Arbres AVL

Algorithmique et structures de données, 2022-2023

P. Albuquerque (B410), P. Künzli et O. Malaspinas (A401), ISC, HEPIA

2023-03-31

En partie inspirés des supports de cours de P. Albuquerque

Questions sur les notions du dernier cours

- Qu'est-ce qu'un arbre AVL?

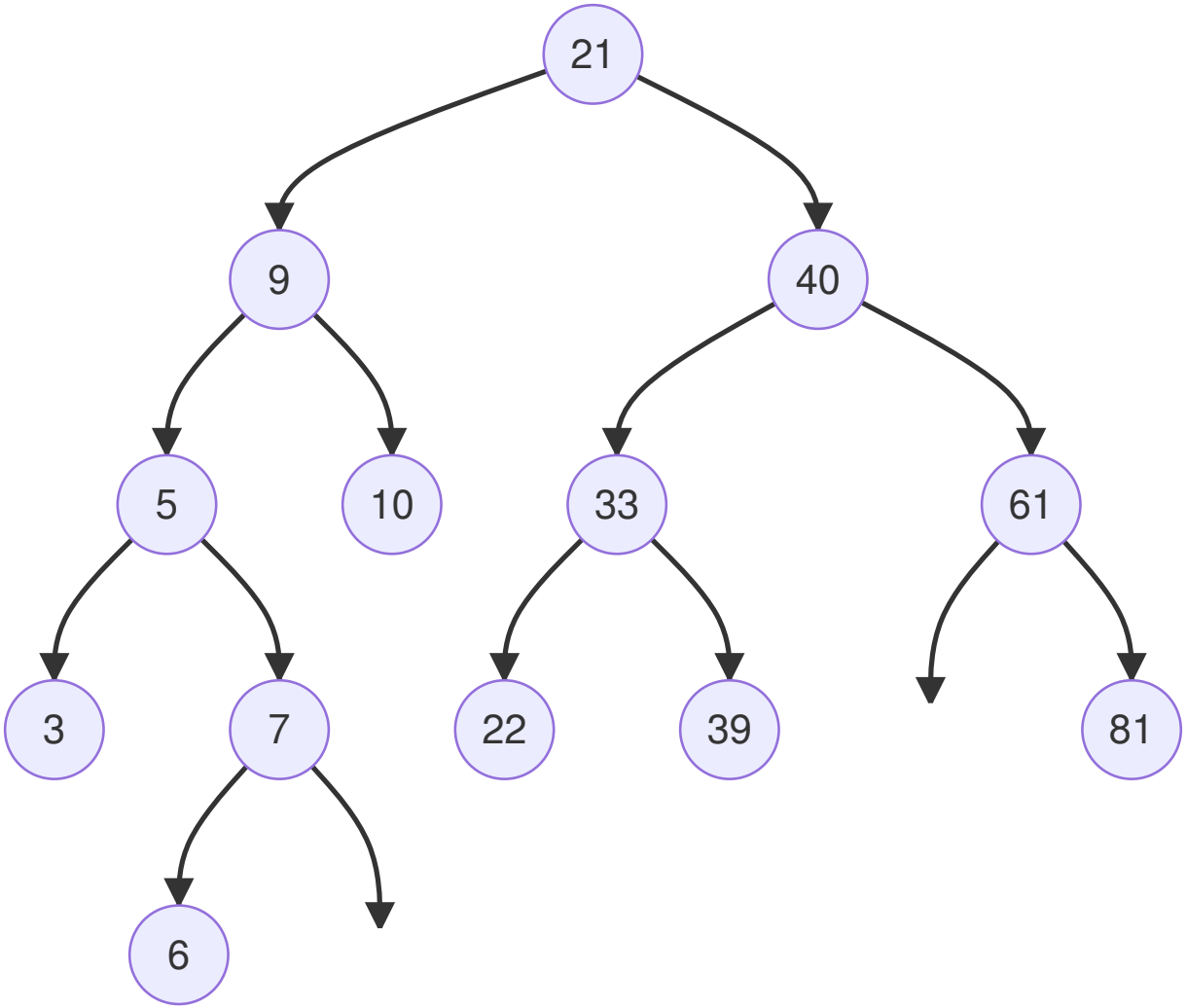
Questions sur les notions du dernier cours

- Qu'est-ce qu'un arbre AVL?
- Un arbre binaire qui a la propriété suivante:
  - La différence de hauteur de chaque noeud est d'au plus 1.
  - Tous les noeuds ont `fe = hd - hg = {-1, 0, 1}`.
- Pourquoi utiliser un arbre AVL plutôt qu'un arbre binaire de recherche?

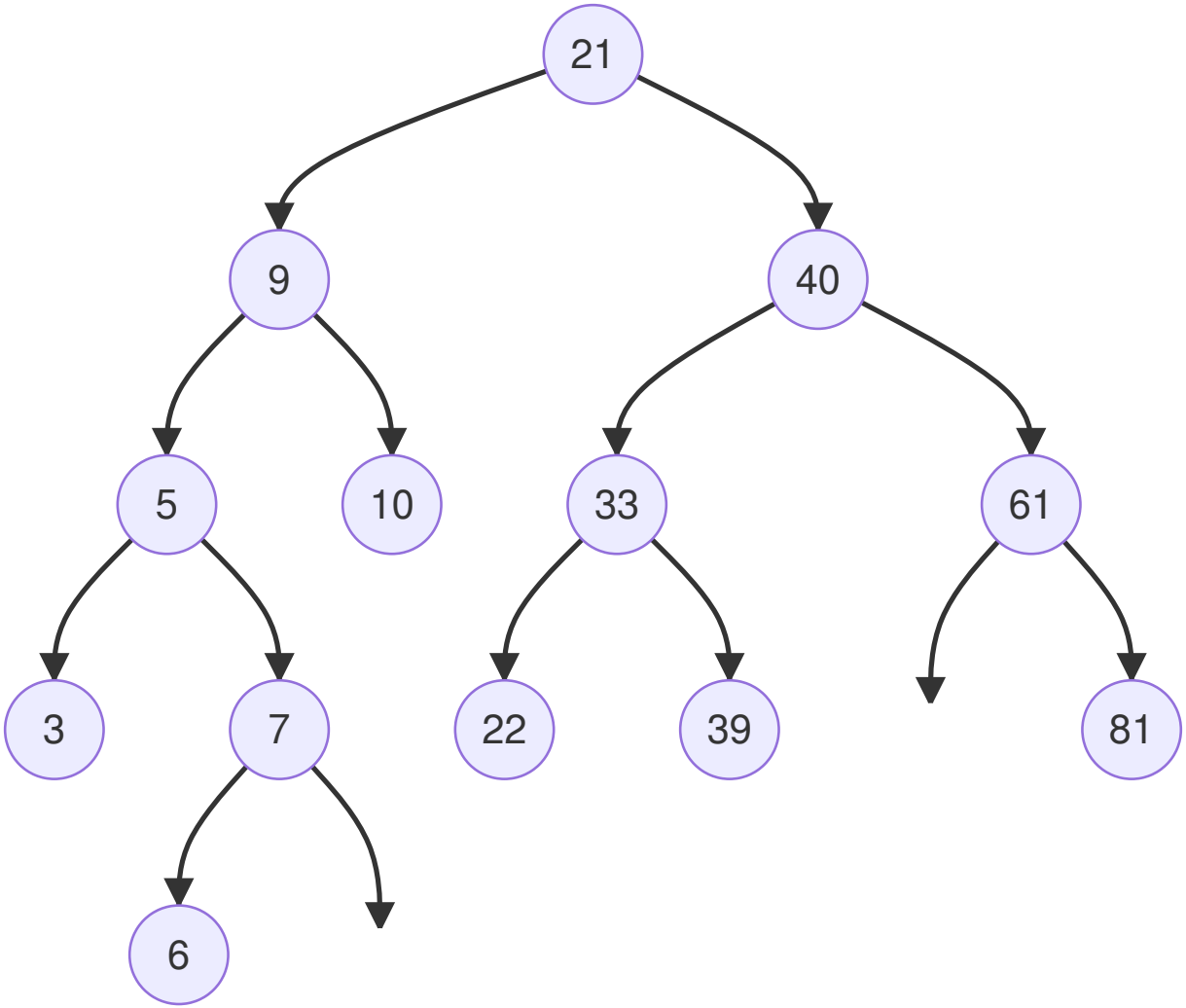
Questions sur les notions du dernier cours

- Qu'est-ce qu'un arbre AVL?
- Un arbre binaire qui a la propriété suivante:
  - La différence de hauteur de chaque noeud est d'au plus 1.
  - Tous les noeuds ont `fe = hd - hg = {-1, 0, 1}`.
- Pourquoi utiliser un arbre AVL plutôt qu'un arbre binaire de recherche?
- Insertion/recherche/... toujours en $O(\log_2(N))$.

AVL ou pas?



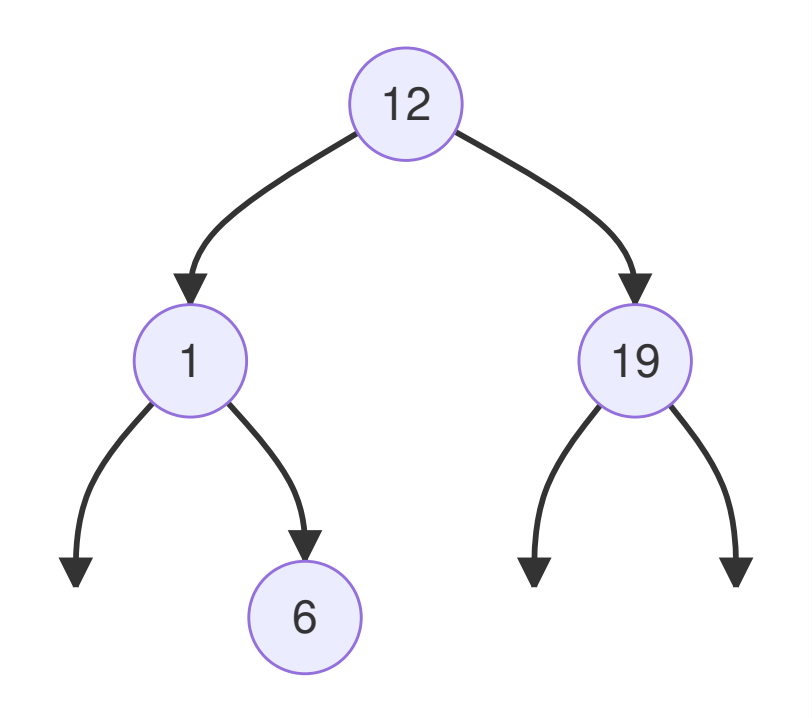
AVL ou pas?



Insertion dans un arbre AVL

1. On part d'un arbre AVL.
2. On insère un nouvel élément.

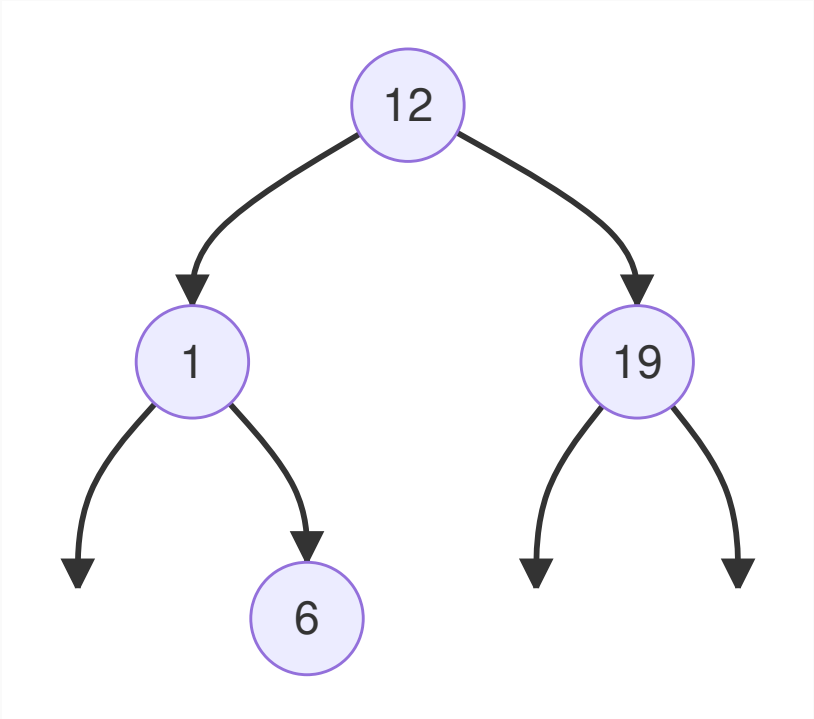
- `hd ? hg`.
- Insertion de `4`?



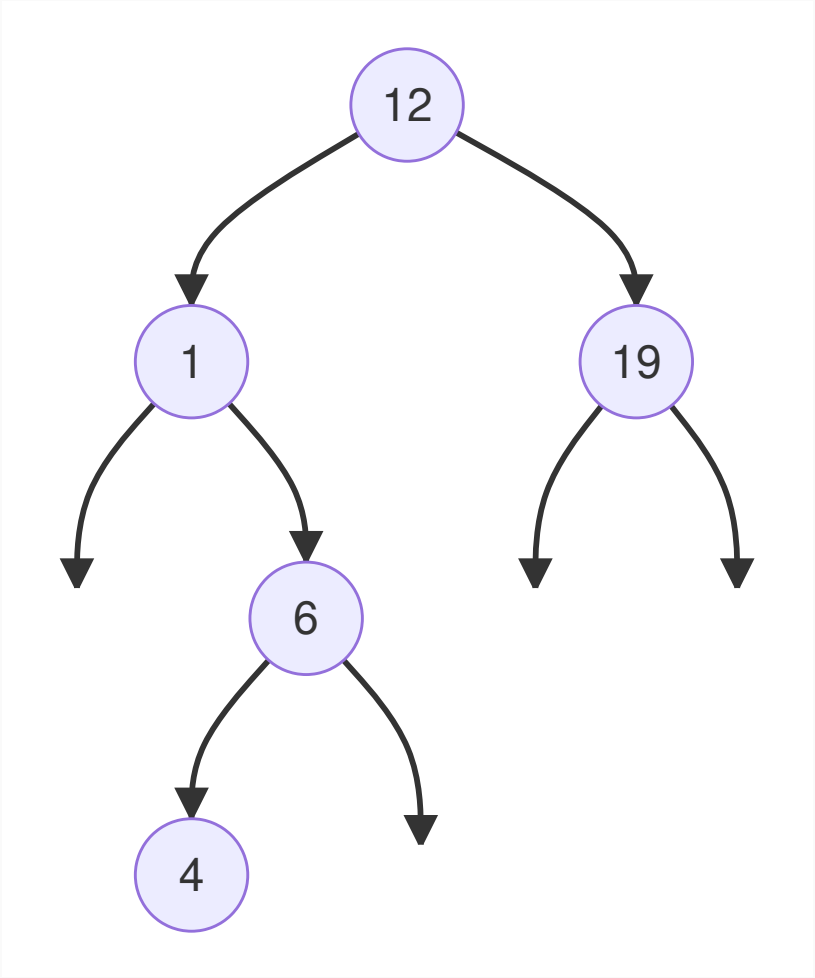
Insertion dans un arbre AVL

1. On part d'un arbre AVL.
2. On insère un nouvel élément.

- `hd ? hg`.
- Insertion de 4?



- `hd > hg`

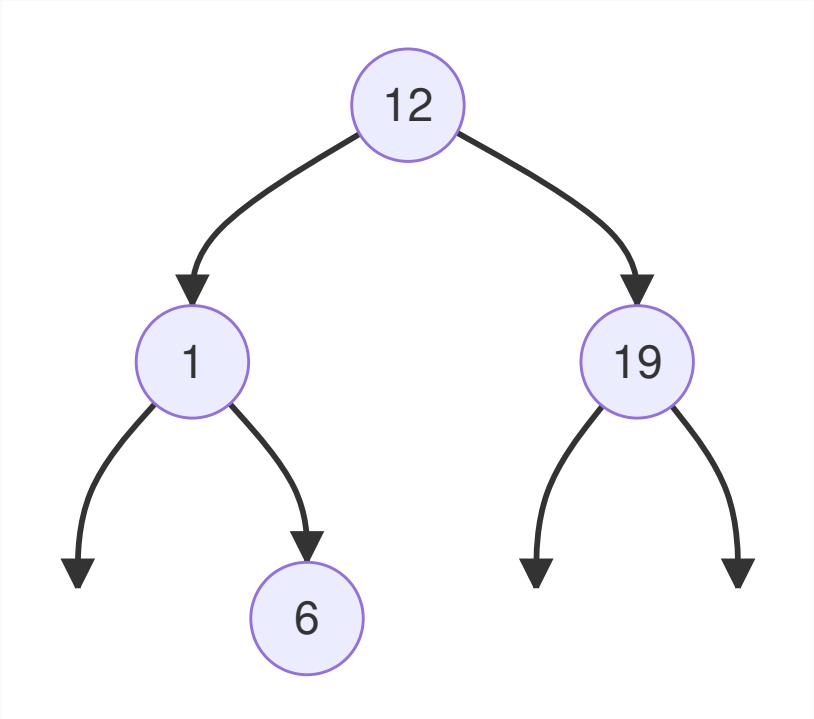


Déséquilibre! Que vaut `fe`?

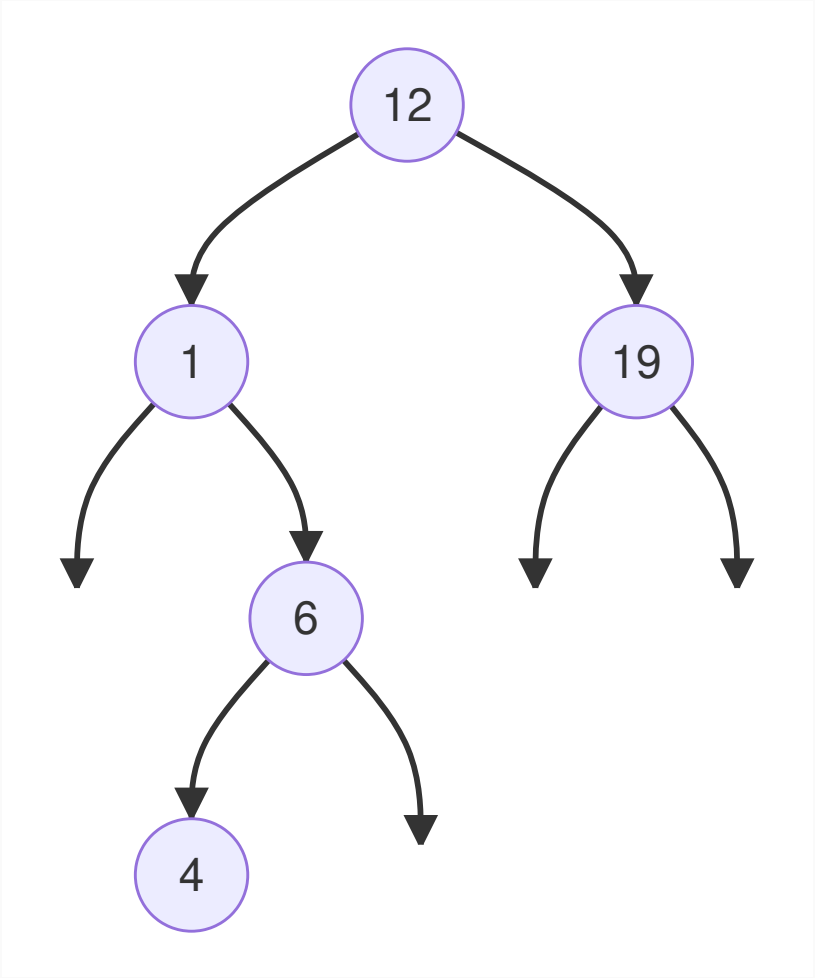
Insertion dans un arbre AVL

1. On part d'un arbre AVL.
2. On insère un nouvel élément.

- `hd ? hg`.
- Insertion de 4?



- `hd > hg`



Déséquilibre! Que vaut `fe`?

Les cas de déséquilibre

Cas 1a

- `u`, `v`, `w` même hauteur.
- déséquilibre en `B` après insertion dans `u`

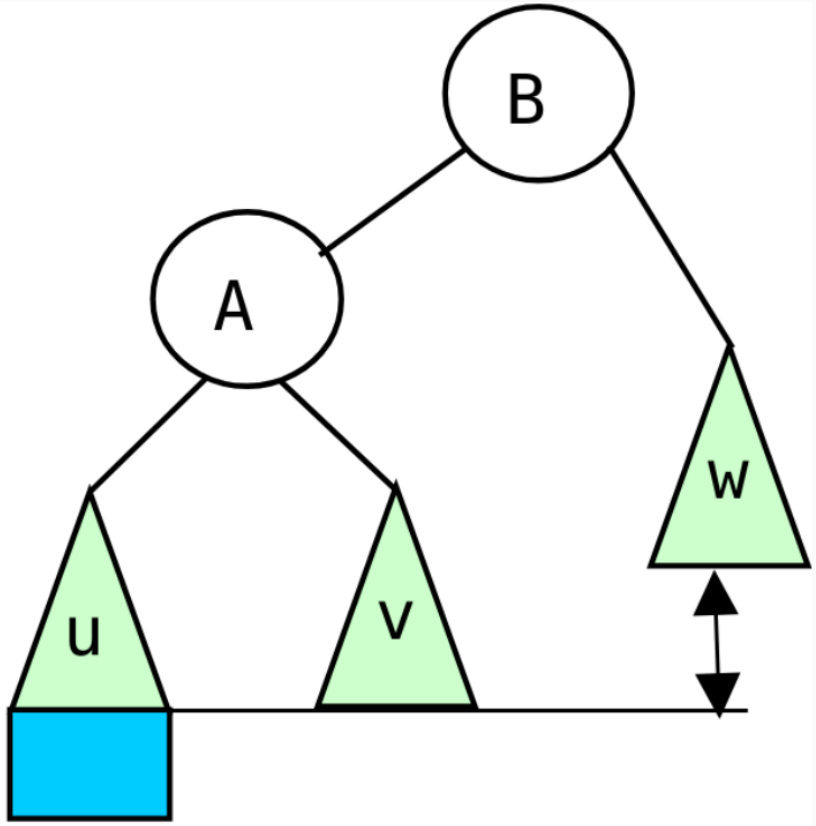


Figure 1: Après insertion

Cas 1a

- Comment rééquilibrer?

Les cas de déséquilibre

Cas 1a

- `u`, `v`, `w` même hauteur.
- déséquilibre en `B` après insertion dans `u`

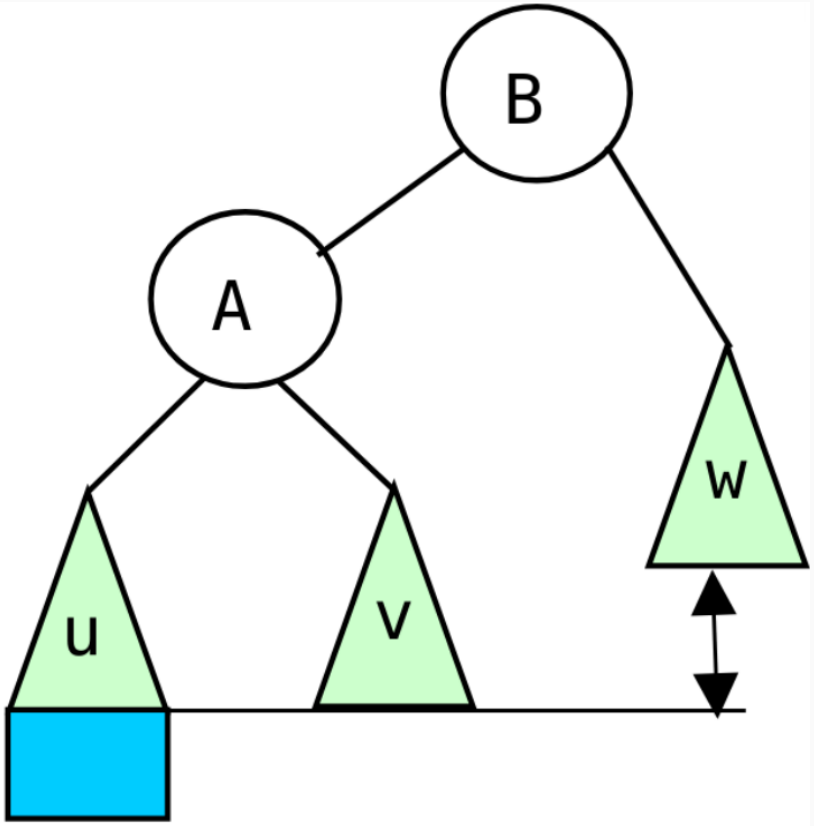


Figure 1: Après insertion

Cas 1a

- Comment rééquilibrer?
- ramène `u`, `v` `w` à la même hauteur.
- `v` à droite de `A` (gauche de `B`)

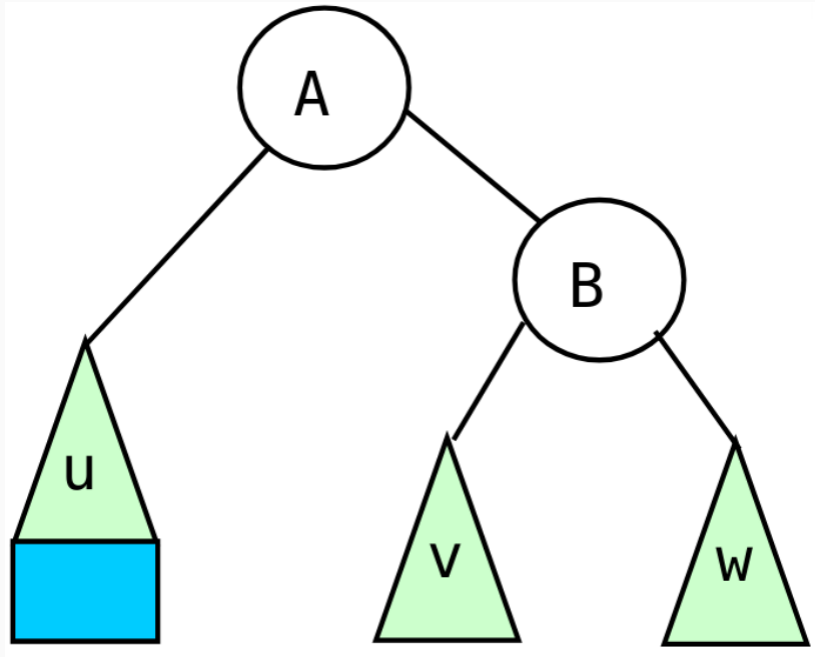


Figure 2: Après équilibrage

Les cas de déséquilibre

Cas 1b (symétrique 1a)

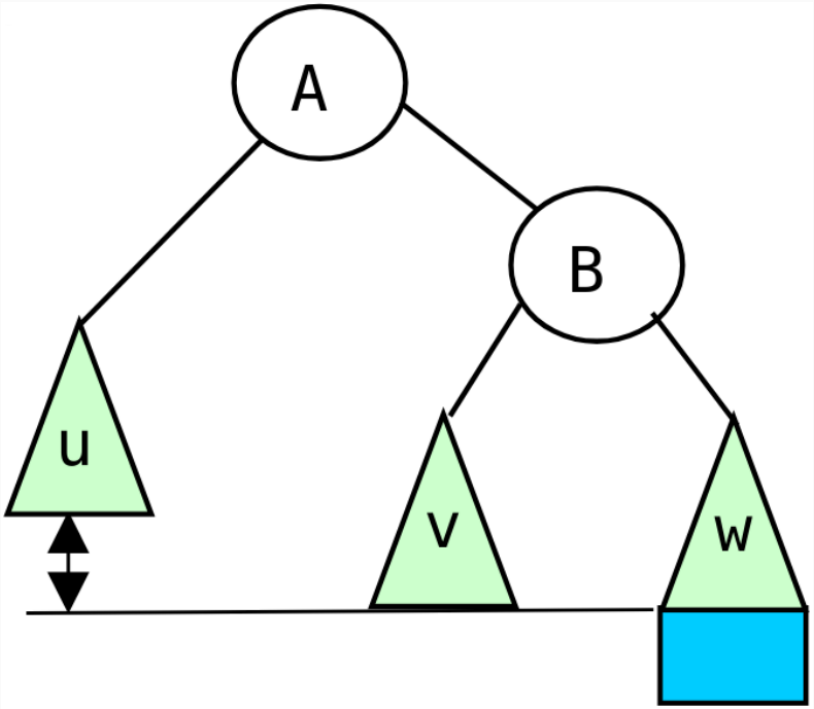


Figure 3: Après insertion

Cas 1b (symétrique 1a)

- Comment rééquilibrer?

Les cas de déséquilibre

Cas 1b (symétrique 1a)

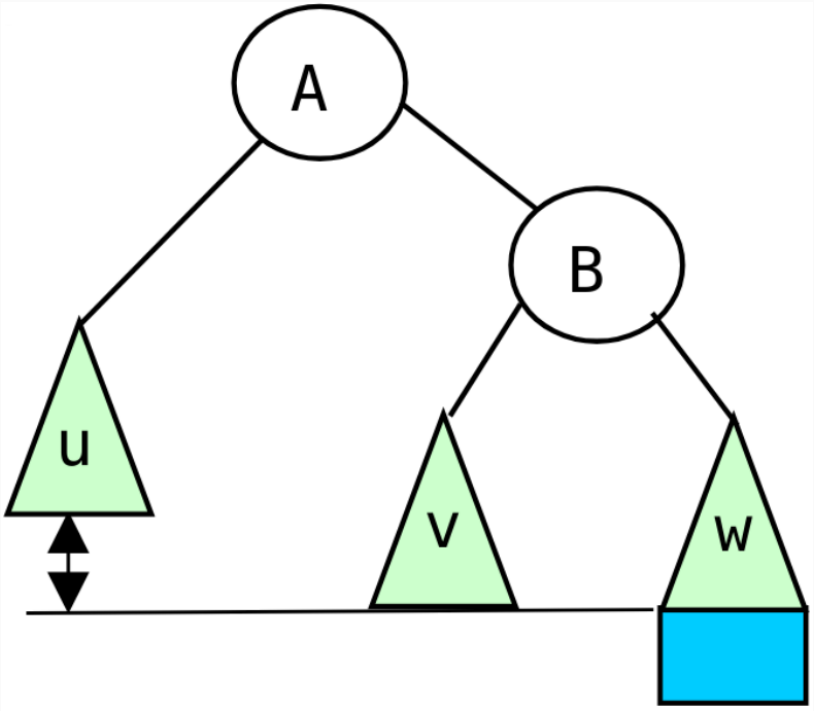


Figure 3: Après insertion

Cas 1b (symétrique 1a)

- Comment rééquilibrer?

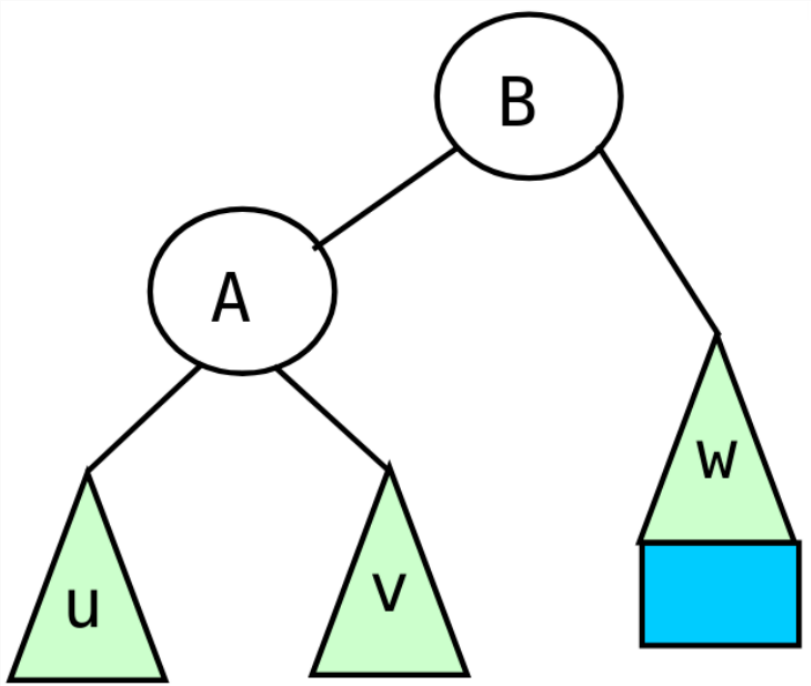
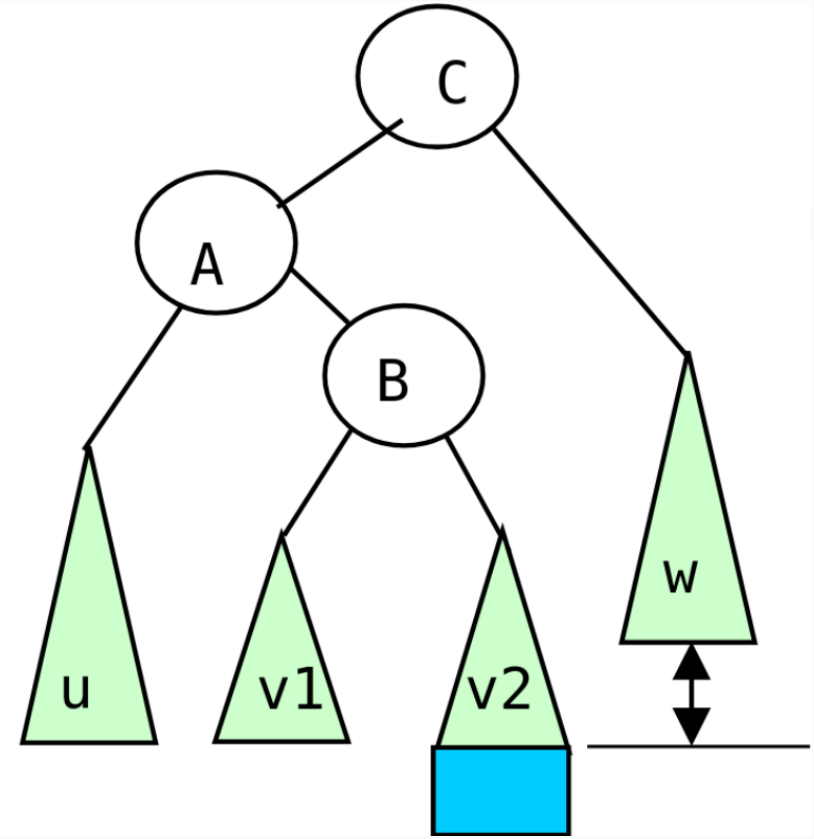


Figure 4: Après équilibrage

Les cas de déséquilibre

Cas 2a

- `h(v1)=h(v2)`, `h(u)=h(w)`.
- déséquilibre en `C` après insertion dans `v2`



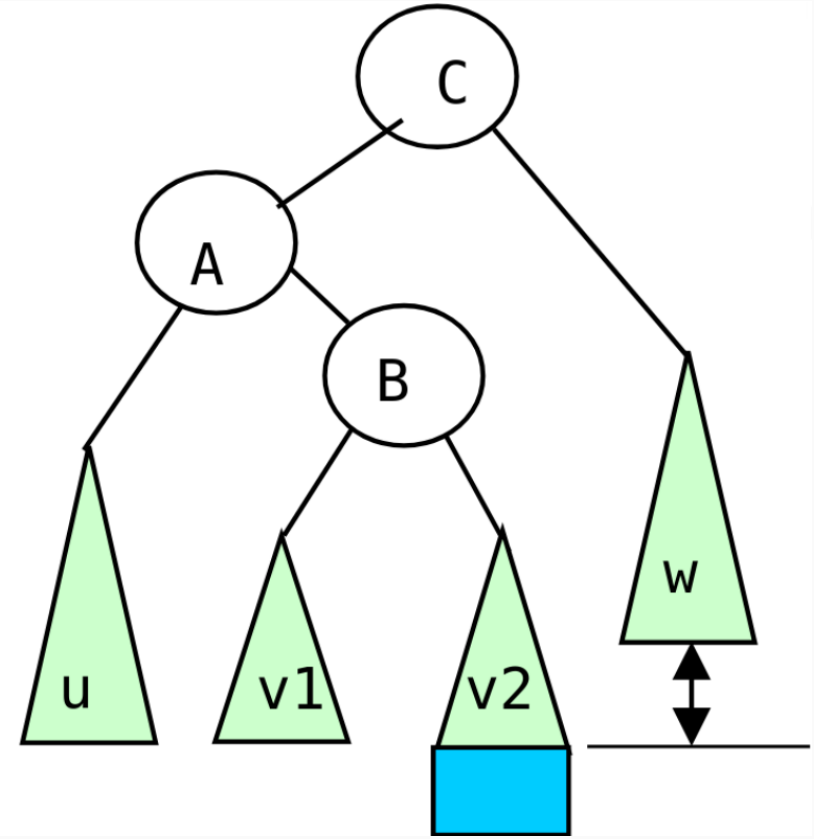
Cas 2a

- Comment rééquilibrer?

Les cas de déséquilibre

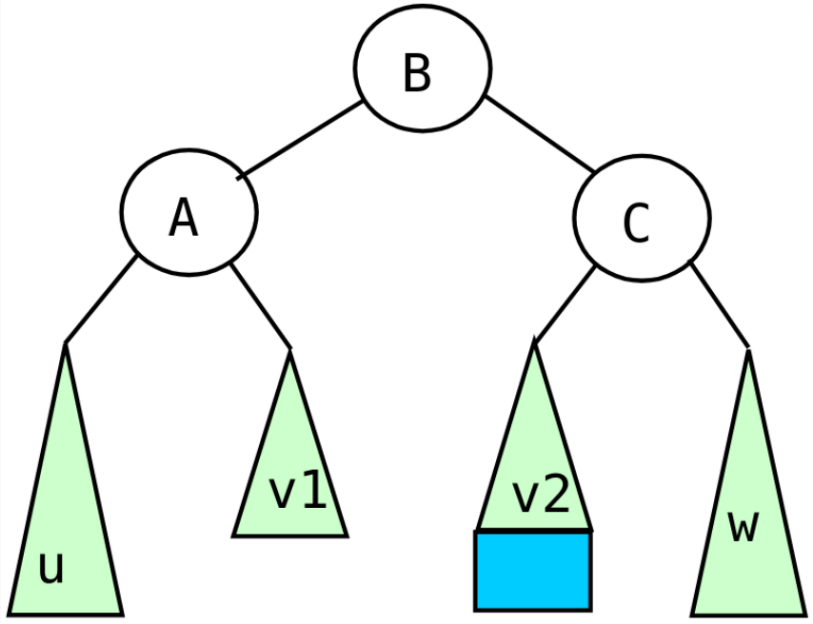
Cas 2a

- `h(v1)=h(v2)`, `h(u)=h(w)`.
- déséquilibre en `C` après insertion dans `v2`



Cas 2a

- Comment rééquilibrer?
- ramène `u`, `v2`, `w` à la même hauteur (`v1` pas tout à fait).
- `v2` à droite de `B` (gauche de `C`)
- `B` à droite de `A` (gauche de `C`)
- `v1` à droite de `A` (gauche de `B`)



Les cas de déséquilibre

Cas 2b (symétrique 2a)

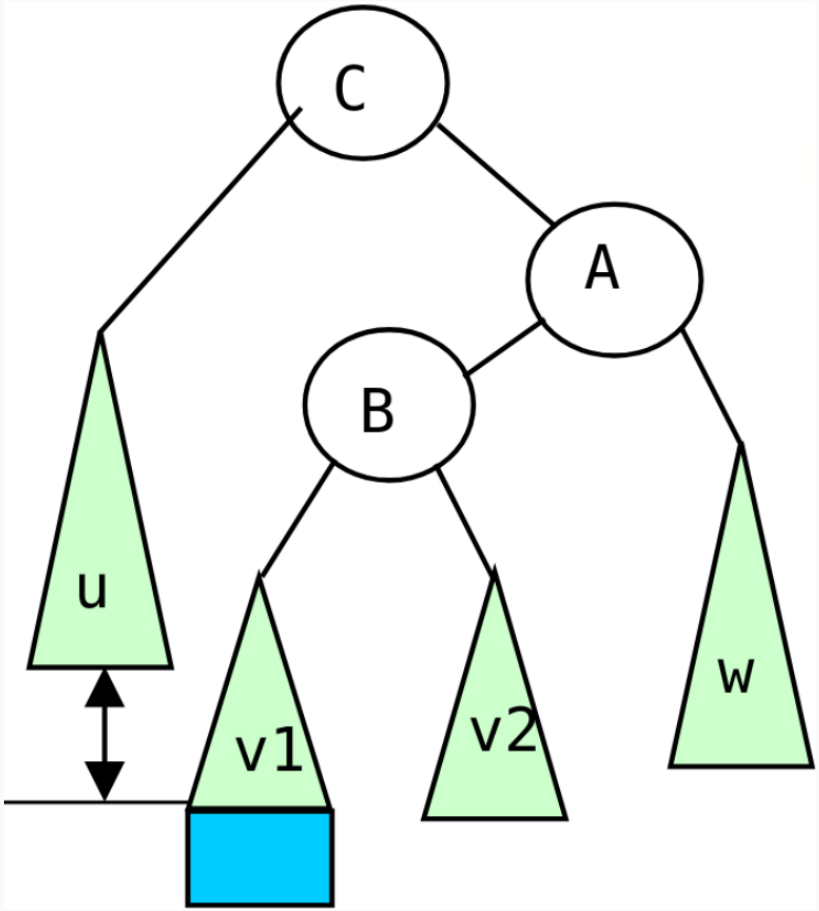


Figure 7: Après insertion

Cas 2b (symétrique 2a)

- Comment rééquilibrer?

Les cas de déséquilibre

Cas 2b (symétrique 2a)

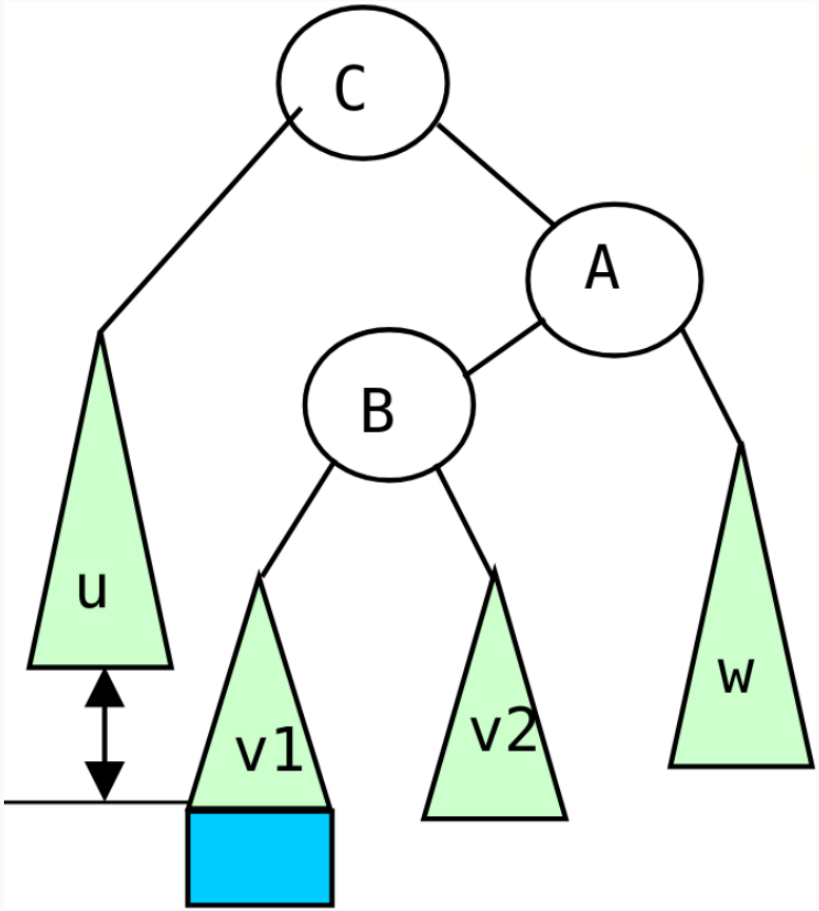


Figure 7: Après insertion

Cas 2b (symétrique 2a)

- Comment rééquilibrer?

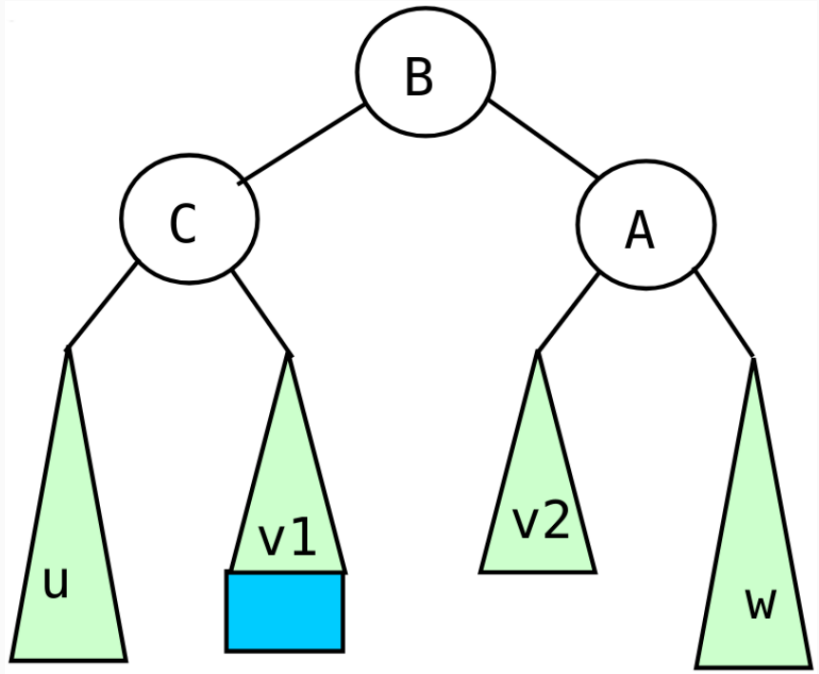


Figure 8: Après équilibrage

Le facteur d'équilibre (balance factor)

Définition

```
fe(arbre) = hauteur(droite(arbre)) - hauteur(gauche(arbre))
```

Valeurs possibles?

Le facteur d'équilibre (balance factor)

Définition

```
fe(arbre) = hauteur(droite(arbre)) - hauteur(gauche(arbre))
```

Valeurs possibles?

```
fe = {-1, 0, 1} // arbre AVL
fe = {-2, 2}    // arbre déséquilibré
```

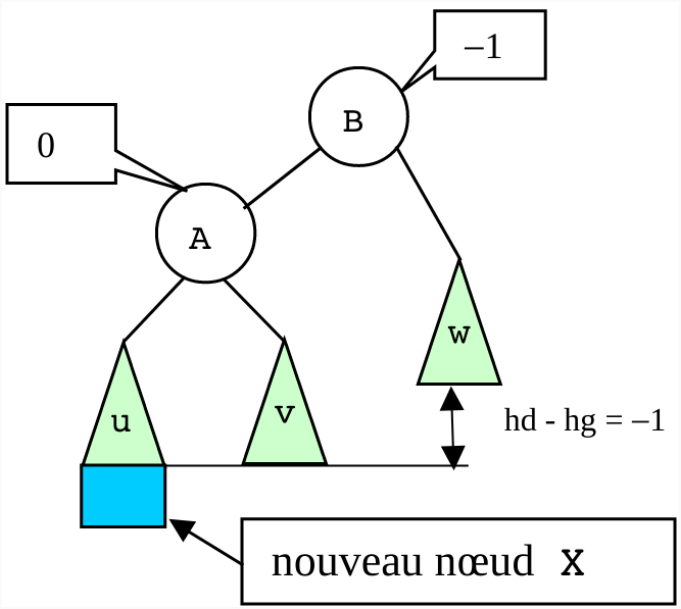


Figure 9: Illustration du `fe`

Algorithme d'insertion

- Insérer le noeud comme d'habitude.
- Mettre à jour les facteurs d'équilibre jusqu'à la racine (ou au premier noeud déséquilibré).
- Rééquilibrer le noeud si nécessaire.

Cas possibles

Sous-arbre gauche (avant)	**Sous-arbre gauche (après)**
`fe(P) =  1`	
`fe(P) =  0`	
`fe(P) = -1`	

Algorithme d'insertion

- Insérer le noeud comme d'habitude.
- Mettre à jour les facteurs d'équilibre jusqu'à la racine (ou au premier noeud déséquilibré).
- Rééquilibrer le noeud si nécessaire.

Cas possibles

Sous-arbre gauche (avant)

```
fe(P) =  1
fe(P) =  0
fe(P) = -1
```

Sous-arbre gauche (après)

```
=> fe(P) =  0
=> fe(P) = -1
=> fe(P) = -2 // Rééquilibrer P
```

Algorithme d'insertion

- Insérer le noeud comme d'habitude.
- Mettre à jour les facteurs d'équilibre jusqu'à la racine (ou au premier noeud déséquilibré).
- Rééquilibrer le noeud si nécessaire.

Cas possibles

Sous-arbre droit (avant)

```
fe(P) =  1
fe(P) =  0
fe(P) = -1
```

Sous-arbre droit (après)

Algorithme d'insertion

- Insérer le noeud comme d'habitude.
- Mettre à jour les facteurs d'équilibre jusqu'à la racine (ou au premier noeud déséquilibré).
- Rééquilibrer le noeud si nécessaire.

Cas possibles

Sous-arbre droit (avant)

```
fe(P) =  1
fe(P) =  0
fe(P) = -1
```

Sous-arbre droit (après)

```
=> fe(P) =  0
=> fe(P) = +1
=> fe(P) = +2 // Rééquilibrer P
```

Rééquilibrage

Lien avec les cas vus plus tôt

```
fe(P) = -2 && fe(gauche(P)) = -1 => cas 1a
fe(P) = -2 && fe(gauche(P)) = +1 => cas 2a

fe(P) = +2 && fe(droite(P)) = -1 => cas 2b
fe(P) = +2 && fe(droite(P)) = +1 => cas 1b
```

Dessiner les différents cas, sur le dessin ci-dessous

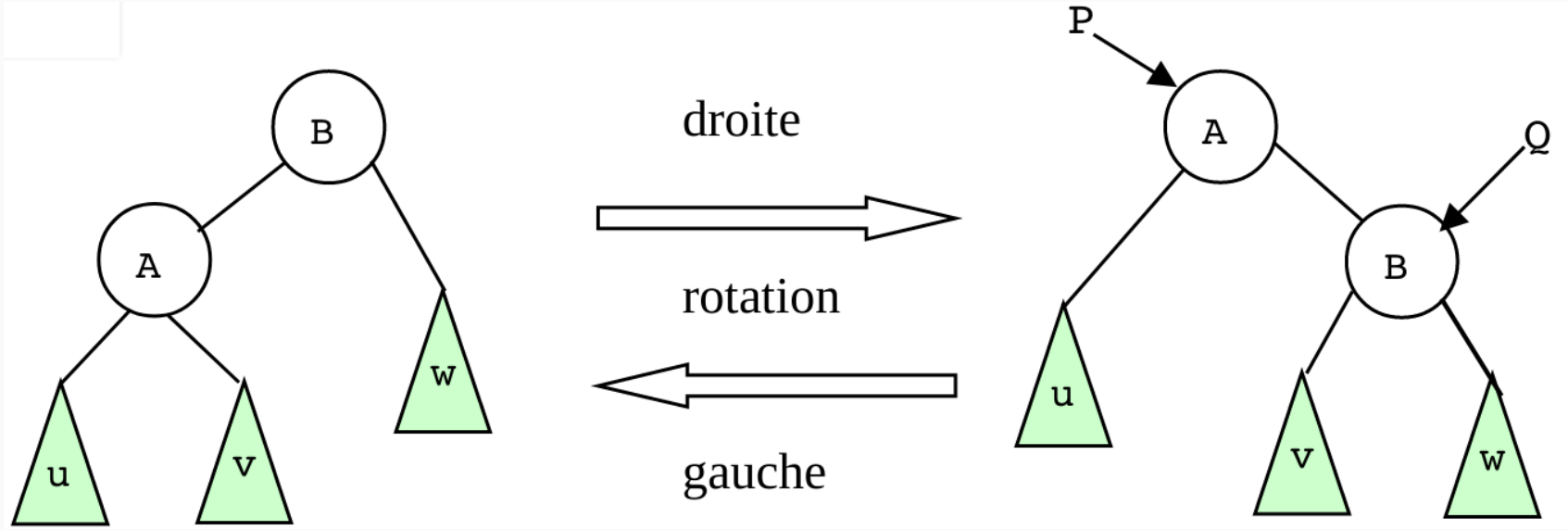


Figure 10: On verra un peu après les rotations.

La rotation

La rotation gauche (5min, matrix)

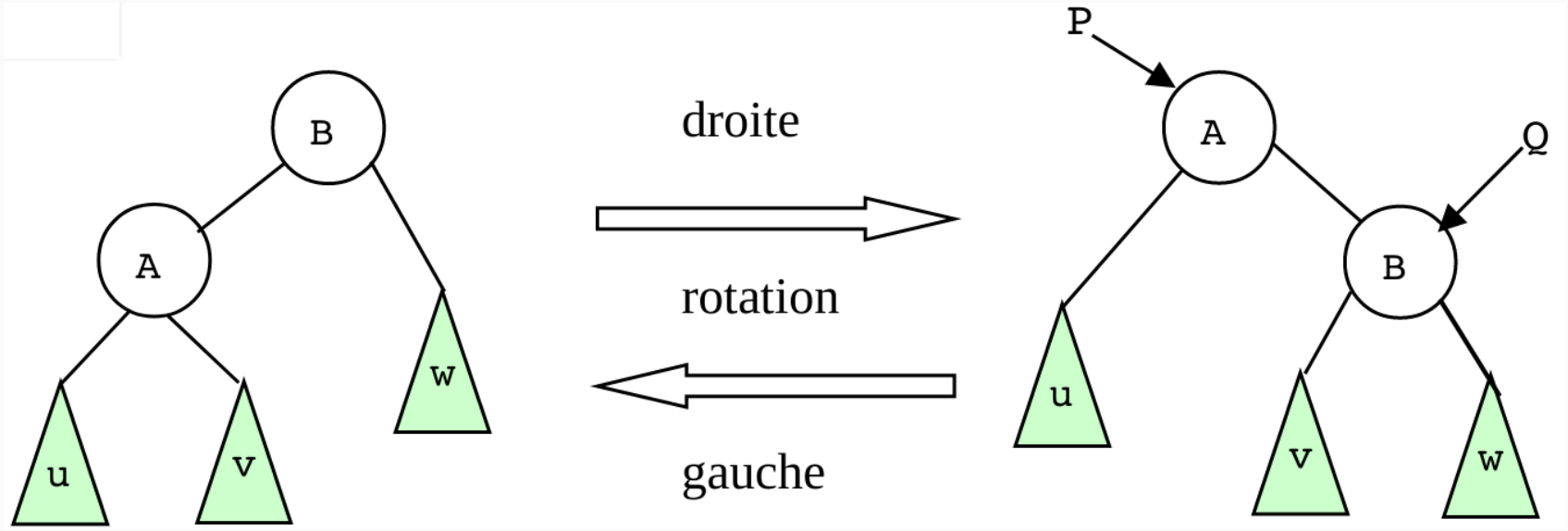


Figure 11: L'arbre de droite devient celui de gauche. Comment?

La rotation

La rotation gauche (5min, matrix)

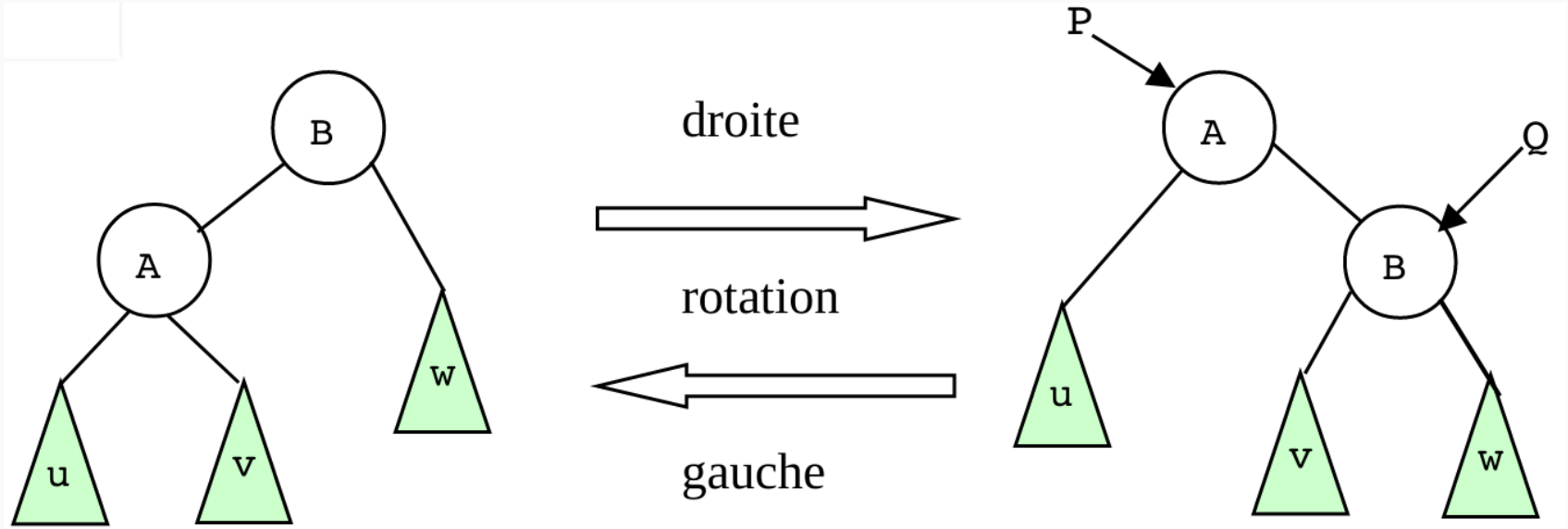


Figure 11: L'arbre de droite devient celui de gauche. Comment?

```
arbre rotation_gauche(arbre P)
    si est_non_vide(P)
        Q = droite(P)
        droite(P) = gauche(Q)
        gauche(Q) = P
        retourne Q
    retourne P
```

La rotation en C (1/2)

La rotation gauche

```
arbre rotation_gauche(arbre P)
    si est_non_vide(P)
        Q = droite(P)
        droite(P) = gauche(Q)
        gauche(Q) = P
        retourne Q
    retourne P
```

Écrire le code C correspondant (5min, matrix)

1. Structure de données
2. Fonction `tree_t rotation_left(tree_t tree)`

La rotation en C (1/2)

La rotation gauche

```
arbre rotation_gauche(arbre P)
    si est_non_vide(P)
        Q = droite(P)
        droite(P) = gauche(Q)
        gauche(Q) = P
        retourne Q
    retourne P
```

Écrire le code C correspondant (5min, matrix)

1. Structure de données
2. Fonction `tree_t rotation_left(tree_t tree)`

```c
typedef struct _node {
    int key;
    struct _node *left, *right;
    int bf; // balance factor
} node;
typedef node *tree_t;
```

La rotation en C (2/2)

```
tree_t rotation_left(tree_t tree) {
    tree_t subtree = NULL;
    if (NULL != tree) {
        subtree = tree->right;
        tree->right = subtree->left;
        subtree->left = tree;
    }
    return subtree;
}
```

La rotation en C (2/2)

```
tree_t rotation_left(tree_t tree) {
    tree_t subtree = NULL;
    if (NULL != tree) {
        subtree = tree->right;
        tree->right = subtree->left;
        subtree->left = tree;
    }
    return subtree;
}
```

- Et la rotation à droite, pseudo-code (5min)?

La rotation en C (2/2)

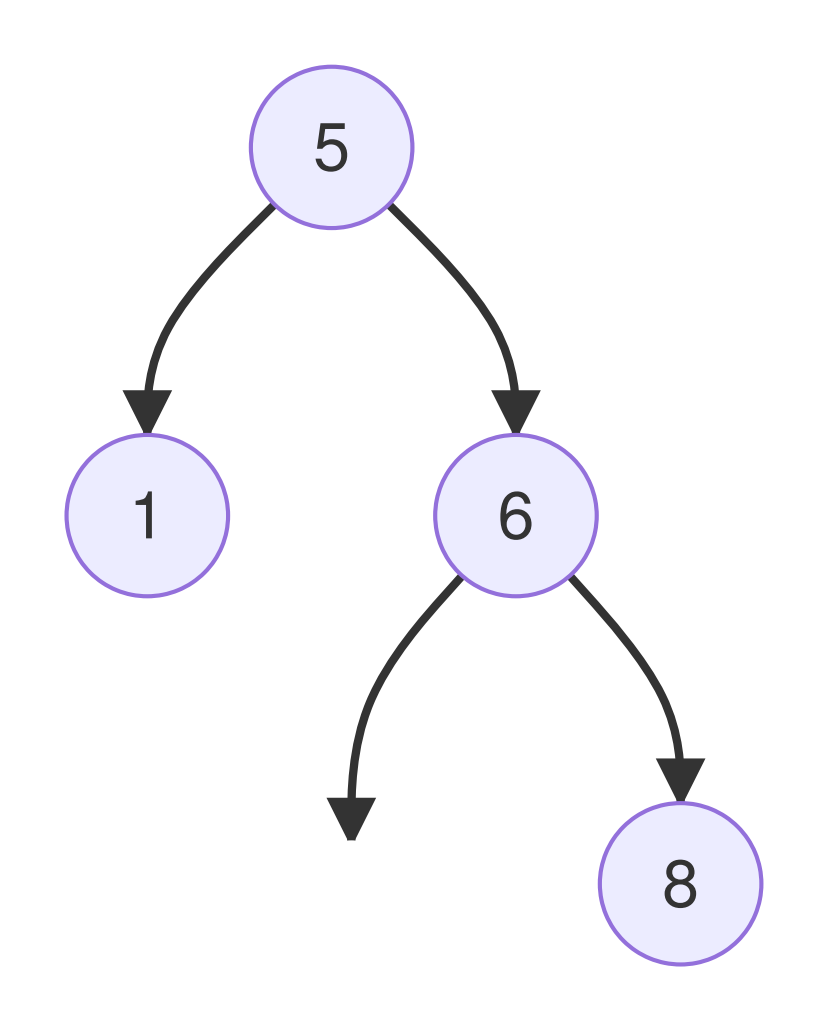
```c
tree_t rotation_left(tree_t tree) {
    tree_t subtree = NULL;
    if (NULL != tree) {
        subtree = tree->right;
        tree->right = subtree->left;
        subtree->left = tree;
    }
    return subtree;
}
```

- Et la rotation à droite, pseudo-code (5min)?

```
arbre rotation_droite(arbre P)
    si est_non_vide(P)
        Q = gauche(P)
        gauche(P) = droite(Q)
        droite(Q) = P
        retourne Q
    retourne P
```

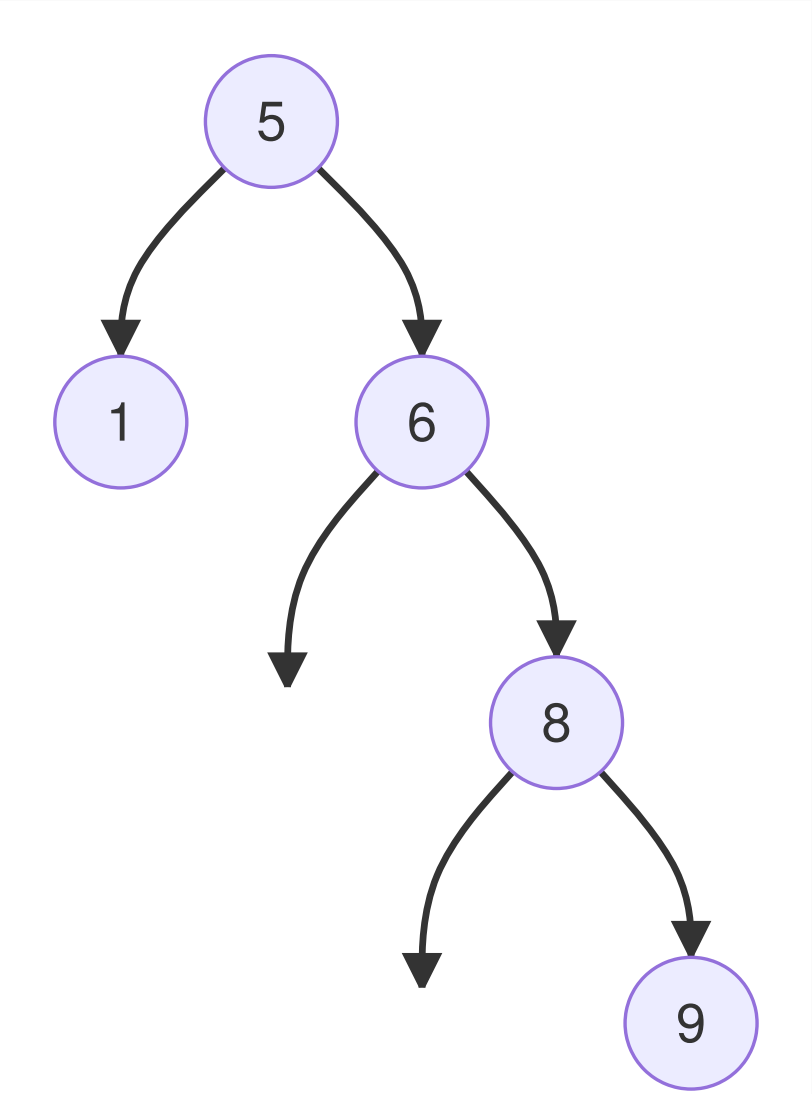
Exemple de rotation (1/2)

Insertion de 9?



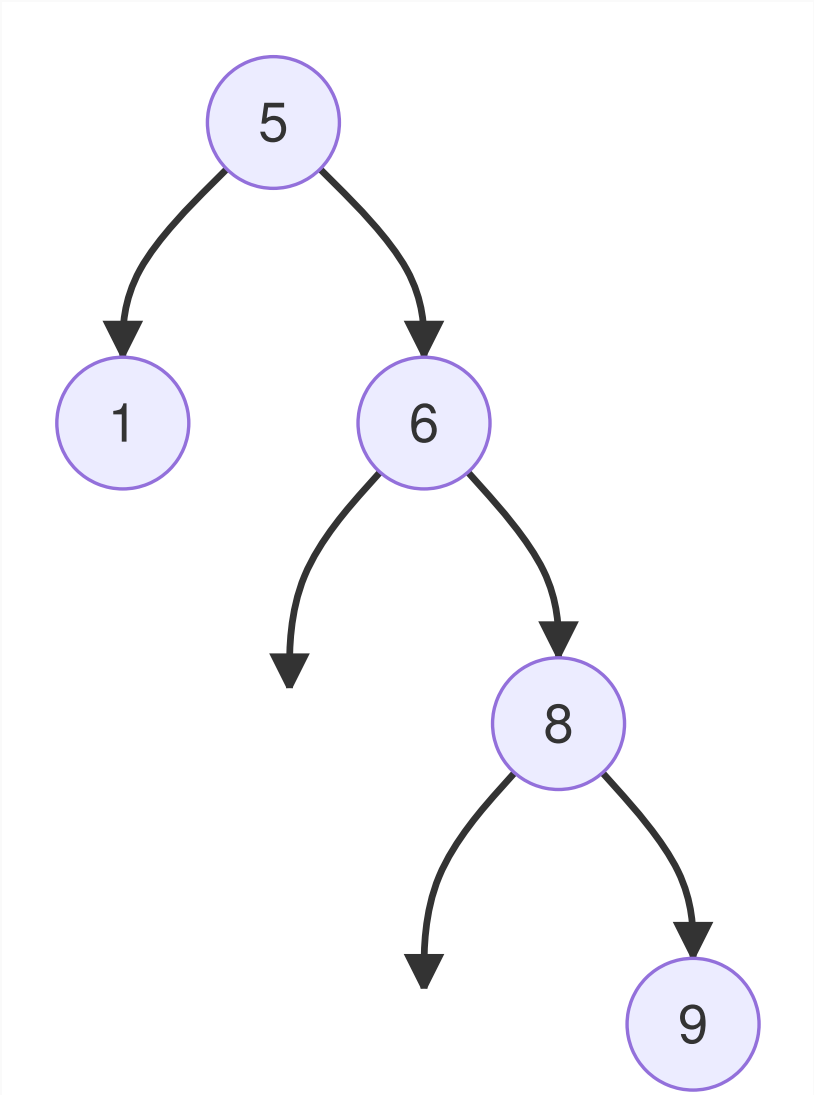
Exemple de rotation (2/2)

Quelle rotation et sur quel noeud (5 ou 6)?

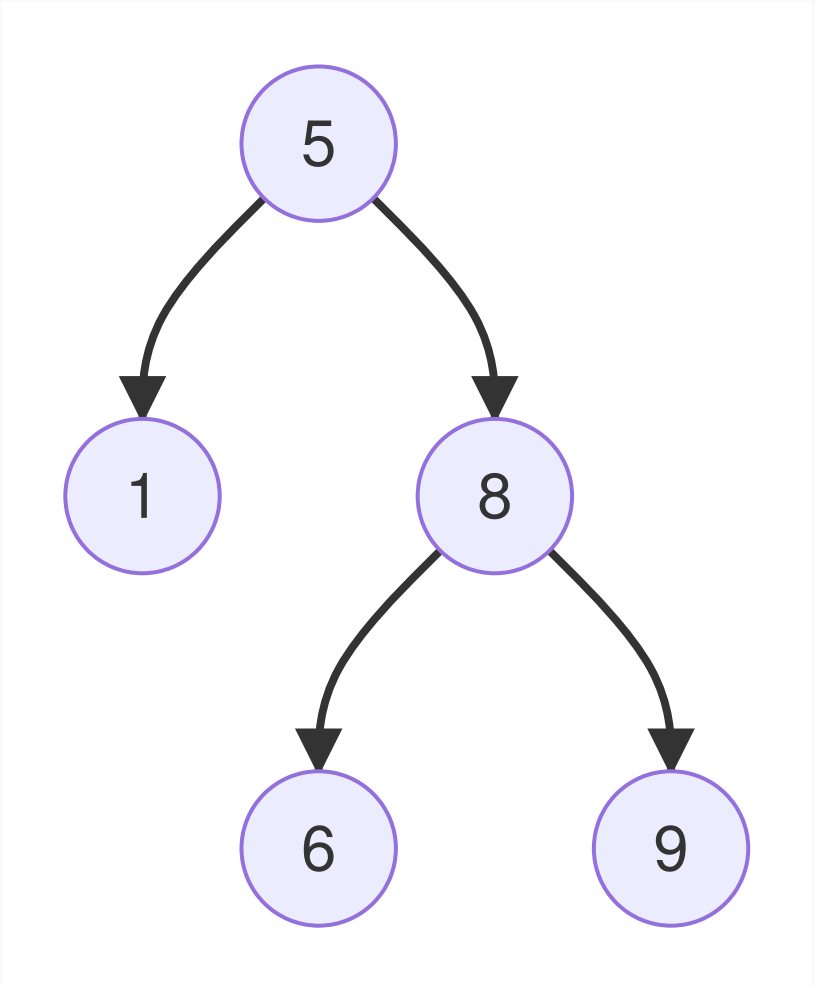


Exemple de rotation (2/2)

Quelle rotation et sur quel noeud (5 ou 6)?

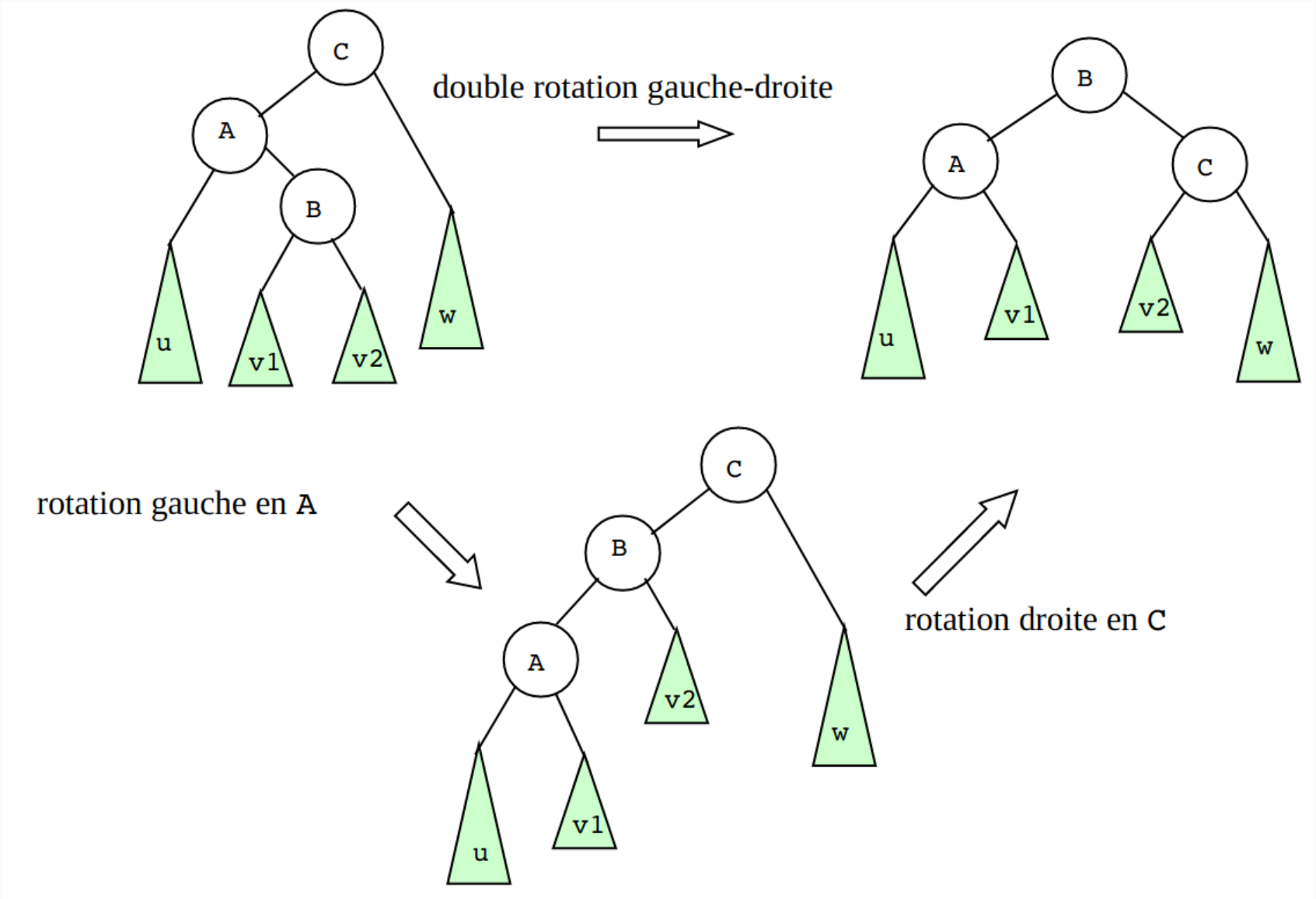


Sur le plus jeune évidemment!



La rotation gauche-droite

Là c'est plus difficile (cas 2a/b)

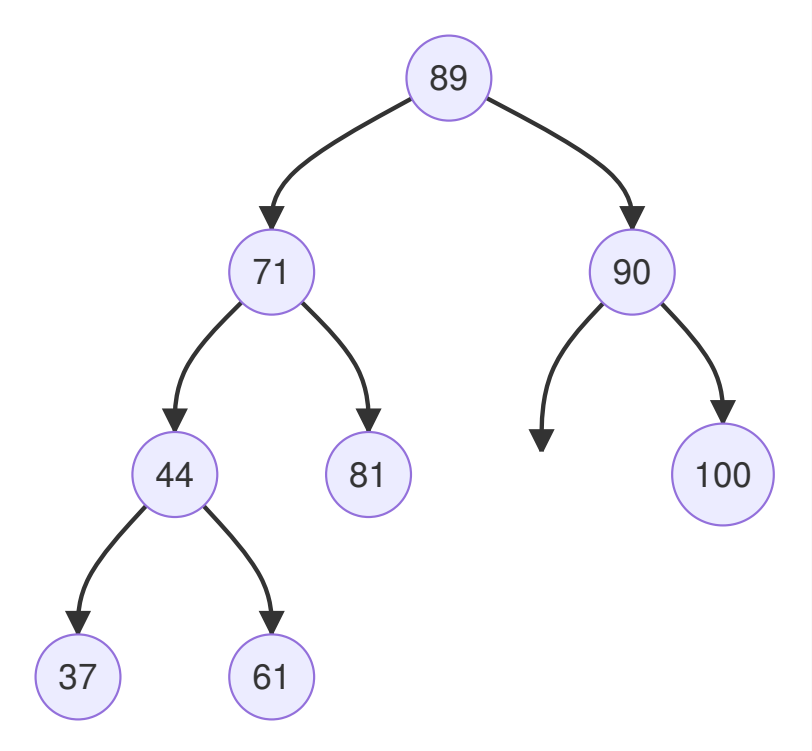


Exercices

Faire l'implémentation de la double rotation (pas corrigé, 5min)

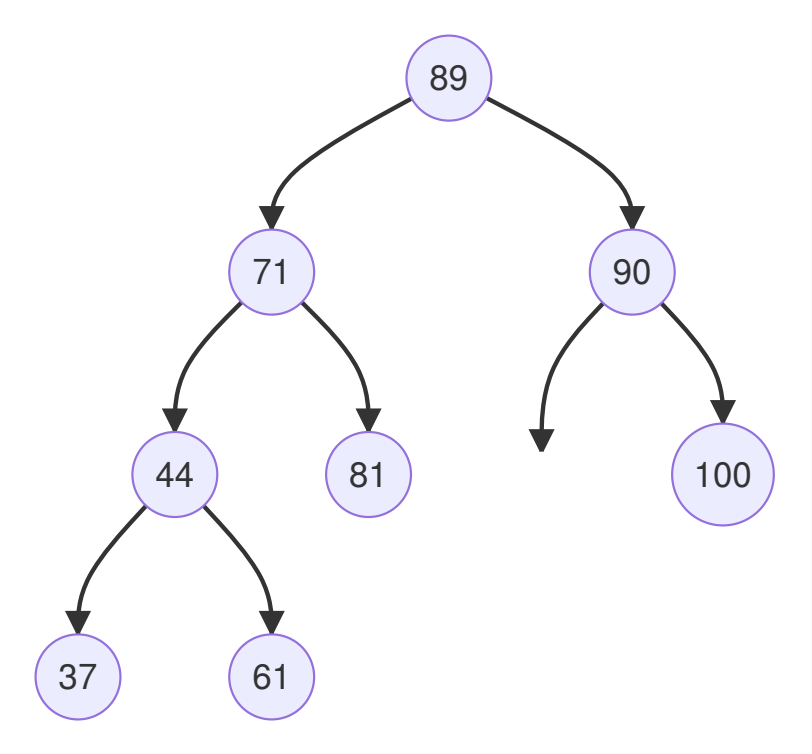
Exercices

Insérer 50, ex 10min (matrix)

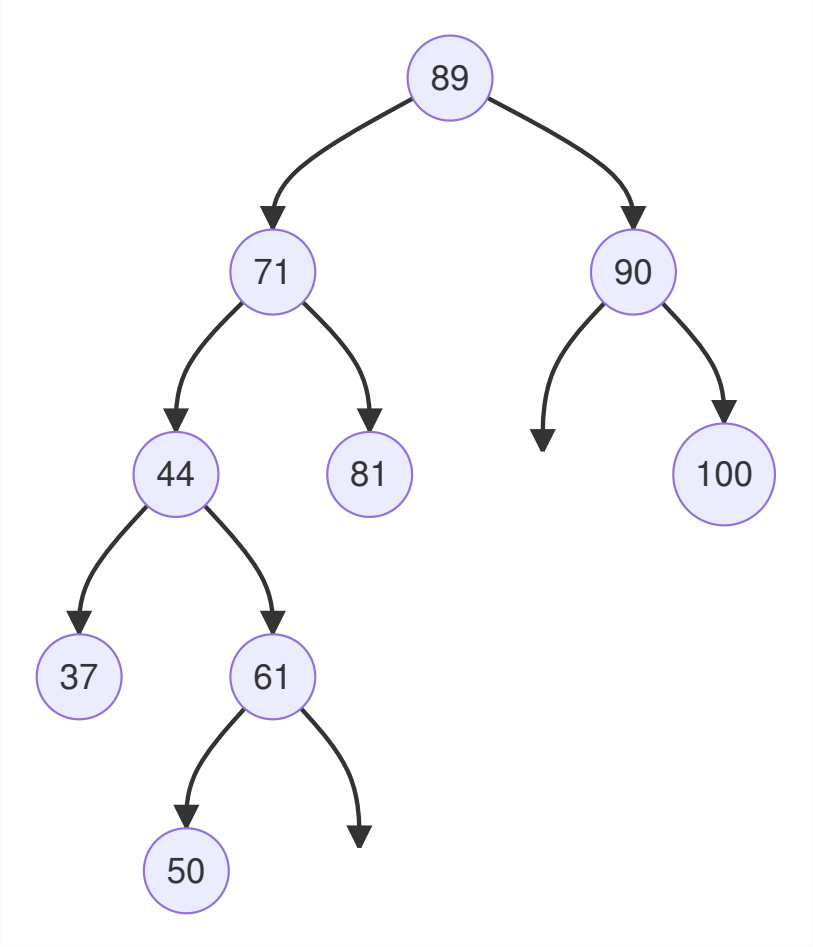


Exercices

Insérer 50, ex 10min (matrix)

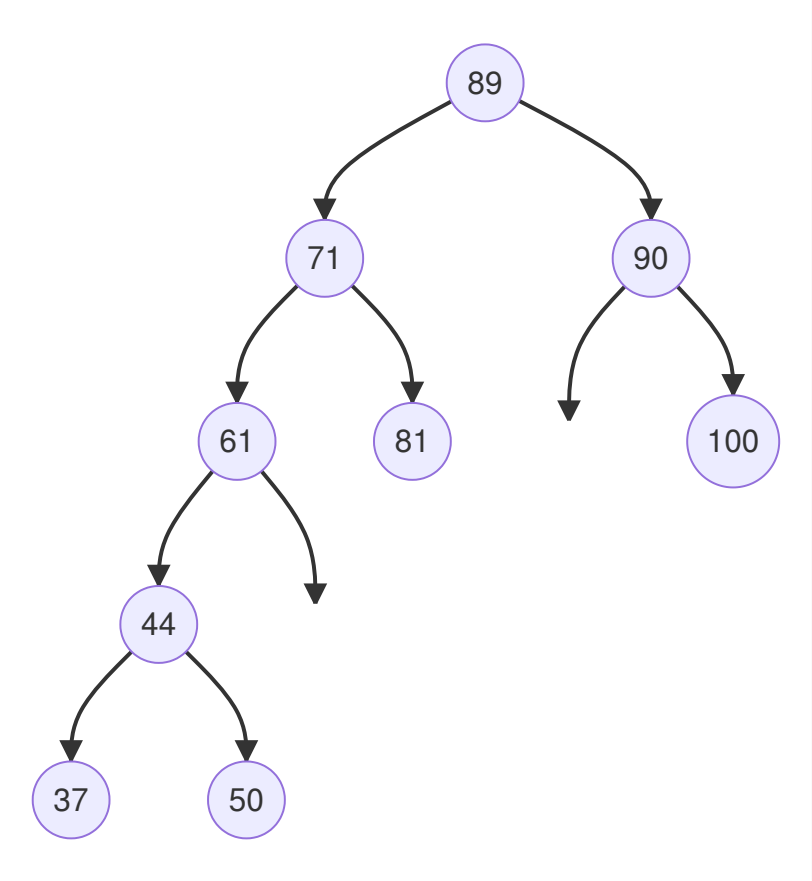


Où se fait la rotation?



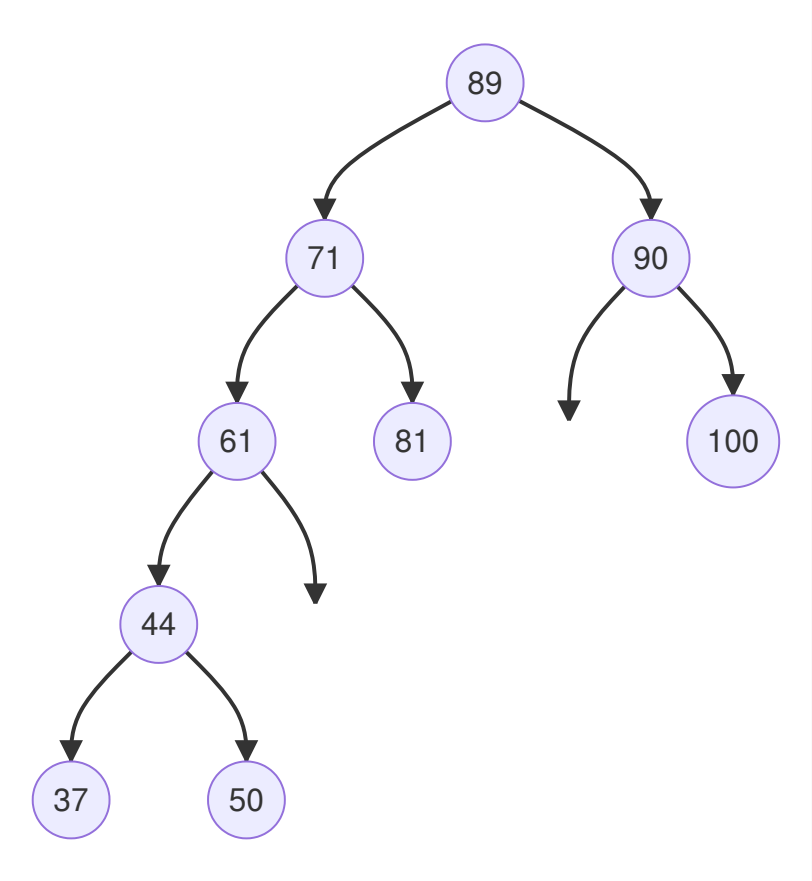
Exercices

Rotation gauche en 44

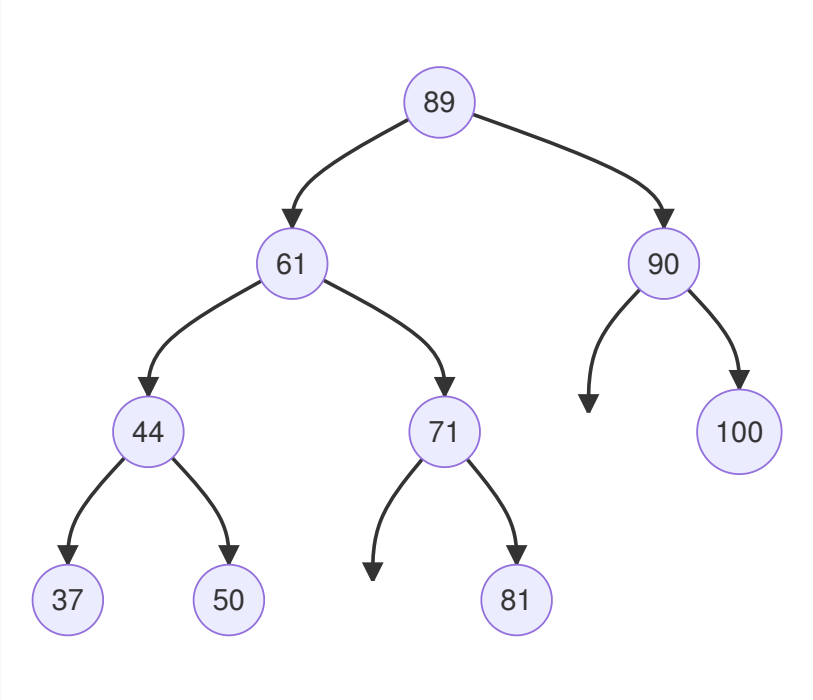


Exercices

Rotation gauche en 44

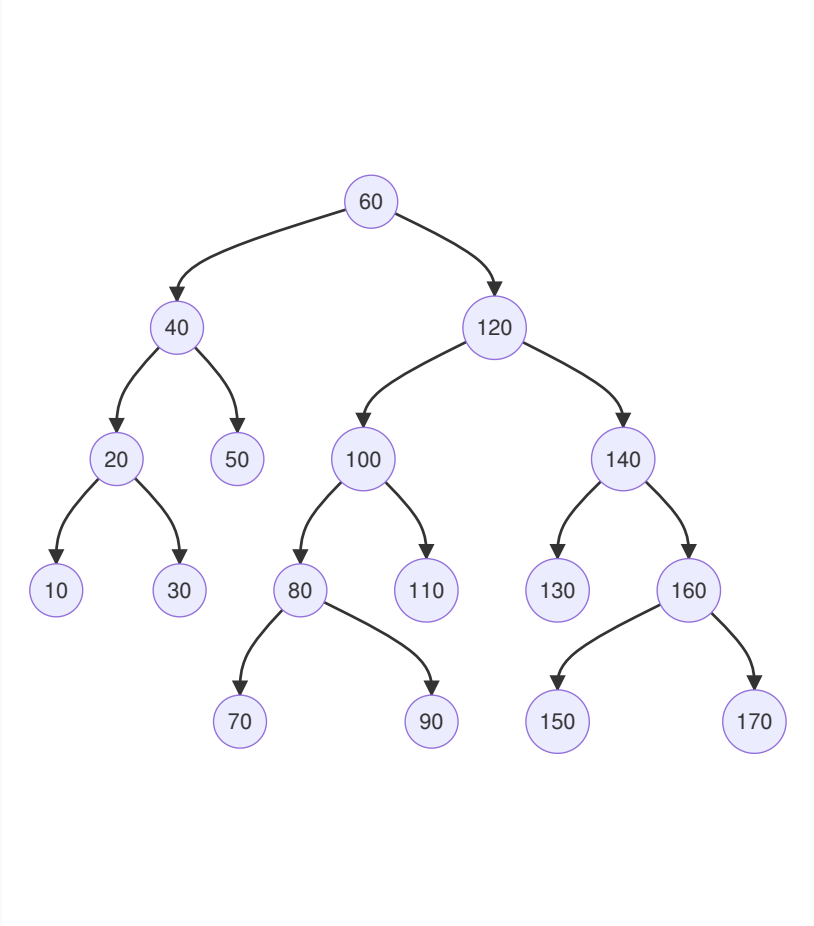


Rotation à droite en 71



Exercice de la mort

Soit l'arbre AVL suivant:



1. Montrer les positions des insertions de feuille qui conduiront à un arbre désequilibré.
2. Donner les facteurs d'equilgaucheibre.
3. Dessiner et expliquer les modifications de l'arbre lors de l'insertion de la valeur 65. On mentionnera les modifications des facteurs d'équilibre.

Encore un petit exercice

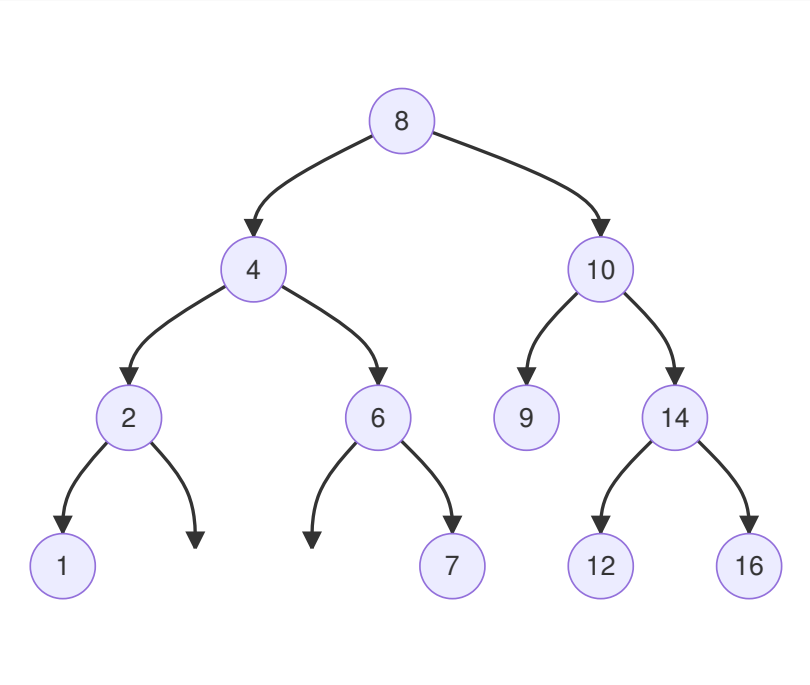
- Insérer les nœuds suivants dans un arbre AVL

25 | 60 | 35 | 10 | 5 | 20 | 65 | 45 | 70 | 40
| 50 | 55 | 30 | 15

Un à un et le/la premier/ère qui poste la bonne réponse sur matrix a un point

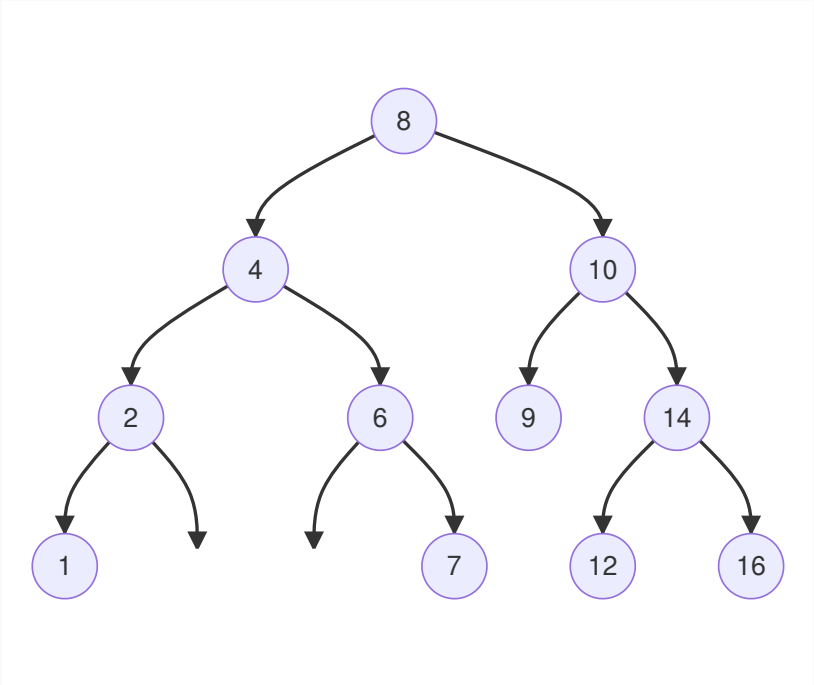
Suppression dans un arbre AVL

Algorithme par problème: supprimer 10

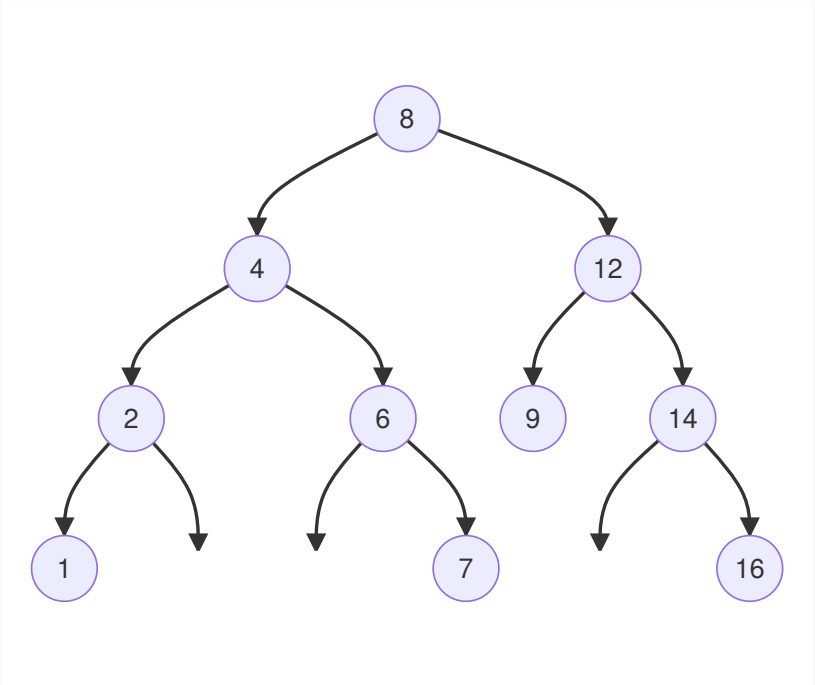


Suppression dans un arbre AVL

Algorithme par problème: supprimer 10

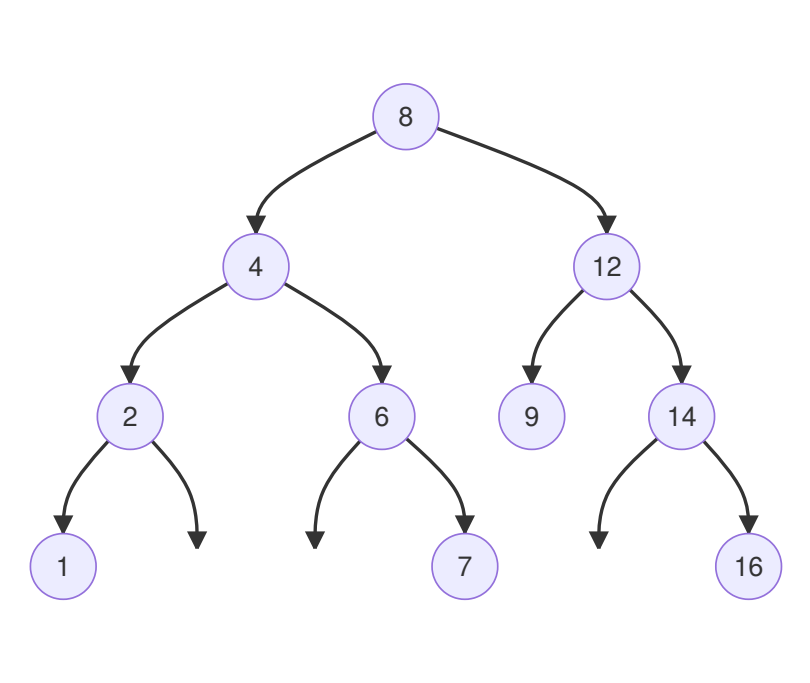


Algorithme par problème: supprimer 10



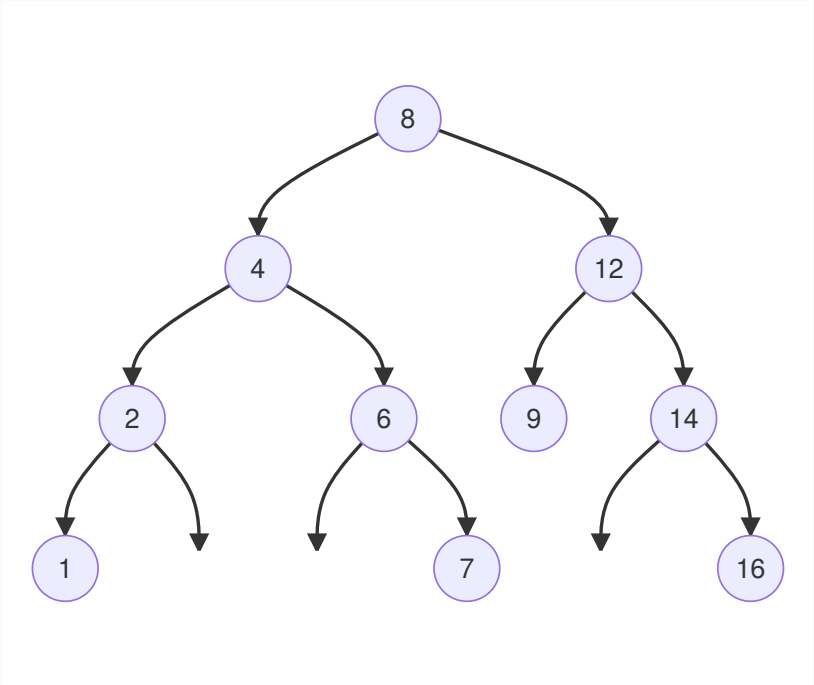
Suppression dans un arbre AVL

Algorithme par problème: supprimer 8

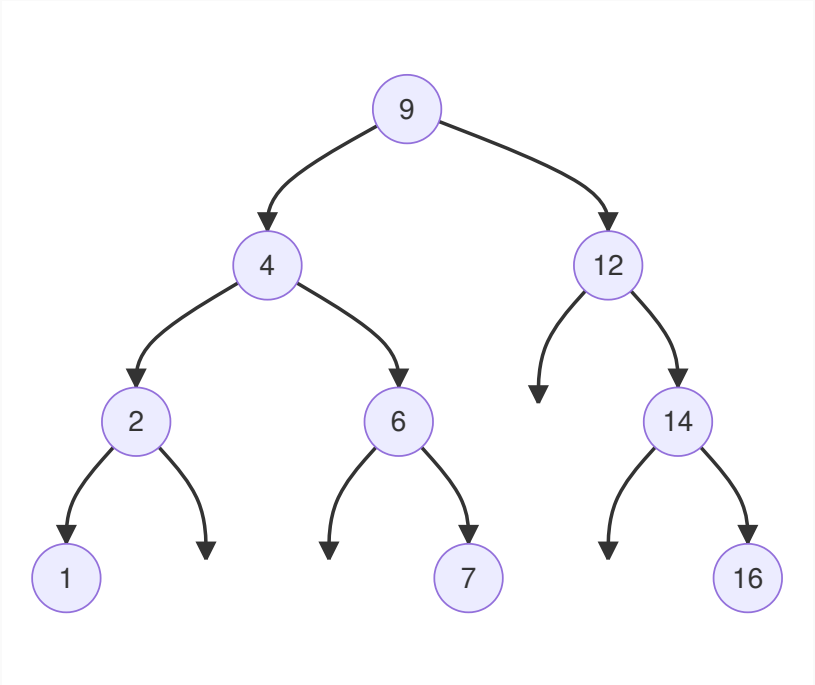


Suppression dans un arbre AVL

Algorithme par problème: supprimer 8

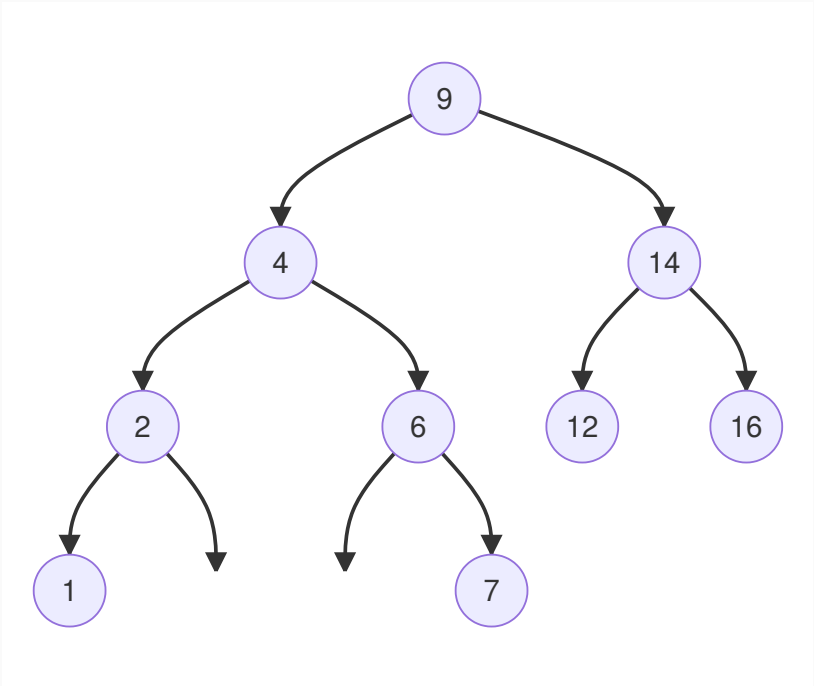


Algorithme par problème: rotation de 12



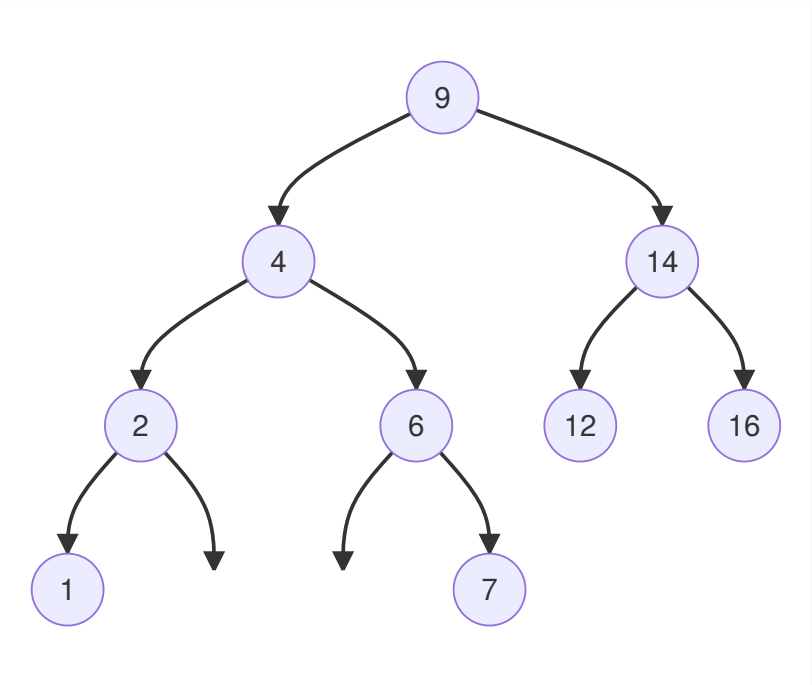
Suppression dans un arbre AVL

Algorithme par problème: rotation de 12



Suppression dans un arbre AVL

Algorithme par problème: rotation de 12

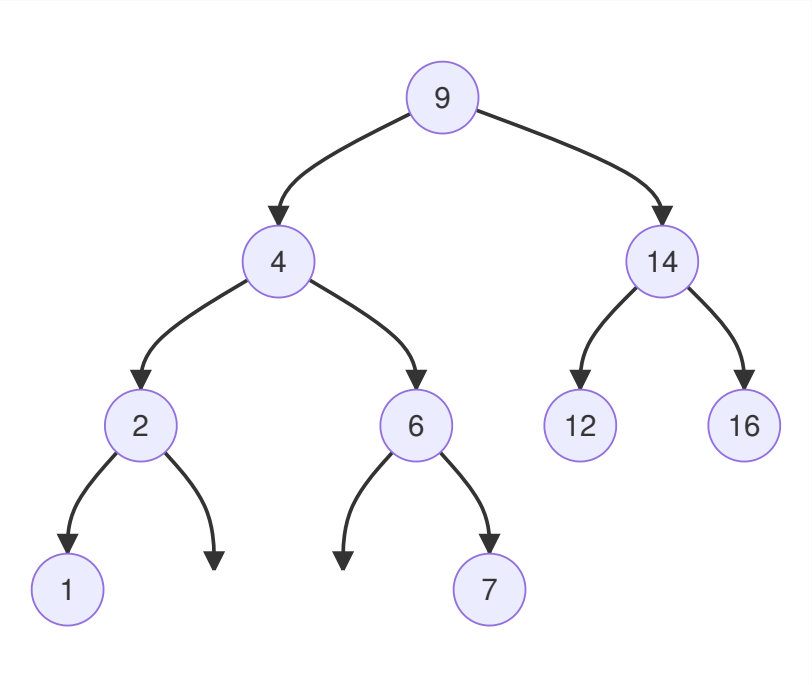


1. On supprime comme d'habitude.
2. On rééquilibre si besoin à l'endroit de la suppression.

- Facile non?

Suppression dans un arbre AVL

Algorithme par problème: rotation de 12

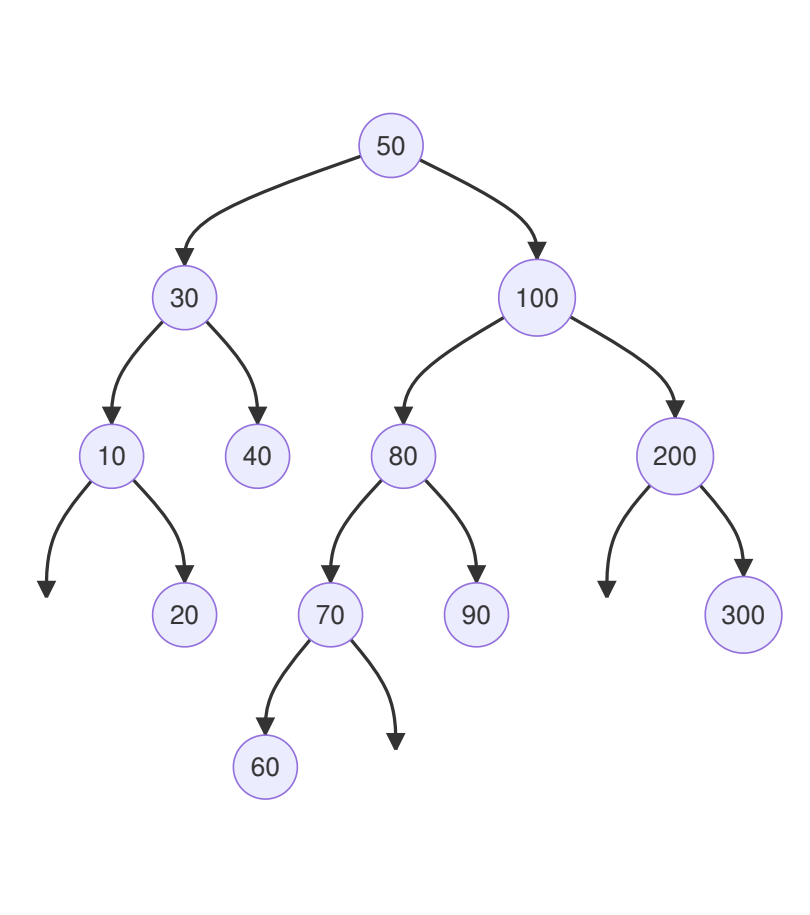


1. On supprime comme d'habitude.
2. On rééquilibre si besoin à l'endroit de la suppression.

- Facile non?
- Plus dur....

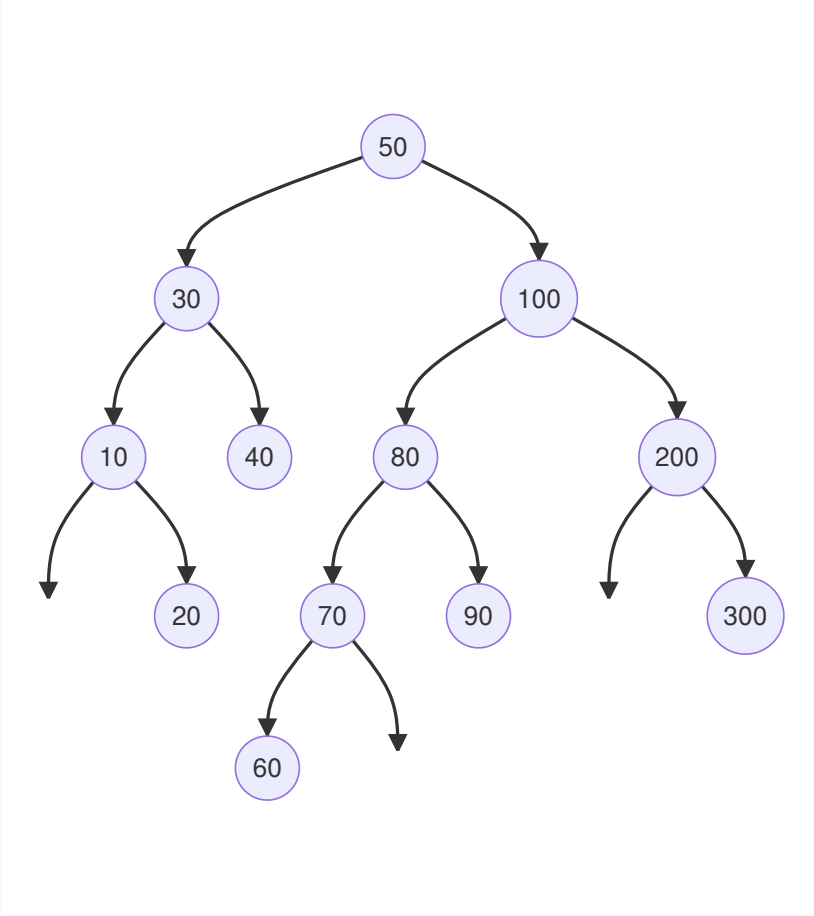
Suppression dans un arbre AVL 2.0

Algorithme par problème: suppression de 30

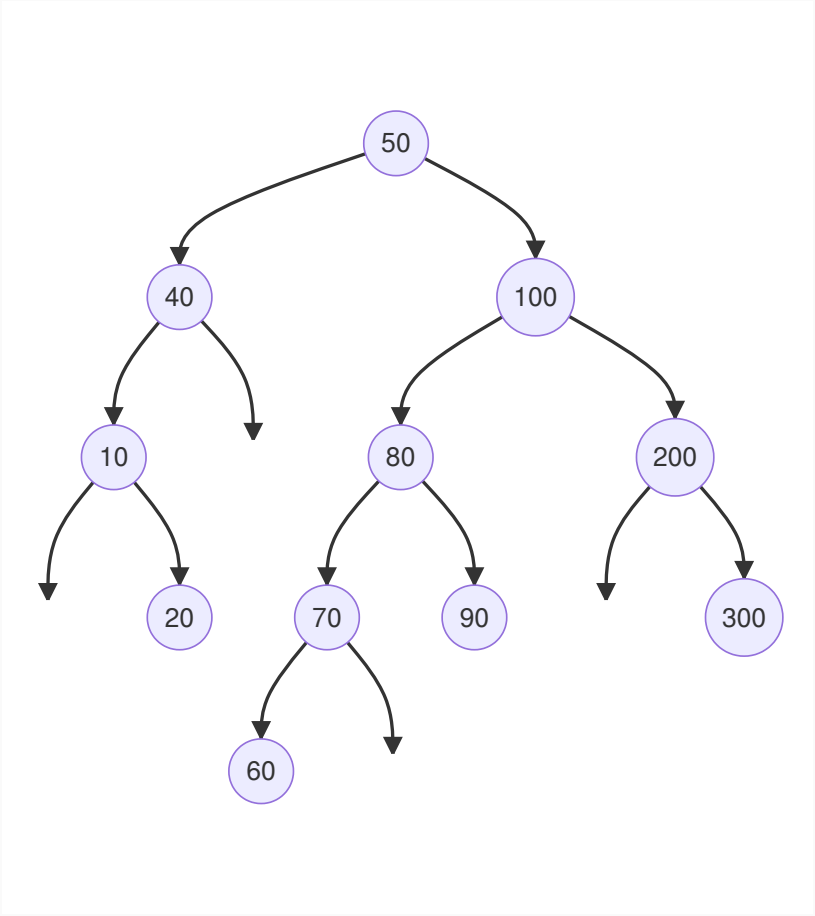


Suppression dans un arbre AVL 2.0

Algorithme par problème: suppression de 30

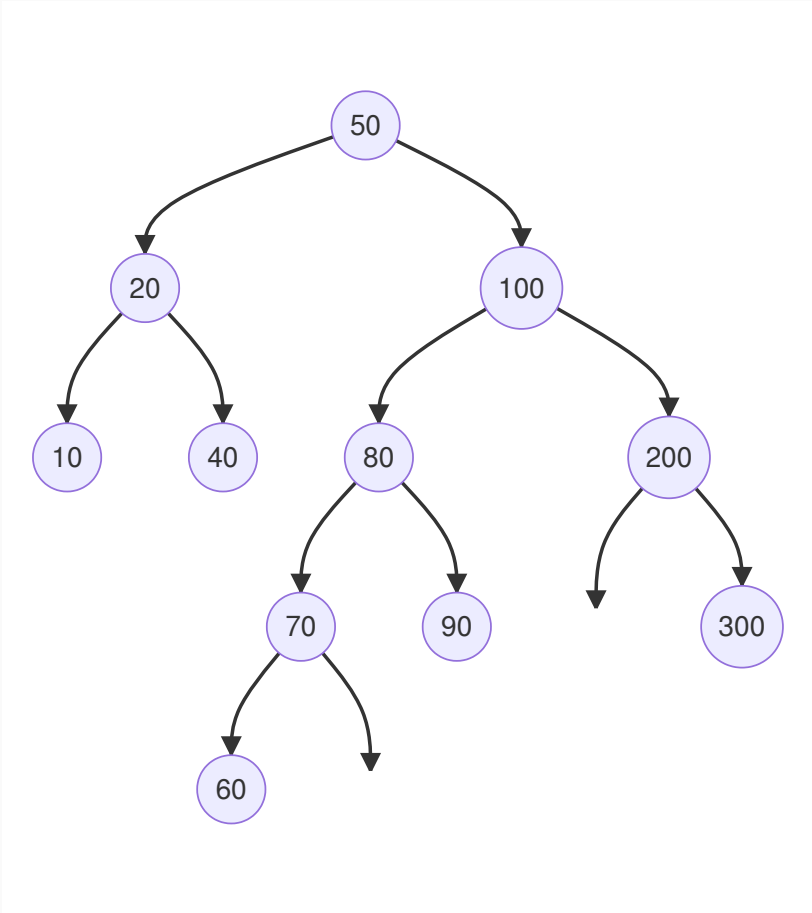


Algorithme par problème: rotation GD autour de 40



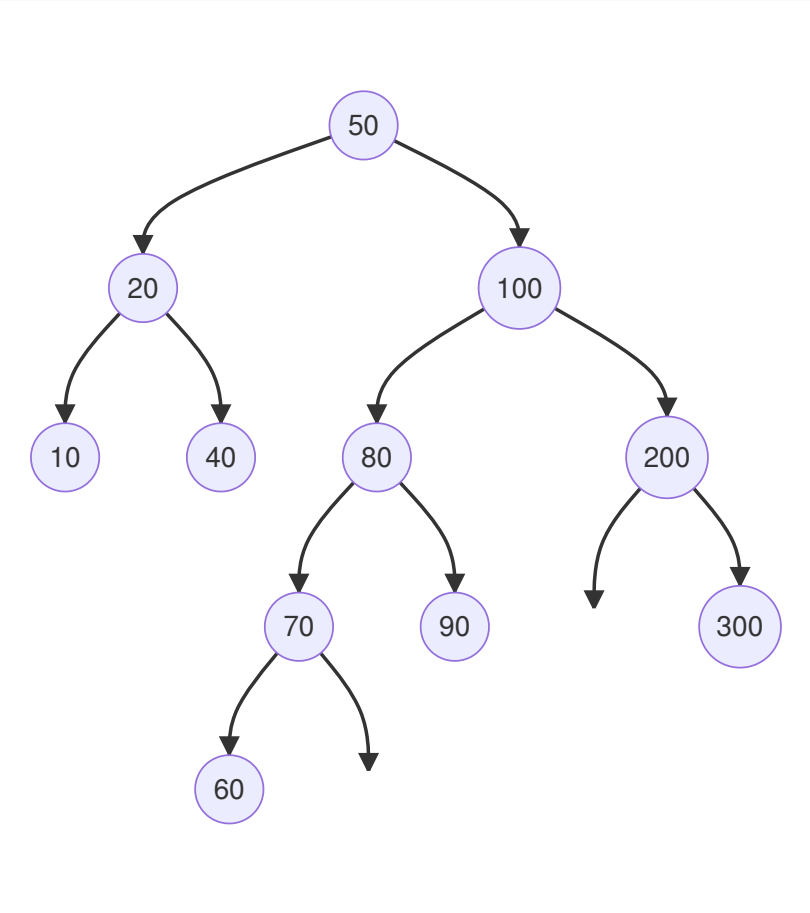
Suppression dans un arbre AVL 2.0

Argl! 50 est déséquilibré!

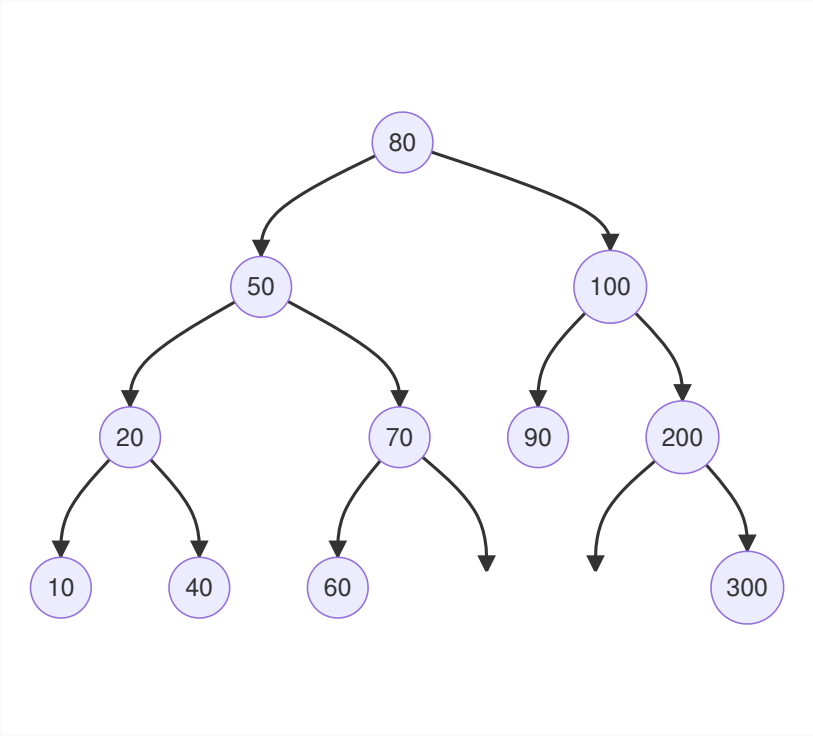


Suppression dans un arbre AVL 2.0

Argl! 50 est déséquilibré!



Algorithme par problème: rotation DG autour de 50



Résumé de la suppression

1. On supprime comme pour un arbre binaire de recherche.
2. Si un nœud est déséquilibré, on le rééquilibre.
   - Cette opération peut déséquilibrer un autre nœud.
3. On continue à rééquilibrer tant qu'il y a des nœuds à équilibrer.