Algorithmique des arbres

Arbres équilibrés

L2 Mathématique et Informatique

Outline

Arbres binaires de recherche : Définition

Définition

Soit E un ensemble ordonné par une relation d'ordre $<$.
Soit aussi A un arbre binaire ayant des nœuds étiquettés par des éléments de E.

A est appelé un *arbre binaire de recherche* (ABR en abrégé) lorsque :

- A est vide

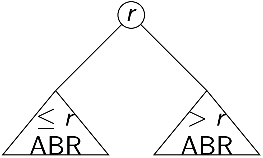
ou

- $A = (r, G, D)$ avec G et D des arbres binaires de recherche tels que :

$$\text{Elements}(G) \leq \text{Element}(r) < \text{Elements}(D)$$

Ici :

Elements(A) $= \{$etiquettes présentes dans l'arbre A $\}$
Element(nœud) $=$ etiquette du nœud



Opérations sur les ABR

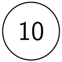
- `creer_ABR_vide();`
- `rechercher(A, x);`
- `ajout(A, x);`
- `supprimer(A, x);`
- `extraire_max(A);`
- `extraire_min(A);`

$\rightsquigarrow$ Compléxité en $\mathcal{O}\big(\mathit{Hauteur}(A)\big)$.

$\rightsquigarrow$ Compléxité en $\mathcal{O}\big(\ln(n)\big)$ en pratique.

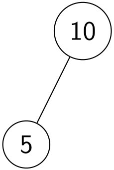
Exemple d'insertions / suppressions

On insère successivement les nombres 10, 5, 12, 15, 2 et 30 dans un arbre binaire de recherche initialement vide, puis on supprime 5 :



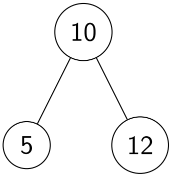
Exemple d'insertions / suppressions

On insère successivement les nombres 10, 5, 12, 15, 2 et 30 dans un arbre binaire de recherche initialement vide, puis on supprime 5 :



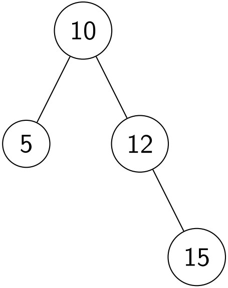
Exemple d'insertions / suppressions

On insère successivement les nombres 10, 5, 12, 15, 2 et 30 dans un arbre binaire de recherche initialement vide, puis on supprime 5 :



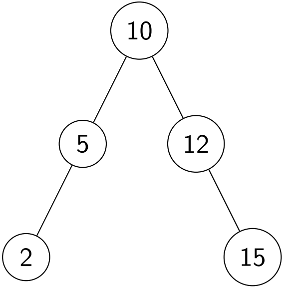
Exemple d'insertions / suppressions

On insère successivement les nombres 10, 5, 12, 15, 2 et 30 dans un arbre binaire de recherche initialement vide, puis on supprime 5 :



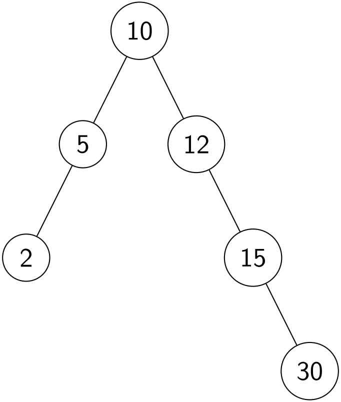
Exemple d'insertions / suppressions

On insère successivement les nombres 10, 5, 12, 15, 2 et 30 dans un arbre binaire de recherche initialement vide, puis on supprime 5 :



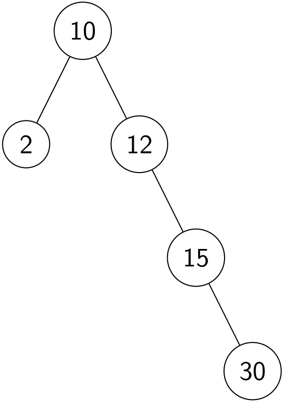
Exemple d'insertions / suppressions

On insère successivement les nombres 10, 5, 12, 15, 2 et 30 dans un arbre binaire de recherche initialement vide, puis on supprime 5 :



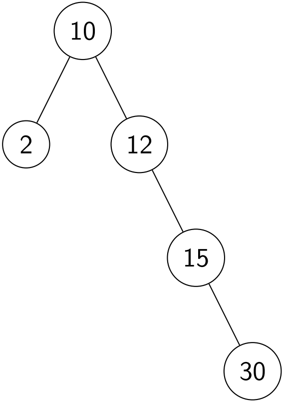
Exemple d'insertions / suppressions

On insère successivement les nombres 10, 5, 12, 15, 2 et 30 dans un arbre binaire de recherche initialement vide, puis on supprime 5 :



Exemple d'insertions / suppressions

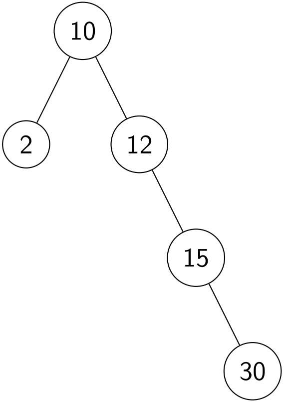
On insère successivement les nombres 10, 5, 12, 15, 2 et 30 dans un arbre binaire de recherche initialement vide, puis on supprime 5 :



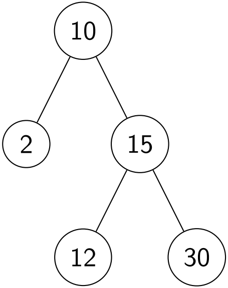
L'arbre penche clairement à droite...

Exemple d'insertions / suppressions

On insère successivement les nombres 10, 5, 12, 15, 2 et 30 dans un arbre binaire de recherche initialement vide, puis on supprime 5 :



On aurait mieux aimé avoir quelque chose comme :



L'arbre penche clairement à droite...

Motivations pour améliorer la structure de données ABR

Objectif : Limiter la différence de longueur des branches dans un ABR.

Parmi les méthodes :

- rééquilibrer les longueurs des branches d'un ABR aprés chaque ajout-suppression pour s'approcher d'un arbre plein.
- controler le nombre d'enfants de chaque nœud d'un arbre n-aire pour que les sous-arbres de même niveau aient un nombre de nœuds équivalent.

Motivations pour améliorer la structure de données ABR

Objectif : Limiter la différence de longueur des branches dans un ABR.

Parmi les méthodes :

- rééquilibrer les longueurs des branches d'un ABR aprés chaque ajout-suppression pour s'approcher d'un arbre plein.

  $\rightsquigarrow$ AVL

- controler le nombre d'enfants de chaque nœud d'un arbre n-aire pour que les sous-arbres de même niveau aient un nombre de nœuds équivalent.

  $\rightsquigarrow$ B-arbres

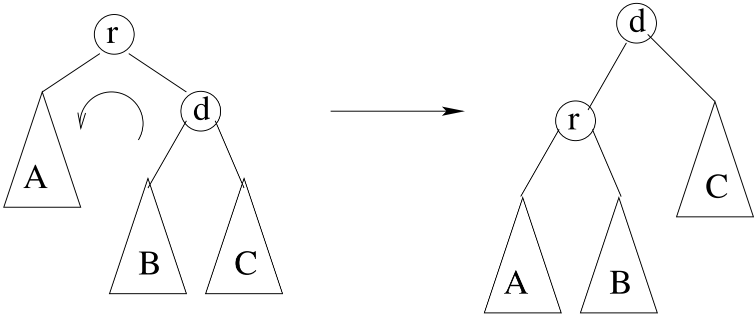
Outline

Outline

Rotation gauche d'ABR 1 / 2

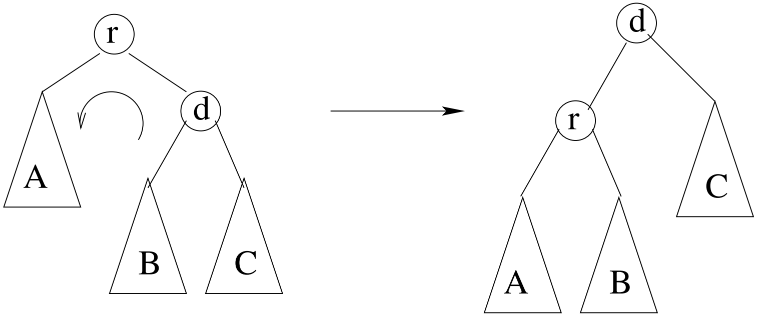
Soit E est un ensemble ordonné par une relation d'ordre $<$.
Soit aussi r, $d \in E$ vérifiant : $r < d$.

On définit alors la rotation gauche comme suit :



Le sous-arbre A est descendu d'un niveau.
Le sous-arbre C est remonté d'un niveau.
Le sous-arbre B n'a pas changé de niveau.

Rotation gauche d'ABR 2 / 2



Propriété :

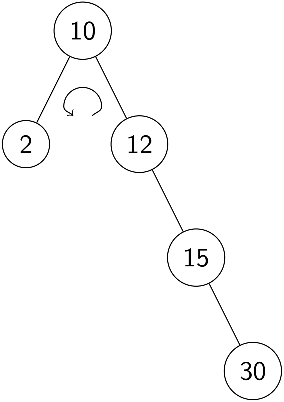
Dans un ABR, les relations d'ordre entre les nœuds, après une rotation gauche, sont conservées.

Preuve : Si a est un nœud de A, on a : $a \leq r$.
Si b est un nœud de B, on a : $r < b \leq d$.
Si c est un nœud de C, on a : $d < c$.

Seule différence : Avant rotation $r < d$; Après rotation $r \leq d$ ...

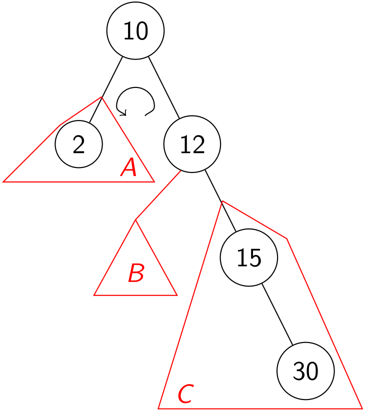
Exemple de rotation gauche

Reprenons l'ABR construit précédemment :



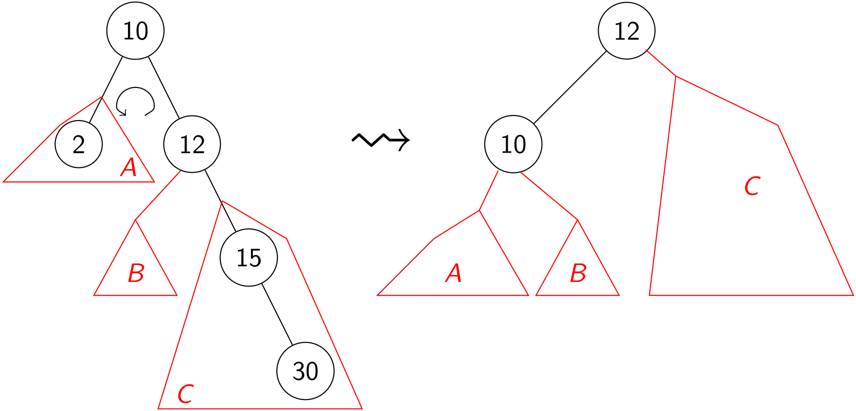
Exemple de rotation gauche

Reprenons l'ABR construit précédemment :



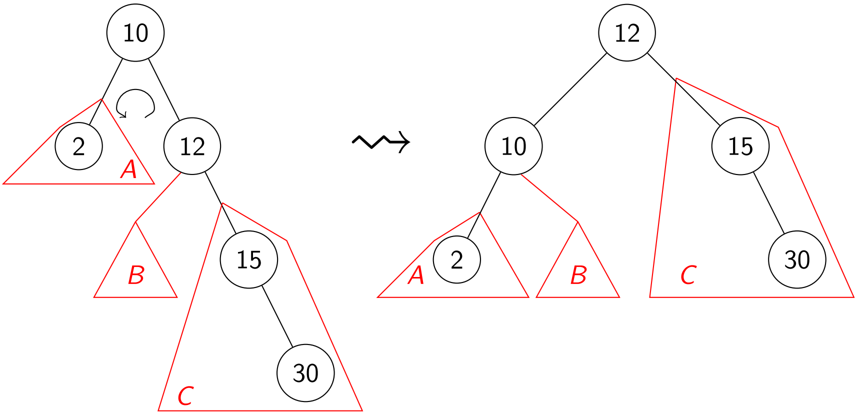
Exemple de rotation gauche

Reprenons l'ABR construit précédemment :



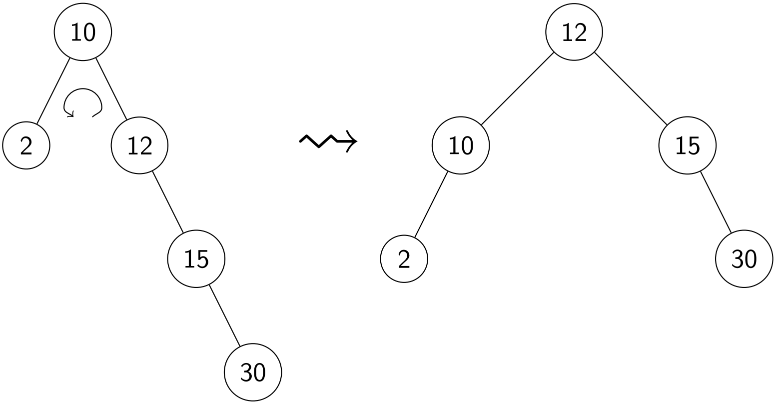
Exemple de rotation gauche

Reprenons l'ABR construit précédemment :



Exemple de rotation gauche

Reprenons l'ABR construit précédemment :

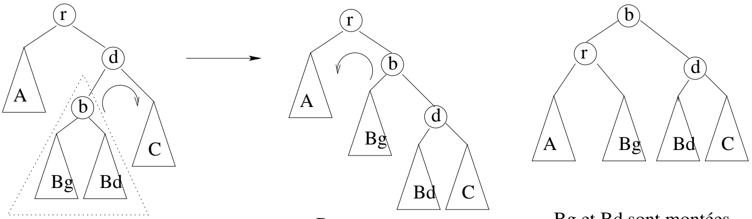


Double rotation droite-gauche d'ABR 1 / 2

Soit E est un ensemble ordonné par une relation d'ordre $<$.
Soit aussi r, d et $b \in E$ vérifiant : $r < b < d$.

Pour changer le niveau du sous arbre B, on effectue une double rotation :

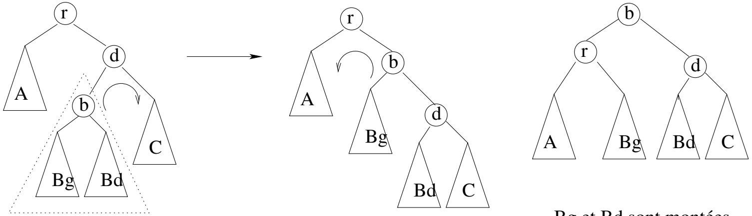
- une rotation droite au niveau de la racine du sous-arbre B ;
- une rotation gauche au niveau de la racine de l'arbre.



Bg monte
C descend
Bd sans changement

Bg et Bd sont montées
A descend
C sans changement

Double rotation droite-gauche d'ABR 2 / 2



Bg monte
C descend
Bd sans changement

Bg et Bd sont montées
A descend
C sans changement

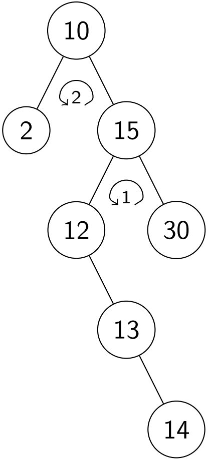
Propriété :

Dans un ABR, les relations d'ordre entre les nœuds, après une double rotation droite-gauche, sont conservées.

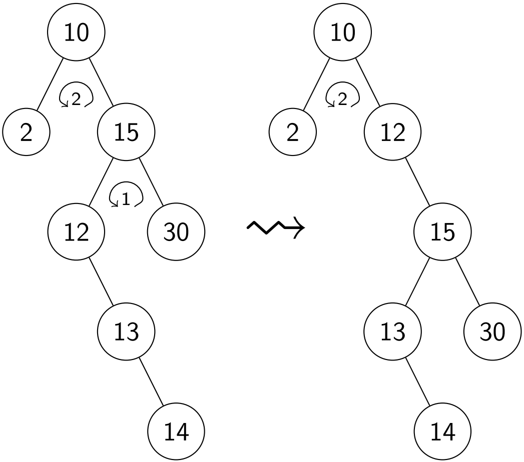
Preuve : Si a est un nœud de A, on a : $a \leq r$.
Si b_g est un nœud de Bg, on a : $r < b_g \leq b \leq d$.
Si b_d est un nœud de Bd, on a : $b < b_d \leq d$.
Si c est un nœud de C, on a : $d < c$.

Seule différence : Avant rotation $r < b \leq d$
Après rotation $r \leq b < d$ ...

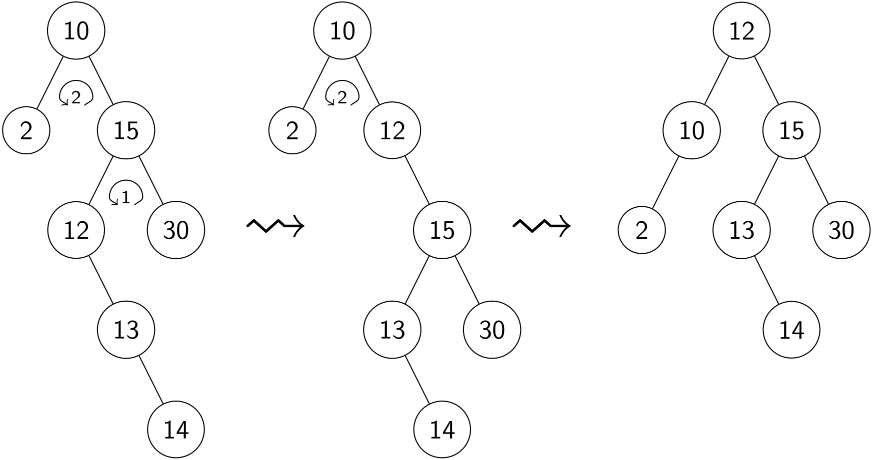
Exemple de double rotation droite-gauche



Exemple de double rotation droite-gauche



Exemple de double rotation droite-gauche



Outline

Définitions

Définition :

Soit A un arbre.
On définit la balance d'un nœud p de A par :

$$bal(p) = hauteur\big(A_d(p)\big) - hauteur\big(A_g(p)\big)$$

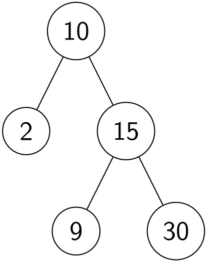
où $A_d(p)$ et $A_g(p)$ désigne respectivement le sous-arbre gauche et droit du nœud p.

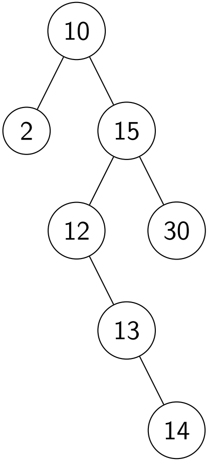
Définition :

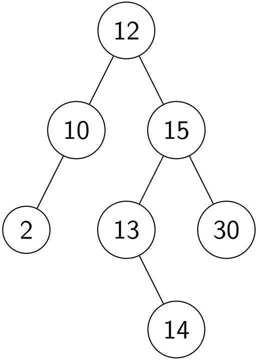
Un arbre AVL est un arbre binaire de recherche tel que la différence de hauteur entre les enfants d'un nœud interne est au plus 1 :

$$\text{pour tout nœud } p \text{ de } A, \ |bal(p) < 2|.$$

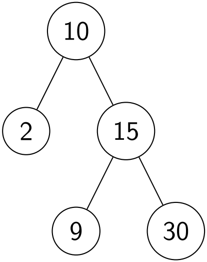
Exemples et contre-exemples



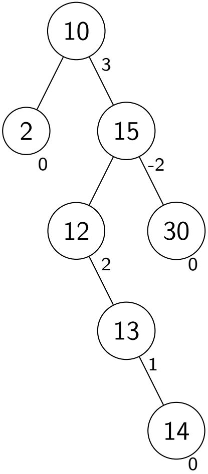




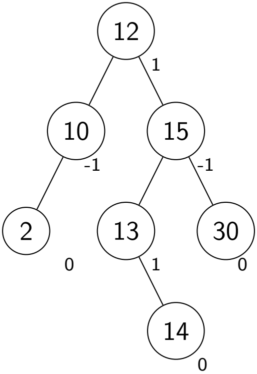
Exemples et contre-exemples



N'est même pas un ABR...



ABR non équilibré
$\Longrightarrow$ n'est pas un AVL



Exemple d'AVL

Hauteur et nombre de nœuds d'un AVL

Soit nb_h^{min} le nombre minimal de nœuds dans un AVL de hauteur h.
Alors :

$$\begin{cases} nb_{-1}^{min} & = & 0 \\ nb_0^{min} & = & 1 \\ nb_1^{min} & = & 2 \\ nb_h^{min} & = & 1 + nb_{h-1}^{min} + nb_{h-2}^{min}, \text{ si } h \geq 1 \end{cases}$$

On peut montrer que si F_n désigne le n-ième nombre de Fibonacci ($F_0 = 0$, $F_1 = 1$, $F_k = F_{k-1} + F_{k-2}$), on a : $nb_h^{min} = F_{h+3} - 1$.

Rappels : $F_n = \frac{1}{\sqrt{5}}(\phi^n - \psi^n)$, avec $\phi = \frac{1+\sqrt{5}}{2}$ et $\psi = \frac{1-\sqrt{5}}{2}$.

$$F_n = \left\lfloor \frac{1}{\sqrt{5}}\phi^n + \frac{1}{2} \right\rfloor$$

Propriété :

Soit A un AVL à n nœuds de hauteur h.
Alors, il existe α, $\beta \in \mathbb{R}$ tel que $h \leq \alpha + \beta \ln(n+3)$, i.e. $h = \mathcal{O}(\log n)$.

Les AVL de Fibonacci 1 / 2

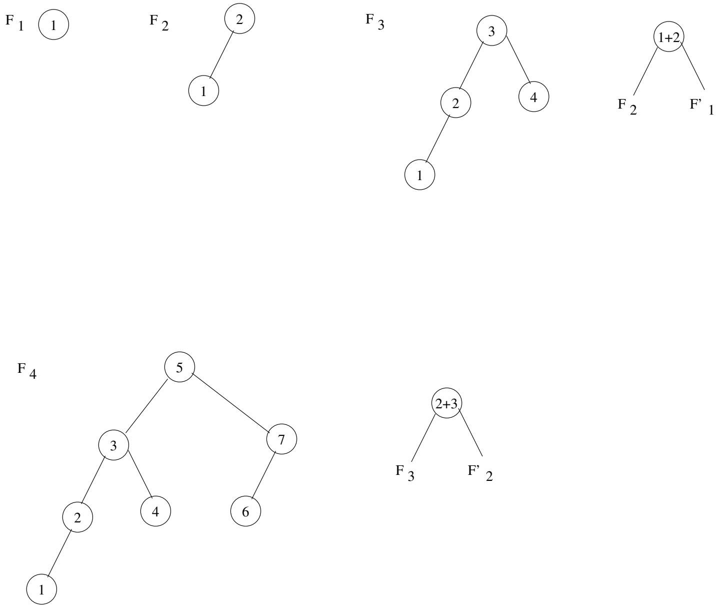
Les arbres de Fibonacci sont un exemple d'arbre AVL de hauteur donné ayant le nombre minimal de nœuds.

En identifiant nœud et étiquette $\in \{1, 2, \ldots\}$, on pose :

- $\mathbf{F}_0 = \Lambda$ arbre vide, de hauteur ($\mathbf{F}_0$)= -1
- $\mathbf{F}_1 = (1, \Lambda, \Lambda)$, de hauteur ($\mathbf{F}_1$)= 0
- $\mathbf{F}_k = (F_{k+1}, \mathbf{F}_{k-1}\ , \mathbf{F'}_{k-2})$, de hauteur($\mathbf{F}_k$) $= k - 1$

  où $\mathbf{F'}_{k-2}$ est l'arbre $\mathbf{F}_{k-2}$ avec les étiquettes augmentées de F_{k+1}

Les AVL de Fibonacci 2 / 2



Opérations dans les AVL

Puisqu'un AVL vérifie $h = \mathcal{O}(\log n)$ où n est le nombre d'éléments de A, on assure que les opérations :

- minimum(A),maximum(A)
- rechercher(A, x)
- ajout(A, x)
- supprimer(A, x)

peuvent être effectuées en $\mathcal{O}(\log n)$, où n est le nombre d'éléments dans A.

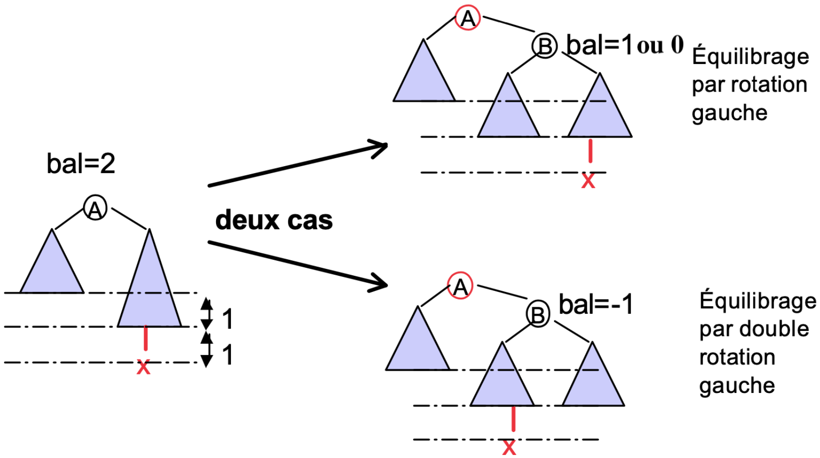
Outline

Algorithme d'ajout dans un AVL

Soit A un AVL. On souhaite y ajouter la valeur x.

- On ajoute x dans A comme dans un ABR.
- Mise à jour des balances des différents nœuds.
- Si nécessaire, on rééquilibre l'arbre en faisant <u>une</u> rotation (pas de cascade remontante !).

Rééquilibrage après un ajout



Rééquilibrage après un ajout : synthèse

Remarque : Lors de l'ajout de x dans un AVL, les nœud déséquilibrés sont nécessairement sur le chemin reliant x à la racine de l'arbre.

Notons r le premier nœud déséquilibré sur le chemin reliant x à la racine, après l'ajout de x dans un AVL.

- Si $bal(r) = 2$ et $bal(\text{fils droit de } r) < 0$, :

  $\rightsquigarrow$ on fait une double rotation droite-gauche ;

- Si $bal(r) = 2$ et $bal(\text{fils droit de } r) \geq 0$:

  $\rightsquigarrow$ on fait une rotation gauche ;

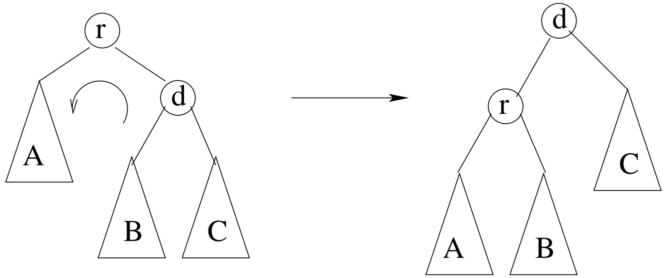
- Si $bal(r) = -2$ et $bal(\text{fils gauche de } r) < 0$:

  $\rightsquigarrow$ on fait une rotation droite ;

- Si $bal(r) = -2$ et $bal(\text{fils gauche de } r) \geq 0$:

  $\rightsquigarrow$ on fait une double rotation gauche-droite.

Preuves 1 / 2



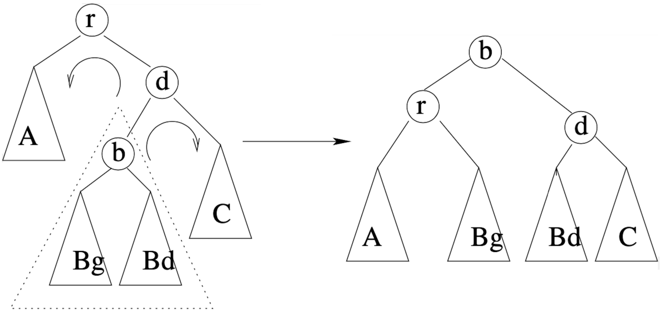
<u>**Hypothèses :**</u> $\begin{cases} bal(r) = 2 \\ bal(d) = 0 \text{ ou } 1 \end{cases}$

On a : $\begin{cases} 1 + \max\big(h(B), h(C)\big) - h(A) = 2 \\ h(C) - h(B) = 0 \text{ ou } 1 \end{cases}$, soit $\begin{cases} h(C) - h(A) = 1 \\ h(C) - h(B) = 0 \text{ ou } 1 \end{cases}$

Après rotation gauche, on a alors :

$$\begin{cases} bal'(r) & = & h(B) - h(A) = \big(h(B) - h(C)\big) + \big(h(C) - h(A)\big) \\ & = & (0 \text{ ou } -1) + 1 = 0 \text{ ou } 1 \\ bal'(d) & = & h(C) - \Big(1 + \max\big(h(A), h(B)\big)\Big) = 0 \text{ ou } -1 \end{cases}$$

Preuves 2 / 2



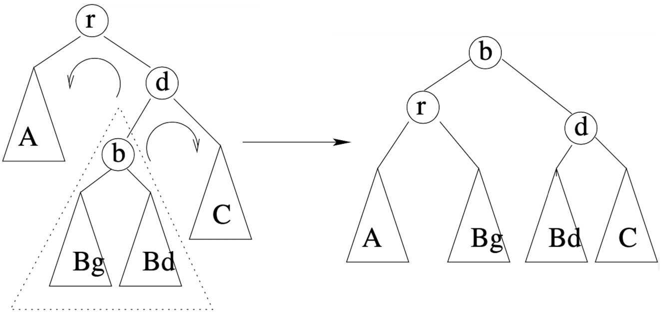
Hypothèses : $\begin{cases} bal(r) = 2 \\ bal(d) = -1 \end{cases}$

On a alors : $\begin{cases} 1 + \max\left(1 + \max\left(h(Bg), h(Bd)\right), h(C)\right) - h(A) = 2 \\ h(C) - \left(1 + \max\left(h(Bg), h(Bd)\right)\right) = -1 \end{cases}$

Soit :

$$\max\left(h(Bg), h(Bd)\right) = h(A) = h(C)$$

Preuves 2 / 2

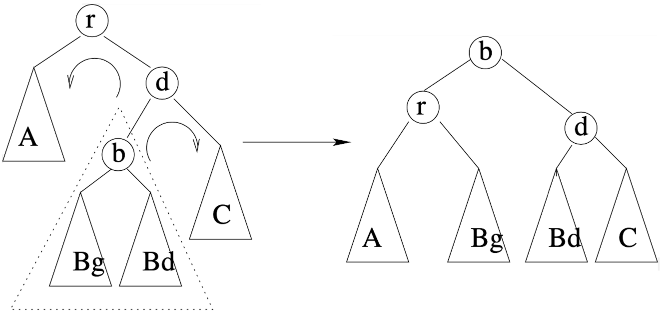


<u>**Hypothèses :**</u> $\begin{cases} bal(r) = 2 \\ bal(d) = -1 \end{cases}$ soit $\max\left(h(Bg), h(Bd)\right) = h(A) = h(C)$

Après la double rotation droite-gauche, on a alors :

$$\begin{aligned} bal'(r) &= h(Bg) - h(A) = -bal(b) + h(Bd) - h(A) \\ &= \begin{cases} 0 & \text{, si } \max\left(h(Bg), h(Bd)\right) = h(Bg) \\ -bal(b) & \text{, si } \max\left(h(Bg), h(Bd)\right) = h(Bd) \end{cases} \end{aligned}$$

Preuves 2 / 2

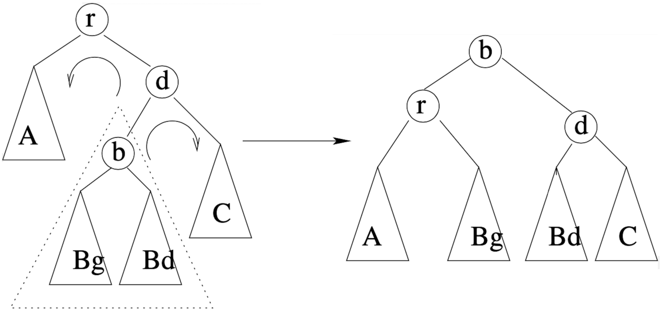


Hypothèses : $\begin{cases} bal(r) = 2 \\ bal(d) = -1 \end{cases}$ soit $\max\left(h(Bg), h(Bd)\right) = h(A) = h(C)$

Après la double rotation droite-gauche, on a alors :

$$\begin{aligned} bal'(d) &= h(Bd) - h(A) = bal(b) + h(Bg) - h(A) \\ &= \begin{cases} bal(b) & \text{, si } \max\left(h(Bg), h(Bd)\right) = h(Bg) \\ 0 & \text{, si } \max\left(h(Bg), h(Bd)\right) = h(Bd) \end{cases} \end{aligned}$$

Preuves 2 / 2

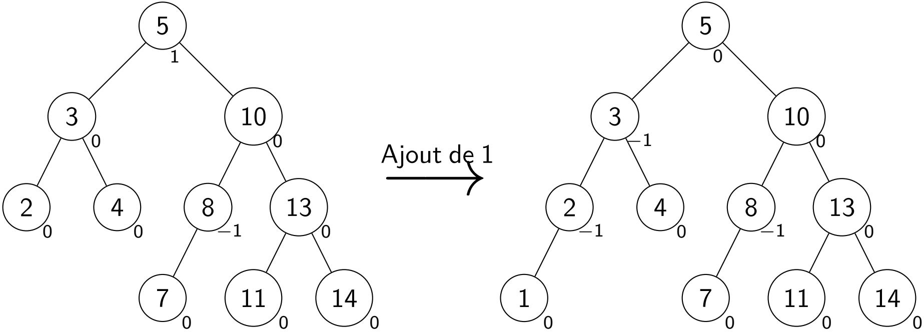


Hypothèses : $\begin{cases} bal(r) = 2 \\ bal(d) = -1 \end{cases}$ soit $\max\big(h(Bg), h(Bd)\big) = h(A) = h(C)$

Après la double rotation droite-gauche, on a alors :

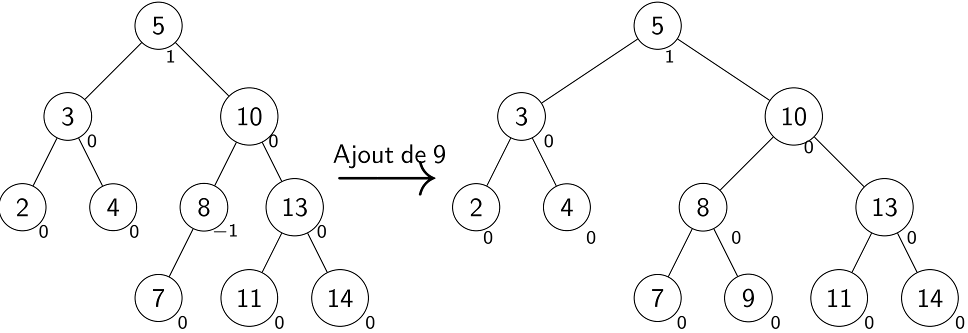
$$\begin{aligned} bal'(b) &= \max\big(h(Bd), h(C)\big) - \max\big(h(Bg), h(A)\big) \\ &= h(C) - h(A) = 0 \end{aligned}$$

Exemples d'ajouts dans un AVL 1 / 4



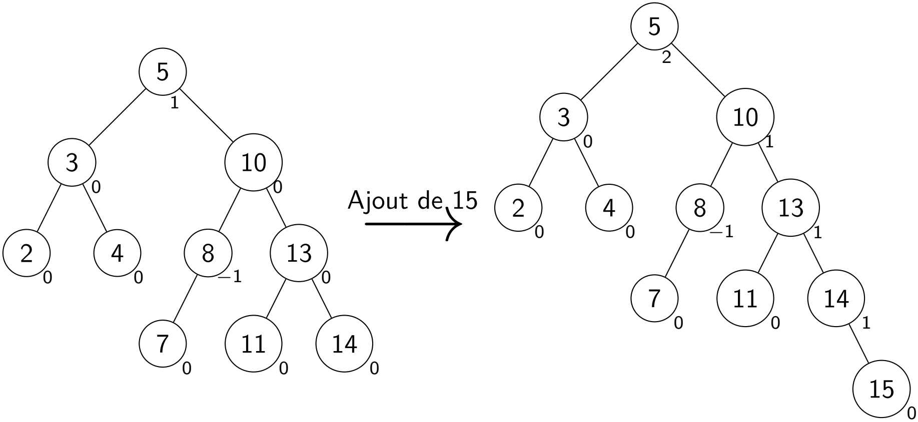
$\Longrightarrow$ Pas de rééquilibrage à réaliser.

Exemples d'ajouts dans un AVL 2 / 4



$\Longrightarrow$ Pas de rééquilibrage à réaliser.

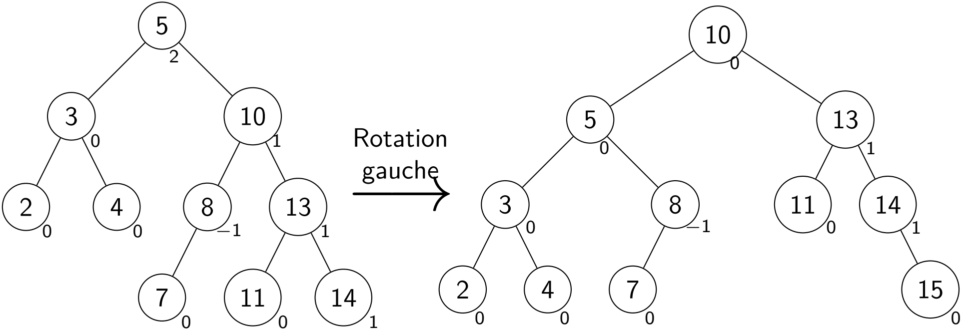
Exemples d'ajouts dans un AVL 3 / 4



$\Longrightarrow$ Rééquilibrage à réaliser au niveau de la racine

$\Longrightarrow$ Rotation gauche

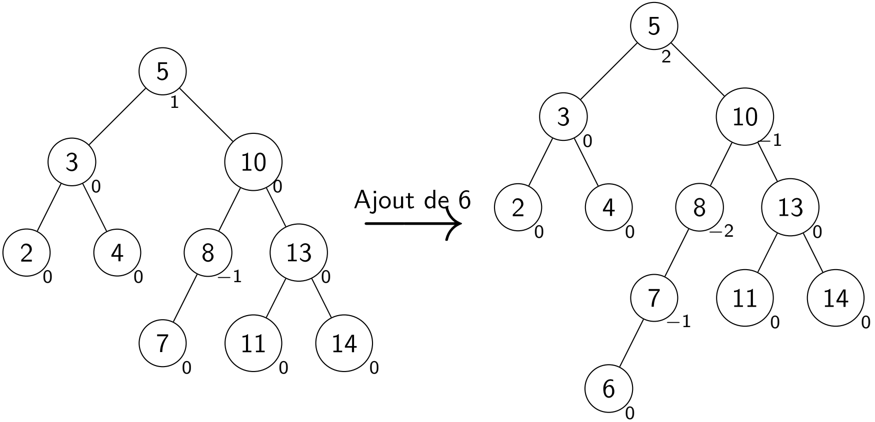
Exemples d'ajouts dans un AVL 3 / 4



$\Longrightarrow$ Rééquilibrage à réaliser au niveau de la racine

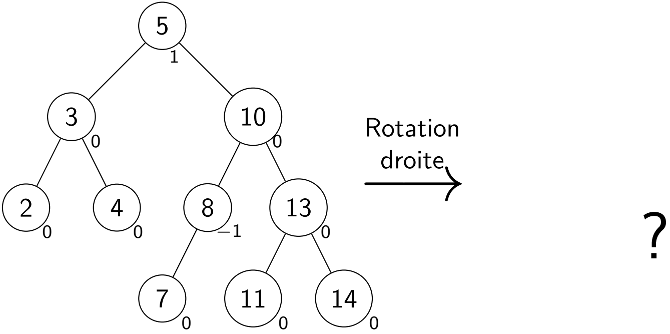
$\Longrightarrow$ Rotation gauche

Exemples d'ajouts dans un AVL 4 / 4



$\Longrightarrow$ Rééquilibrage à réaliser au niveau du nœud 8

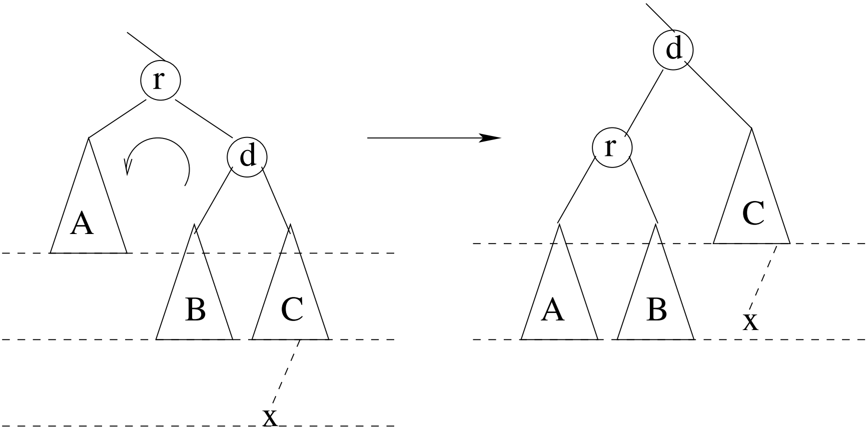
$\Longrightarrow$ Rotation droite



$\Longrightarrow$ Rééquilibrage à réaliser au niveau du nœud 8
$\Longrightarrow$ Rotation droite

Soit `r` est le plus bas noeud déséquilibré d'un arbre AVL après un ajout (comme dans un ABR).

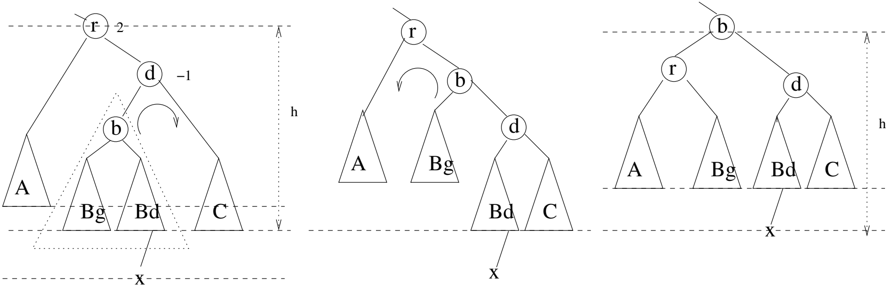
- Si l'ajout entraine une rotation simple :



Variation de hauteur lors d'un ajout 2 / 3

Soit `r` est le plus bas noeud déséquilibré d'un arbre AVL après un ajout (comme dans un ABR).

- Si l'ajout entraine une rotation double :



r est le plus bas noeud dont la balance passe à2

Variation de hauteur lors d'un ajout 3 / 3

Soit `r` est le plus bas noeud déséquilibré d'un arbre AVL après un ajout (comme dans un ABR).

On remarque qu'après une rotation, la hauteur du sous-arbre de racine r (qui est devenu déséquilibré par l'ajout de x) est redevenue égale à la hauteur qu'il avait avant l'ajout.

Conséquence : Après un ajout, le rééquilibrage est local et ne se propage pas le long d'une branche.

Complexité de l'ajout d'un élément dans un AVL

Ajout d'un élément dans un AVL :

- Recherche de la place d'insertion
- Mise à jour des balances des différents nœuds.
- Si besoin, réalisation d'une rotation (simple ou double)

Complexité de l'ajout d'un élément dans un AVL

Ajout d'un élément dans un AVL :

- Recherche de la place d'insertion $\rightsquigarrow \mathcal{O}(\log n)$
- Mise à jour des balances des différents nœuds. $\rightsquigarrow \mathcal{O}(\log n)$ opérations
- Si besoin, réalisation d'<u>une</u> rotation (simple ou double) $\rightsquigarrow \mathcal{O}(1)$

L'ajout d'un élément dans un AVL à n éléments a une compléxité en :

$$\mathcal{O}(\log(n))$$

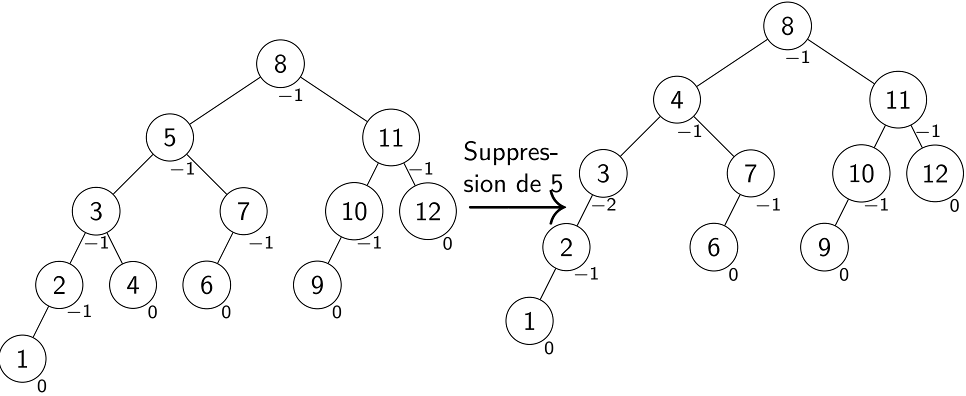
Outline

Algorithme de suppression dans un AVL

Soit A un AVL. On souhaite y supprimer la valeur x.

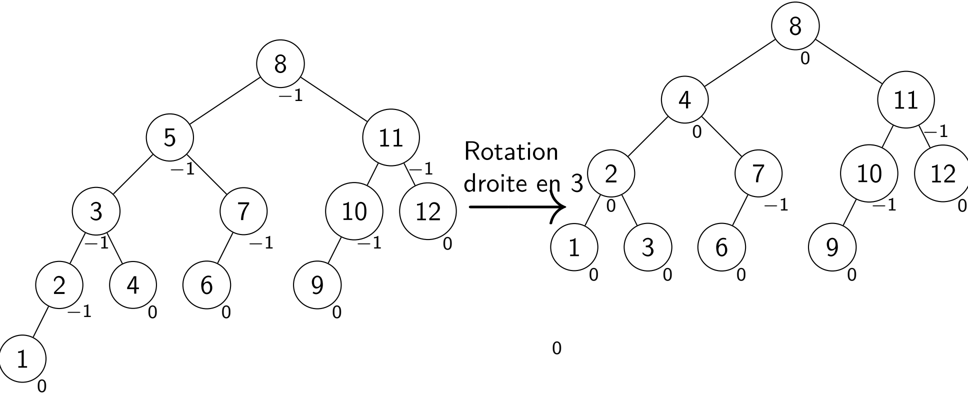
- On supprime x dans A comme dans un ABR (en le remplacant par l'élémént immédiatement inférieur (le plus grand du sous-arbre gauche) ou par celui qui lui est immédiatement supérieur (le plus petit du sous-arbre droit).
- Mise à jour des balances des différents nœuds.
- Si nécessaire, on rééquilibre l'arbre en faisant des rotations (qui peuvent être en cascade)

Exemples de suppression dans un AVL 1 / 2



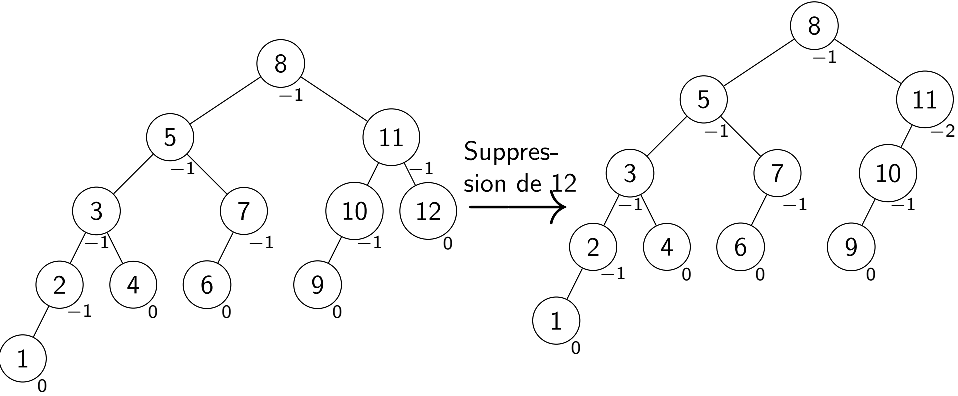
$\Longrightarrow$ Rééquilibrage à réaliser au niveau du nœud 3

Exemples de suppression dans un AVL 1 / 2



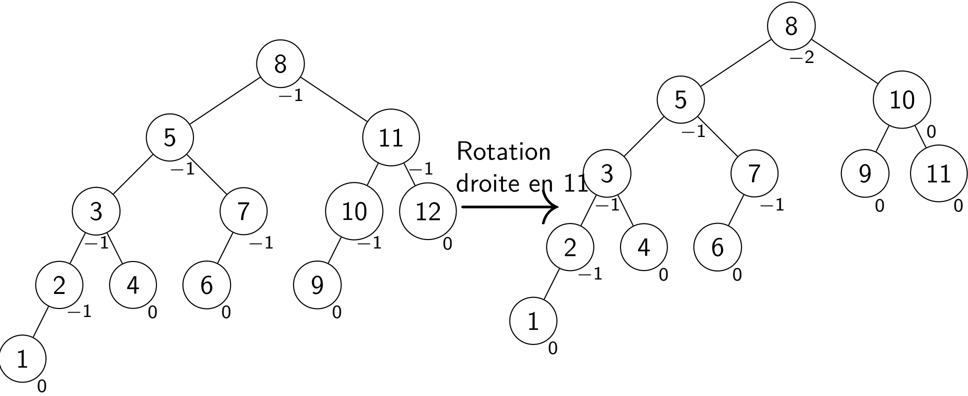
$\Longrightarrow$ Rééquilibrage à réaliser au niveau du nœud 3 : rotation droite

Exemples de suppression dans un AVL 2 / 2



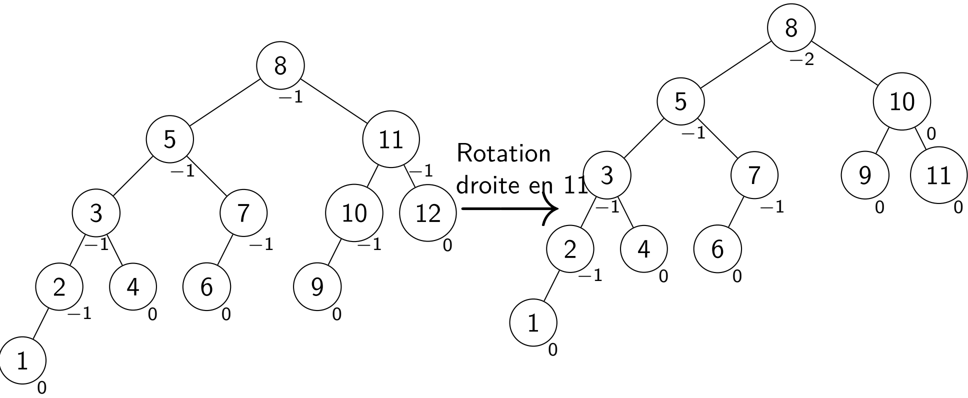
$\Longrightarrow$ Rééquilibrage à réaliser au niveau du nœud 11

Exemples de suppression dans un AVL 2 / 2



$\Longrightarrow$ Rééquilibrage à réaliser au niveau du nœud 11 : rotation droite

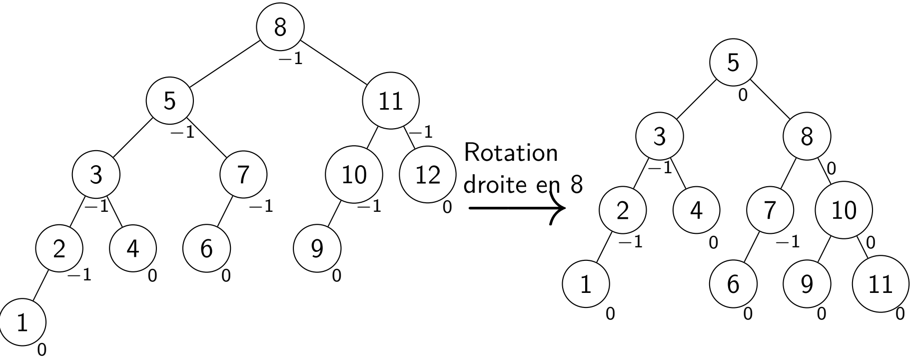
Exemples de suppression dans un AVL 2 / 2



$\Longrightarrow$ Rééquilibrage à réaliser au niveau du nœud 11 : rotation droite

$\Longrightarrow$ Rééquilibrage à réaliser au niveau de la racine : rotation droite

Exemples de suppression dans un AVL 2 / 2



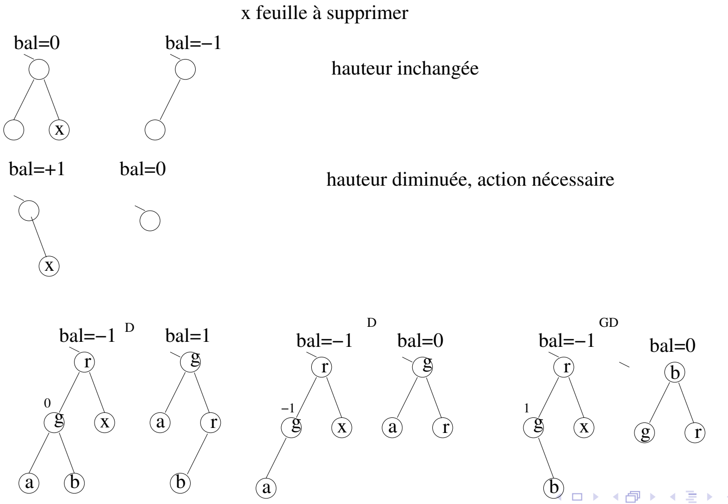
$\Longrightarrow$ Rééquilibrage à réaliser au niveau du nœud 11 : rotation droite

$\Longrightarrow$ Rééquilibrage à réaliser au niveau de la racine : rotation droite

Cela prouve que, contrairement à l'ajout, les rotations peuvent être en cascade pour la suppression.

Variation de hauteur après suppression d'une feuille.

D'une manière directe, ou après remontée du min du sous-arbre droite ou du max du sous-arbre gauche, supprimer un élément d'un AVL revient à rééquilibrer l'arbre comme si l'on supprimer une feuille :



Complexité de la suppression d'un élément dans un AVL

Suppression d'un élément dans un AVL :

- Recherche de la place de suppression
- Remontée du min du sous-arbre droit ou du max du sous-arbre gauche
- Mise à jour des balances des différents nœuds.
- Si besoin, réalisation <u>des</u> rotations (simples ou doubles)

Complexité de la suppression d'un élément dans un AVL

Suppression d'un élément dans un AVL :

- Recherche de la place de suppression $\rightsquigarrow \mathcal{O}(\log n)$ opérations
- Remontée du min du sous-arbre droit ou du max du sous-arbre gauche $\rightsquigarrow \mathcal{O}(1)$ opérations
- Mise à jour des balances des différents nœuds. $\rightsquigarrow \mathcal{O}(\log n)$ opérations
- Si besoin, réalisation des rotations (simples ou doubles) au plus, $h(A)$ rotation à faire, donc $\mathcal{O}(\log n)$ opérations

La suppression d'un élément dans un AVL à n élément a une compléxité en :

$$\mathcal{O}(\log(n))$$

Outline

Structure de base pour l'implémentation des arbres AVL

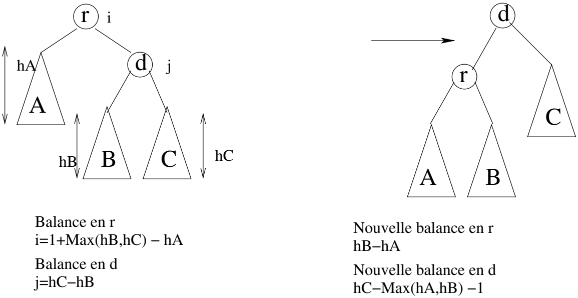
On utilise la structure :

```
typedef struct _noeud{
    int etiquette;
    int bal;
    struct _noeud *g;
    struct _noeud *d;
} noeud,*arbre;
```

Toute la difficulté de l'implémentation est de :

MAINTENIR LA BALANCE AU COURS DES DIFFÉRENTES ROTATIONS NÉCESSAIRES AU RÉÉQUILIBRAGE.

Mise à jour de la balance en cas de rotation gauche

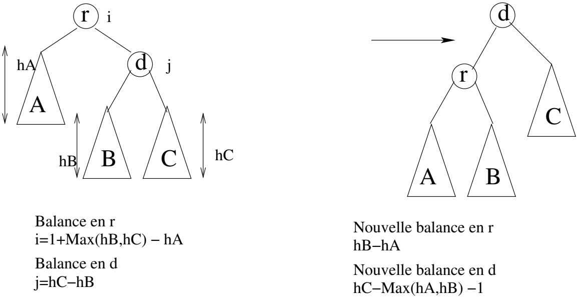


Nouvelle balance $bal'(r)$ en r :

- Si $j \leq 0$, $\max(hB, hC) = hB$. Donc, $i = bal(r) = 1 + hB - hA$.
  La nouvelle balance en r est $bal'(r) = i - 1$

- Sinon, $\max(hB, hC) = hC$. Donc, $i = bal(r) = 1 + hC - hA$.
  La nouvelle balance en r est :
  $bal'(r) = hB - hA = (1 + hC - hA) - (hC - hB) - 1 = i - j - 1$

Mise à jour de la balance en cas de rotation gauche

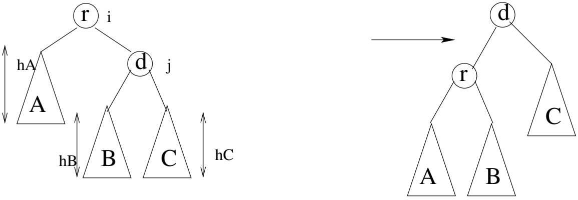


Nouvelle balance $bal'(d)$ en d :

- Si $bal'(r) \geq 0$, nouvelle $hauteur(r) = hB$.
  Donc, la nouvelle balance en d est $bal'(d) = hC - (1 + hB) = j - 1$.

- Sinon, nouvelle $hauteur(r) = hA$.
  La nouvelle balance en d est donc :
  $bal'(d) = hC - (1 + hA) = (hC - hB) + (hB - hA - 1) = j + bal'(r) - 1$.

Mise à jour de la balance en cas de rotation gauche



Balance en r
i=1+Max(hB,hC) – hA

Balance en d
j=hC–hB

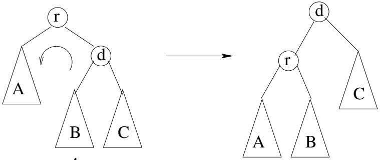
Nouvelle balance en r
hB–hA

Nouvelle balance en d
hC–Max(hA,hB) –1

Résumé des nouvelles balances :

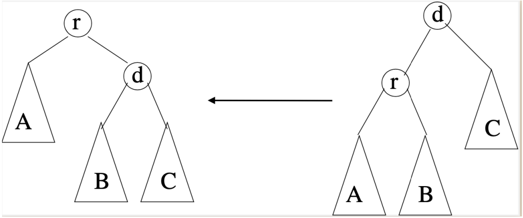
- Soit $i = bal(r)$ et $j = bal(d)$
- $bal'(r) = \begin{cases} i-1 & \text{, si } j \leq 0 \\ i-j-1 & \text{, sinon} \end{cases}$
- $bal'(d) = \begin{cases} j-1 & \text{, si } bal'(r) \geq 0 \\ j+bal'(r)-1 & \text{, sinon} \end{cases}$

Code de la rotation gauche




```
void RotationG (arbre *r){
    arbre d;  int i, j;
    /* Mise à jour des sous-arbres */
    d = (*r)->d; (*r)->d = d->g; d->g = *r;
    /* Mise à jour des balances */
    i = (*r)->bal; j = d->bal;
    if (j >= 0)
        (*r)->bal = i - j - 1;
    else
        (*r)->bal = i - 1;
    if ((*r)->bal <= 0)
        d->bal = j + (*r)->bal - 1;
    else
        d->bal = j - 1;
    /* Changement de racine de l'arbre obtenu */
    *r = d;
}
```

Mise à jour de la balance en cas de rotation droite



Formules donnant les nouvelles balances en fonction des anciennes :

- Soit $i = bal(r)$ et $j = bal(g)$
- $bal'(r) = \begin{cases} i+1 & \text{, si } j \leq 0 \\ i-j+1 & \text{, sinon} \end{cases}$
- $bal'(d) = \begin{cases} j+bal'(r)+1 & \text{, si } bal'(r) \geq 0 \\ j+1 & \text{, sinon} \end{cases}$

Fonction de rééquilibrage d'un arbre nouvellement déséquilibré

```
void Equilibrer (arbre *a){
    if ((*a)->bal == 2){
        if ((*a)->d->bal >= 0)
            RotationG(a);
        else
            RotationDG(a);
    }
    if ((*a)->bal == -2){
        if ((*a)->g->bal <= 0)
            RotationD(a);
        else
            RotationGD(a);
    }
}
```

Insertion dans un AVL

```
/* Sans gestion des problemes d'allocation */
/* Renvoie la participation au deseqilibre */
int Inserer(arbre *a, int i){
    int var;
    if (*a == NULL) {
        *a = CreerNoeud(i);
        return 1;
    } else if (i <= (*a)->etiquette)
        var = -Inserer(&(*a)->g,i);
    else
        var = Inserer(&(*a)->d,i);
    if (var == 0)
        return 0;
    else {
        (*a)->bal = (*a)->bal + var;
        Equilibrer(a);
        if ((*a)->bal == 0)  return 0;
        else  return 1;
    }
}
```