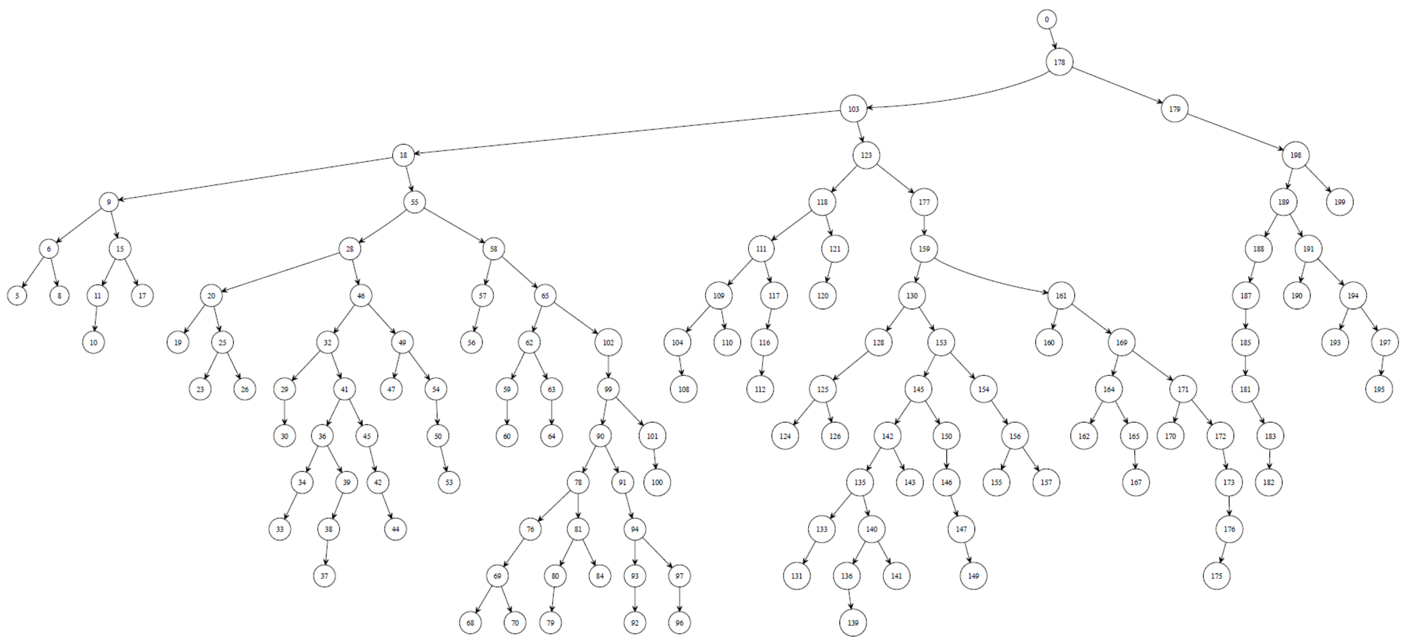
Algorithmique

Arbres binaires de recherche



INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Eric Guérin
INSA de Lyon – Département Informatique
3IF

Introduction

- Arbre binaire avec des propriétés
- Toutes les valeurs du sous-arbre de gauche sont inférieures à la valeur du nœud
- Toutes les valeurs du sous-arbre de droite sont supérieures à la valeur du nœud
- Propriété récursive (s'applique à chaque nœud)

Introduction

- Formellement
  - a est un nœud
  - l(a) le sous-arbre de gauche
  - r(a) le sous-arbre de droite

$$\begin{cases} \forall i \in l(a) & i < a \\ \forall i \in r(a) & i > a \end{cases}$$

- Propriété récursive

Avantages

- Parcours infixe = ordonné
- Pour trouver une valeur
  - Si la valeur est plus petite que le nœud courant on sait qu'elle se trouve dans le sous-arbre de gauche
  - Inversement si elle est supérieure, on sait qu'elle se trouve à droite
- Recherche = un chemin dans l'arbre vers le nœud cible

Recherche

Structure *ABR*
- entier $valeur$
- ABR * $gauche$
- ABR * $droit$

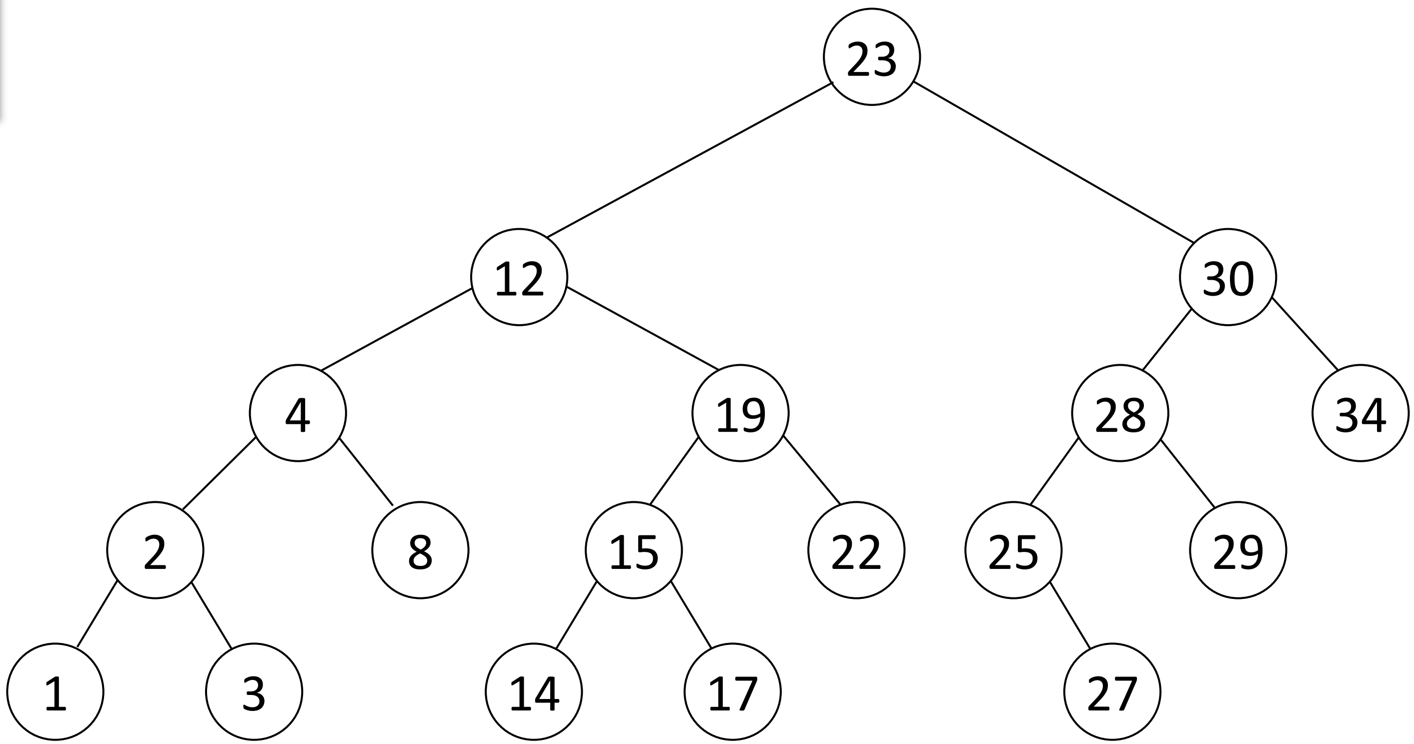
Fonction $\text{RechercherABR}(abr,v)$

Entrée : ABR * abr
Entrée : entier v
Précondition : abr contient un arbre binaire de recherche bien formé (éventuellement vide donc pointeur nul).
Postcondition : Retourne un pointeur vers le nœud trouvé ou nul si la valeur n'a pas été trouvée.

```
si abr = ∅ ou abr → valeur = v alors
    retourne abr
sinon
    si v > abr → valeur alors
        retourne RechercherABR(abr → droit,v)
    sinon
        retourne RechercherABR(abr → gauche,v)
```

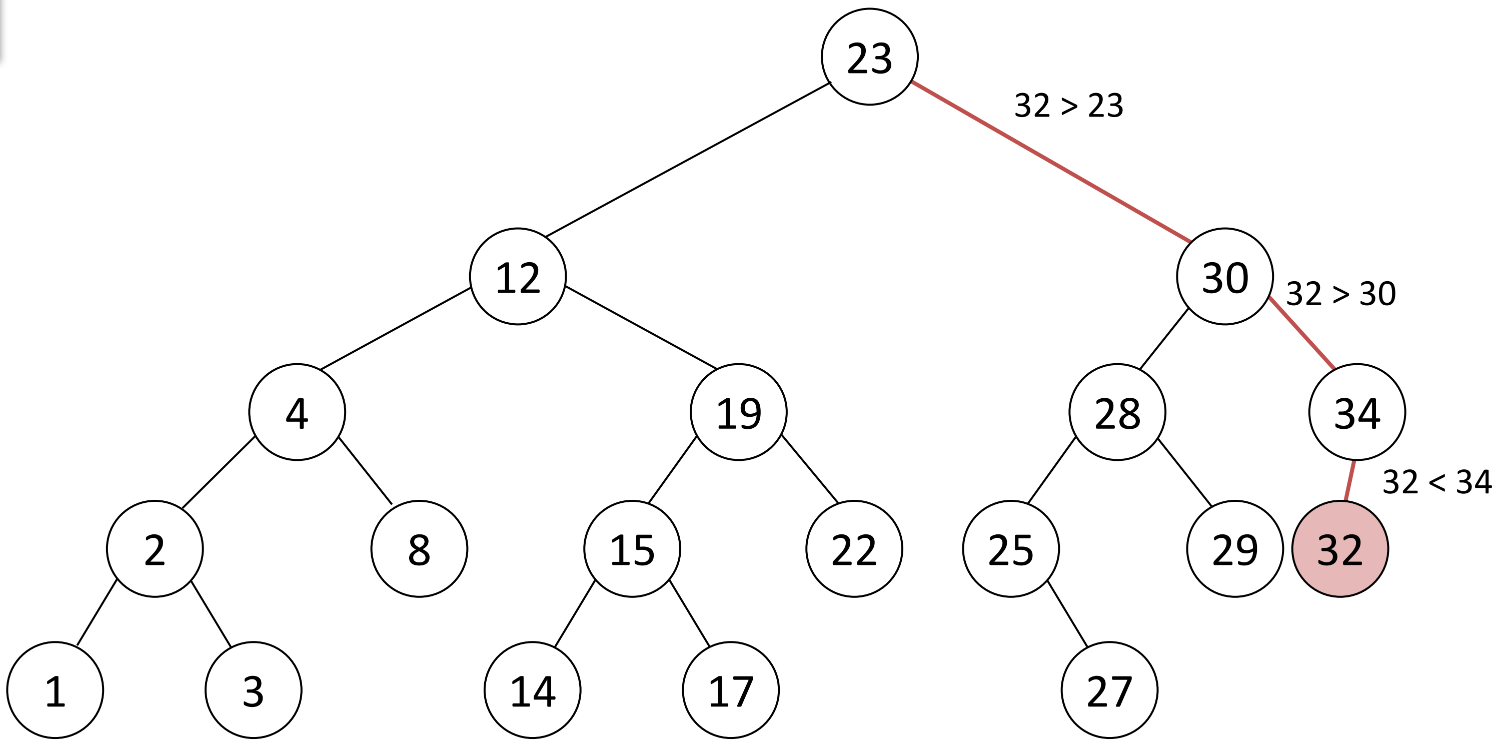
Insertion

- Insertion de 32 ?



Insertion

- Résultat



Insertion

- Insertion très proche de la recherche
- On cherche l'endroit où la valeur devrait se trouver
- Quand on arrive à un nœud nul c'est que l'on a trouvé une place qui convient
- Peut-il y avoir plusieurs places qui conviennent ?
- Exercice : insertion des valeurs 20, 24 et 5 dans l'arbre précédent

Insertion

Procédure InsererABR(abr,v)

Entrée/Sortie : ABR * abr
Entrée : entier v
Précondition : abr contient un arbre binaire de recherche bien formé (éventuellement vide donc pointeur nul).
Postcondition : L'élément v est inséré au bon endroit dans l'arbre binaire de recherche, sauf si cette valeur existe déjà dans l'arbre.
Déclaration : ABR * $nouveau$

si $abr = \varnothing$ **alors**
    $nouveau \leftarrow$ Allouer(*ABR*)
    $nouveau \rightarrow valeur \leftarrow v$
    $nouveau \rightarrow gauche \leftarrow \varnothing$
    $nouveau \rightarrow droit \leftarrow \varnothing$
    $abr \leftarrow nouveau$
sinon
    si $v > abr \rightarrow valeur$ **alors**
        InsererABR($abr \rightarrow droit$,v)
    sinon si $v < abr \rightarrow valeur$ **alors**
        InsererABR($abr \rightarrow gauche$,v)

Insertion

Introduction
Insertion
Suppression
Equilibrage

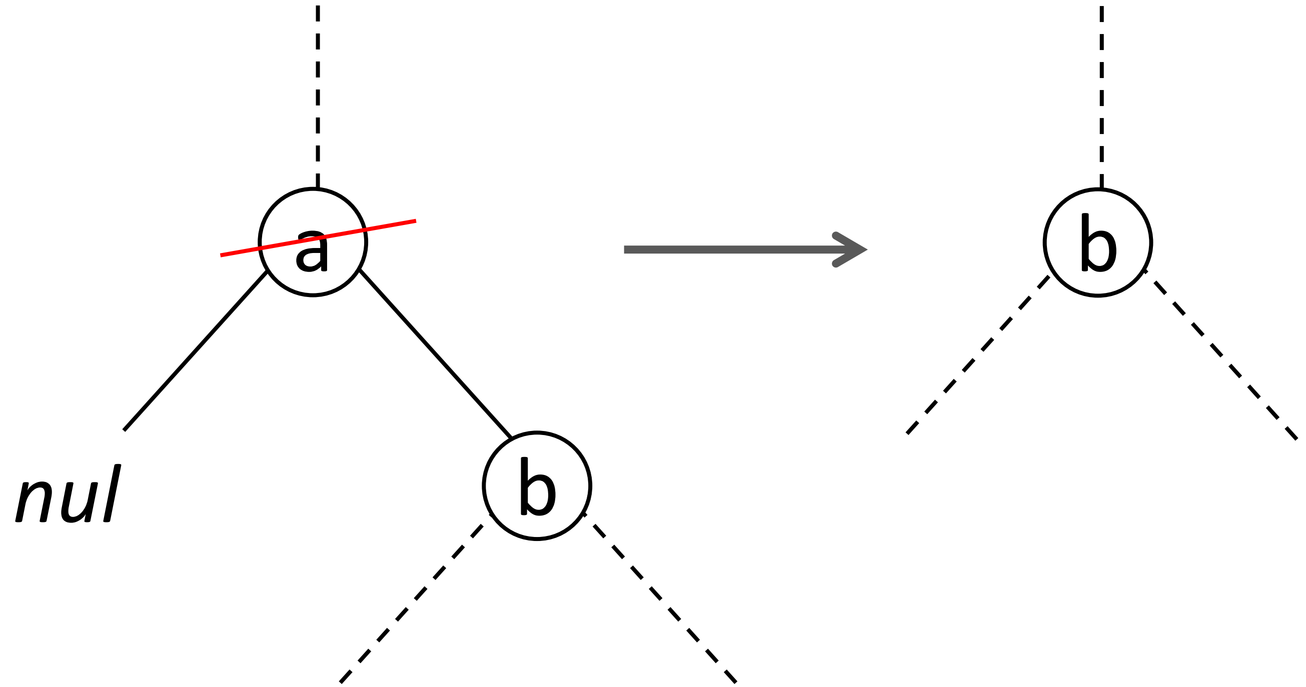
- Exercice : transformer cet algorithme pour qu'il ne soit plus récursif

Suppression

- Algorithme un peu plus complexe
- Il se peut que le nœud à supprimer ait un ou plusieurs fils
- Quatre cas de figure

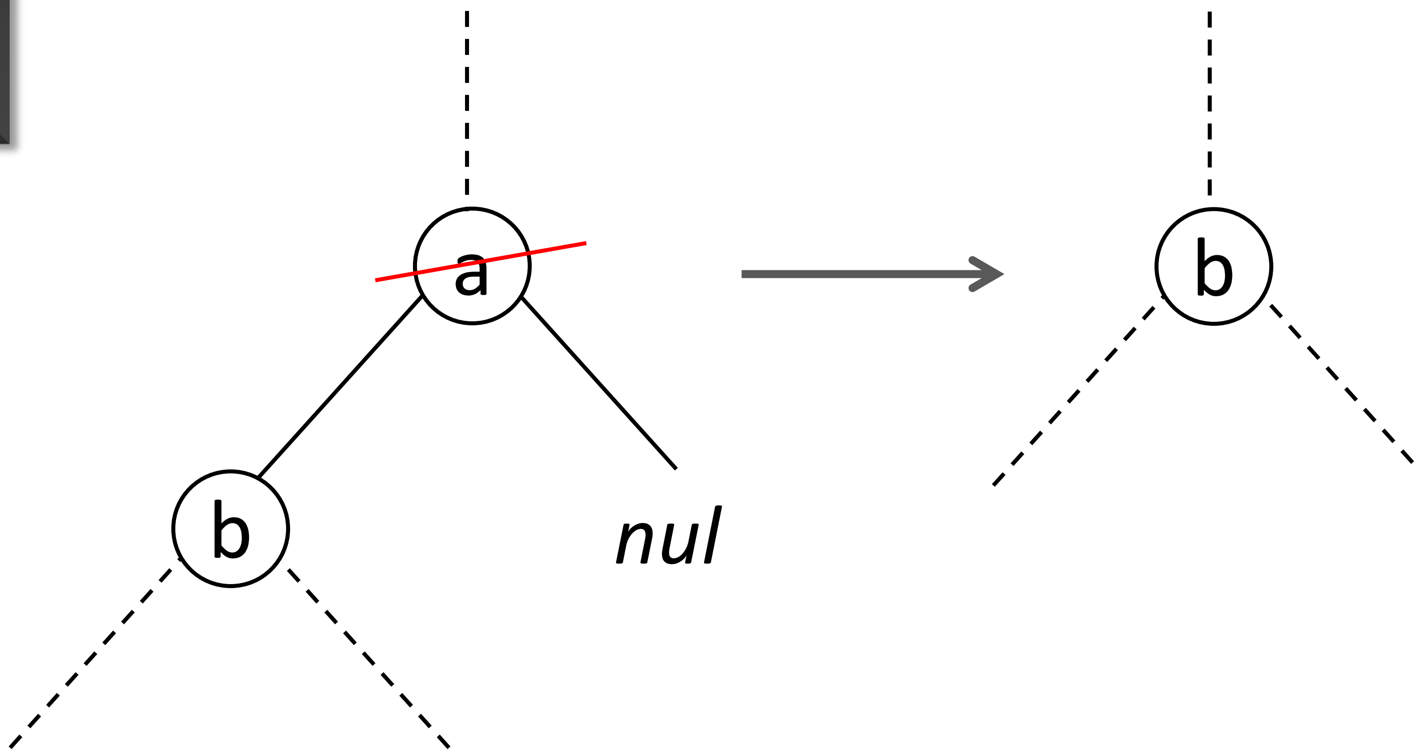
Suppression

- Cas 1 : un seul fils à droite



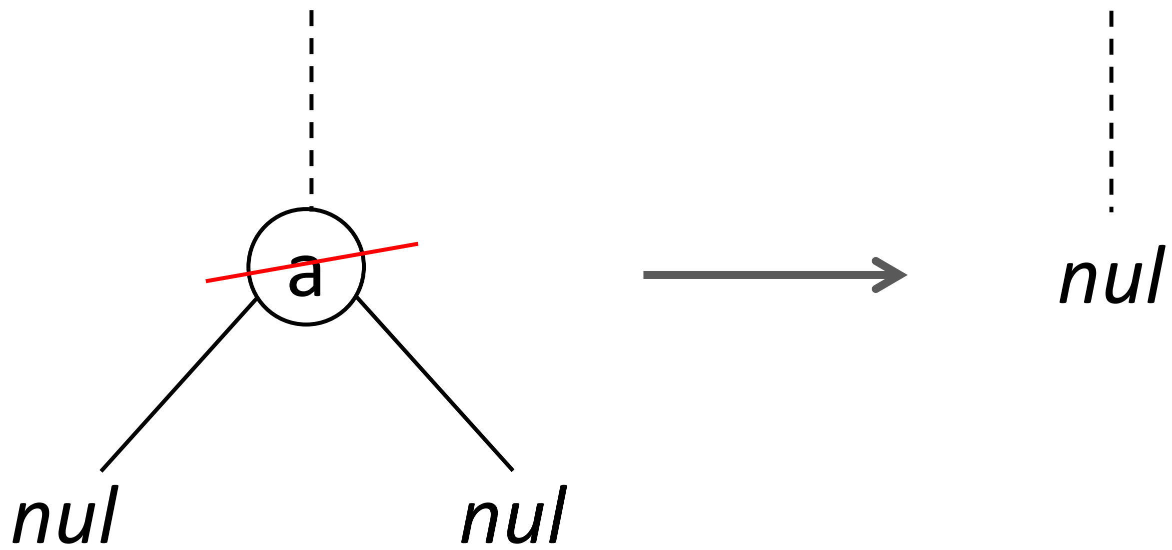
Suppression

- Cas 2 : un seul fils à gauche



Suppression

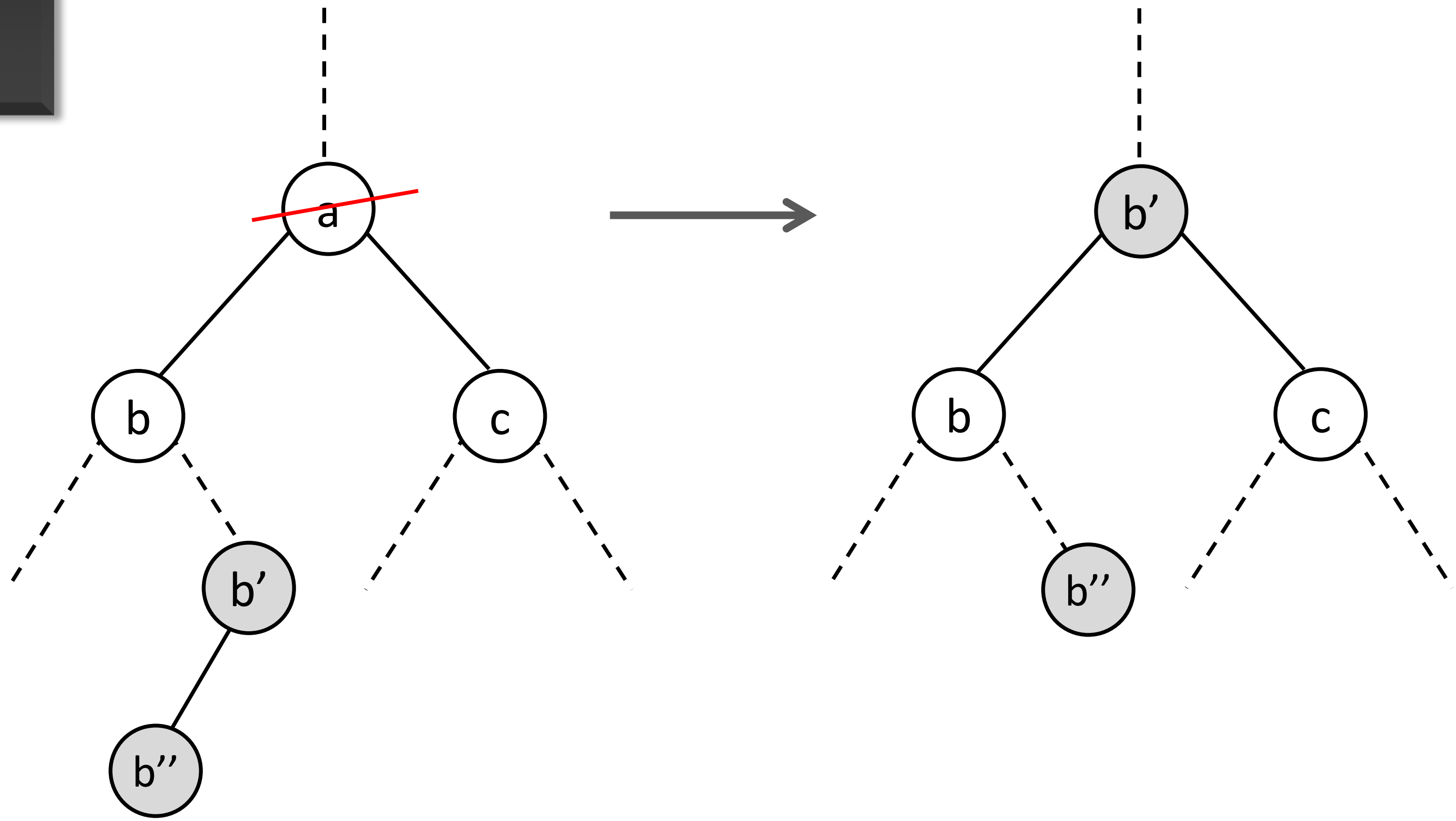
- Cas 3 : pas de fils



Suppression

Introduction
Exemples
La suite

- Cas 4 : deux fils



Suppression

Introduction
Exemples
La suite

- b’ est la valeur inférieure la plus proche de a
- Comment la trouver ?
  - On part du fils gauche de a et on ne fait qu’aller sur la sous-branche droite
  - Si pas de branche droite, on s’arrête
  - Il ne peut pas y avoir de valeur plus grande dans ce sous-arbre
- Si b’ a un fils gauche b’’ (il ne peut pas avoir de fils droit), alors on le met à la place de b’
- On place b’ à la place de a

Suppression

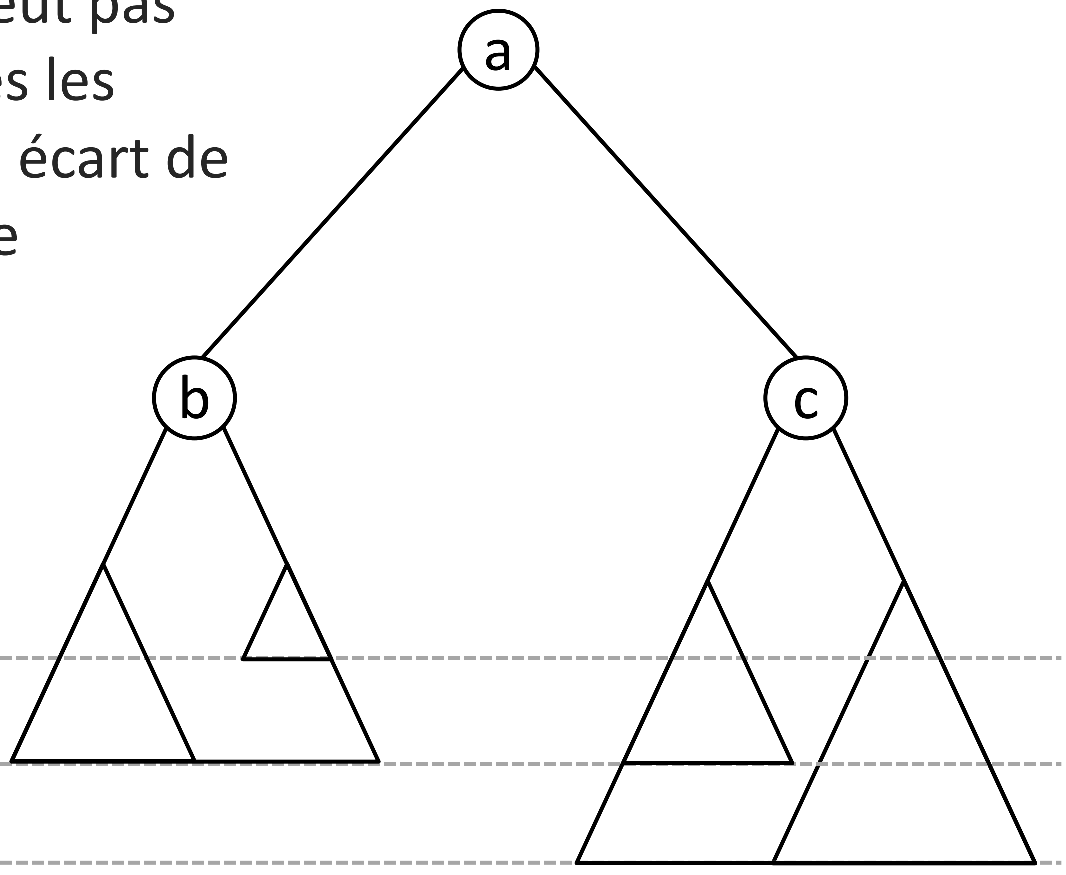
- On peut faire exactement la même chose mais de l'autre côté
- Dans ce cas-là, on remplace a par la valeur la plus proche mais supérieure, donc le fils le plus à gauche du sous-arbre de droite
- Autre variante : choisir le côté qui équilibre le mieux l'arbre

Equilibrage

- L'insertion dans un ABR peut rendre l'arbre déséquilibré
- Définition
  - Un arbre est équilibré (H-équilibré) si l'écart **maximum** entre la profondeur des sous-arbres gauche et droit est de 1, **pour tous les nœuds**
- Un arbre complet est H-équilibré (cas particulier)

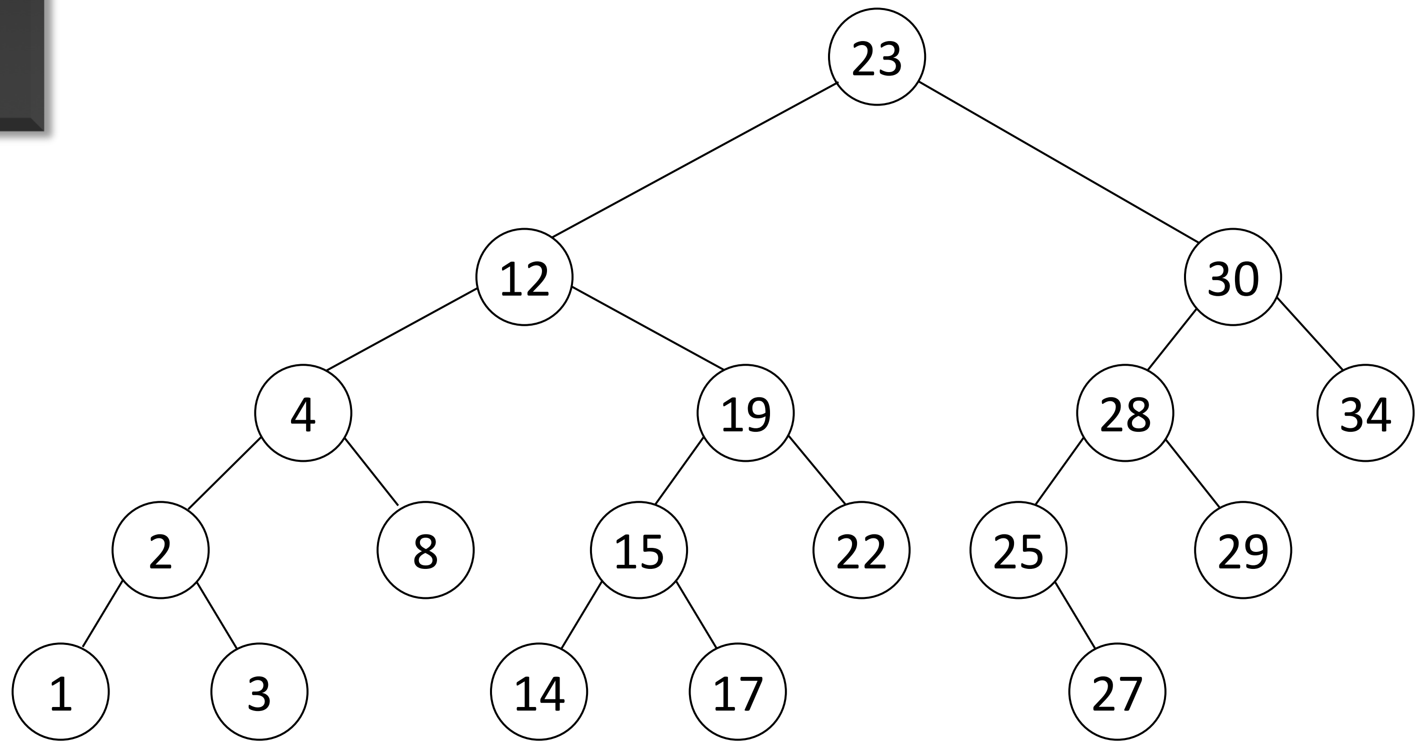
Equilibrage

- Equilibre ne veut pas dire que toutes les feuilles ont un écart de maximum 1 de profondeur !



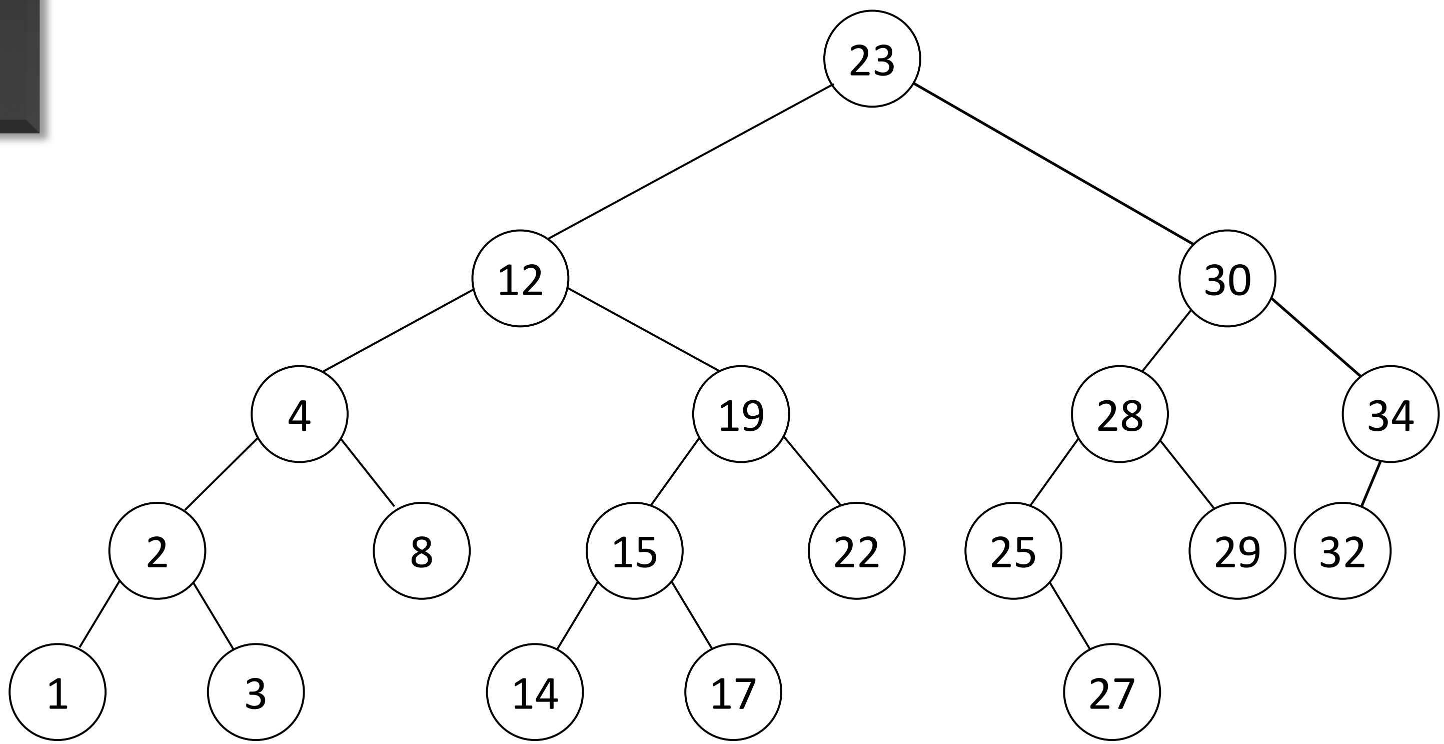
Equilibrage

- Cet arbre est-il équilibré ?



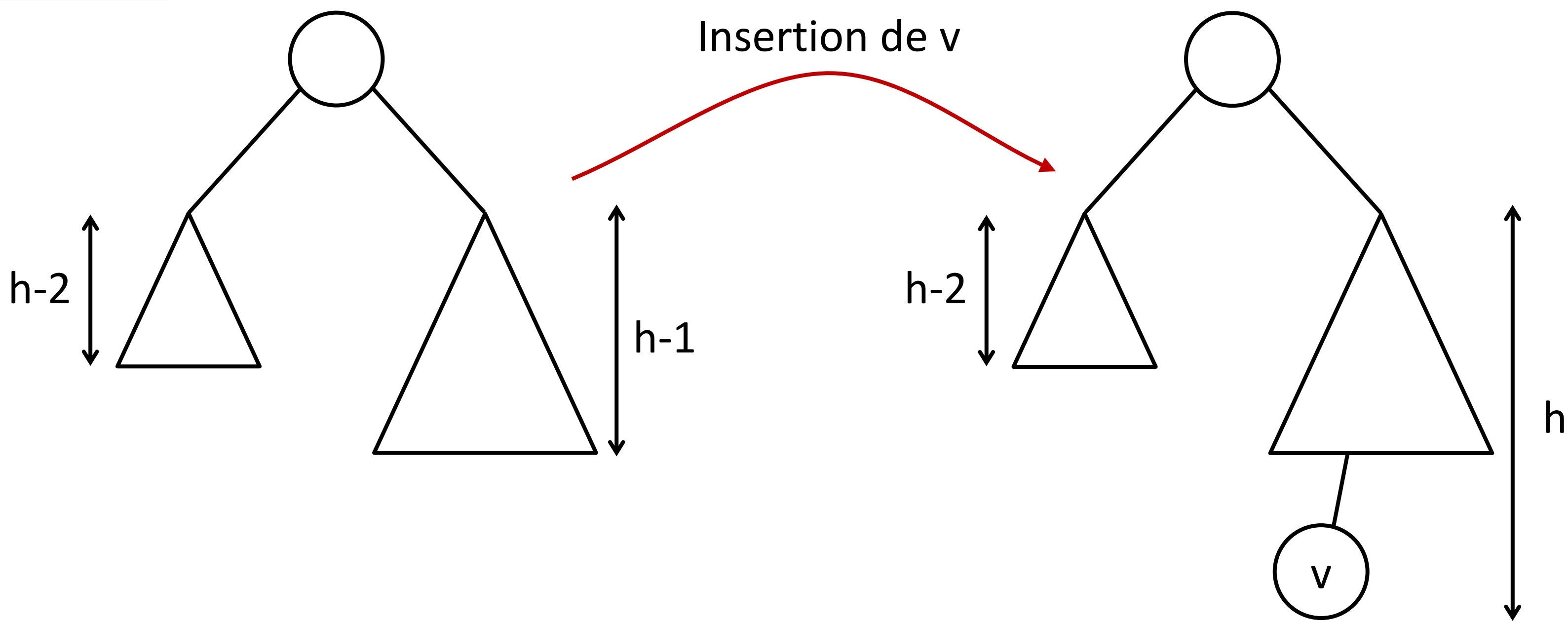
Equilibrage

- Et celui-là ?



Equilibrage

- Insertion ou suppression peuvent faire disparaître la propriété d 'équilibre



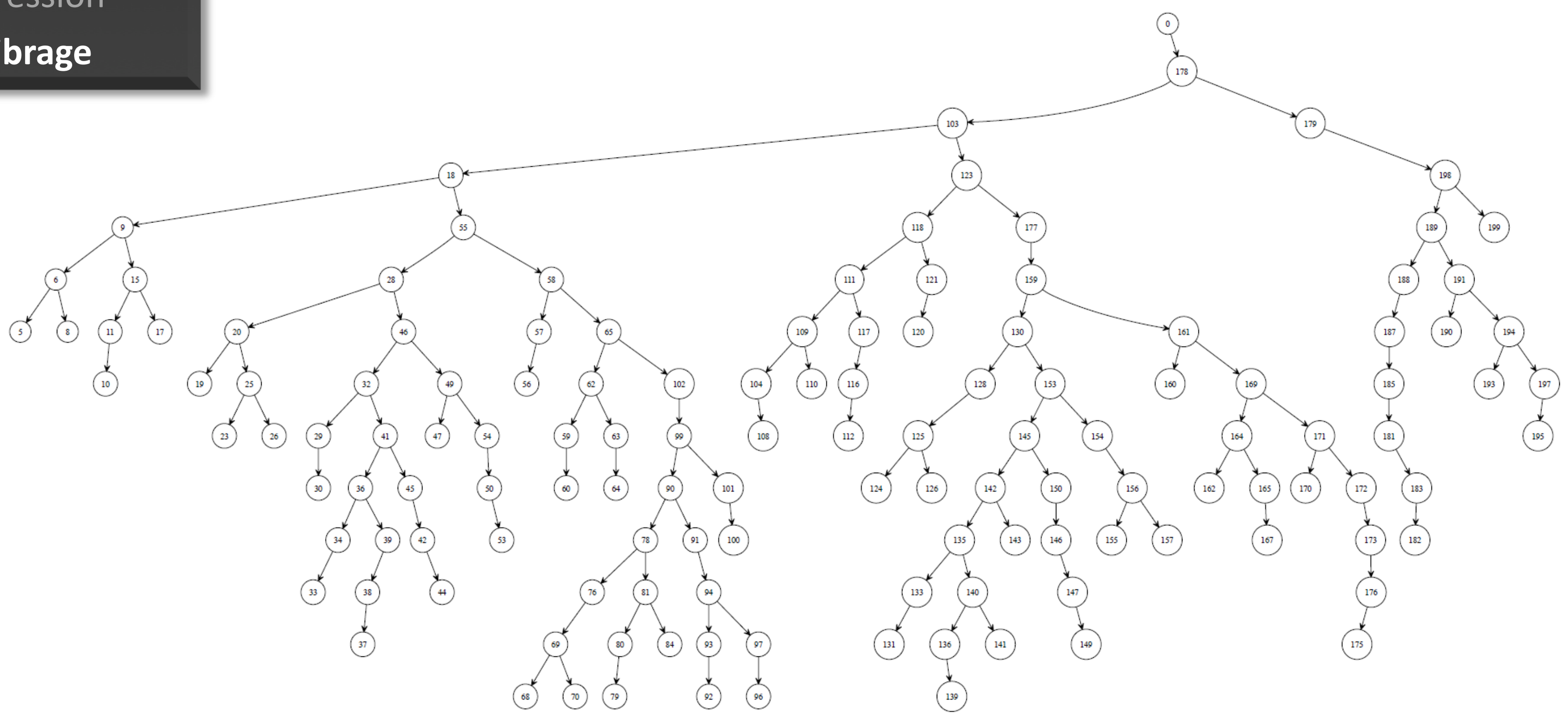
Equilibrage

- Exemple de remplissage d’un ABR

Equilibrage

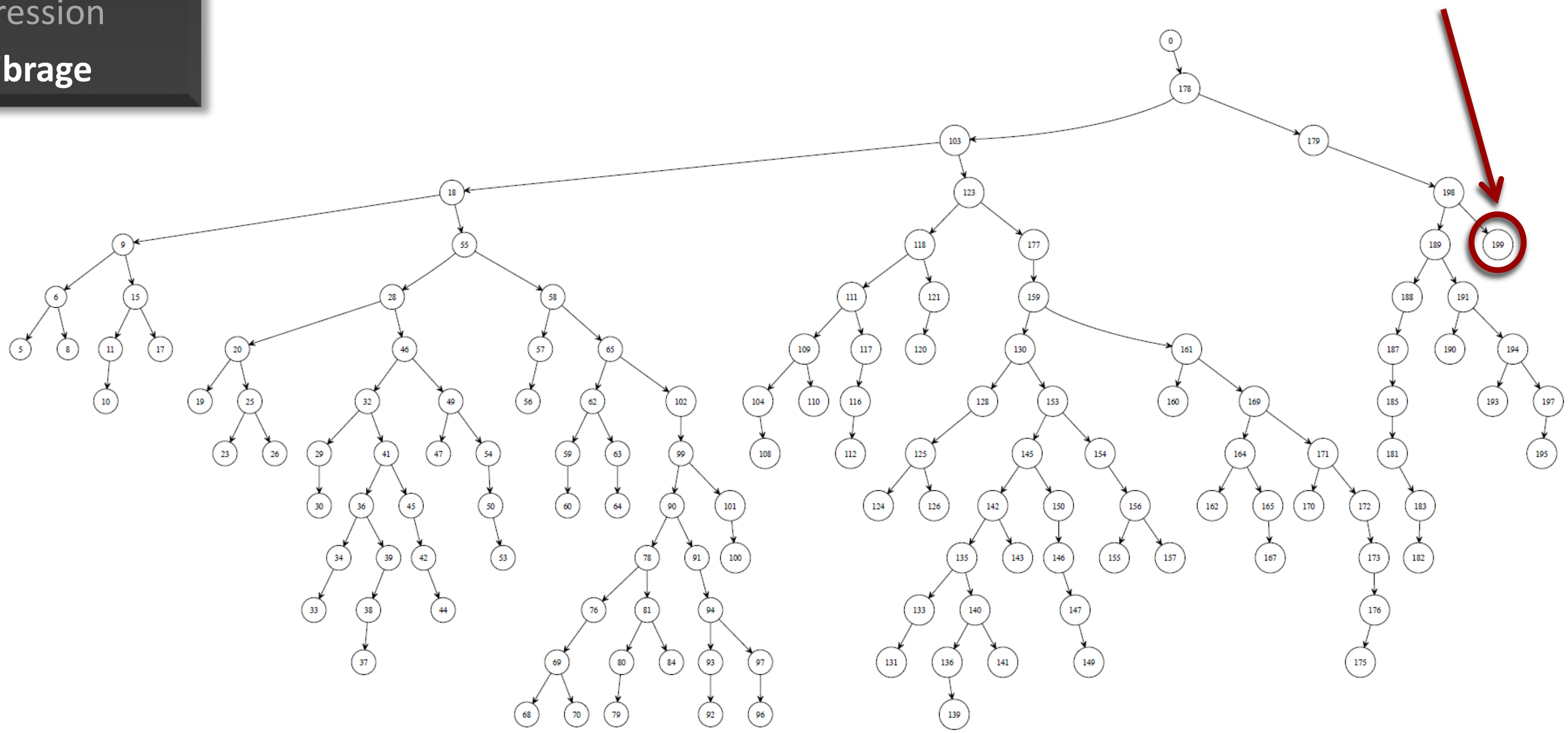
Introduction
Insertion
Suppression
Equilibrage

- Hauteur finale 12 au lieu de 7 s'il était complet (donc parfaitement équilibré)



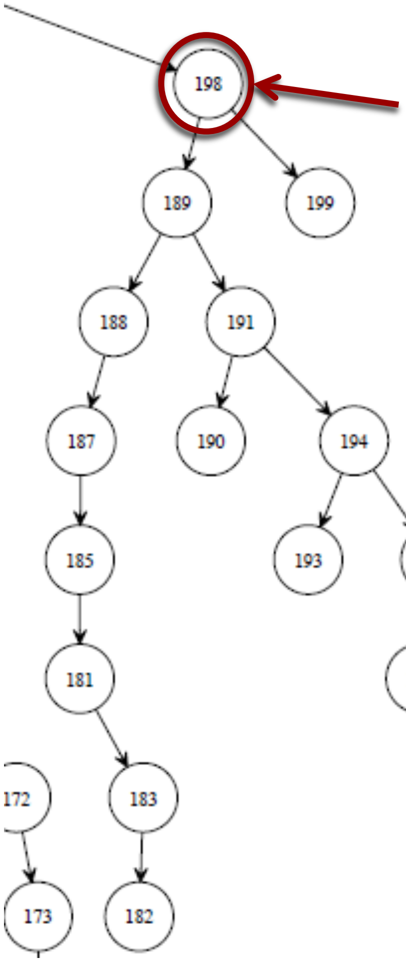
Equilibrage

- Profondeur minimum = 3 !



Equilibrage

- Déséquilibre énorme !



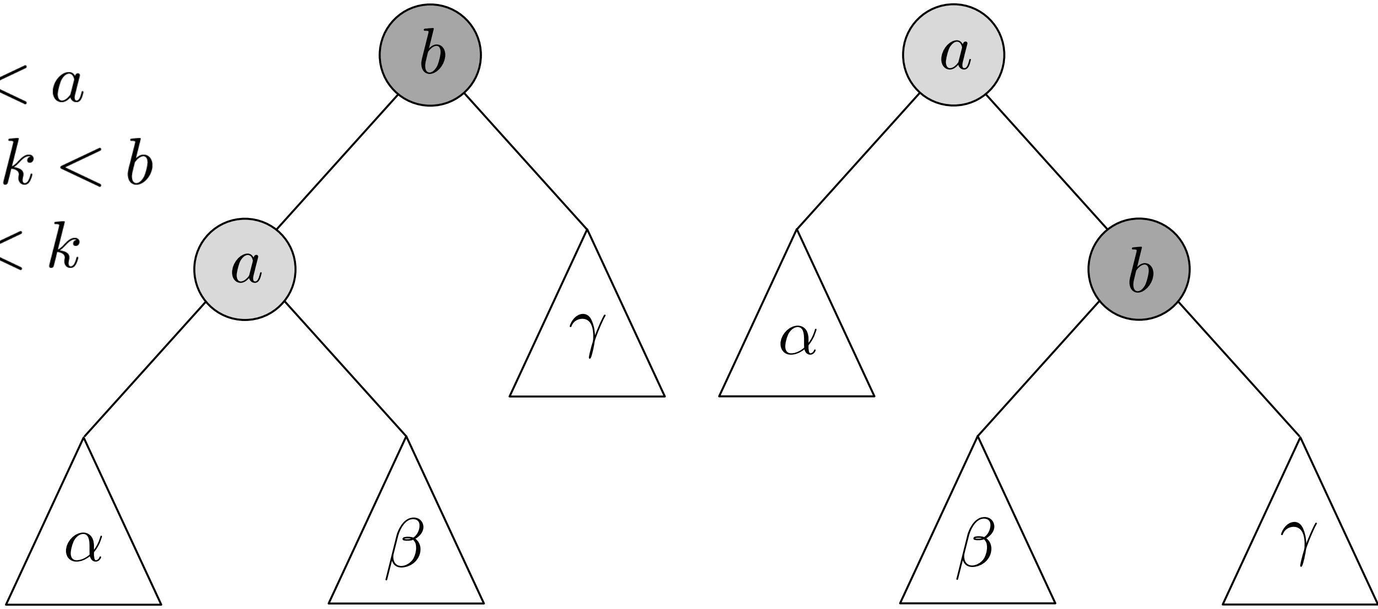
Equilibrage

- Il est possible de maintenir un arbre équilibré au cours de sa création
- Les AVL
  - Auteurs Adelson-Velskii et Landis
  - Rotations locales

Equilibrage

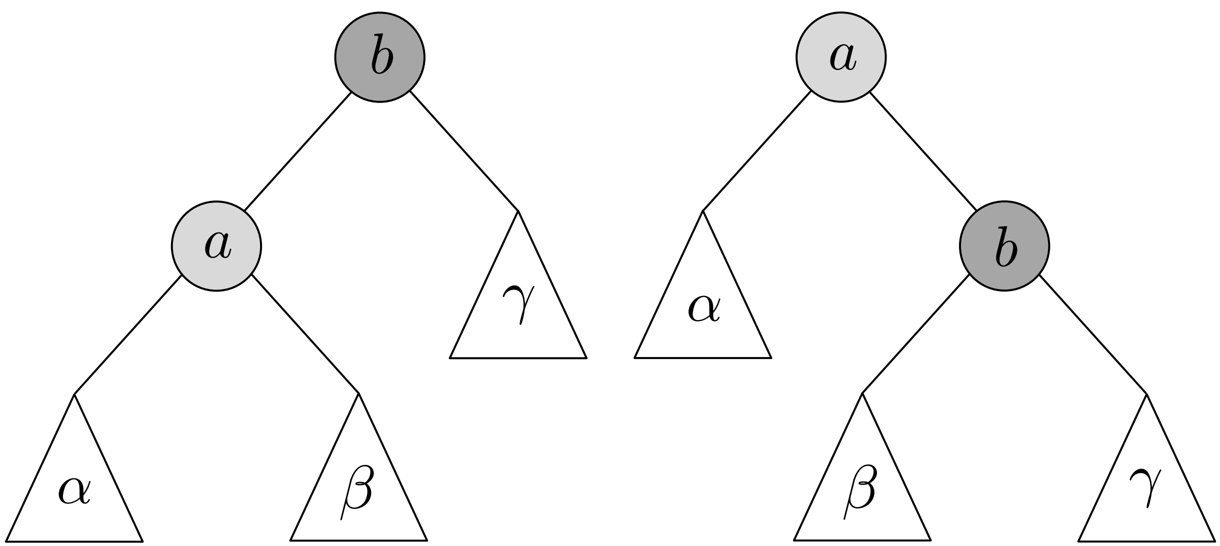
- Rotation
- Deux configurations équivalentes

$$\begin{cases} \forall k \in \alpha & k < a \\ \forall k \in \beta & a < k < b \\ \forall k \in \gamma & b < k \end{cases}$$



Equilibrage

- Modification de la hauteur
- A gauche $\max(h(\alpha)+2, h(\beta)+2, h(\gamma)+1)$
- A droite $\max(h(\alpha)+1, h(\beta)+2, h(\gamma)+2)$



Equilibrage

- Ne change pas la hauteur globale de γ
- Augmente la hauteur globale de α
- Diminue la hauteur globale de β

Equilibrage

- Arbre rouge-noir
- Etiquetage de chaque nœud avec deux couleurs
- Plus performant que l'AVL pour l'insertion
- Mais l'équilibre est moins bien respecté