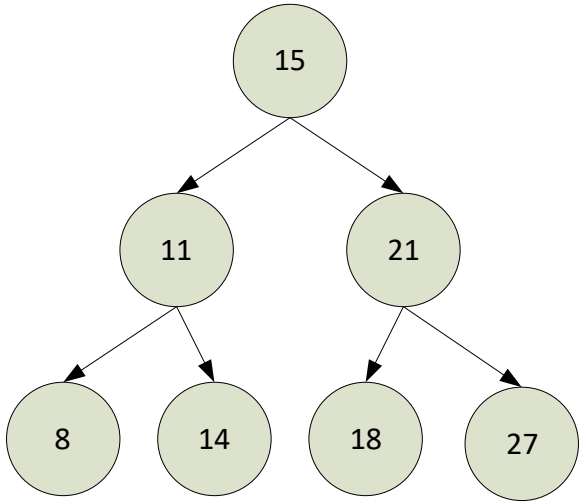
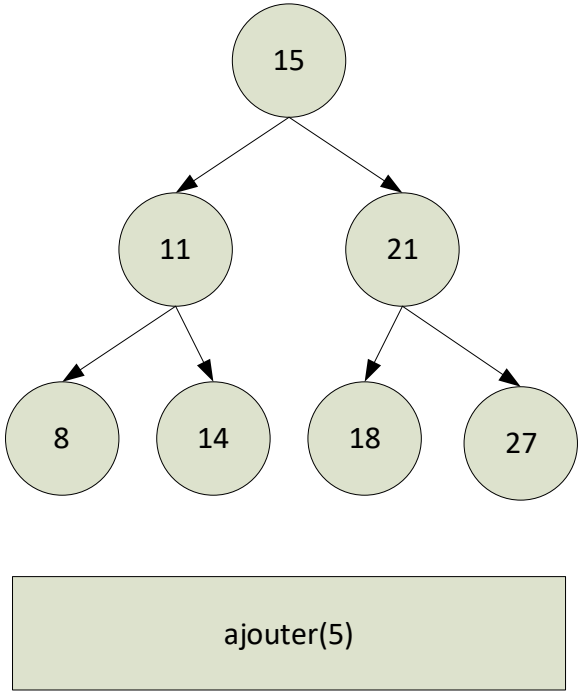
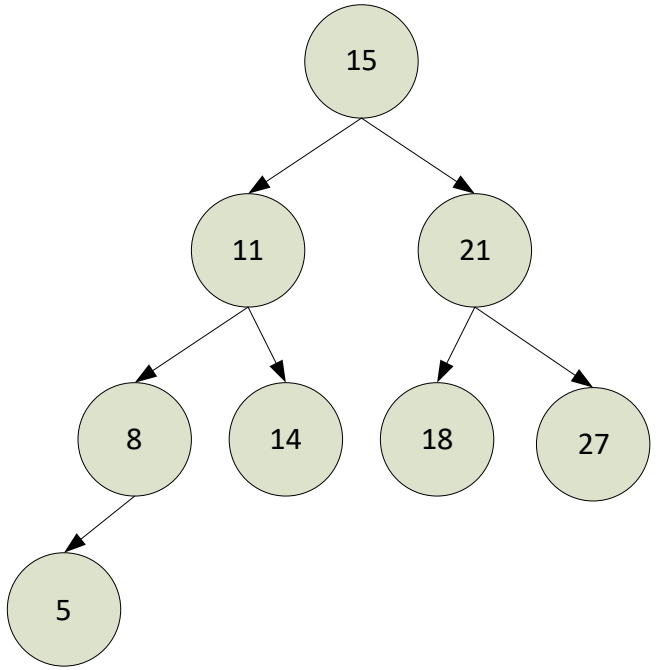
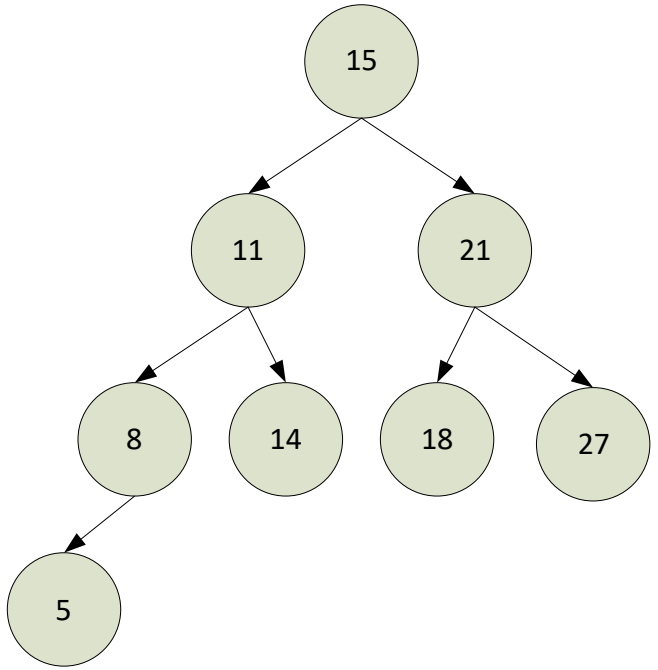
Arbres AVL et rotations

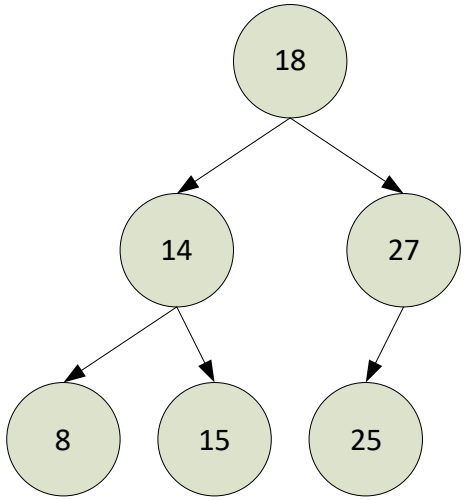

15
11
21
8
14
18
27

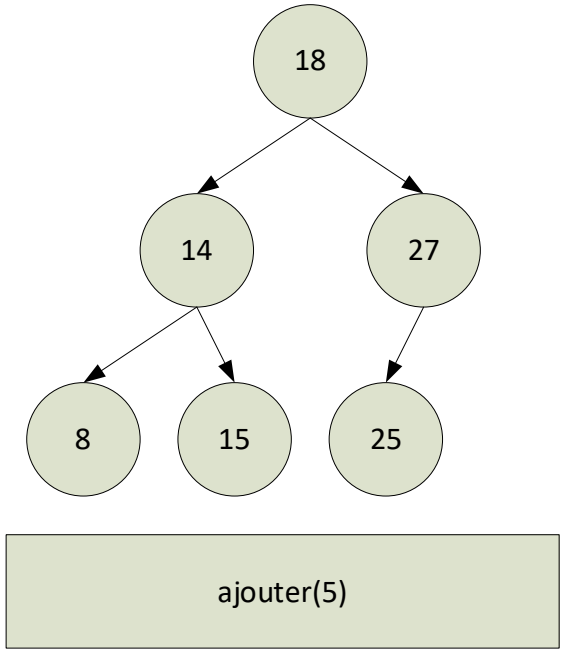

15
11
21
8
14
18
27
ajouter(5)

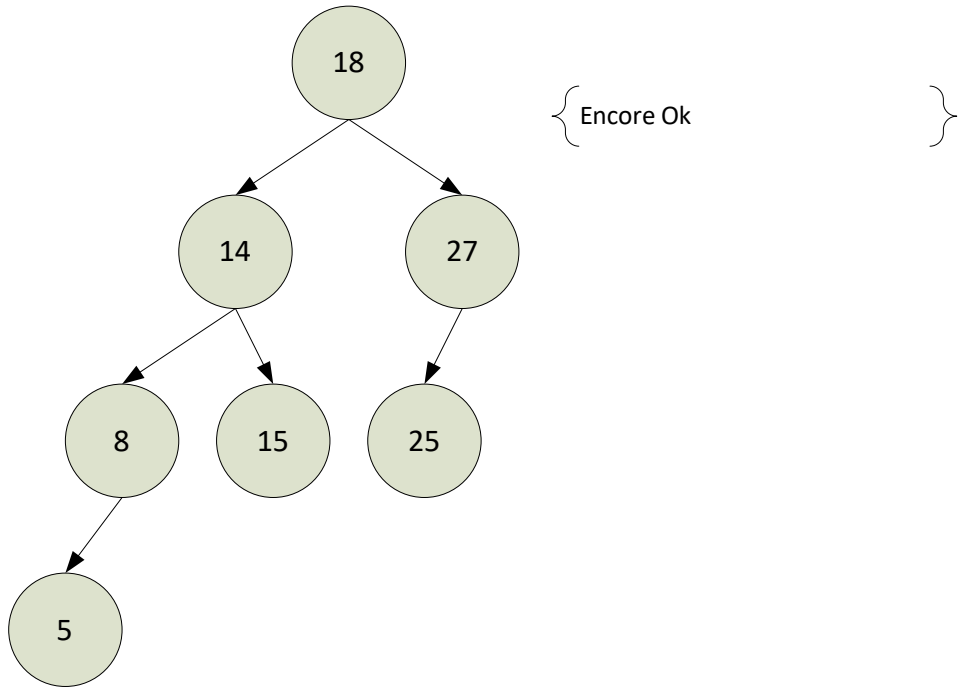

15
11
21
8
14
18
27
5

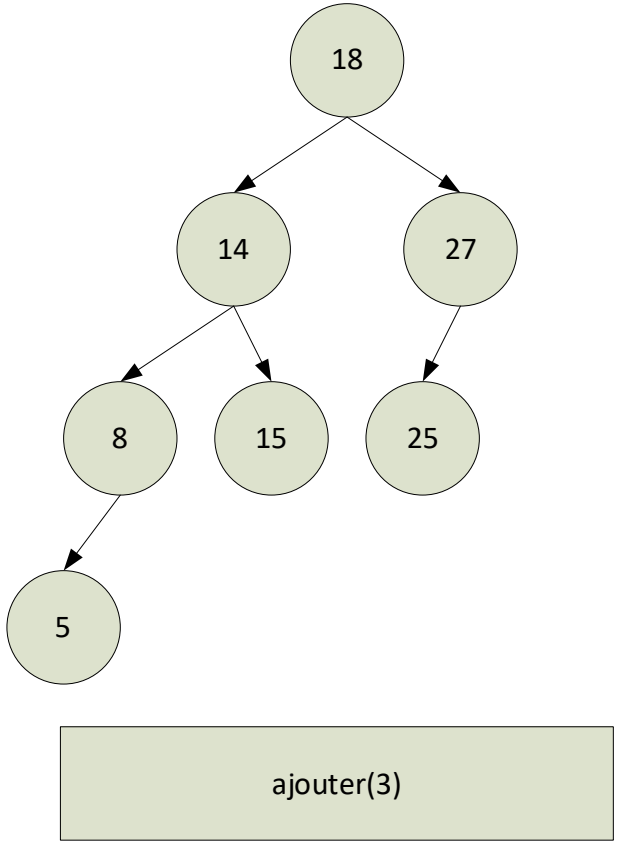
Tout va bien. Parfois, la vie est juste facile

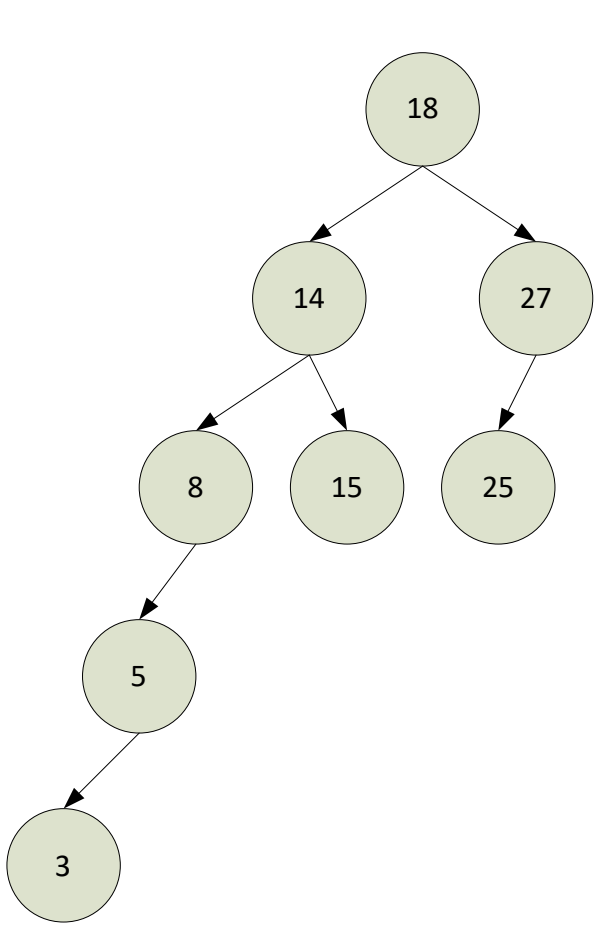



18
14
27
8
15
25


18
14
27
8
15
25
ajouter(5)

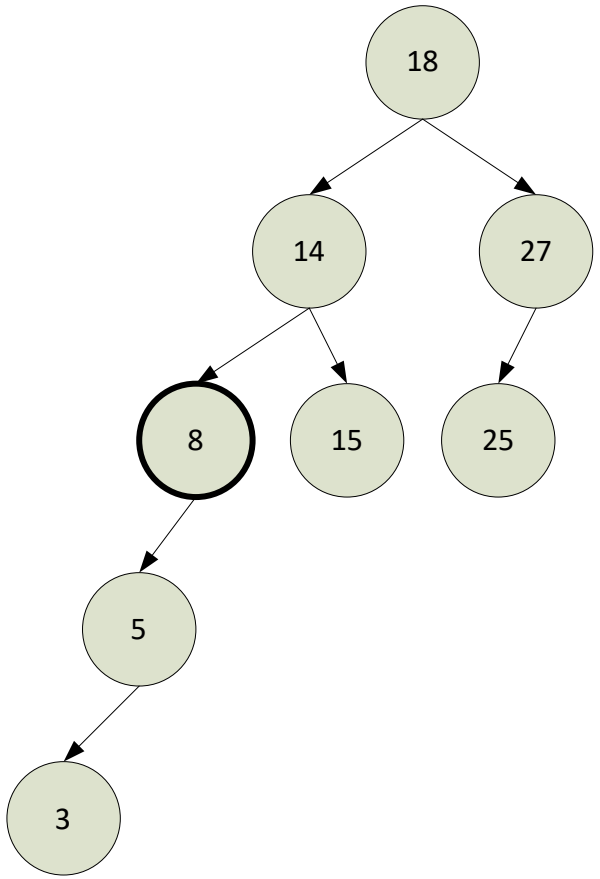

18
14
27
8
15
25
5
Encore Ok


18
14
27
8
15
25
5
ajouter(3)



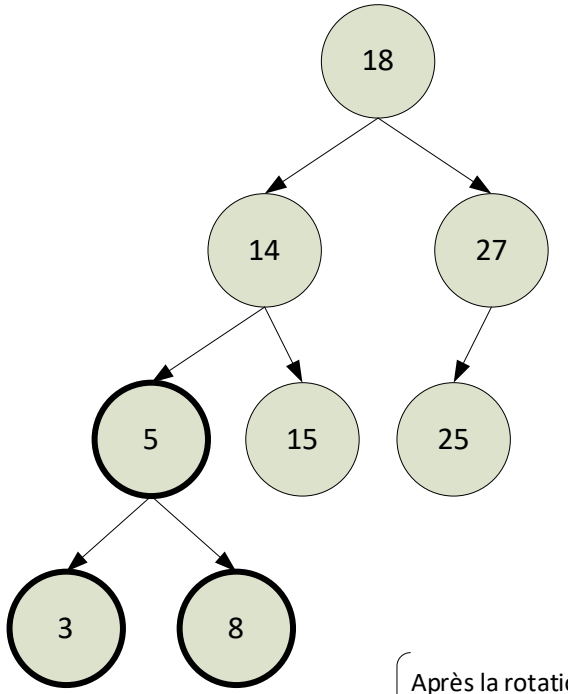
Débalancement à gauche (différence de hauteur de 2), détectable à partir de (8)

Note : ceci est une fois l'insertion initiale faite mais avant les rotations (normalement, pas visible au code client)

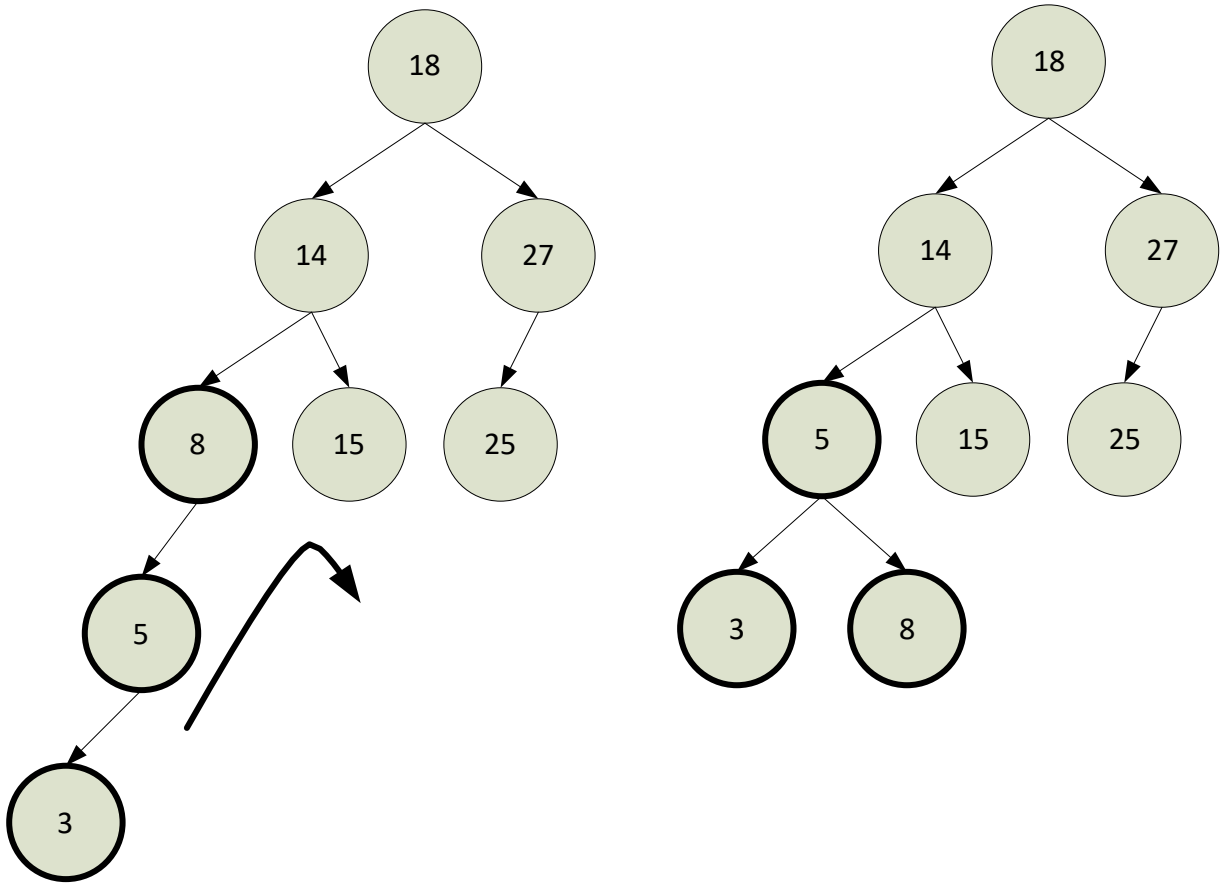


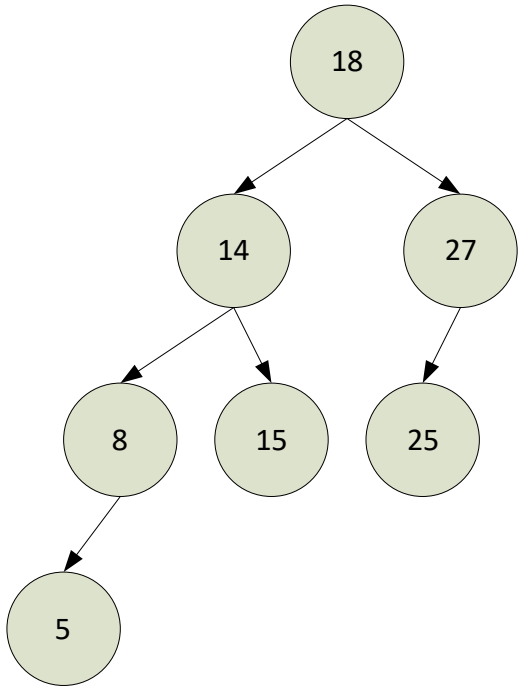
On balance…

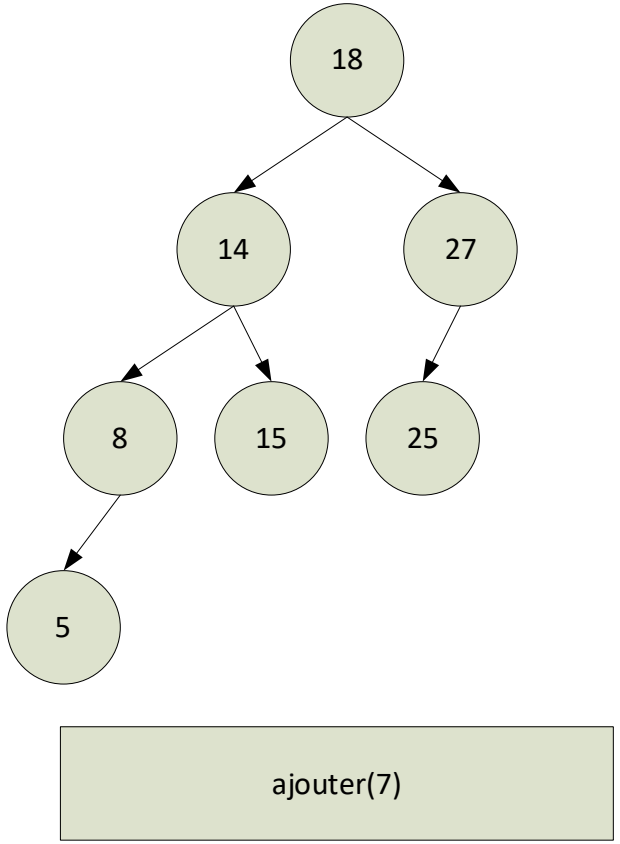
À partir de (8), il faut une rotation à droite car la hauteur de gauche est 2 et la hauteur de droite est 0

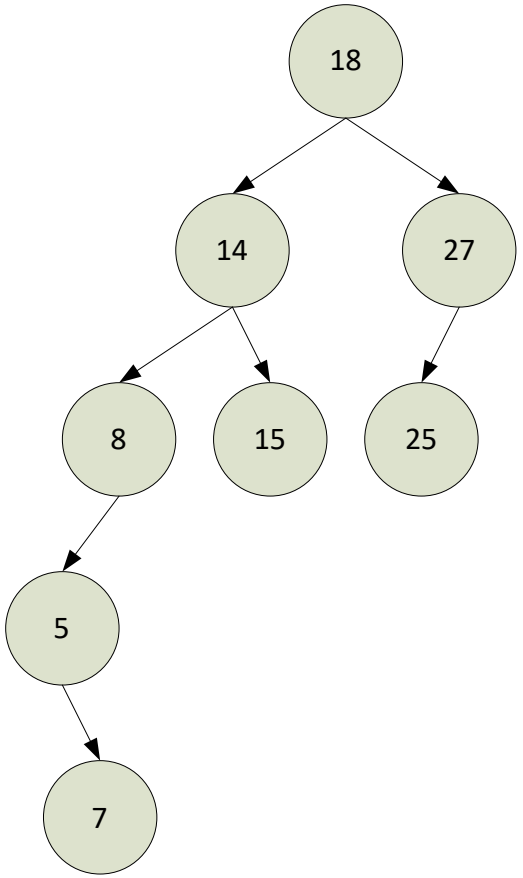


Après la rotation à droite, on en est à ceci. On a un arbre AVL, mais on le saura une fois le balancement remonté jusqu'au point où on peut le constater


18
14
27
8
15
25
5
3
18
14
27
5
15
25
3
8

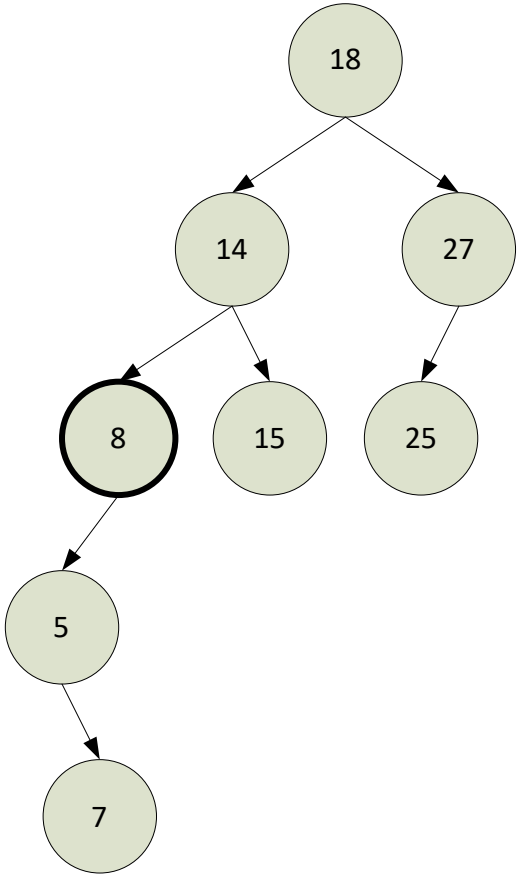

18
14
27
8
15
25
5


18
14
27
8
15
25
5
ajouter(7)



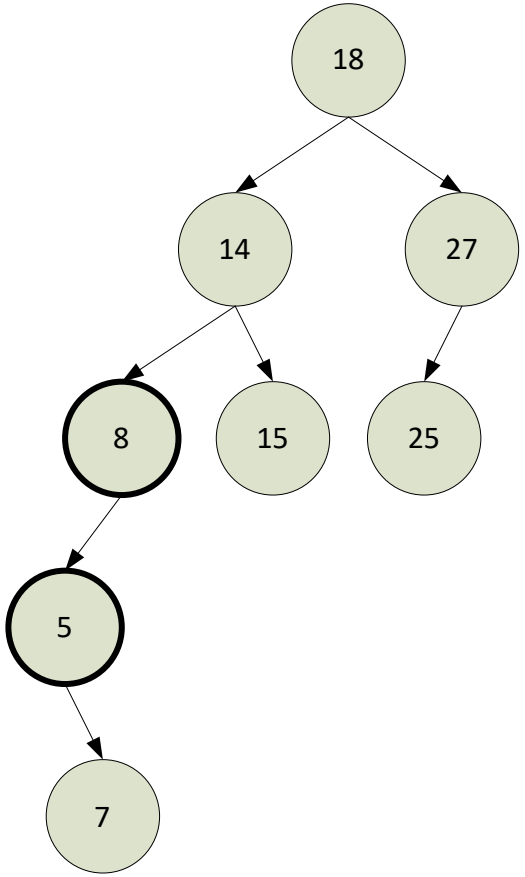
Débalancement à gauche (différence de hauteur de 2), détectable à partir de (8)

Note : ceci est une fois l'insertion initiale faite mais avant les rotations (normalement, pas visible au code client)

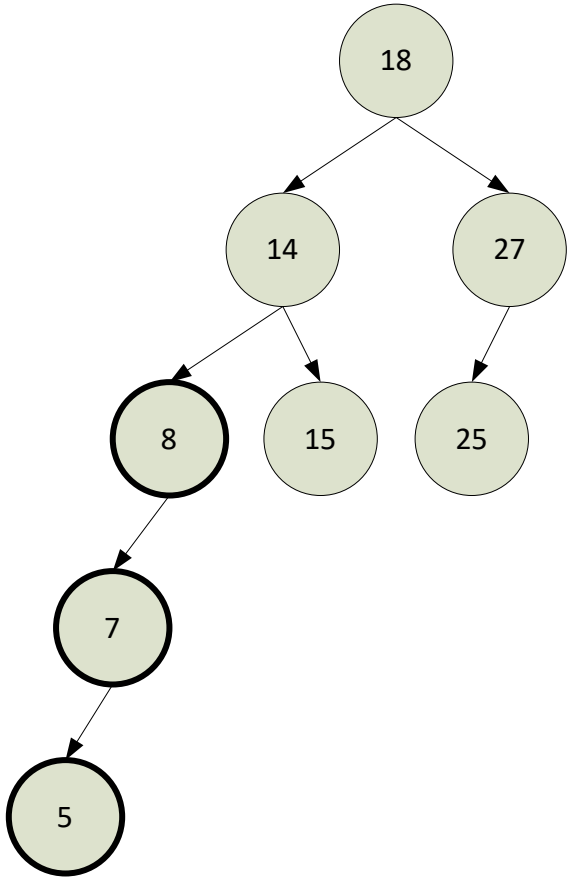


On balance…

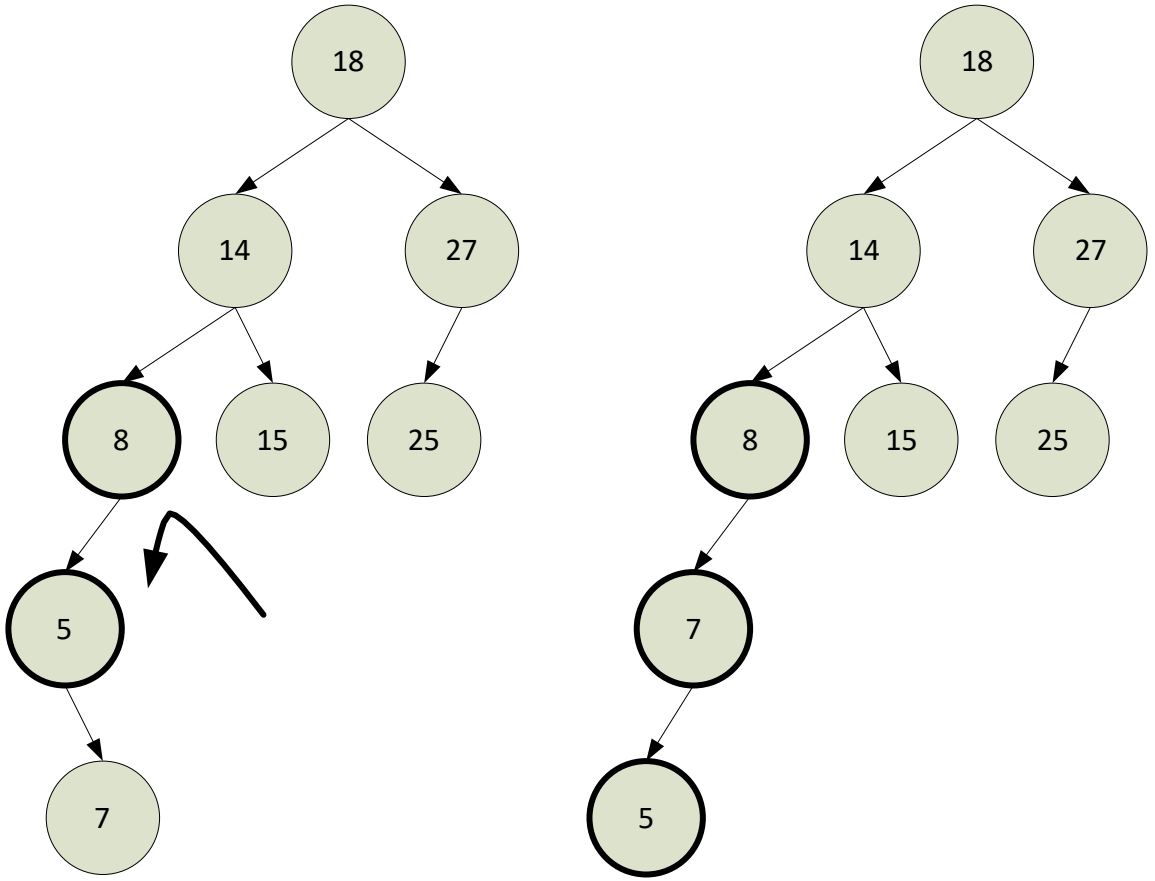
À partir de (8), il faut une rotation à droite car la hauteur de gauche est 2 et la hauteur de droite est 0…

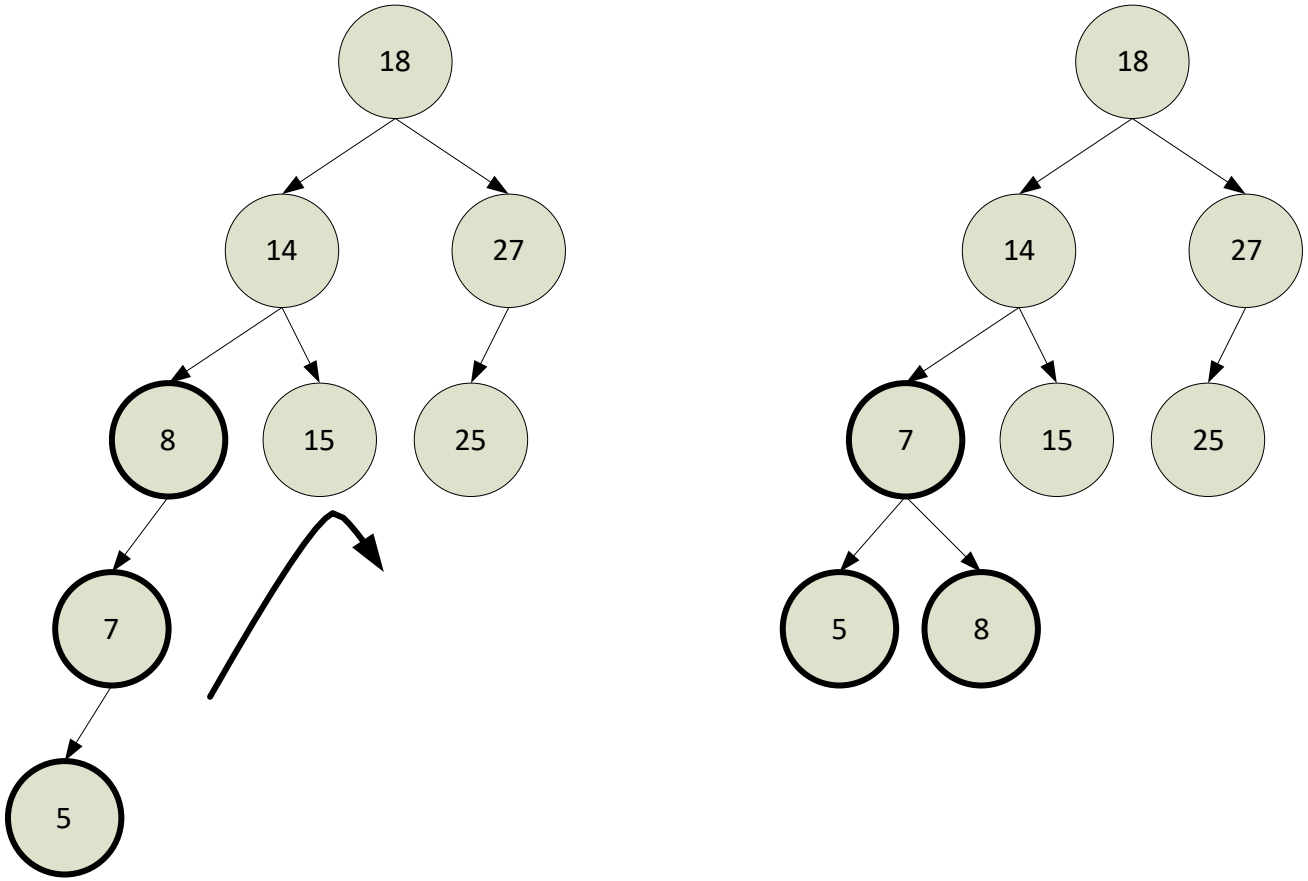


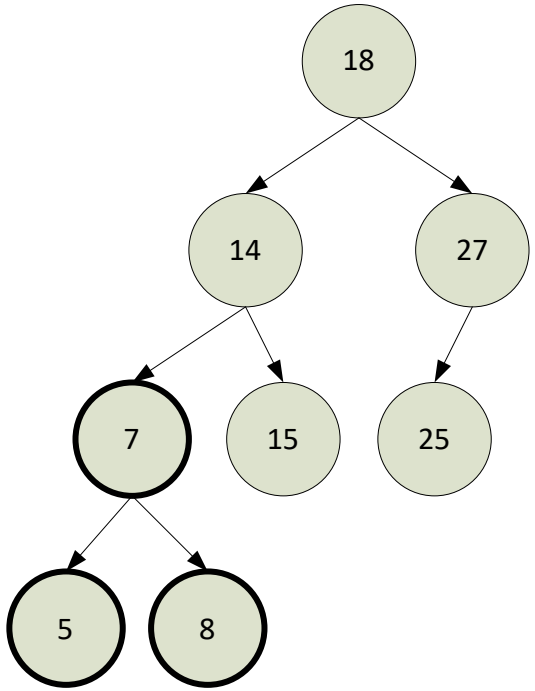
… mais dans ce cas il faut d'abord réaliser une rotation à gauche autour de 5…



Après la rotation à gauche
autour de 5, on en est à ceci…


18
14
27
8
15
25
5
7
18
14
27
8
15
25
7
5


18
14
27
8
15
25
7
5
18
14
27
7
15
25
5
8



… puis après la rotation à droite autour de (8), on obtient ceci. On a un arbre AVL, mais on le saura une fois le balancement remonté jusqu'au point où on peut le constater

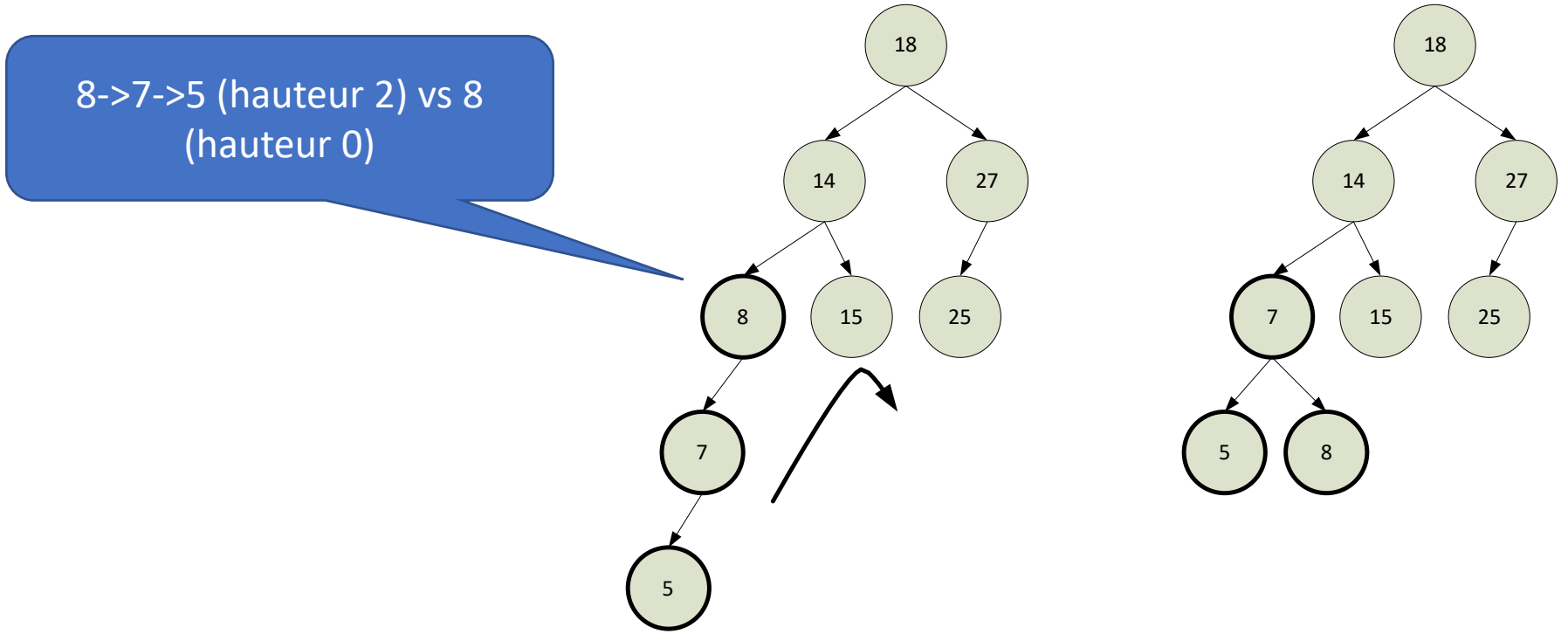
Règles

Règles

- L’objectif d’une rotation est de diminuer d'un la hauteur d’un sous-arbre pour augmenter de la même façon l’autre

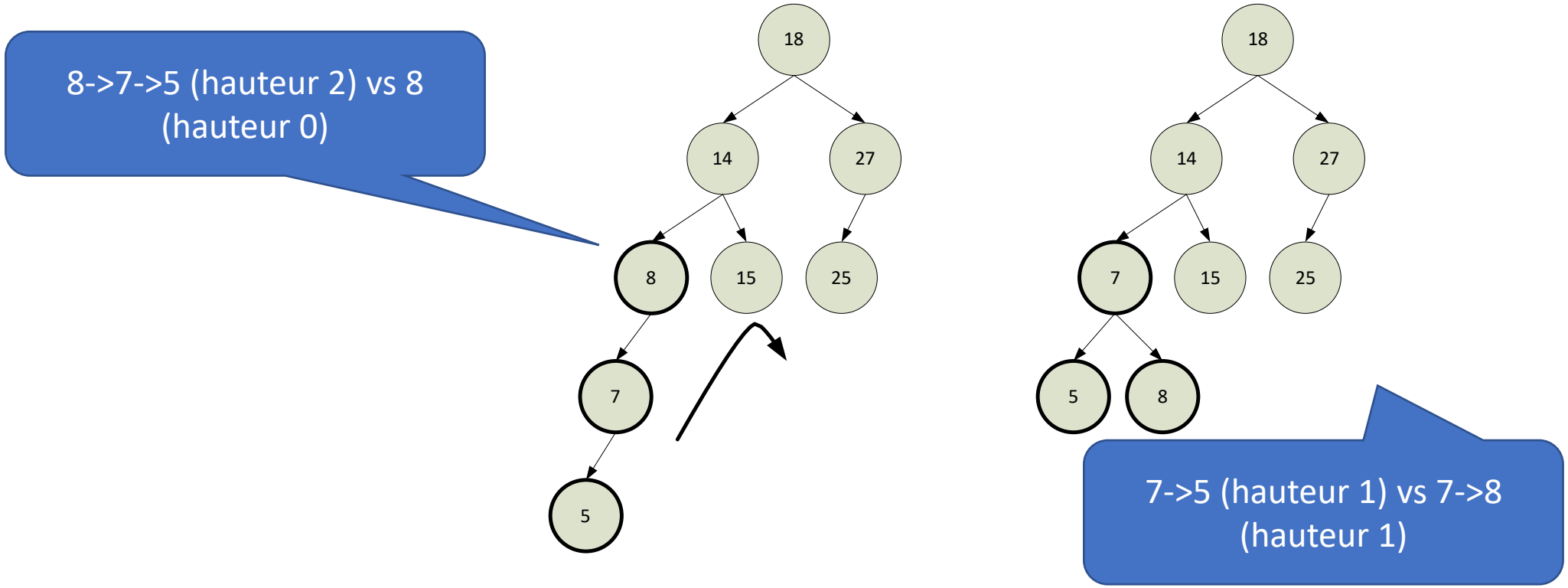
Règles

- L’objectif d’une rotation est de diminuer d'un la hauteur d’un sous-arbre pour augmenter de la même façon l’autre



Règles

- L’objectif d’une rotation est de diminuer d'un la hauteur d’un sous-arbre pour augmenter de la même façon l’autre

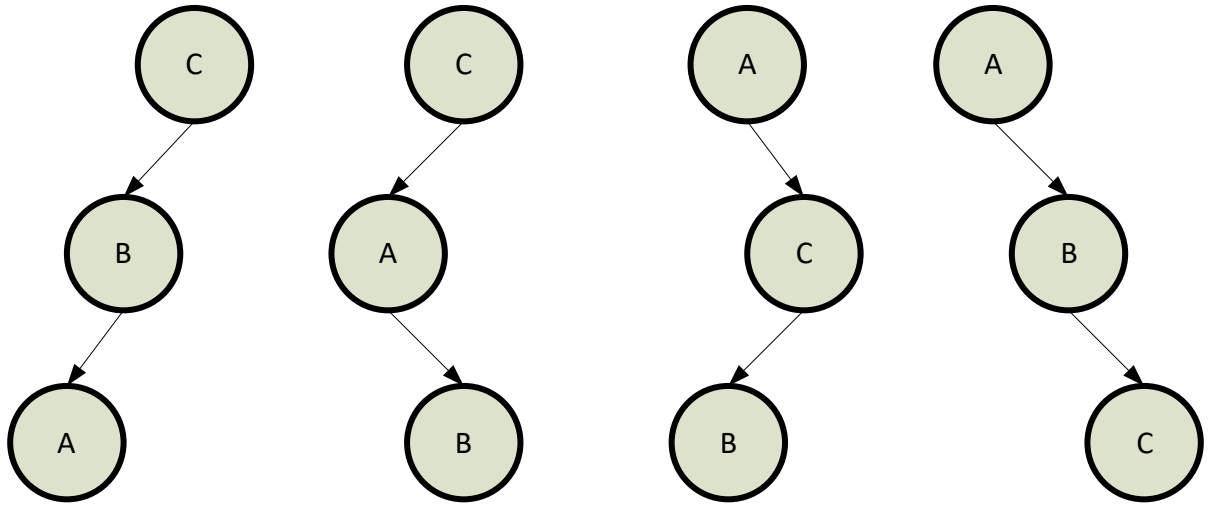


Règles

- Il y a quatre situation différentes à traiter

Règles

- Il y a quatre situation différentes à traiter

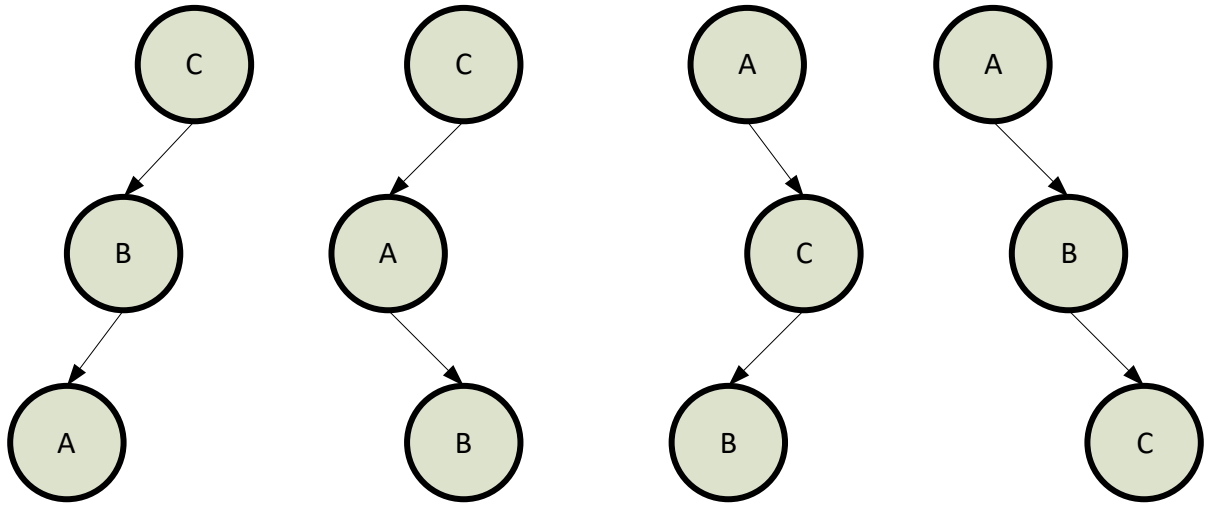


Règles

- Il y a quatre situation différentes à traiter
- On peut reconnaitre dans quelle situation on se trouve en examinant l'indice d'équilibre des nœuds
- **L'indice d'équilibre d'un noeud donné est la différence entre la hauteur de son sous arbre de gauche et celle de son sous arbre de droite**
  - Cet indice ne peut être que -1, 0 ou 1 (on garde le signe pour faciliter le choix des rotations)
  - Si cet indice est 2 ou -2, il y a déséquilibre

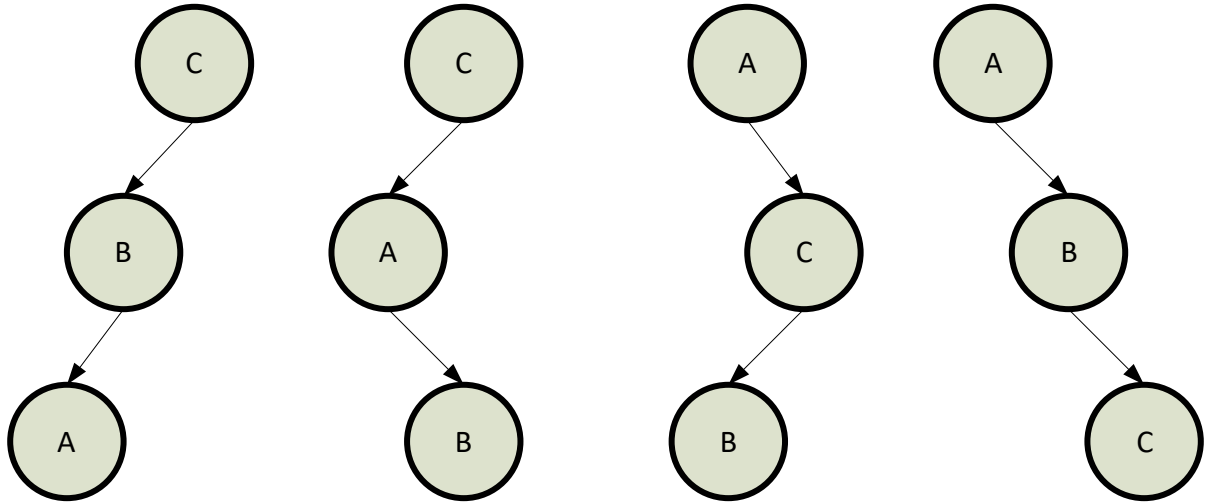
Règles

- Tous ces cas sont déséquilibrés (indice d'équilibre de A est 2 ou -2 selon les cas)



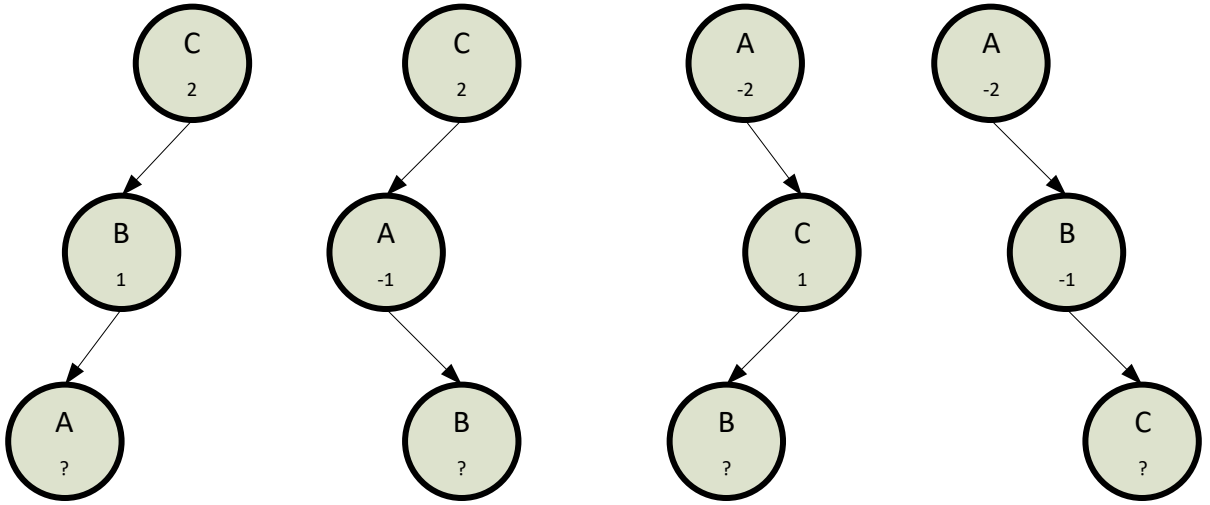
Règles

- Il y a quatre situation différentes à traiter
- Pour savoir dans laquelle des quatre situations on se trouve, on doit regarder l'indice d'un des sous arbres, ce qui détermine définitivement les rotations à effectuer



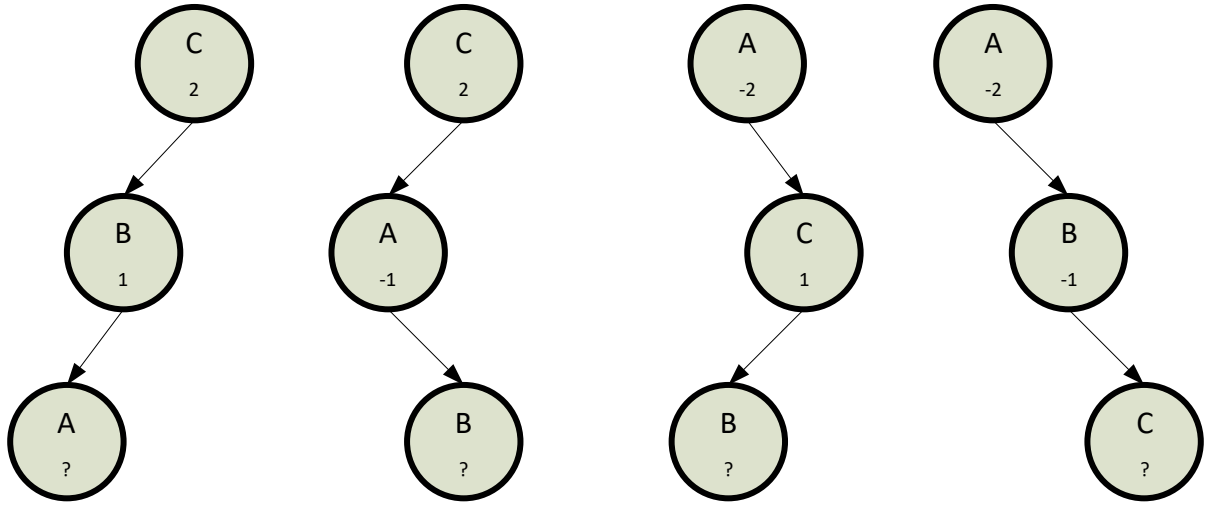
Règles

- Il y a quatre situation différentes à traiter
- Pour savoir dans laquelle des quatre situations on se trouve, on doit regarder l'indice d'un des sous arbres, ce qui détermine définitivement les rotations à effectuer



Règles

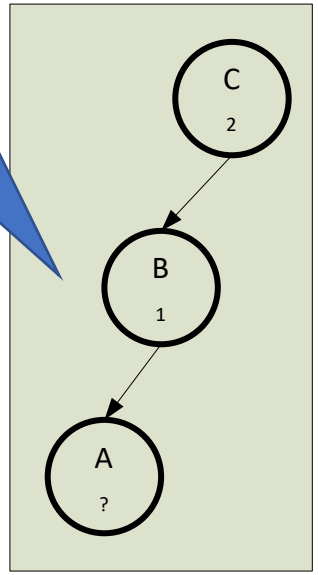
- Il y a toujours deux noeuds qui dominent une rotation
  - D’abord le noeud déséquilibré lui-même
  - Ensuite, celui de ses deux enfants qui est la racine de son sous-arbre le plus haut

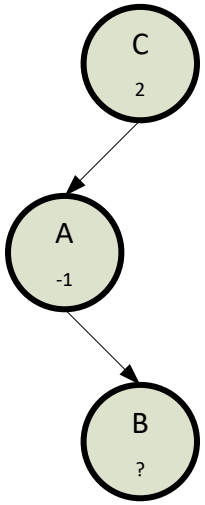


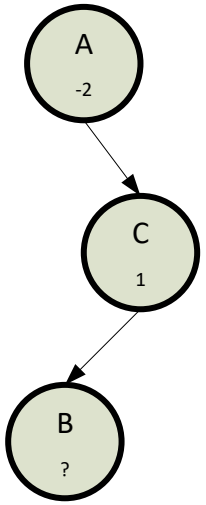
Règles

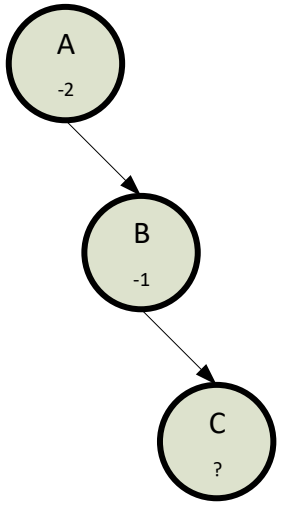
- Il y a toujours deux noeuds qui dominent une rotation
  - D’abord le noeud déséquilibré lui-même
  - Ensuite, celui de ses deux enfants qui est la racine de son sous-arbre le plus haut

Ici, le noeud principal est C et B est son enfant du côté le plus haut





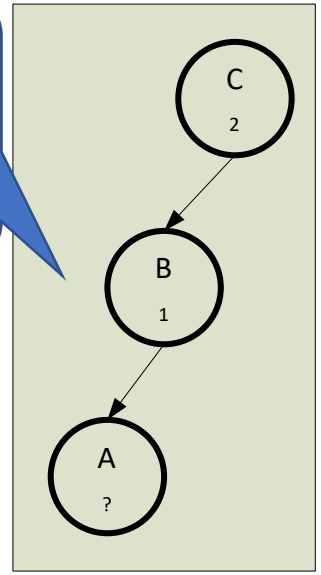


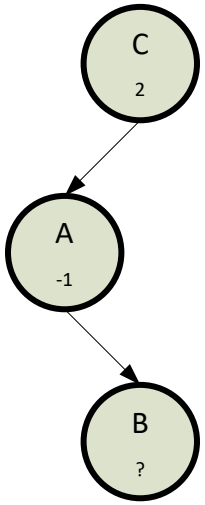


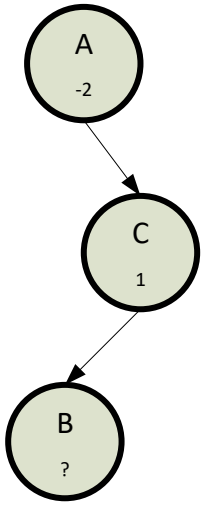
Règles

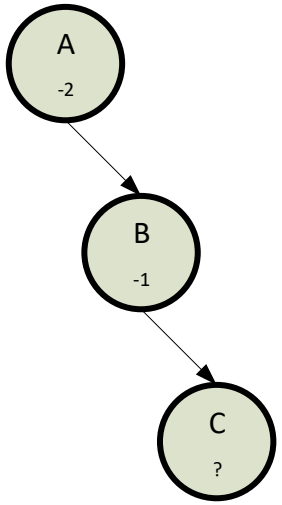
- Il y a toujours deux noeuds qui dominent une rotation
  - D'abord le noeud déséquilibré lui-même
  - Ensuite, celui de ses deux enfants qui est la racine de son sous-arbre le plus haut

Puisque les indices de C et B sont de même signe, une seule rotation sera nécessaire





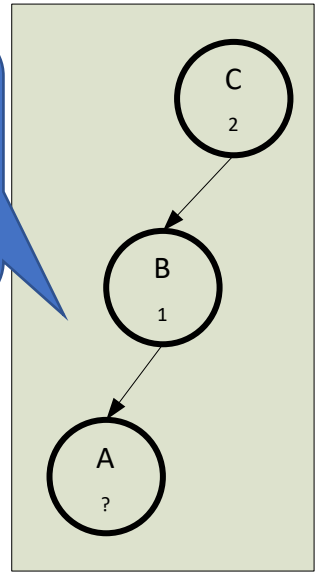


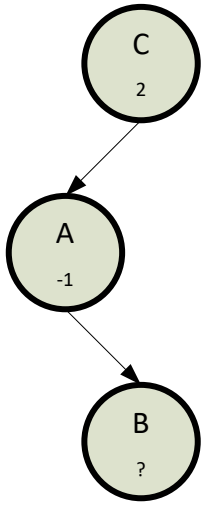


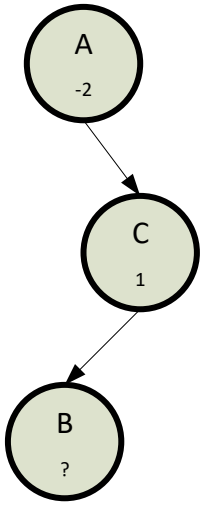
Règles

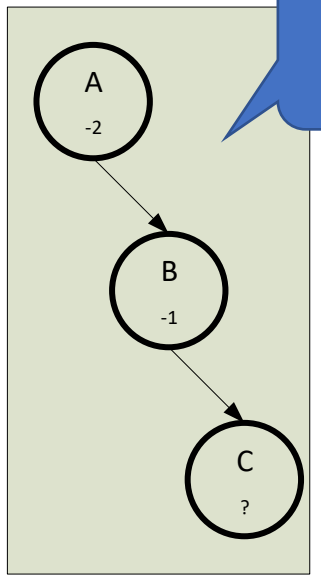
- Il y a toujours deux noeuds qui dominent une rotation
  - D'abord le noeud déséquilibré lui-même
  - Ensuite, celui de ses deux enfants qui est la racine de son sous-arbre le plus haut

Puisque les indices de C et B sont de même signe, une seule rotation sera nécessaire





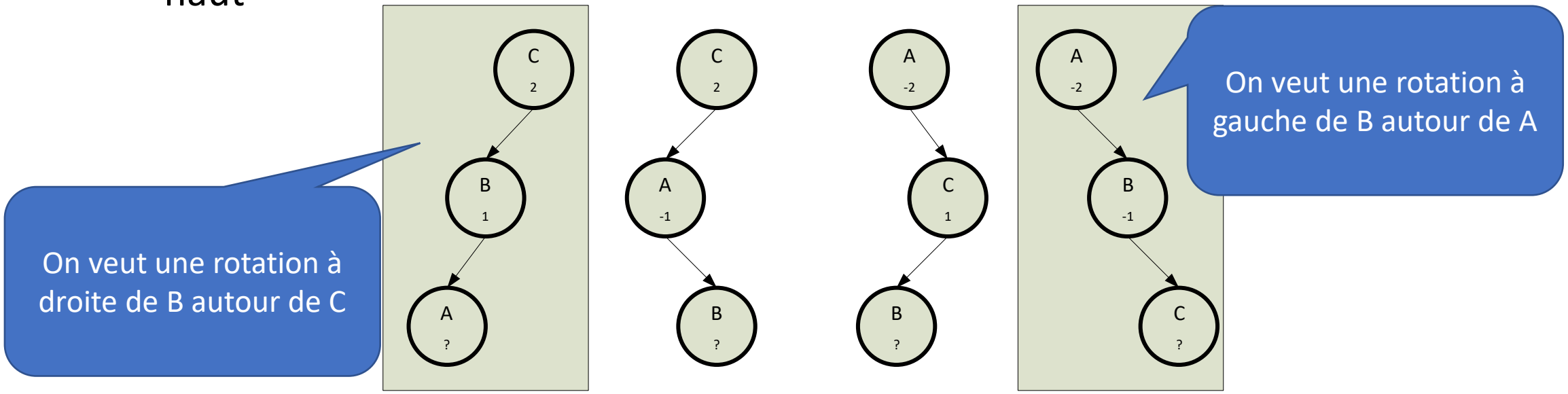




Idem dans ce cas-ci

Règles

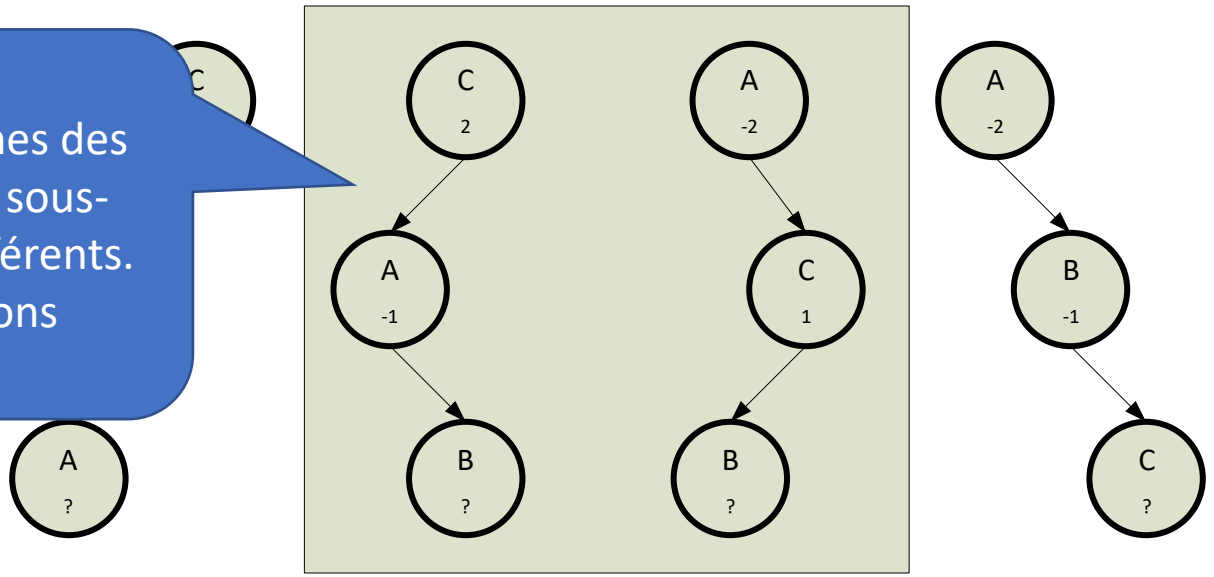
- Il y a toujours deux noeuds qui dominent une rotation
  - D’abord le noeud déséquilibré lui-même
  - Ensuite, celui de ses deux enfants qui est la racine de son sous-arbre le plus haut



Règles

- Il y a toujours deux noeuds qui dominent une rotation
  - D'abord le noeud déséquilibré lui-même
  - Ensuite, celui de ses deux enfants qui est la racine de son sous-arbre le plus haut

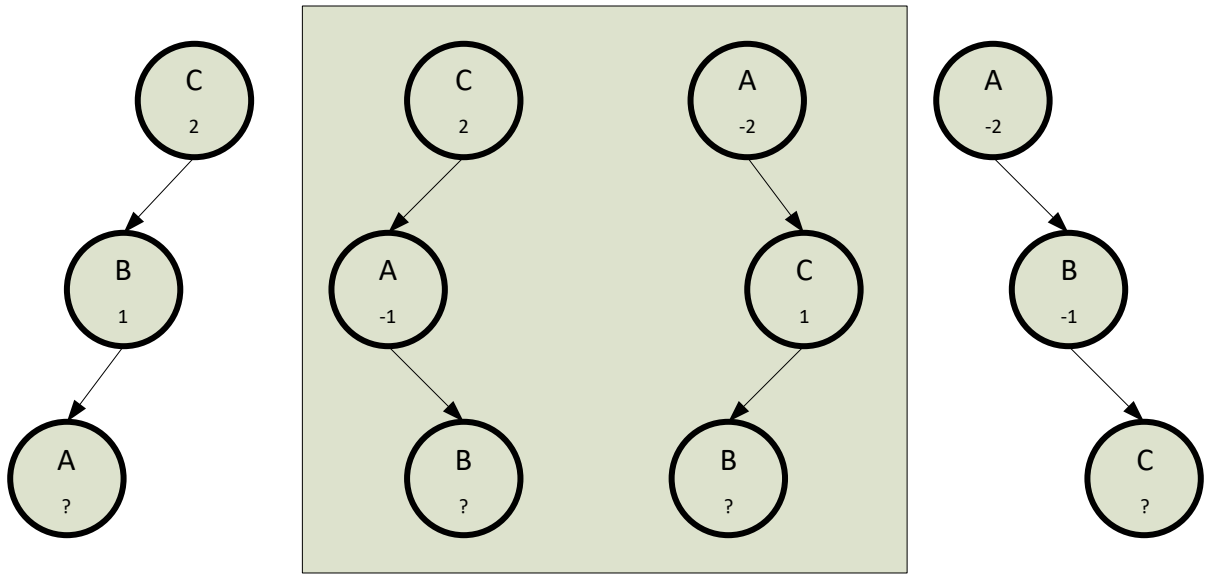
Dans ces deux cas, les signes des indices de la racine et du sous-arbre le plus haut sont différents. On voudra deux rotations



Règles

- Il y a touj minent une rotation
  - D'abor ême
  - Ensuite, celui de ses deux nts qui est la racine de son sous-arbre le plus haut

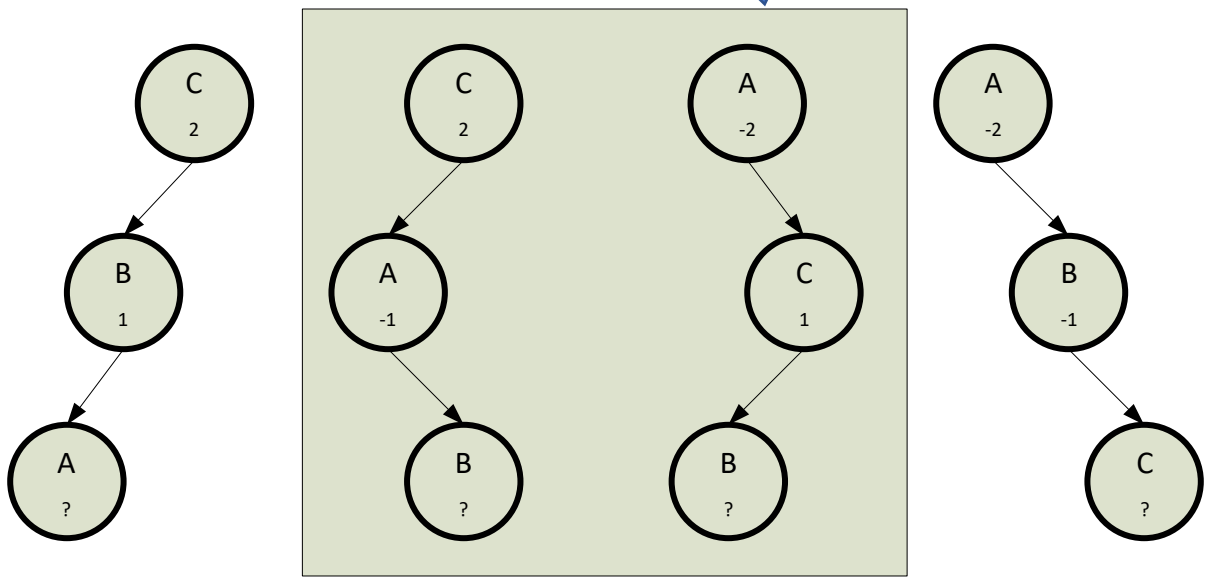
Ici, une rotation à droite de B autour de A, ce qui nous ramène à l'exemple de gauche, puis une rotation à gauche autour de C



Règles

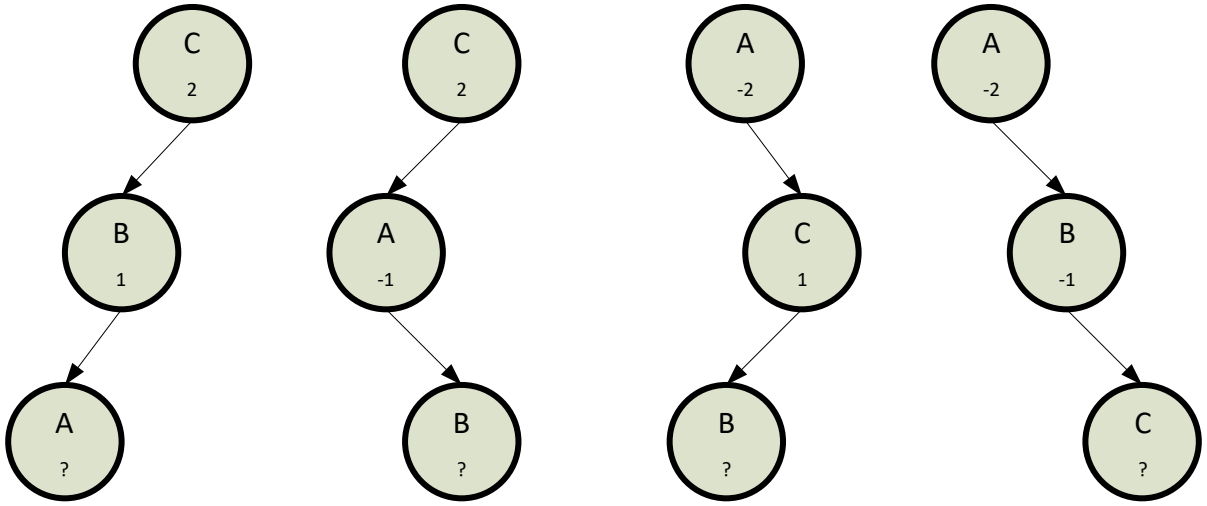
- Il y a toujours deux noeuds qui
  - D'abord le noeud déséquilibré lu
  - Ensuite, celui de ses deux enfants qui e acine de son sous-arbre le plus haut

Ici, une rotation à gauche de B autour de C, ce qui nous ramène à l'exemple de droite, puis une rotation à droite autour de A

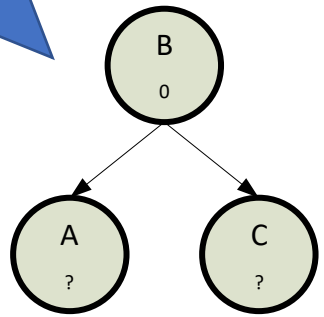


Règles

- Il y a toujours deux noeuds qui dominent une rotation
  - D'abord le noeud déséquilibré lui-même
  - Ensuite, celui de ses deux enfants qui est la ra le plus haut



Dans tous les cas, après les rotations, on obtient ceci



Remarques

Remarques

- Un arbre AVL est un arbre binaire de recherche dont chaque sous-arbre est aussi un arbre AVL

Remarques

- Un arbre AVL est un arbre binaire de recherche dont chaque sous-arbre est aussi un arbre AVL
- Une telle caractéristique d'un type est ce qu'on nomme un **invariant**

Remarques

- Un arbre AVL est un arbre binaire de recherche dont chaque sous-arbre est aussi un arbre AVL
- Une telle caractéristique d'un type est ce qu'on nomme un **invariant**
- **Un invariant d'un type T est une caractéristique qui s'avère pour toute instance de T entre deux appels de fonctions**

Remarques

- Un arbre AVL est un arbre binaire de recherche dont chaque sous-arbre est aussi un arbre AVL
- Une telle caractéristique d'un type est ce qu'on nomme un **invariant**
- **Un invariant d'un type T est une caractéristique qui s'avère pour toute instance de T entre deux appels de fonctions**
- **L'encapsulation est la capacité qu'a un type T de garantir le respect de ses invariants**

Remarques

- Ce ne sont pas tous les types qui ont des invariants à faire respecter

Remarques

- Ce ne sont pas tous les types qui ont des invariants à faire respecter
  - Un point du plan cartésien n’a pas d’invariants à garantir

Remarques

- Ce ne sont pas tous les types qui ont des invariants à faire respecter
  - Un point du plan cartésien n'a pas d'invariants à garantir
  - Un point {x,y} du quadrant 1 du plan cartésien, par contre, doit garantir que
    - x>=0 && y>=0

Remarques

- Insérer un nœud dans un arbre AVL semble susceptible de briser ses invariants…

Remarques

- Insérer un nœud dans un arbre AVL semble susceptible de briser ses invariants…
  - … mais il n’en est rien

Remarques

- Insérer un nœud dans un arbre AVL semble susceptible de briser ses invariants…
  - … mais il n’en est rien
- Un invariant d’un type T est une caractéristique qui s’avère pour toute instance de T entre deux appels de fonctions

Remarques

- Insérer un nœud dans un arbre AVL semble susceptible de briser ses invariants…
  - … mais il n’en est rien
- Un invariant d’un type T est une **caractéristique qui s’avère pour toute instance de T entre deux appels de fonctions**

Remarques

- Insérer un nœud dans un arbre AVL semble susceptible de briser ses invariants…
  - … mais il n’en est rien
- Un invariant d’un type T est une **caractéristique qui s’avère pour toute instance de T entre deux appels de fonctions**
- Il est normal de briser des invariants pendant l’exécution d’une fonction…

Remarques

- Insérer un nœud dans un arbre AVL semble susceptible de briser ses invariants…
  - … mais il n'en est rien
- Un invariant d'un type T est une **caractéristique qui s'avère pour toute instance de T entre deux appels de fonctions**
- Il est normal de briser des invariants pendant l'exécution d'une fonction…
  - …dans la mesure où ce bris est invisible au code client, et où il est réparé avant que la fonction ne se complète