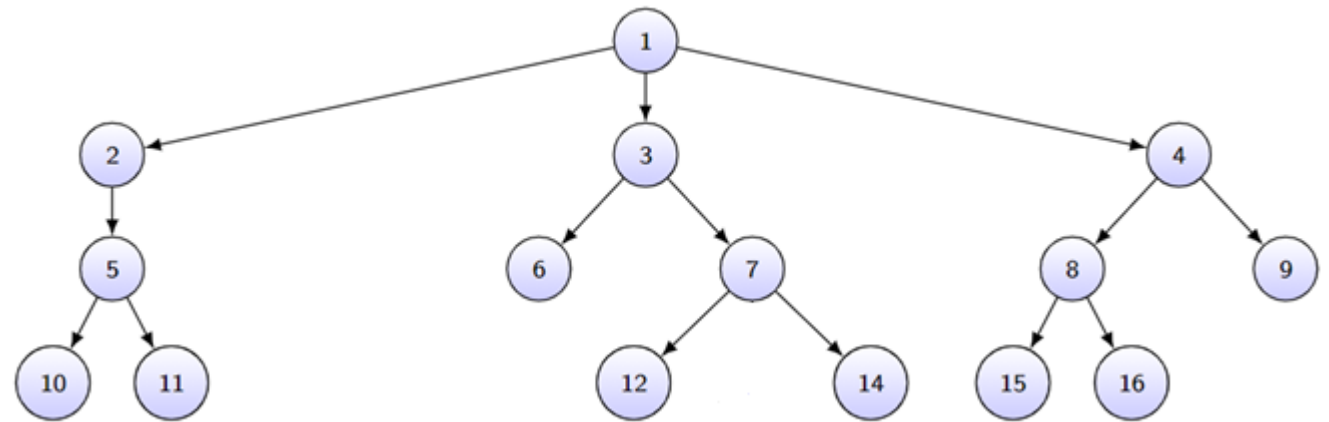


# INFORMATIQUE 3

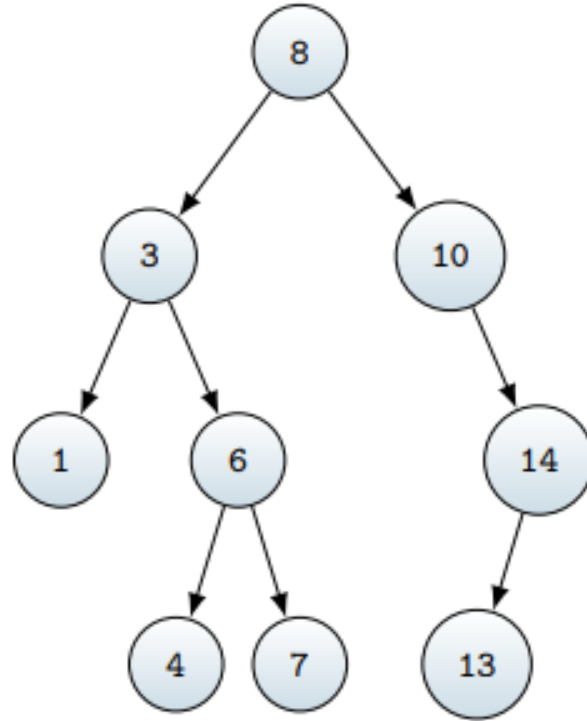
V. AVL



# I. Optimisation de la recherche dans un ABR

# ABR : Rappel

- L'arbre binaire de recherche est un type arbre binaire permettant d'optimiser la recherche d'élément dans l'arbre. Il obéit à des règles de construction.

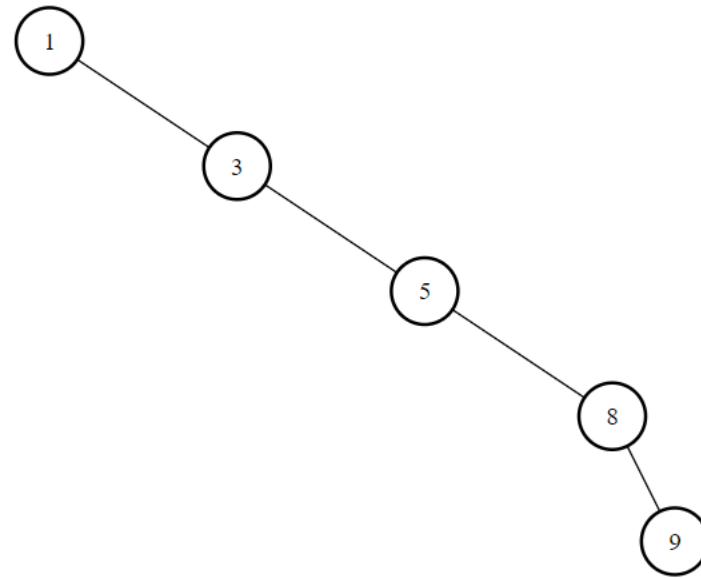


# Configuration de l'ABR

- Construire un ABR en insérant les éléments suivant dans l'ordre : 1 , 3 , 5 , 8, 9

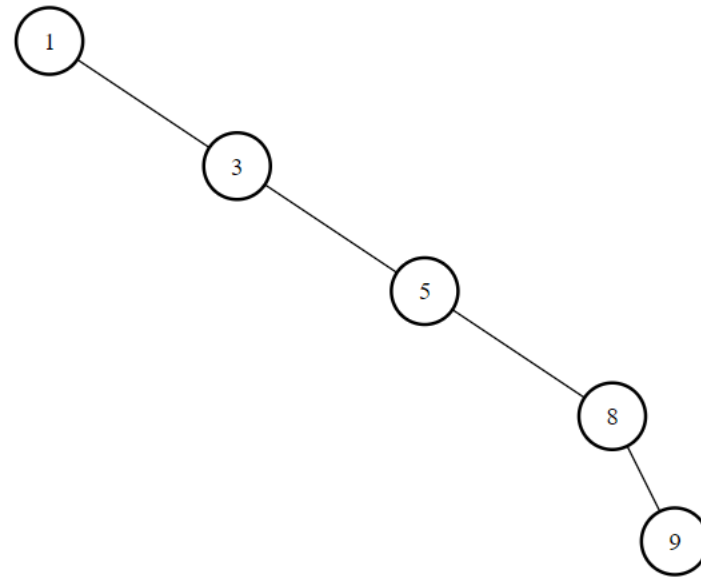
# Configuration de l'ABR

- Construire un ABR en insérant les éléments suivant dans l'ordre : 1 , 3 , 5 , 8 , 9



# Configuration de l'ABR

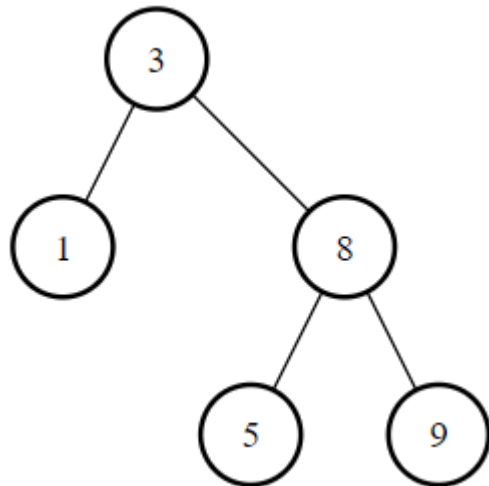
- Construire un ABR en insérant les éléments suivant dans l'ordre : 1 , 3 , 5 , 8 , 9



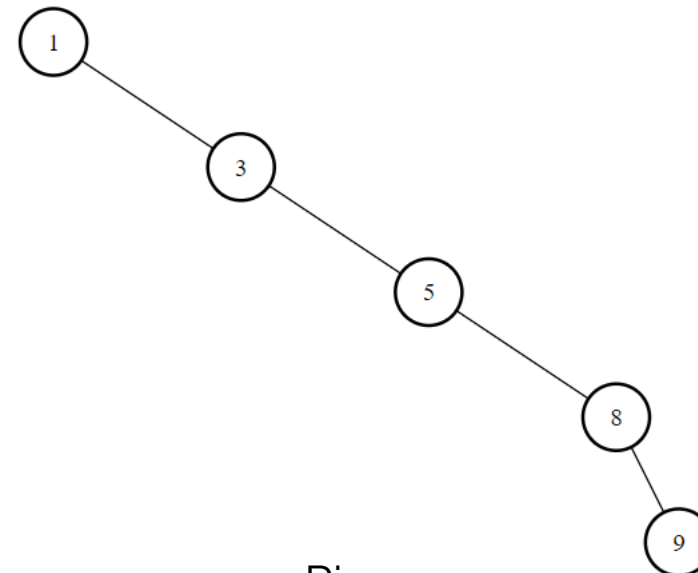
- Dans un arbre filiforme, la complexité de l'algorithme de recherche est en  $O(n)$

# Complexité et équilibre

- Complexité: Le nombre d'appels récurifs pour la recherche ou les autres opérations dépend de la configuration de l'ABR.
- Dans le pire des cas, l'arbre est filiforme, l'opération est en  $O(n)$ .
- Dans le meilleur des cas, l'arbre est *équilibré* entre sa partie droite et gauche ( $O(\log_2(n))$ )



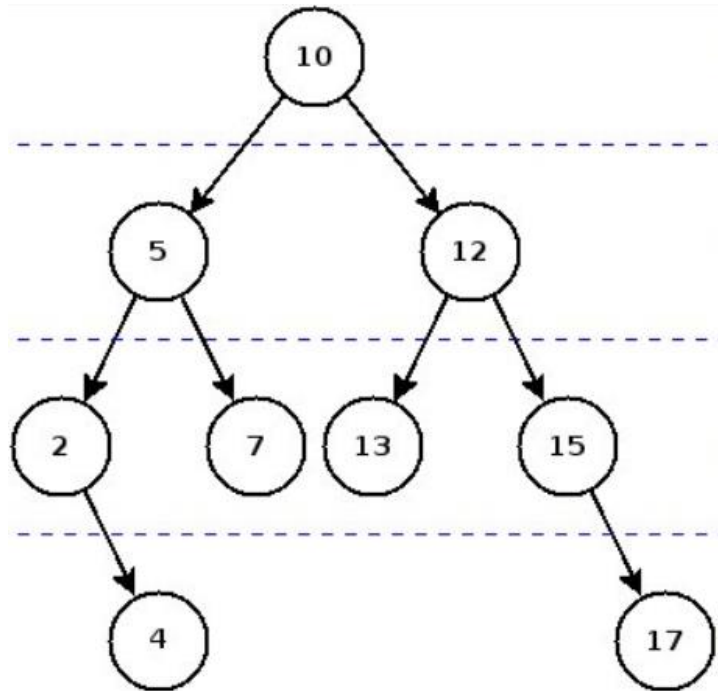
Meilleur cas



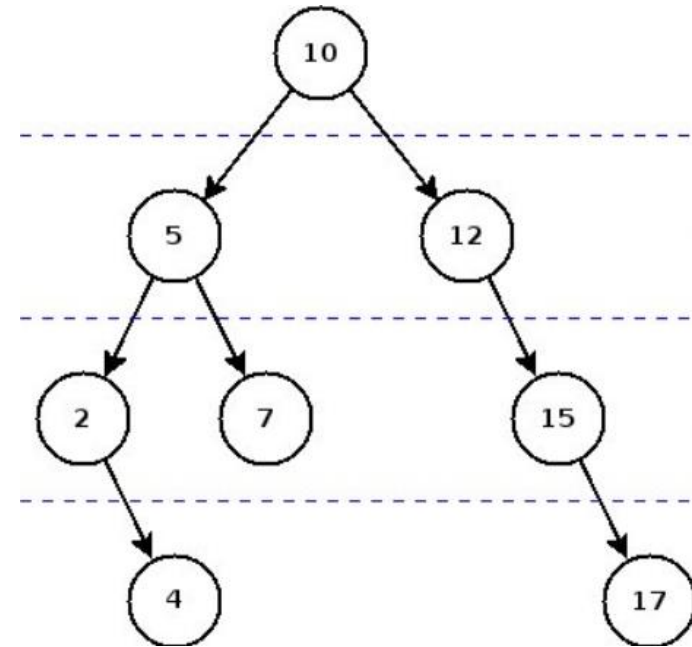
Pire cas

# Arbre équilibré

- Un **arbre équilibré** est un arbre qui maintient une profondeur équilibrée entre ses branches. Chaque nœud interne a le nombre maximum de fils.



Arbre équilibré



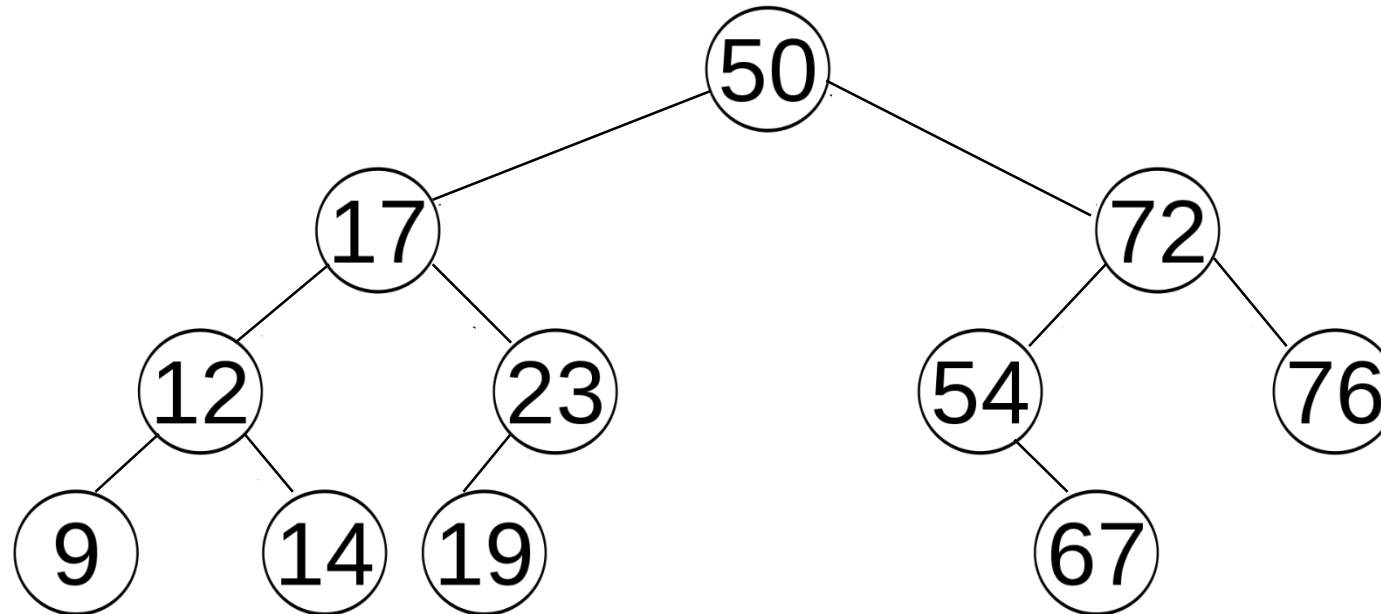
Arbre non-équilibré ou dégénéré



## II. AVL : introduction

# AVL : définition

- Un **AVL**, arbre binaire de recherche automatiquement équilibré, est un **arbre binaire de recherche « auto-équilibré »**: il reste équilibré après opération.
- Une série d'ajouts ou de suppressions dans un ABR peut déséquilibrer l'arbre et conduire à des opérations de complexité  $O(n)$  dans le pire des cas. Les arbres AVL utilisent des algorithmes de rééquilibrage pour éviter cela.

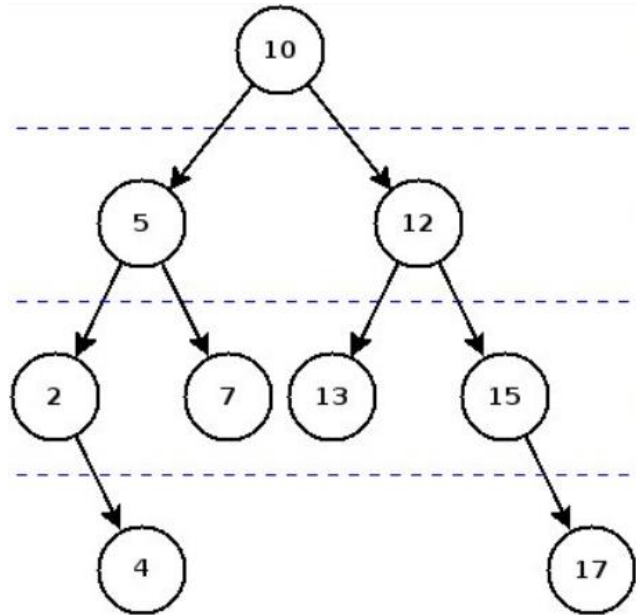


# Facteur d'équilibrage

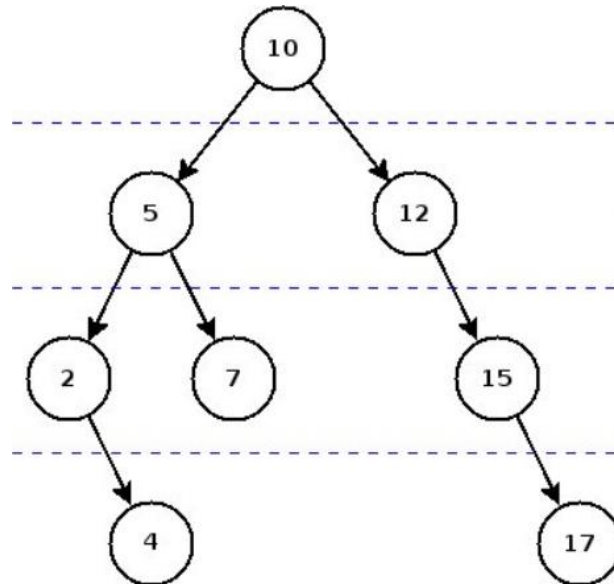
- Pour chaque nœud, on considère la hauteur du nœud.
- Lors d'une opération de transformation, la hauteur d'un nœud peut être modifiée. On calcule alors un facteur d'équilibre de l'arbre :

**hauteur du sous arbre droit - hauteur du sous arbre gauche**

- Il est recalculé après chaque opération
- Il permet de définir si un rééquilibrage est nécessaire



Arbre équilibré



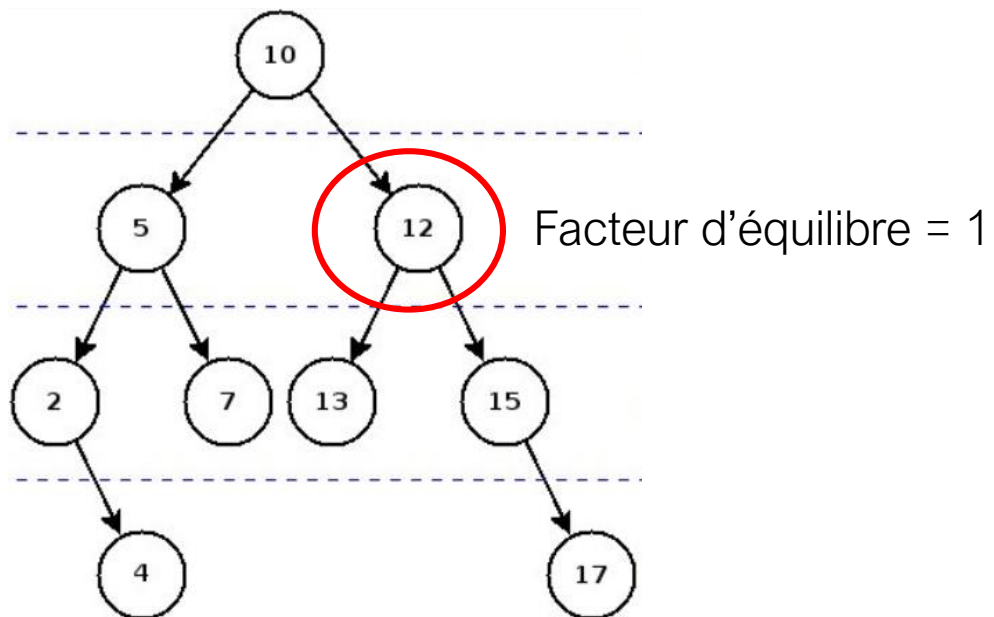
Arbre non-équilibré ou dégénéré

# Facteur d'équilibrage

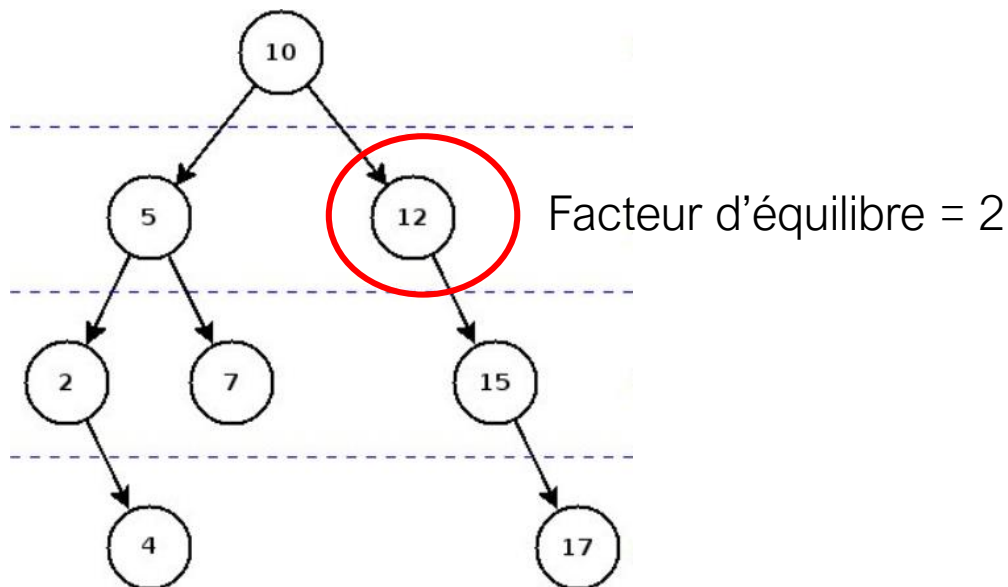
- Pour chaque nœud, on considère la hauteur du nœud
- Lors d'une opération de transformation, la hauteur d'un nœud peut être modifiée On calcule alors un facteur d'équilibre de l'arbre :

**hauteur du sous arbre droit - hauteur du sous arbre gauche**

- Il est recalculé après chaque opération
- Il permet de définir si un rééquilibrage est nécessaire



Arbre équilibré



Arbre non-équilibré ou dégénéré

# Facteur d'équilibrage

- Pour chaque nœud, on considère la hauteur du nœud
  - Lors d'une opération de transformation, la hauteur d'un nœud peut être modifiée On calcule alors un facteur d'équilibre de l'arbre :  
**hauteur du sous arbre droit - hauteur du sous arbre gauche**
  - Il est recalculé après chaque opération
  - Il permet de définir si un rééquilibrage est nécessaire
- 
- Un arbre est équilibré  $\Leftrightarrow$   $|\text{facteur d'équilibre}| < 2$  pour tous les nœuds.

# Construction d'un AVL

- Structure d'un nœud d'un AVL:

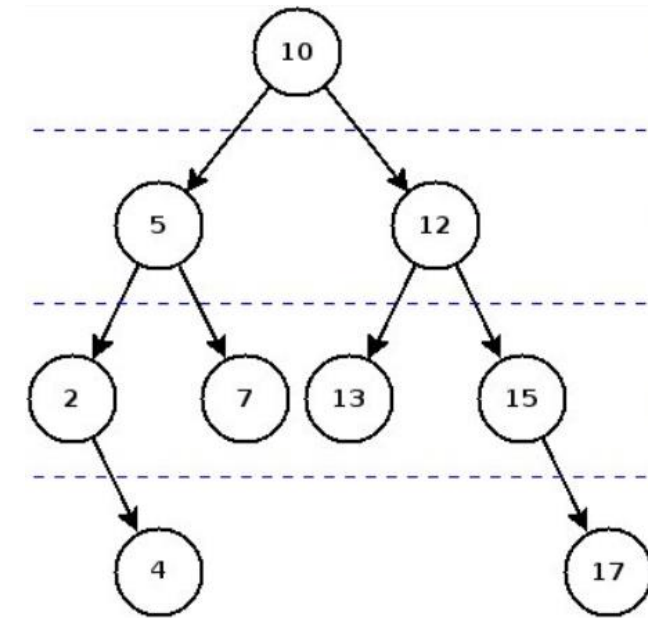
Structure Arbre :

elmt : Element // contenu du nœud

fg, fd : pointeur sur structure Arbre // fils gauche et droit

equilibre : entier // facteur d'équilibre du nœud = hauteur sous arbre droit

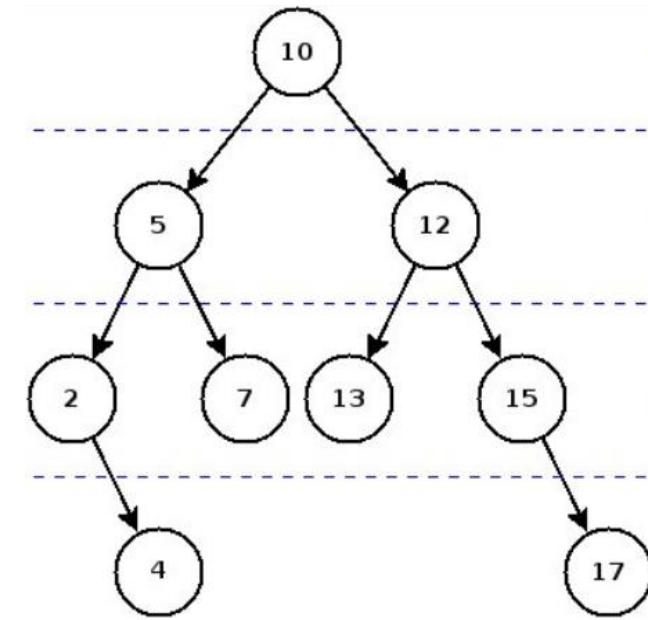
- hauteur sous-arbre gauche.



# Construction d'un AVL

- Déclaration et initialisation d'un nœud

```
FONCTION creerArbre(r: Element) : ptr sur Arbre
VARIABLE
    noeud : ptr sur Arbre
DEBUT
    noeud ← reserverMemoire(Arbre)
    elmt(noeud) ← r
    fg(noeud) ← ??
    fd(noeud) ← ??
    equilibre(noeud) ← ??
    RETOURNER noeud
FIN
```



# Construction d'un AVL

- Déclaration et initialisation d'un nœud

```
FONCTION creerArbre(r: Element) : ptr sur Arbre
```

```
VARIABLE
```

```
  noeud : ptr sur Arbre
```

```
DEBUT
```

```
  noeud ← reserverMemoire(Arbre)
```

```
  elmt(noeud) ← r
```

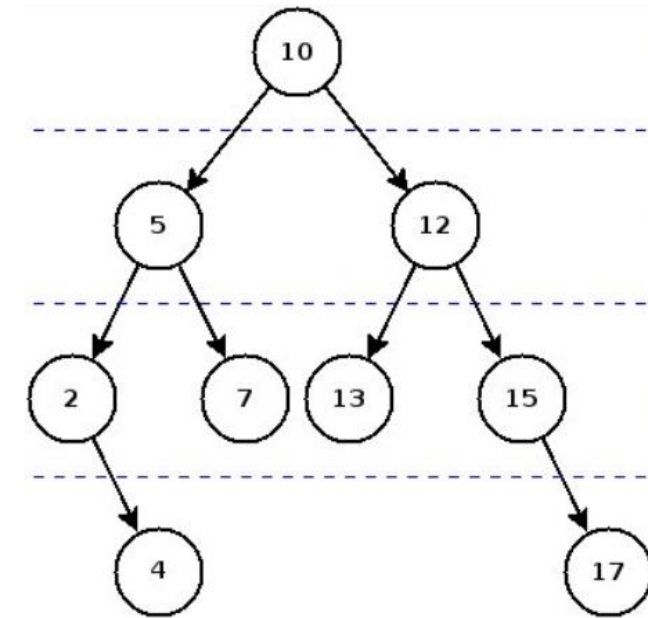
```
  fg(noeud) ← NULL // Le noeud n'a pas de fils à sa création
```

```
  fd(noeud) ← NULL
```

```
  equilibre(noeud) ← 0 // Le noeud n'a pas de fils: son equilibre est 0
```

```
  RETOURNER noeud
```

```
FIN
```





### III. AVL : opérations

# Opération de recherche

- Puisque l'AVL respecte les propriétés d'un ABR, l'algorithme de recherche ne change pas :
  - Recherche dans le sous-arbre gauche si la valeur recherchée est inférieure au nœud
  - Recherche dans le sous-arbre droit si la valeur recherchée est supérieure au nœud

FONCTION recherche(a: pointeur sur Arbre, e: Element) :  
pointeur sur Arbre

DEBUT

SI (a EST EGAL A NULL) Alors  
RETOURNER NULL

SINON SI (element(a) EST EGAL A e) Alors  
RETOURNER a

SINON SI (e EST INF. STRICT. A element(a)) ALORS  
RETOURNER recherche(fg(a),e)

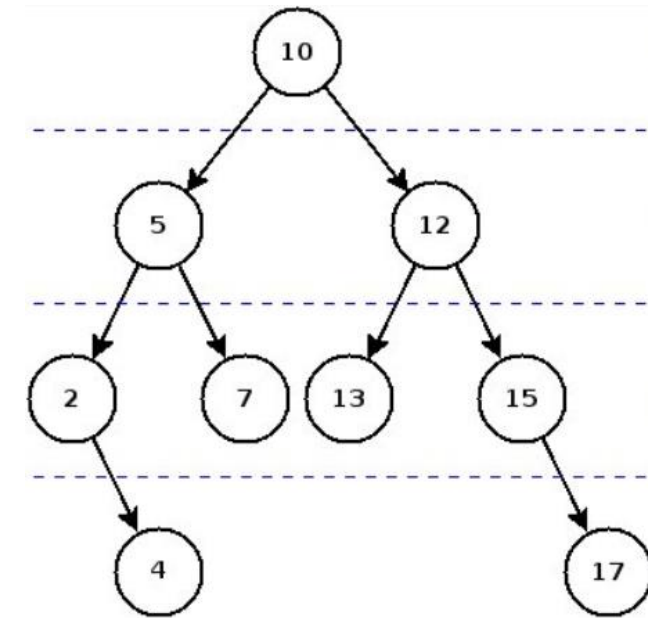
SINON

RETOURNER recherche(fd(a),e)

FIN SI

Fin

- L'opération de recherche ne modifie pas l'arbre : pas besoin de rééquilibrage.

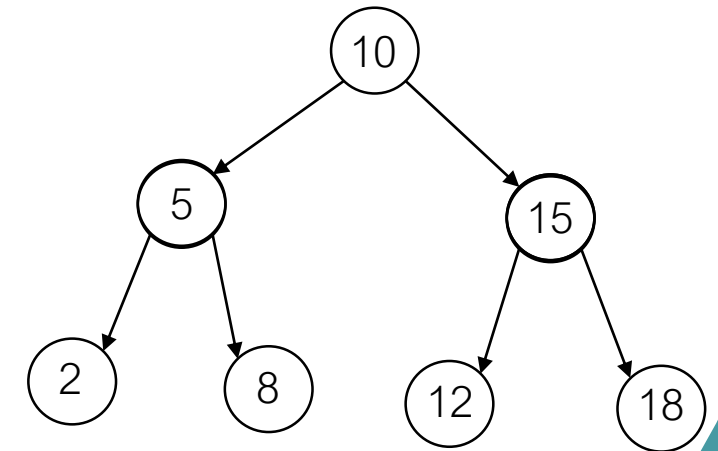


# Insertion

- 1<sup>ère</sup> étape : l'insertion se déroule de la même manière que pour un ABR :
- Algorithme :

```
FONCTION insertionABR(a: pointeur sur Arbre, e: Element) :  
pointeur sur Arbre  
DEBUT  
  SI (a EST EGAL A NULL) ALORS  
    RETOURNER creerArbre(e)  
  SINON SI (e EST INF. STRICT. A element(a)) ALORS  
    fg(a) ← insertionABR(fg(a), e)  
  SINON SI (e SUP. STRICT. A element(a)) Alors  
    fd(a) ← insertionABR(fd(a), e)  
  FIN SI  
  RETOURNER a  
FIN
```

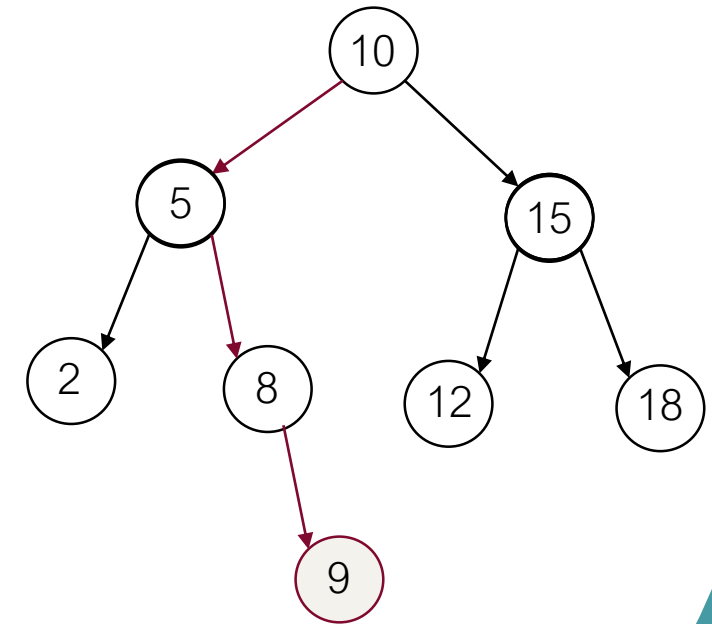
Exemple : insertionABR(A, 9)



# Insertion

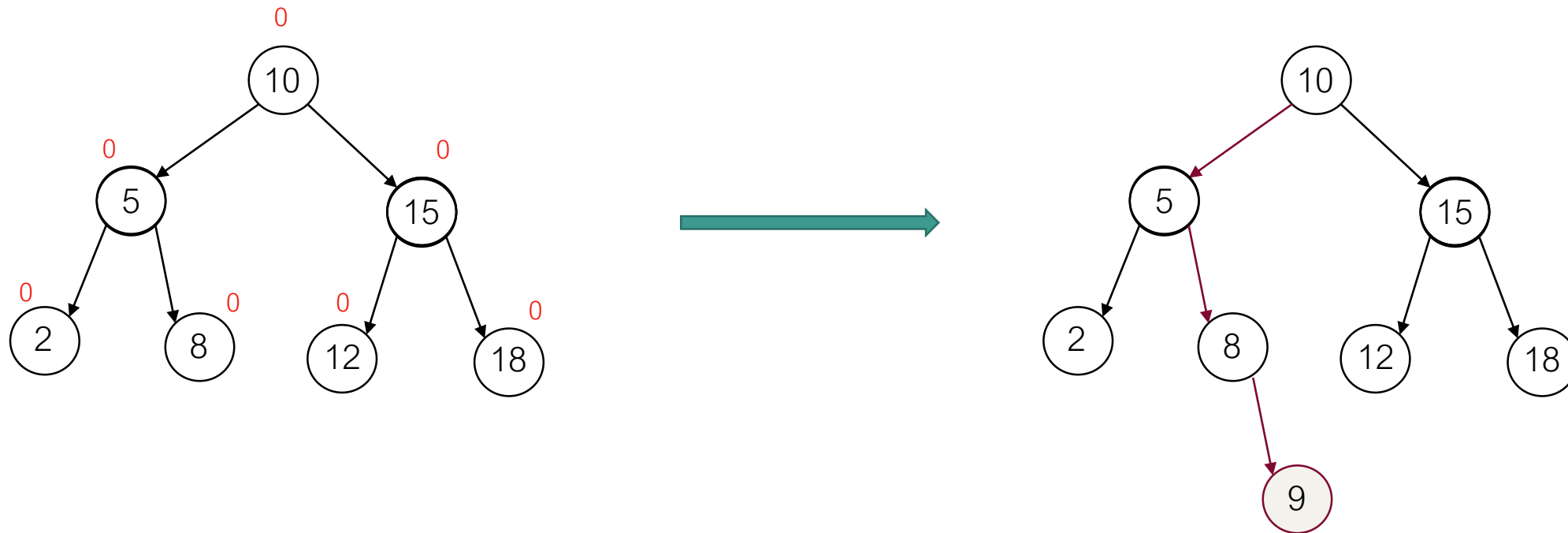
- 1<sup>ère</sup> étape : l'insertion se déroule de la même manière que pour un ABR :
- Algorithme :

```
FONCTION insertionABR(a: pointeur sur Arbre, e: Element) :  
pointeur sur Arbre  
DEBUT  
  SI (a EST EGAL A NULL) ALORS  
    RETOURNER creerArbre(e)  
  SINON SI (e EST INF. STRICT. A element(a)) ALORS  
    fg(a) ← insertionABR(fg(a), e)  
  SINON SI (e SUP. STRICT. A element(a)) Alors  
    fd(a) ← insertionABR(fd(a), e)  
  FIN SI  
  RETOURNER a  
FIN
```



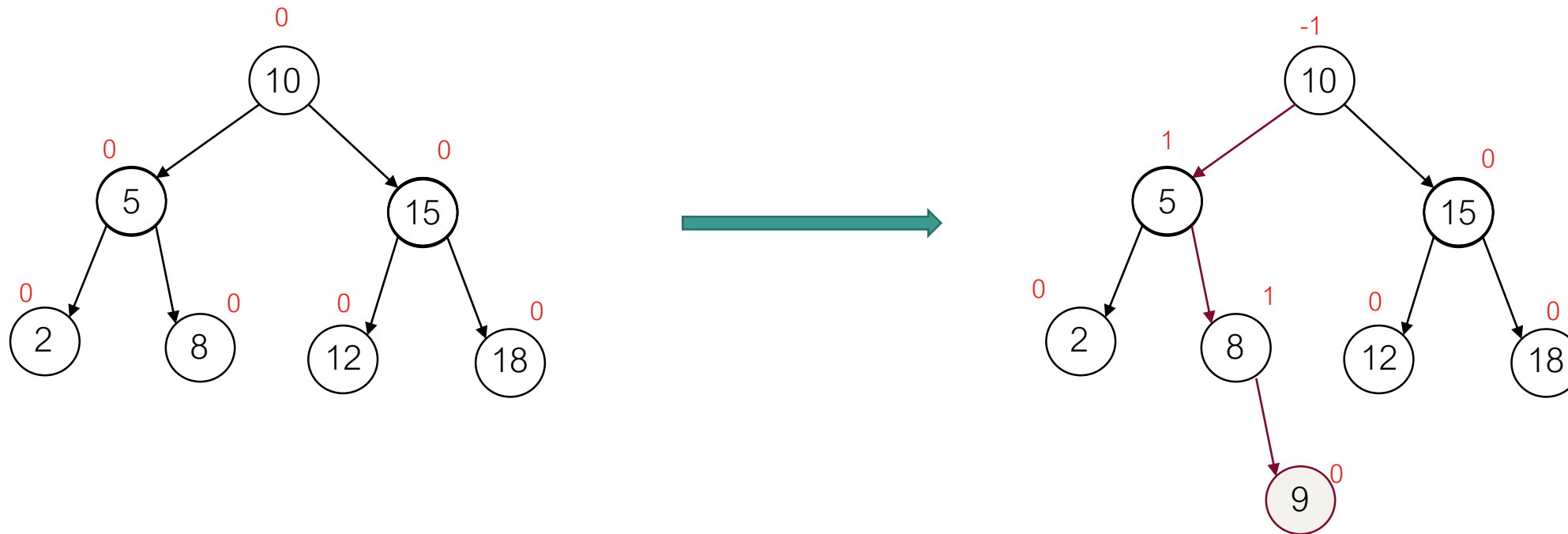
# Insertion

- 2<sup>ème</sup> étape : il faut mettre à jour l'équilibre des nœuds.



# Insertion

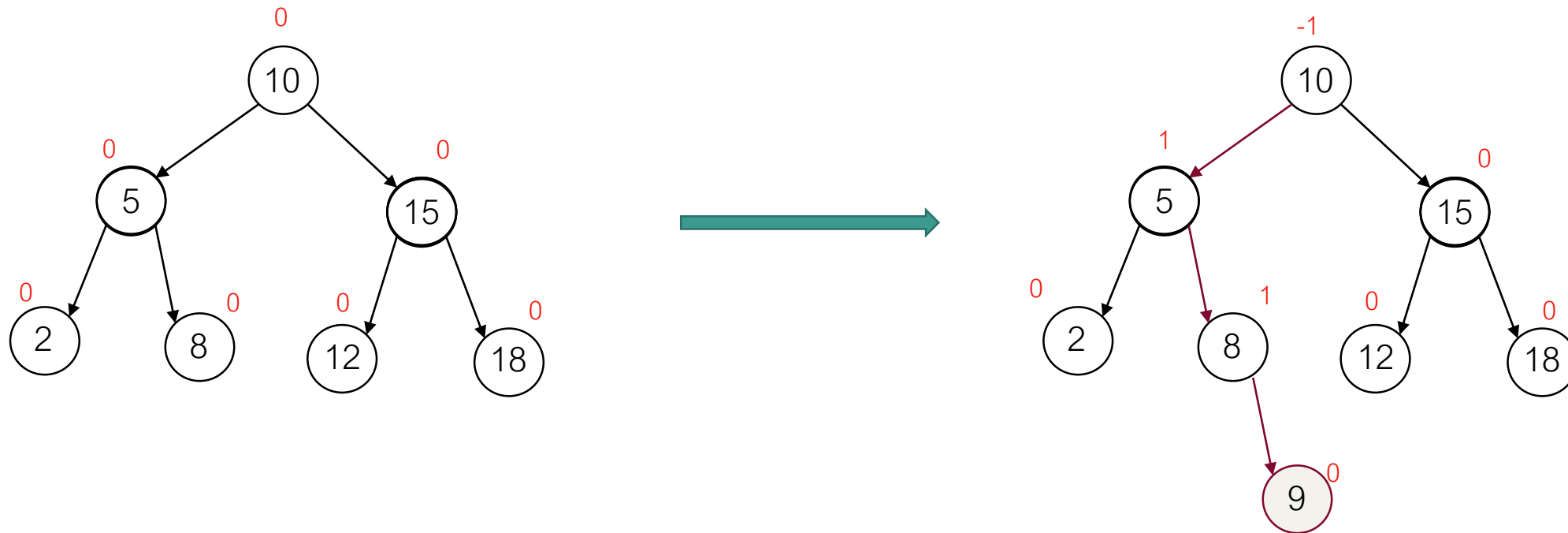
- 2<sup>ème</sup> étape : il faut mettre à jour l'équilibre des nœuds.



- Seuls les nœuds ancêtres ont un facteur d'équilibre modifié.

# Insertion

- 2<sup>ème</sup> étape : il faut mettre à jour l'équilibre des nœuds.



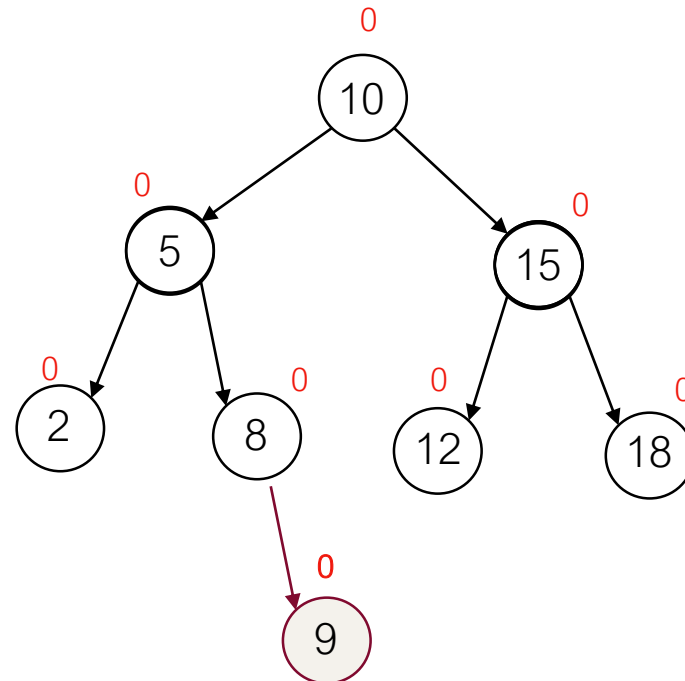
- Dans ce cas, pas besoin de rééquilibrer l'arbre.

# Insertion

- 2<sup>ème</sup> étape : il faut mettre à jour l'équilibre des nœuds.

Principes :

- Le nœud inséré est une feuille : son équilibre est à 0.



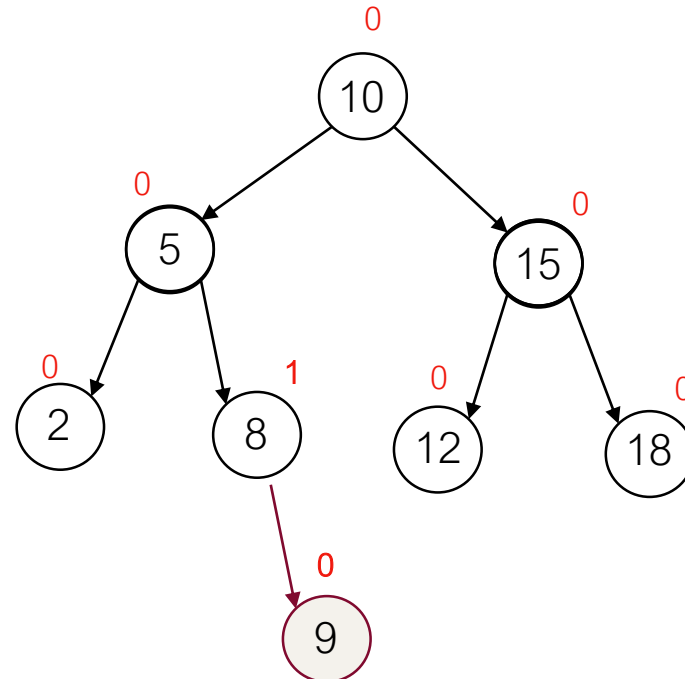


# Insertion

- 2<sup>ème</sup> étape : il faut mettre à jour l'équilibre des nœuds.

Principes :

- Le nœud inséré est une feuille : son équilibre est à 0.
- L'insertion d'un enfant induit un changement dans l'équilibre du nœud parent.
  - -1 si le nouveau nœud est à gauche
  - +1 si le nouveau nœud est à droite

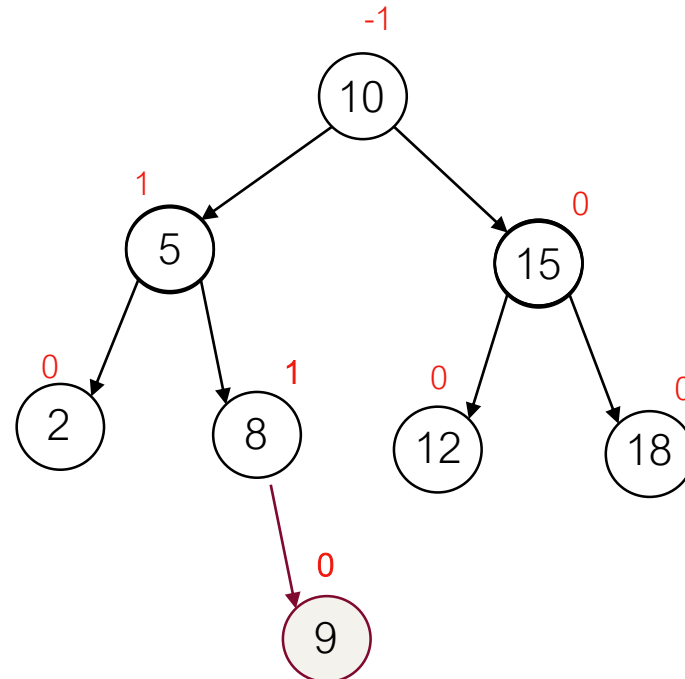


# Insertion

- 2<sup>ème</sup> étape : il faut mettre à jour l'équilibre des nœuds.

Principes :

- Le nœud inséré est une feuille : son équilibre est à 0.
- L'insertion d'un enfant induit un changement dans l'équilibre du nœud parent.
  - -1 si le nouveau nœud est à gauche
  - +1 si le nouveau nœud est à droite
- Ce changement est effectué sur tous les ancêtres



# Insertion

- 2<sup>ème</sup> étape : il faut mettre à jour l'équilibre des nœud → Algorithme :

```
FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre
DEBUT
```

**h : différence d'équilibre**

```
    SI (a EST EGAL A NULL) ALORS
        *h=1
        RETOURNER creerArbre(e)
    SINON Si (e INF. STRICT. A element(a)) ALORS
        fg(a) ← insertionAVL(fg(a), e, h)
        *h ← -*h
    SINON Si (e SUP. STRICT. A element(a)) ALORS
        fd(a) ← insertionAVL(fd(a), e, h)
    SINON
        *h= 0
        RETOURNER a
    FIN SI
    SI (*h DIFFERENT DE 0) ALORS
        equilibre (a) ← equilibre(a) + *h
        SI (equilibre(a) EST EGAL A 0) ALORS
            *h ← 0
        SINON
            *h ← 1
        FIN SI
    FIN SI
    RETOURNER A
FIN
```

// on ajoute un élément : l'équilibre va changer

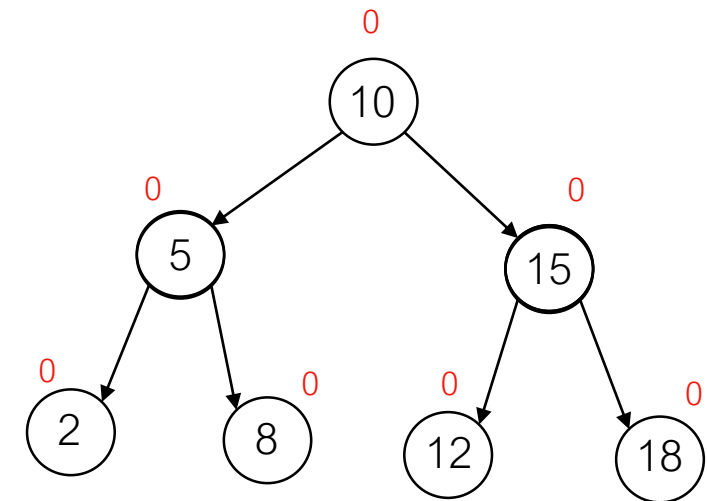
// difference de -1 si on insère à gauche

// cas où l'élément est déjà dans l'arbre  
// pas de changement d'équilibre

// s'il y a changement...  
// ... mise à jour du facteur d'équilibre  
// si arbre équilibré à nouveau...  
// ses ancêtres ne changent pas

# Insertion

- Exemple : insertionAVL(a, 9)



# Insertion

- Exemple : insertionAVL(a, 9)

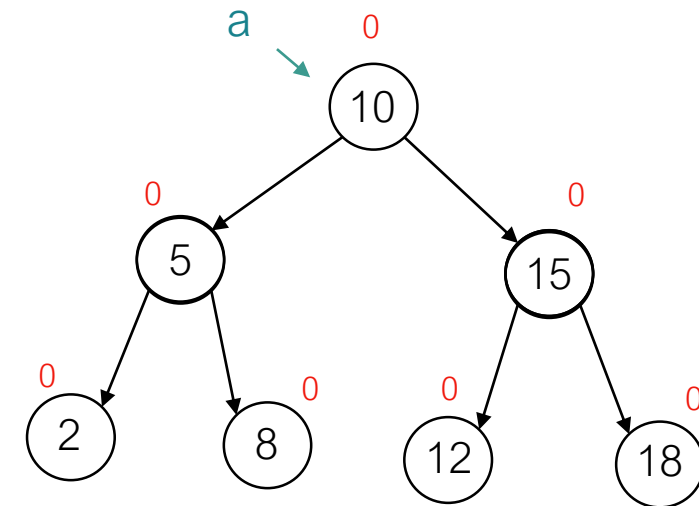
FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

→ DEBUT

```
SI (a EST EGAL A NULL) ALORS
  *h=1
  RETOURNER creerArbre(e)
SINON Si (e INF. STRICT. A element(a)) ALORS
  fg(a) ← insertionAVL(fg(a), e, h)
  *h ← -*h
SINON Si (e SUP. STRICT. A element(a)) ALORS
  fd(a) ← insertionAVL(fd(a), e, h)
SINON
  *h= 0
  RETOURNER a
FIN SI
SI (*h DIFFERENT DE 0) ALORS
  equilibre (a) ← equilibre(a) + *h
  SI (equilibre(a) EST EGAL A 0) ALORS
    *h ← 0
  SINON
    *h ← 1
  FIN SI
FIN SI
RETOURNER a
```

FIN

\*h = ??



# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

➔ SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

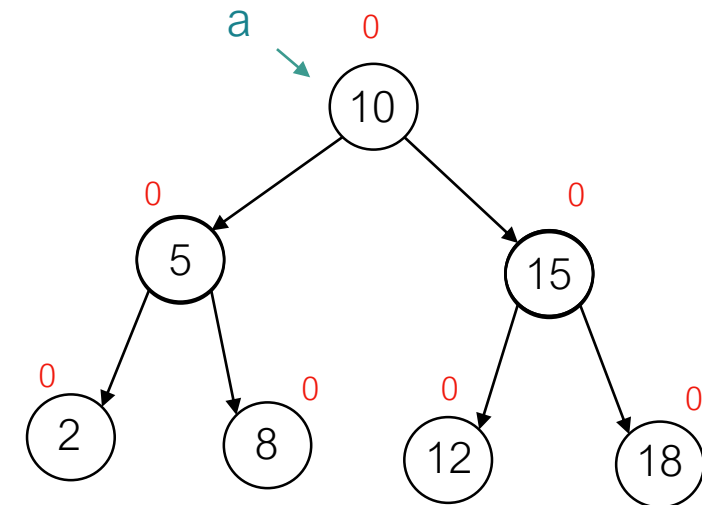
FIN SI

FIN SI

RETOURNER a

FIN

\*h = ??



# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

➔ SINON Si (e INF. STRICT. A element(a)) ALORS

fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

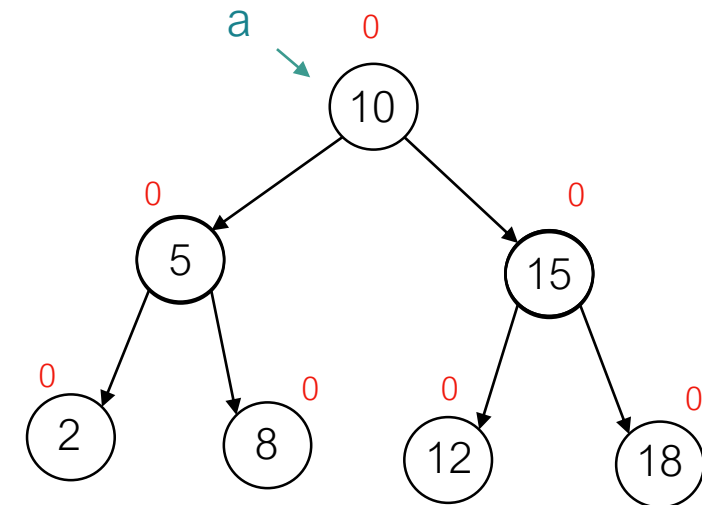
FIN SI

FIN SI

RETOURNER a

FIN

\*h = ??



# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

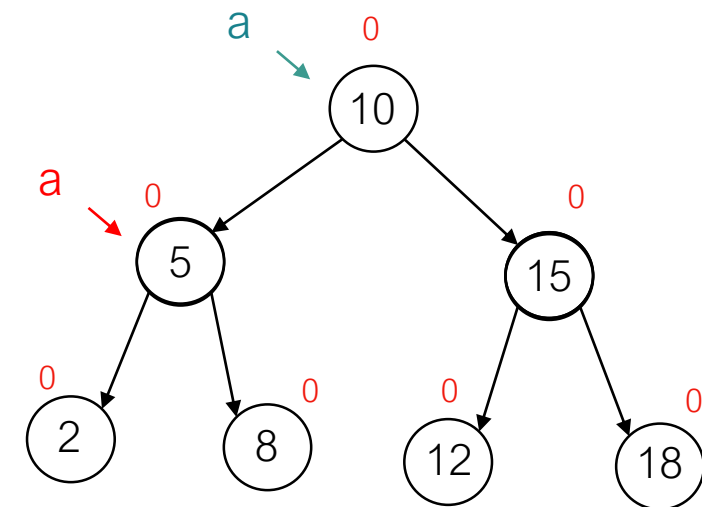
FIN SI

FIN SI

RETOURNER a

FIN

\*h = ??





# Insertion

- Exemple : insertionAVL(a, 9)

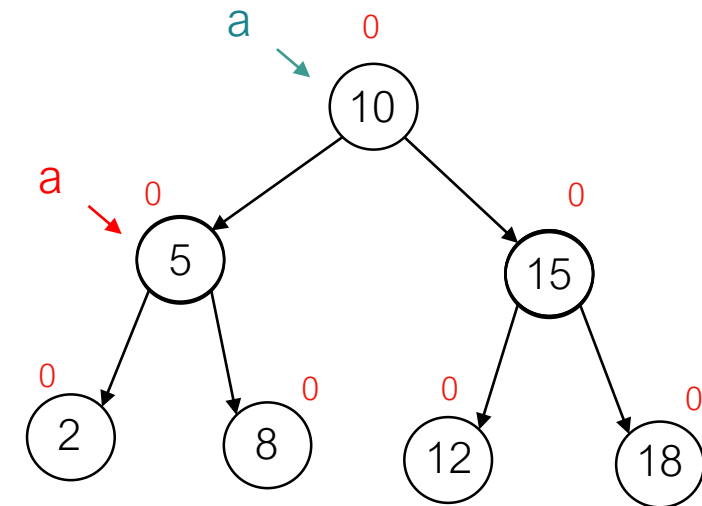
FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

→ DEBUT

```
SI (a EST EGAL A NULL) ALORS
  *h=1
  RETOURNER creerArbre(e)
SINON Si (e INF. STRICT. A element(a)) ALORS
  → fg(a) ← insertionAVL(fg(a), e, h)
  *h ← -*h
SINON Si (e SUP. STRICT. A element(a)) ALORS
  fd(a) ← insertionAVL(fd(a), e, h)
SINON
  *h= 0
  RETOURNER a
FIN SI
SI (*h DIFFERENT DE 0) ALORS
  equilibre (a) ← equilibre(a) + *h
  SI (equilibre(a) EST EGAL A 0) ALORS
    *h ← 0
  SINON
    *h ← 1
  FIN SI
FIN SI
RETOURNER a
```

FIN

\*h = ??



# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

➔ SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

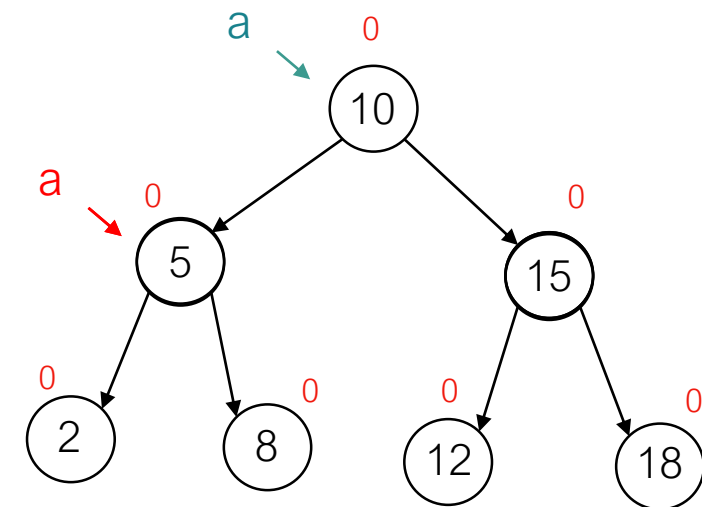
FIN SI

FIN SI

RETOURNER a

FIN

\*h = ??



# Insertion

- Exemple : insertionAVL(a, 9)

\*h = ??

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

→ SINON Si (e INF. STRICT. A element(a)) ALORS

→ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

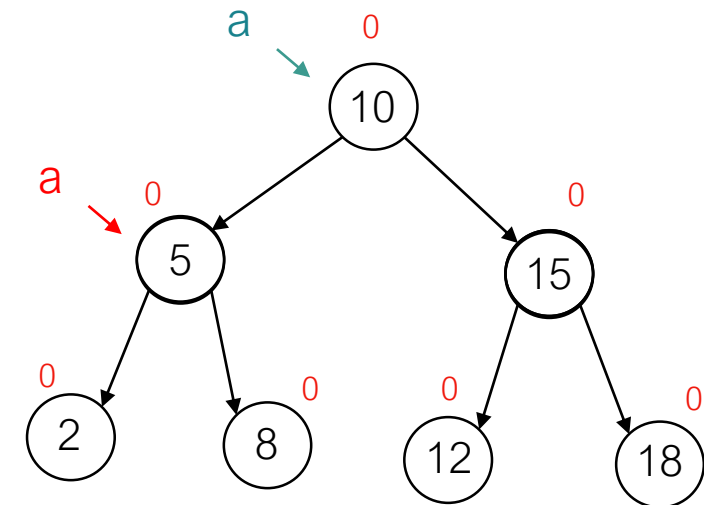
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 9)

\*h = ??

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

→ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

→ SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

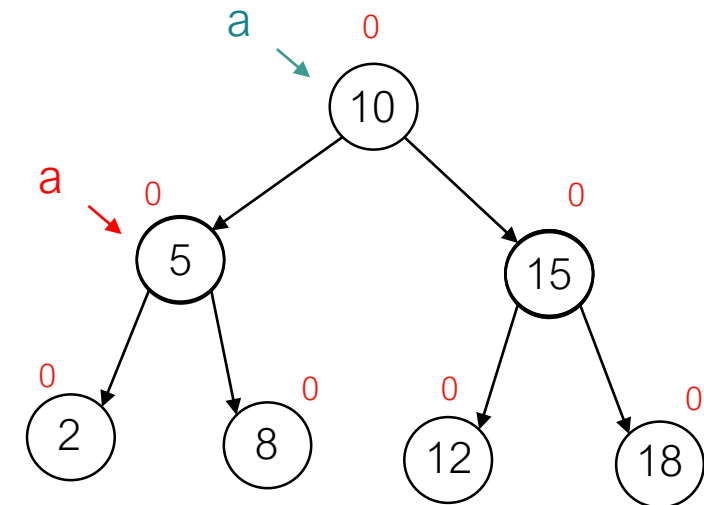
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 9)

\*h = ??

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

→ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

→ fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

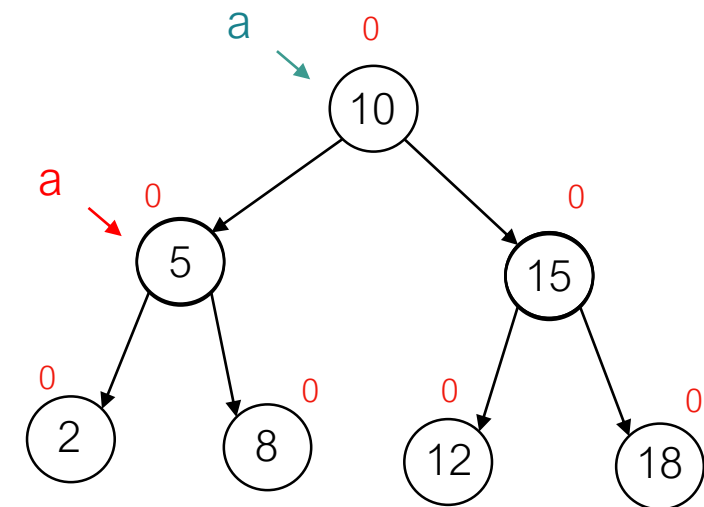
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 9)

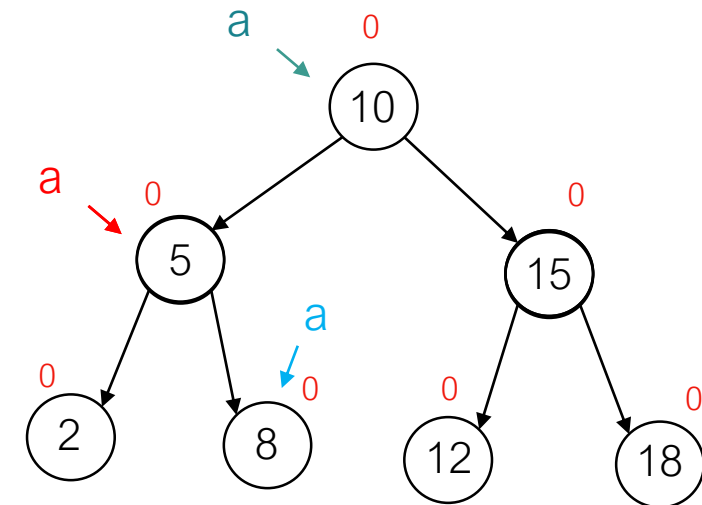
FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

→ DEBUT

```
SI (a EST EGAL A NULL) ALORS
  *h=1
  RETOURNER creerArbre(e)
SINON Si (e INF. STRICT. A element(a)) ALORS
  → fg(a) ← insertionAVL(fg(a), e, h)
  *h ← -*h
SINON Si (e SUP. STRICT. A element(a)) ALORS
  → fd(a) ← insertionAVL(fd(a), e, h)
SINON
  *h= 0
  RETOURNER a
FIN SI
SI (*h DIFFERENT DE 0) ALORS
  equilibre (a) ← equilibre(a) + *h
  SI (equilibre(a) EST EGAL A 0) ALORS
    *h ← 0
  SINON
    *h ← 1
  FIN SI
FIN SI
RETOURNER a
```

FIN

\*h = ??



# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

➡ SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➡ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

➡ fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

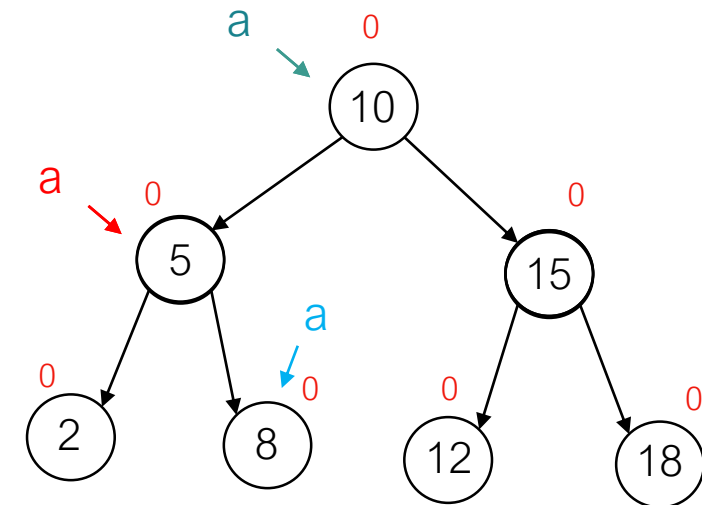
FIN SI

FIN SI

RETOURNER a

FIN

\*h = ??



# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

→ SINON Si (e INF. STRICT. A element(a)) ALORS

→ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

→ fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

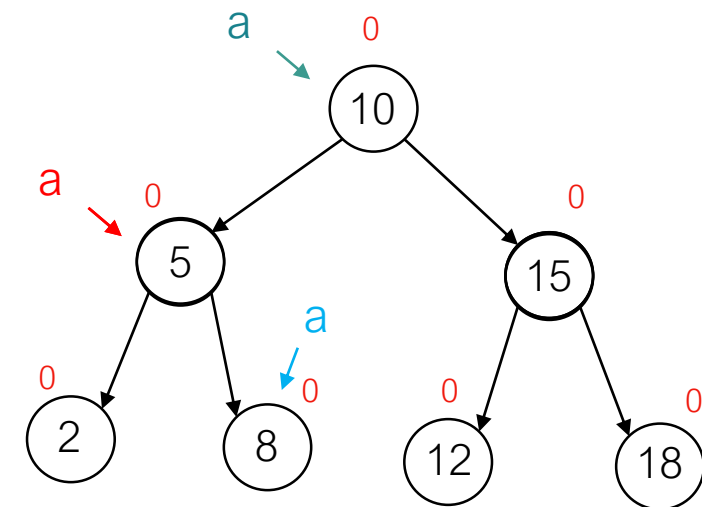
FIN SI

FIN SI

RETOURNER a

FIN

\*h = ??





# Insertion

- Exemple : insertionAVL(a, 9)

\*h = ??

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

→ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

→ SINON Si (e SUP. STRICT. A element(a)) ALORS

→ fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

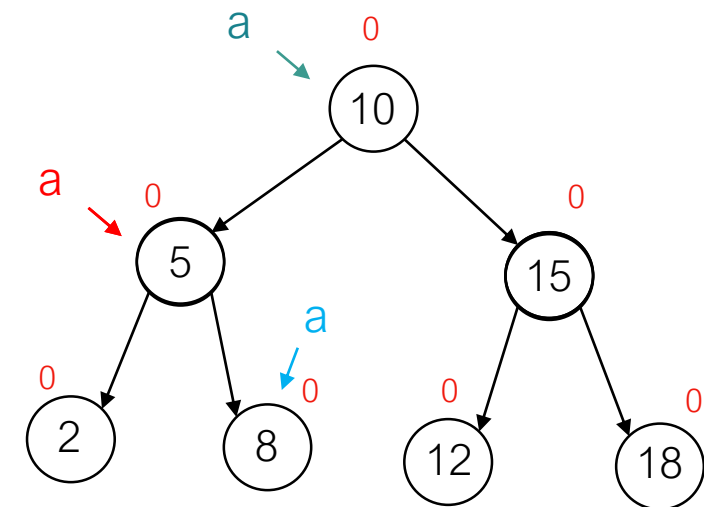
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 9)

\*h = ??

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

➔ fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

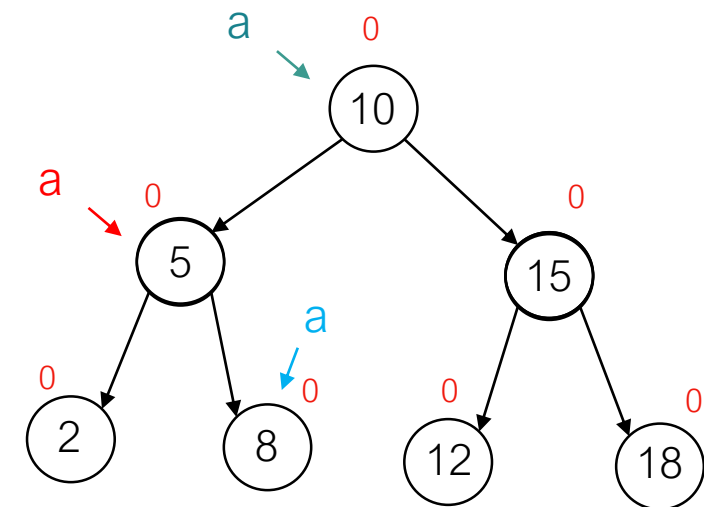
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 9)

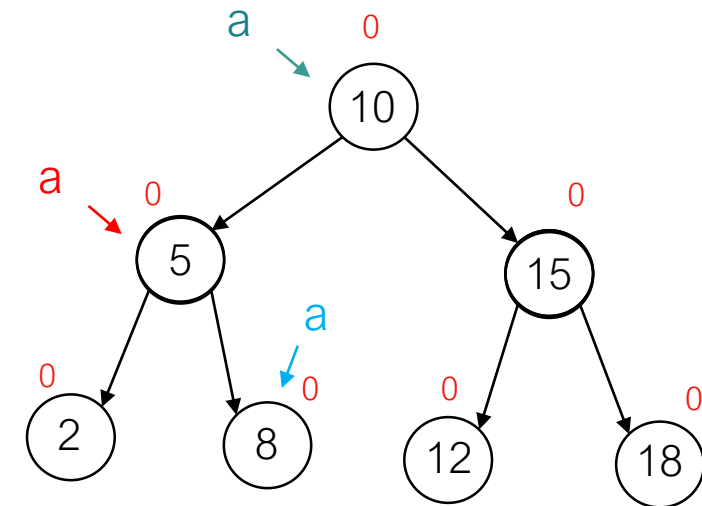
FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

➡ DEBUT

```
SI (a EST EGAL A NULL) ALORS
    *h=1
    RETOURNER creerArbre(e)
SINON Si (e INF. STRICT. A element(a)) ALORS
    ➡ fg(a) ← insertionAVL(fg(a), e, h)
    *h ← -*h
SINON Si (e SUP. STRICT. A element(a)) ALORS
    ➡ fd(a) ← insertionAVL(fd(a), e, h)
SINON
    *h= 0
    RETOURNER a
FIN SI
SI (*h DIFFERENT DE 0) ALORS
    equilibre (a) ← equilibre(a) + *h
    SI (equilibre(a) EST EGAL A 0) ALORS
        *h ← 0
    SINON
        *h ← 1
    FIN SI
FIN SI
RETOURNER a
```

FIN

\*h = ??



a → NULL

# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

➔ SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

➔ fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

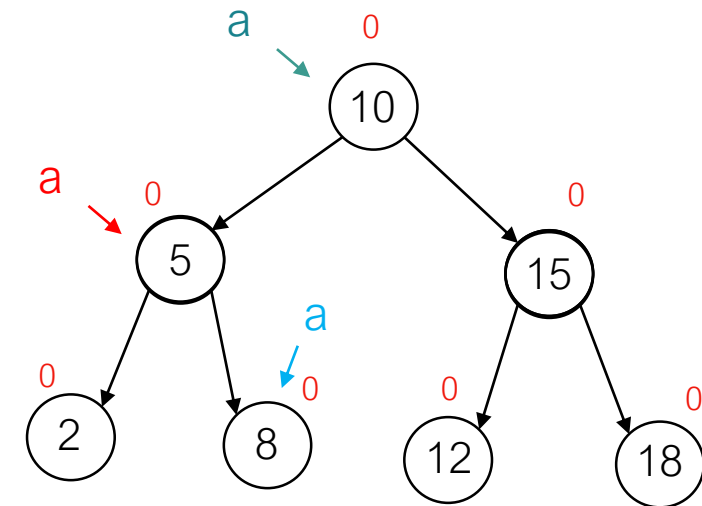
FIN SI

FIN SI

RETOURNER a

FIN

\*h = ??



a → NULL

# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

    ➡ \*h=1

    RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

    ➡ fg(a) ← insertionAVL(fg(a), e, h)

    \*h ← -\*h

SINON Si (e SUP. STRICT. A element(a)) ALORS

    ➡ fd(a) ← insertionAVL(fd(a), e, h)

SINON

    \*h = 0

    RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

    equilibre(a) ← equilibre(a) + \*h

    SI (equilibre(a) EST EGAL A 0) ALORS

        \*h ← 0

    SINON

        \*h ← 1

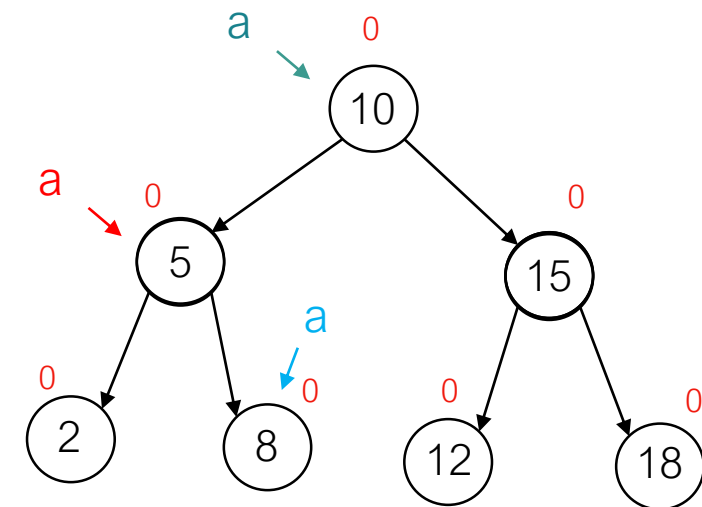
    FIN SI

FIN SI

RETOURNER a

FIN

\*h = ??



a → NULL

# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

➡ RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➡ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

➡ fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

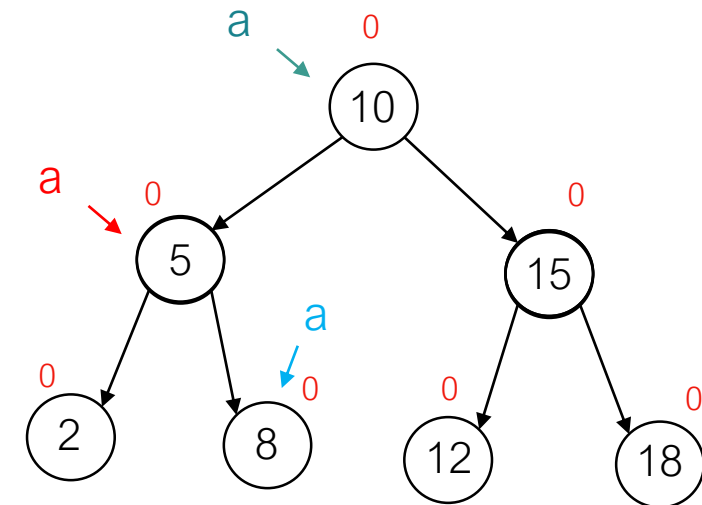
FIN SI

FIN SI

RETOURNER a

FIN

\*h = 1



a → NULL

# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

➔ fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

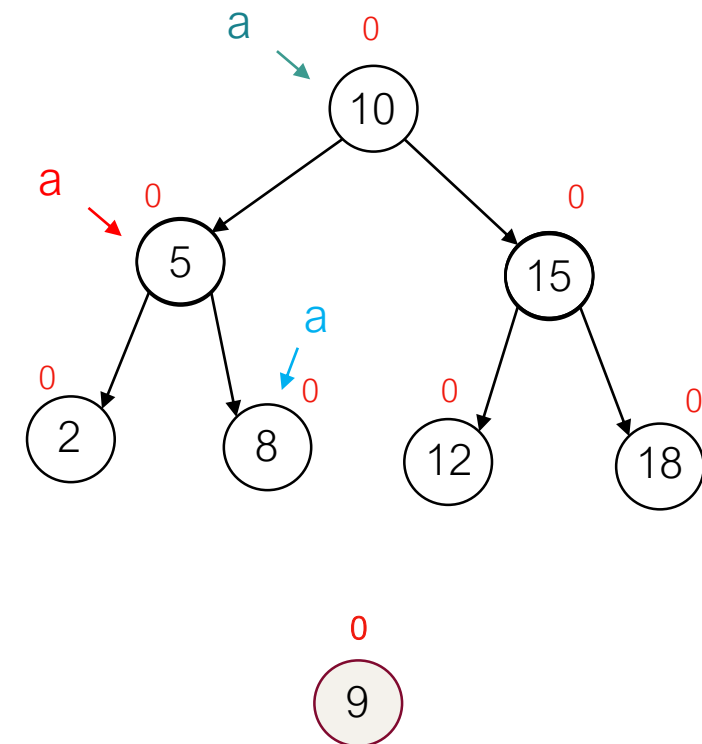
FIN SI

FIN SI

RETOURNER a

FIN

\*h = 1



# Insertion

- Exemple : insertionAVL(a, 9)

\*h = 1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

➔ fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h = 0**

RETOURNER a

FIN SI

➔ SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

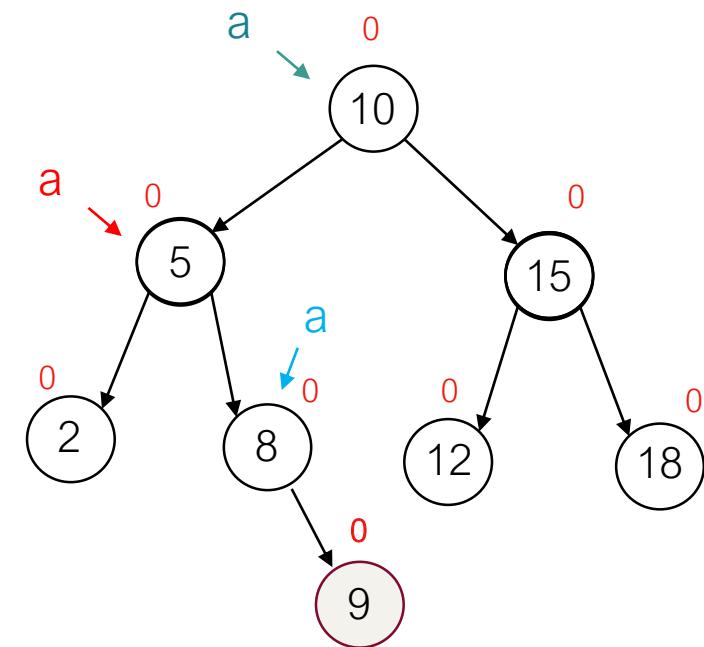
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN





# Insertion

- Exemple : insertionAVL(a, 9)

\*h = 1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

➔ fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h = 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

➔ **equilibre(a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

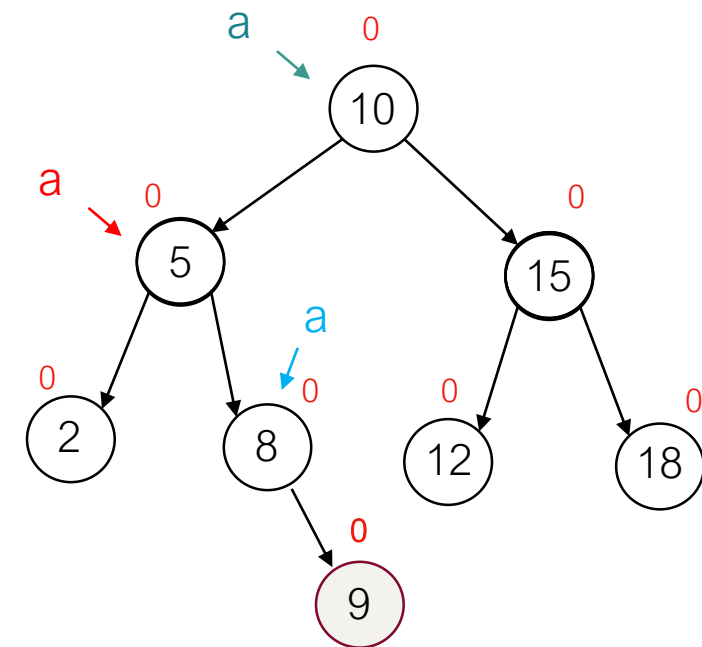
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 9)

\*h = 1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➡ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

➡ fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

➡ SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

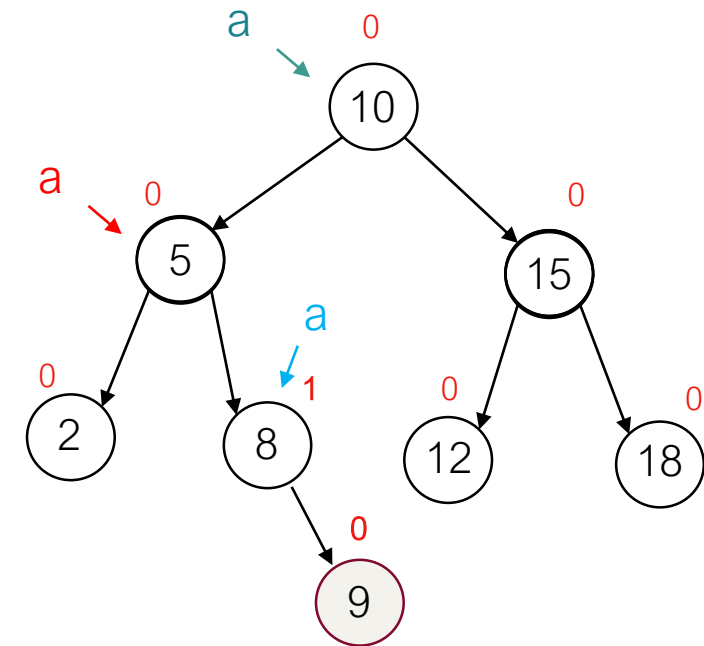
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 9)

\*h = 1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

➔ fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

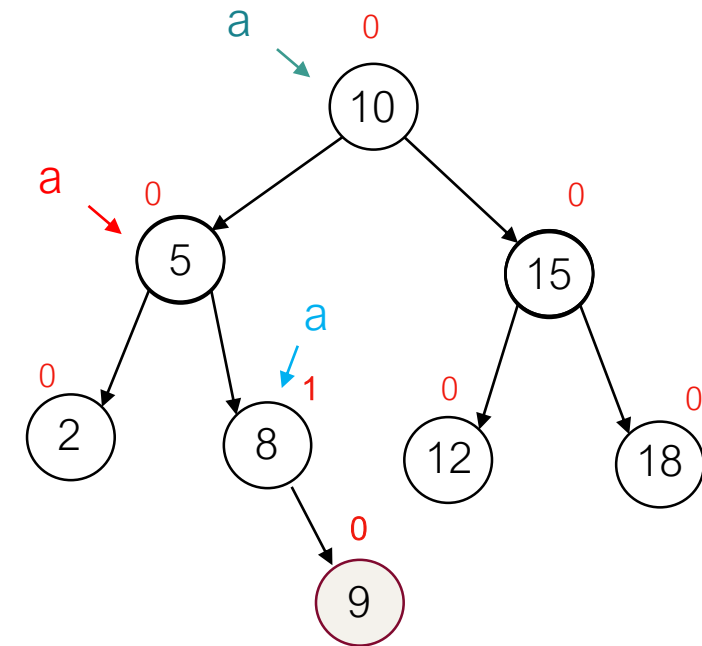
➔ **\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➡ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

➡ fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

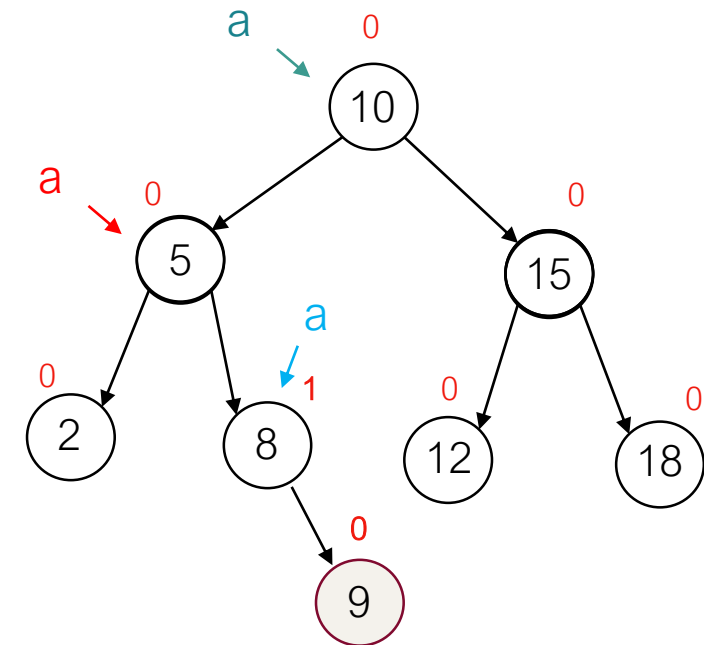
FIN SI

FIN SI

➡ RETOURNER a

FIN

\*h = 1



# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➡ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

➡ fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

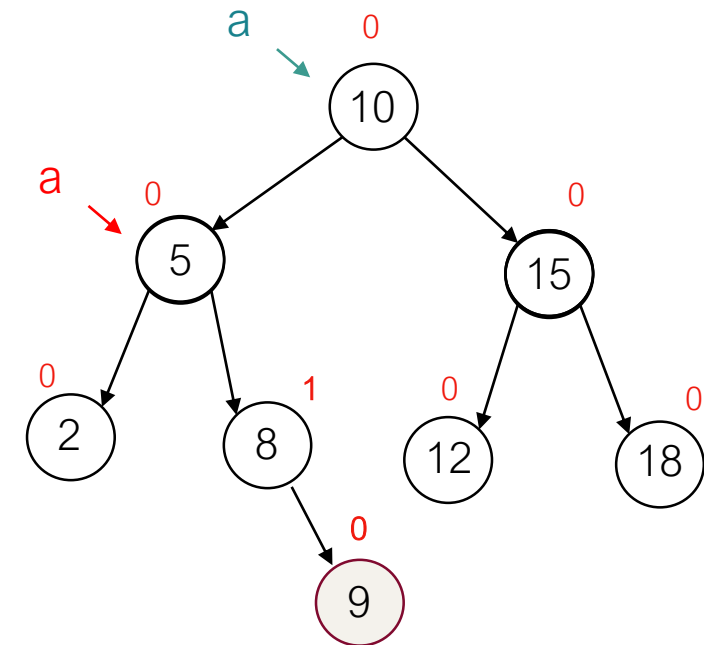
FIN SI

FIN SI

RETOURNER a

FIN

\*h = 1



# Insertion

- Exemple : insertionAVL(a, 9)

\*h = 1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

→ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

→ SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

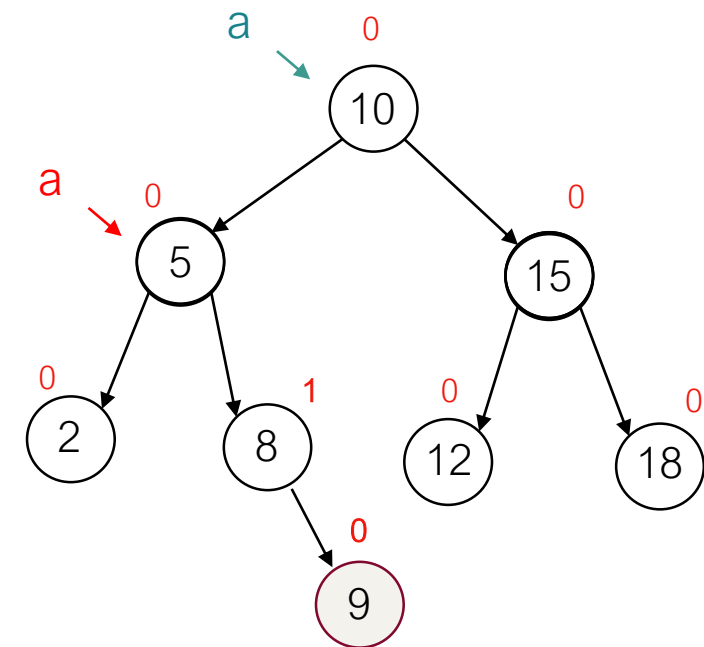
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 9)

\*h = 1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

➔ **equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

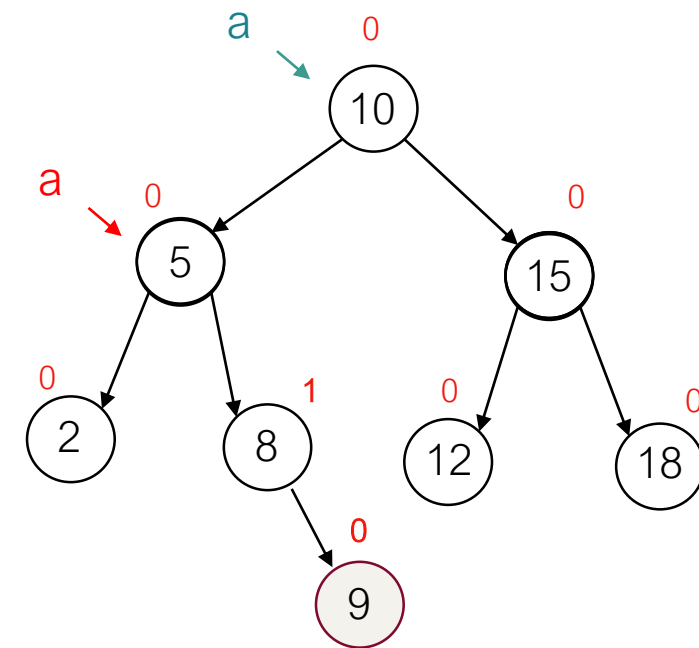
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 9)

\*h = 1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

    RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

    ➡ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

    fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

    RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

    ➡ SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

    SINON

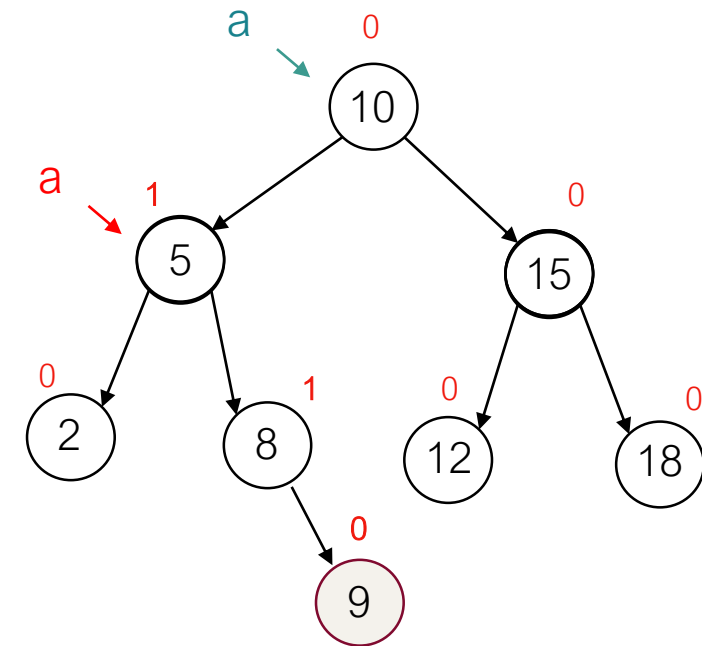
**\*h ← 1**

    FIN SI

FIN SI

RETOURNER a

FIN





# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

→ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

→ **\*h ← 1**

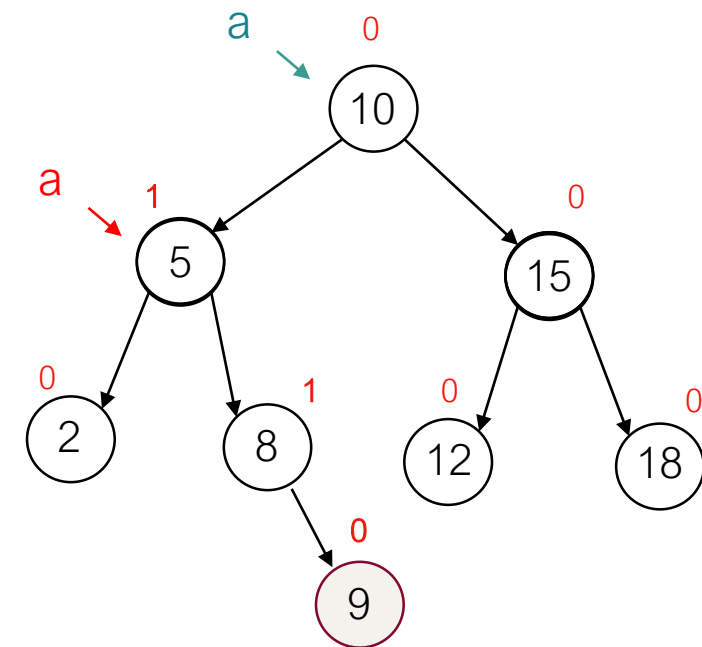
FIN SI

FIN SI

RETOURNER a

FIN

\*h = 1



# Insertion

- Exemple : insertionAVL(a, 9)

\*h = 1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

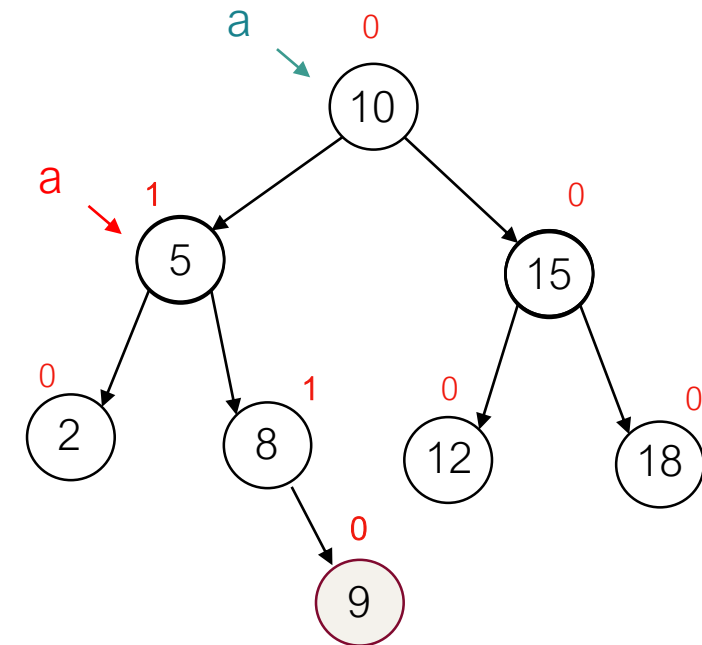
**\*h ← 1**

FIN SI

FIN SI

➔ RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

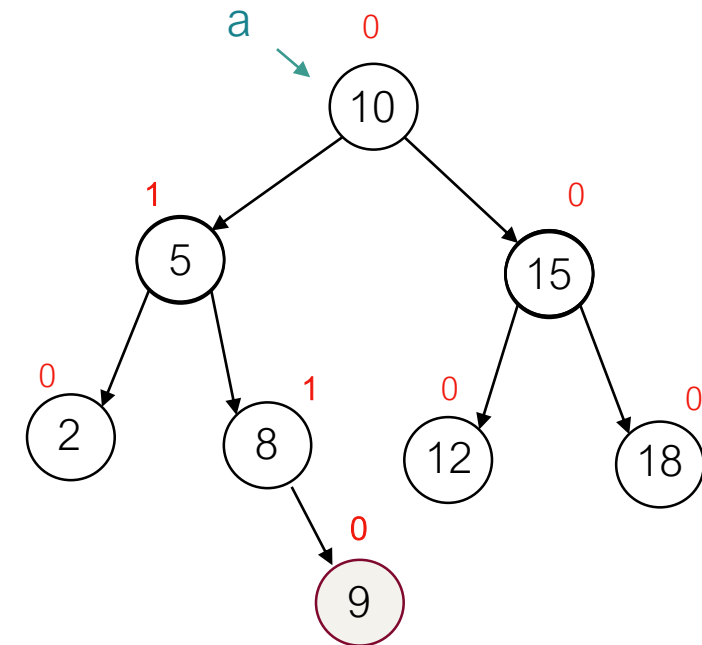
FIN SI

FIN SI

RETOURNER a

FIN

\*h = 1



# Insertion

- Exemple : insertionAVL(a, 9)

\*h = 1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

fg(a) ← insertionAVL(fg(a), e, h)

➔ **\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

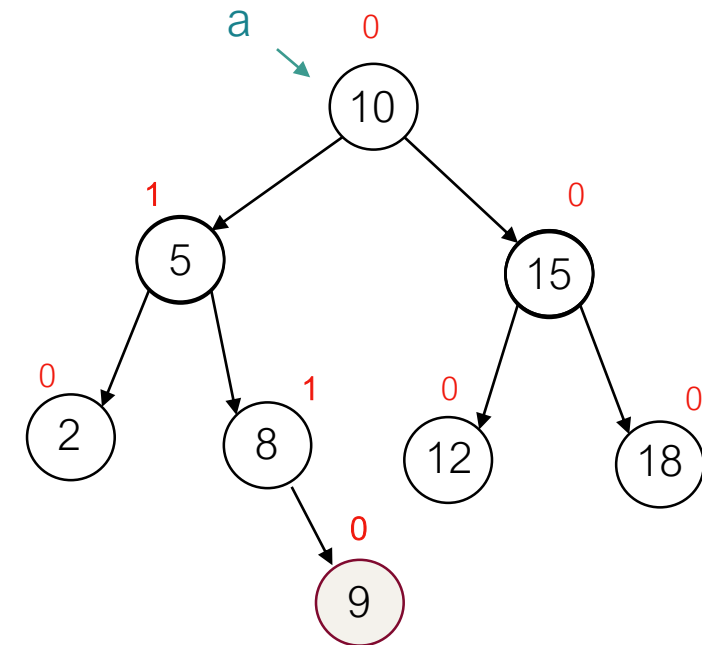
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

➔ SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

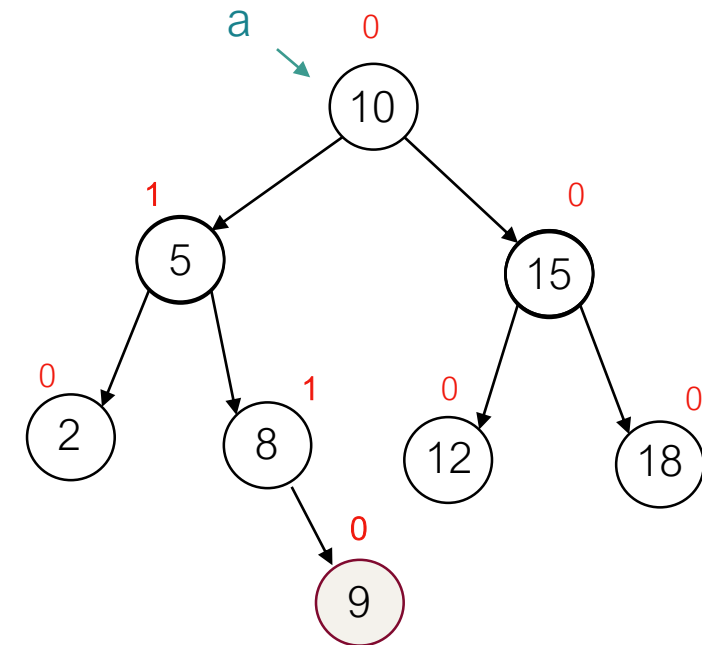
FIN SI

FIN SI

RETOURNER a

FIN

\*h = -1



# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

➔ **equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

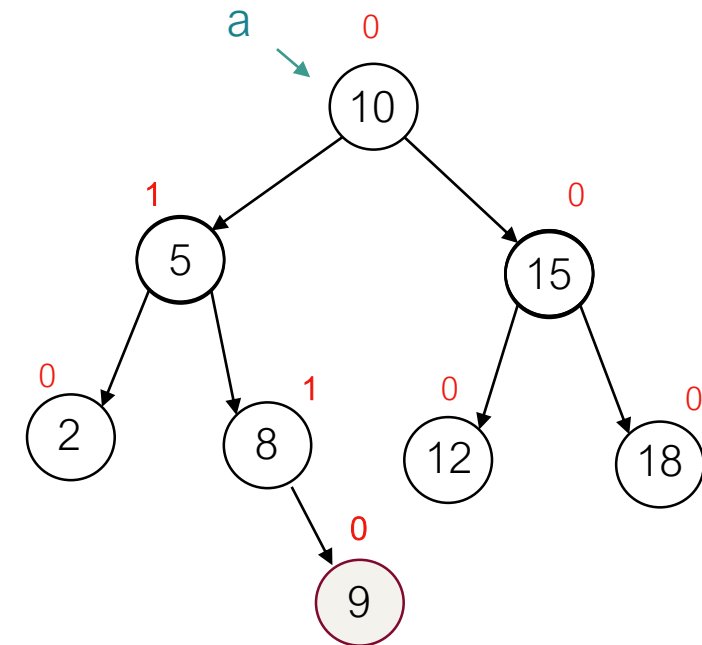
FIN SI

FIN SI

RETOURNER a

FIN

\*h = -1



# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

➔ SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

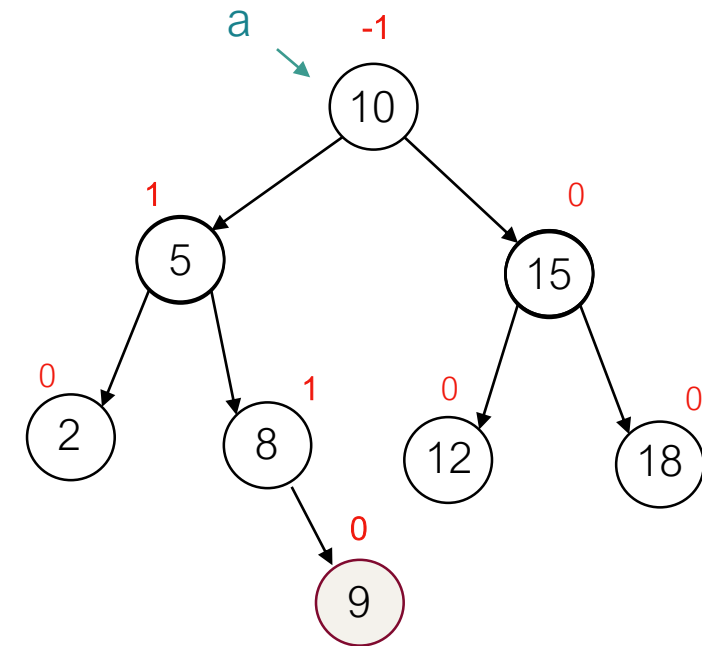
FIN SI

FIN SI

RETOURNER a

FIN

\*h = -1



# Insertion

- Exemple : insertionAVL(a, 9)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

➡ **\*h ← 1**

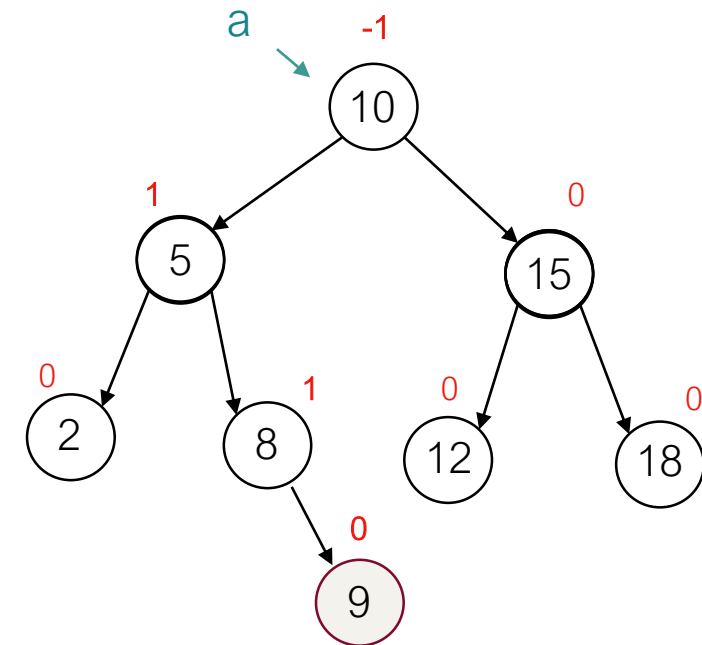
FIN SI

FIN SI

RETOURNER a

FIN

\*h = -1





# Insertion

- Exemple : insertionAVL(a, 9)

\*h = 1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

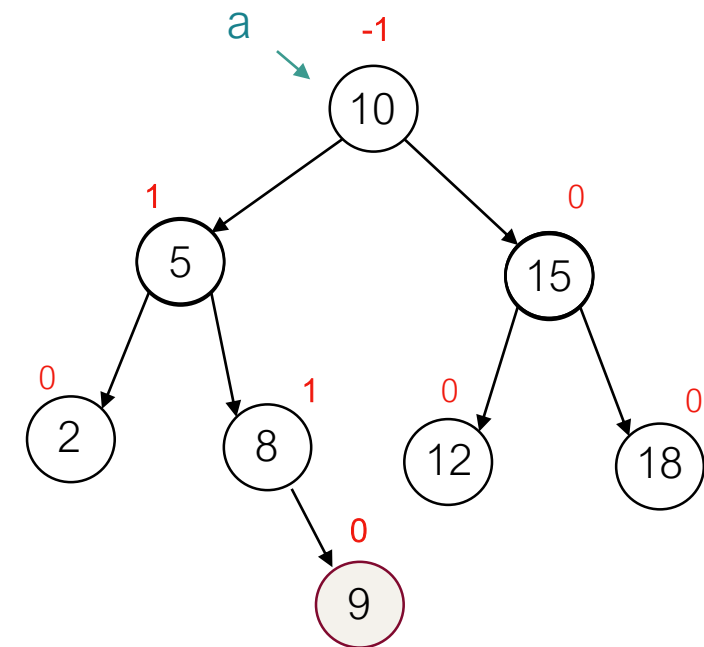
**\*h ← 1**

FIN SI

FIN SI

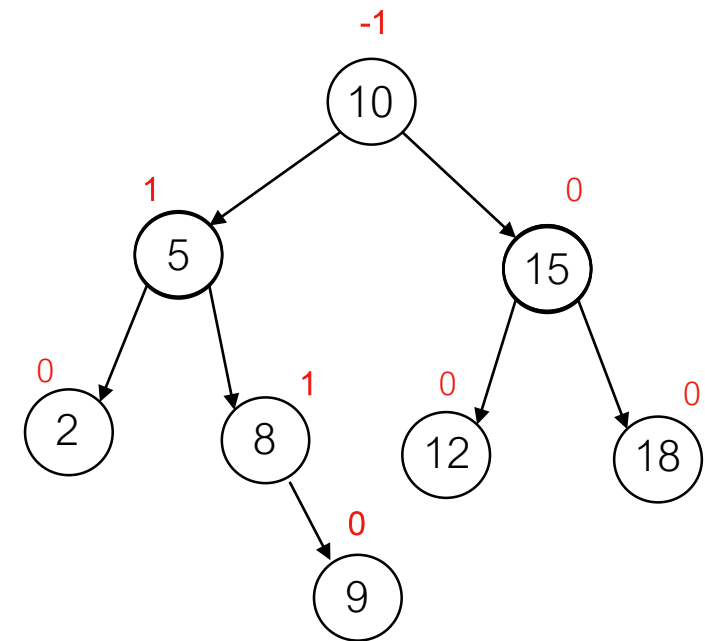
➔ RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)



# Insertion

- Exemple : insertionAVL(a, 1)

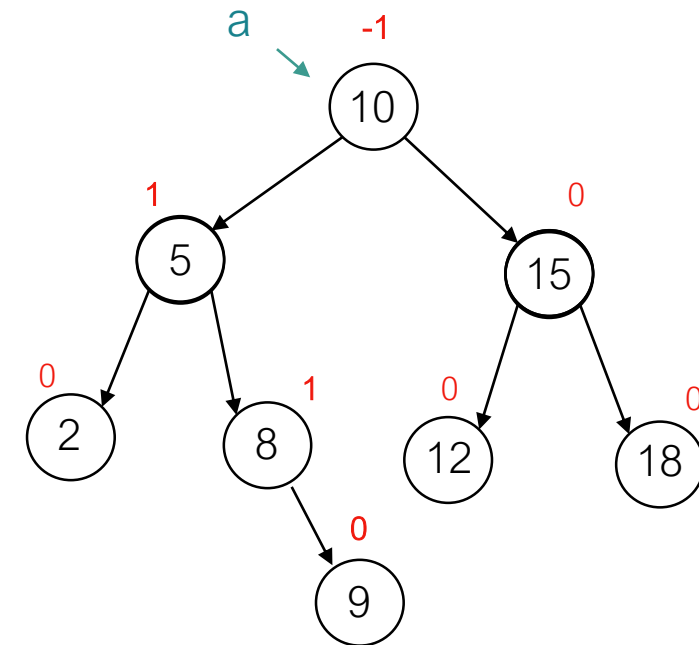
FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

→ DEBUT

```
SI (a EST EGAL A NULL) ALORS
  *h=1
  RETOURNER creerArbre(e)
SINON Si (e INF. STRICT. A element(a)) ALORS
  fg(a) ← insertionAVL(fg(a), e, h)
  *h ← -*h
SINON Si (e SUP. STRICT. A element(a)) ALORS
  fd(a) ← insertionAVL(fd(a), e, h)
SINON
  *h= 0
  RETOURNER a
FIN SI
SI (*h DIFFERENT DE 0) ALORS
  equilibre (a) ← equilibre(a) + *h
  SI (equilibre(a) EST EGAL A 0) ALORS
    *h ← 0
  SINON
    *h ← 1
  FIN SI
FIN SI
RETOURNER a
```

FIN

\*h = ??



# Insertion

- Exemple : insertionAVL(a, 1)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

➔ SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

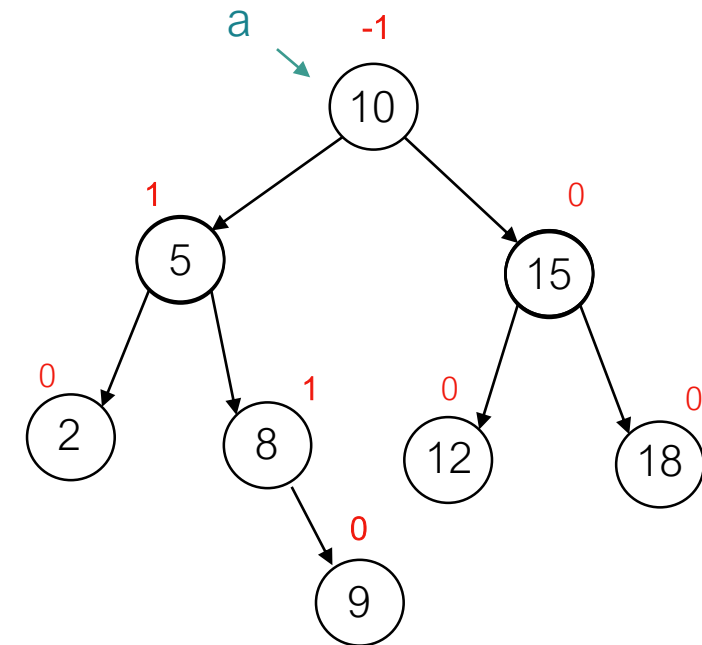
FIN SI

FIN SI

RETOURNER a

FIN

\*h = ??



# Insertion

- Exemple : insertionAVL(a, 1)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

➔ SINON Si (e INF. STRICT. A element(a)) ALORS

fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

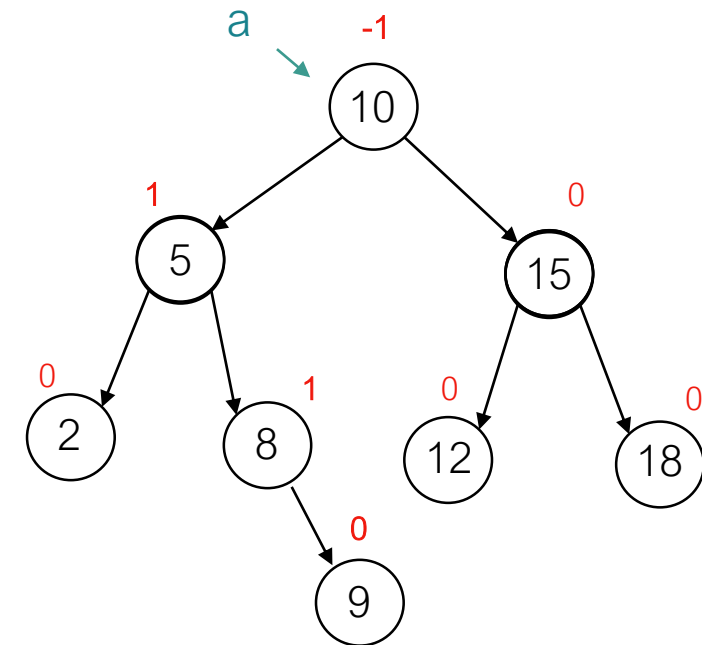
FIN SI

FIN SI

RETOURNER a

FIN

\*h = ??



# Insertion

- Exemple : insertionAVL(a, 1)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

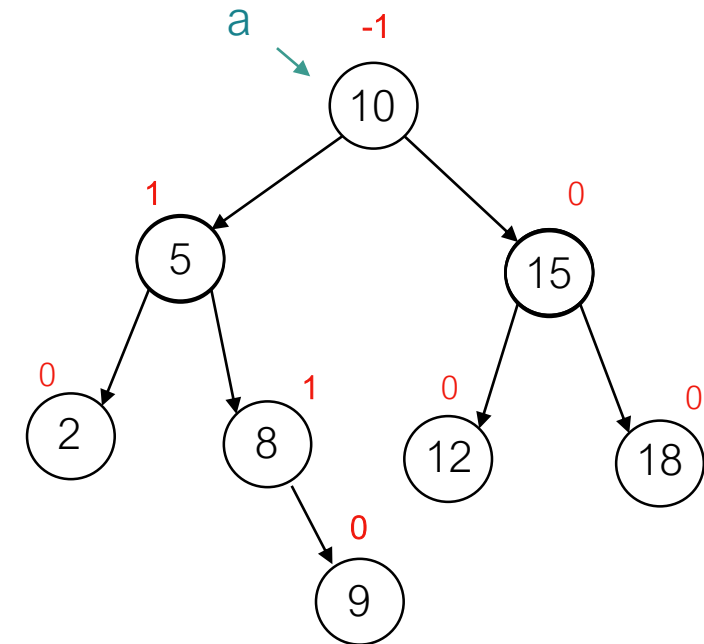
FIN SI

FIN SI

RETOURNER a

FIN

\*h = ??



# Insertion

- Exemple : insertionAVL(a, 1)

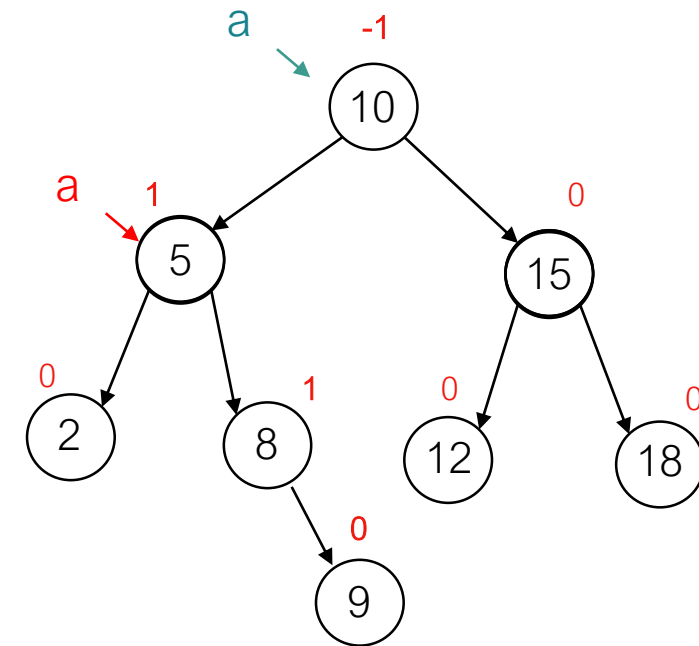
FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

→ DEBUT

```
SI (a EST EGAL A NULL) ALORS
  *h=1
  RETOURNER creerArbre(e)
SINON Si (e INF. STRICT. A element(a)) ALORS
  → fg(a) ← insertionAVL(fg(a), e, h)
  *h ← -*h
SINON Si (e SUP. STRICT. A element(a)) ALORS
  fd(a) ← insertionAVL(fd(a), e, h)
SINON
  *h= 0
  RETOURNER a
FIN SI
SI (*h DIFFERENT DE 0) ALORS
  equilibre (a) ← equilibre(a) + *h
  SI (equilibre(a) EST EGAL A 0) ALORS
    *h ← 0
  SINON
    *h ← 1
  FIN SI
FIN SI
RETOURNER a
```

FIN

\*h = ??



# Insertion

- Exemple : insertionAVL(a, 1)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

➔ SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

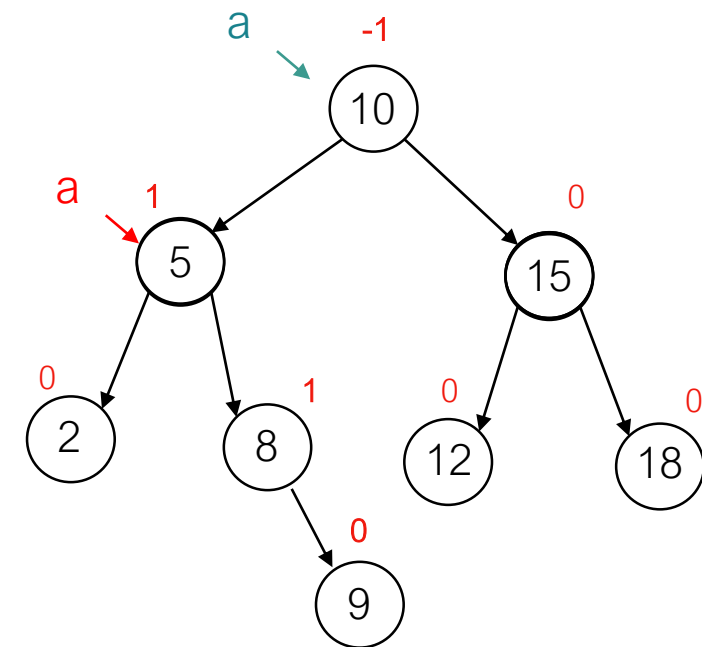
FIN SI

FIN SI

RETOURNER a

FIN

\*h = ??





# Insertion

- Exemple : insertionAVL(a, 1)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

➔ SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

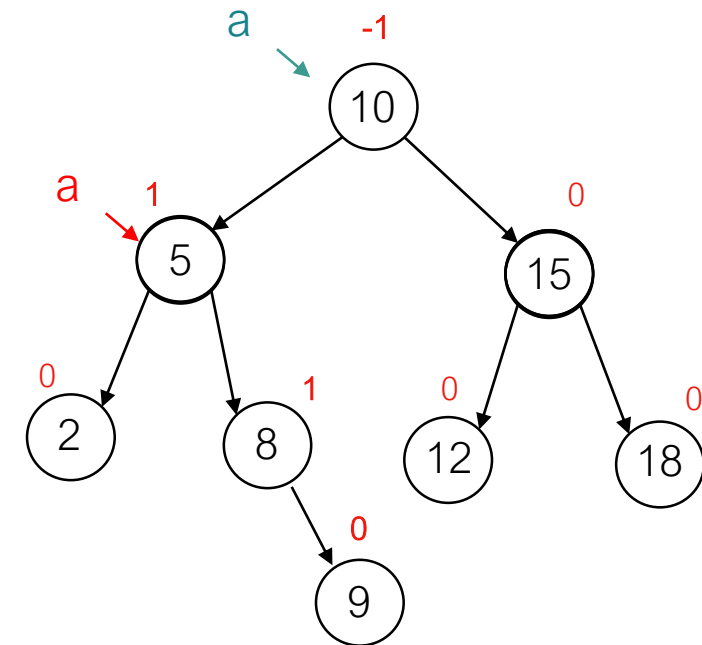
FIN SI

FIN SI

RETOURNER a

FIN

\*h = ??



# Insertion

- Exemple : insertionAVL(a, 1)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

**fg(a) ← insertionAVL(fg(a), e, h)**

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

**fd(a) ← insertionAVL(fd(a), e, h)**

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

**\*h ← 1**

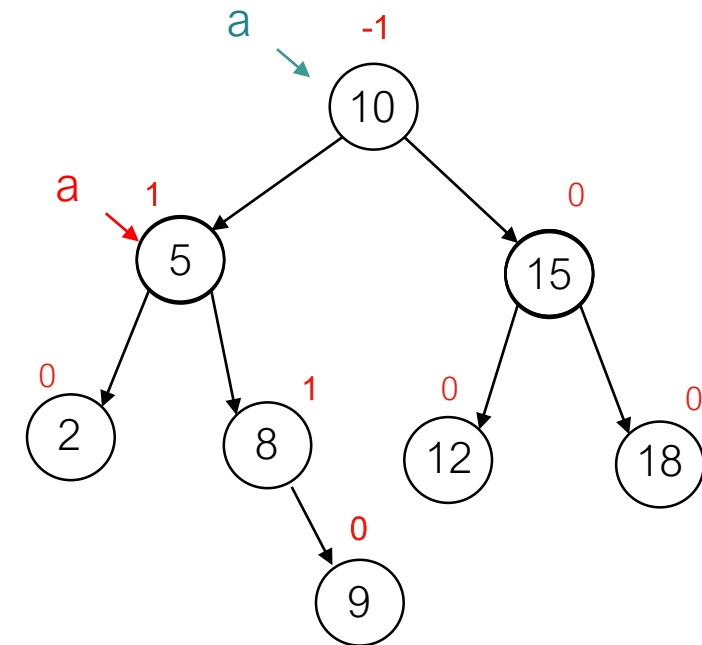
FIN SI

FIN SI

RETOURNER a

FIN

\*h = ??



# Insertion

- Exemple : insertionAVL(a, 1)

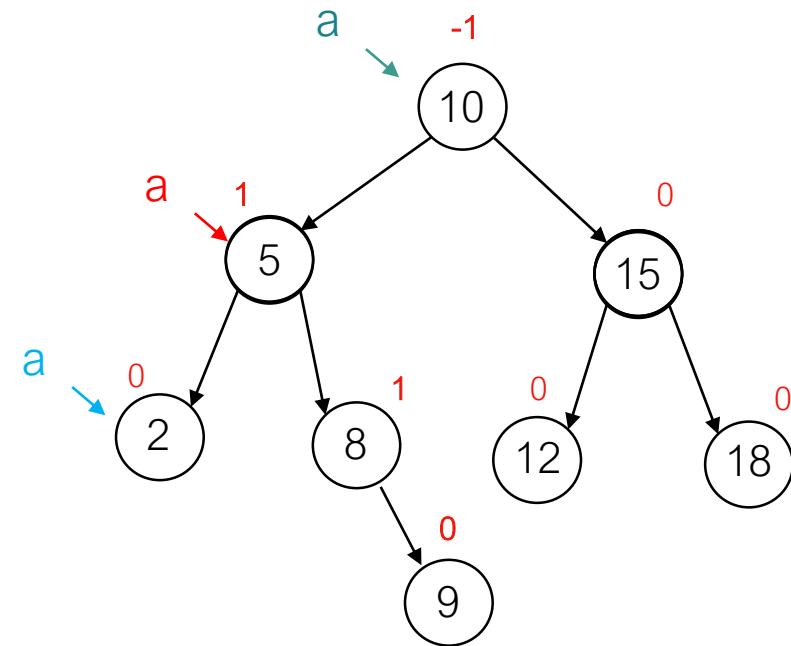
FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

→ DEBUT

```
SI (a EST EGAL A NULL) ALORS
  *h=1
  RETOURNER creerArbre(e)
SINON Si (e INF. STRICT. A element(a)) ALORS
  → fg(a) ← insertionAVL(fg(a), e, h)
  *h ← -*h
SINON Si (e SUP. STRICT. A element(a)) ALORS
  fd(a) ← insertionAVL(fd(a), e, h)
SINON
  *h= 0
  RETOURNER a
FIN SI
SI (*h DIFFERENT DE 0) ALORS
  equilibre (a) ← equilibre(a) + *h
  SI (equilibre(a) EST EGAL A 0) ALORS
    *h ← 0
  SINON
    *h ← 1
  FIN SI
FIN SI
RETOURNER a
```

FIN

\*h = ??



# Insertion

- Exemple : insertionAVL(a, 1)

\*h = ??

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

➔ SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

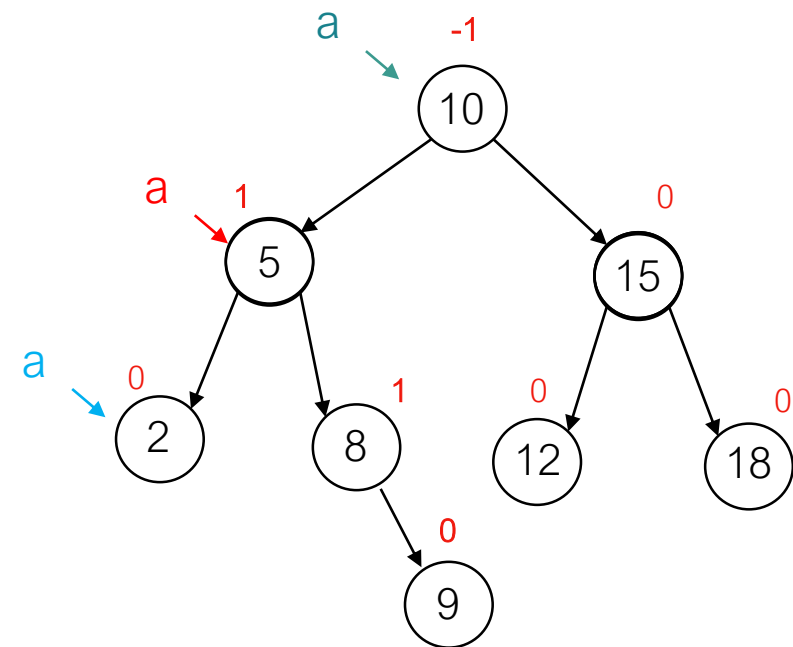
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

\*h = ??

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

➡ SINON Si (e INF. STRICT. A element(a)) ALORS

➡➡ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

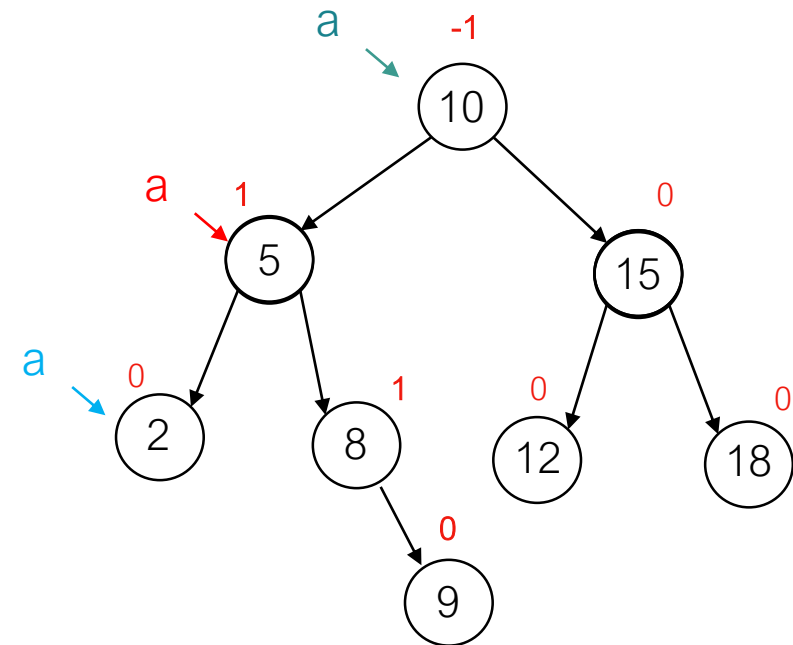
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

\*h = ??

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

→ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

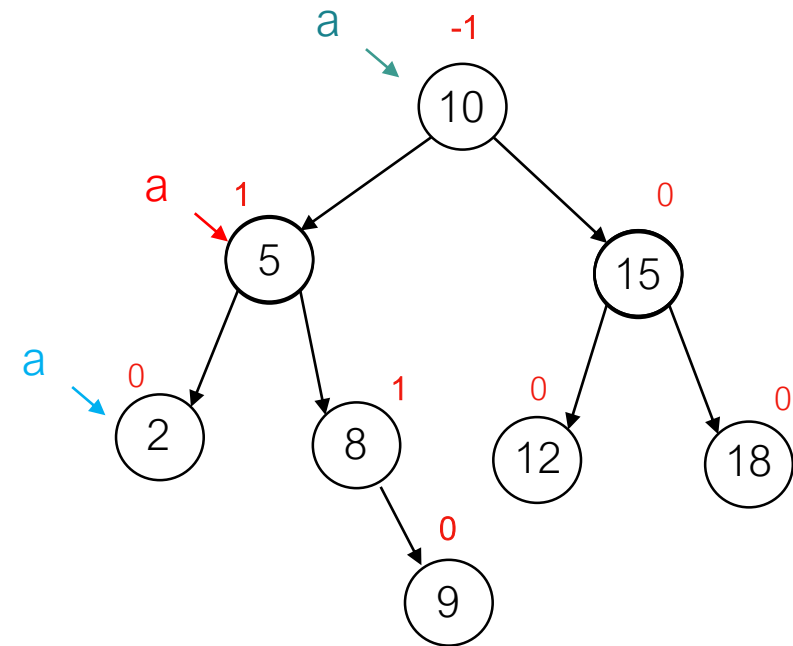
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

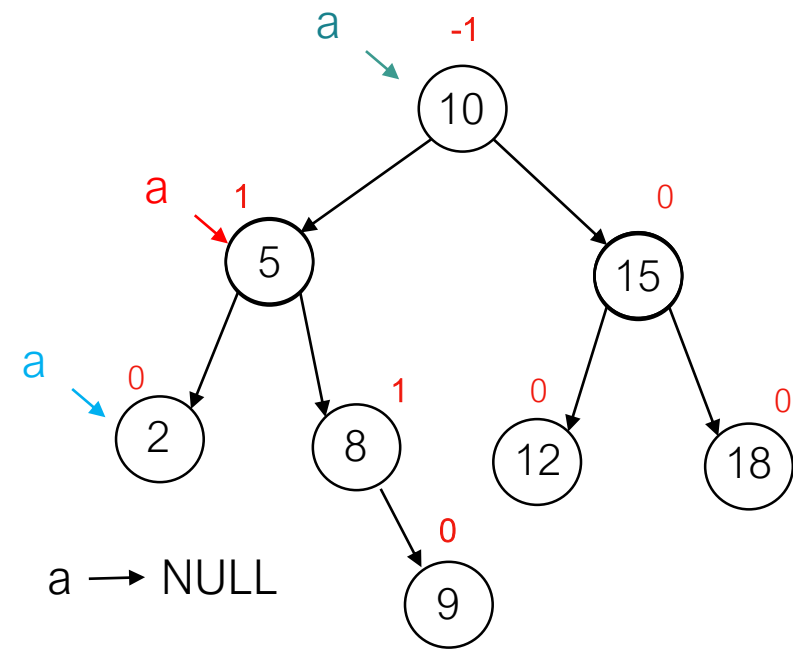
- Exemple : insertionAVL(a, 1)

\*h = ??

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

```

➡ DEBUT
  SI (a EST EGAL A NULL) ALORS
    *h=1
    RETOURNER creerArbre(e)
  SINON Si (e INF. STRICT. A element(a)) ALORS
    ➡ fg(a) ← insertionAVL(fg(a), e, h)
    *h ← -*h
  SINON Si (e SUP. STRICT. A element(a)) ALORS
    fd(a) ← insertionAVL(fd(a), e, h)
  SINON
    *h= 0
    RETOURNER a
  FIN SI
  SI (*h DIFFERENT DE 0) ALORS
    equilibre (a) ← equilibre(a) + *h
    SI (equilibre(a) EST EGAL A 0) ALORS
      *h ← 0
    SINON
      *h ← 1
    FIN SI
  FIN SI
  RETOURNER a
FIN
  
```



# Insertion

- Exemple : insertionAVL(a, 1)

\*h = ??

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

➔ SI (a EST EGAL A NULL) ALORS

    \*h=1

    RETOURNER creerArbre(e)

    SINON Si (e INF. STRICT. A element(a)) ALORS

➔➔➔ fg(a) ← insertionAVL(fg(a), e, h)

    \*h ← -\*h

    SINON Si (e SUP. STRICT. A element(a)) ALORS

    fd(a) ← insertionAVL(fd(a), e, h)

    SINON

        \*h= 0

        RETOURNER a

    FIN SI

    SI (\*h DIFFERENT DE 0) ALORS

        equilibre (a) ← equilibre(a) + \*h

        SI (equilibre(a) EST EGAL A 0) ALORS

            \*h ← 0

        SINON

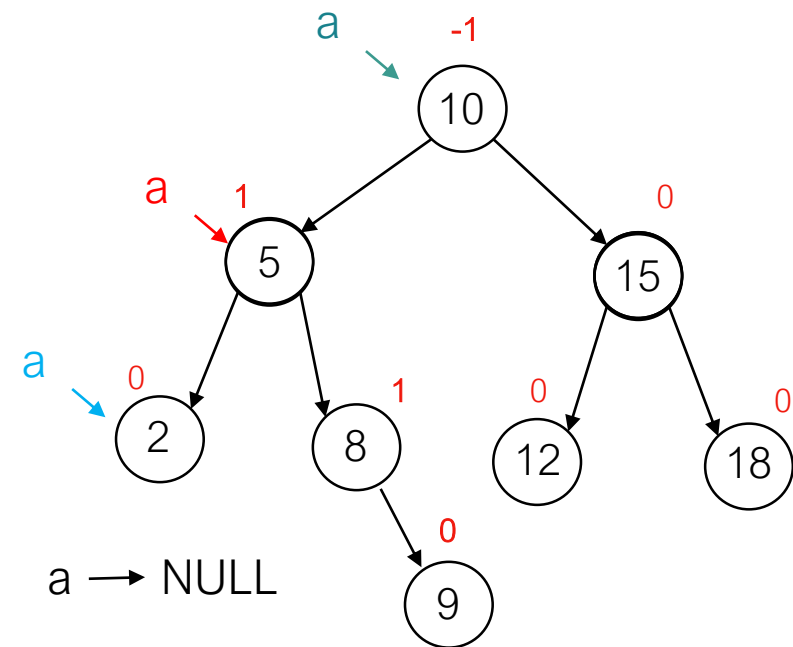
            \*h ← 1

        FIN SI

    FIN SI

    RETOURNER a

FIN





# Insertion

- Exemple : insertionAVL(a, 1)

\*h = ??

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

    ➔ \*h=1

    RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔➔➔ fg(a) ← insertionAVL(fg(a), e, h)

    \*h ← -\*h

SINON Si (e SUP. STRICT. A element(a)) ALORS

    fd(a) ← insertionAVL(fd(a), e, h)

SINON

    \*h = 0

    RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

    equilibre(a) ← equilibre(a) + \*h

    SI (equilibre(a) EST EGAL A 0) ALORS

        \*h ← 0

    SINON

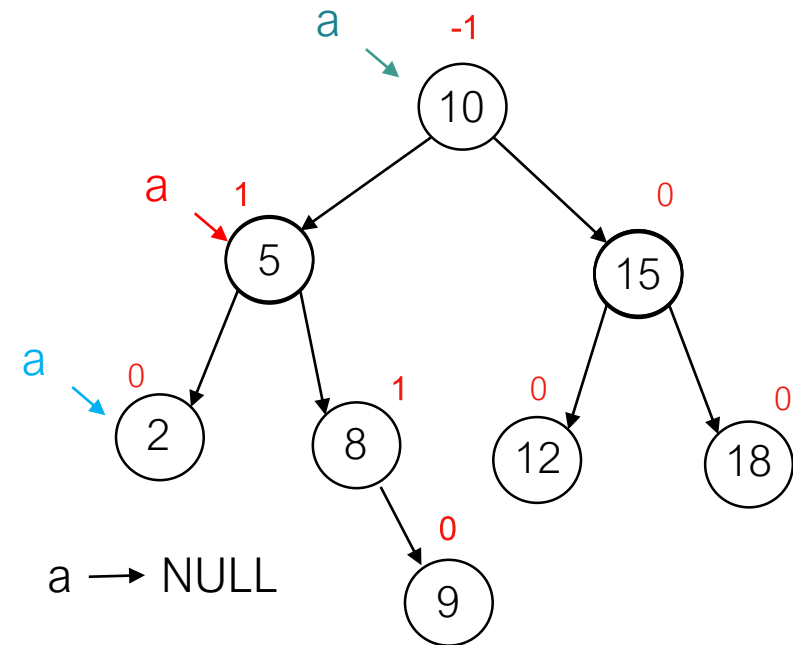
        \*h ← 1

    FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

\*h = 1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

➡ RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➡➡➡ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h = 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

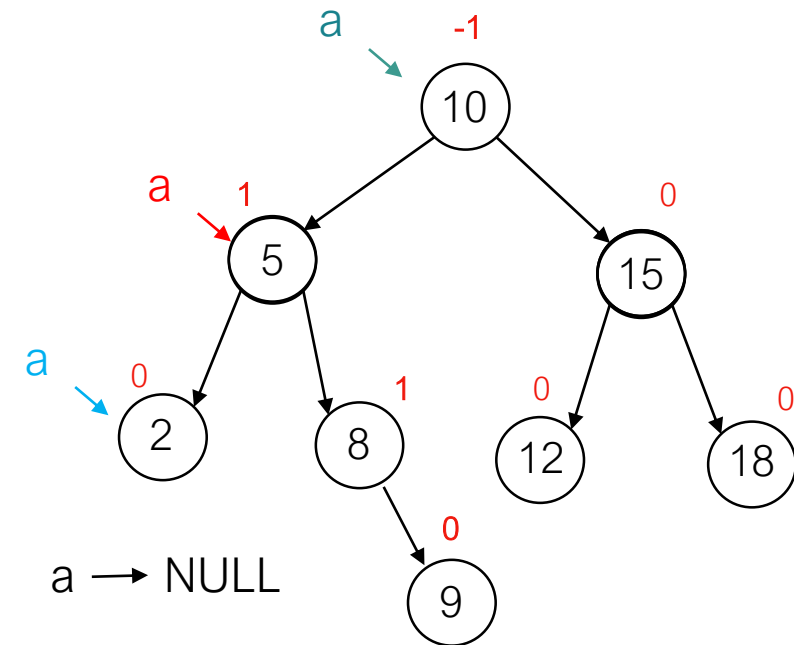
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

\*h = 1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

→ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

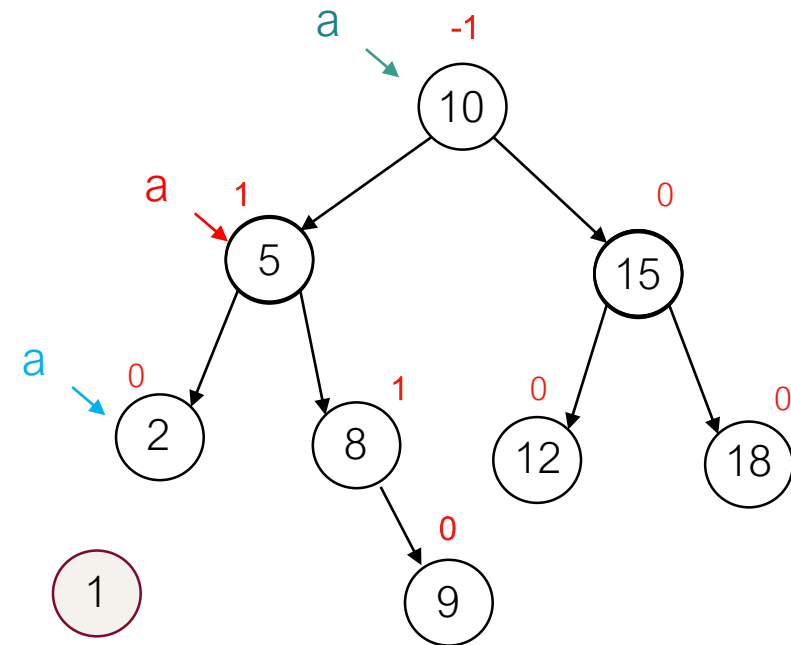
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

\*h = 1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

**→ fg(a) ← insertionAVL(fg(a), e, h)**

**→ \*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

**fd(a) ← insertionAVL(fd(a), e, h)**

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

**SI (equilibre(a) EST EGAL A 0) ALORS**

**\*h ← 0**

SINON

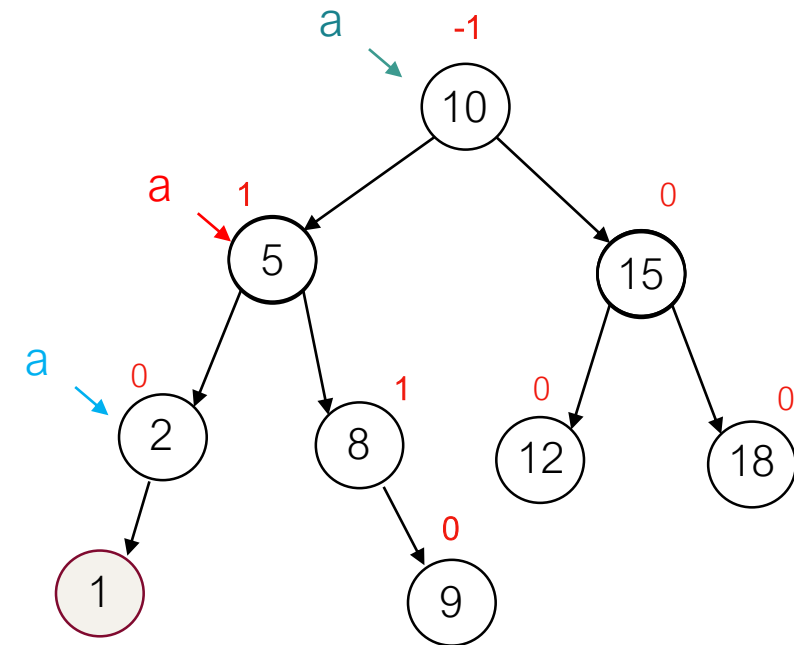
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

\*h = -1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

**→ fg(a) ← insertionAVL(fg(a), e, h)**

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

**→ SI (\*h DIFFERENT DE 0) ALORS**

**equilibre (a) ← equilibre(a) + \*h**

**SI (equilibre(a) EST EGAL A 0) ALORS**

**\*h ← 0**

SINON

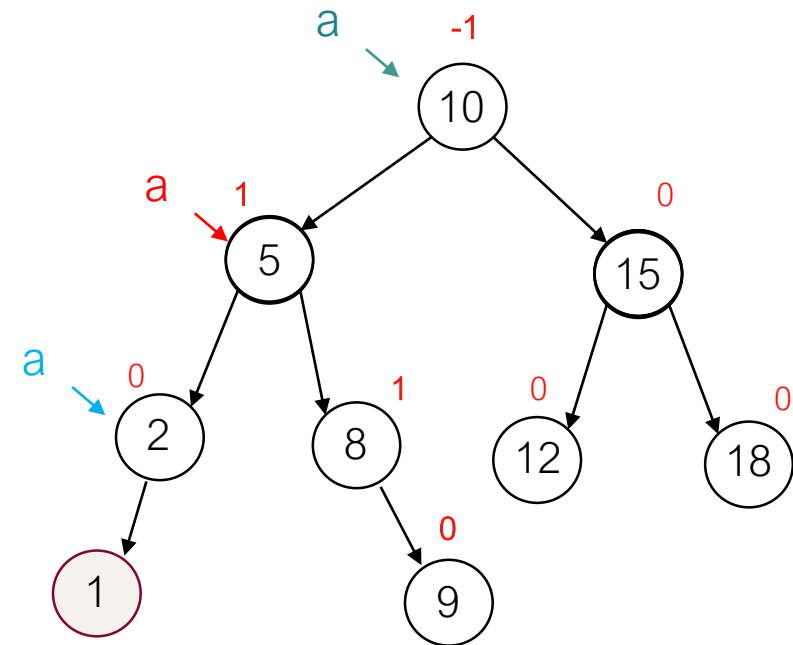
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

\*h = -1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

**→ fg(a) ← insertionAVL(fg(a), e, h)**

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

**fd(a) ← insertionAVL(fd(a), e, h)**

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**→ equilibre (a) ← equilibre(a) + \*h**

**SI (equilibre(a) EST EGAL A 0) ALORS**

**\*h ← 0**

SINON

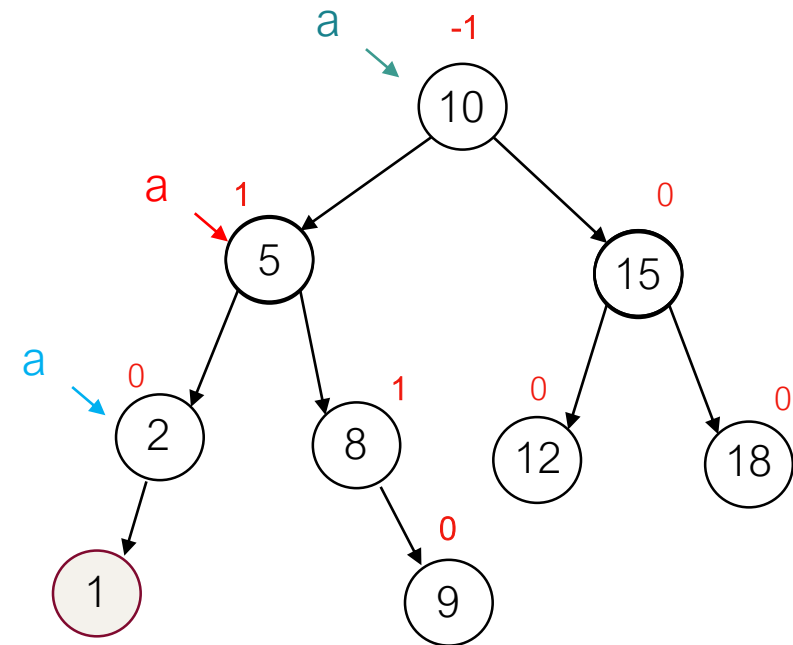
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

\*h = -1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

**→ fg(a) ← insertionAVL(fg(a), e, h)**

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

**fd(a) ← insertionAVL(fd(a), e, h)**

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

**→ SI (equilibre(a) EST EGAL A 0) ALORS**

**\*h ← 0**

SINON

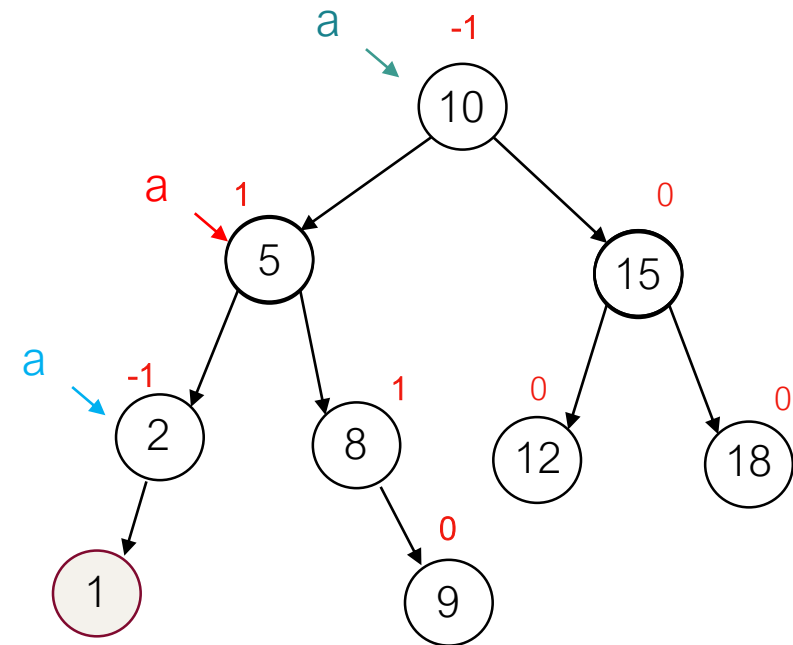
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

$*h = -1$

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

$*h=1$

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

$\rightarrow$  fg(a)  $\leftarrow$  insertionAVL(fg(a), e, h)

$*h \leftarrow -*h$

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a)  $\leftarrow$  insertionAVL(fd(a), e, h)

SINON

$*h = 0$

RETOURNER a

FIN SI

SI ( $*h$  DIFFERENT DE 0) ALORS

$\text{equilibre}(a) \leftarrow \text{equilibre}(a) + *h$

SI ( $\text{equilibre}(a)$  EST EGAL A 0) ALORS

$*h \leftarrow 0$

SINON

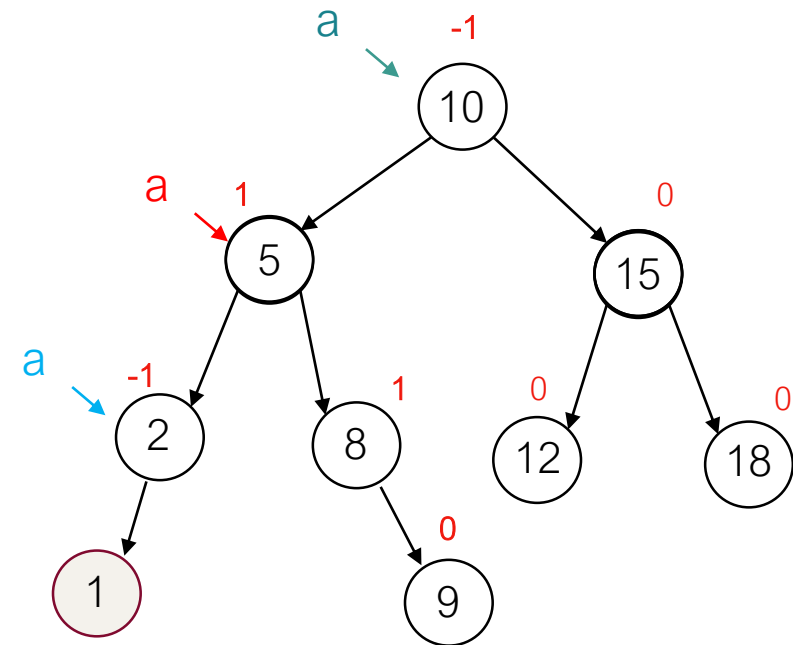
$\rightarrow *h \leftarrow 1$

FIN SI

FIN SI

RETOURNER a

FIN





# Insertion

- Exemple : insertionAVL(a, 1)

\*h = 1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

**→ fg(a) ← insertionAVL(fg(a), e, h)**

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

**fd(a) ← insertionAVL(fd(a), e, h)**

SINON

**\*h = 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre(a) ← equilibre(a) + \*h**

**SI (equilibre(a) EST EGAL A 0) ALORS**

**\*h ← 0**

SINON

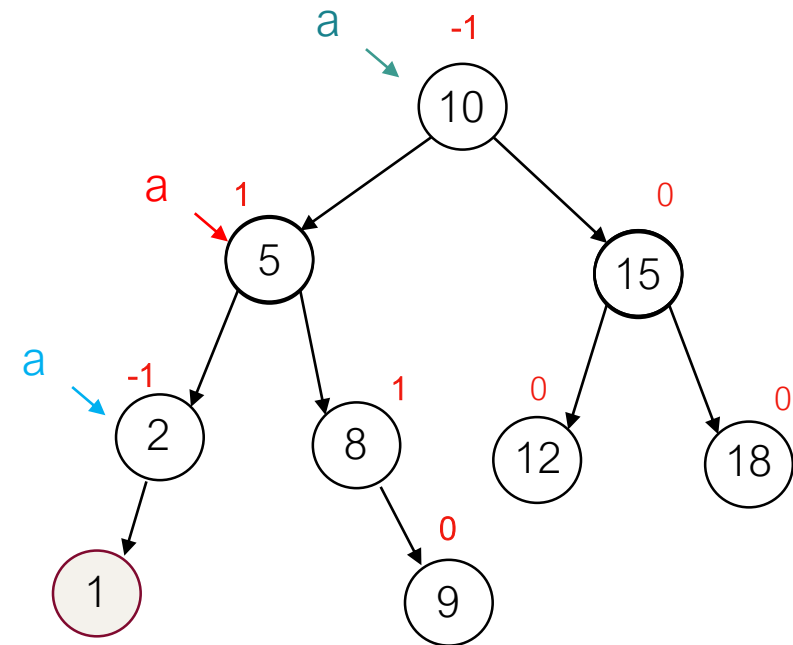
**\*h ← 1**

FIN SI

FIN SI

**→ RETOURNER a**

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

\*h = 1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

**→ fg(a) ← insertionAVL(fg(a), e, h)**

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

**fd(a) ← insertionAVL(fd(a), e, h)**

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

**SI (equilibre(a) EST EGAL A 0) ALORS**

**\*h ← 0**

SINON

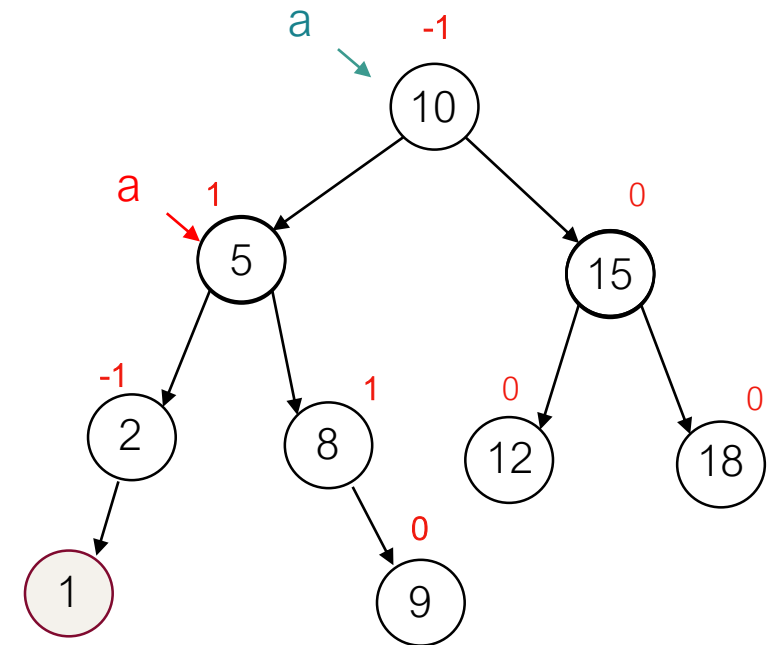
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

\*h = 1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

→ fg(a) ← insertionAVL(fg(a), e, h)

→ \*h ← -\*h

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

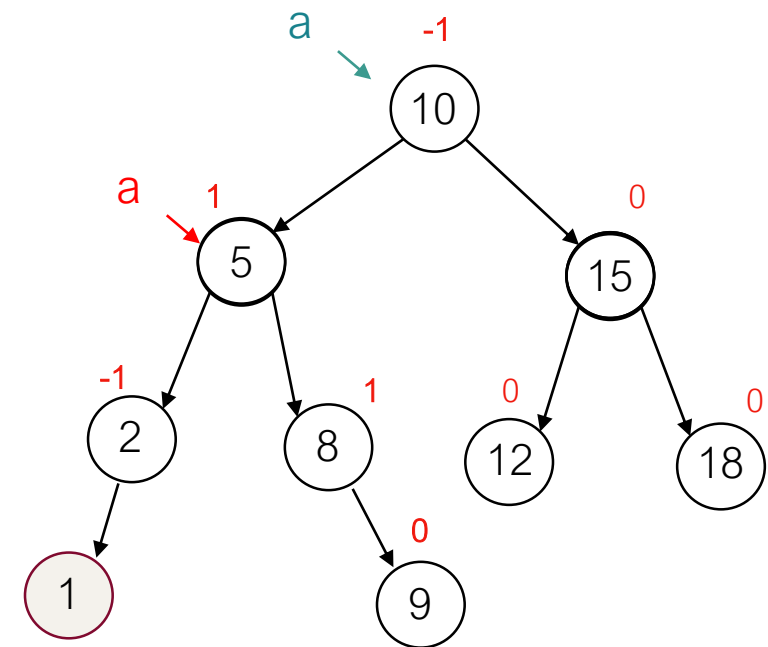
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

$*h = -1$

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

$*h=1$

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

→ fg(a) ← insertionAVL(fg(a), e, h)

$*h \leftarrow -*h$

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

$*h = 0$

RETOURNER a

FIN SI

→ SI ( $*h$  DIFFERENT DE 0) ALORS

$\text{equilibre}(a) \leftarrow \text{equilibre}(a) + *h$

SI ( $\text{equilibre}(a)$  EST EGAL A 0) ALORS

$*h \leftarrow 0$

SINON

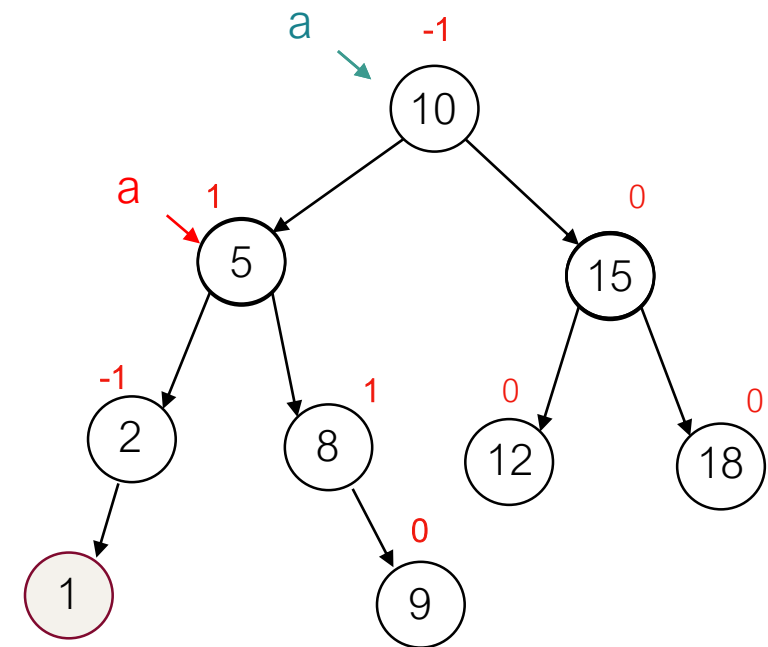
$*h \leftarrow 1$

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

\*h = -1

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

➔ **equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

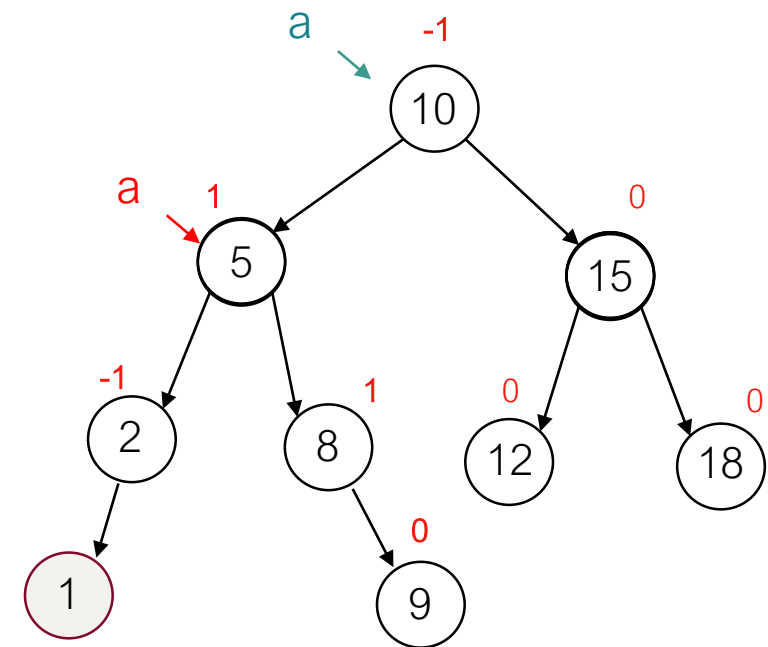
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

$*h = -1$

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

$*h=1$

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

→ fg(a) ← insertionAVL(fg(a), e, h)

$*h \leftarrow -*h$

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

$*h = 0$

RETOURNER a

FIN SI

SI ( $*h$  DIFFERENT DE 0) ALORS

$\text{equilibre}(a) \leftarrow \text{equilibre}(a) + *h$

→ SI ( $\text{equilibre}(a)$  EST EGAL A 0) ALORS

$*h \leftarrow 0$

SINON

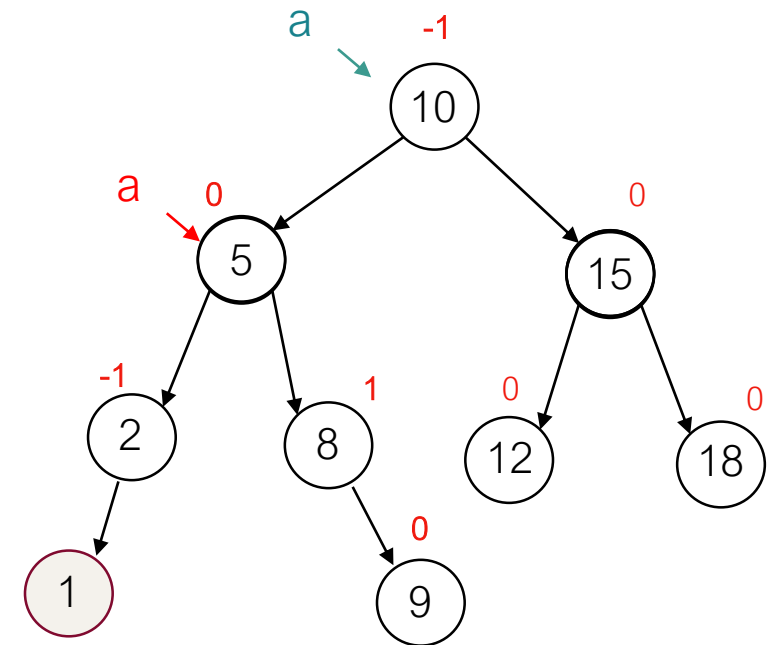
$*h \leftarrow 1$

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

→ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

→ **\*h ← 0**

SINON

**\*h ← 1**

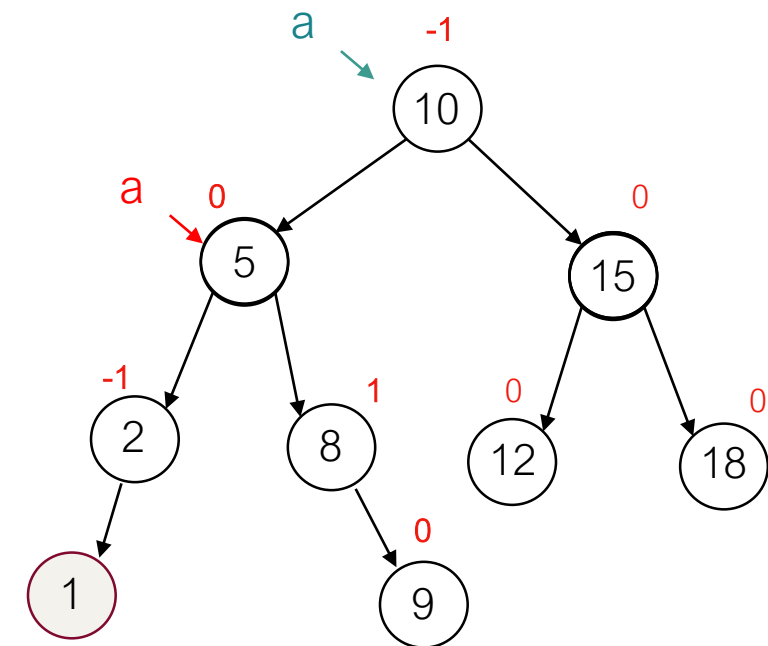
FIN SI

FIN SI

RETOURNER a

FIN

\*h = -1



# Insertion

- Exemple : insertionAVL(a, 1)

\*h = 0

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h = 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre(a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

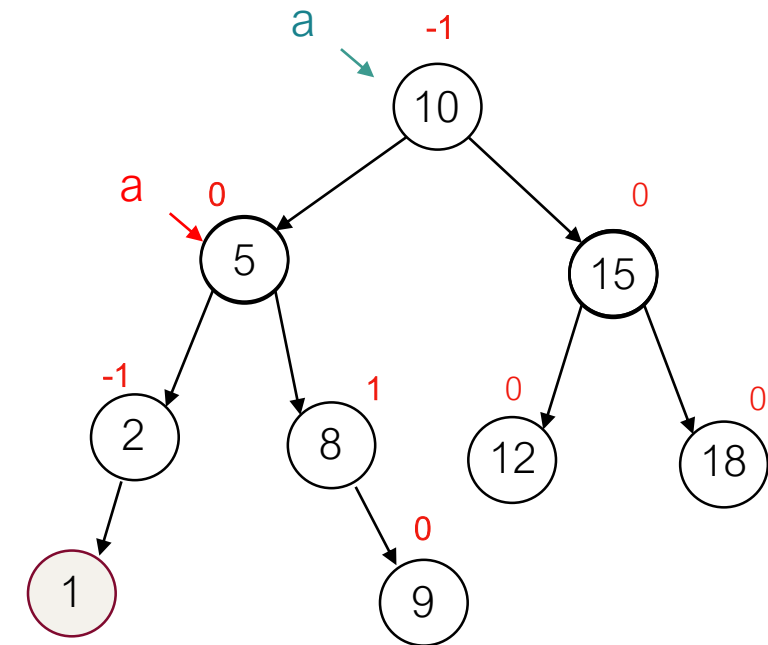
**\*h ← 1**

FIN SI

FIN SI

➔ RETOURNER a

FIN





# Insertion

- Exemple : insertionAVL(a, 1)

\*h = 0

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

➔ fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

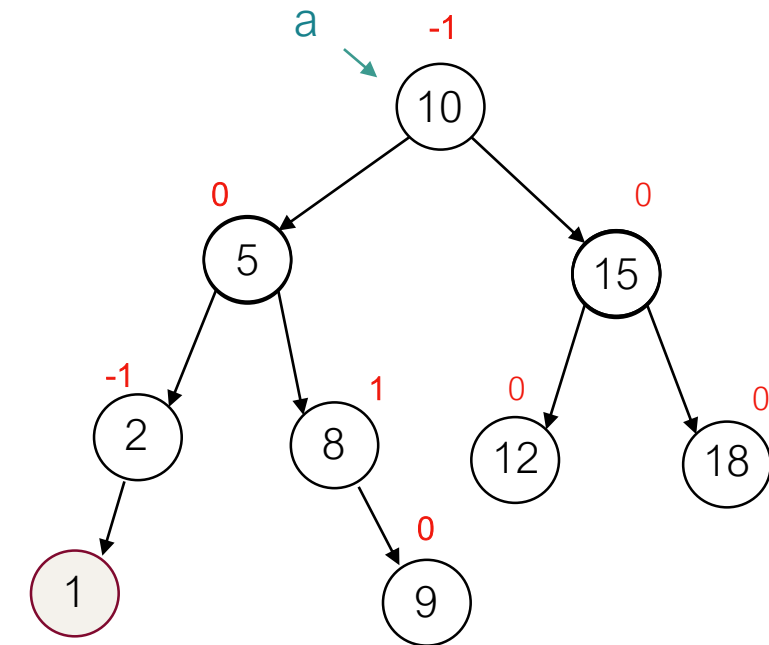
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

\*h = 0

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

fg(a) ← insertionAVL(fg(a), e, h)

→ **\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

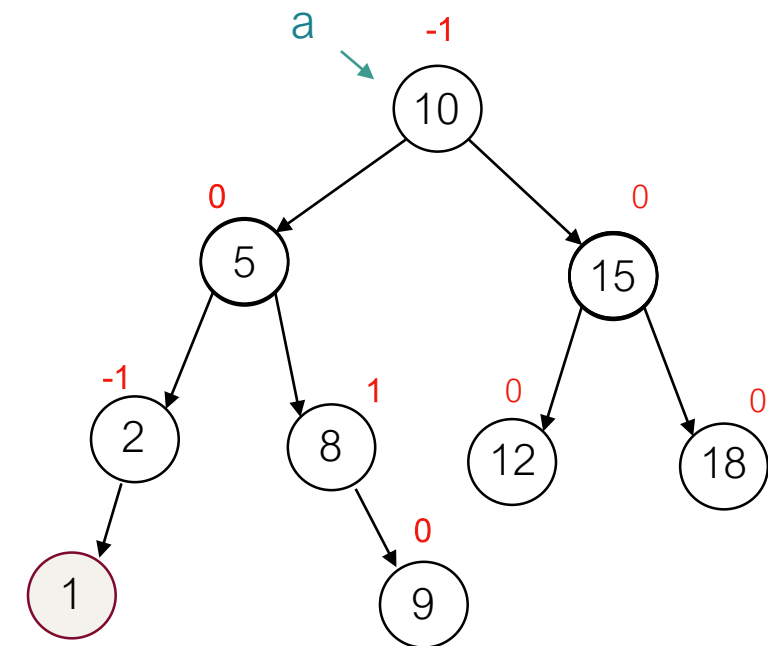
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

\*h = 0

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

➔ SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

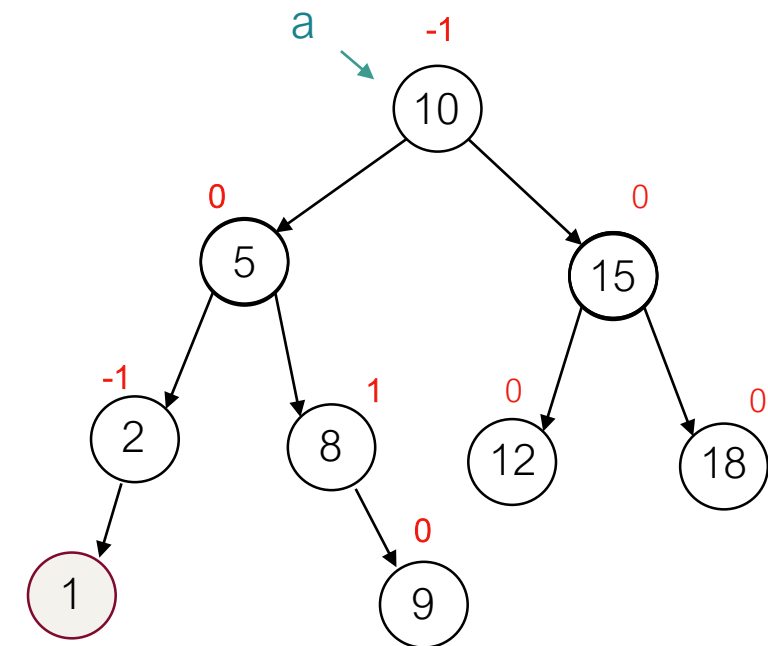
**\*h ← 1**

FIN SI

FIN SI

RETOURNER a

FIN



# Insertion

- Exemple : insertionAVL(a, 1)

`*h = 0`

FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr sur entier) : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

**\*h=1**

RETOURNER creerArbre(e)

SINON Si (e INF. STRICT. A element(a)) ALORS

fg(a) ← insertionAVL(fg(a), e, h)

**\*h ← -\*h**

SINON Si (e SUP. STRICT. A element(a)) ALORS

fd(a) ← insertionAVL(fd(a), e, h)

SINON

**\*h= 0**

RETOURNER a

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

**equilibre (a) ← equilibre(a) + \*h**

SI (equilibre(a) EST EGAL A 0) ALORS

**\*h ← 0**

SINON

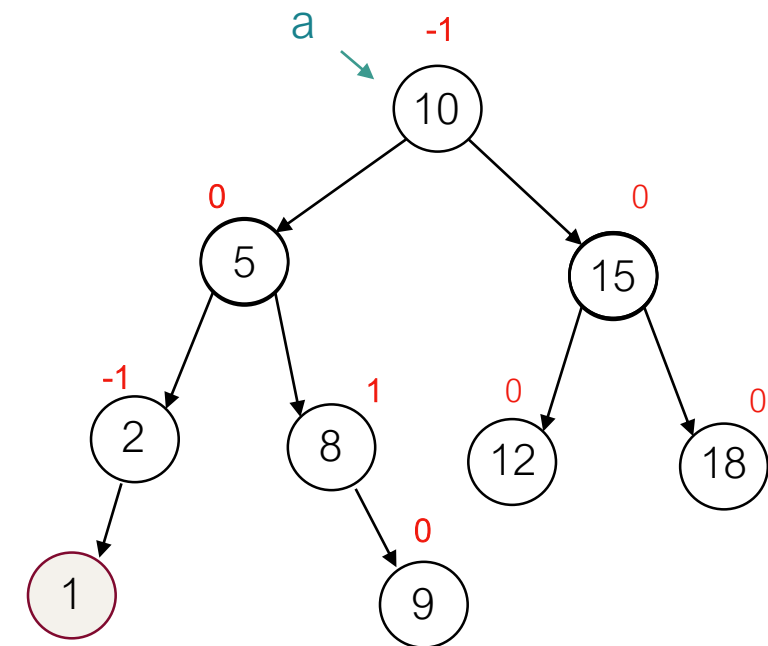
**\*h ← 1**

FIN SI

FIN SI

➔ RETOURNER a

FIN



# Insertion

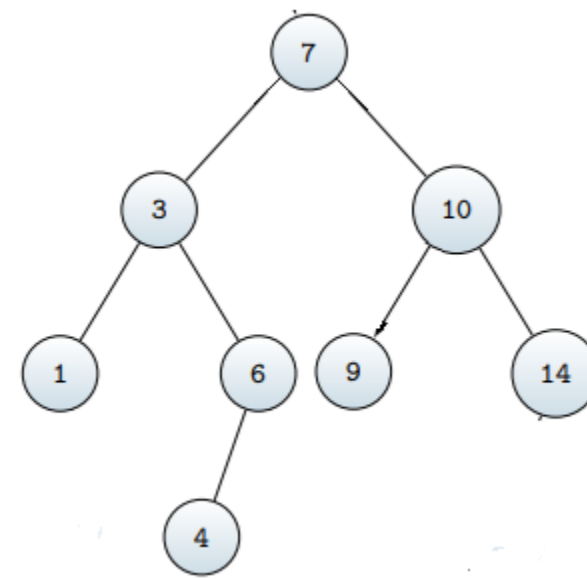
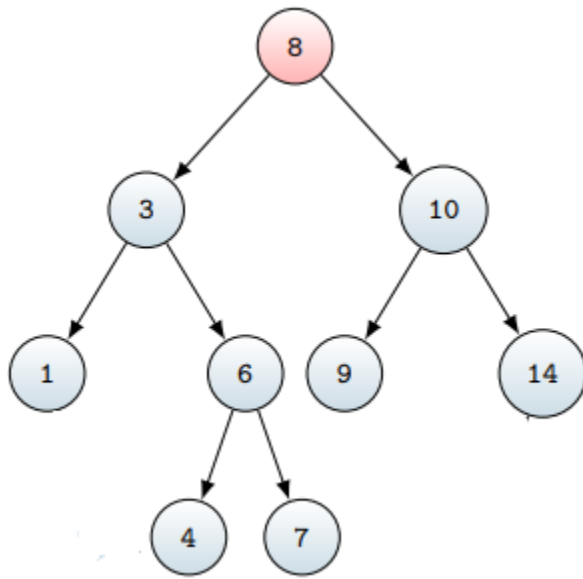
- 3<sup>ème</sup> étape : le rééquilibrage.
- Algorithme :

```
FONCTION insertionAVL(a: ptr Arbre, e: Element, h: ptr
sur entier) : ptr sur Arbre
DEBUT
  SI (a EST EGAL A NULL) ALORS
    *h=1
    RETOURNER creerArbre(e)
  SINON Si (e INF. STRICT. A element(a)) ALORS
    fg(a) ← insertionAVL(fg(a), e, h)
    *h ← -*h
  SINON Si (e SUP. STRICT. A element(a)) ALORS
    fd(a) ← insertionAVL(fd(a), e, h)
  SINON
    *h= 0
    RETOURNER a
FIN SI
```

```
SI (*h DIFFERENT DE 0) ALORS
  equilibre (a) ← equilibre(a) + *h
  a ← equilibrageAVL(a)
  SI (equilibre(a) EST EGAL A 0) ALORS
    *h ← 0
  SINON
    *h ← 1
  FIN SI
FIN SI
RETOURNER a
FIN
```

# Suppression

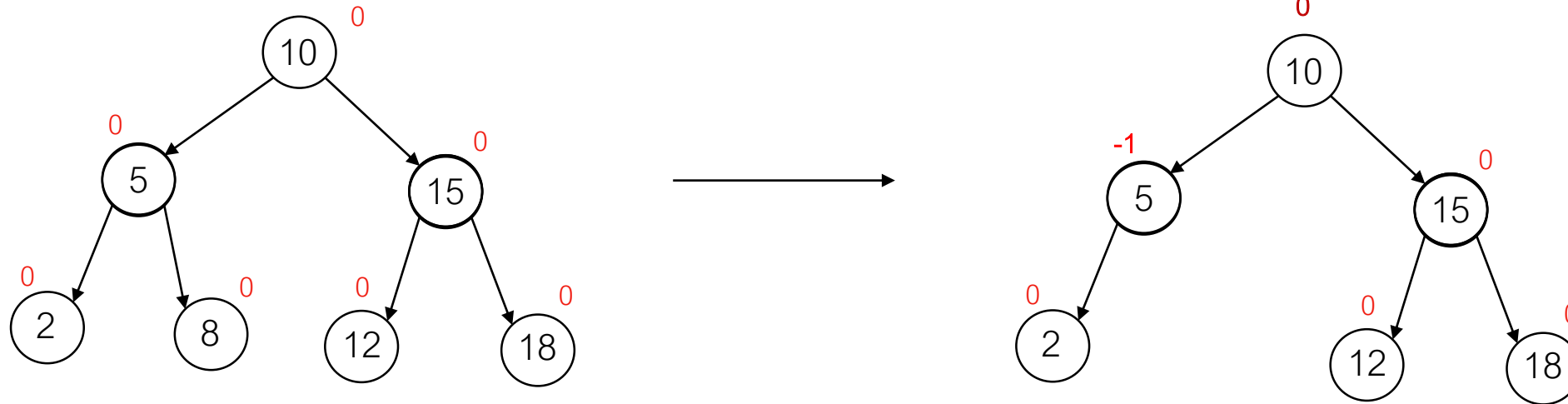
- 1. L'opération de suppression fonctionne comme avec les ABR :
  - Si le nœud à supprimer est une feuille ou n'a qu'un fils, on supprime le nœud / on le remplace par son fils.
  - Sinon on doit échanger les valeurs du nœud à supprimer avec la valeur min du sous arbre droit ou la valeur max du sous arbre gauche.



- Ensuite, un ou plusieurs rééquilibrages peuvent être nécessaires après opération .

# Suppression

- 2. Comme pour l'insertion, on met à jour l'équilibre des nœuds en remontant à partir du nœud supprimé :



- 3. On rééquilibre l'AVL si nécessaire (cf. plus bas, fonction *equilibrageAVL*)

# Suppression

- Algorithme:

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

```
DEBUT
  SI (a EST EGAL A NULL) ALORS
    *h ← 1
    RETOURNER a
  SINON SI (e SUP. STRICT. A element(a)) ALORS // parcours pour trouver le noeud
    fd(a) ← suppressionAVL(fd(a), e)
  SINON SI (e INF. STRICT. A element(a)) ALORS
    fg(a) ← suppressionAVL(fg(a), e)
    *h ← -*h
  SINON SI (existeFilsDroit(a)) ALORS // si il y a un fils droit...
    fg(a) ← suppMinAVL( fd(a), h, adresse(element(a)) ) // ... on cherche le minimum dedans
  SINON
    tmp ← a // le noeud n'a qu'un fils gauche ou aucun fils
    a ← fg(a) // échange avec le fils gauche et suppression
    libérer(tmp)
    *h ← -1
  FIN SI
  SI (*h DIFFERENT DE 0) ALORS
    equilibre(a) ← equilibre (a) + *h
    SI (equilibre(a) EST EGAL A 0) ALORS
      *h ← 0
    SINON
      *h ← 1
    FIN SI
  FIN SI
  RETOURNER a
```

```
SI (a EST EGAL A NULL) ALORS
  *h ← 0
FIN SI
```

FIN



# Suppression

- Algorithme:

```
FONCTION supMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre
VARIABLE
tmp : ptr sur Arbre
DEBUT
  SI (fg(a) EST EGAL A NULL) ALORS                                // Si il n'y a plus de fils gauche...
    *pe ← element(a)                                             // ... alors on a trouvé la plus petite valeur de l'arbre
    *h ← -1
    tmp ← a
    a ← fd(a)                                                    // on remplace le noeud actuel par le fils droit...
    liberer(tmp)                                                 // ... et on libère la mémoire du noeud
    RETOURNER(a)

  SINON
    fg(a) ← supMinAVL(fg(a), h, pe) // appel récursif sur le sous-arbre de gauche
    *h ← -*h

  FIN SI
  SI (*h DIFFERENT DE 0) ALORS
    equilibre(a) ← equilibre(a) + *h // mise à jour du facteur d'équilibrage
    SI (equilibre(a) EST EGAL A 0) ALORS
      *h ← -1

    SINON
      *h ← 0

    FIN SI
  FIN SI
  RETOURNER a
FIN
```

# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

→ DEBUT

SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

SINON SI (e SUP. STRICT. A element(a)) ALORS

fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

\*h ← 1

FIN SI

FIN SI

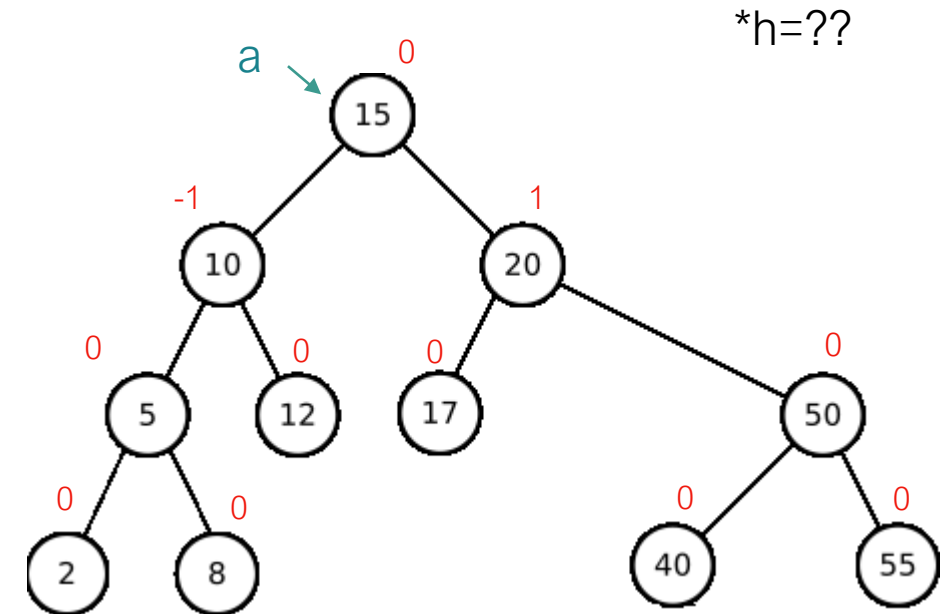
RETOURNER a

FIN

SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI



# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

DEBUT

➔ SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

SINON SI (e SUP. STRICT. A element(a)) ALORS

fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

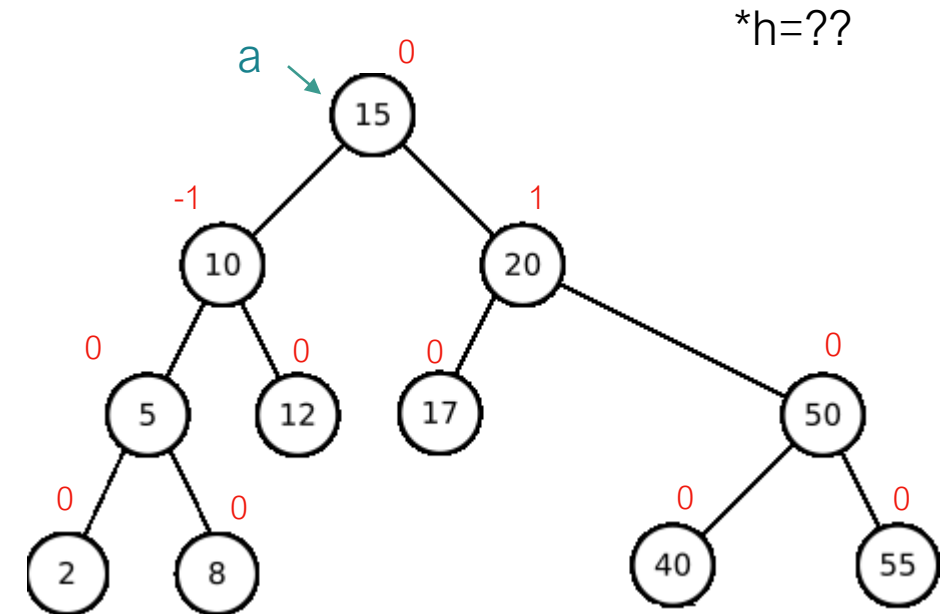
\*h ← 1

FIN SI

FIN SI

RETOURNER a

FIN



SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI

# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

→ SINON SI (e SUP. STRICT. A element(a)) ALORS  
fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS  
fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

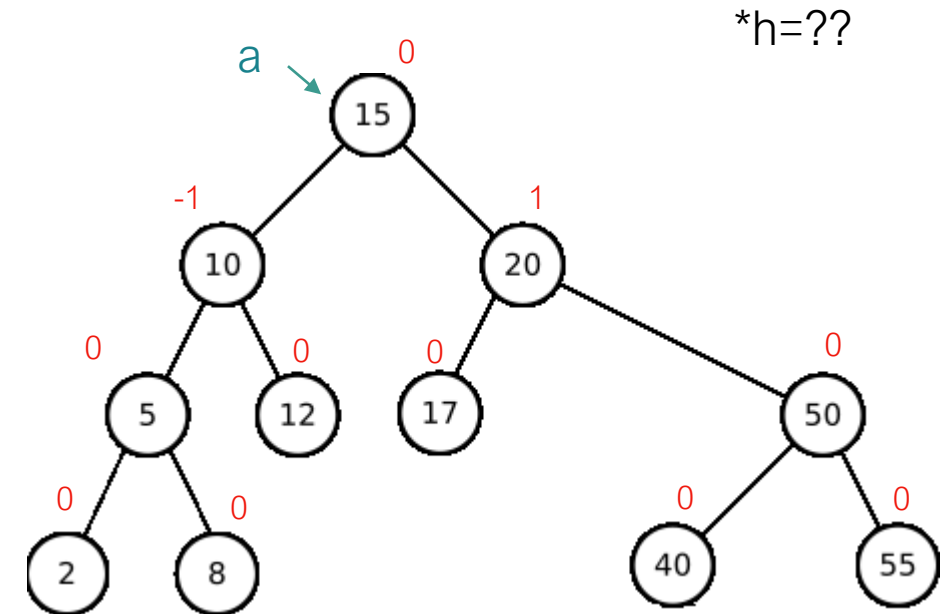
\*h ← 1

FIN SI

FIN SI

RETOURNER a

FIN



SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI

# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

SINON SI (e SUP. STRICT. A element(a)) ALORS

➔ fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

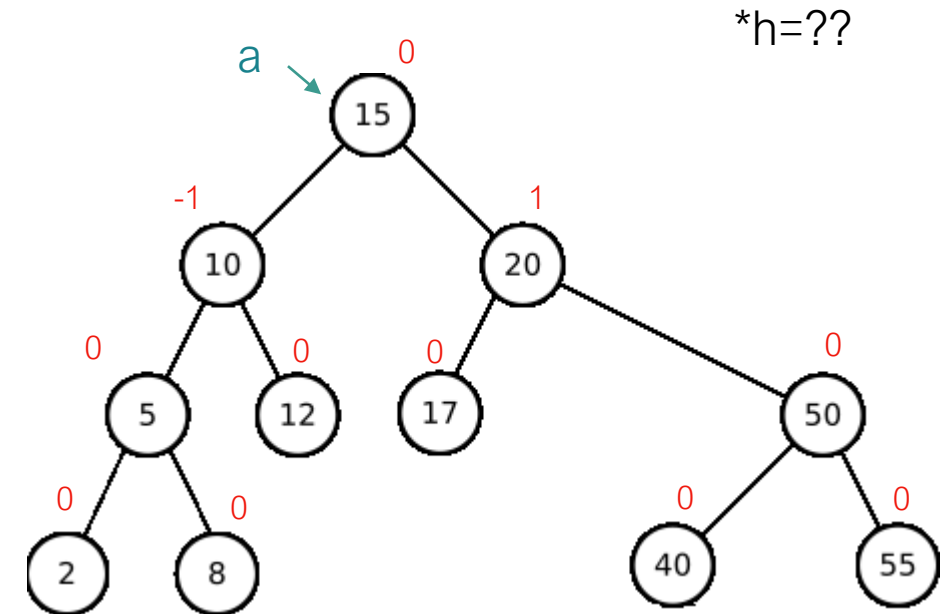
\*h ← 1

FIN SI

FIN SI

RETOURNER a

FIN



SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI

# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

→ DEBUT

SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

SINON SI (e SUP. STRICT. A element(a)) ALORS

→ fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

\*h ← 1

FIN SI

FIN SI

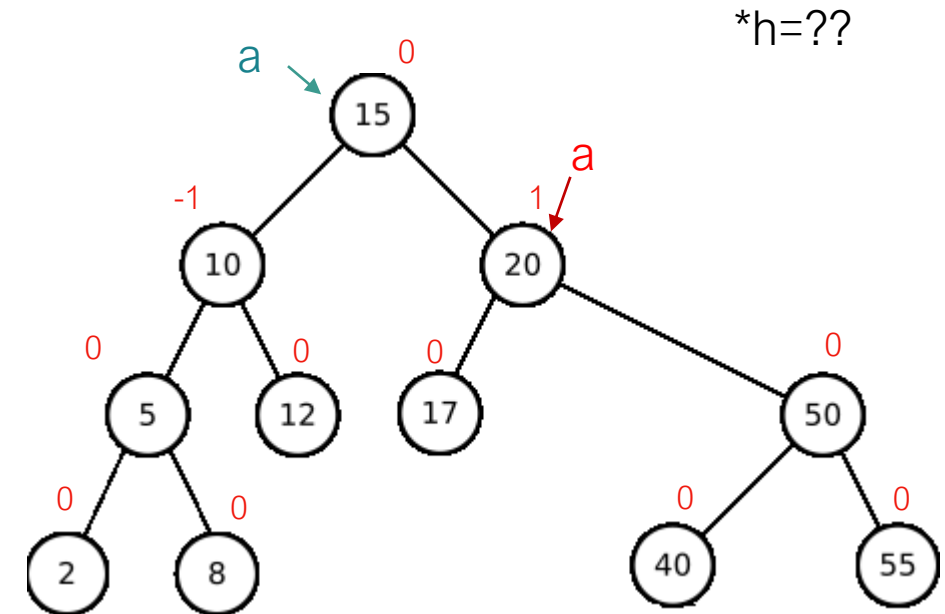
RETOURNER a

FIN

SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI



# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

DEBUT

➔ SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

SINON SI (e SUP. STRICT. A element(a)) ALORS

➔ fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

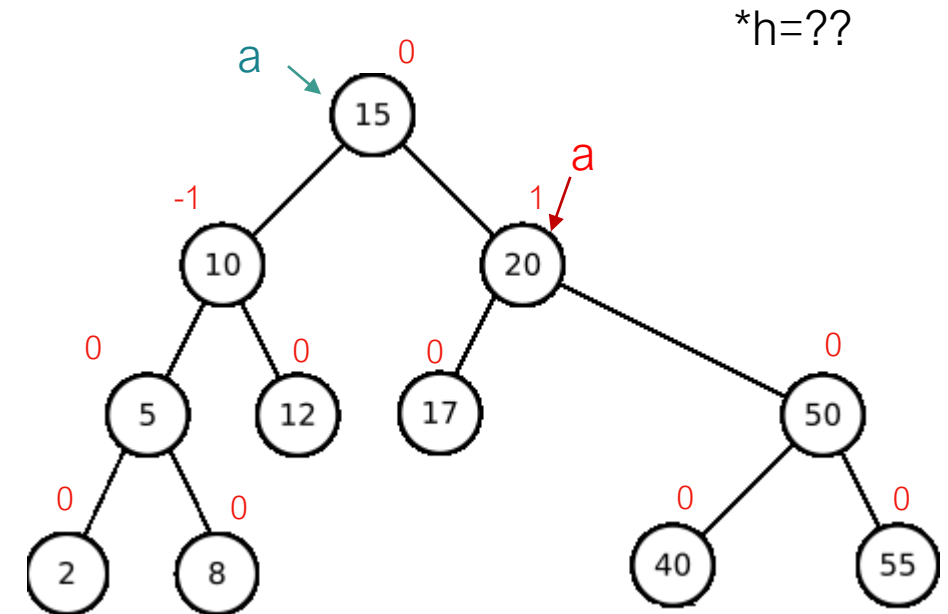
\*h ← 1

FIN SI

FIN SI

RETOURNER a

FIN



SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI

# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

→ SINON SI (e SUP. STRICT. A element(a)) ALORS

→ fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

\*h ← 1

FIN SI

FIN SI

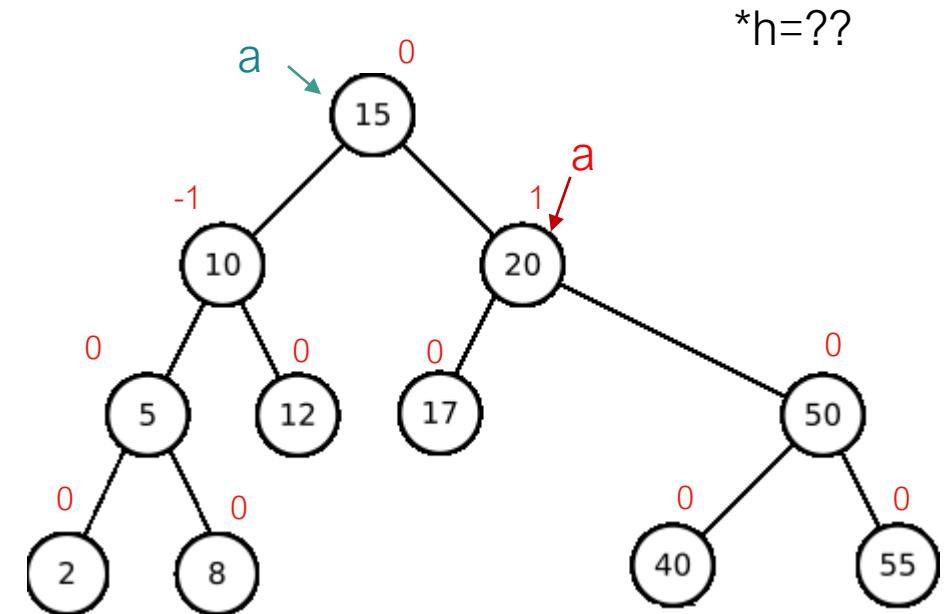
RETOURNER a

FIN

SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI





# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

SINON SI (e SUP. STRICT. A element(a)) ALORS

→ fd(a) ← suppressionAVL(fd(a), e)

→ SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

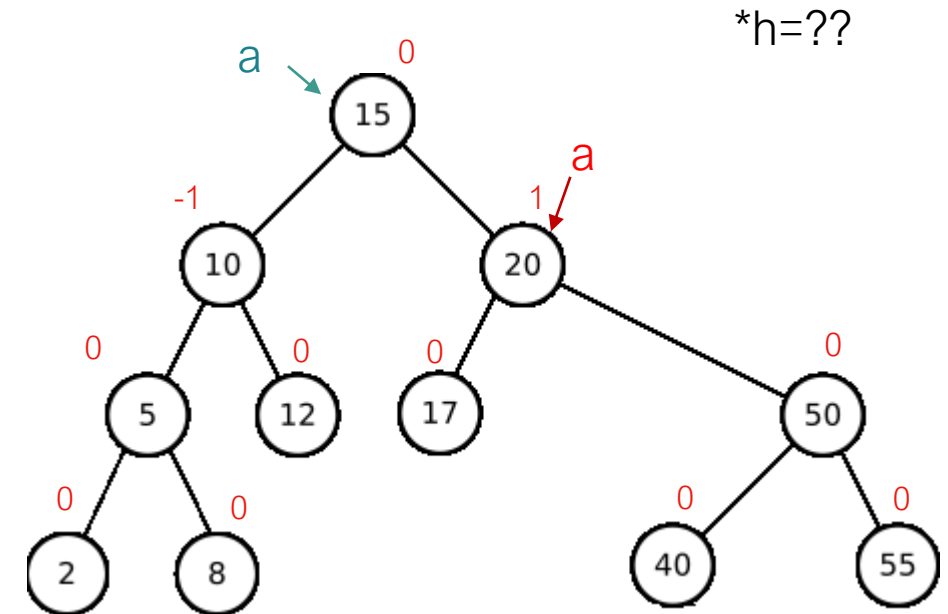
\*h ← 1

FIN SI

FIN SI

RETOURNER a

FIN



SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI

# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

SINON SI (e SUP. STRICT. A element(a)) ALORS

→ fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

→ SINON SI (existeFilsDroit(a)) ALORS

fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

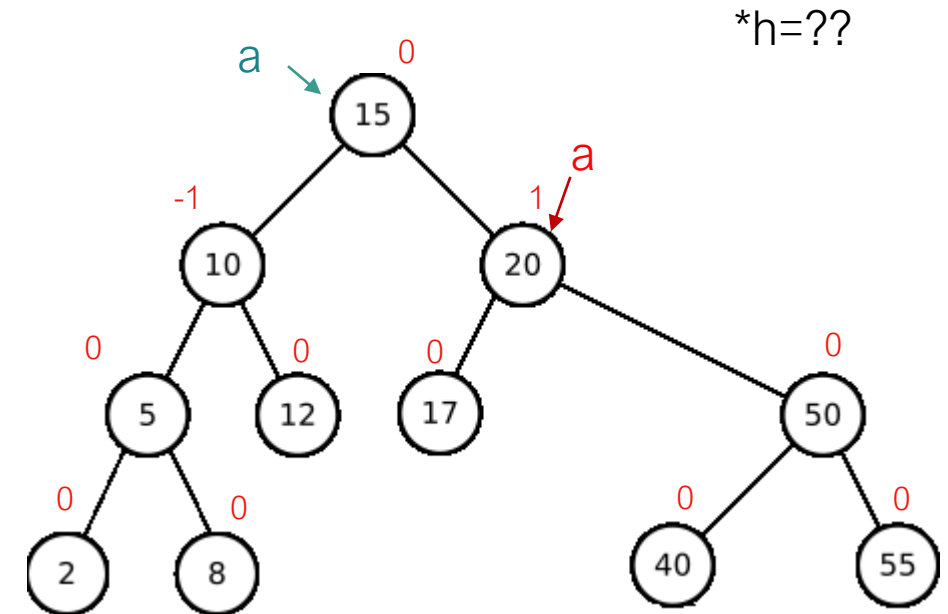
\*h ← 1

FIN SI

FIN SI

RETOURNER a

FIN



SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI

# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

SINON SI (e SUP. STRICT. A element(a)) ALORS

➔ fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

➔ fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

\*h ← 1

FIN SI

FIN SI

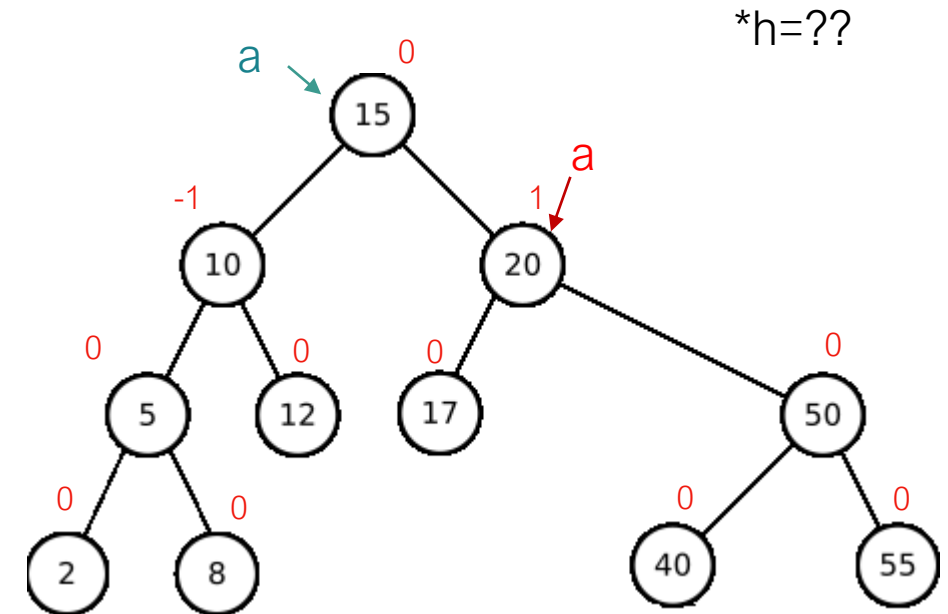
RETOURNER a

FIN

SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

→ DEBUT

SI (`fg(a)` EST EGAL A NULL) ALORS

`*pe ← element(a)`

`*h ← -1`

`tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

RETOURNER(`a`)

SINON

`fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI (`*h` DIFFERENT DE 0) ALORS

`equilibre(a) ← equilibre(a) + *h`

SI (`equilibre(a)` EST EGAL A 0) ALORS

`*h ← -1`

SINON

`*h ← 0`

FIN SI

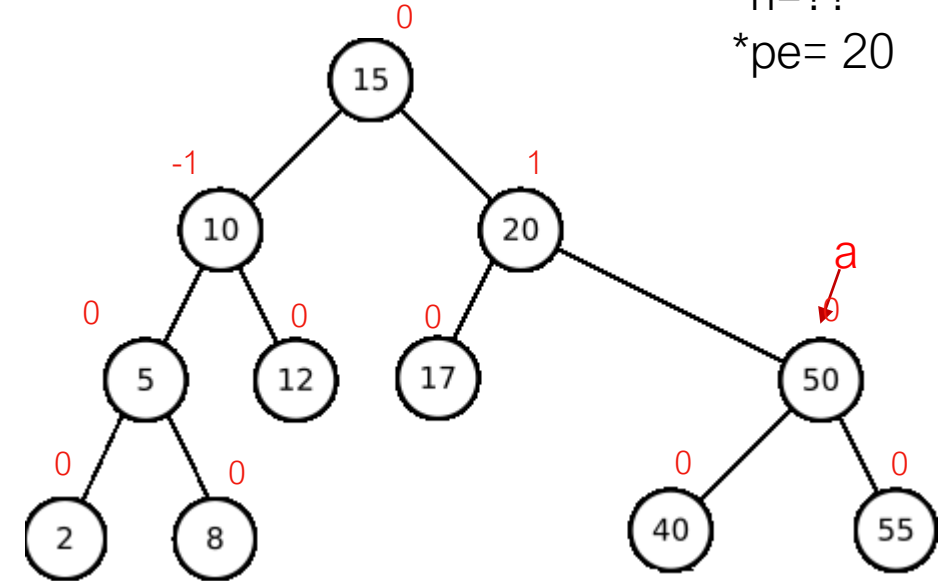
FIN SI

RETOURNER `a`

FIN

\*h=??

\*pe= 20



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

➔ SI (`fg(a)` EST EGAL A NULL) ALORS

`*pe ← element(a)`

`*h ← -1`

`tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

RETOURNER(`a`)

SINON

`fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI (`*h` DIFFERENT DE 0) ALORS

`equilibre(a) ← equilibre(a) + *h`

SI (`equilibre(a)` EST EGAL A 0) ALORS

`*h ← -1`

SINON

`*h ← 0`

FIN SI

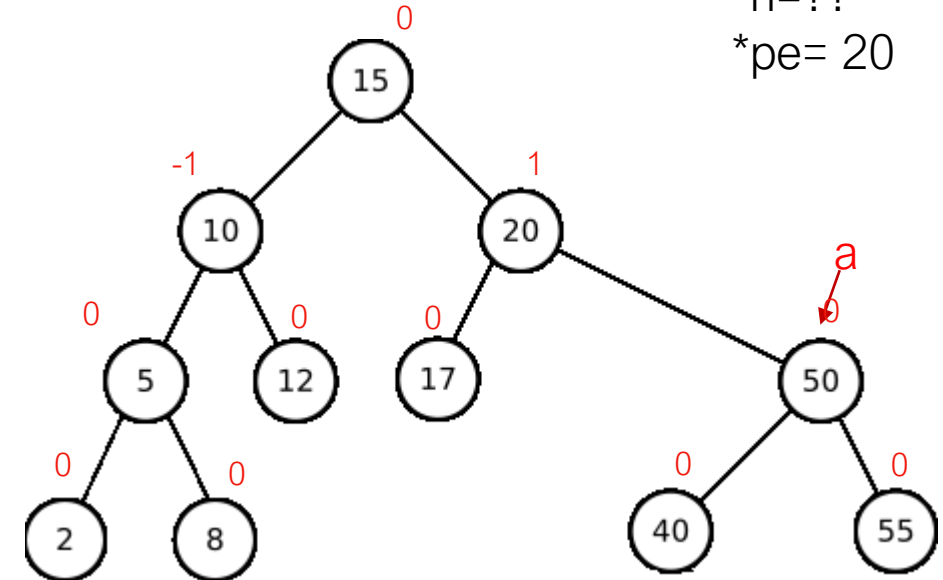
FIN SI

RETOURNER `a`

FIN

\*h=??

\*pe= 20



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

SI (fg(a) EST EGAL A NULL) ALORS

`*pe ← element(a)`

`*h ← -1`

`tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

RETOURNER(a)

→ SINON

`fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

`equilibre(a) ← equilibre(a) + *h`

SI (equilibre(a) EST EGAL A 0) ALORS

`*h ← -1`

SINON

`*h ← 0`

FIN SI

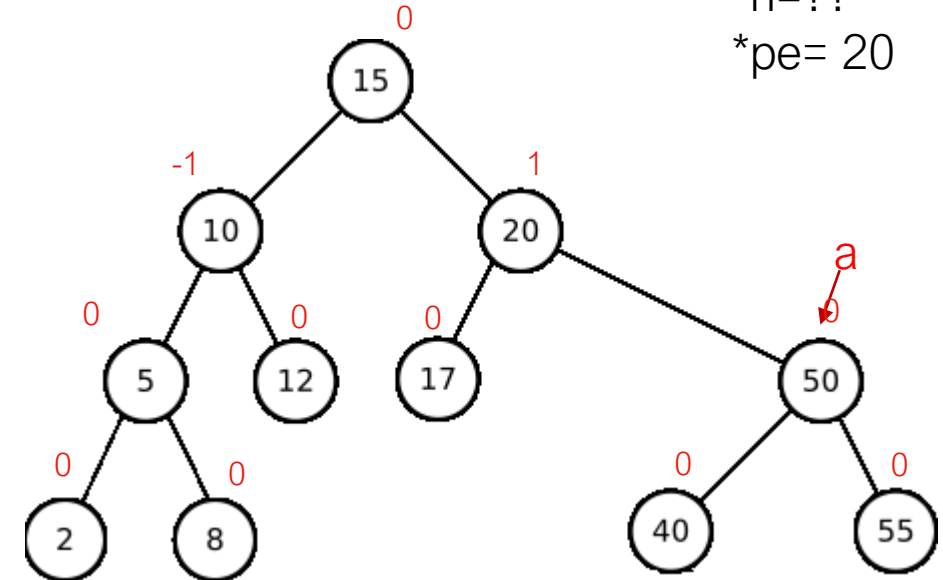
FIN SI

RETOURNER a

FIN

\*h=??

\*pe= 20



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

SI `(fg(a) EST EGAL A NULL) ALORS`

`*pe ← element(a)`

`*h ← -1`

`tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

RETOURNER(a)

SINON

➔ `fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI `(*h DIFFERENT DE 0) ALORS`

`equilibre(a) ← equilibre(a) + *h`

SI `(equilibre(a) EST EGAL A 0) ALORS`

`*h ← -1`

SINON

`*h ← 0`

FIN SI

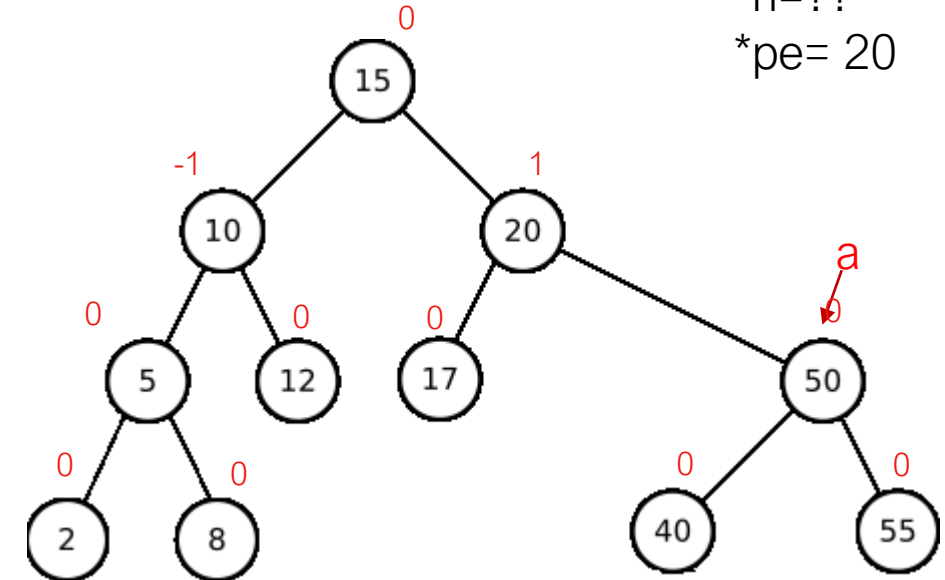
FIN SI

RETOURNER a

FIN

\*h=??

\*pe= 20



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

→ DEBUT

SI `(fg(a) EST EGAL A NULL) ALORS`

`*pe ← element(a)`

`*h ← -1`

`tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

RETOURNER(a)

SINON

→ `fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI `(*h DIFFERENT DE 0) ALORS`

`equilibre(a) ← equilibre(a) + *h`

SI `(equilibre(a) EST EGAL A 0) ALORS`

`*h ← -1`

SINON

`*h ← 0`

FIN SI

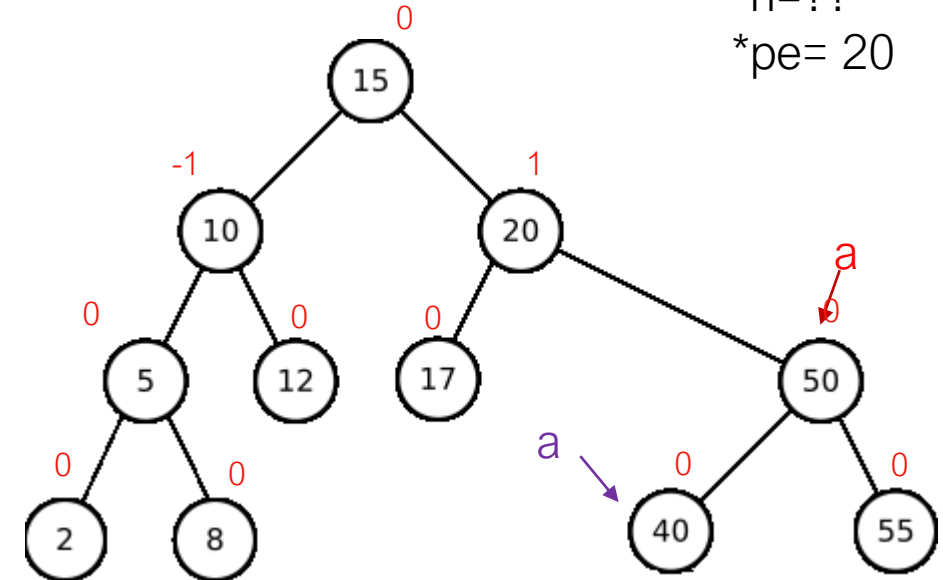
FIN SI

RETOURNER a

FIN

\*h=??

\*pe= 20





# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

➔ SI (`fg(a)` EST EGAL A NULL) ALORS

`*pe ← element(a)`

`*h ← -1`

`tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

RETOURNER(`a`)

SINON

➔ `fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI (`*h` DIFFERENT DE 0) ALORS

`equilibre(a) ← equilibre(a) + *h`

SI (`equilibre(a)` EST EGAL A 0) ALORS

`*h ← -1`

SINON

`*h ← 0`

FIN SI

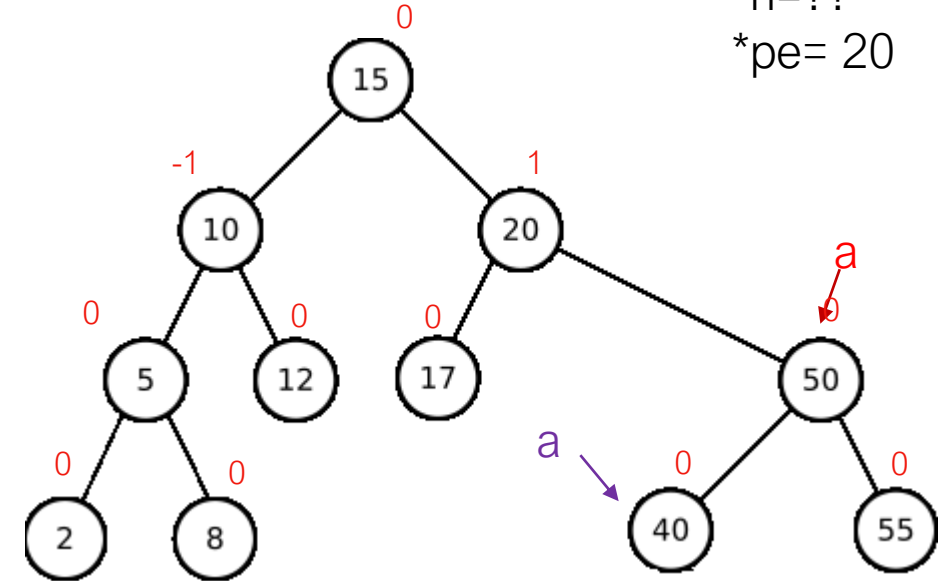
FIN SI

RETOURNER `a`

FIN

`*h=??`

`*pe= 20`



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

SI `(fg(a) EST EGAL A NULL) ALORS`

    ➡ `*pe ← element(a)`

`*h ← -1`

`tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

    RETOURNER(a)

SINON

    ➡ `fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI `(*h DIFFERENT DE 0) ALORS`

`equilibre(a) ← equilibre(a) + *h`

    SI `(equilibre(a) EST EGAL A 0) ALORS`

`*h ← -1`

    SINON

`*h ← 0`

    FIN SI

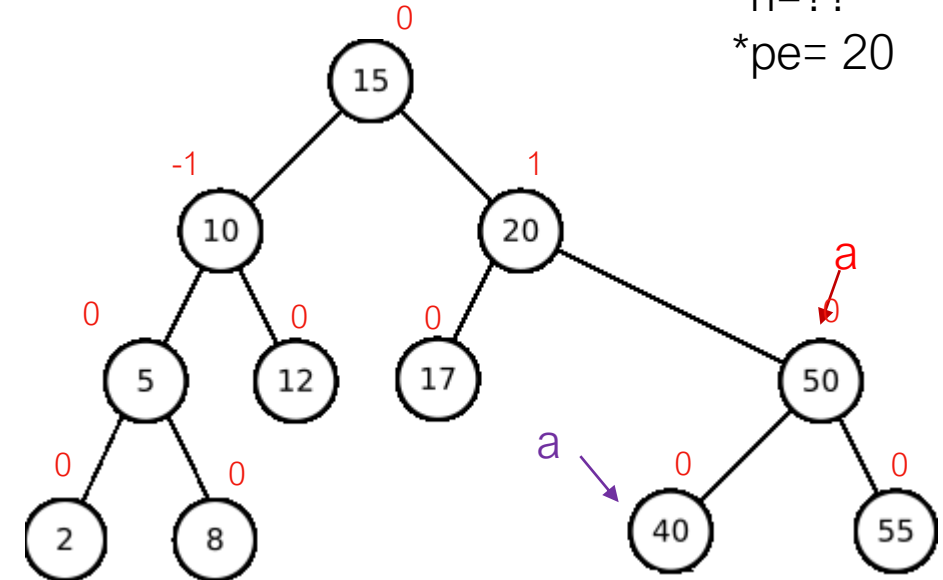
FIN SI

RETOURNER a

FIN

\*h=??

\*pe= 20



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

SI `(fg(a) EST EGAL A NULL) ALORS`

`*pe ← element(a)`

➡ `*h ← -1`

`tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

RETOURNER(a)

SINON

➡ `fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI `(*h DIFFERENT DE 0) ALORS`

`equilibre(a) ← equilibre(a) + *h`

SI `(equilibre(a) EST EGAL A 0) ALORS`

`*h ← -1`

SINON

`*h ← 0`

FIN SI

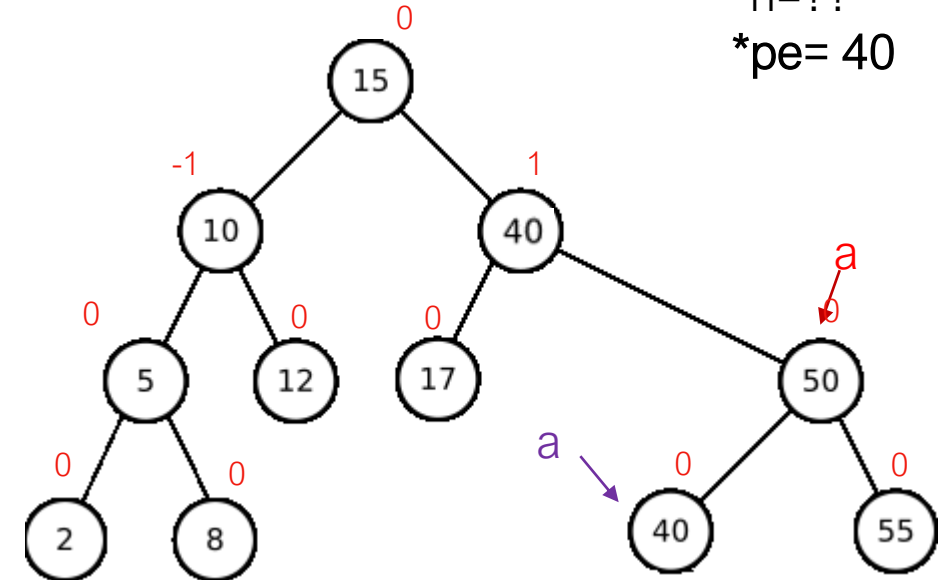
FIN SI

RETOURNER a

FIN

\*h=??

\*pe= 40



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

SI `(fg(a) EST EGAL A NULL)` ALORS

`*pe ← element(a)`

`*h ← -1`

➡ `tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

RETOURNER `(a)`

SINON

➡ `fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI `(*h DIFFERENT DE 0)` ALORS

`equilibre(a) ← equilibre(a) + *h`

SI `(equilibre(a) EST EGAL A 0)` ALORS

`*h ← -1`

SINON

`*h ← 0`

FIN SI

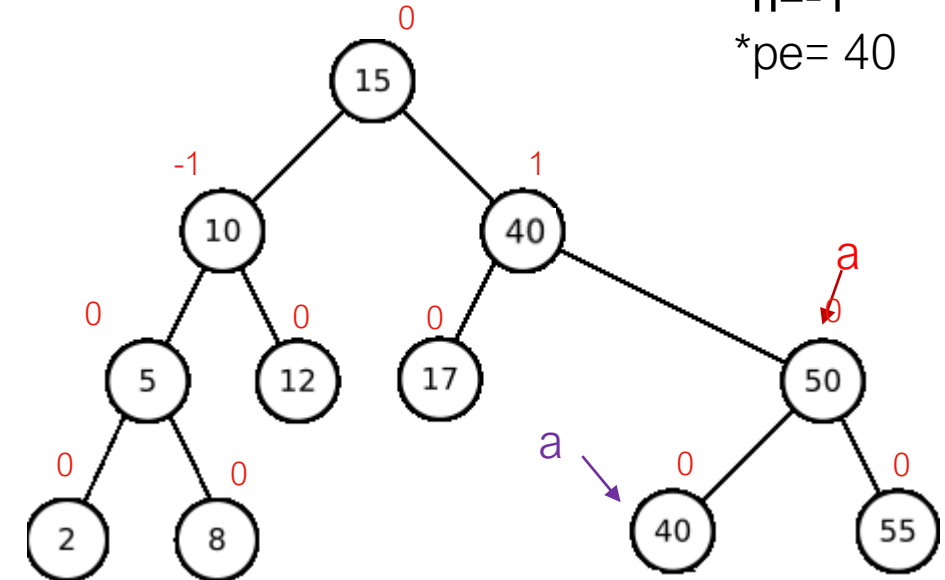
FIN SI

RETOURNER `a`

FIN

`*h = -1`

`*pe = 40`



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

SI (`fg(a)` EST EGAL A NULL) ALORS

`*pe ← element(a)`

`*h ← -1`

`tmp ← a`

➔ `a ← fd(a)`

`liberer(tmp)`

RETOURNER(`a`)

SINON

➔ `fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI (`*h` DIFFERENT DE 0) ALORS

`equilibre(a) ← equilibre(a) + *h`

SI (`equilibre(a)` EST EGAL A 0) ALORS

`*h ← -1`

SINON

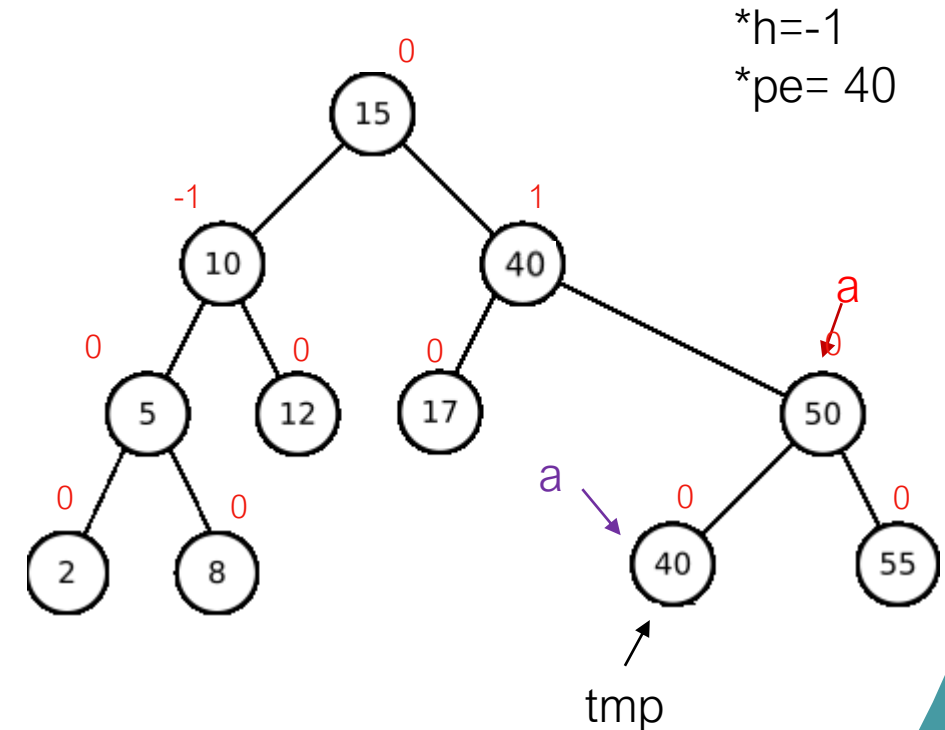
`*h ← 0`

FIN SI

FIN SI

RETOURNER `a`

FIN



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

SI (`fg(a)` EST EGAL A NULL) ALORS

`*pe ← element(a)`

`*h ← -1`

`tmp ← a`

`a ← fd(a)`

➡ `liberer(tmp)`

RETOURNER(`a`)

SINON

➡ `fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI (`*h` DIFFERENT DE 0) ALORS

`equilibre(a) ← equilibre(a) + *h`

SI (`equilibre(a)` EST EGAL A 0) ALORS

`*h ← -1`

SINON

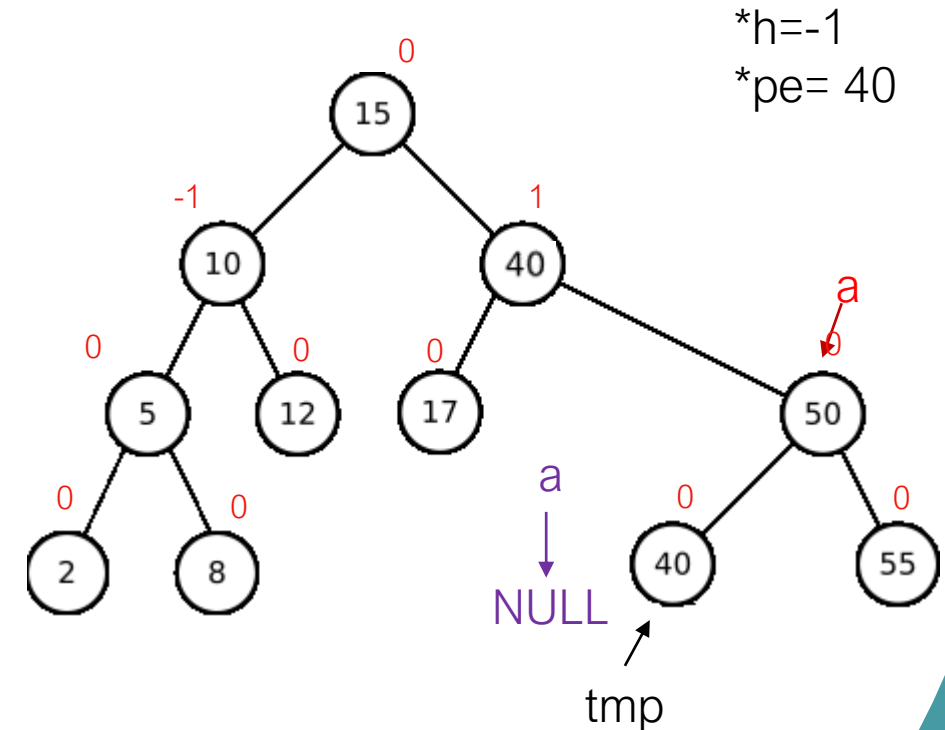
`*h ← 0`

FIN SI

FIN SI

RETOURNER `a`

FIN



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

SI (`fg(a)` EST EGAL A NULL) ALORS

`*pe ← element(a)`

`*h ← -1`

`tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

➡ `RETOURNER(a)`

SINON

➡ `fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI (`*h` DIFFERENT DE 0) ALORS

`equilibre(a) ← equilibre(a) + *h`

SI (`equilibre(a)` EST EGAL A 0) ALORS

`*h ← -1`

SINON

`*h ← 0`

FIN SI

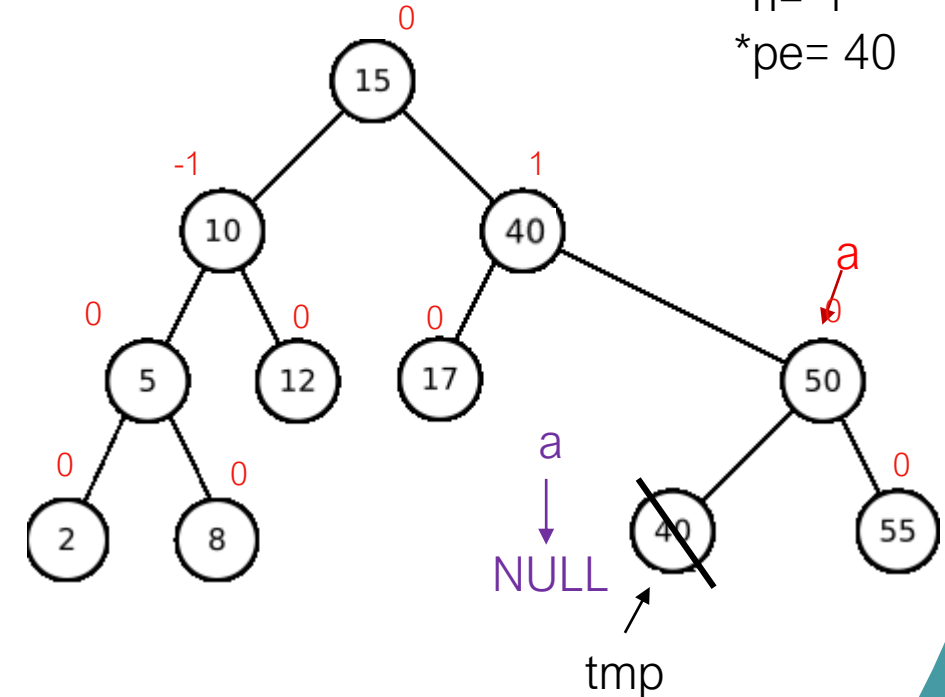
FIN SI

`RETOURNER a`

FIN

`*h = -1`

`*pe = 40`



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

SI `(fg(a) EST EGAL A NULL) ALORS`

`*pe ← element(a)`

`*h ← -1`

`tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

RETOURNER(a)

SINON

➔ `fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI `(*h DIFFERENT DE 0) ALORS`

`equilibre(a) ← equilibre(a) + *h`

SI `(equilibre(a) EST EGAL A 0) ALORS`

`*h ← -1`

SINON

`*h ← 0`

FIN SI

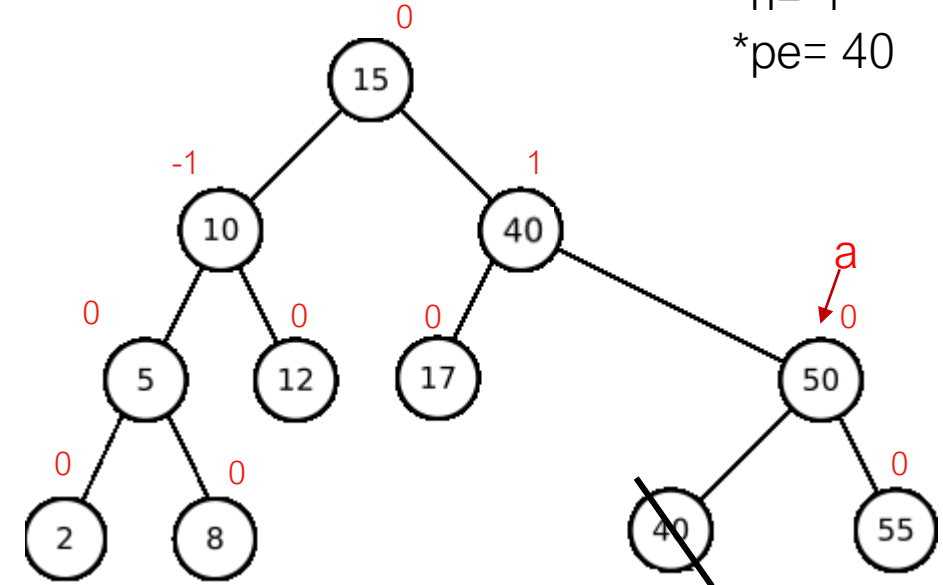
FIN SI

RETOURNER a

FIN

\*h=-1

\*pe= 40





# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

SI `(fg(a) EST EGAL A NULL) ALORS`

`*pe ← element(a)`

`*h ← -1`

`tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

RETOURNER(a)

SINON

`fg(a) ← suppMinAVL(fg(a), h, pe)`

→ `*h ← -*h`

FIN SI

SI `(*h DIFFERENT DE 0) ALORS`

`equilibre(a) ← equilibre(a) + *h`

SI `(equilibre(a) EST EGAL A 0) ALORS`

`*h ← -1`

SINON

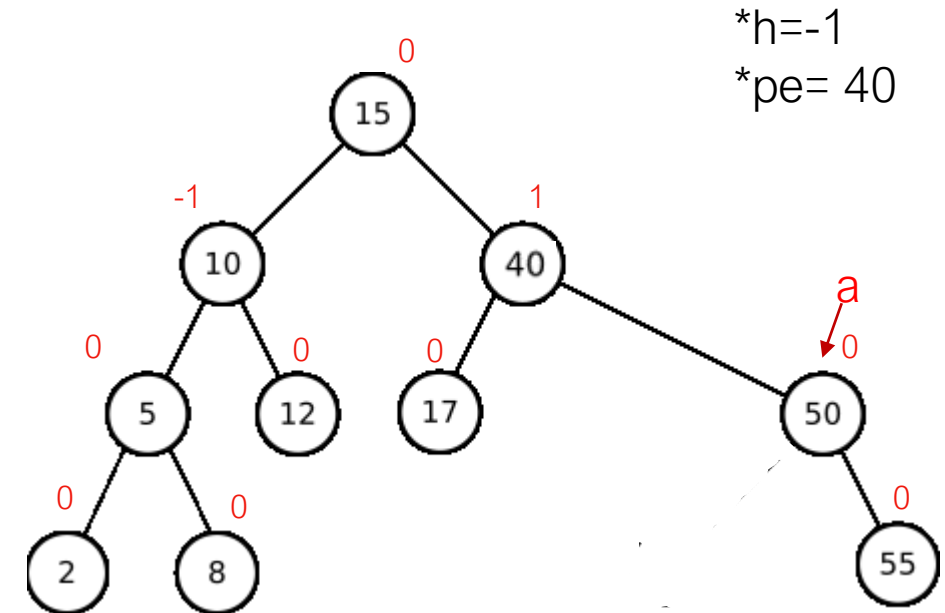
`*h ← 0`

FIN SI

FIN SI

RETOURNER a

FIN



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

SI `(fg(a) EST EGAL A NULL) ALORS`

`*pe ← element(a)`

`*h ← -1`

`tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

RETOURNER(a)

SINON

`fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

➔ SI `(*h DIFFERENT DE 0) ALORS`

`equilibre(a) ← equilibre(a) + *h`

SI `(equilibre(a) EST EGAL A 0) ALORS`

`*h ← -1`

SINON

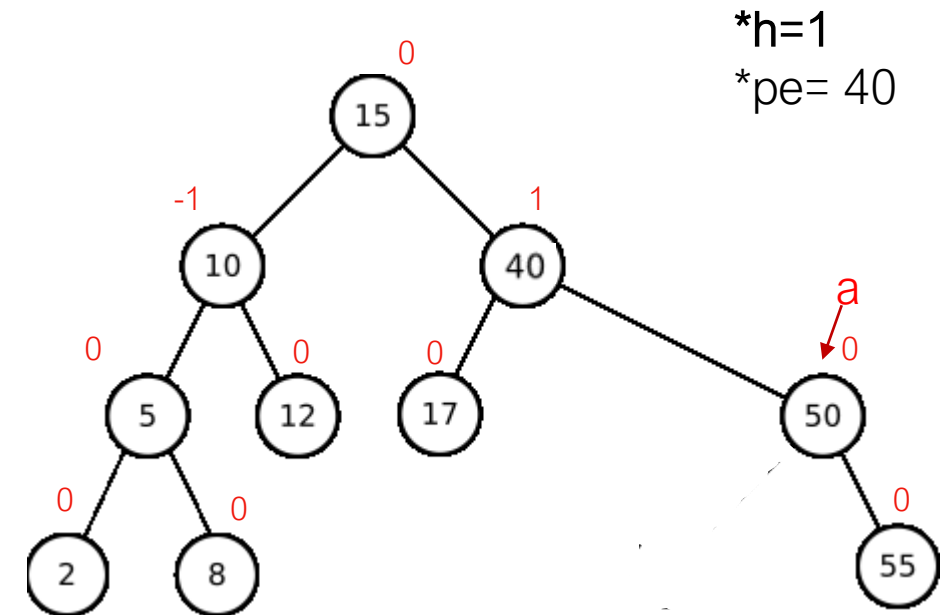
`*h ← 0`

FIN SI

FIN SI

RETOURNER a

FIN



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

SI `(fg(a) EST EGAL A NULL) ALORS`

`*pe ← element(a)`

`*h ← -1`

`tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

RETOURNER(a)

SINON

`fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI `(*h DIFFERENT DE 0) ALORS`

➔ `equilibre(a) ← equilibre(a) + *h`

SI `(equilibre(a) EST EGAL A 0) ALORS`

`*h ← -1`

SINON

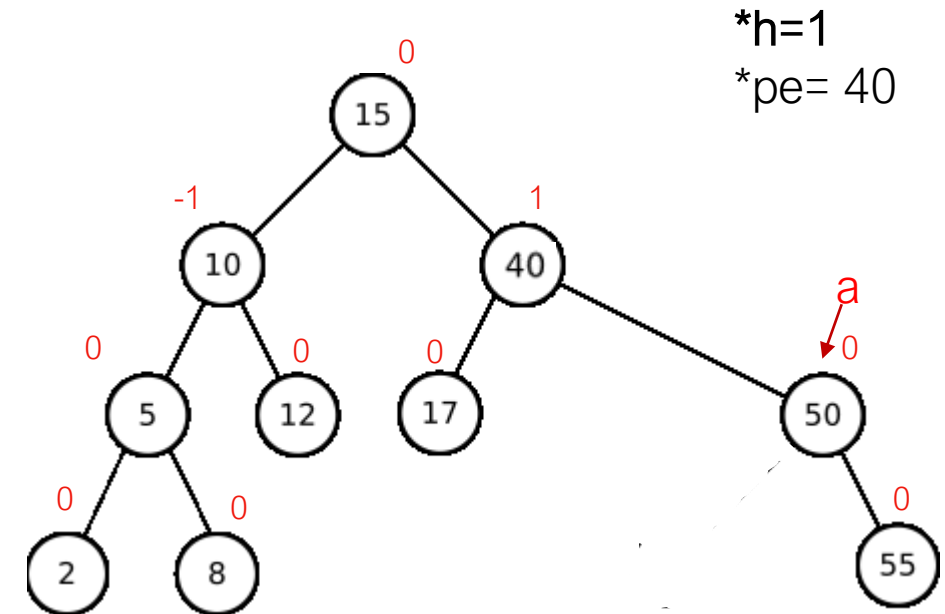
`*h ← 0`

FIN SI

FIN SI

RETOURNER a

FIN



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

SI `(fg(a) EST EGAL A NULL) ALORS`

`*pe ← element(a)`

`*h ← -1`

`tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

RETOURNER(a)

SINON

`fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI `(*h DIFFERENT DE 0) ALORS`

`equilibre(a) ← equilibre(a) + *h`

➔ SI `(equilibre(a) EST EGAL A 0) ALORS`

`*h ← -1`

SINON

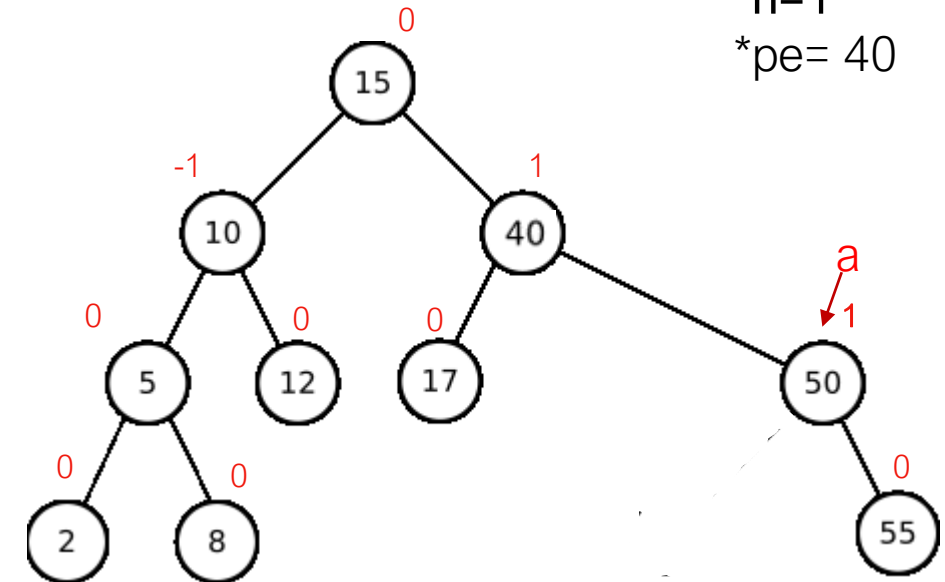
`*h ← 0`

FIN SI

FIN SI

RETOURNER a

FIN



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

SI (`fg(a)` EST EGAL A NULL) ALORS

`*pe ← element(a)`

`*h ← -1`

`tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

RETOURNER(`a`)

SINON

`fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI (`*h` DIFFERENT DE 0) ALORS

`equilibre(a) ← equilibre(a) + *h`

SI (`equilibre(a)` EST EGAL A 0) ALORS

`*h ← -1`

SINON

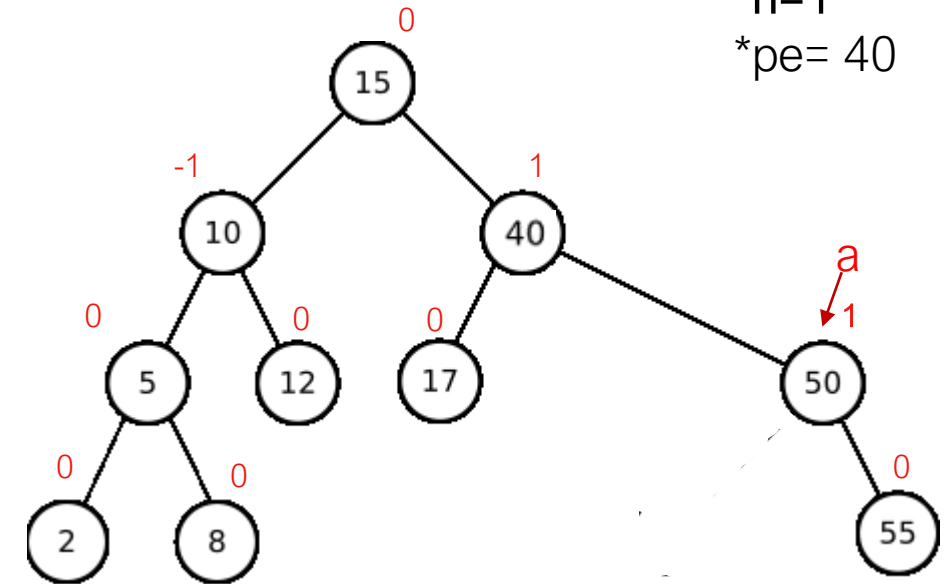
→ `*h ← 0`

FIN SI

FIN SI

RETOURNER `a`

FIN



# Suppression

suppressionAVL(A, 20, h)

FONCTION `suppMinAVL(a: Arbre; h: pointeur sur entier, pe: ptr sur Element) : ptr sur Arbre`

VARIABLE

`tmp : ptr sur Arbre`

DEBUT

SI (`fg(a)` EST EGAL A NULL) ALORS

`*pe ← element(a)`

`*h ← -1`

`tmp ← a`

`a ← fd(a)`

`liberer(tmp)`

RETOURNER(`a`)

SINON

`fg(a) ← suppMinAVL(fg(a), h, pe)`

`*h ← -*h`

FIN SI

SI (`*h` DIFFERENT DE 0) ALORS

`equilibre(a) ← equilibre(a) + *h`

SI (`equilibre(a)` EST EGAL A 0) ALORS

`*h ← -1`

SINON

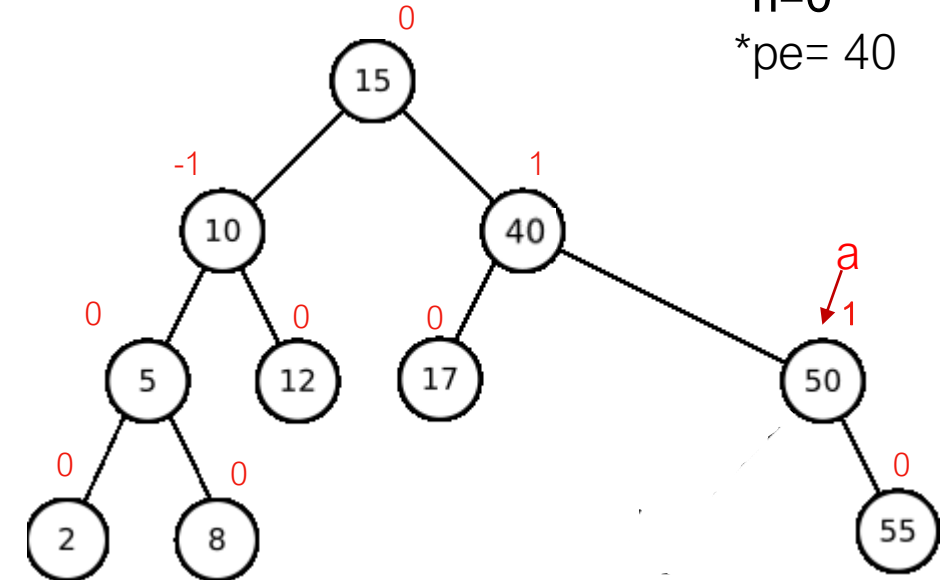
`*h ← 0`

FIN SI

FIN SI

➔ RETOURNER `a`

FIN



# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

SINON SI (e SUP. STRICT. A element(a)) ALORS

➔ fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

➔ fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

\*h ← 1

FIN SI

FIN SI

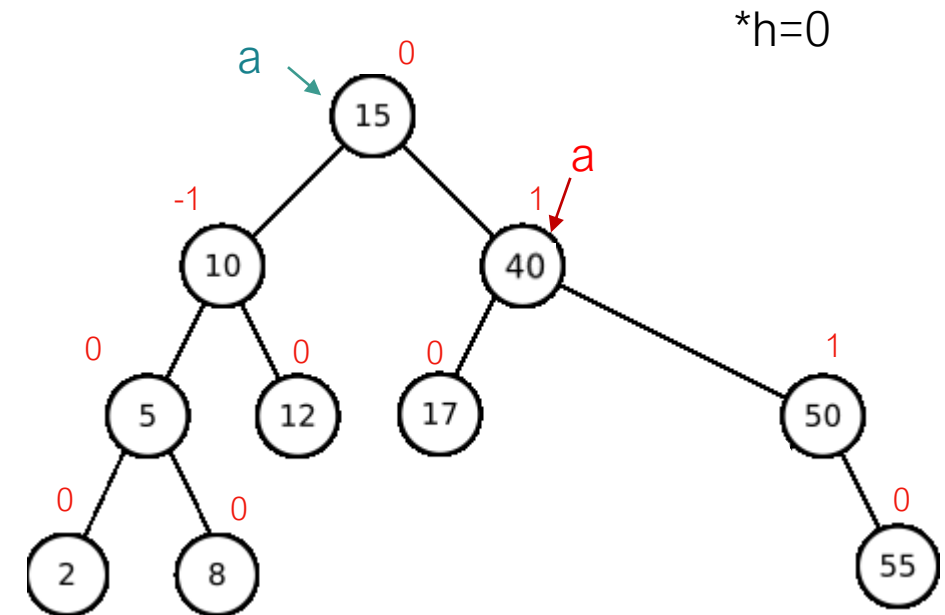
RETOURNER a

FIN

SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI



# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

SINON SI (e SUP. STRICT. A element(a)) ALORS

➡ fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

➡ SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

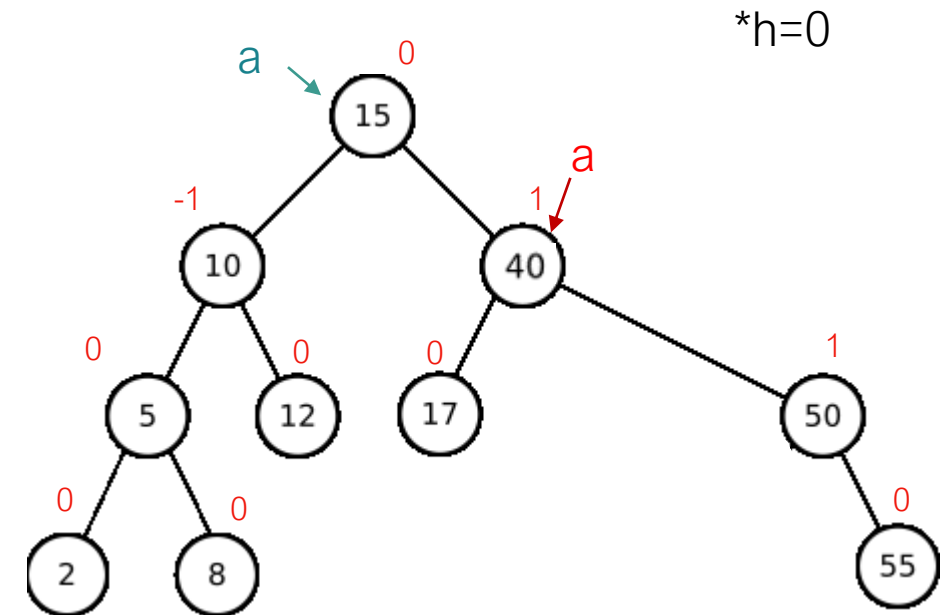
\*h ← 1

FIN SI

FIN SI

RETOURNER a

FIN



SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI



# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

SINON SI (e SUP. STRICT. A element(a)) ALORS

➔ fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

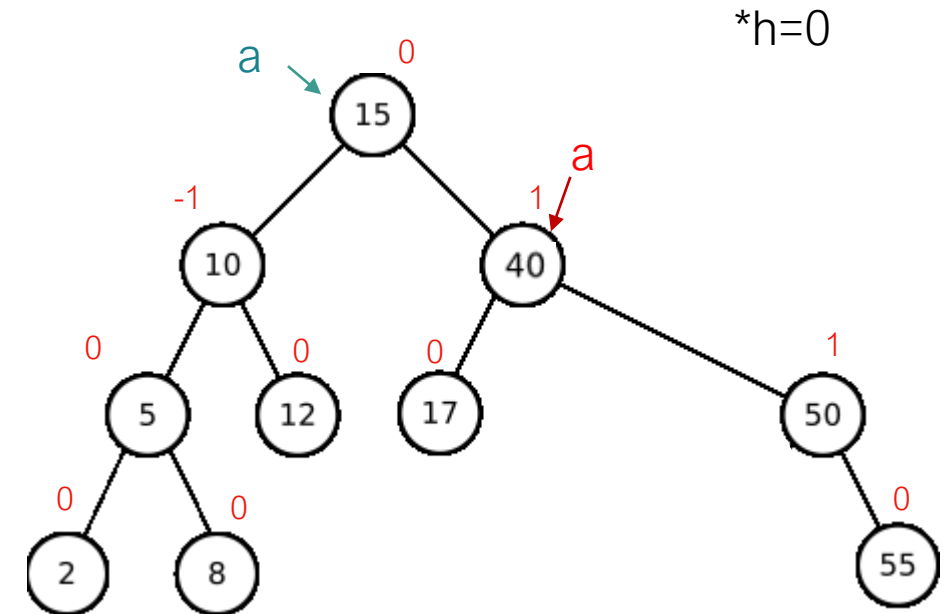
\*h ← 1

FIN SI

FIN SI

➔ RETOURNER a

FIN



SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI

# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

SINON SI (e SUP. STRICT. A element(a)) ALORS

➔ fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

\*h ← 1

FIN SI

FIN SI

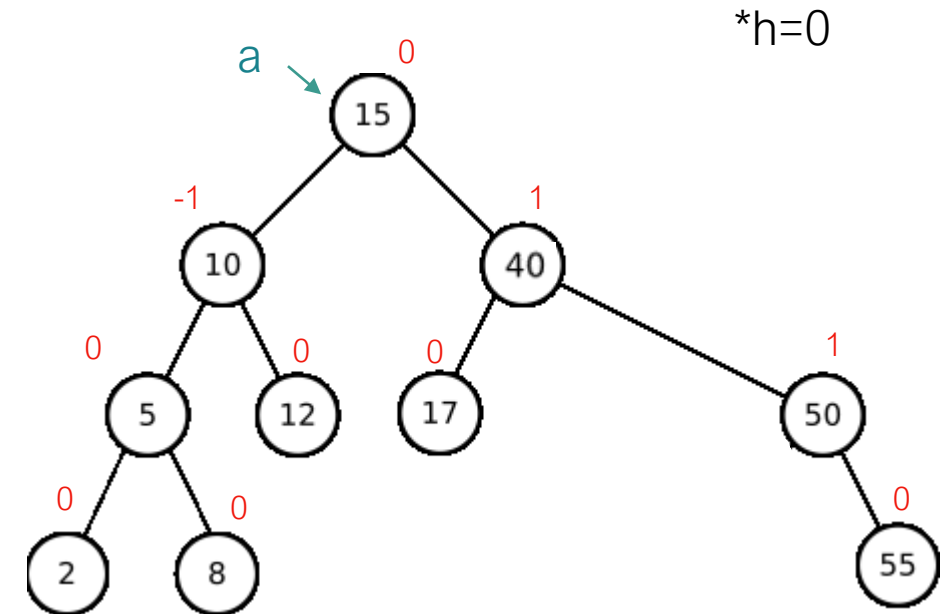
RETOURNER a

FIN

SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI



# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

SINON SI (e SUP. STRICT. A element(a)) ALORS

fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

→ SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

\*h ← 1

FIN SI

FIN SI

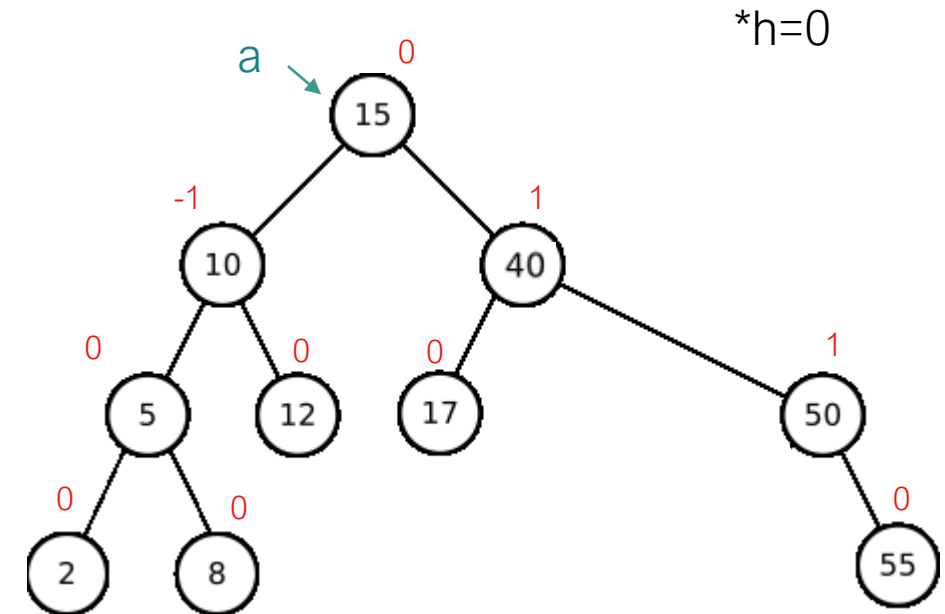
RETOURNER a

FIN

SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI



# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

SINON SI (e SUP. STRICT. A element(a)) ALORS

fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

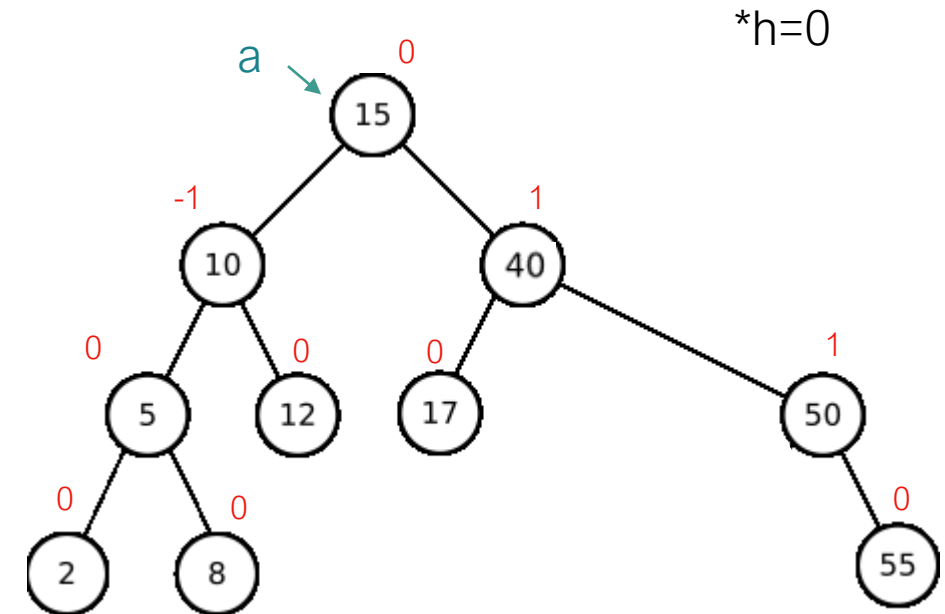
\*h ← 1

FIN SI

FIN SI

➔ RETOURNER a

FIN



SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI

# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

SINON SI (e SUP. STRICT. A element(a)) ALORS

fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

\*h ← 1

FIN SI

FIN SI

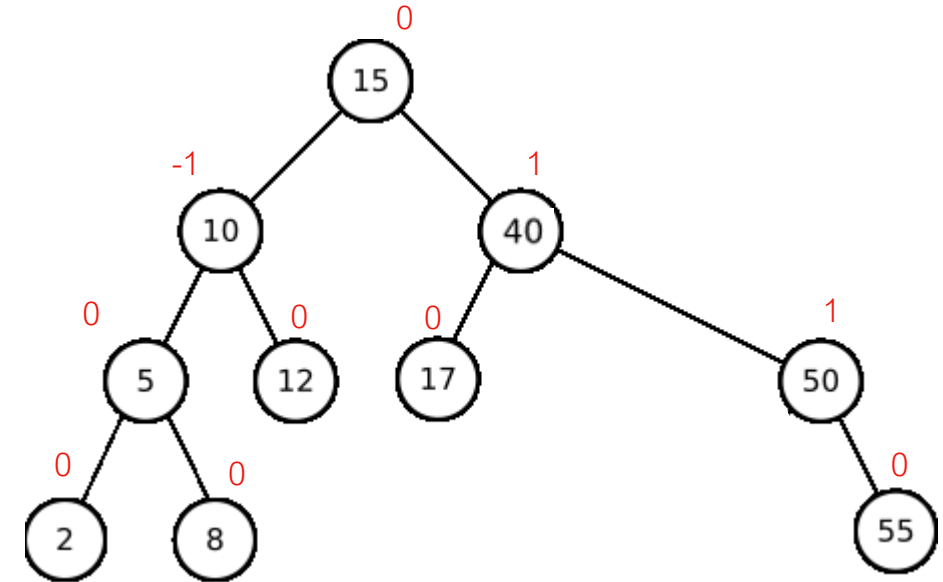
RETOURNER a

FIN

SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI



# Suppression

suppressionAVL(a, 20, h)

FONCTION suppressionAVL(a: ptr sur Arbre, e: element, h: ptr sur Element) : pointeur sur Arbre  
VARIABLE

tmp : ptr sur Arbre

DEBUT

SI (a EST EGAL A NULL) ALORS

\*h ← 1

RETOURNER A

SINON SI (e SUP. STRICT. A element(a)) ALORS

fd(a) ← suppressionAVL(fd(a), e)

SINON SI (e INF. STRICT. A element(a)) ALORS

fg(a) ← suppressionAVL(fg(a), e)

\*h ← -\*h

SINON SI (existeFilsDroit(a)) ALORS

fd(a) ← suppMinAVL( fd(a), h, adresse(element(a)) )

SINON

tmp ← a

a ← fg(a)

libérer(tmp)

\*h ← -1

FIN SI

SI (\*h DIFFERENT DE 0) ALORS

equilibre(a) ← equilibre (a) + \*h

SI (equilibre(a) EST EGAL A 0) ALORS

\*h ← 0

SINON

\*h ← 1

FIN SI

FIN SI

RETOURNER a

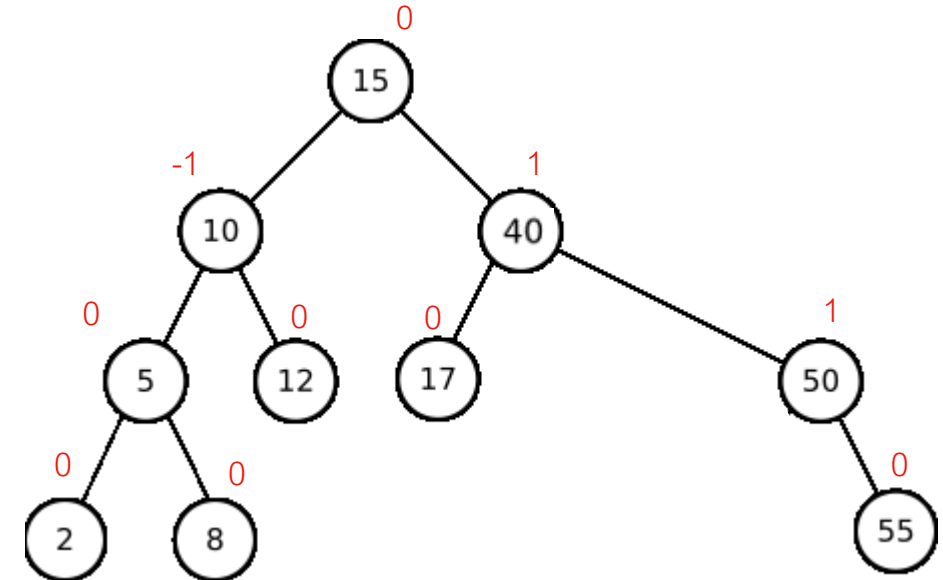
FIN

SI (a EST EGAL A NULL) ALORS

\*h ← 0

FIN SI

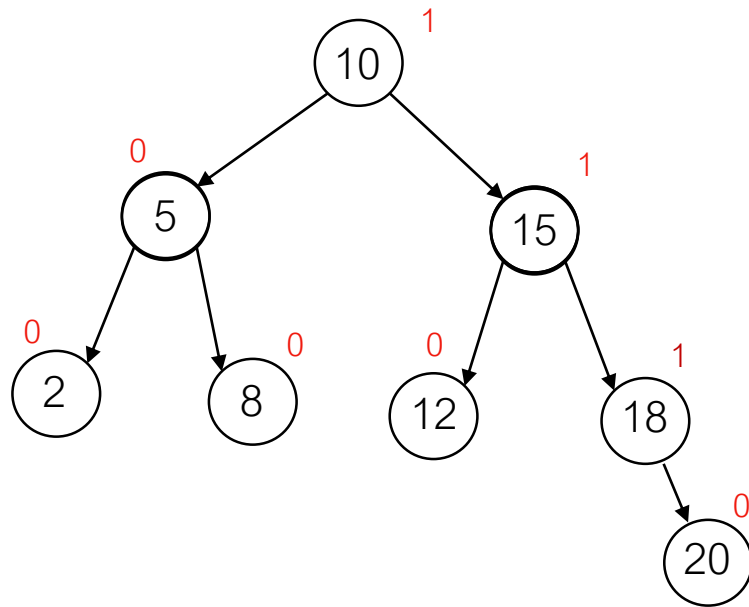
**a ← equilibrageAVL(a)**



## IV. Equilibrage d'un AVL

# Rotation simple

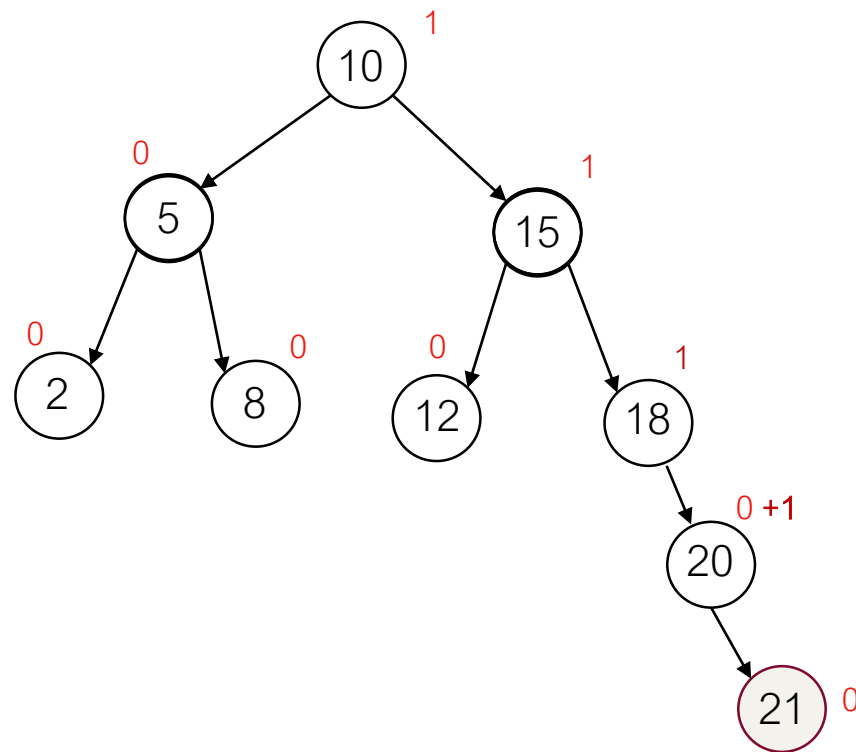
- L'opération de rééquilibrage dépend de la structure de l'arbre après une opération d'insertion/suppression.
- Cas 1 : L'élément ajouté est tout à droite de l'arbre.





# Rotation simple

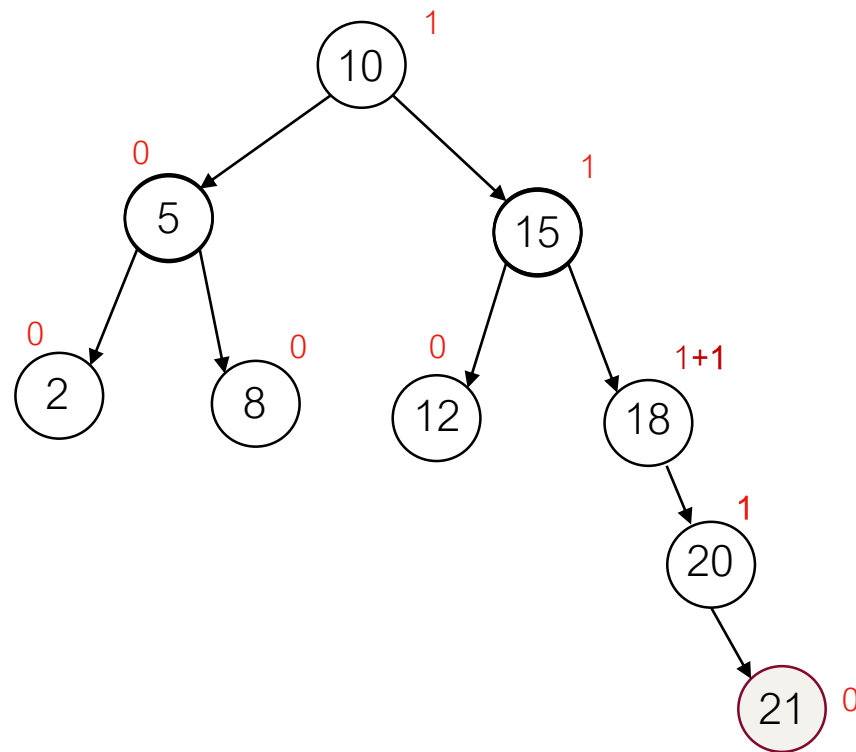
- Cas 1 : L'élément ajouté est tout à droite de l'arbre.



La modification de l'équilibre des nœuds se fait en remontant les parents

# Rotation simple

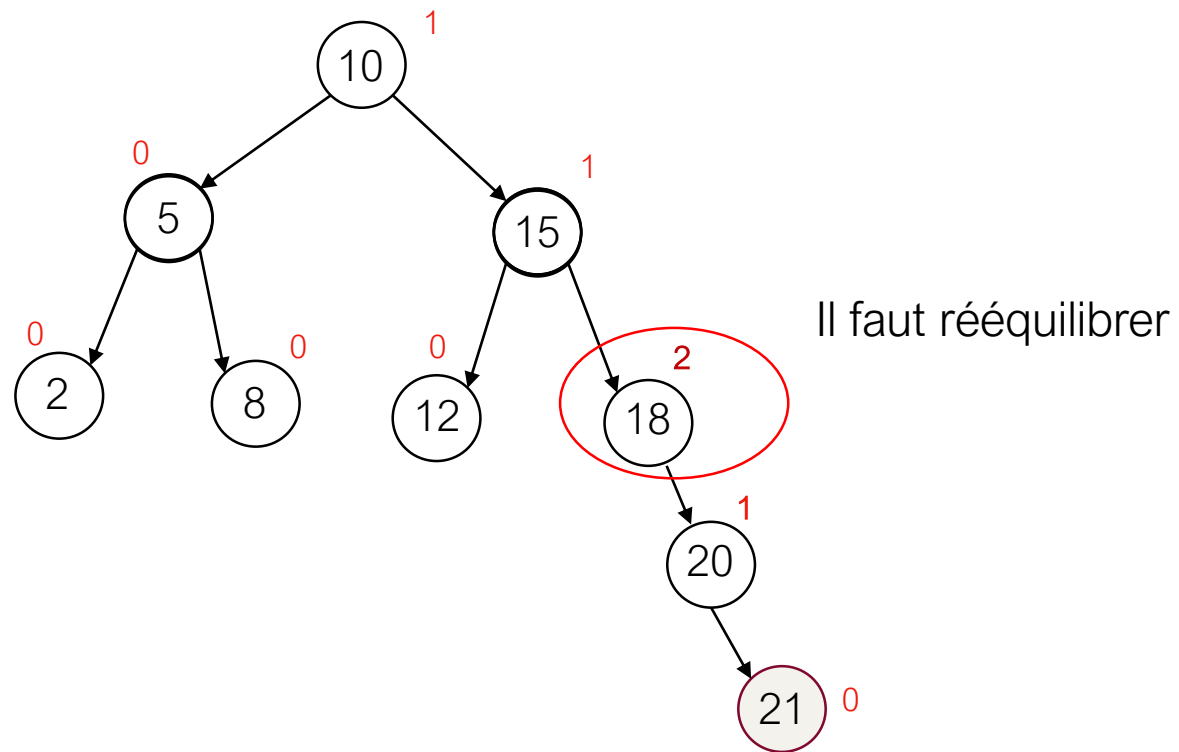
- Cas 1 : L'élément ajouté est tout à droite de l'arbre.



La modification de l'équilibre des nœuds se fait en remontant les parents

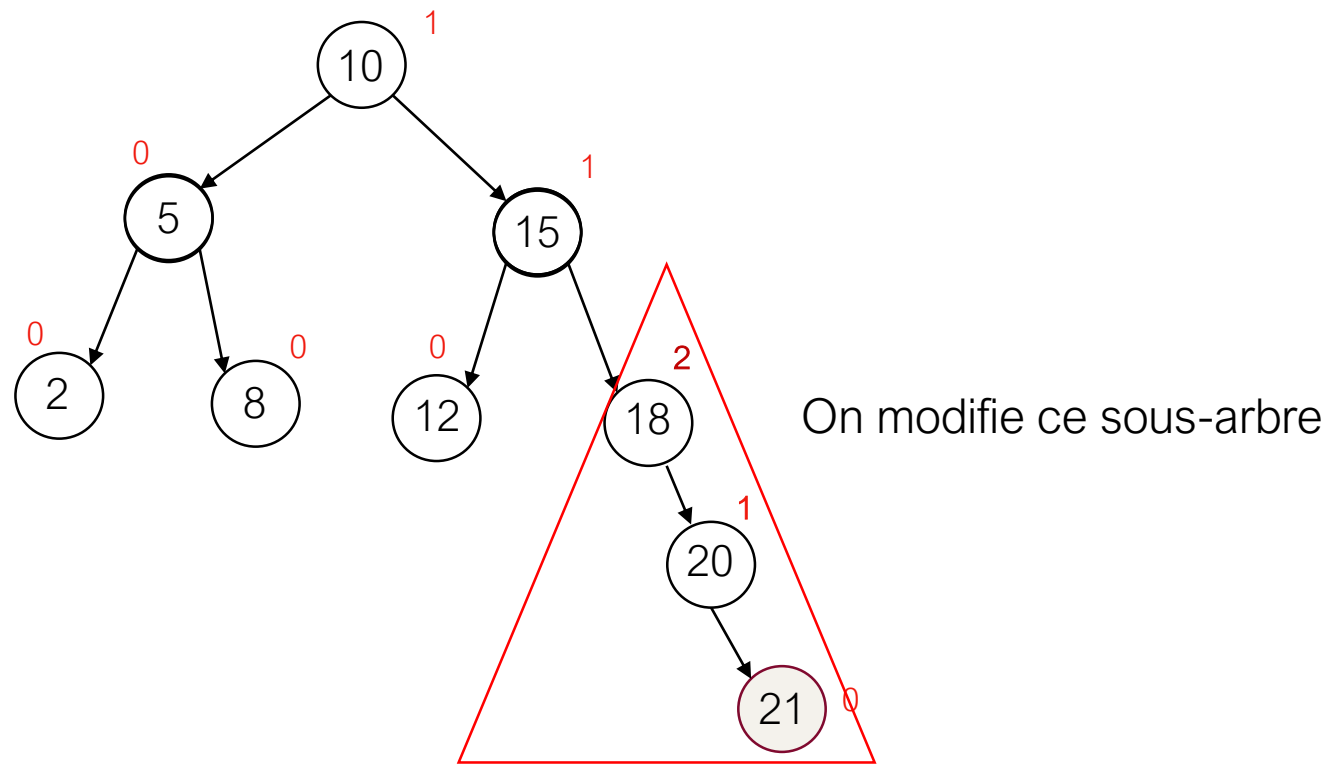
# Rotation simple

- Cas 1 : L'élément ajouté est tout à droite de l'arbre.



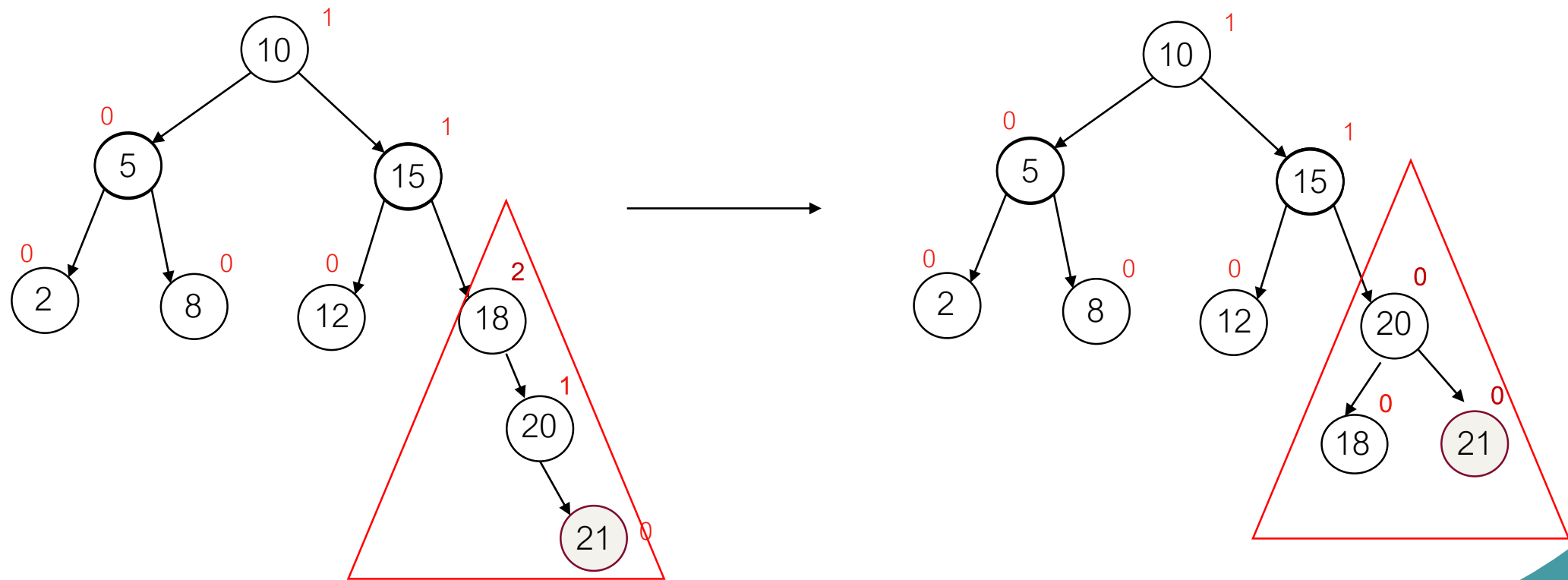
# Rotation simple

- Cas 1 : L'élément ajouté est tout à droite de l'arbre.



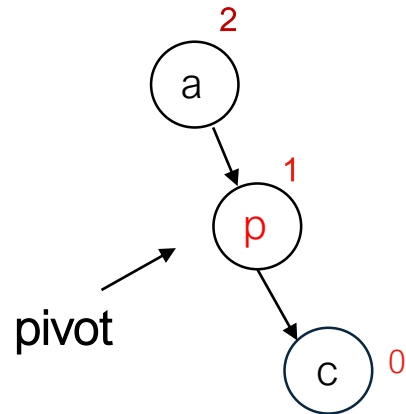
# Rotation simple

- Cas 1 : L'élément ajouté est tout à droite de l'arbre **rotation simple à gauche**



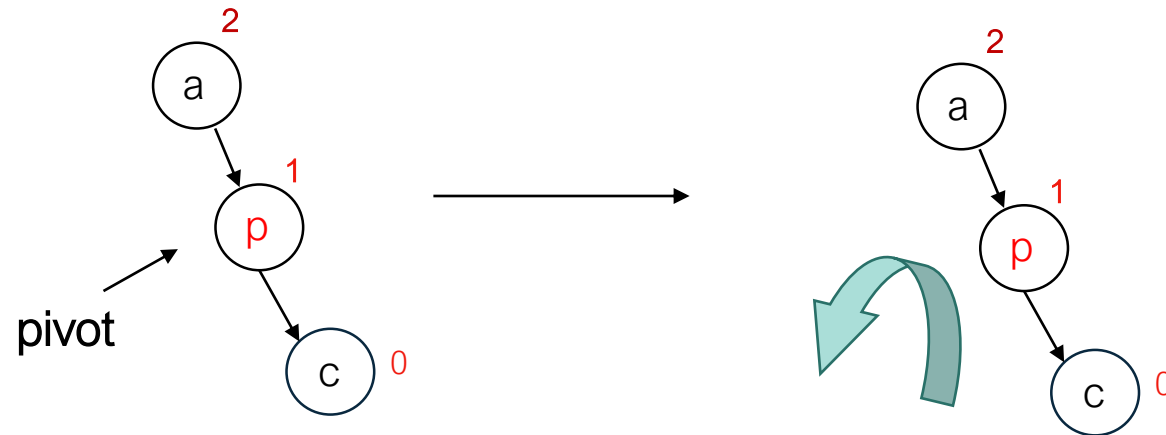
# Rotation simple

- Cas 1 : L'élément ajouté est tout à droite de l'arbre **rotation simple à gauche**
  - L'élément intermédiaire doit devenir la racine du sous-arbre équilibré ; l'arbre doit tourner autour de cet élément : c'est le **pivot**.



# Rotation simple

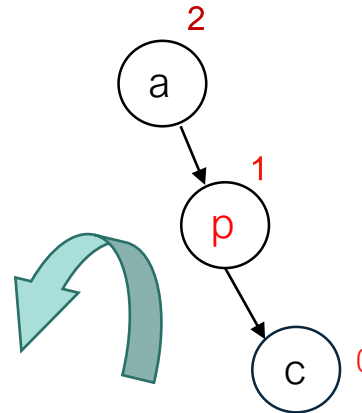
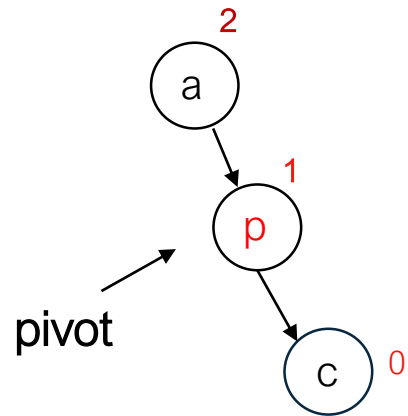
- Cas 1 : L'élément ajouté est tout à droite de l'arbre **rotation simple à gauche**
  - L'élément intermédiaire doit devenir la racine du sous-arbre équilibré ; l'arbre doit tourner autour de cet élément : c'est le **pivot**.



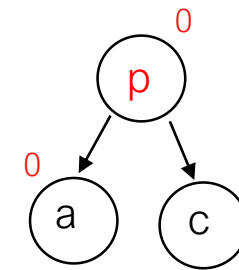
On tourne le sous arbre vers la gauche autour du **pivot**.

# Rotation simple

- Cas 1 : L'élément ajouté est tout à droite de l'arbre **rotation simple à gauche**
  - L'élément intermédiaire doit devenir la racine du sous-arbre équilibré ; l'arbre doit tourner autour de cet élément : c'est le **pivot**.



On tourne le sous arbre vers la gauche autour du **pivot**.

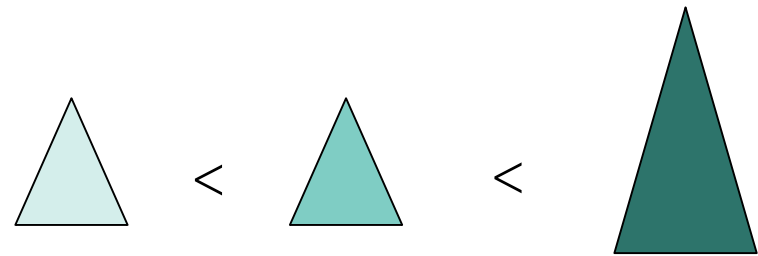
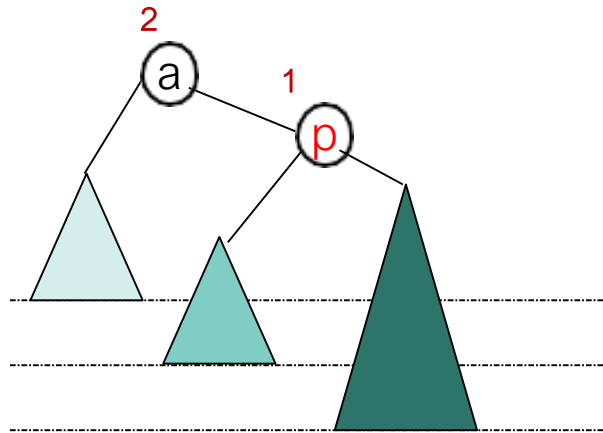


Arbre équilibré



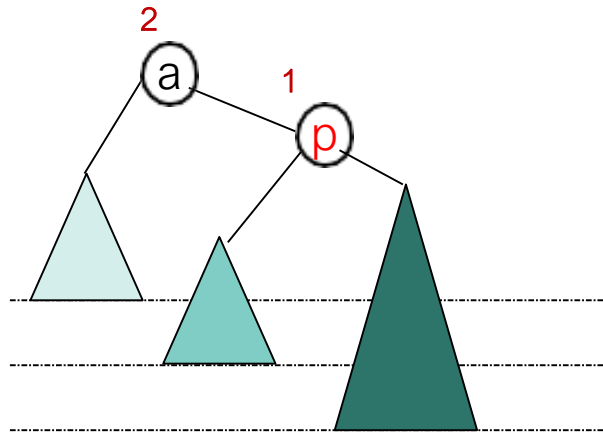
# Rotation simple

- Cas 1 : L'élément ajouté est tout à droite de l'arbre **rotation simple à gauche**
  - Schema general de la rotation gauche :

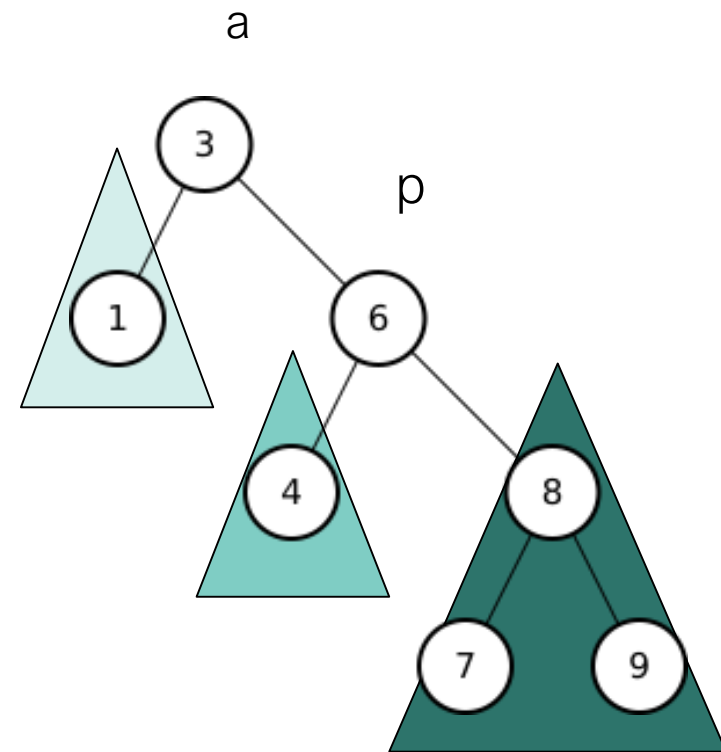


# Rotation simple

- Cas 1 : L'élément ajouté est tout à droite de l'arbre **rotation simple à gauche**
  - Schema general de la rotation gauche :

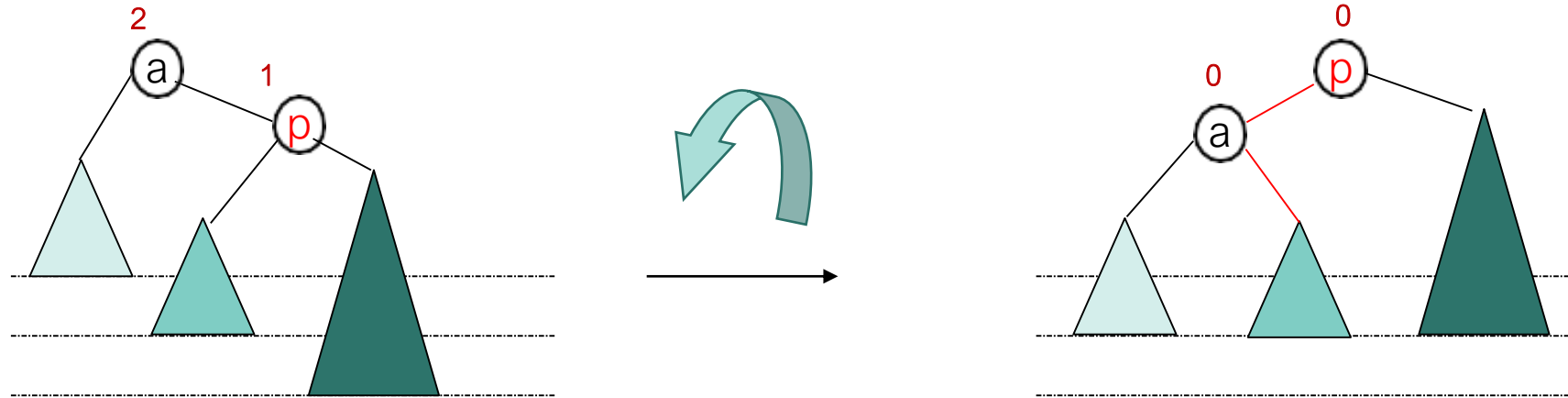


Exemple :



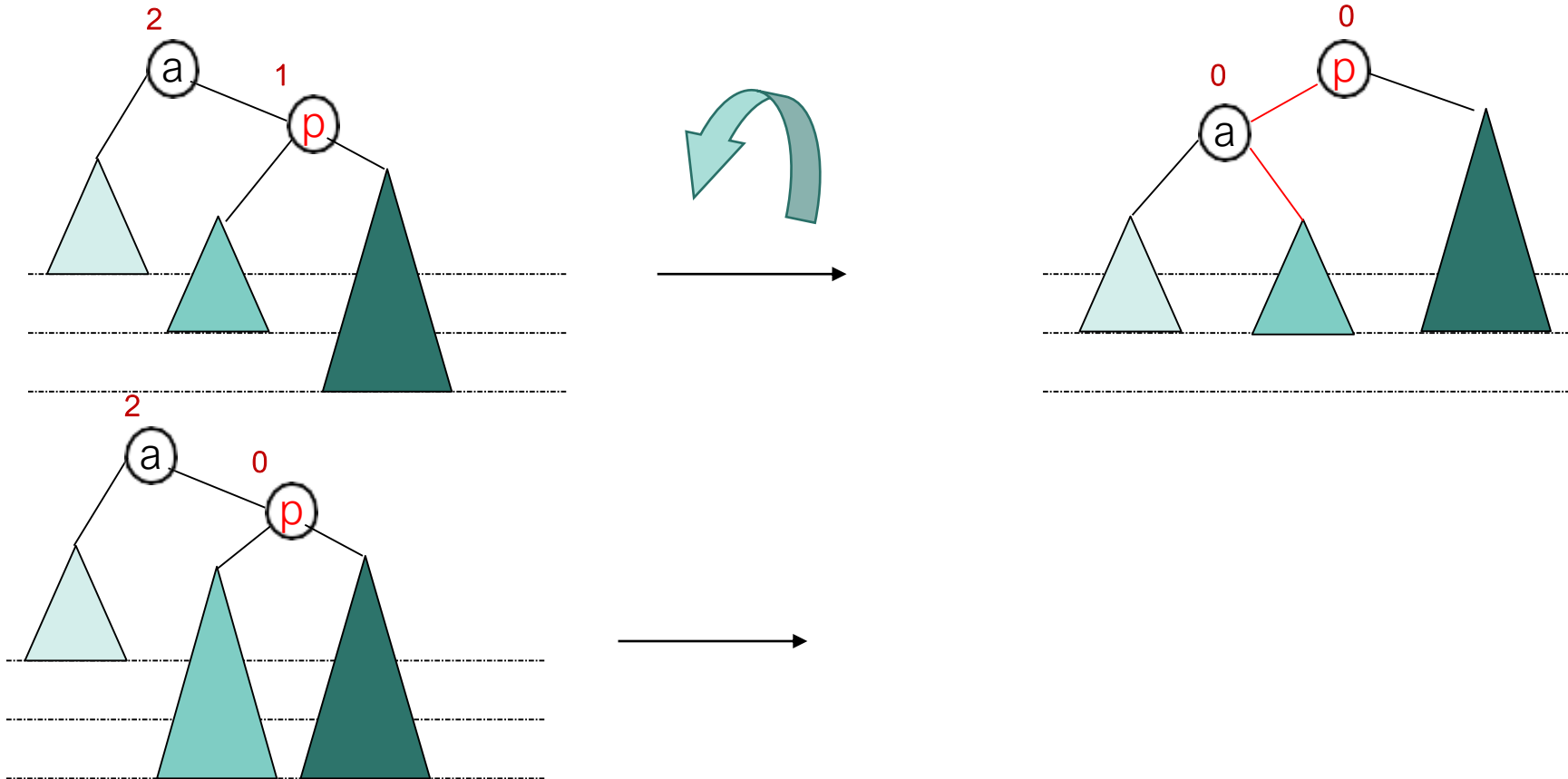
# Rotation simple

- Cas 1 : L'élément ajouté est tout à droite de l'arbre **rotation simple à gauche**
  - Schema general de la rotation gauche :



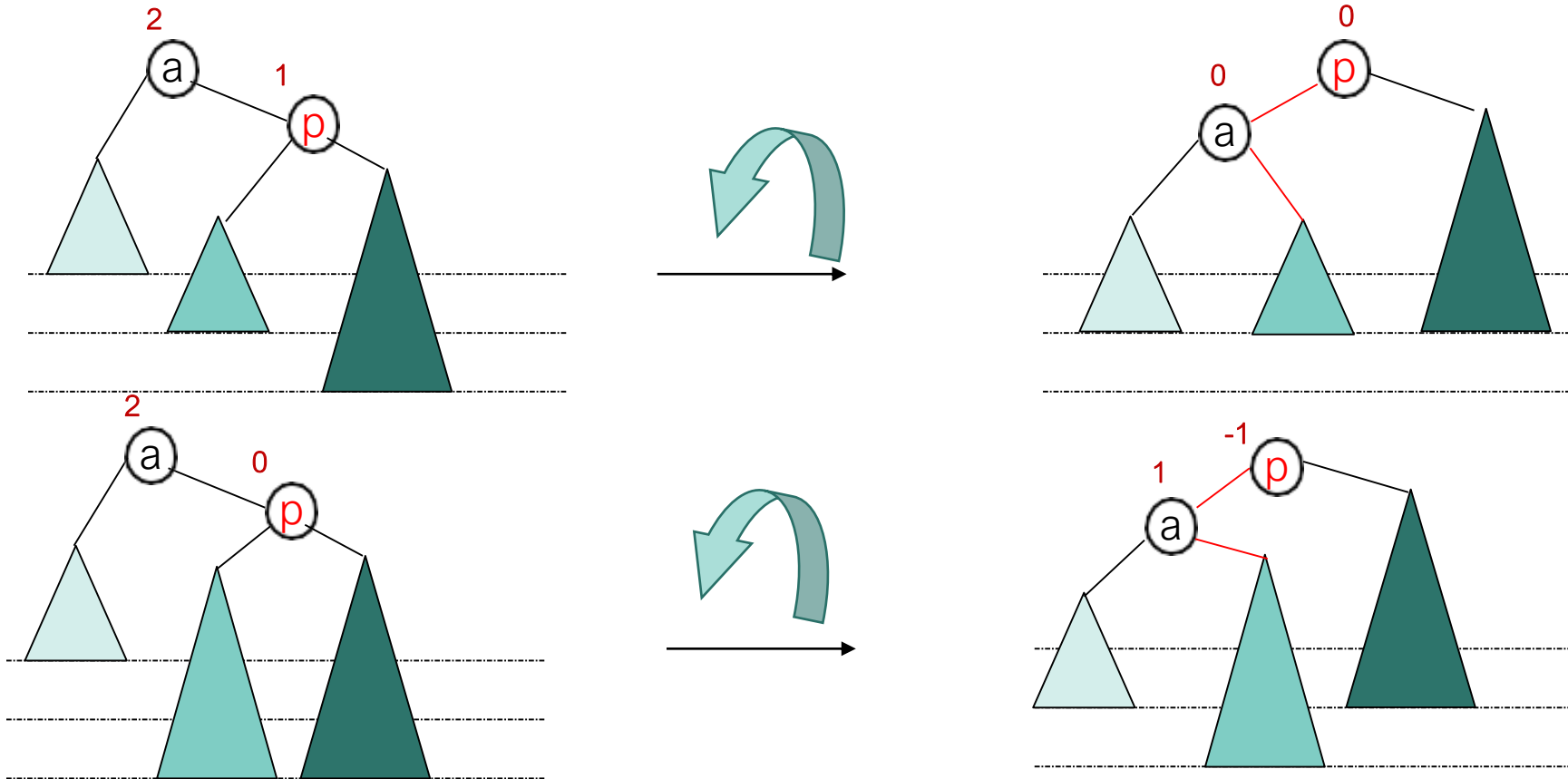
# Rotation simple

- Cas 1 : L'élément ajouté est tout à droite de l'arbre **rotation simple à gauche**
  - Schema general de la rotation gauche :



# Rotation simple

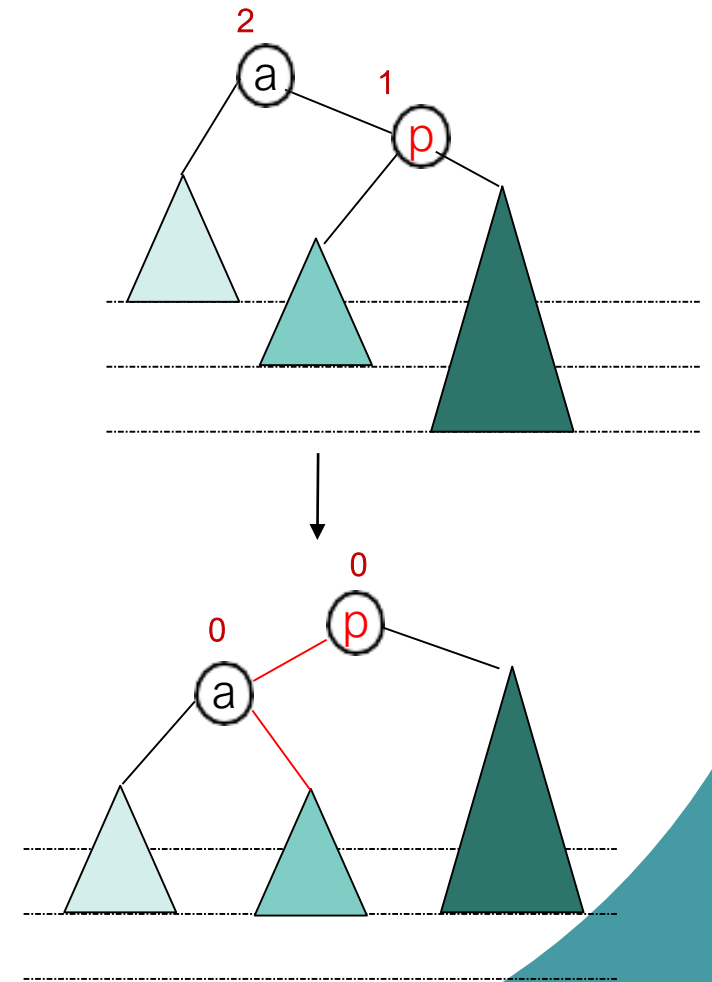
- Cas 1 : L'élément ajouté est tout à droite de l'arbre **rotation simple à gauche**
  - Schema general de la rotation gauche :



# Rotation simple

- Cas 1 : L'élément ajouté est tout à droite de l'arbre **rotation simple à gauche**
  - Algorithme:

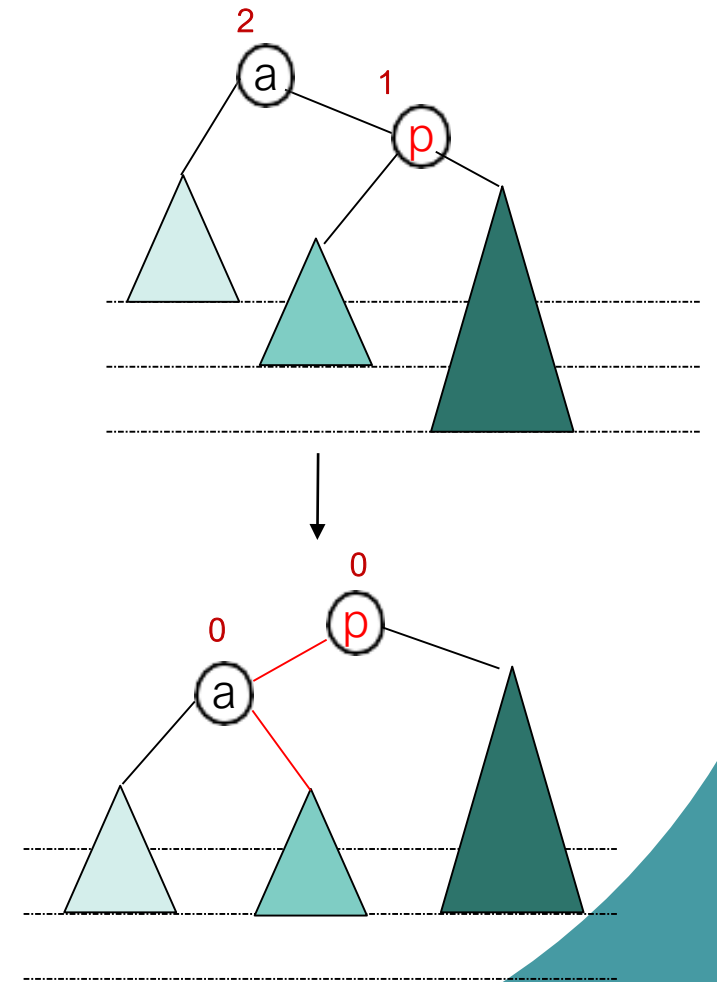
```
FONCTION rotationGauche(a: ptr sur Arbre) : ptr sur Arbre
VARIABLE
    pivot : ptr sur Arbre
    eq_a, eq_p : entier
DEBUT
    pivot ← ???
    ...
    RETOURNER a
FIN
```



# Rotation simple

- Cas 1 : L'élément ajouté est tout à droite de l'arbre **rotation simple à gauche**
  - Algorithme:

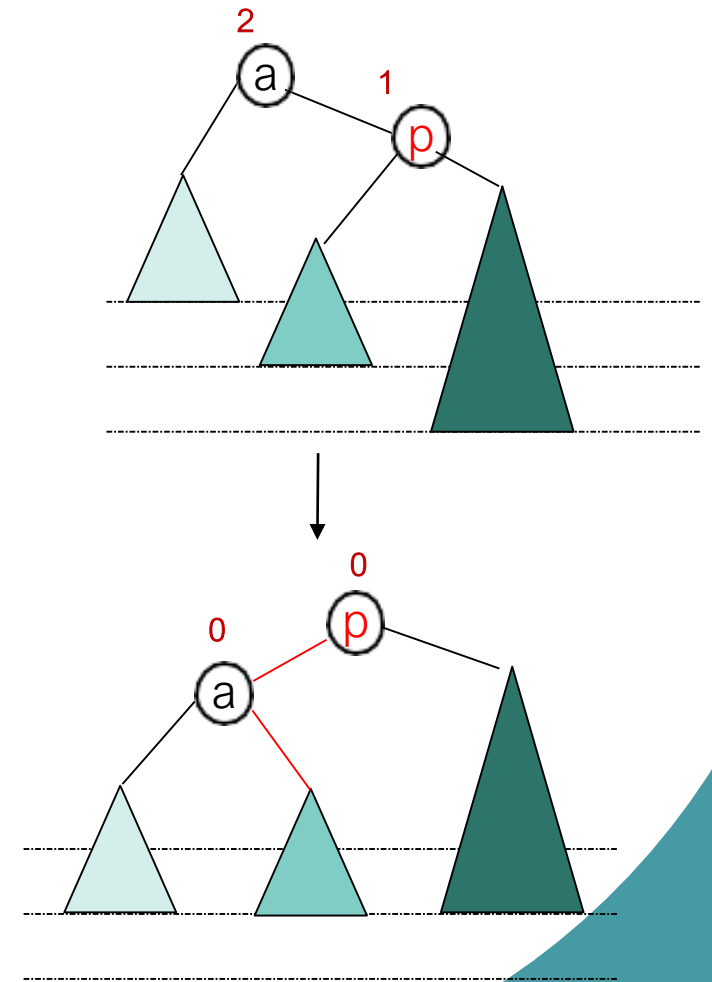
```
FONCTION rotationGauche(a: ptr sur Arbre) : ptr sur Arbre
VARIABLE
    pivot : ptr sur Arbre
    eq_a, eq_p : entier
DEBUT
    pivot ← fd(a)
    fd(a) ← fg(pivot)
    fg(pivot) ← a
    ...
    a ← pivot
    RETOURNER a
FIN
```



# Rotation simple

- Cas 1 : L'élément ajouté est tout à droite de l'arbre **rotation simple à gauche**
  - Algorithme:

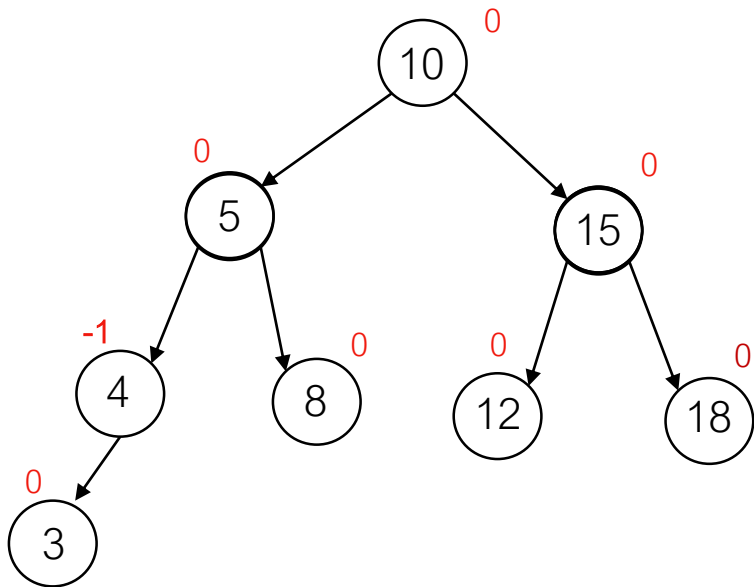
```
FONCTION rotationGauche(a: ptr sur Arbre) : ptr sur Arbre
VARIABLE
    pivot : ptr sur Arbre
    eq_a, eq_p : entier
DEBUT
    pivot ← fd(a)
    fd(a) ← fg(pivot)
    fg(pivot) ← a
    eq_a ← equilibre(a)
    eq_p ← equilibre(pivot)
    equilibre(a) ← eq_a - max(eq_p, 0) - 1
    equilibre(pivot) ← min( eq_a-2, eq_a+eq_p-2, eq_p-1 )
    a ← pivot
    RETOURNER a
FIN
```





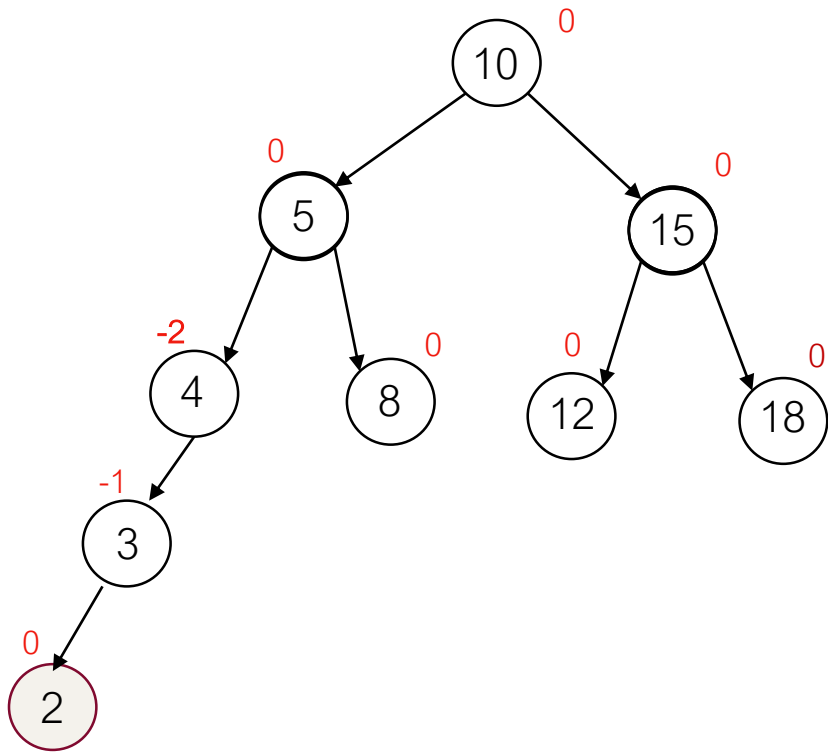
# Rotation simple

- Cas 2 : L'élément ajouté est tout à gauche de l'arbre.



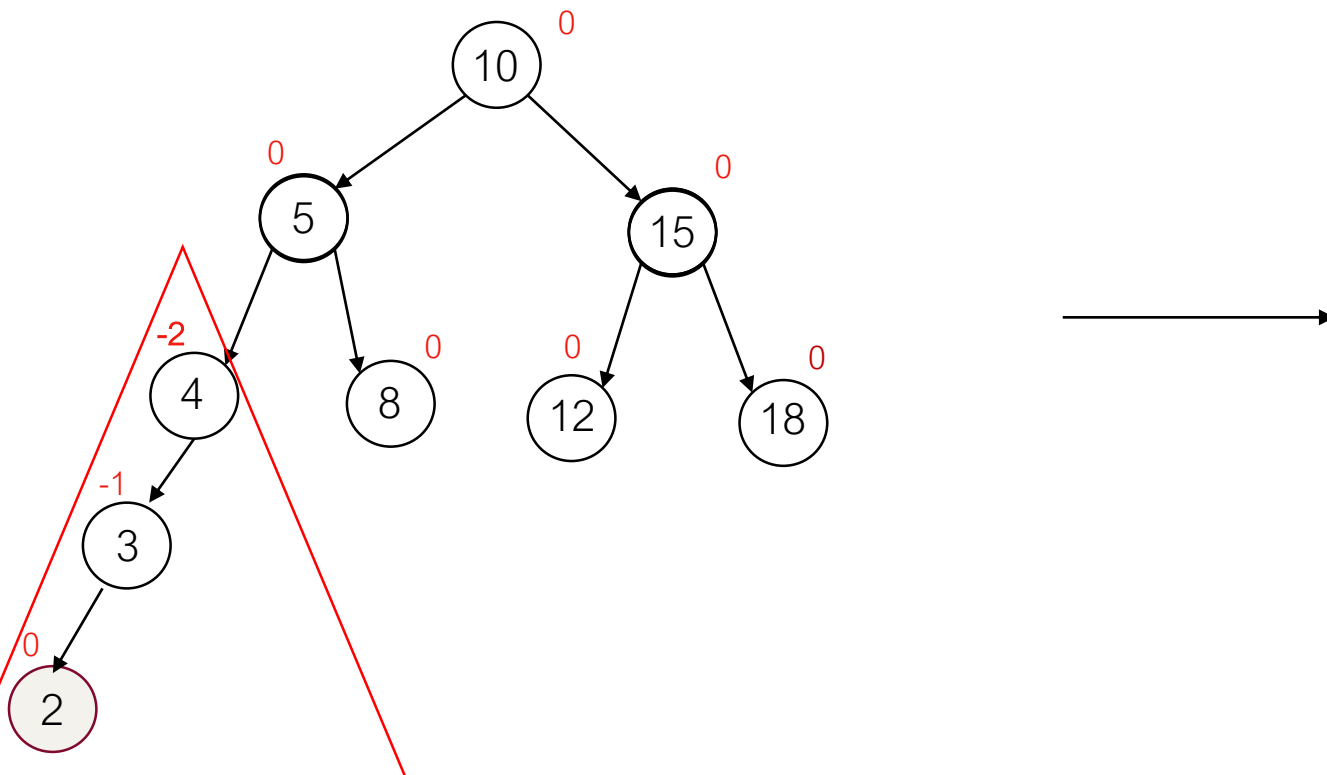
# Rotation simple

- Cas 2 : L'élément ajouté est tout à gauche de l'arbre.



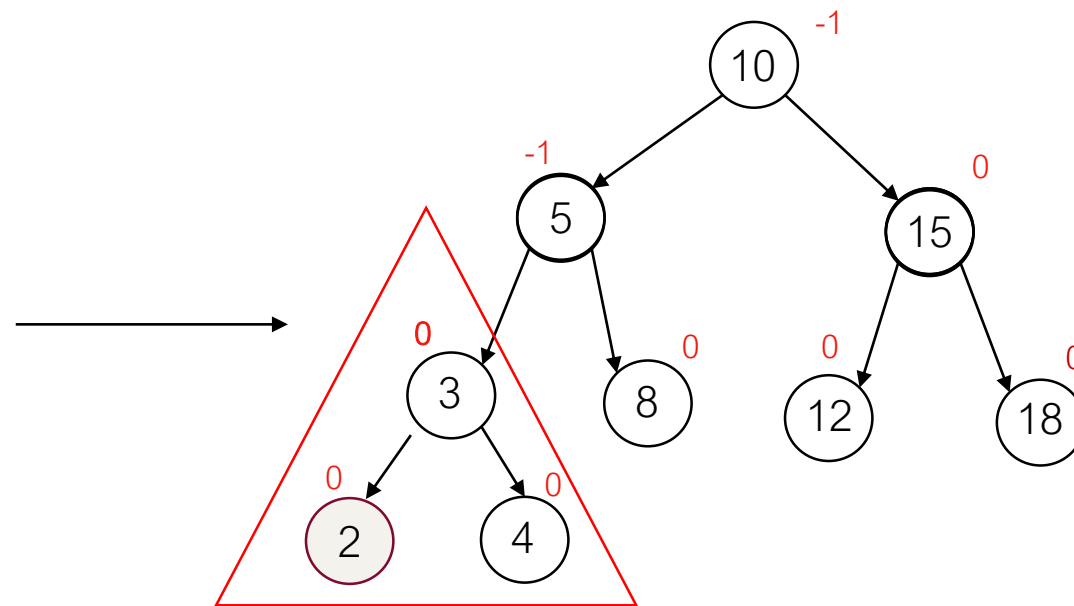
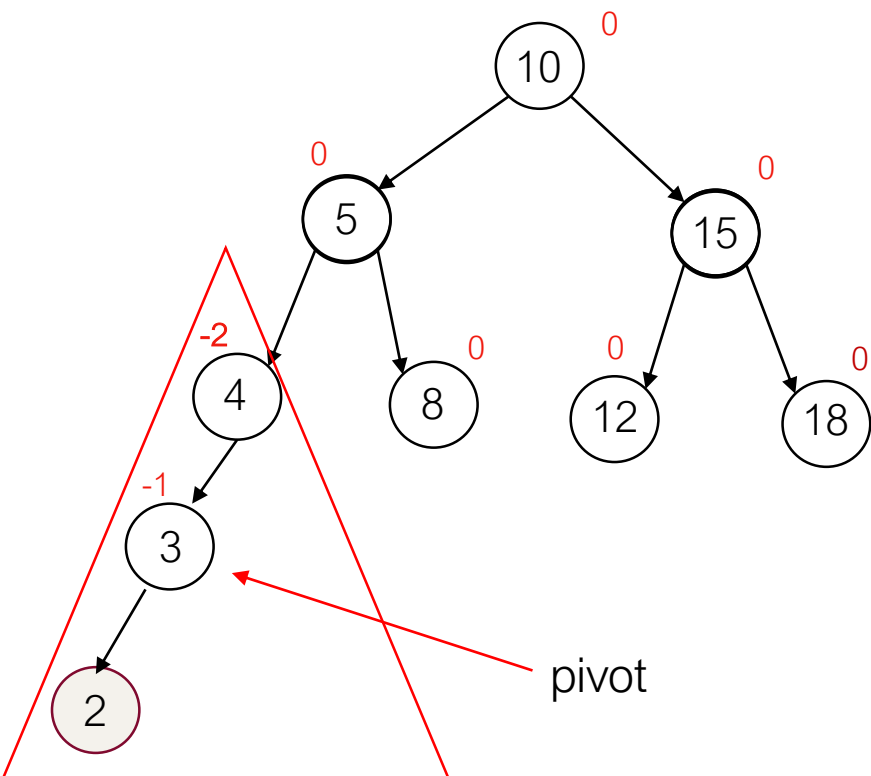
# Rotation simple

- Cas 2 : L'élément ajouté est tout à gauche de l'arbre.



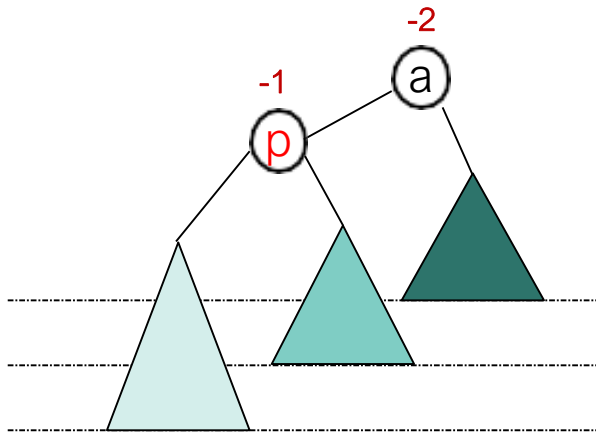
# Rotation simple

- Cas 2 : L'élément ajouté est tout à gauche de l'arbre : **rotation simple à droite**



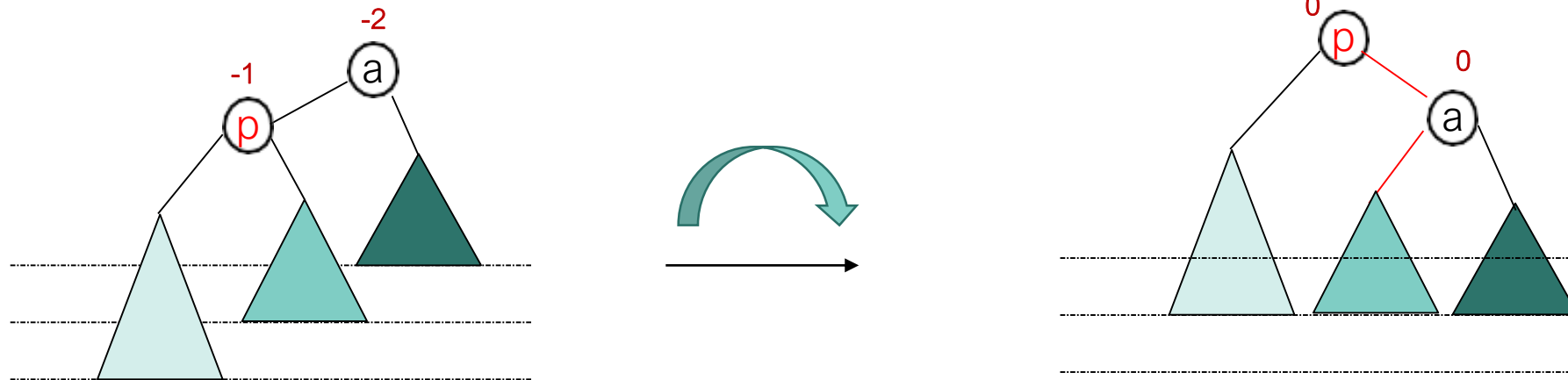
# Rotation simple

- Cas 2 : L'élément ajouté est tout à gauche de l'arbre : **rotation simple à droite**
  - Schema general rotation droite :



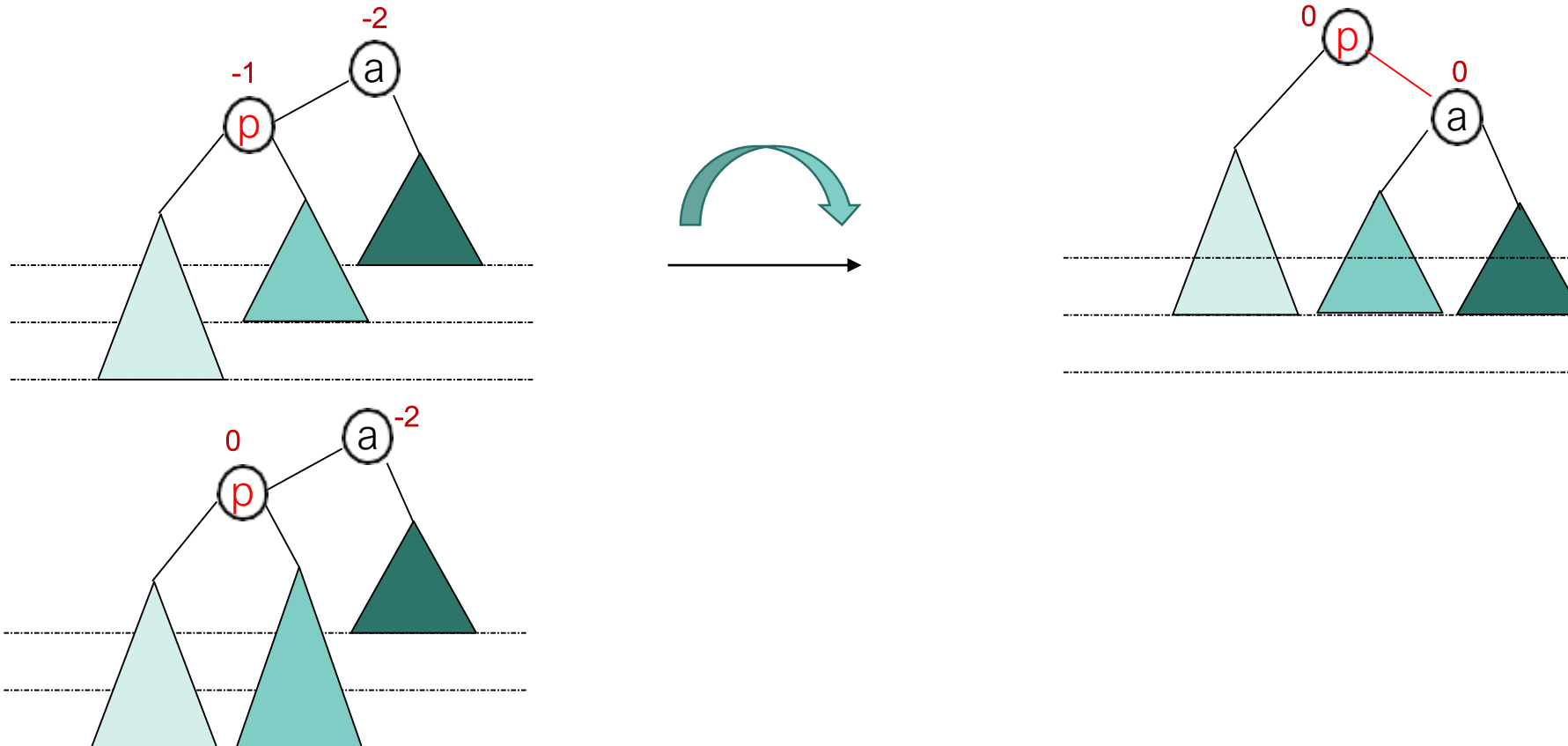
# Rotation simple

- Cas 2 : L'élément ajouté est tout à gauche de l'arbre : **rotation simple à droite**
  - Schema general rotation droite :



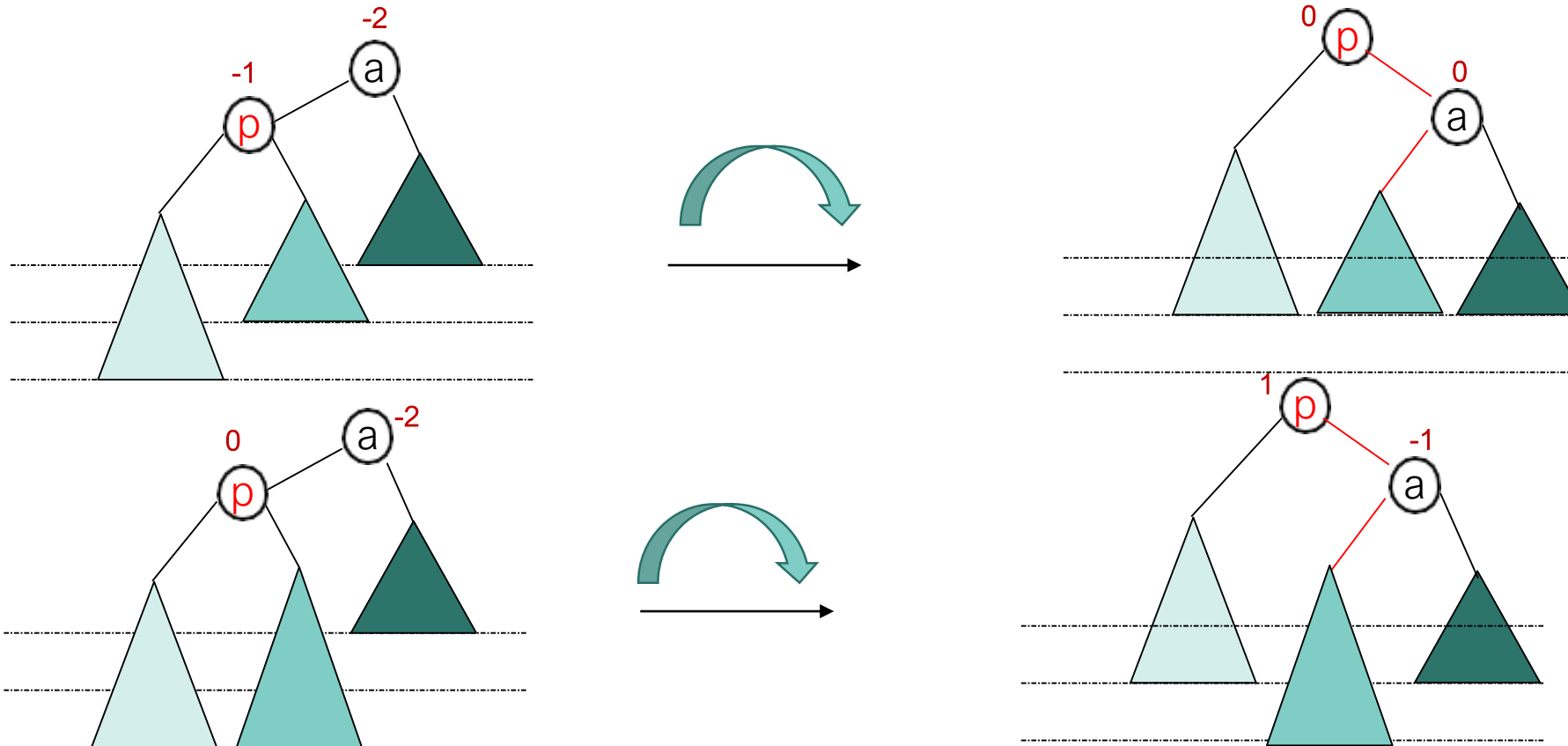
# Rotation simple

- Cas 2 : L'élément ajouté est tout à gauche de l'arbre : **rotation simple à droite**
  - Schema general rotation droite :



# Rotation simple

- Cas 2 : L'élément ajouté est tout à gauche de l'arbre : **rotation simple à droite**
  - Schema general rotation droite :

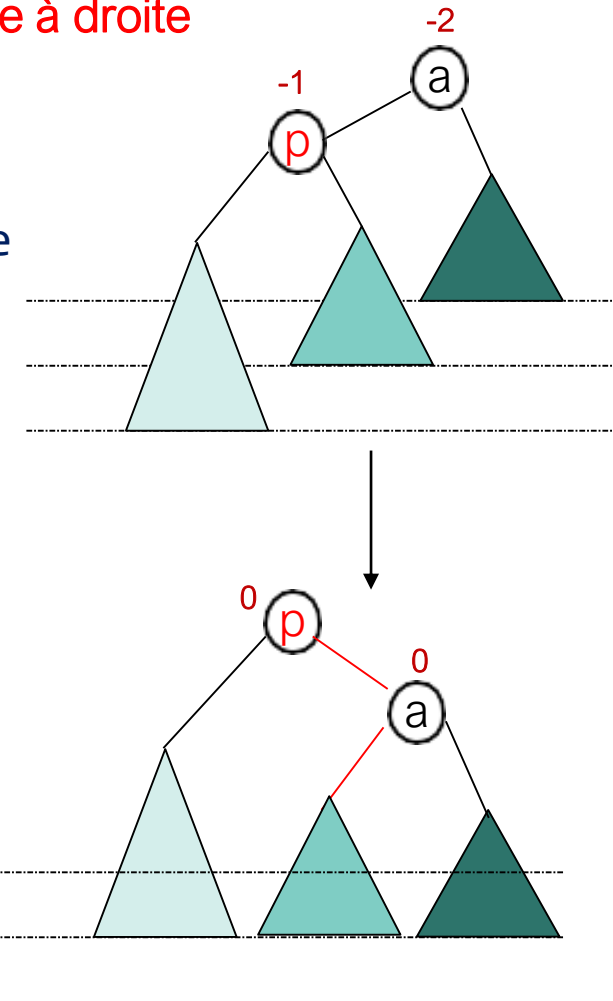




# Rotation simple

- Cas 2 : L'élément ajouté est tout à gauche de l'arbre : **rotation simple à droite**
  - Algorithme:

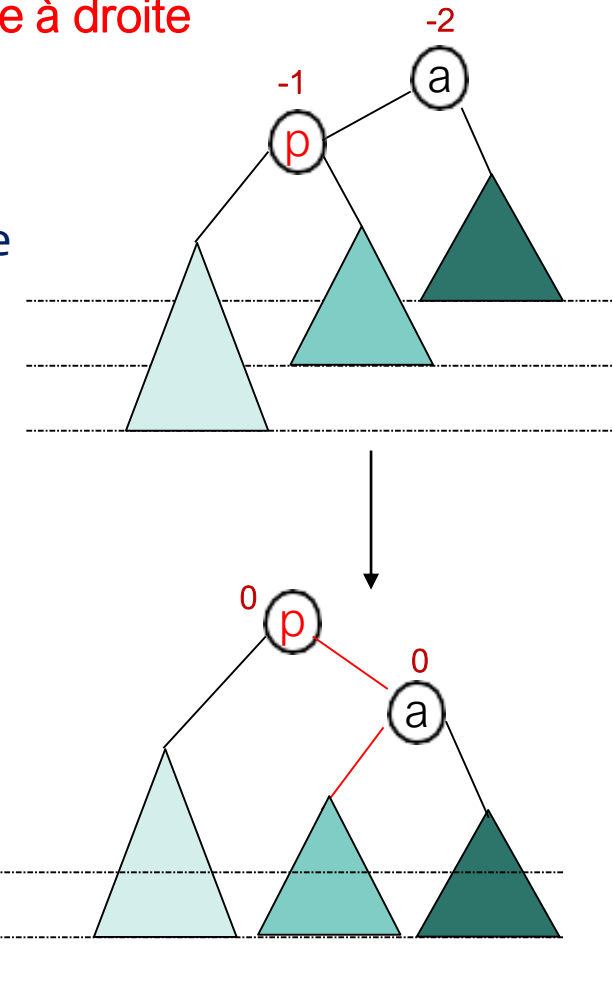
```
FONCTION rotationDroite(a: ptr sur Arbre) : ptr sur Arbre
VARIABLE
    pivot : ptr sur Arbre
    eq_a, eq_p : entier
DEBUT
    pivot ← ???
    ...
    RETOURNER a
FIN
```



# Rotation simple

- Cas 2 : L'élément ajouté est tout à gauche de l'arbre : **rotation simple à droite**
  - Algorithme:

```
FONCTION rotationDroite(a: ptr sur Arbre) : ptr sur Arbre
VARIABLE
    pivot : ptr sur Arbre
    eq_a, eq_p : entier
DEBUT
    pivot ← fg(a)
    fg(a) ← fd(pivot)
    fd(pivot) ← a
    ...
    a ← pivot
    RETOURNER a
FIN
```

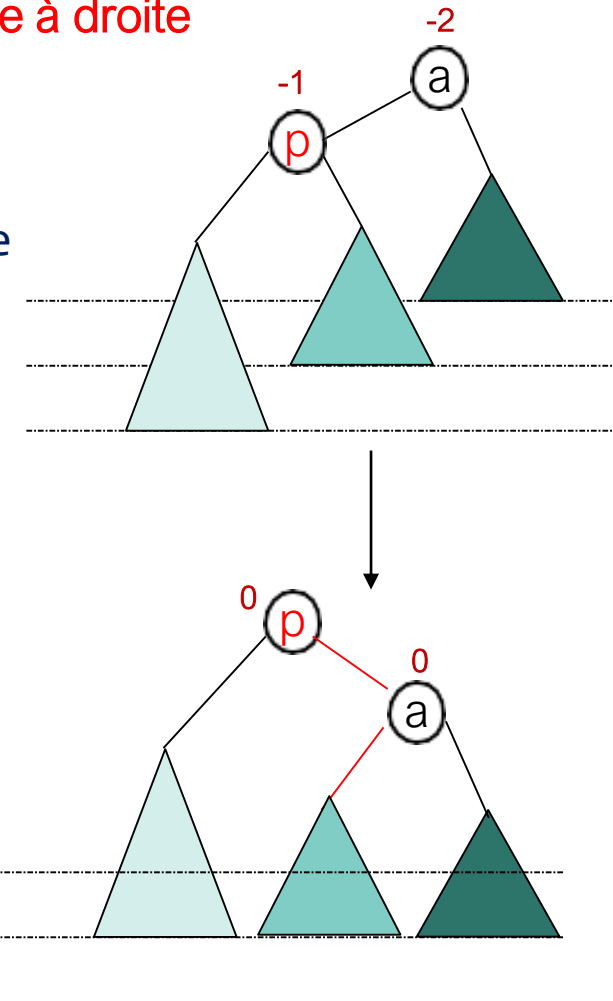


# Rotation simple

- Cas 2 : L'élément ajouté est tout à gauche de l'arbre : **rotation simple à droite**
  - Algorithme:

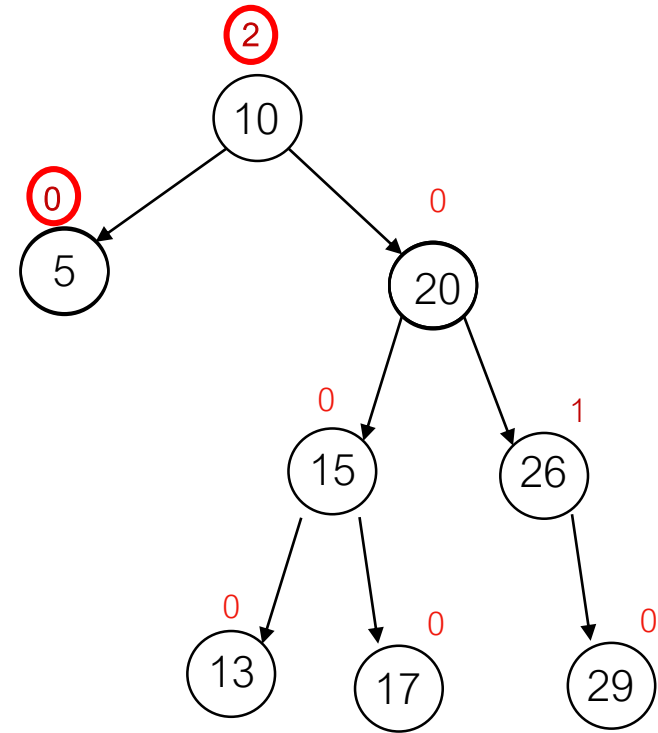
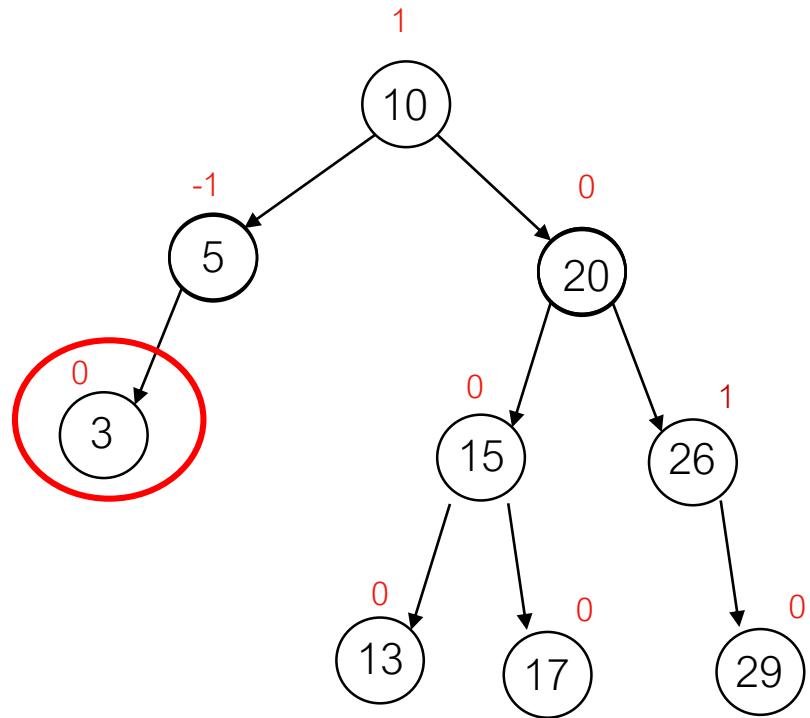
```

FONCTION rotationDroite(a: ptr sur Arbre) : ptr sur Arbre
VARIABLE
    pivot : ptr sur Arbre
    eq_a, eq_p : entier
DEBUT
    pivot ← fg(a)
    fg(a) ← fd(pivot)
    fd(pivot) ← a
    eq_a ← equilibre(a)
    eq_p ← equilibre(pivot)
    equilibre(a) ← eq_a - min(eq_p, 0) + 1
    equilibre(pivot) ← max( eq_a+2, eq_a+eq_p+2, eq_p+1 )
    a ← pivot
    RETOURNER a
FIN
    
```



# Rotation simple

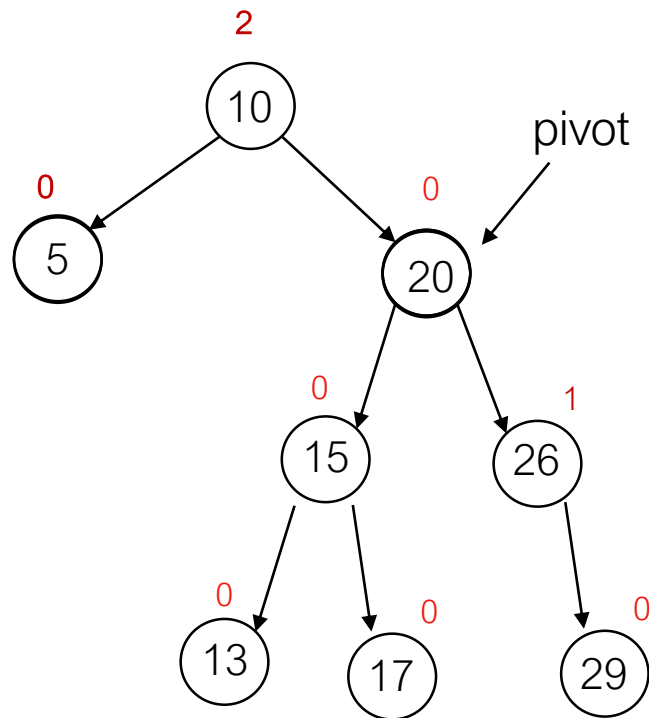
- Autre exemple de rotation simple avec la suppression :



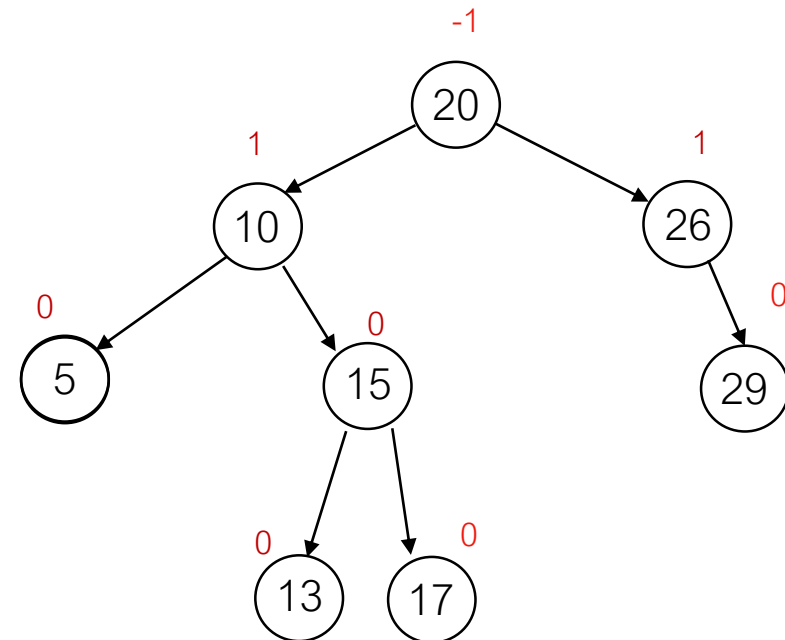
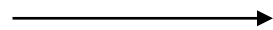
# Rotation simple

- Autre exemple de rotation simple avec la suppression :

$pivot \leftarrow fd(a)$   
 $fd(a) \leftarrow fg(pivot)$   
 $fg(pivot) \leftarrow a$   
...  
 $a \leftarrow pivot$

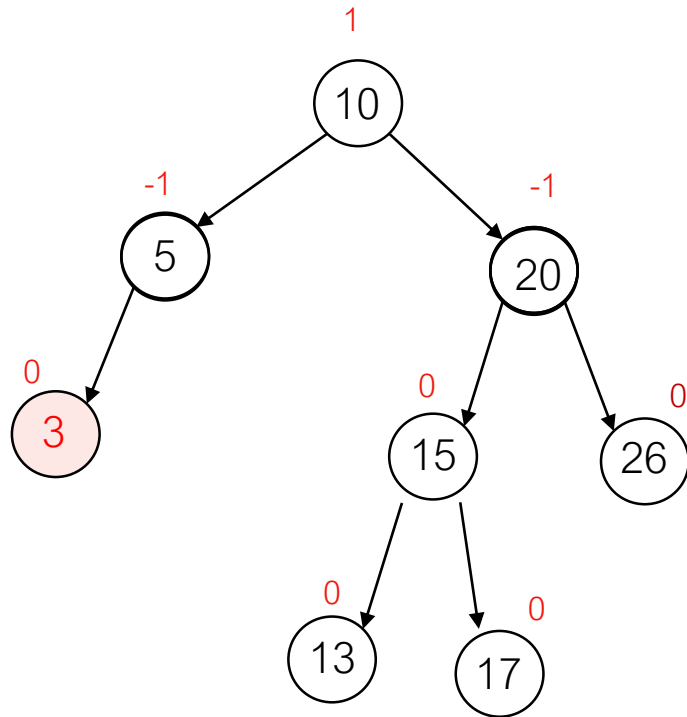


Rotation gauche



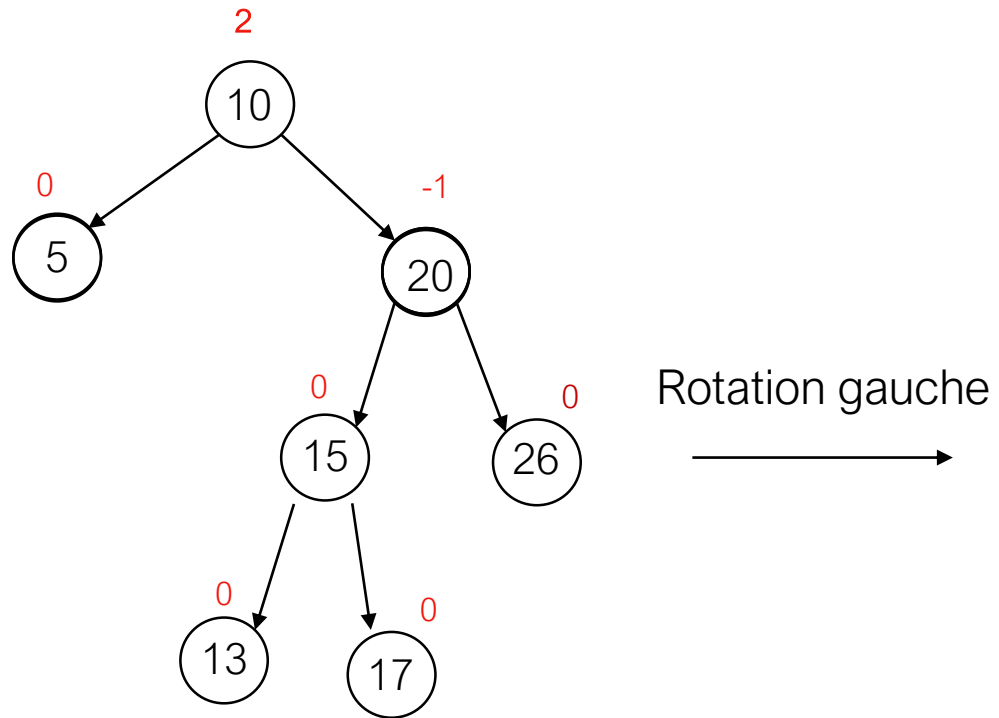
# Rotation double

- Cas 3 : élément supprimé dans le sous-arbre gauche suivant :



# Rotation double

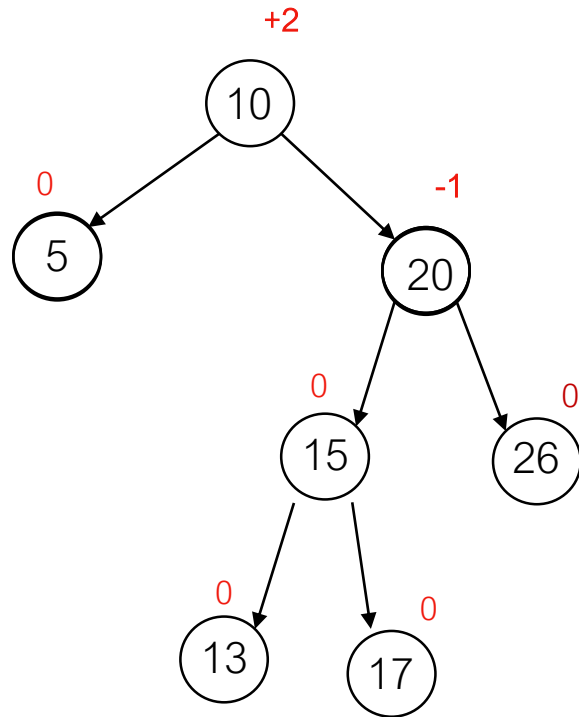
- Cas 3 : élément supprimé dans le sous-arbre gauche suivant :



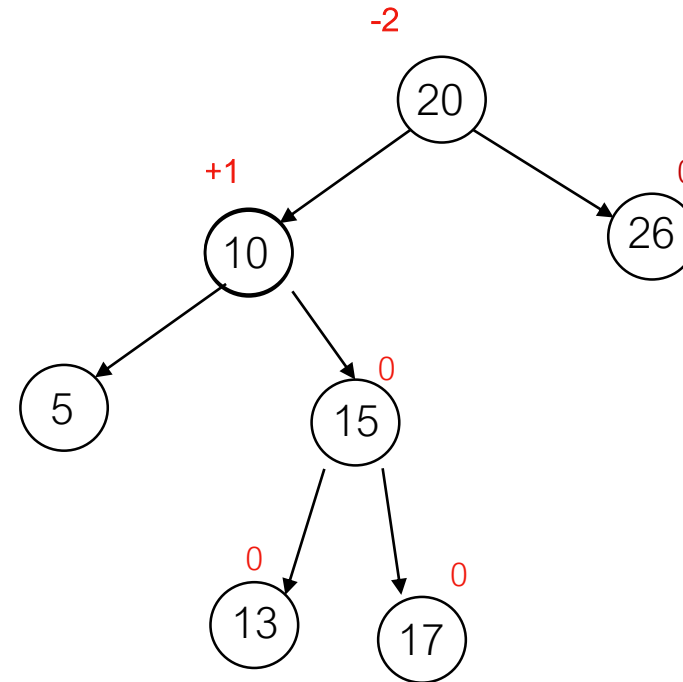
# Rotation double

- Cas 3 : élément supprimé dans le sous-arbre gauche suivant :

$pivot \leftarrow fd(a)$   
 $fd(a) \leftarrow fg(pivot)$   
 $fg(pivot) \leftarrow a$   
...  
 $a \leftarrow pivot$



Rotation gauche

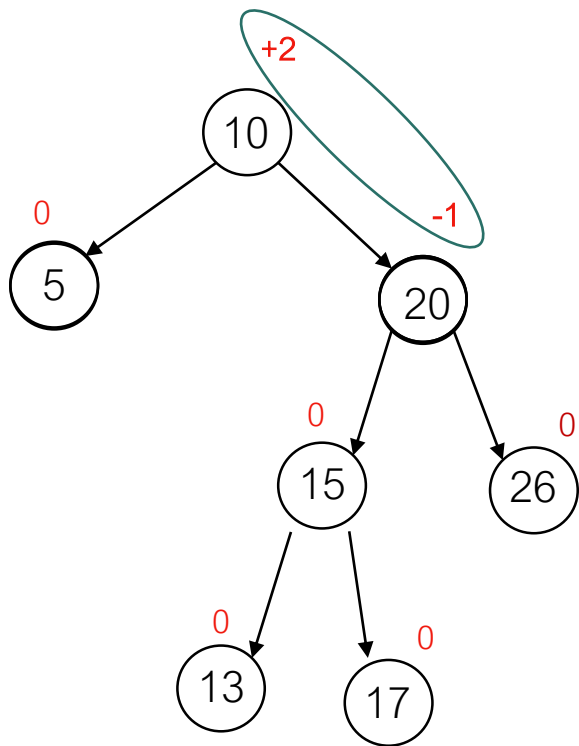




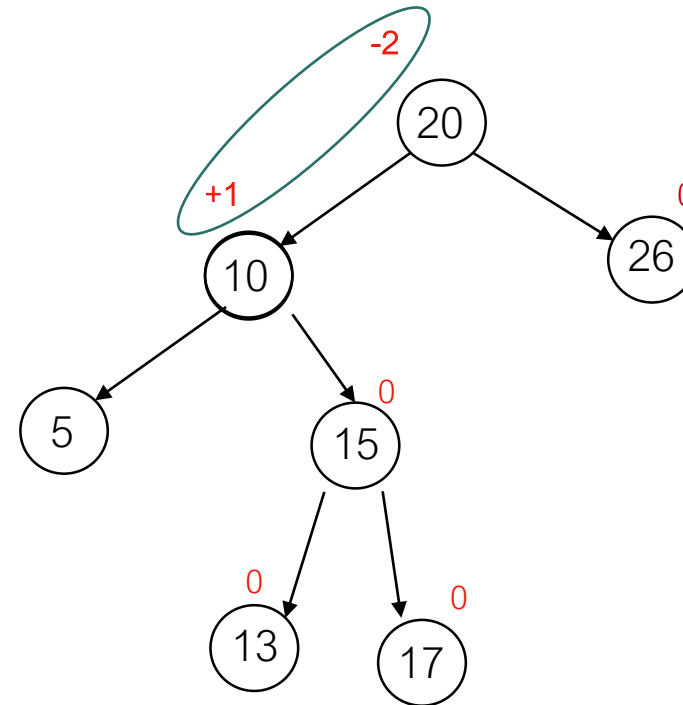
# Rotation double

- Cas 3 : élément supprimé dans le sous-arbre gauche suivant :

$pivot \leftarrow fd(a)$   
 $fd(a) \leftarrow fg(pivot)$   
 $fg(pivot) \leftarrow a$   
...  
 $a \leftarrow pivot$



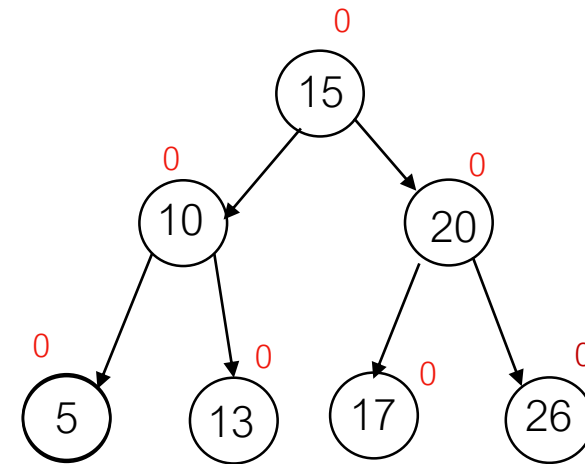
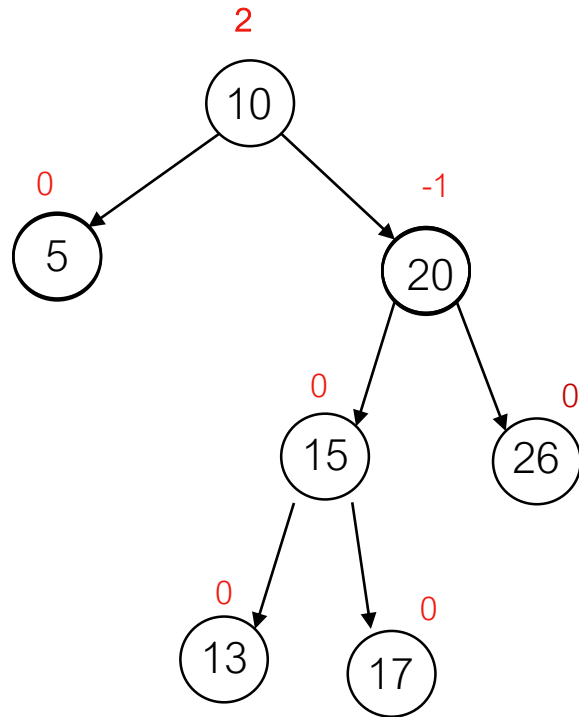
Rotation gauche



La rotation à gauche ne fonctionne pas !

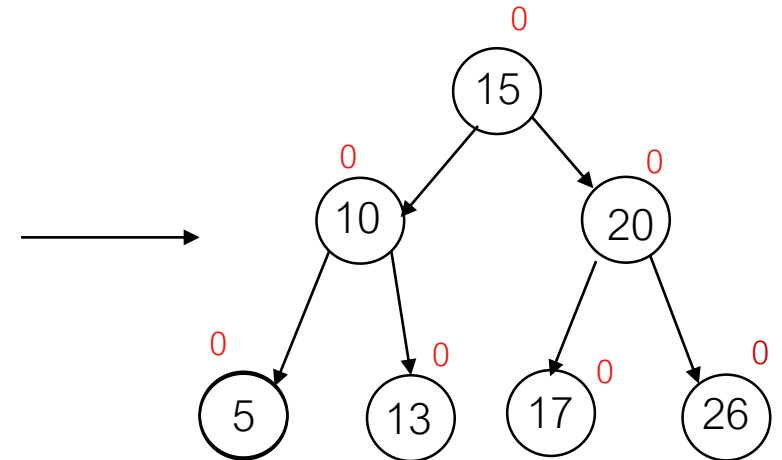
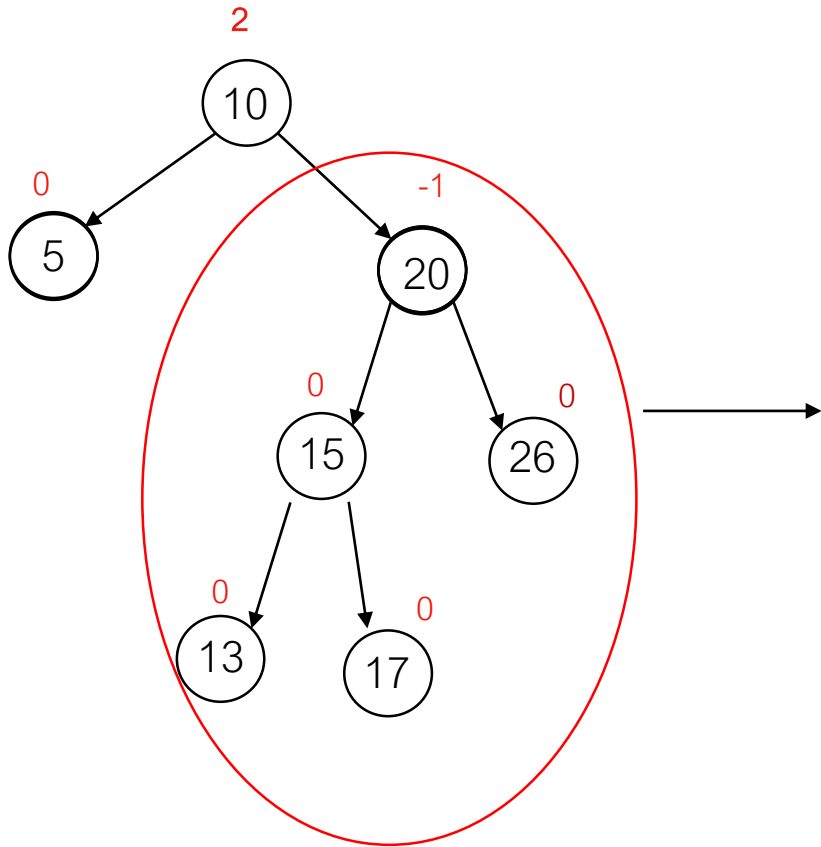
# Rotation double

- Cas 3 : élément supprimé dans le sous-arbre gauche suivant :



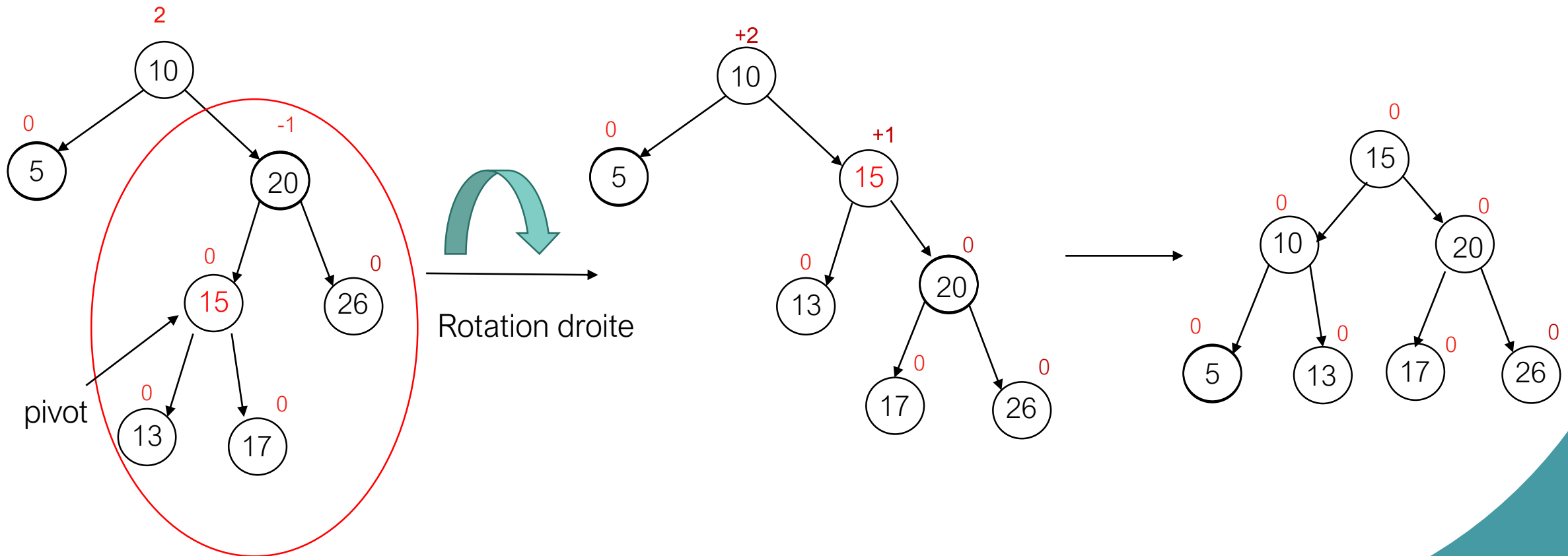
# Rotation double

- Cas 3 : élément supprimé dans le sous-arbre gauche suivant : **double rotation gauche**



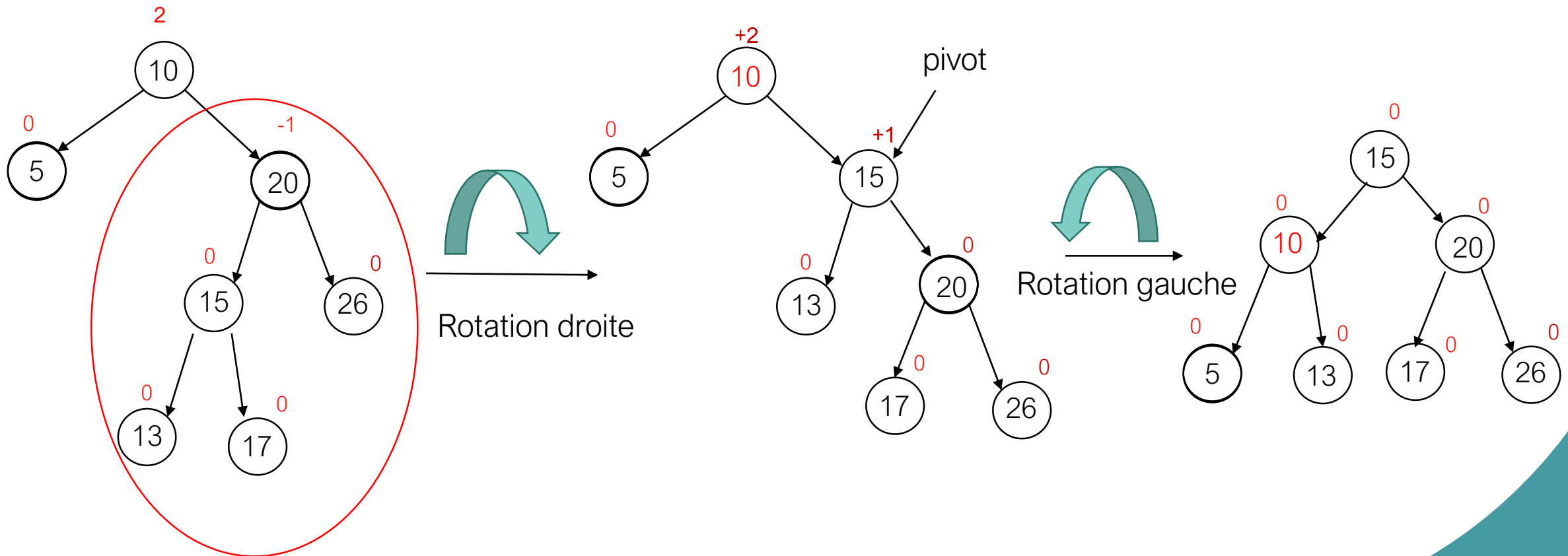
# Rotation double

- Cas 3 : élément supprimé dans le sous-arbre gauche suivant : **double rotation gauche**



# Rotation double

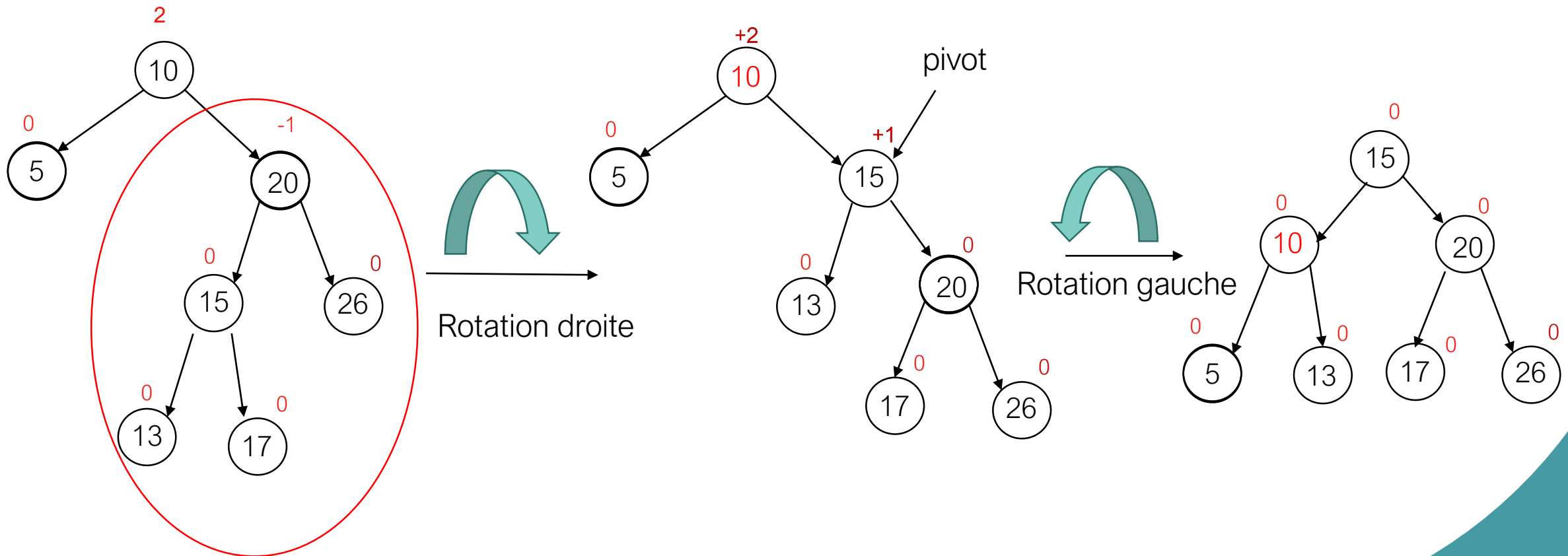
- Cas 3 : élément supprimé dans le sous-arbre gauche suivant : **double rotation gauche**



# Rotation double

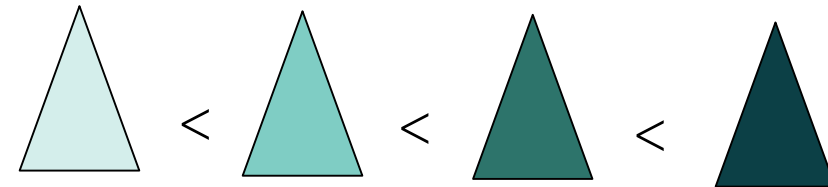
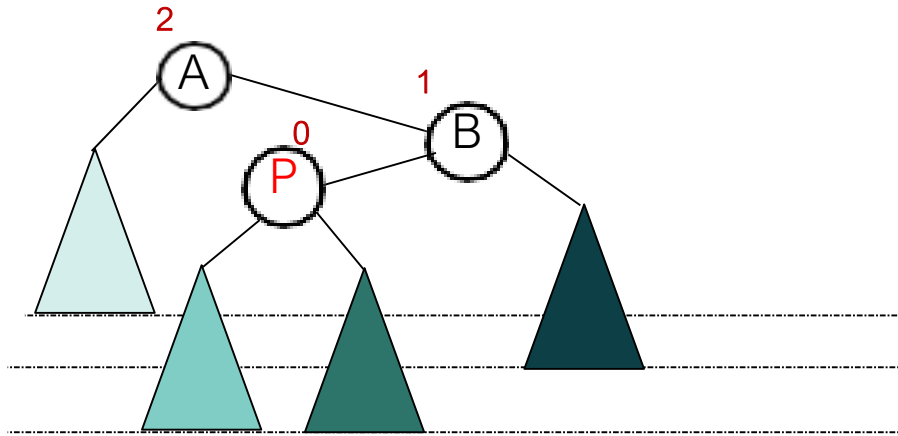
- Cas 3 : élément supprimé dans le sous-arbre gauche suivant : **double rotation gauche**

A.K.A. : rotation droite gauche



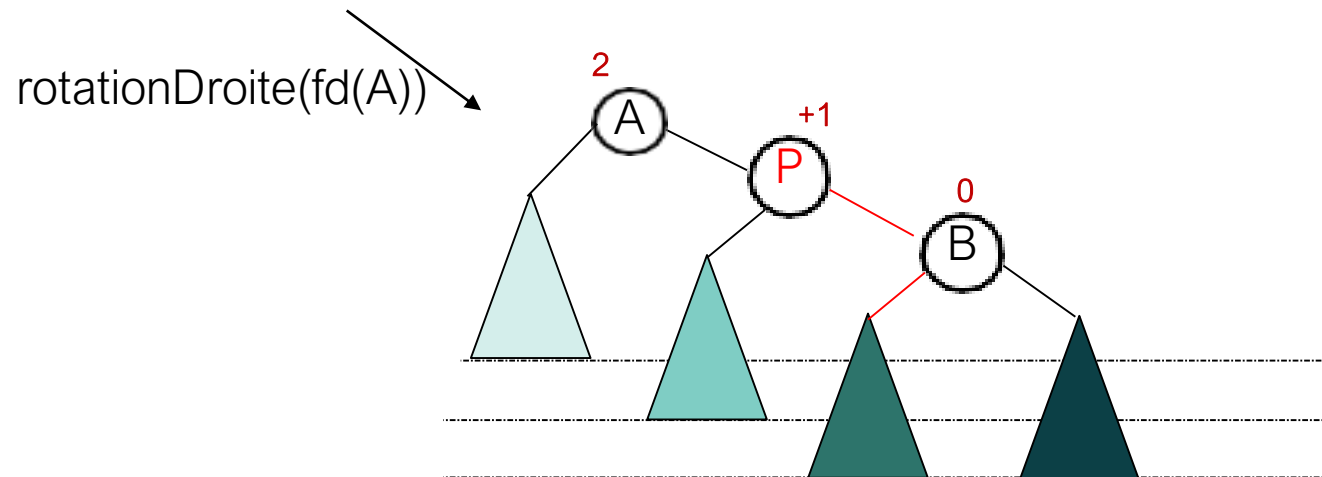
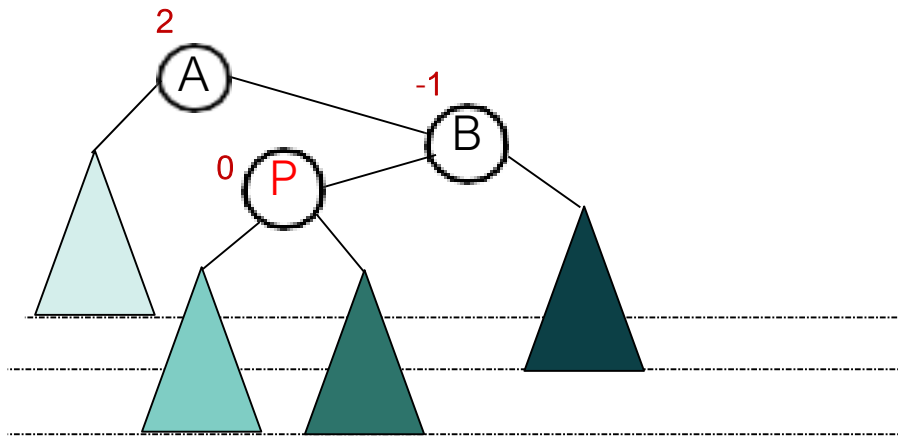
# Rotation double

- Cas 3 : hauteur trop importante dans la partie gauche du sous-arbre droit: **double rotation gauche**



# Rotation double

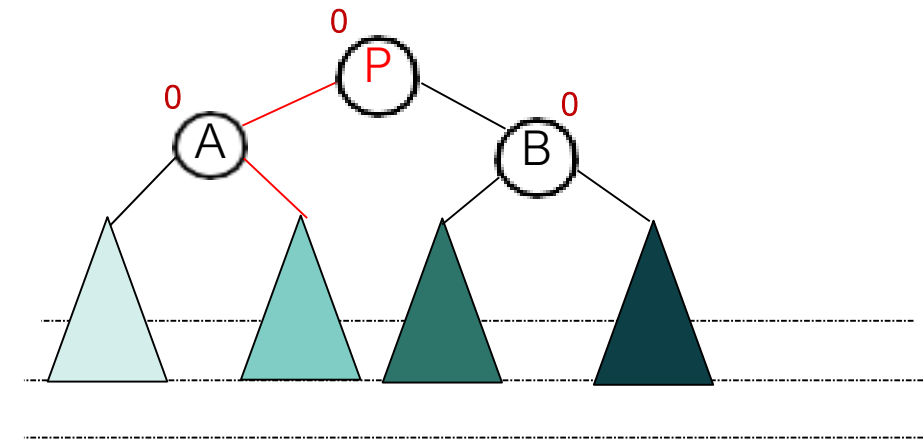
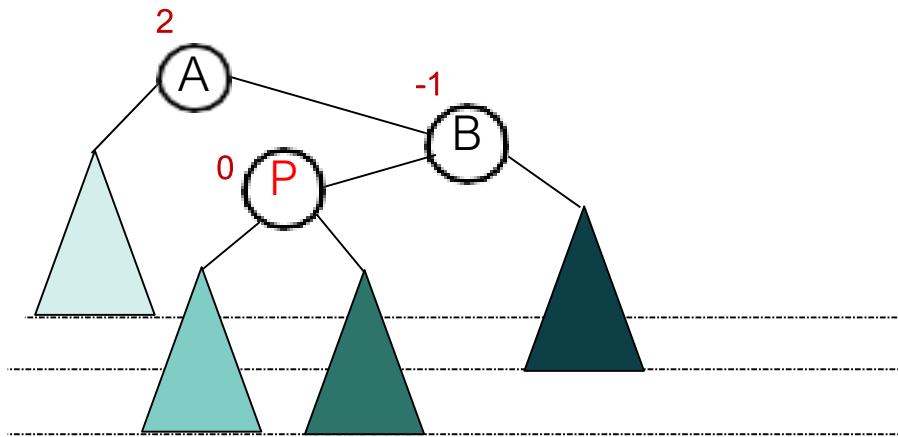
- Cas 3 : hauteur trop importante dans la partie gauche du sous-arbre droit: **double rotation gauche**



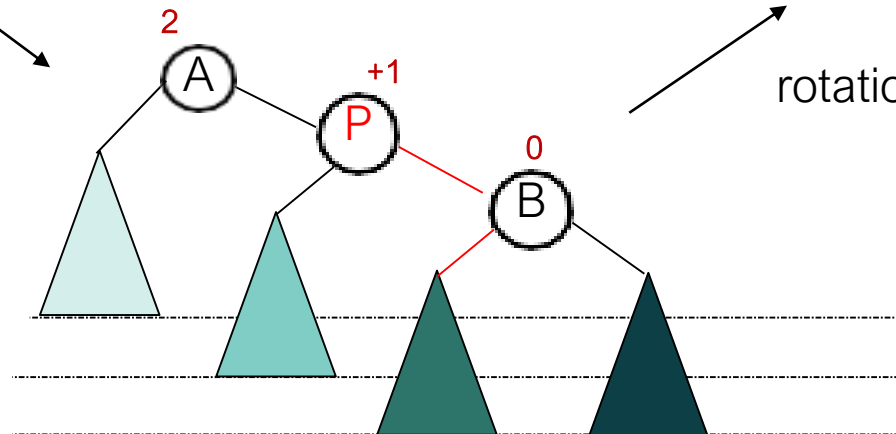


# Rotation double

- Cas 3 : hauteur trop importante dans la partie gauche du sous-arbre droit: **double rotation gauche**



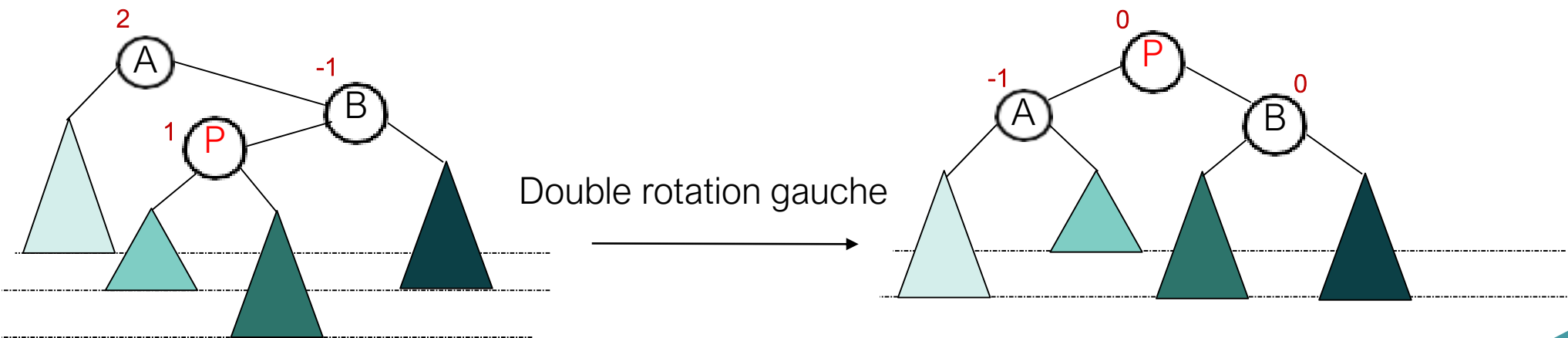
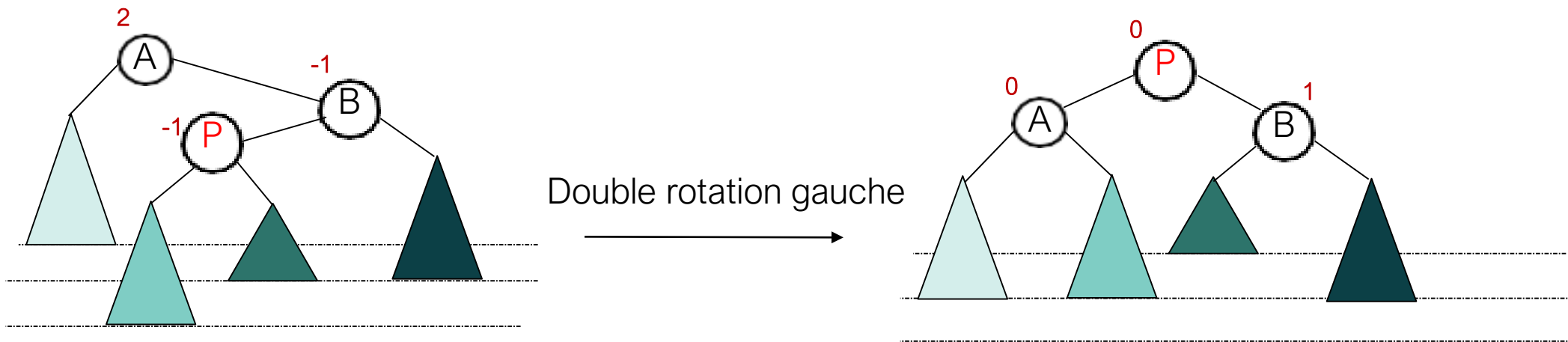
rotationDroite(fd(A))



rotationGauche(A)

# Rotation double

- Cas 3 : hauteur trop importante dans la partie gauche du sous-arbre droit: **double rotation gauche**



# Rotation double

- Cas 3 : hauteur trop importante dans la partie gauche du sous-arbre droit: **double rotation gauche**
- Algorithme :

FONCTION doubleRotationGauche(a: ptr sur Arbre) :

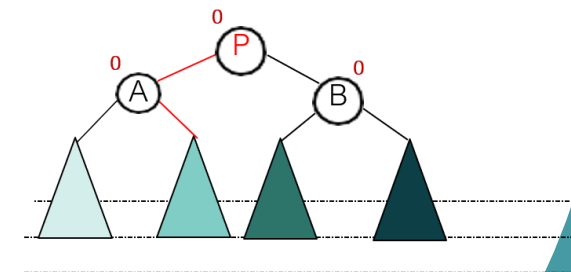
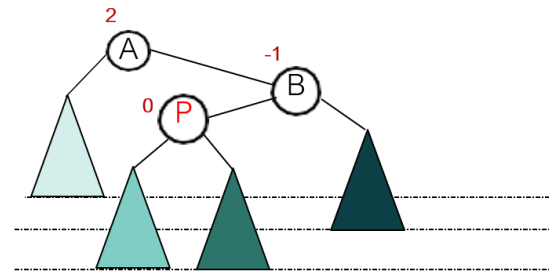
ptr sur Arbre

DEBUT

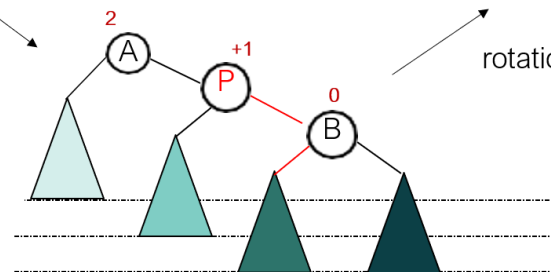
$fd(a) \leftarrow rotationDroite(fd(a))$

RETOURNER rotationGauche(a)

FIN



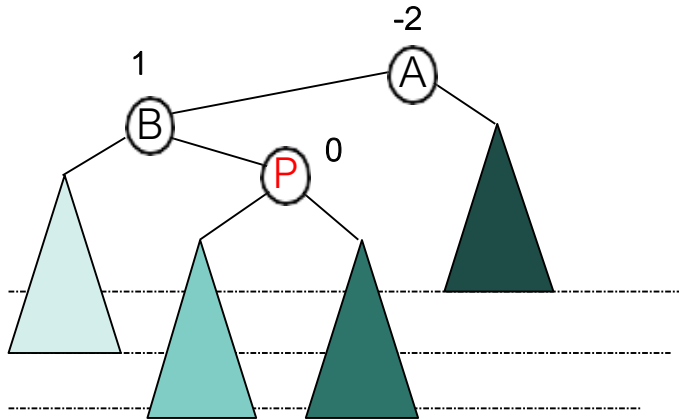
rotationDroite(fd(A))



rotationGauche(A)

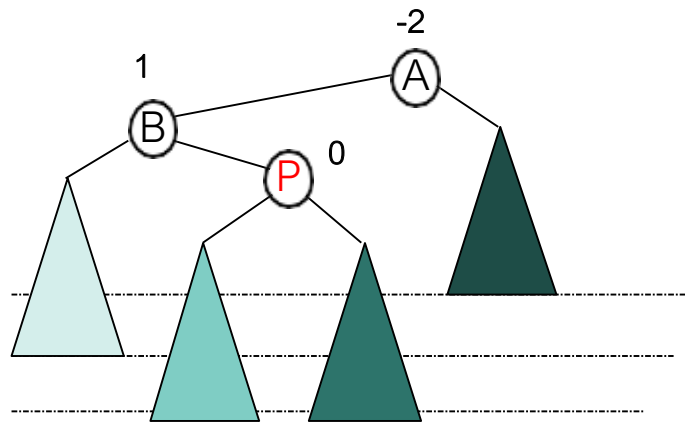
# Rotation double

- Cas 4 : hauteur trop importante dans la partie droite du sous-arbre gauche: **double rotation droite**

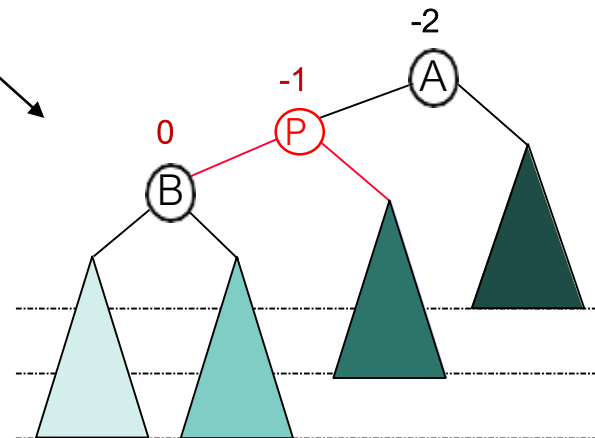


# Rotation double

- Cas 4 : hauteur trop importante dans la partie droite du sous-arbre gauche: **double rotation droite**

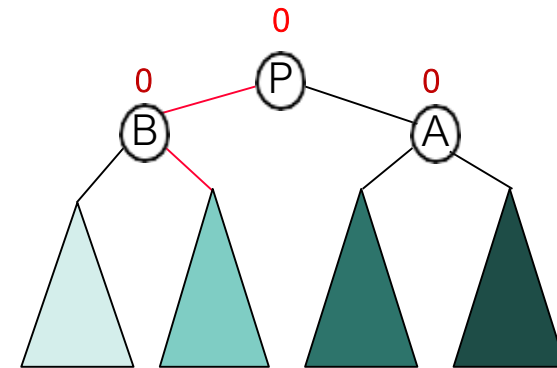
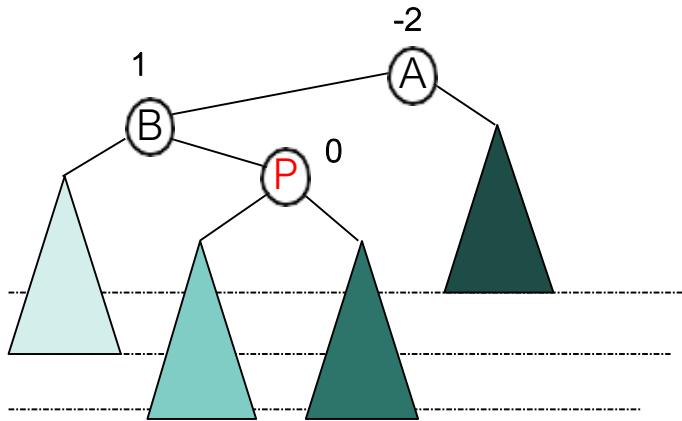


rotationGauche(fg(A))

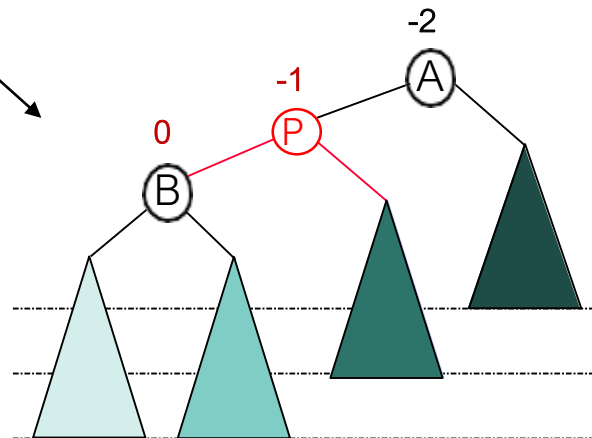


# Rotation double

- Cas 4 : hauteur trop importante dans la partie droite du sous-arbre gauche: **double rotation droite**



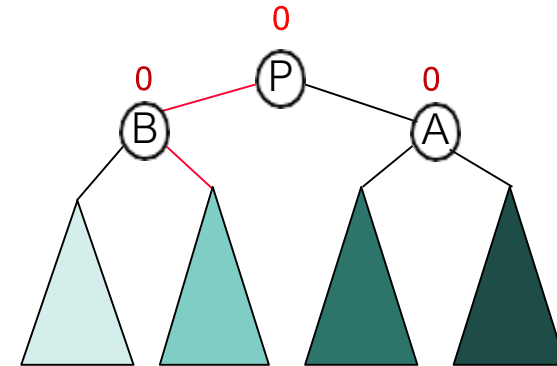
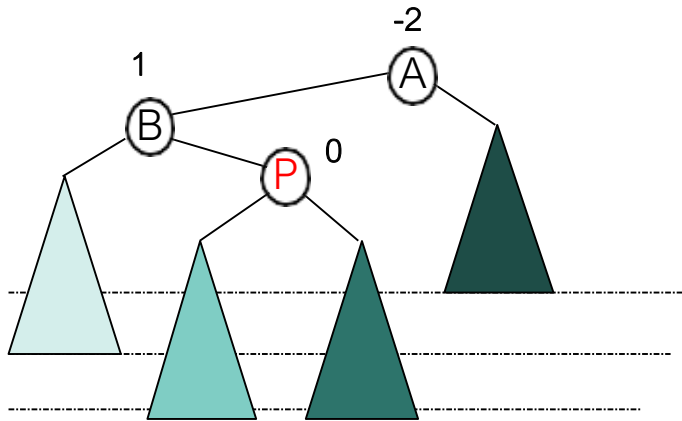
rotationGauche(fg(A))



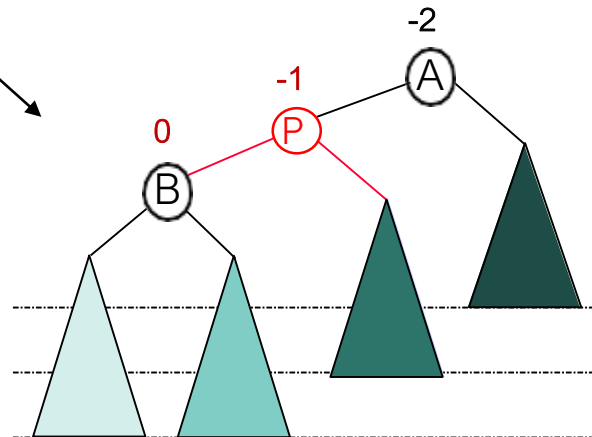
rotationDroite(A)

# Rotation double

- Cas 4 : hauteur trop importante dans la partie droite du sous-arbre gauche: **double rotation droite**  
A.K.A. : rotation gauche droite



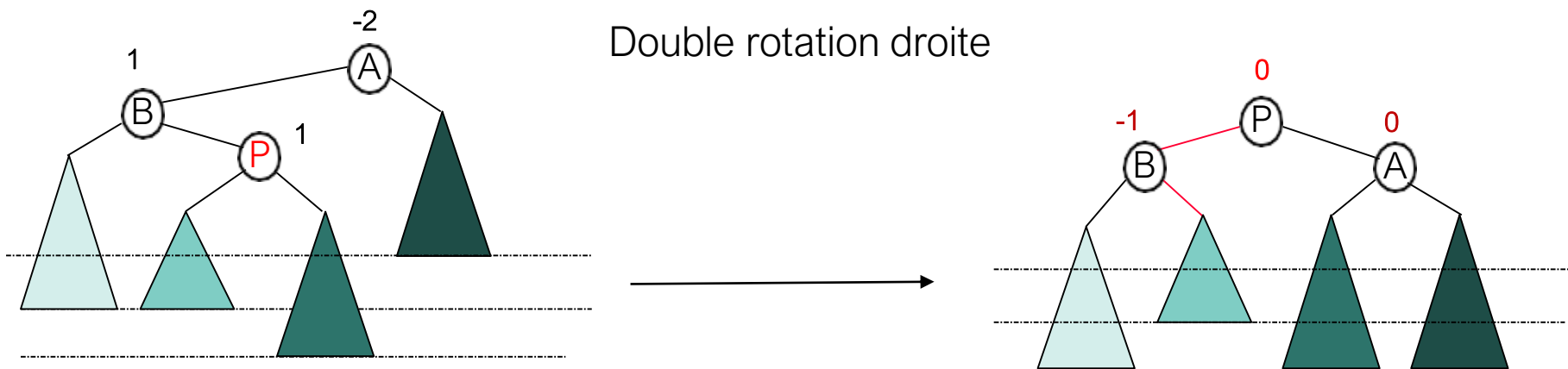
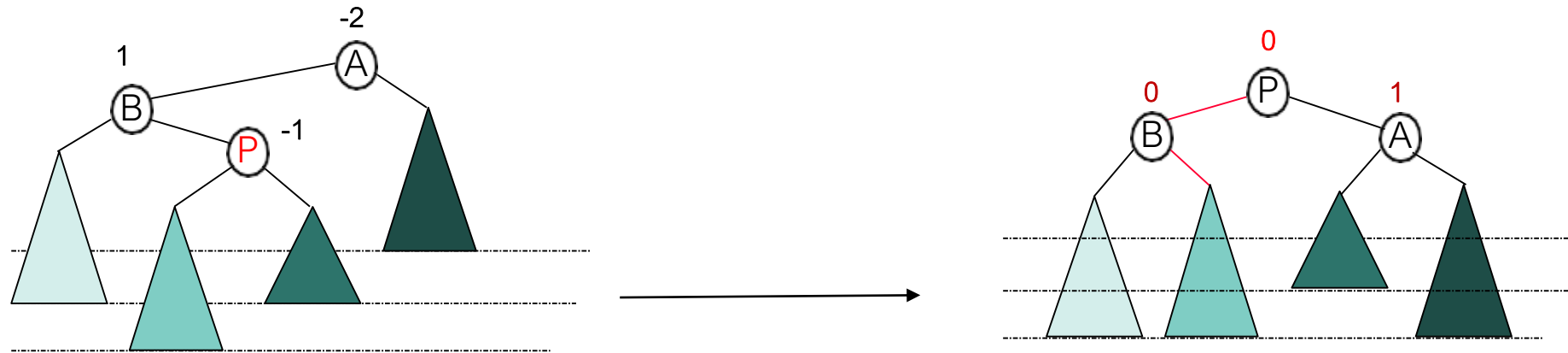
rotationGauche(fg(A))



rotationDroite(A)

# Rotation double

- Cas 4 : hauteur trop importante dans la partie droite du sous-arbre gauche: **double rotation droite**





# Rotation double

- Cas 4 : hauteur trop importante dans la partie droite du sous-arbre gauche: **double rotation droite**

- Algorithme :

FONCTION doubleRotationDroite(a: ptr sur Arbre) :

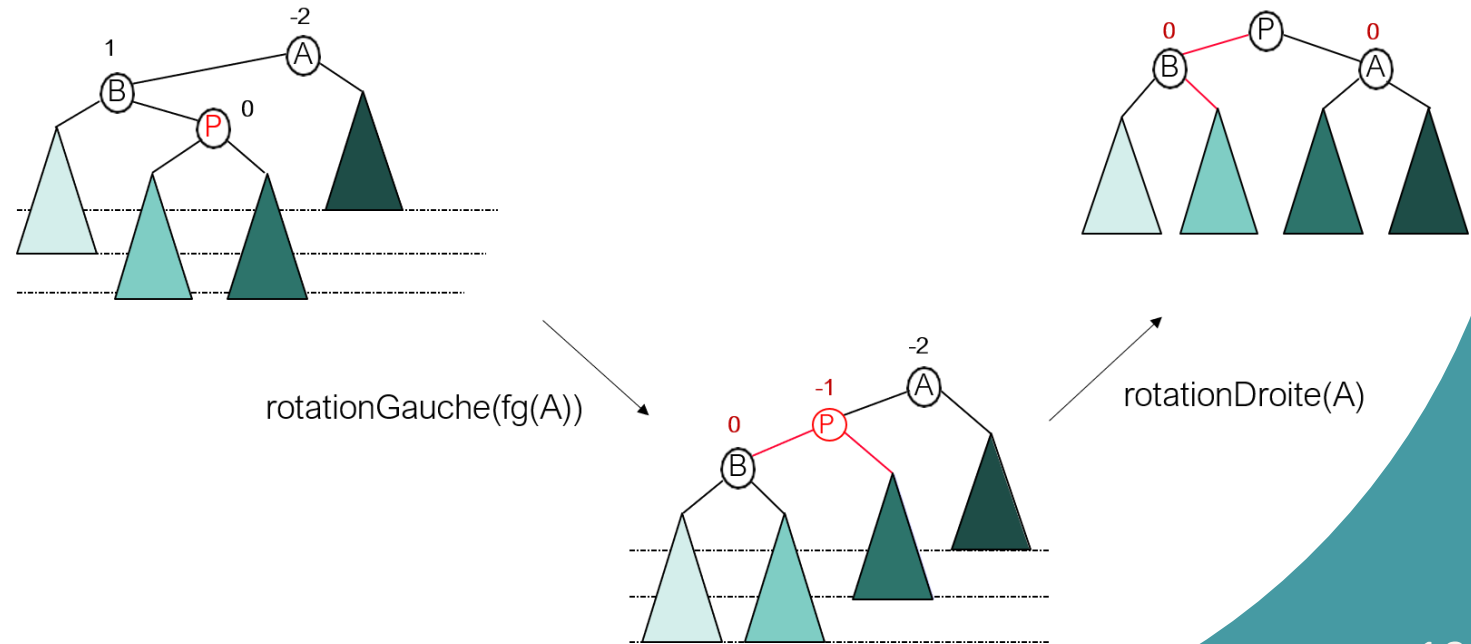
ptr sur Arbre

DEBUT

$fg(a) \leftarrow rotationGauche(fg(a))$

RETOURNER rotationDroite(a)

FIN



# Choix de la rotation

- La rotation à effectuer dépend de la structure de l'arbre (de quel côté se situe le déséquilibre).
- Pour connaître la rotation à effectuer, il suffit de regarder l'équilibre de la racine (nœud dont l'équilibre vaut +2 ou -2) et de ses fils :
  - Si **equilibre(racine) EST EGAL A -2**
    - → rotation droite
  - Si **equilibre(racine) EST EGAL A +2** → rotation gauche
    - → rotation gauche

# Choix de la rotation

- La rotation à effectuer dépend de la structure de l'arbre (de quel côté se situe le déséquilibre).
- Pour connaître la rotation à effectuer, il suffit de regarder l'équilibre de la racine (nœud dont l'équilibre vaut +2 ou -2) et de ses fils :
  - Si **équilibre(racine)** EST EGAL A -2
    - → rotation droite
    - Si **équilibre(fg(racine))** EST INF. OU EGAL A 0
      - → rotation simple
    - SINON
      - → rotation double
  - Si **équilibre(racine)** EST EGAL A +2 → rotation gauche
    - → rotation gauche
    - Si **équilibre(fd(racine))** EST SUP. OU EGAL A 0
      - → rotation simple
    - SINON
      - → rotation double

# Choix de la rotation

- Algorithme :

```
FONCTION equilibrerAVL(a: ptr sur Arbre) : ptr sur Arbre
DEBUT
  SI (equilibre(a) EST SUP. OU EGAL A 2) ALORS // sous-arbre droit plus profond
    SI (equilibre(fd(A)) SUP. OU EGAL A 0) ALORS
      RETOURNER rotationGauche(a)
    SINON
      RETOURNER doubleRotationGauche(a)
    FIN SI
  SINON SI (equilibre(a) EST INF. OU EGAL A -2) ALORS // sous-arbre gauche plus profond
    SI (equilibre(fg(a)) INF. OU EGAL A 0) ALORS
      RETOURNER rotationDroite(a)
    SINON
      RETOURNER doubleRotationDroite(a)
    FIN SI
  FIN SI
  RETOURNER a
FIN
```

# Complexité de l'AVL

- Les opérations de rotations sont à complexité constante  $O(1)$
- L'opération d'insertion ajoute une hauteur de 1 sur la branche où est inséré l'élément : une rotation suffit pour rééquilibrer l'arbre.
- L'opération de suppression peut exécuter une rotation sur chaque ancêtre du nœud supprimé. Mais celles-ci étant en temps constant, le nombre de rotations dépend de la hauteur de l'arbre.
- Donc, pour un arbre AVL de  $n$  nœuds, le temps total d'ajout ou suppression est :  **$O(\log_2(n))$**

# Résumé

- Le facteur d'équilibre d'un arbre représente la différence de hauteur entre le sous-arbre gauche et droit : il permet de quantifier l'équilibre de l'arbre.
- Les AVL sont des arbres auto-équilibrés garantissant une structure de l'arbre permettant d'optimiser les opérations de recherche, d'insertion et de suppression.
- L' AVL exploite des opérations de rotation autour d'un nœud pivot pour rééquilibrer un arbre.
- Il existe d'autres types d'arbres auto-équilibrés comme les arbres rouge-noirs!