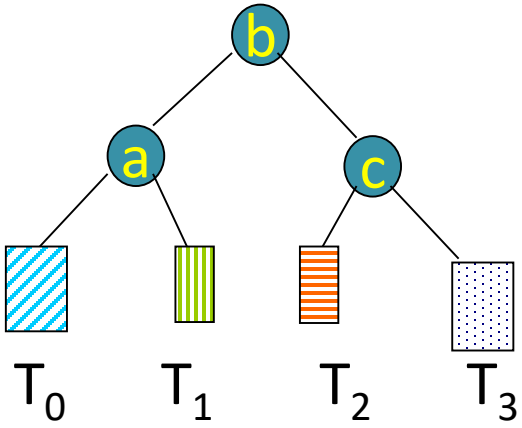
- Étape 2 : les nœuds x, y et z ont 4 sous-arbres connectés à eux. Soient T_0, T_1, T_2, T_3 ces arbres nommés de gauche à droite
- Étape 3 : Renommer les nœuds x, y, z avec a, b, c selon le parcours infixé. Par exemple, si y, x, z est l'ordre des ces nœuds dans le parcours infixé, alors soit y est a, x est b et z est c



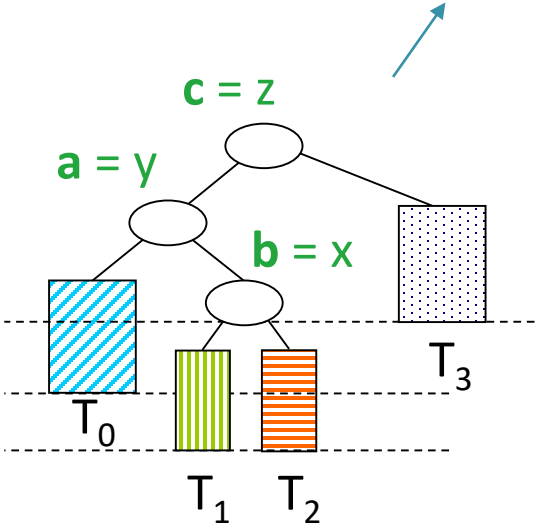


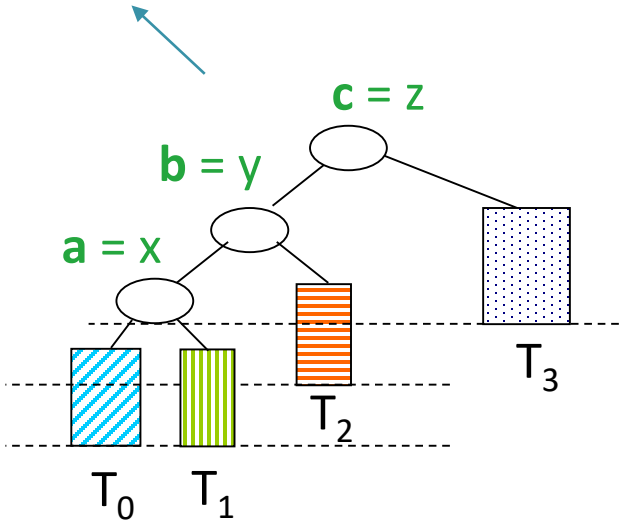
Rééquilibrage après insertion

- Étape 4 : remplacer l'arbre de racine z par l'arbre suivant :

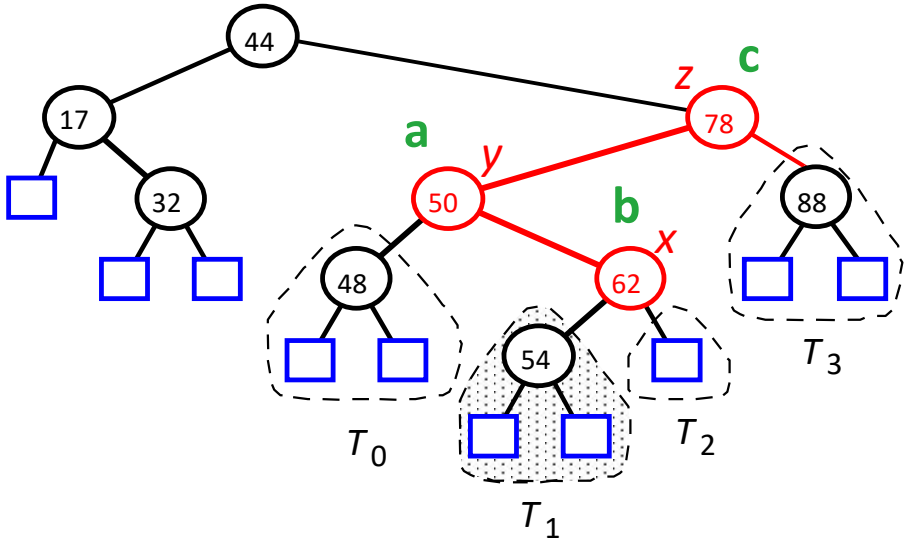


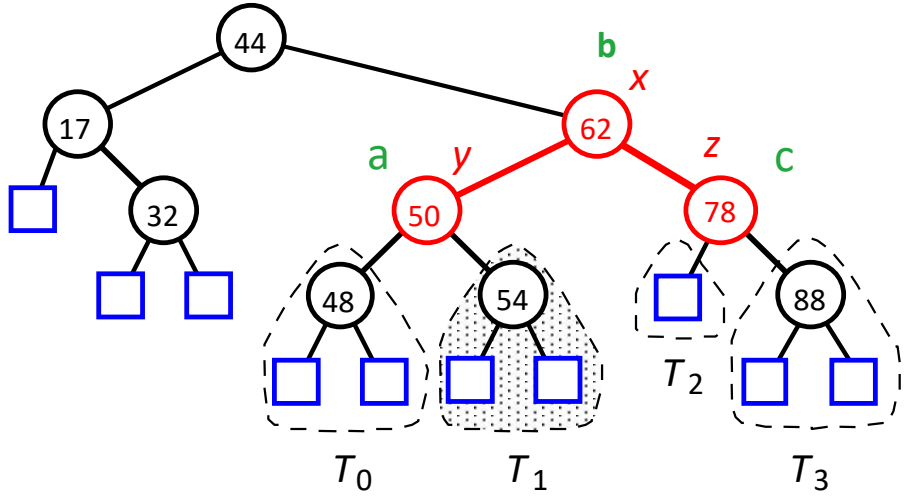
équilibre rétabli !





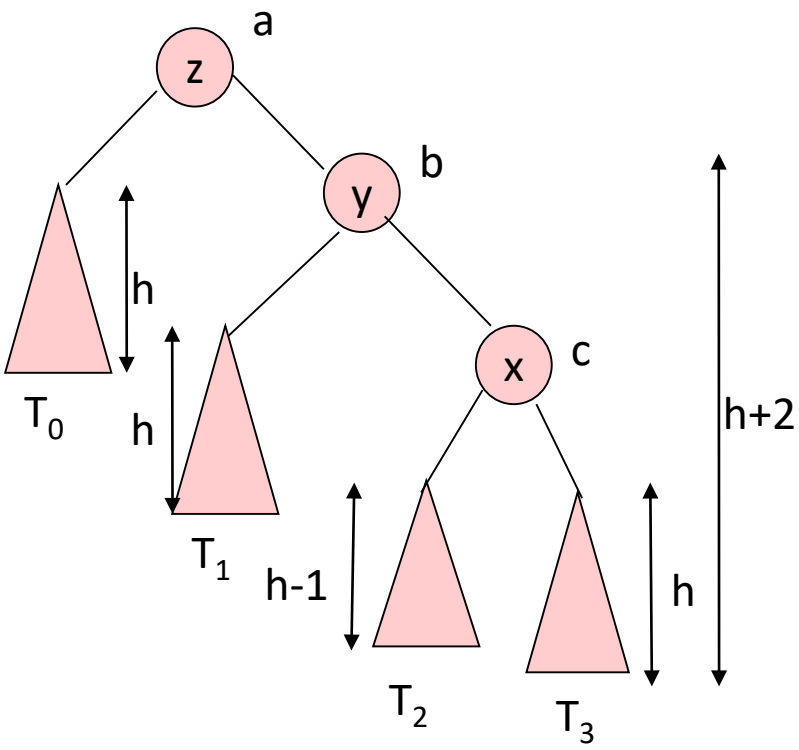
Exemple

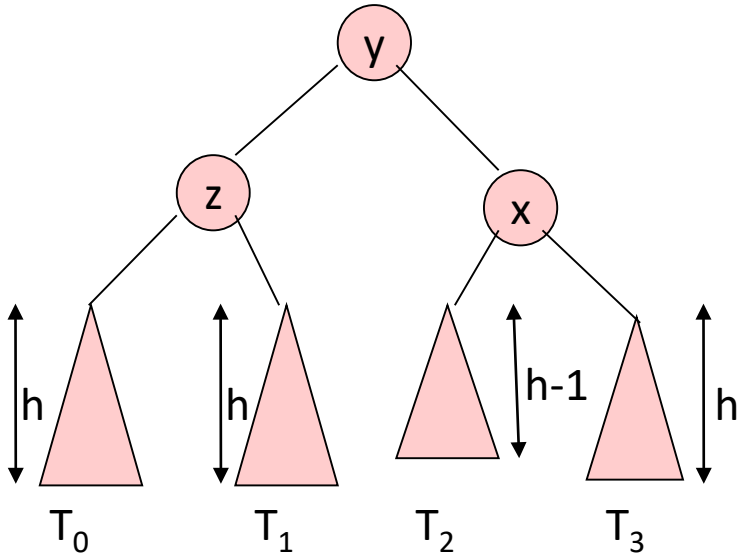

44
17
32
z
c
78
a
y
50
b
x
62
48
54
88
T_0
T_1
T_2
T_3


44
17
32
b
x
62
a
y
50
z
c
78
48
54
88
T_0
T_1
T_2
T_3

AVL- rééquilibrage: rotation gauche

L'arbre de racine z penche à droite et son sous arbre droit de racine y penche à droite ; fe(z)=2 et fe(y)=1 => Rotation gauche de z

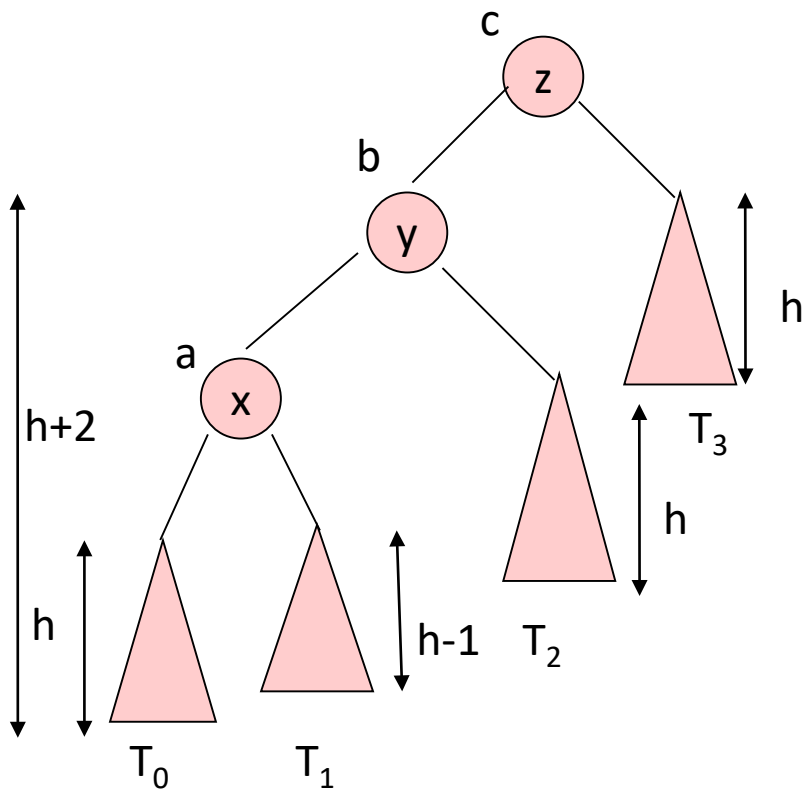


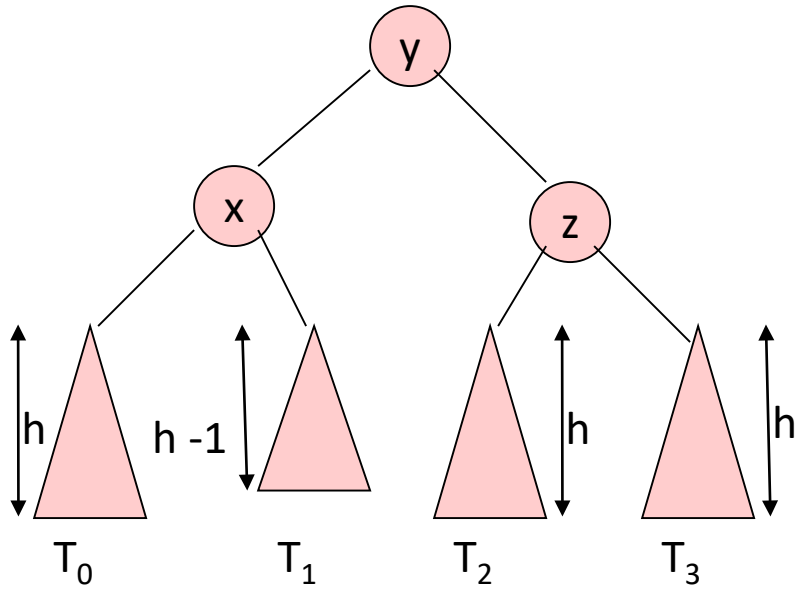


Parcours infixé : T_0 z T_1 y T_2 x T_3

AVL- rééquilibrage: rotation droite

L'arbre de racine z penche à gauche et son sous arbre gauche de racine y penche à gauche; fe(z)=-2 et fe(y)=-1 => Rotation droite de z

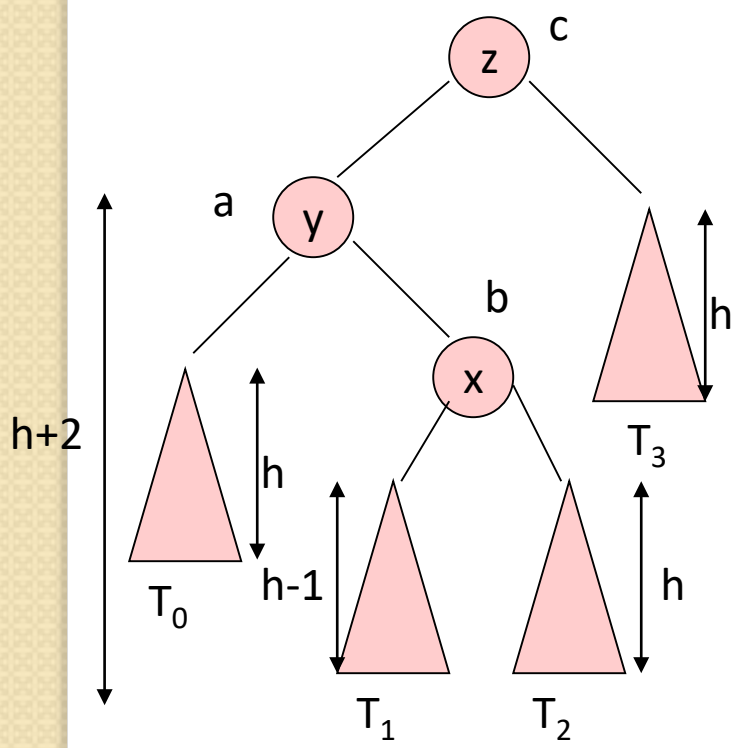


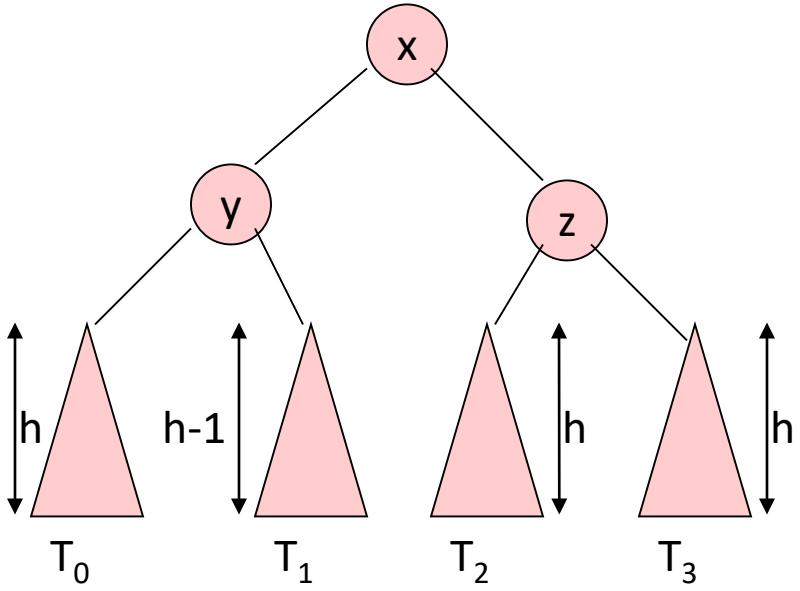


Parcours infixé : T_0 x T_1 y T_2 z T_3

AVL- rééquilibrage: rotation double gauche-droite

L'arbre de racine z penche à gauche et son sous arbre gauche de racine y penche à droite; fe(z)=-2 et fe(y)=1 => Rotation gauche-droite
= rotation gauche de y suivie d'une rotation droite de z

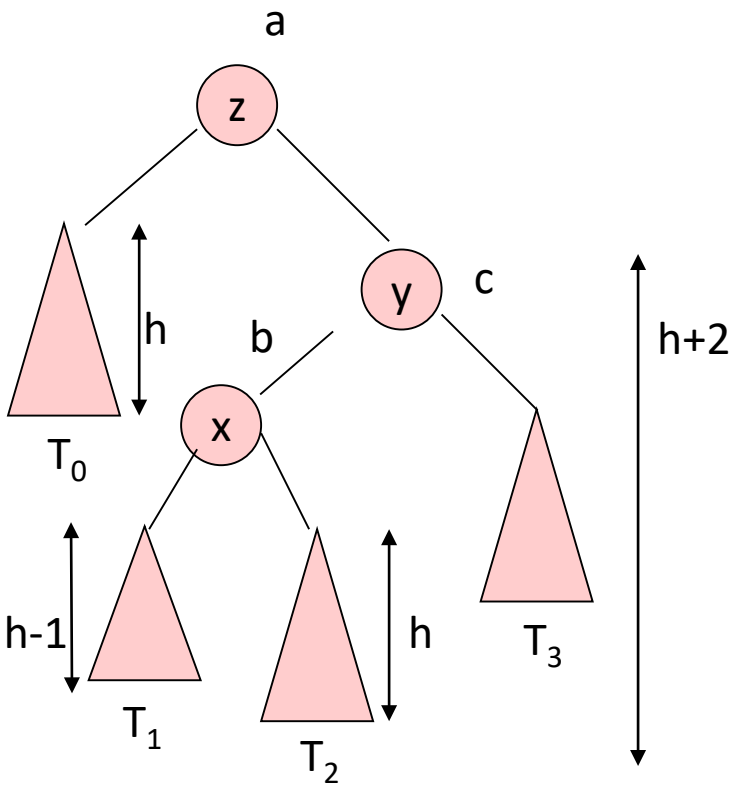


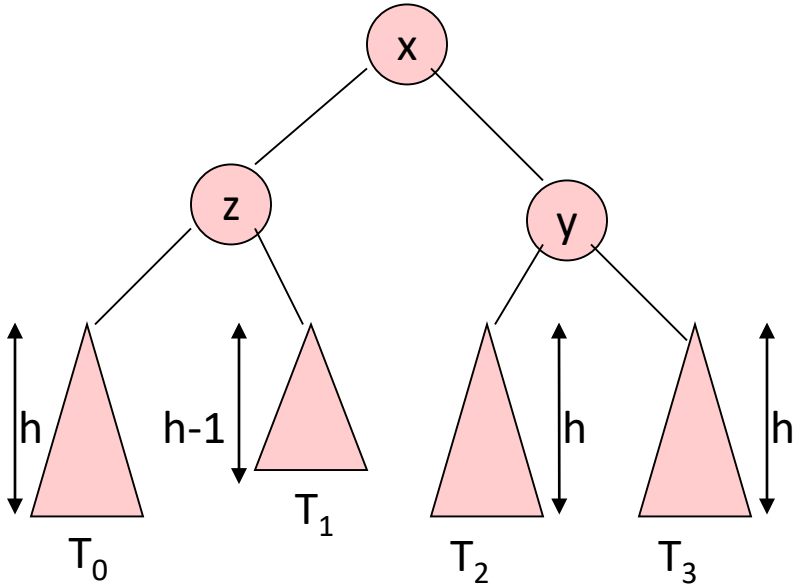


Parcours infixe: T_0 y T_1 x T_2 z T_3

AVL- rééquilibrage: rotation double droite-gauche

L'arbre de racine z penche à droite et son sous arbre gauche de racine y penche à gauche ; fe(z)=2 et fe(y)=-1 => Rotation droite-gauche
= rotation droite de y suivie d'une rotation gauche de z





Parcours infixe: T_0 z T_1 x T_2 y T_3

Rééquilibrage d'un AVL après insertion

- En partant du nœud inséré w, considérons le premier sous-arbre S de l'AVL déséquilibré suite à l'insertion de ce nouveau nœud.
- Le déséquilibre de S est causé par l'insertion de w qui a augmenté de 1 la hauteur d'un sous-arbre de S. Or la hauteur de S est elle-même incrémentée du coup, ce qui peut entrainer le déséquilibre global de l'AVL.
- Le rééquilibrage de S rétablit son équilibre local et réduit sa taille de 1 et du coup S retrouve sa hauteur initiale (avant insertion de w). Ainsi on rétablit l'équilibre global de l'AVL .
- Un seul rééquilibrage est donc suffisant après l'insertion d'un nouveau nœud dans un arbre AVL.

Implémentation

- On ajoute un attribut à l'arbre
  - sa profondeur

  ou
  - son facteur d'équilibre
- A mettre à jour à chaque modification (ajout ou suppression)

Insertion dans un arbre AVL

Le principe de l'insertion dans un arbre AVL est le suivant :

- insérer le nouveau nœud au bon endroit
- au fur et à mesure de la remontée dans l'arbre (du nœud père du nœud inséré, à la racine), rééquilibrer l'arbre en effectuant les rotations appropriées

Choix des rotations

- Notations
  - Soient N le noeud courant, Ng son fils gauche et Nd sont fils droit.
  - Soient Ngg le fils gauche de Ng et Ngd le fils droit de Ng
  - Soient Ndg le fils gauche de Nd et Ndd le fils droit de Nd
  - Soit h(x) la hauteur de l'arbre de racine le nœud x.
- Algorithme
  - Si $|h(Ng)-h(Nd)| \leq 1$, ne rien faire
  - Sinon
    - Si h(Ng)-h(Nd) = 2
      - Si h(Ngg) > h(Ngd) alors rd(N)
      - Sinon rg(Ng) puis rd(N)
    - Sinon // (h(Ng)-h(Nd) = -2)
      - Si h(Ndd) > h(Ndg) Alors rg(N)
      - Sinon rd(Nd) puis rg(N)
    - Fsi
  - Fsi

Insertion AVL

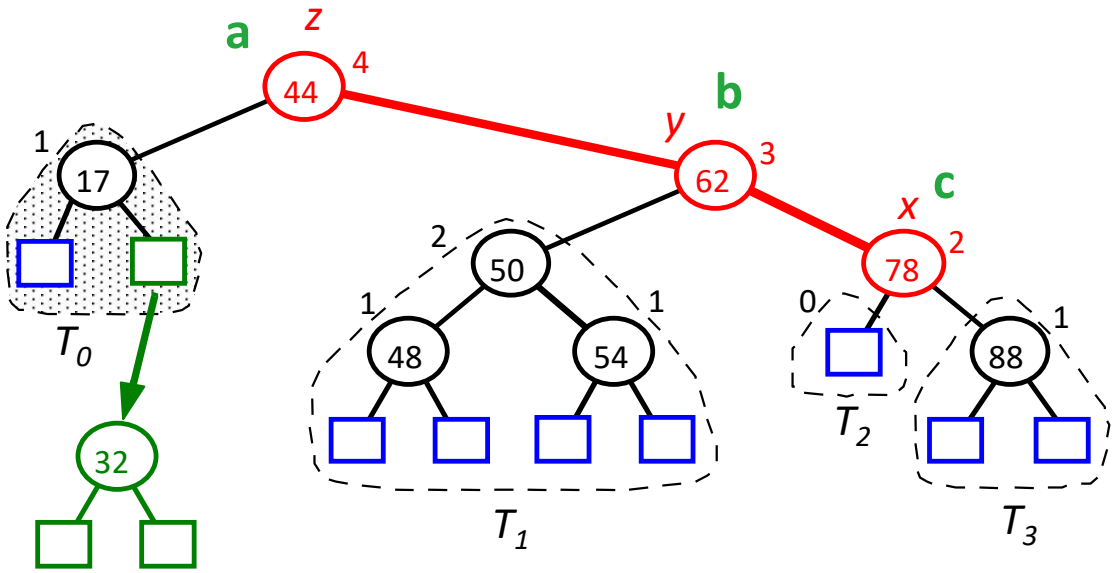
```
action insertionAVL(consulté v : entier, élaboré a : Arbre)
algorithme
si a = NIL
alors créer(a) ; a↑.el ← v ; a↑.gauche ← NIL ; a↑.droit ← NIL
sinon si v < a↑.el
        alors insertionAVL (v, a↑.gauche)
                si déséquilibre(a)
                alors si v < (a↑.gauche)↑.el
                        alors rotationDroite(a)
                        sinon rotationGaucheDroite(a)
                        fsi
                fsi
        sinon insertionAVL (v, a↑.droite)
                 si déséquilibre(a)
                alors si v > (a↑.droite)↑.el
                        alors rotationGauche(a)
                        sinon rotationDroiteGauche(a)
                        fsi
                fsi
        fsi
fsi
```

Suppression dans un AVL

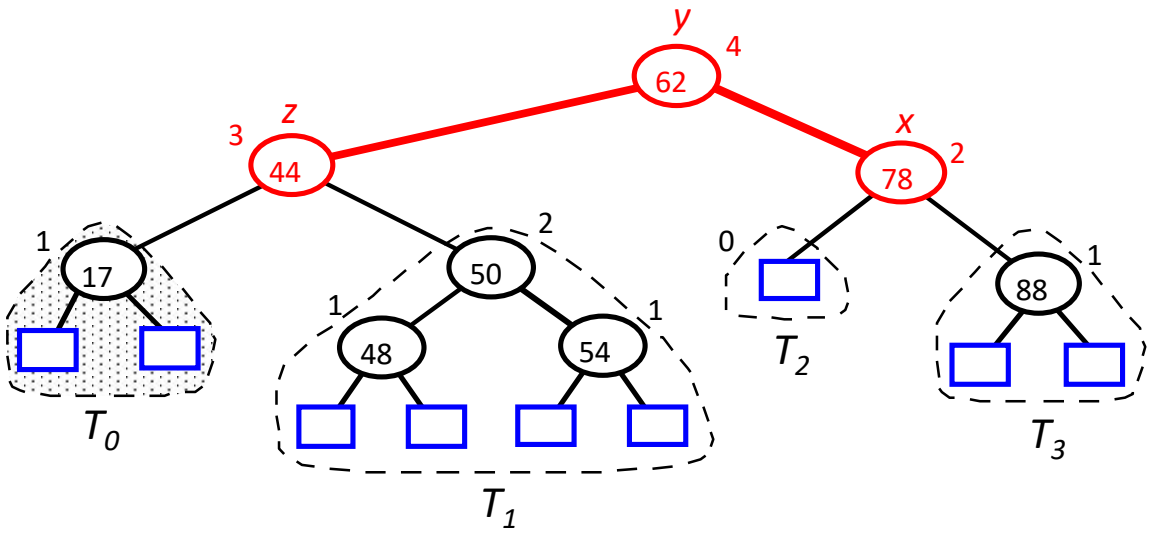
- On constate facilement que la suppression d'un nœud w peut causer un déséquilibre dans un AVL
- Soit z le premier nœud déséquilibré rencontré en remontant l'arbre vers la racine à partir de w (ou du nœud qui s'est substitué à w). Soit y l'enfant de z ayant la plus grande hauteur, et x l'enfant de y ayant la plus grande hauteur. (y et x ne sont pas ancêtres de w)
- On applique la même stratégie de restructuration = restructure(x) pour rééquilibrer le sous-arbre de racine z
- Cette restructuration réduit de 1 la hauteur du sous-arbre initialement enraciné en z et ainsi pourrait déséquilibrer un autre nœud plus haut dans l'arbre. Alors, nous devons continuer à vérifier l'équilibre jusqu'à ce que la racine de l'arbre soit atteinte.
- On peut donc faire jusqu'à h rotations lors d'une suppression, h étant la hauteur de l'arbre.

Suppression

Le choix de x n'est pas unique !!!



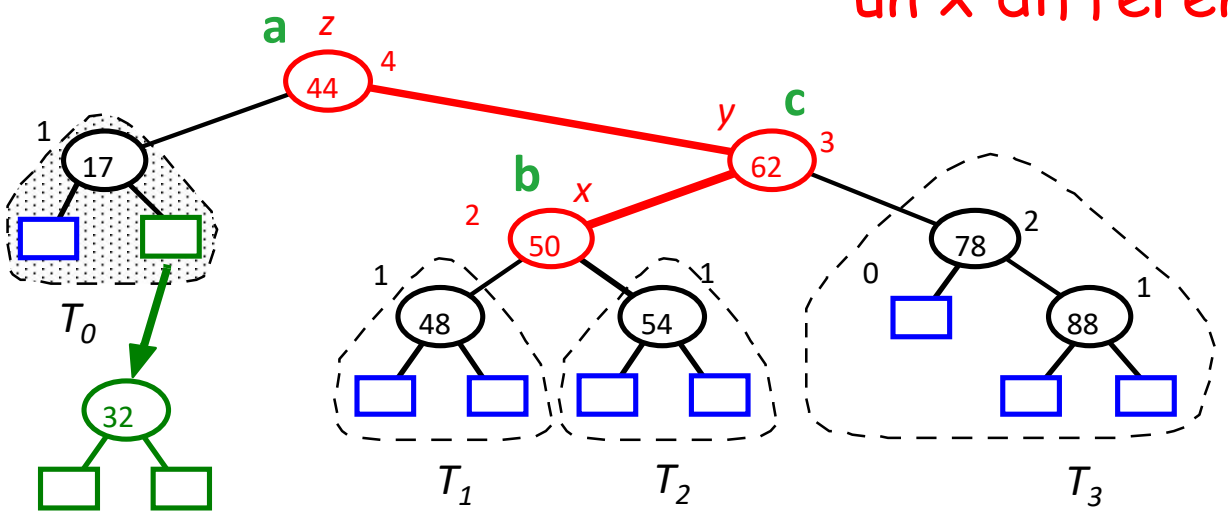
Déséquilibré !



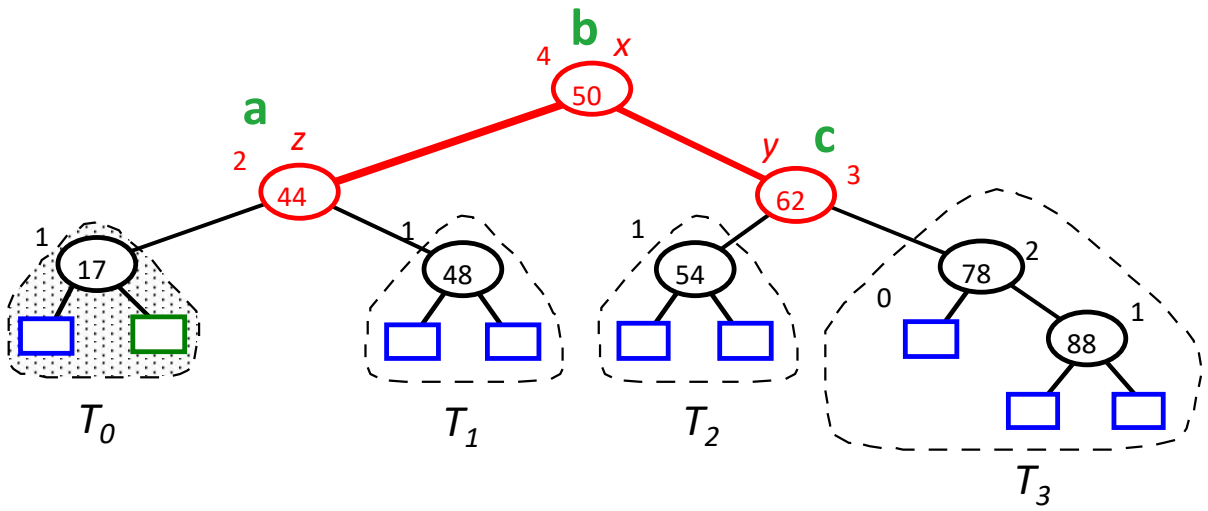
Rééquilibré !

Suppression

Nous pourrions choisir un x différent



Déséquilibré !



Rééquilibré !