Déséquilibres, rotations et arbres AVL

4R-IN1 – ESIEE Paris

dernière modification 19 novembre 2026

Point de départ : ce que vous savez déjà

- Construire un arbre binaire de recherche (ABR) par insertion.
- Exploiter l'invariant : gauche $<$ racine $<$ droite.
- Rechercher en $O(h)$, où h est la hauteur de l'arbre.
- Comprendre le contraste : $h \in O(\log n)$ si l'arbre reste équilibré ; $h = n - 1$ dans le pire cas.

Question du jour

Comment reconnaître un déséquilibre, puis le corriger **sans changer l'ordre des clés** ?

Les arbres AVL

Un arbre AVL est un **arbre binaire de recherche (ABR)** qui reste **équilibré** au fil des insertions et des suppressions.

- Dans un AVL, **aucun nœud n'est déséquilibré**.
- Après chaque insertion ou suppression, l'arbre est **rééquilibré si nécessaire**.
- L'objectif est de conserver des opérations de recherche, d'insertion et de suppression efficaces : $O(\log n)$

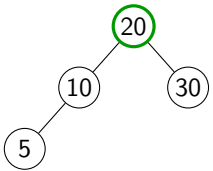
D'où vient le nom AVL ?

AVL est l'acronyme des noms de ses inventeurs : **Georgy Adelson-Velsky** et **Evgenii Landis**.
Cet arbre a été proposé en **1962** : c'est l'une des premières structures d'arbres de recherche auto-équilibrés.

Qu'est-ce qu'un déséquilibre ?

Définition

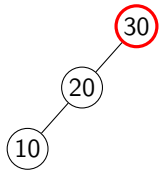
Un nœud est **déséquilibré** lorsque les hauteurs de ses sous-arbres gauche et droit diffèrent de plus de 1.



Hauteurs : gauche $= 1$, droite $= 0$

$$|h_g - h_d| = 1$$

Équilibré



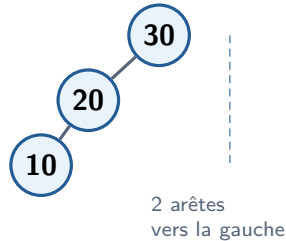
Hauteurs : gauche $= 1$, droite $= -1$

$$|h_g - h_d| = 2$$

Déséquilibré

D'où vient un déséquilibre ?

- Après une insertion **dans un arbre équilibré**, le sous arbre dont le parent direct est la racine ne peut pas être déséquilibré[1]
- En revanche en remontant d'un niveau supplémentaire, on peut trouver un sous arbre déséquilibré
- On cherche une correction **locale** qui préserve l'ordre des clés.



Question

Quels sont les différents cas de déséquilibre possibles ?

[1]on ajoute à gauche ou à droite d'une feuille, à gauche d'un noeud qui n'avait qu'un fils droit feuille ou à droite d'un noeud qui n'avait qu'un fils gauche feuille : le parent du nouveau noeud à donc 1 ou 2 fils tous deux des feuilles

Terminologie : les quatre *cas* de déséquilibre

On part du **premier nœud déséquilibré** A, rencontré en remontant après une insertion. Les deux premières arêtes du chemin vers le nœud inséré suffisent à classer le cas.

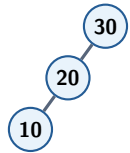
Cas	**Chemin à partir de A**	**Exemple d'insertions**
LL	gauche puis gauche	30, 20, 10
RR	droite puis droite	10, 20, 30
LR	gauche puis droite	30, 10, 20
RL	droite puis gauche	10, 30, 20

Distinction essentielle

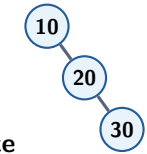
LL / RR / LR / RL nomment des *configurations de déséquilibre*, **pas** les directions des rotations.

Reconnaître visuellement les quatre formes

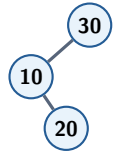
LL : gauche puis gauche



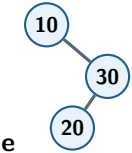
RR : droite puis droite



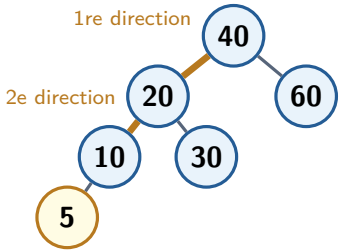
LR : gauche puis droite



RL : droite puis gauche



Pourquoi pas des cas LLL, RLR, RRLR ?



- Le chemin vers la feuille 5 est **LLL**.
- Pourtant le premier ancêtre déséquilibré est 40.
- Ses deux premières arêtes forment **LL** : on classe le cas LL.
- Les niveaux inférieurs sont préservés avec leurs **sous-arbres** lors de la correction.

Principe

Les chemins peuvent être longs ; la correction AVL repose sur une **configuration locale** de trois niveaux.

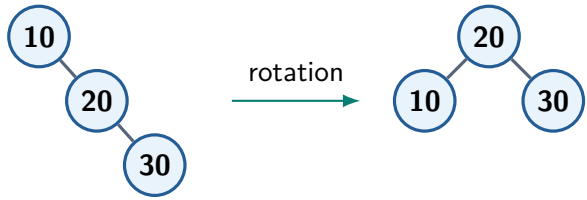
Qu'est-ce qu'une *rotation simple* ?

- C'est une transformation **locale** des liens entre un nœud A et son fils B.
- Le "pivot" B devient la racine du sous-arbre ; A descend d'un niveau.
- On parle de **rotation gauche** (L) ou **rotation droite** (R), d'après le mouvement de A : sens trigonométrique ou horaire.

Attention

Cas LL $\neq$ rotation gauche : le cas LL se corrige justement par une **rotation droite (R)**.

Pourquoi tourner ?



À justifier

Les deux arbres contiennent les mêmes valeurs et ont le même parcours infixe : 10, 20, 30.

Rotation droite : la structure générale



Conservation de l'ordre

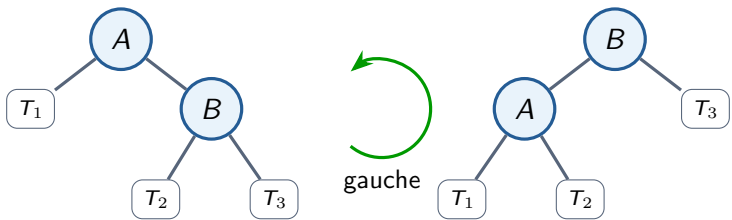
$T_1 < B < T_2 < A < T_3$: aucun élément ne change de côté dans l'ordre infixe.

Rotation droite : les quatre étapes

1. **Pivot :** $B \leftarrow A.\mathrm{ls}$.
2. **Transfert :** $A.\mathrm{ls} \leftarrow B.\mathrm{rs}$; le sous-arbre T_2 ne disparaît pas.
3. **Rebranchement :** $B.\mathrm{rs} \leftarrow A$.
4. **Racine locale :** le pointeur qui désignait A désigne maintenant B.

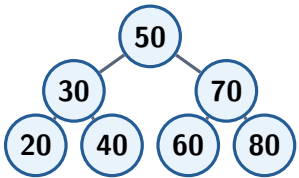
Démonstrateur : « Transfert de T_2 », puis sélectionner 40 et tourner à droite.

Rotation gauche : symétrique



- $B = A.\mathrm{rs}$; $A.\mathrm{rs} = B.\mathrm{ls}$; $B.\mathrm{ls} = A$; racine locale $= B$.
- Le sous-arbre T_2 est transféré **à droite de** A.

Une rotation n'agit pas nécessairement sur la racine globale



- Sélectionner **30**.
- Effectuer une rotation droite.
- La racine globale reste **50**.
- C'est le lien `50->ls` qui doit changer.

Démonstrateur

Bouton « Rotation interne », puis scénario ou rotation manuelle sur 30.

3. Qu'est-ce qu'une *rotation double* ?

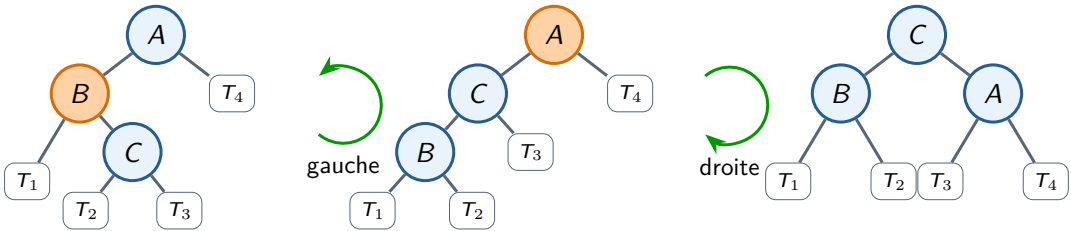
- Dans une forme en **zigzag** (LR ou RL), une seule rotation ne suffit pas.
- Une **rotation double** est la composition de deux rotations simples.
- On commence par **le fils**, puis on tourne autour de **l'ancien ancêtre déséquilibré**.

Cas	**Deux gestes**
LR	gauche sur le fils gauche puis droite sur la racine
RL	droite sur le fils droit puis gauche sur la racine

Pourquoi cet ordre ?

La première rotation transforme le zigzag en ligne ; la seconde réduit la hauteur du sous-arbre.

Cas LR : d'abord le fils, puis la racine

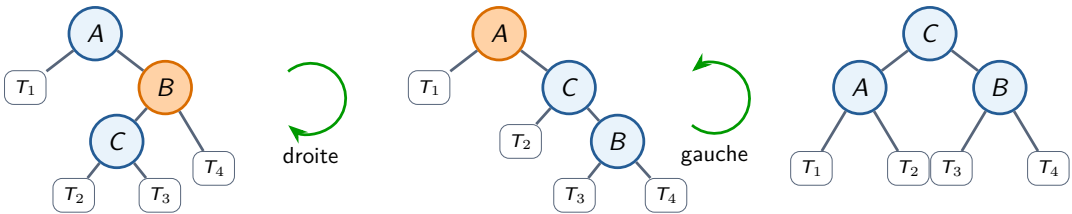


Principe

La première rotation transforme LR en LL. La seconde termine l'équilibrage.

Démonstrateur : « LR », puis deux fois « Étape suivante ».

Cas RL : le symétrique du cas LR



Démonstrateur : « RL » ; faire prédire le résultat après chaque rotation.

4. Retenir la *correspondance* cas / correction

Cas	**Forme**	**Opération(s)**	**Type**
LL	gauche–gauche	**R** : droite sur A	simple
RR	droite–droite	**L** : gauche sur A	simple
LR	gauche–droite	**L** sur fils gauche, puis **R** sur A	double
RL	droite–gauche	**R** sur fils droit, puis **L** sur A	double

Vérifier les correspondances avec les insertions

Cas	Insertion	Correction	Nature
LL	30, 20, 10	droite sur 30	simple
RR	10, 20, 30	gauche sur 10	simple
LR	30, 10, 20	gauche sur 10, droite sur 30	double
RL	10, 30, 20	droite sur 30, gauche sur 10	double

Question

Pourquoi une rotation simple ne suffit-elle pas dans les cas LR et RL ?

Petite activité : prédire avant de cliquer

1. Préparer le cas LR. Quelle rotation **ne faut-il pas** commencer par faire ?
2. Prévoir la racine après chacune des deux rotations.
3. Exécuter les étapes et vérifier le parcours infixe.
4. Recommencer en RL, sans consigne.

À retenir

La direction du deuxième niveau du chemin d'insertion détermine si la correction est simple ou double.

Rotation droite en C : traduction directe

```c
void rotate_right(BST *t)
{
    BSTNode *a, *b;
    if (t == NULL || *t == NULL || (*t)->ls == NULL)
        return;
    a = *t;
    b = a->ls;
    a->ls = b->rs;  /* transferer T2 */
    b->rs = a;      /* A devient fils droit */
    *t = b;         /* nouvelle racine locale */
}
```

Questions

Quel bug si on oublie la ligne de transfert ? Si on oublie `*t = b` ?

Rotation gauche en C : le symétrique

```c
void rotate_left(BST *t)
{
    BSTNode *a, *b;
    if (t == NULL || *t == NULL || (*t)->rs == NULL)
        return;
    a = *t;
    b = a->rs;
    a->rs = b->ls;  /* transferer T2 */
    b->ls = a;      /* A devient fils gauche */
    *t = b;         /* nouvelle racine locale */
}
```

Lien avec le TP3

Comme pour une tête de liste, modifier la racine d'un sous-arbre exige l'adresse du pointeur concerné.

Invariants d'une rotation

- Aucun nœud créé, détruit ou dupliqué.
- L'ordre infixe reste **identique**.
- La propriété ABR est conservée.
- Le coût est constant : $O(1)$ modifications de liens.
- La **hauteur** et la racine **locale** peuvent changer.

Question

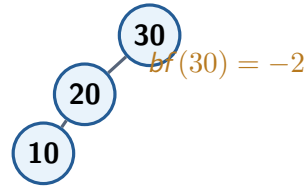
Une rotation rend-elle **toujours** un ABR plus équilibré ?

Le facteur d'équilibrage

- Pour automatiser, il faut mesurer le déséquilibre **sans reconnaître visuellement** un dessin.
- Convention du TP4 :

$$\boxed{bf(A) = h(A_{\text{droite}}) - h(A_{\text{gauche}})}$$

- $bf < 0$: plus haut à gauche ; $bf > 0$: plus haut à droite.
- AVL équilibré : $bf \in \{-1, 0, +1\}$.
- Après insertion, $bf(A) = \pm 2$ déclenche une correction.



Définition formelle d'un AVL

Un AVL est un ABR tel que, pour chaque nœud A :

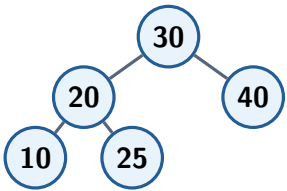
$$|h(A_{\text{droite}}) - h(A_{\text{gauche}})| \leq 1.$$

Convention du TP

$$\boxed{bf(A) = h(A_{\text{droite}}) - h(A_{\text{gauche}})}$$

- $bf = -1$: côté gauche plus haut ; $bf = +1$: côté droit plus haut.
- $bf = \pm 2$: déséquilibre temporaire à corriger.

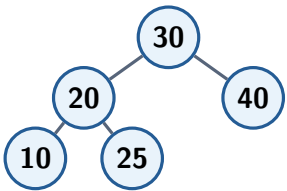
Calculer les facteurs d'équilibrage



1. Feuille : $bf = 0$.
2. $bf(20) = ?$.
3. $bf(30) = ?$.
4. Et si l'on supprime 40 ?

Réponses :

Calculer les facteurs d'équilibrage

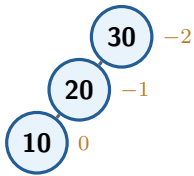


1. Feuille : $bf = 0$.
2. $bf(20) = ?$.
3. $bf(30) = ?$.
4. Et si l'on supprime 40 ?

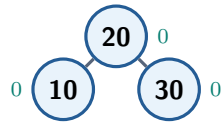
Réponses : $bf(20) = 0$; $bf(30) = -1$; après suppression de 40, $bf(30) = -2$.

Effet sur les *bf* : rotation simple après insertion

Cas LL : avant



Après rotation droite



Ce qui change

Ce n'est pas seulement le *bf* de l'ancienne racine : les facteurs des **deux nœuds impliqués** doivent être recalculés (ou mis à jour par formule).

Facteurs après rotation : pas toujours zéro

Insertion RR : insérer 10, 20, 30.

- Avant rotation : $bf(10) = +2$, $bf(20) = +1$.
- Après rotation gauche : $bf(10) = 0$, $bf(20) = 0$.

Contre-exemple après suppression :

- Insérer 4, 2, 5, 1, 3 puis retirer 5.
- Avant rotation : $bf(4) = -2$, $bf(2) = 0$.
- Après rotation droite : $bf(4) = -1$, $bf(2) = +1$.

Démonstrateur

Boutons « RR » et « Facteur après suppression » ; afficher les *bf*.

Effet sur les *bf* : les deux gestes du cas LR

	Avant	**Après L sur 10**	**Après R sur 30**
Forme	zigzag LR	ligne LL	équilibrée
$bf(30)$	-2	-2	0
$bf(10)$	$+1$	0	0
$bf(20)$	0	-1	0

- La première rotation **prépare** la correction ; l'ancien ancêtre peut encore avoir $bf = -2$.
- La seconde rotation restaure l'équilibre.
- En C, chaque rotation met à jour les facteurs de ses deux nœuds concernés.

Démonstrateur

« Insertion AVL · LR » avec affichage des *bf* et journal des opérations.

Rotation gauche : retrouver les nouveaux facteurs

Avec $B = A.\mathrm{rs}$ et $bf = h(\text{droite}) - h(\text{gauche})$:

$$\boxed{bf'(A) = bf(A) - 1 - \max(bf(B), 0)}$$

$$\boxed{bf'(B) = bf(B) - 1 + \min(bf'(A), 0)}$$

Tester sur deux exemples :

- $bf(A) = +2$, $bf(B) = +1$: $bf'(A) = 0$, $bf'(B) = 0$.
- $bf(A) = +2$, $bf(B) = 0$: $bf'(A) = +1$, $bf'(B) = -1$.

Rotation droite : les formules symétriques

Avec $B = A.\mathrm{ls}$ et $bf = h(\mathrm{droite}) - h(\mathrm{gauche})$:

$$\boxed{bf'(A) = bf(A) + 1 - \min(bf(B), 0)}$$

$$\boxed{bf'(B) = bf(B) + 1 + \max(bf'(A), 0)}$$

De *bf* à la décision : trois questions

1. **Le nœud A est-il déséquilibré ?** $bf(A) = -2$ ou $bf(A) = +2$?
2. **Quel fils B est du côté trop haut ?** $A.\mathrm{ls}$ si -2, $A.\mathrm{rs}$ si $+2$.
3. **Le côté trop haut de B est-il externe ou interne ?** On lit le signe de $bf(B)$.

Test en A	**Test en B**	**Décision**
$bf(A) = -2$	$bf(B) \leq 0$	cas LL $\rightarrow$ droite sur A
$bf(A) = -2$	$bf(B) > 0$	cas LR $\rightarrow$ gauche sur B, droite sur A
$bf(A) = +2$	$bf(B) \geq 0$	cas RR $\rightarrow$ gauche sur A
$bf(A) = +2$	$bf(B) < 0$	cas RL $\rightarrow$ droite sur B, gauche sur A

Attention : le cas $bf(B) = 0$ peut apparaître après une **suppression** ; il se traite par la rotation simple du côté trop haut.

Algorithme de décision : lire les facteurs avant d'agir

1. En remontant d'une insertion, mettre à jour $bf(A)$.
2. Si $-1 \leq bf(A) \leq +1$: **aucune rotation** à ce nœud.
3. Si $bf(A) = -2$: regarder $bf(A.\mathrm{ls})$; **LR** si positif, sinon **LL**.
4. Si $bf(A) = +2$: regarder $bf(A.\mathrm{rs})$; **RL** si négatif, sinon **RR**.
5. Effectuer la ou les rotations ; mettre à jour les bf des nœuds déplacés.

Exemple de journal (insertion de 30, 10, 20)

```
bf(30)=-2 : trop haut à gauche.
bf(10)=+1 : le fils gauche penche à droite.
Cas LR : gauche sur 10, puis droite sur 30.
Mettre à jour bf(10), bf(20), bf(30).
```

balance(AVL *t) : squelette de décision

```
if ((*t)->bf == -2) {
    if ((*t)->ls->bf > 0)
        rotate_left(&(*t)->ls);
    rotate_right(t);
} else if ((*t)->bf == +2) {
    if ((*t)->rs->bf < 0)
        rotate_right(&(*t)->rs);
    rotate_left(t);
}
```

- Les rotations de l'AVL mettent également à jour les facteurs.
- `t` est toujours l'adresse de la racine *locale*.
- Le fils de facteur zéro peut apparaître après une suppression.

Pourquoi la correction peut être automatique

Insertion ABR

↓

Calcul des changements de hauteur

↓

$bf = \pm 2$ **?**

↓ si oui

Choisir rotation simple ou double

Question

Comment reconnaître une configuration LL, RR, LR ou RL *sans regarder le dessin* ?

Insertion AVL : la même descente, une autre remontée

1. Descendre dans l'ABR pour trouver la place de la nouvelle valeur.
2. Insérer le nouveau nœud, avec $bf = 0$.
3. Remonter récursivement et mettre à jour les facteurs.
4. Corriger le premier ancêtre déséquilibré si nécessaire.
5. Remonter uniquement si la hauteur du sous-arbre a augmenté.

Démonstrateur : « Insertion AVL · LR », puis « Étape suivante » ; observer les facteurs avant les rotations.

Insertion AVL : propagation de hauteur

Modification locale	**Résultat**
bf passe de 0 à -1 ou $+1$	Hauteur augmente : remonter.
bf passe de ± 1 à 0	Hauteur inchangée : arrêter.
bf atteint ± 2	Rotation(s) ; hauteur restaurée pour l'insertion AVL.

Contrat du TP : `add(...)` renvoie 1 si la hauteur a augmenté, 0 sinon (doublon compris), -1 en cas d'échec d'allocation.

Insertion AVL : scène commentée

1. Insérer 30 : racine, $bf = 0$.
2. Insérer 10 : $bf(30) = -1$, pas de rotation.
3. Insérer 20 : $bf(10) = +1$ et $bf(30) = -2$.
4. Rotation **gauche sur 10**.
5. Rotation **droite sur 30**.

Question

Pourquoi une rotation simple sur 30 ne suffit-elle pas à l'étape 3 ?

Démonstrateur : « Insertion AVL · LR » ; les cinq étapes correspondent exactement à cette liste.

Suppression AVL : prolongement facultatif

- Les mêmes rotations sont utiles après une suppression.
- La hauteur d'un sous-arbre peut **diminuer**.
- Plusieurs rééquilibrages peuvent alors être nécessaires jusqu'à la racine.
- Le cas du fils de facteur zéro est notamment important.

Conformément au TP4

L'implémentation C de la suppression AVL est **facultative**. Nous nous concentrons ici sur les rotations et l'insertion.

Synthèse : cinq questions

1. Qu'est-ce qui prouve qu'une rotation conserve l'invariant ABR ?
2. Où va le sous-arbre intermédiaire T_2 ?
3. Dans quel ordre effectuer les rotations LR ?
4. Que signifie $bf = -2$ avec notre convention ?
5. Pourquoi la recherche dans un AVL est-elle en $O(\log n)$?

Trois idées à retenir : conserver l'ordre ; réorganiser les liens ; décider avec les facteurs.

Pour réviser et expérimenter

- Rejouer manuellement LL, RR, LR, RL.
- Sur « Transfert de T_2 », expliquer le devenir de chaque lien.
- Sur « Rotation interne », identifier le `BST *t` modifié.
- Générer un ABR aléatoire (10–15 nœuds), repérer les facteurs non AVL et essayer des rotations.
- Tester l'insertion AVL automatique depuis un arbre vide.

Rappel

Une rotation locale préserve l'ABR, mais seule une stratégie systématique d'équilibrage maintient les garanties d'un AVL.

Insertion AVL : combien de rotations ?

Propriété fondamentale

Pour **une insertion** dans un arbre initialement AVL, on corrige uniquement le **premier ancêtre déséquilibré** rencontré en remontant (depuis le nouveau nœud).

Cas détecté	**Correction**	**Coût**
LL	rotation droite	1 rotation
RR	rotation gauche	1 rotation
LR	gauche, puis droite	2 rotations
RL	droite, puis gauche	2 rotations

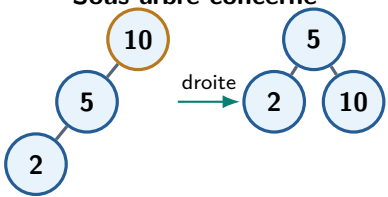
Pourquoi s'arrêter ?

Après cette correction, le sous-arbre retrouve **la hauteur qu'il avait avant l'insertion**. Les ancêtres au-dessus ne changent finalement ni de hauteur ni de facteur d'équilibrage.

Une seule correction locale, même loin de la racine

À tester : insérer successivement `50, 20, 80, 10, 30, 70, 90, 5, 2` en mode AVL.

Sous-arbre concerné



Évolution des facteurs

Nœud	Avant 2	Après 2	Corrigé
5	0	-1	0
10	-1	-2	0
20	-1	-2	-1
50	-1	-2	-1

Observation

Les nœuds 20 et 50 *semblent* aussi déséquilibrés juste après l'insertion ; **la seule rotation droite sur 10** suffit à les corriger.

Remontée des facteurs : quand s'arrêter ?

Convention : $bf(A) = h(\text{droit}) - h(\text{gauche})$. On ne remonte que si la hauteur du sous-arbre vient d'augmenter.

Évolution de $bf(A)$	**Hauteur de** A	**Action**
$0 \to \pm 1$	augmente de 1	**Continuer** vers le parent
$\pm 1 \to 0$	inchangée	**Arrêter**
$\pm 1 \to \pm 2$ (même signe)	augmente temporairement	**Corriger**, puis arrêter

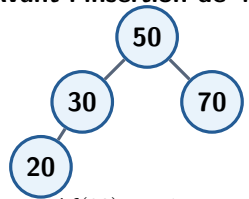
Attention

Cette règle d'arrêt est propre à l'**insertion AVL**. Après une suppression, une rotation peut réduire la hauteur et imposer une remontée supplémentaire.

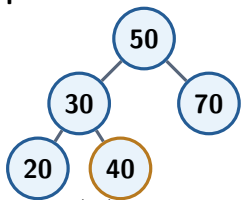
Exemple : aucun besoin de remonter jusqu'à la racine

À tester : insérer 50, 30, 70, 20, puis 40.

Avant l'insertion de 40



Après l'insertion de 40

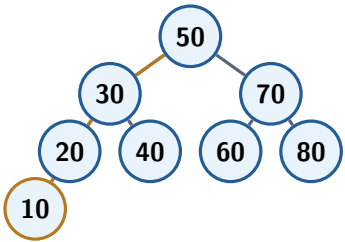


Conclusion

$bf(30)$ passe de -1 à 0 : la hauteur du sous-arbre enraciné en 30 **ne change pas**. Donc $bf(50)$ reste -1 ; aucune rotation, et la mise à jour peut s'arrêter à 30.

Exemple : la remontée va jusqu'à la racine

À tester : insérer 50, 30, 70, 20, 40, 60, 80, puis 10.



Nœud	*bf* avant 10	*bf* après 10
20	0	-1
30	0	-1
50	0	-1

Chaque sous-arbre gagne un niveau : il faut **propager les mises à jour jusqu'à 50**.

Mais aucune rotation n'est nécessaire

Tous les facteurs restent dans $\{-1, 0, +1\}$.

Insertion AVL : stratégie optimisée de remontée

Après insertion de la feuille (comme dans un ABR), on traite les ancêtres du bas vers le haut, **tant que la hauteur a augmenté**.

1. Mettre à jour $bf(A)$ de -1 si la branche gauche a grandi, de $+1$ si c'est la branche droite.
2. Si $bf(A) = 0$: **arrêter** (la hauteur de A n'a pas grandi).
3. Si $|bf(A)| = 1$: **continuer** vers le parent.
4. Si $|bf(A)| = 2$: identifier LL/RR/LR/RL, effectuer **une rotation simple ou double**, mettre à jour les facteurs des nœuds tournés, puis **arrêter**.

Complexité

Au plus $O(\log n)$ ancêtres sont examinés ; au plus **deux rotations élémentaires** sont effectuées, soit $O(1)$ pour la restructuration.