

15. Arbres AVL

G. Adelson-Velskii et E. M. Landis, *An Algorithm for the Organization of Information*. Soviet Mathematics Doklady, 3:1259–1263, 1962





Un arbre AVL est

- un Arbre Binaire de Recherche
- qui se maintient équilibré à chaque opération
(insertion / suppression)

Il garantit donc un équilibre entre -1 et 1 pour chaque noeud.



- Rappel: on sait mesurer hauteur et équilibre en $O(n)$

```
fonction hauteur (r)
    si r == ∅,
        retourner 0
    sinon,
        retourner 1 + max(hauteur(r.gauche), hauteur(r.droit))
```

```
fonction équilibre (r)
    si r == ∅, retourner 0
    sinon,
        retourner hauteur(r.gauche) - hauteur(r.droit)
```

- Pour le faire en $O(1)$, il faut modifier la structure des noeuds



Version simple : stocker la hauteur du sous-arbre dont le noeud est la racine

- mis à jour à chaque insertion / suppression / équilibrage
- 8 bits suffisent, la hauteur croissant logarithmiquement avec le nombre d'éléments stockés

```
structure Noeud<K>  
    K clé  
    Noeud* gauche  
    Noeud* droit  
    Entier hauteur
```

Version compact : stocker le déséquilibre seul sur 2 bits dans la racine, sur 1 bit dans ses enfants. (pas étudié ici)



- La fonction `hauteur` devient de complexité $O(1)$

```
fonction hauteur (r)
    si r == ∅, retourner 0
    sinon,      retourner r.hauteur
```

- Après toute opération qui modifie la hauteur d'un enfant de `r`, il faudra recalculer `r.hauteur` (en sortie de récursion)

```
fonction calculer_hauteur (r)
    si r != ∅,
        r.hauteur = 1 + max(hauteur(r.gauche), hauteur(r.droit))
```

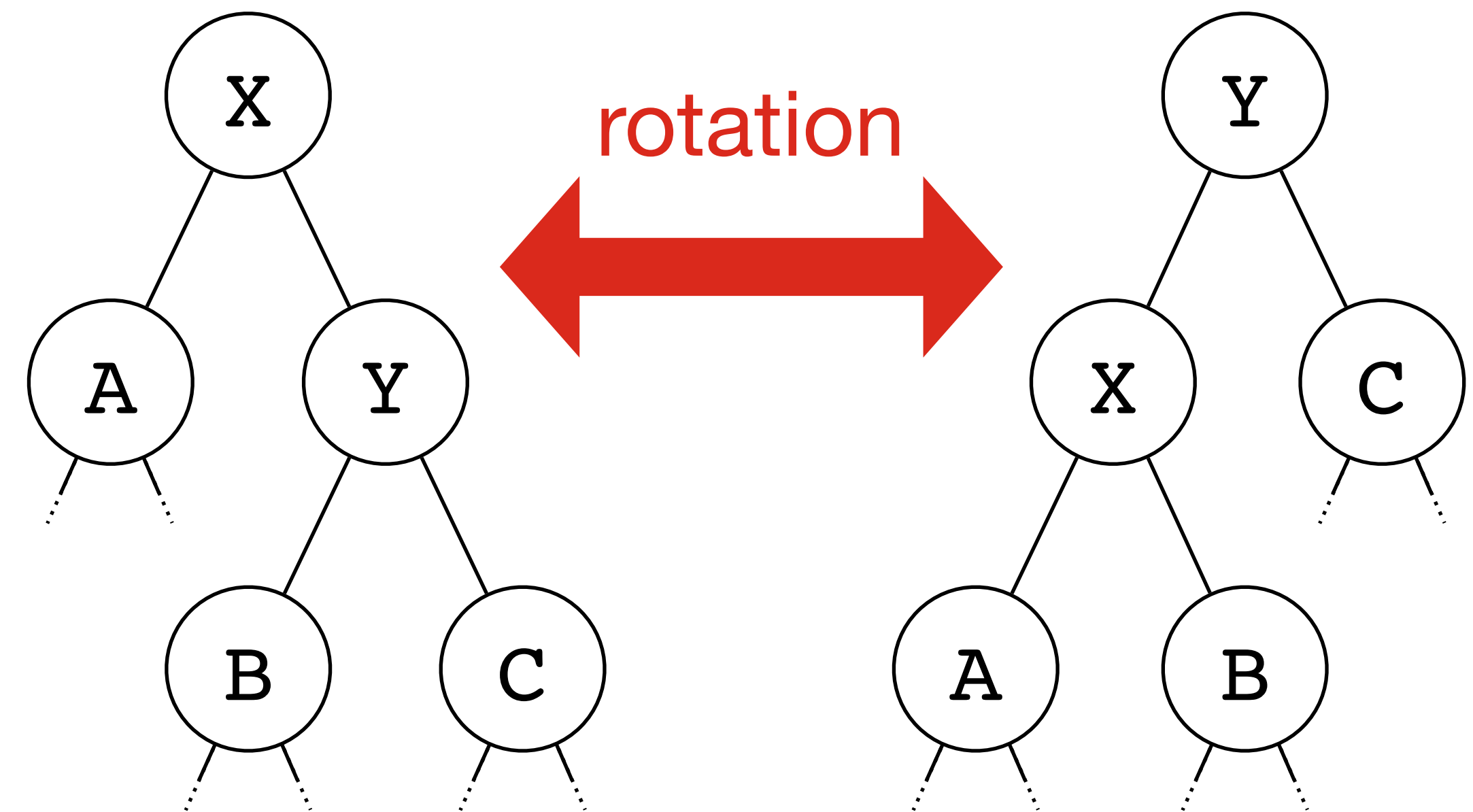


Idée

Dès qu'un déséquilibre de $+2$ ou -2 est détecté, le ramener dans $[-1, 1]$.

À tout moment, le pire déséquilibre est donc de ± 2 .

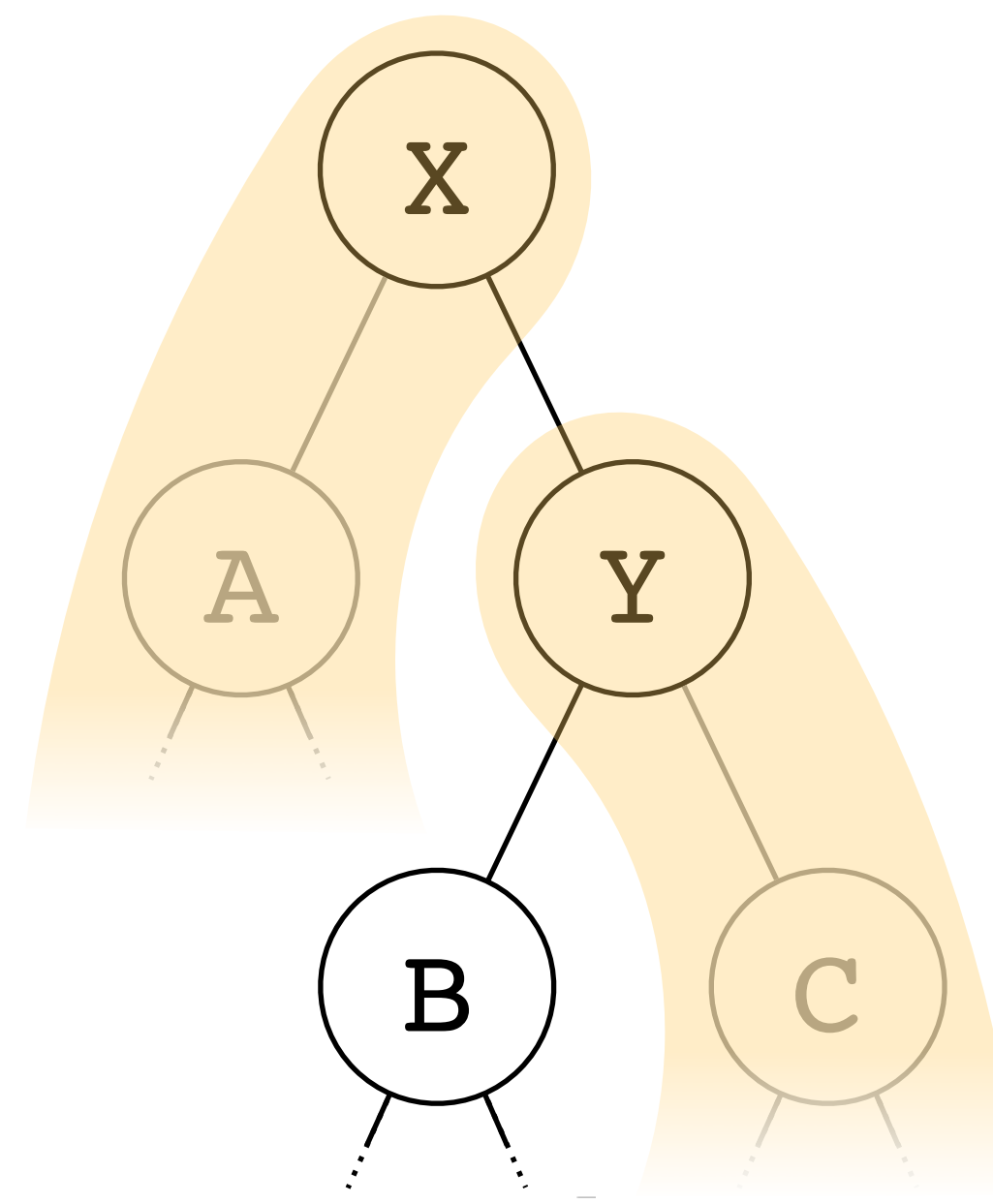
L'équilibrage se fait par **rotation**, en **retour de récursion**, du plus profond d'abord vers la racine.



Rotation gauche



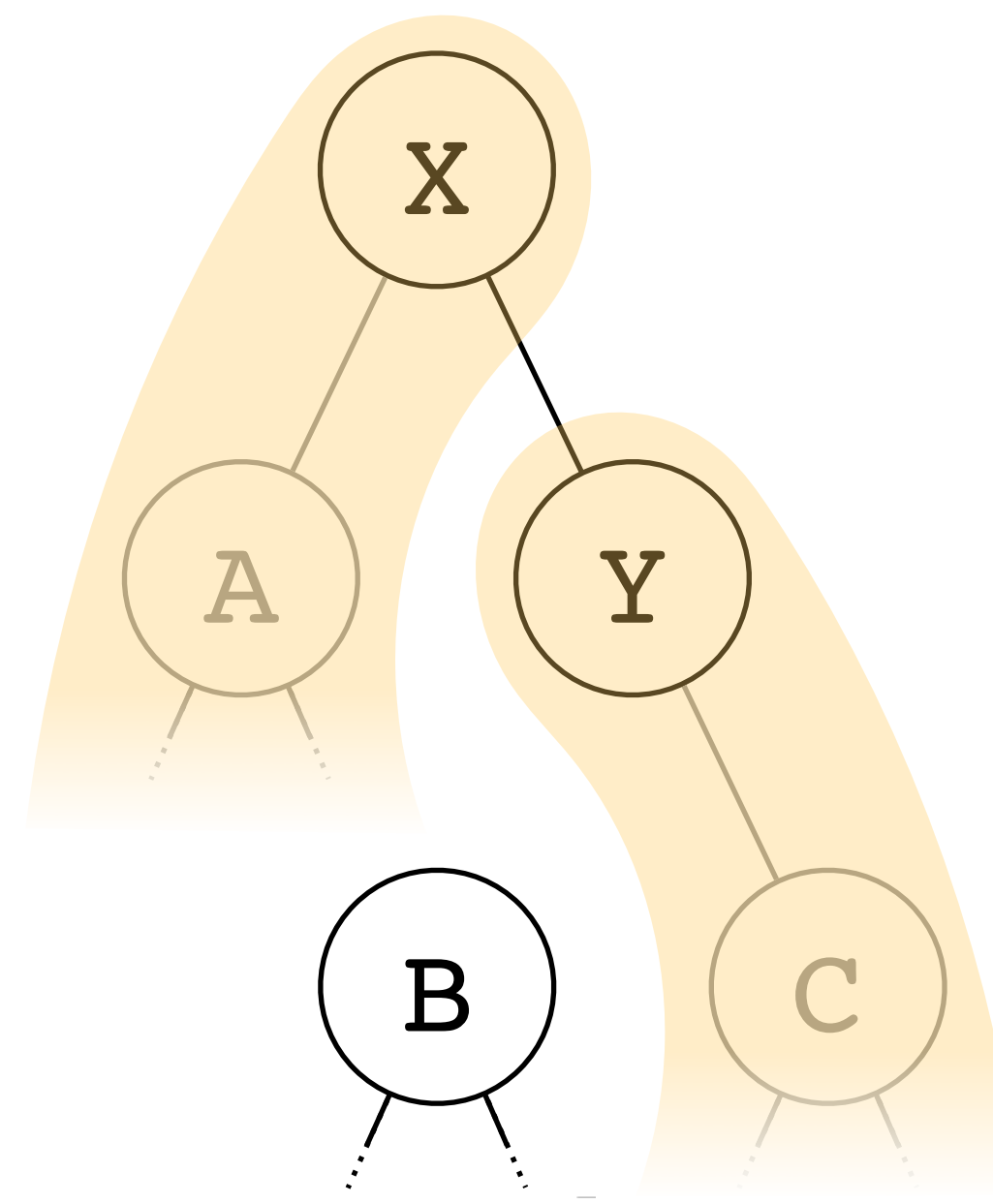
- Faire monter l'enfant droit Y à la place de son parent X
(On peut visualiser tous les noeuds tourner vers la gauche autour de B)



Rotation gauche



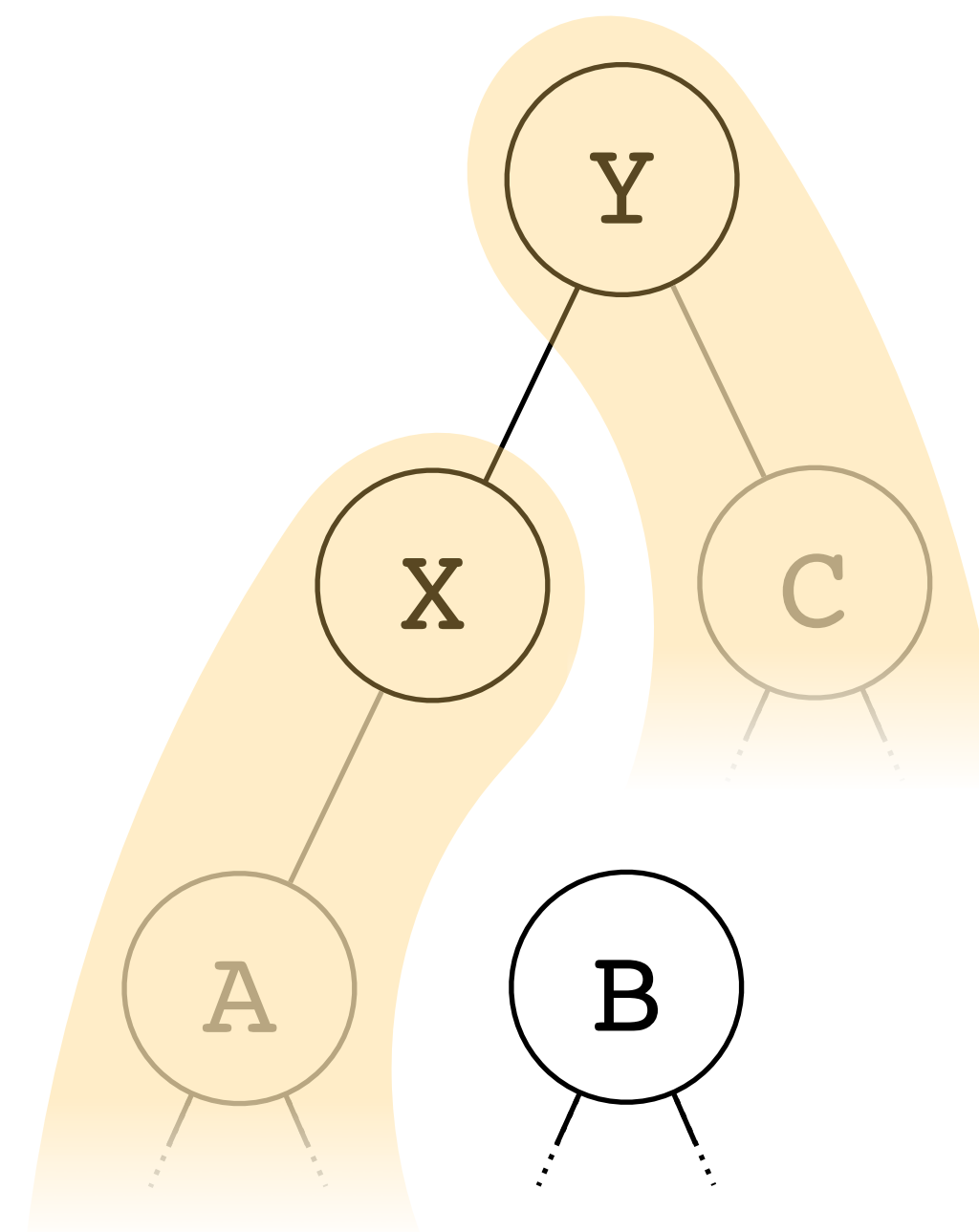
- Faire monter l'enfant droit Y à la place de son parent X
(On peut visualiser tous les noeuds tourner vers la gauche autour de B)



Rotation gauche



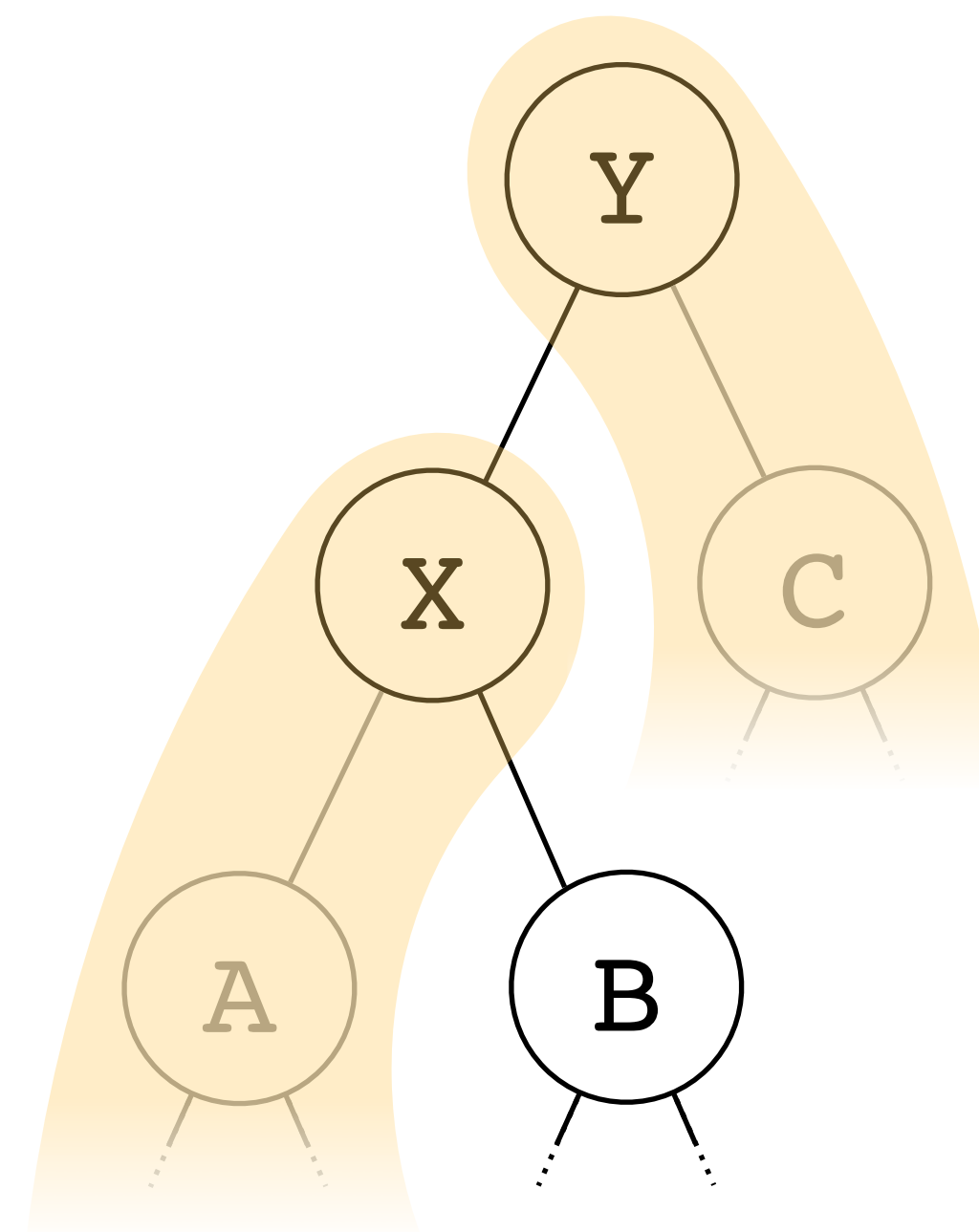
- Faire monter l'enfant droit Y à la place de son parent X
(On peut visualiser tous les noeuds tourner vers la gauche autour de B)



Rotation gauche



- Faire monter l'enfant droit Y à la place de son parent X
(On peut visualiser tous les noeuds tourner vers la gauche autour de B)

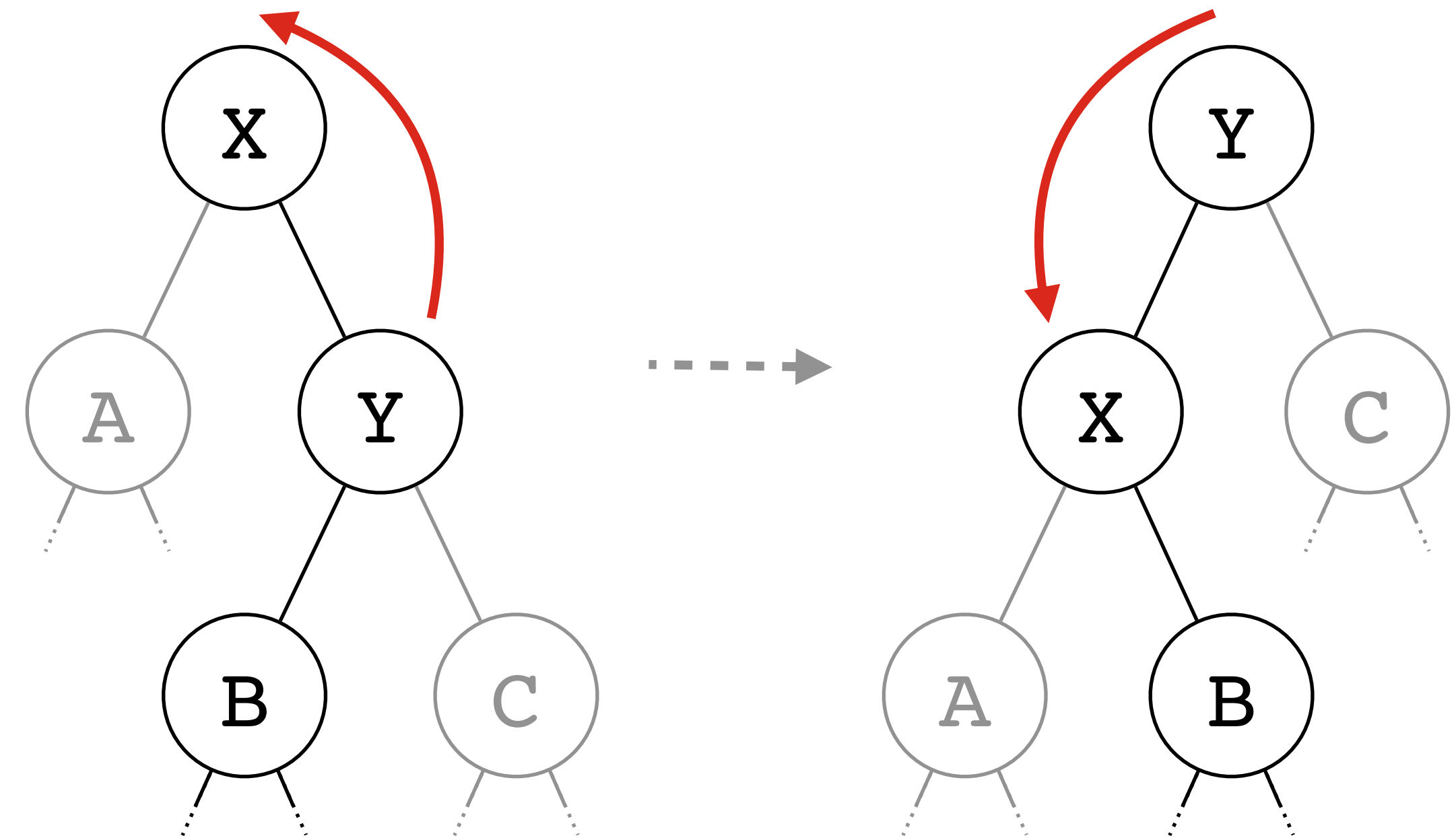


Rotation gauche



- Faire monter l'enfant droit Y à la place de son parent X
(On peut visualiser tous les noeuds tourner vers la gauche autour de B)
- En conservant la relation d'arbre binaire de recherche

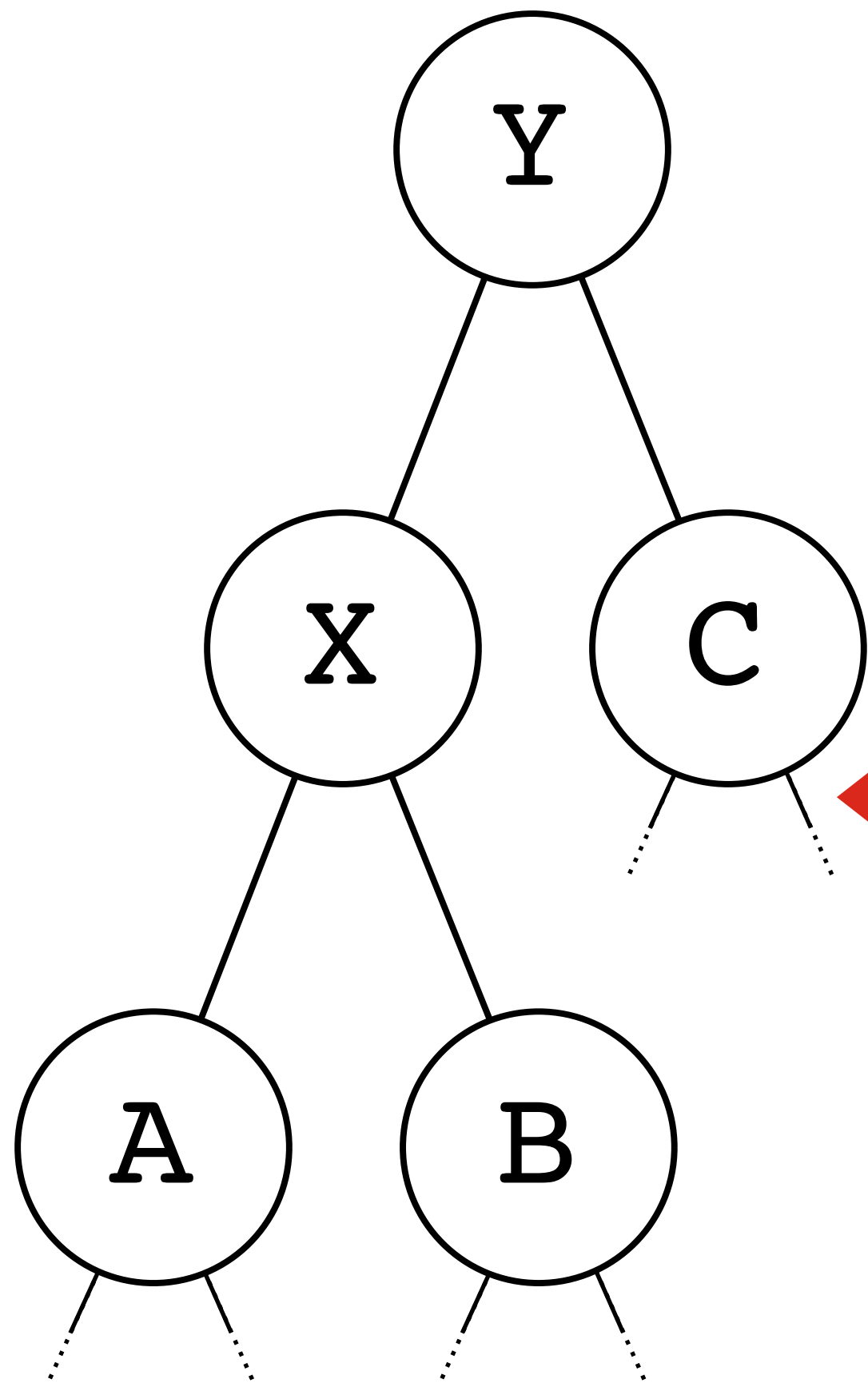
$$A < X < B < Y < C$$



- Rotation droite : l'inverse

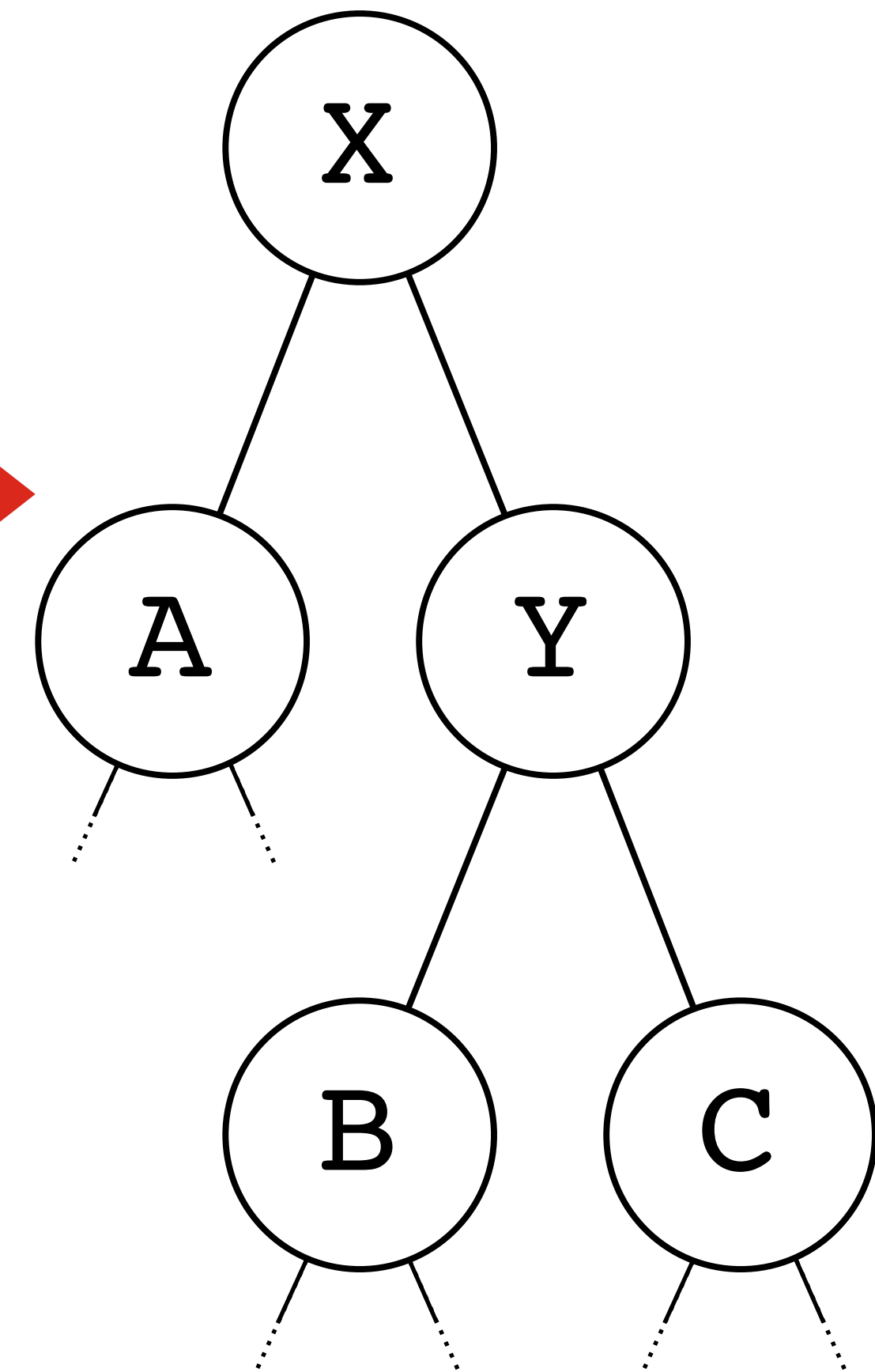


Rotations



```

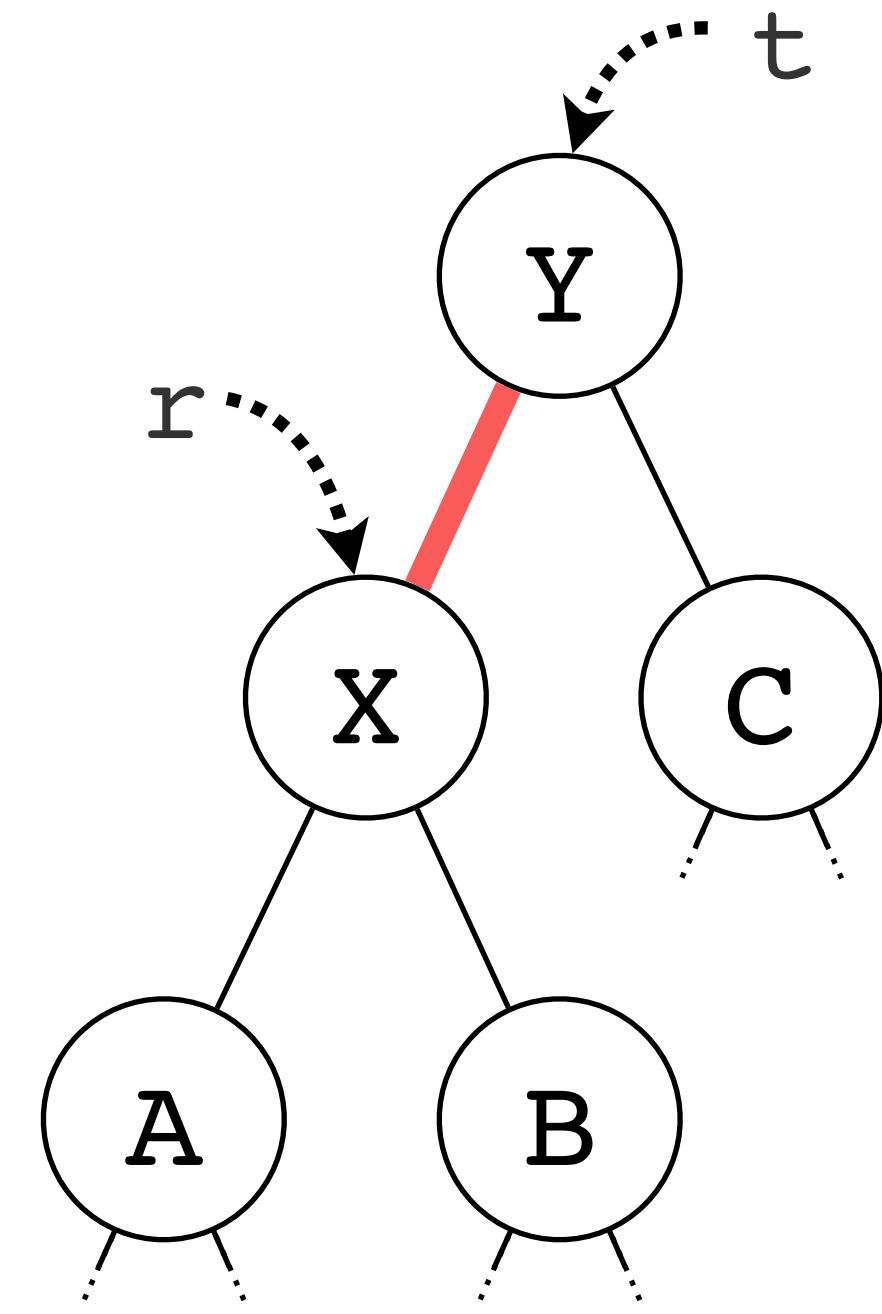
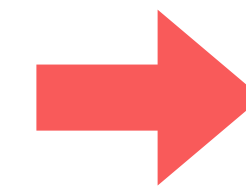
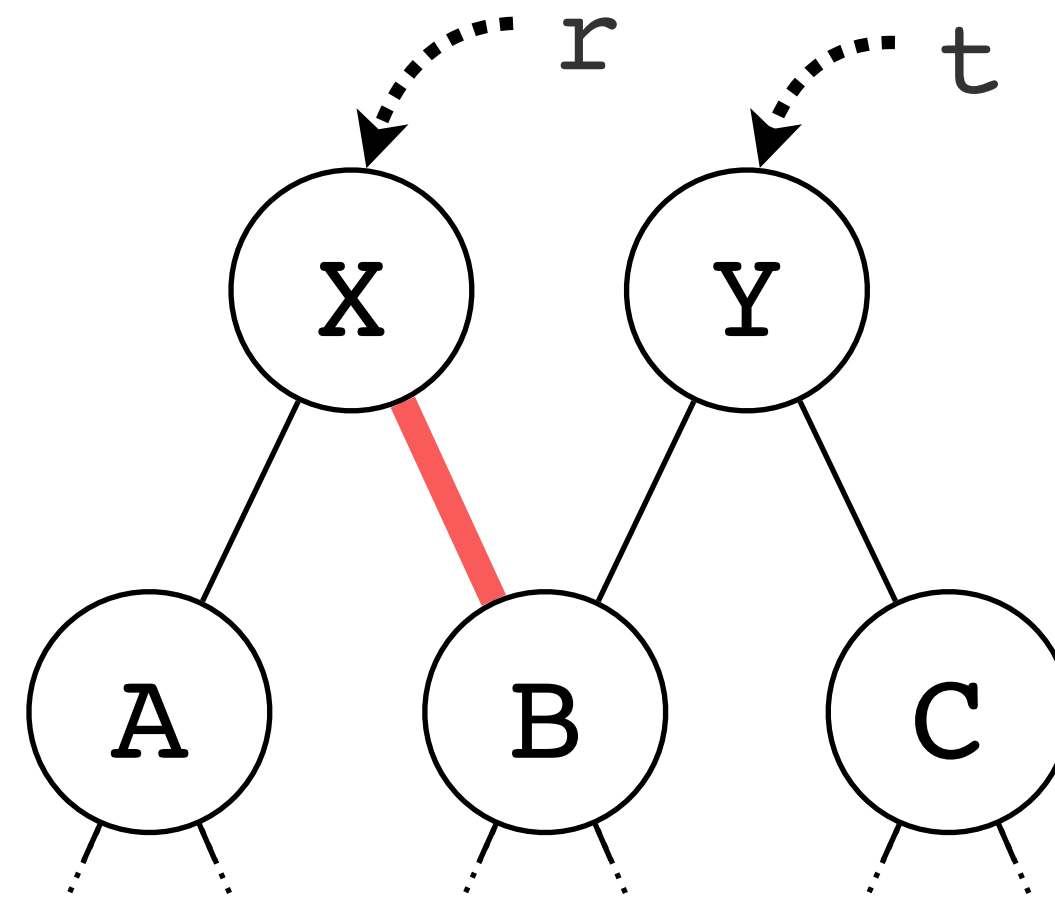
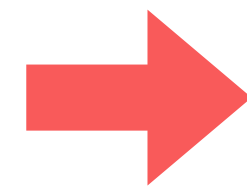
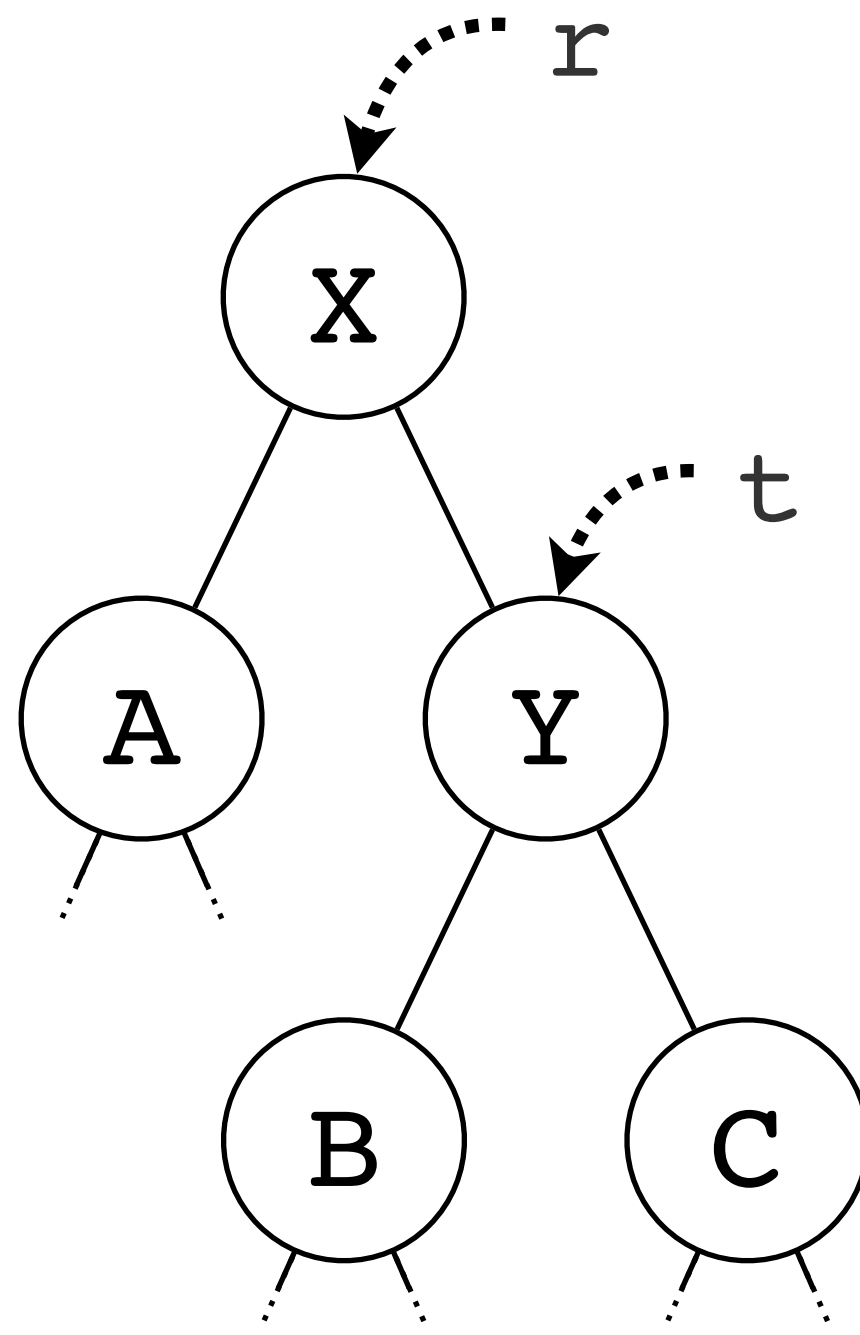
fonction rotation_droite (ref r) // Y
  t ← r.gauche // X
  r.gauche ← t.droit // B
  t.droit ← r // Y
  r ← t // X
  
```



```

fonction rotation_gauche (ref r) // X
  t ← r.droit // Y
  r.droit ← t.gauche // B
  t.gauche ← r // X
  r ← t // Y
  
```

Exemple - rotation gauche



```

fonction rotation_gauche (ref r) // X
  t ← r.droit // Y
  r.droit ← t.gauche // B
  t.gauche ← r // X
  r ← t // Y
  
```

Effet de la rotation



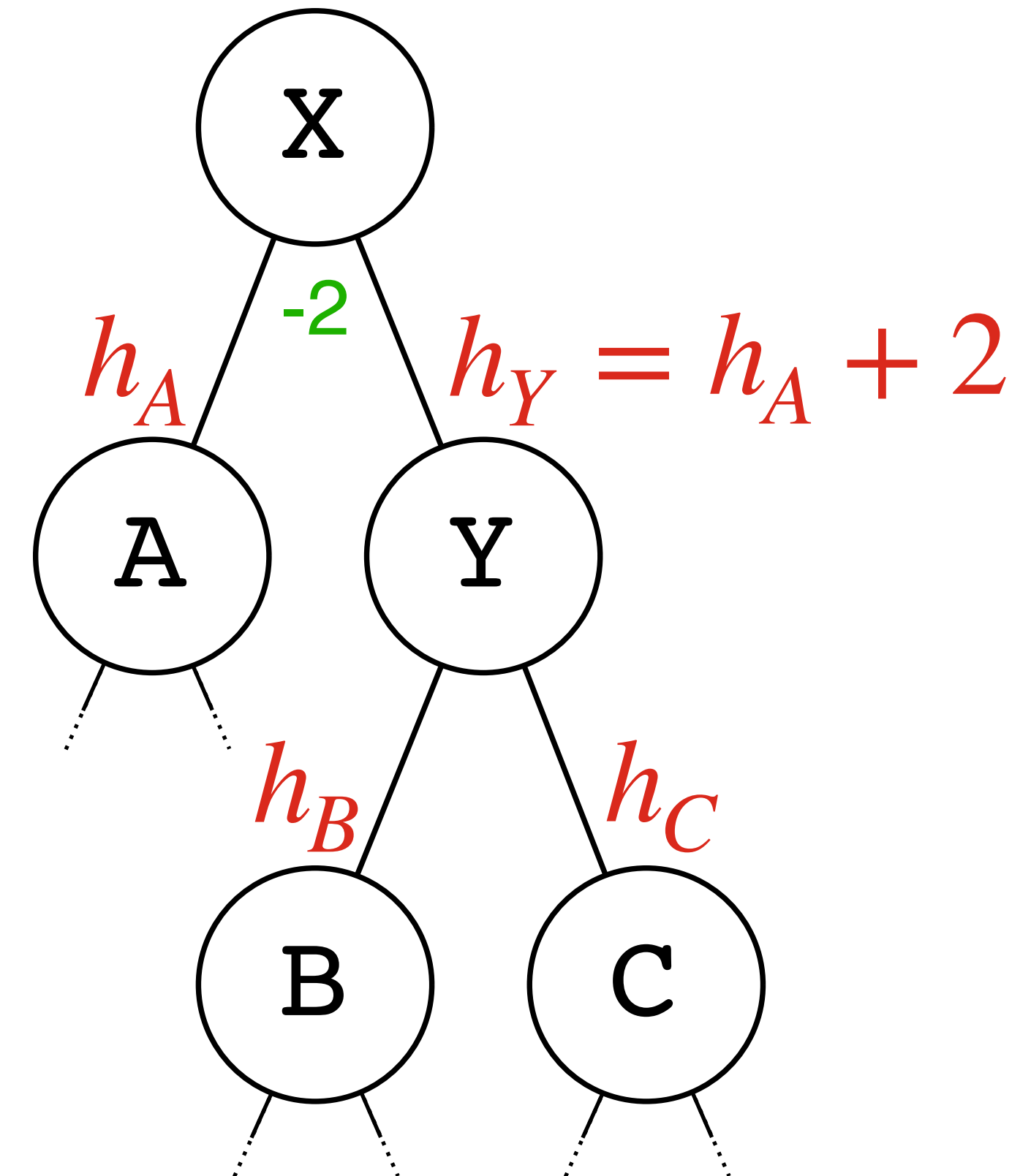
Dénotons h_N la hauteur d'un noeud N .

Supposons tous les noeuds équilibrés, sauf un déséquilibre à droite en X , donc

$$\begin{aligned} \text{équilibre}(X) &= h_A - h_Y = -2 \\ \Leftrightarrow h_Y &= h_A + 2 \end{aligned}$$

On sait aussi que $h_Y = 1 + \max(h_B, h_C)$ et parce que Y est équilibré,

$$\text{équilibre}(Y) = h_B - h_C \in \{-1, 0, 1\}$$



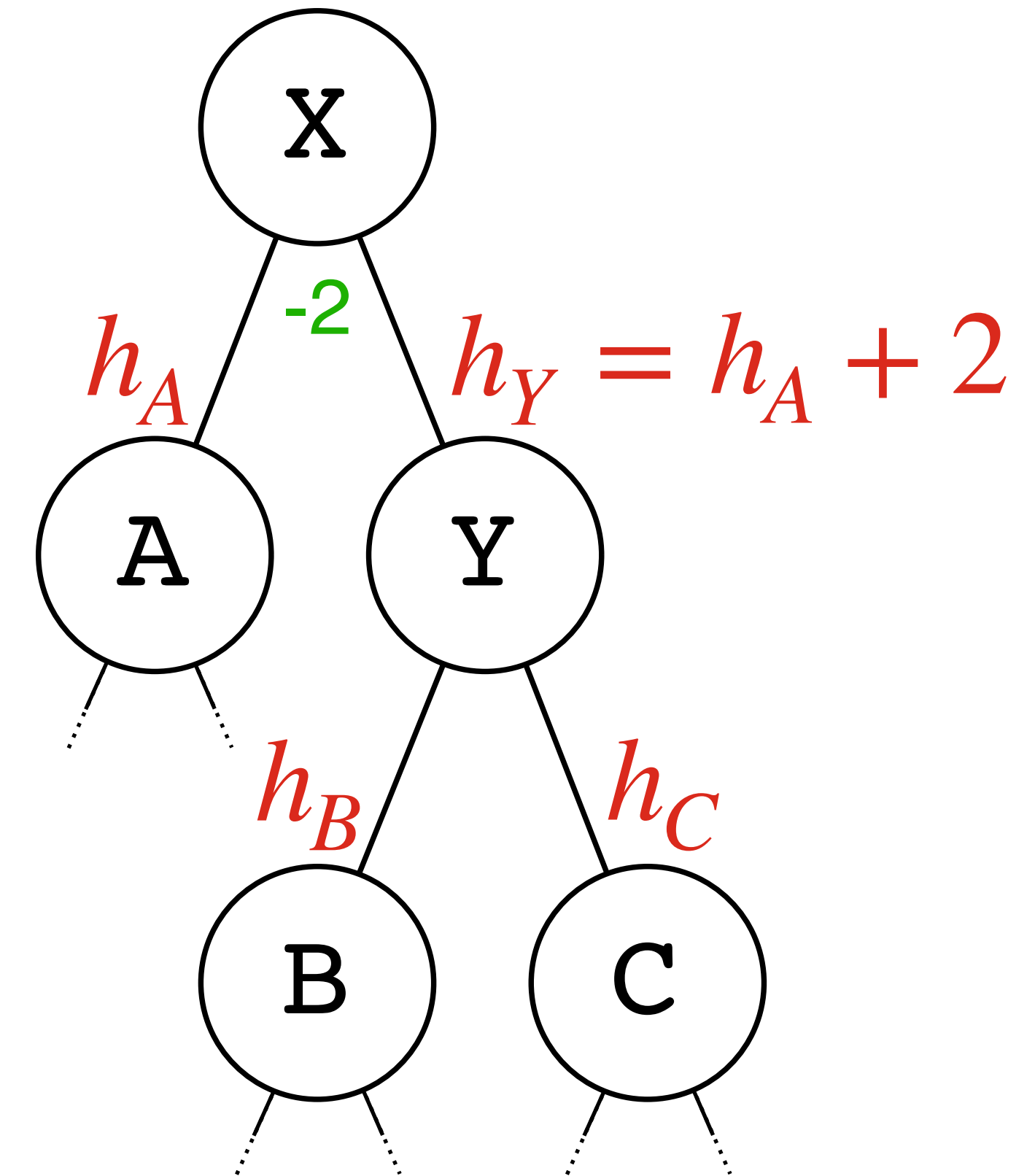
Effet de la rotation



On sait aussi que $h_Y = 1 + \max(h_B, h_C)$
et parce que Y est équilibré,
 $\text{équilibre}(Y) = h_B - h_C \in \{-1, 0, 1\}$

On a donc trois possibilités :

$h_B = h_A$	$h_B = h_A + 1$	$h_B = h_A + 1$
$h_C = h_A + 1$	$h_C = h_A + 1$	$h_C = h_A$



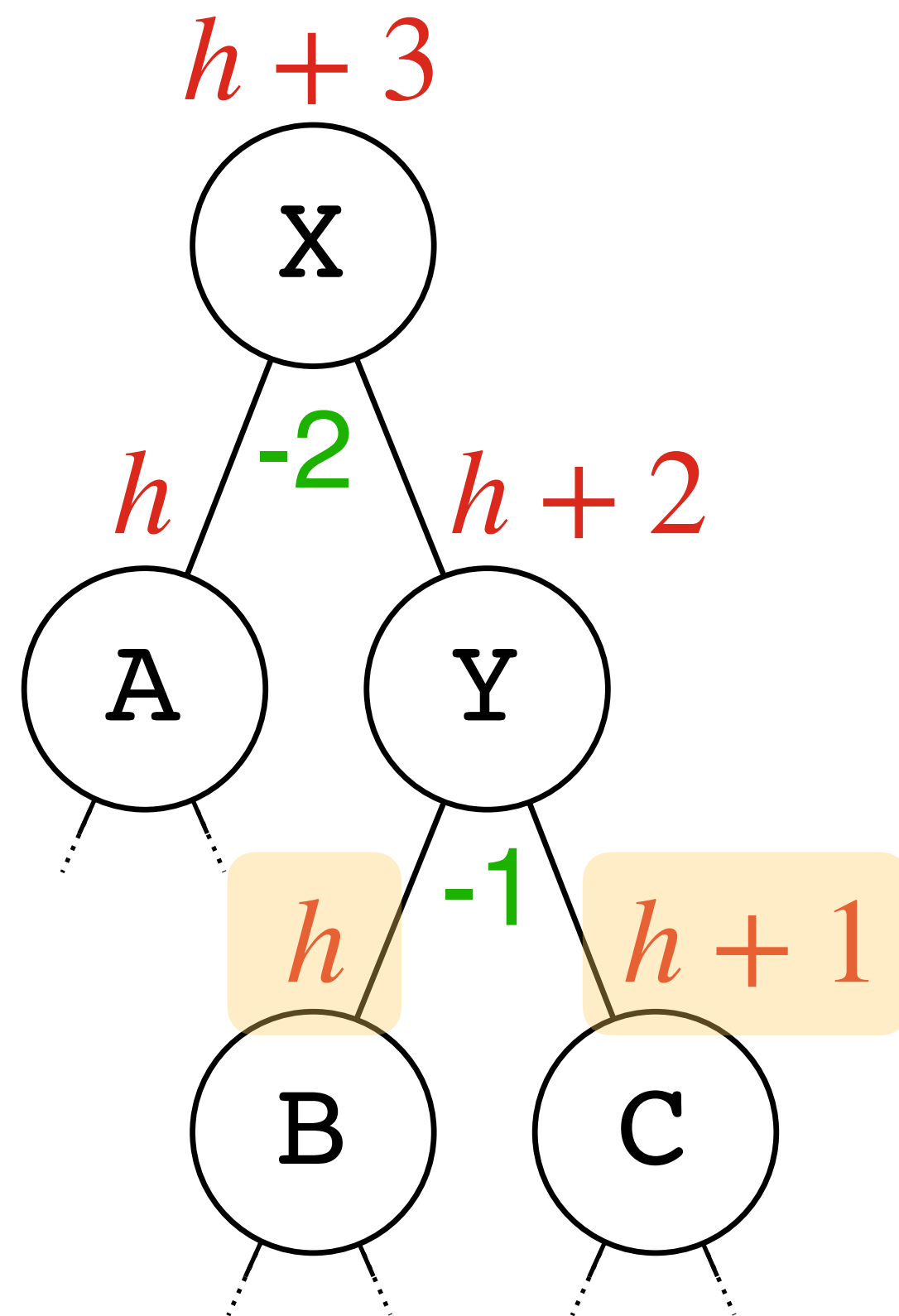
Effet de la rotation



On a donc trois possibilités :

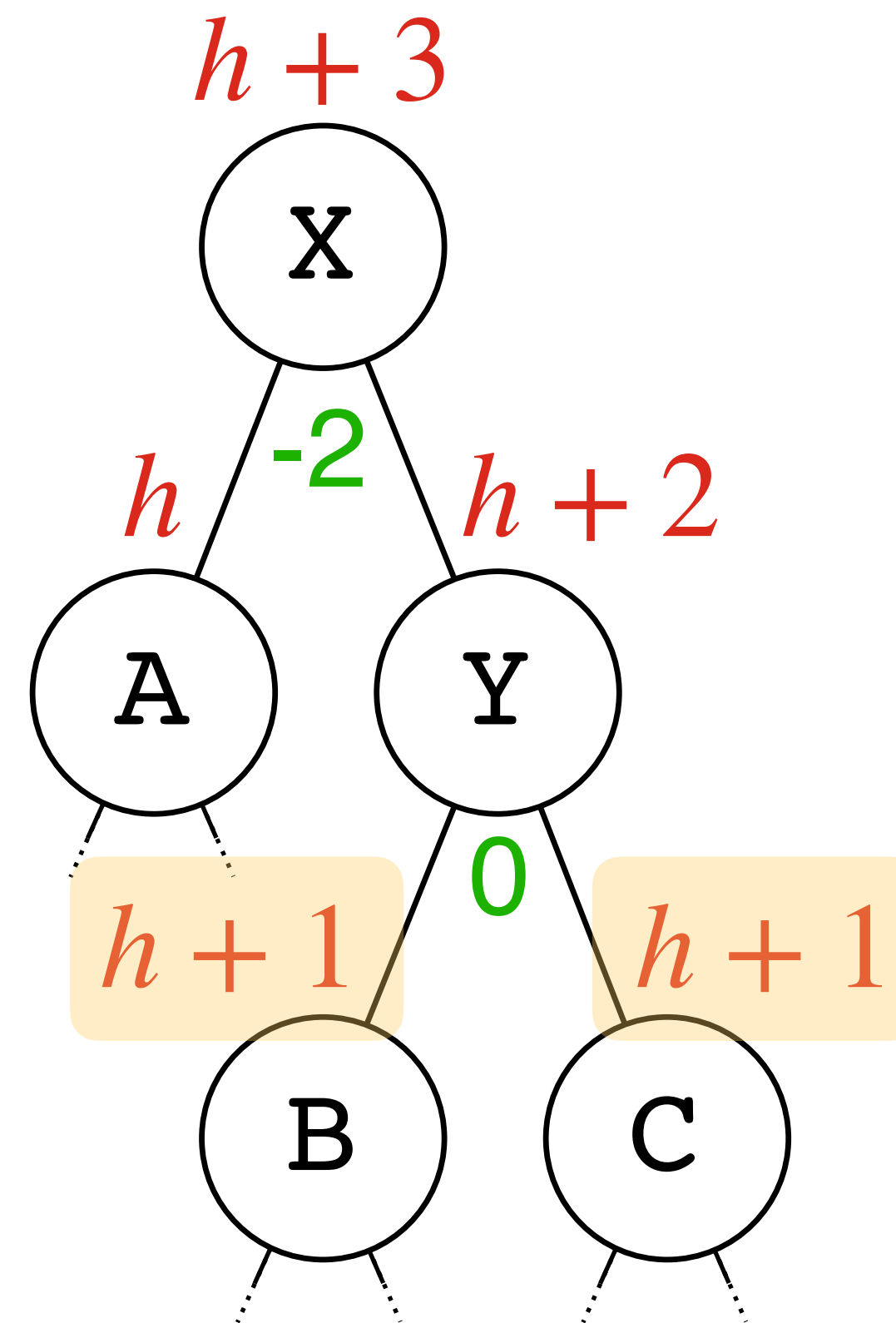
$$h_B = h_A$$

$$h_C = h_A + 1$$



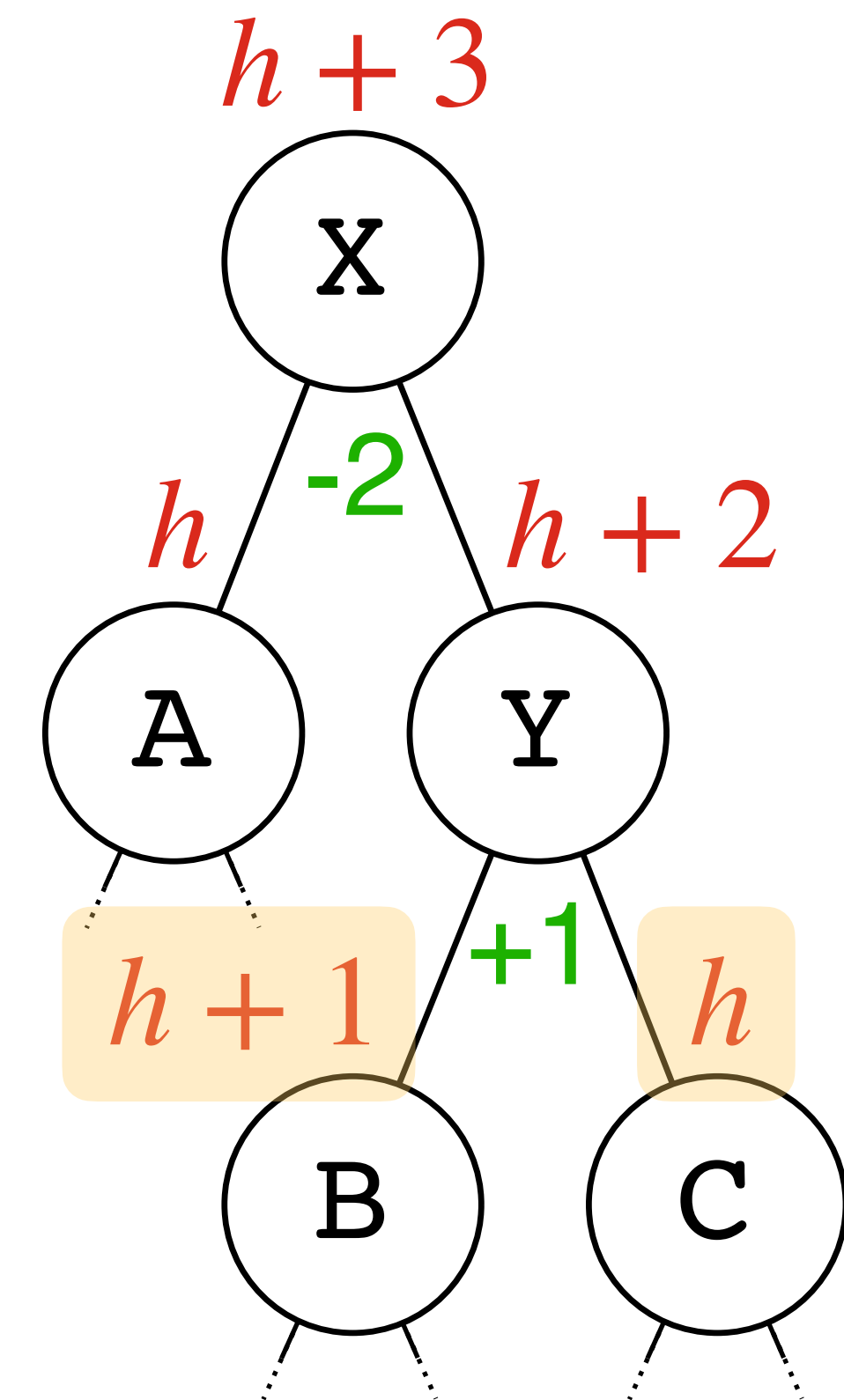
$$h_B = h_A + 1$$

$$h_C = h_A + 1$$



$$h_B = h_A + 1$$

$$h_C = h_A$$



Cas 1: équilibre(Y) = -1

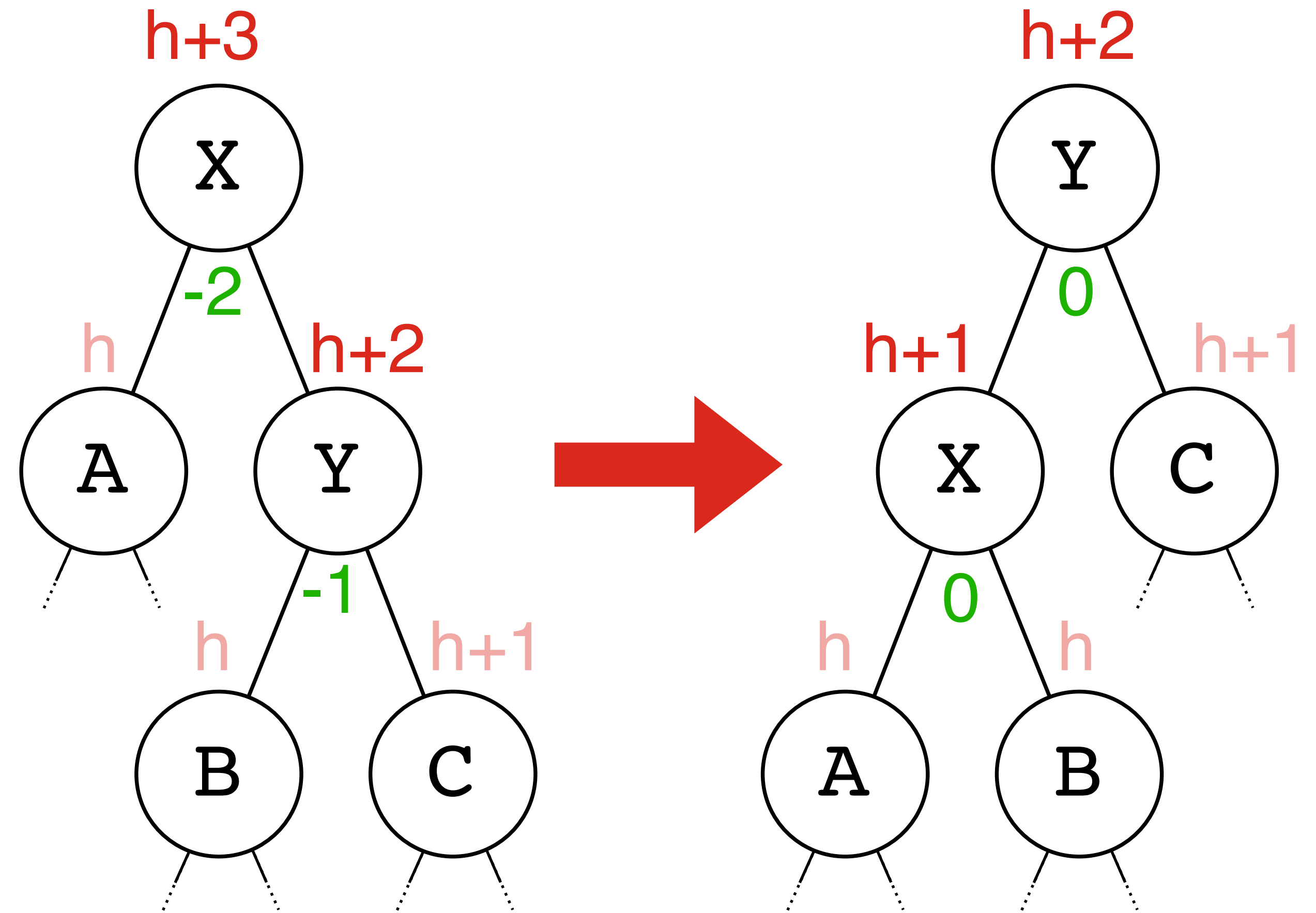


Après rotation,

- X et Y sont équilibrés
- La hauteur globale diminue de 1. Donc potentiel déséquilibre créé plus haut.

```

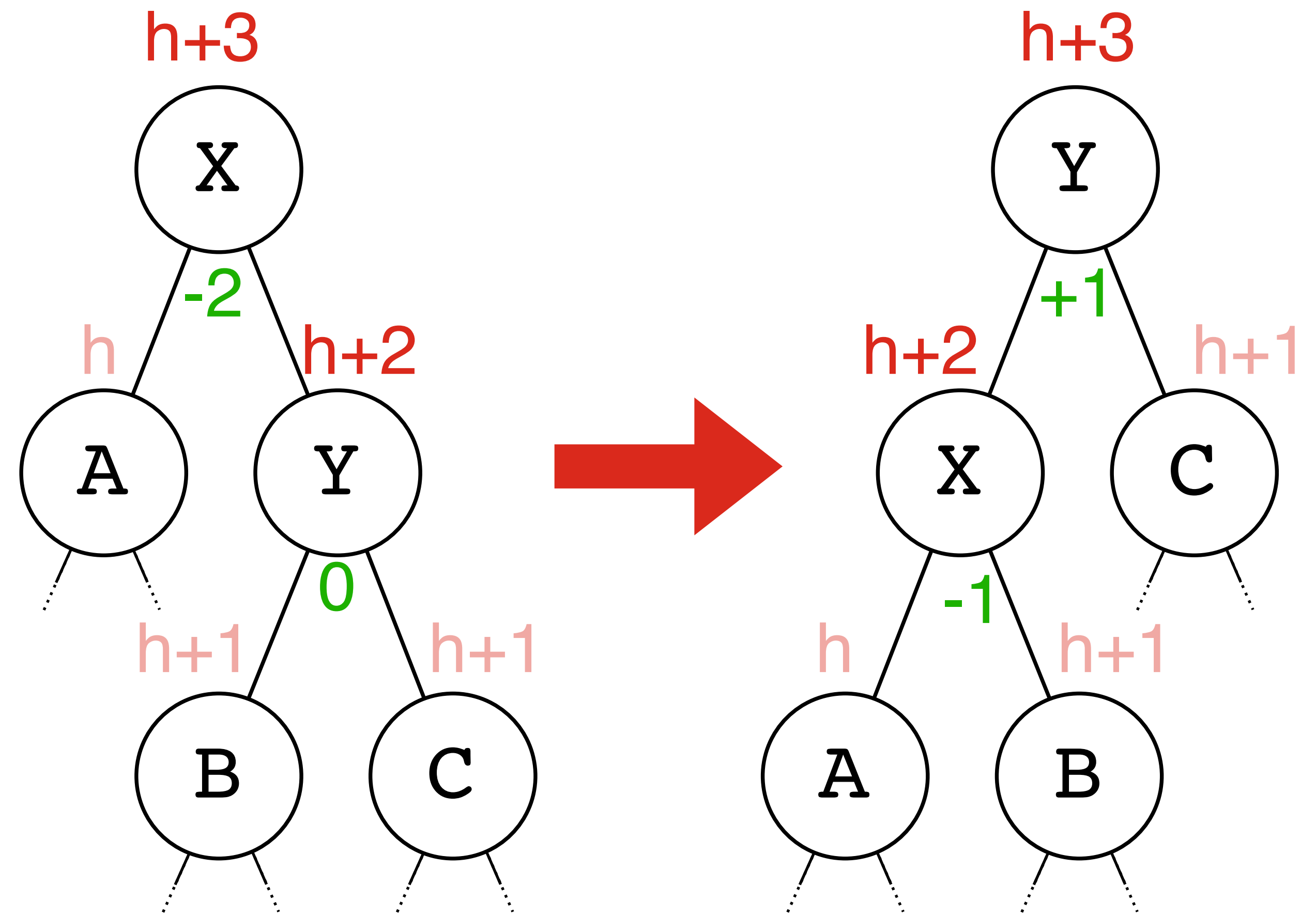
fonction rotation_gauche (ref r)
  t ← r.droit
  r.droit ← t.gauche
  t.gauche ← r
  r ← t
  calculer_hauteur(r.gauche)
  calculer_hauteur(r)
  
```



Cas 2: $\text{équilibre}(Y) = 0$ 

Après rotation,

- Les équilibres de X et Y sont -1 ou $+1$.
- La hauteur ne change pas. Pas besoin de rééquilibrer plus haut dans l'arbre.

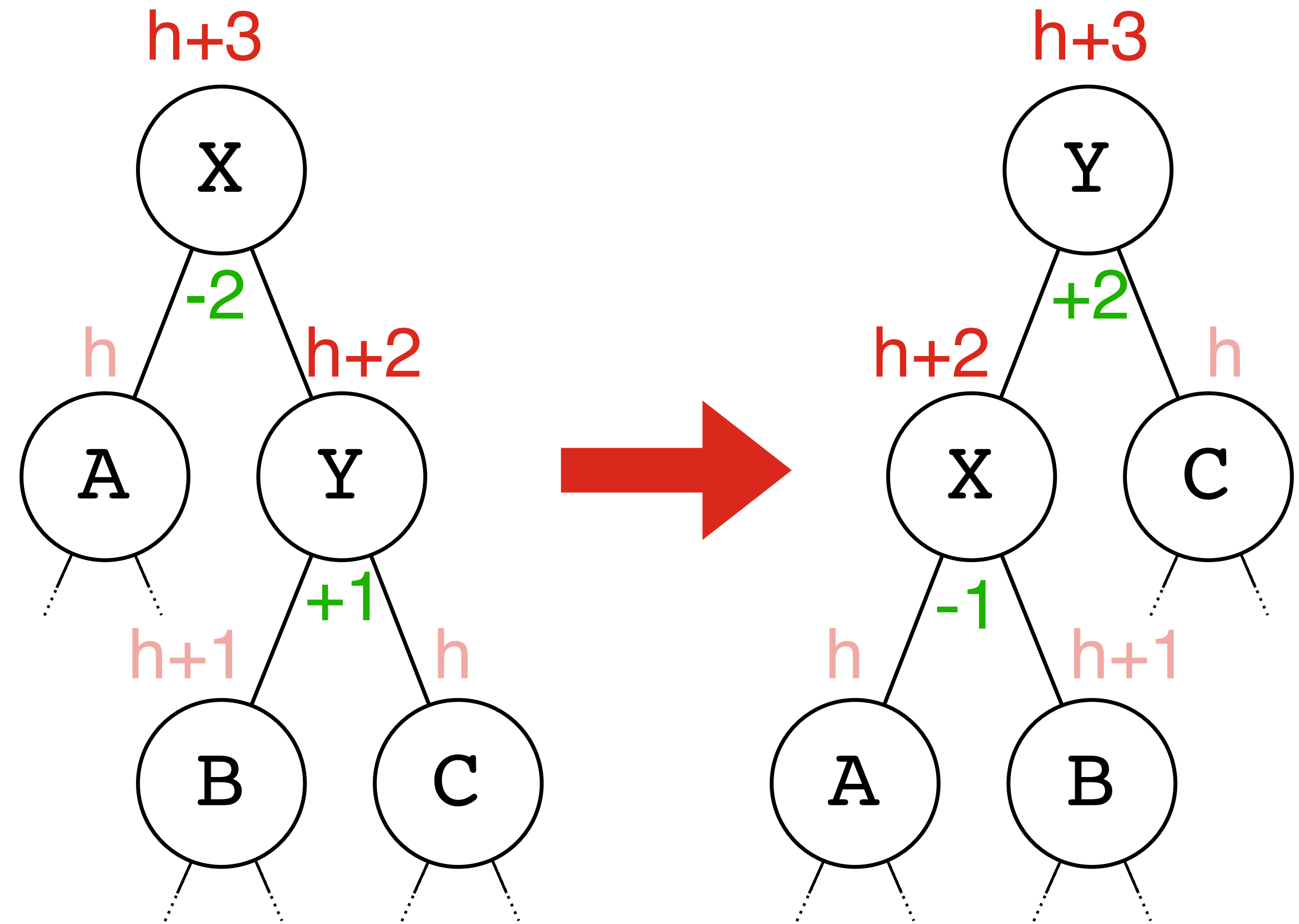


Cas 3: $\text{équilibre}(Y) = +1$



Après rotation,

- $\text{équilibre}(Y) = +2$
- la rotation simple ne résout pas le problème



Cas 3: $\text{équilibre}(Y) = +1$

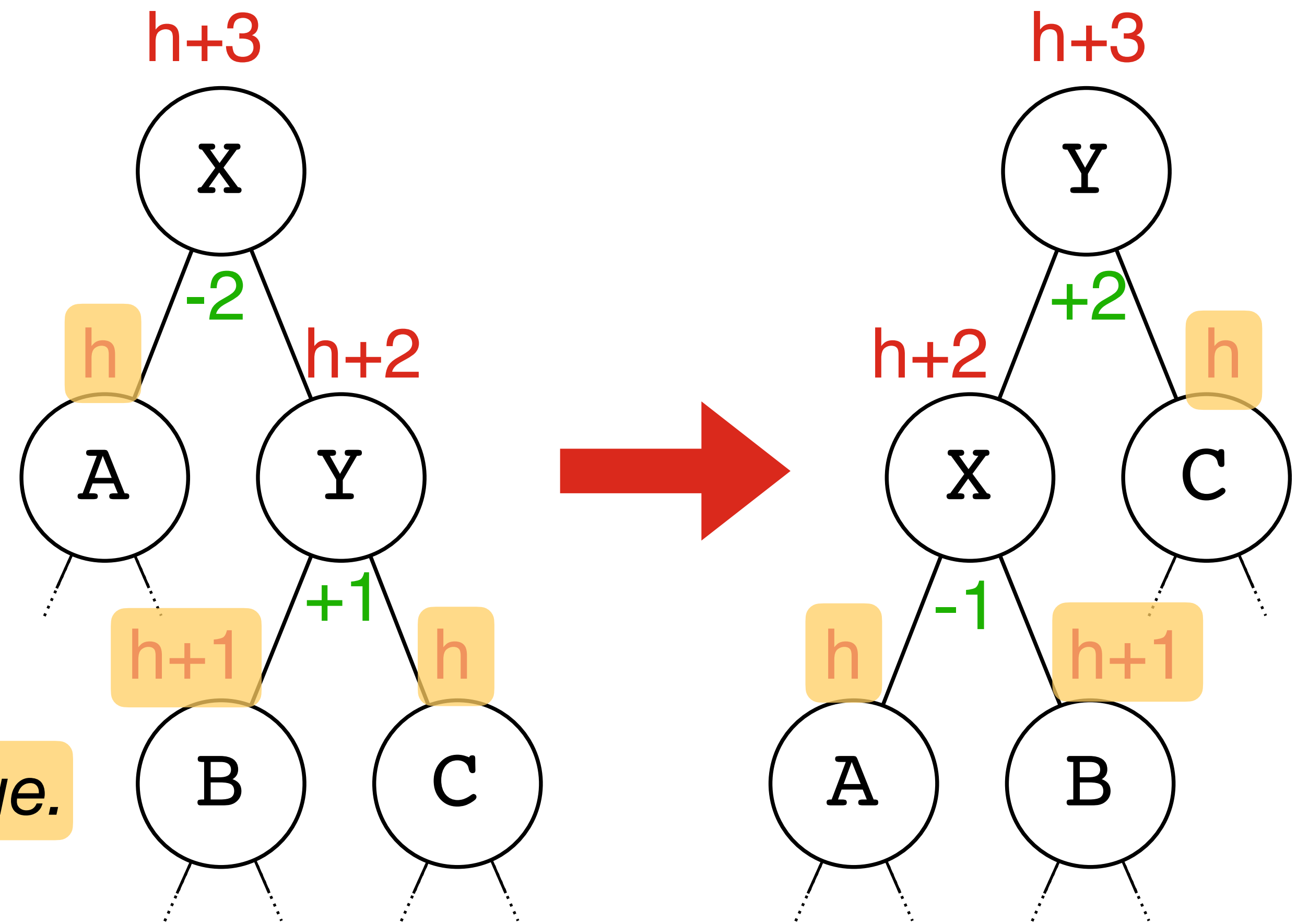


Après rotation,

- $\text{équilibre}(Y) = +2$
- la rotation simple ne résout pas le problème

Pourquoi ?

Parce que la situation est **symétrique**.

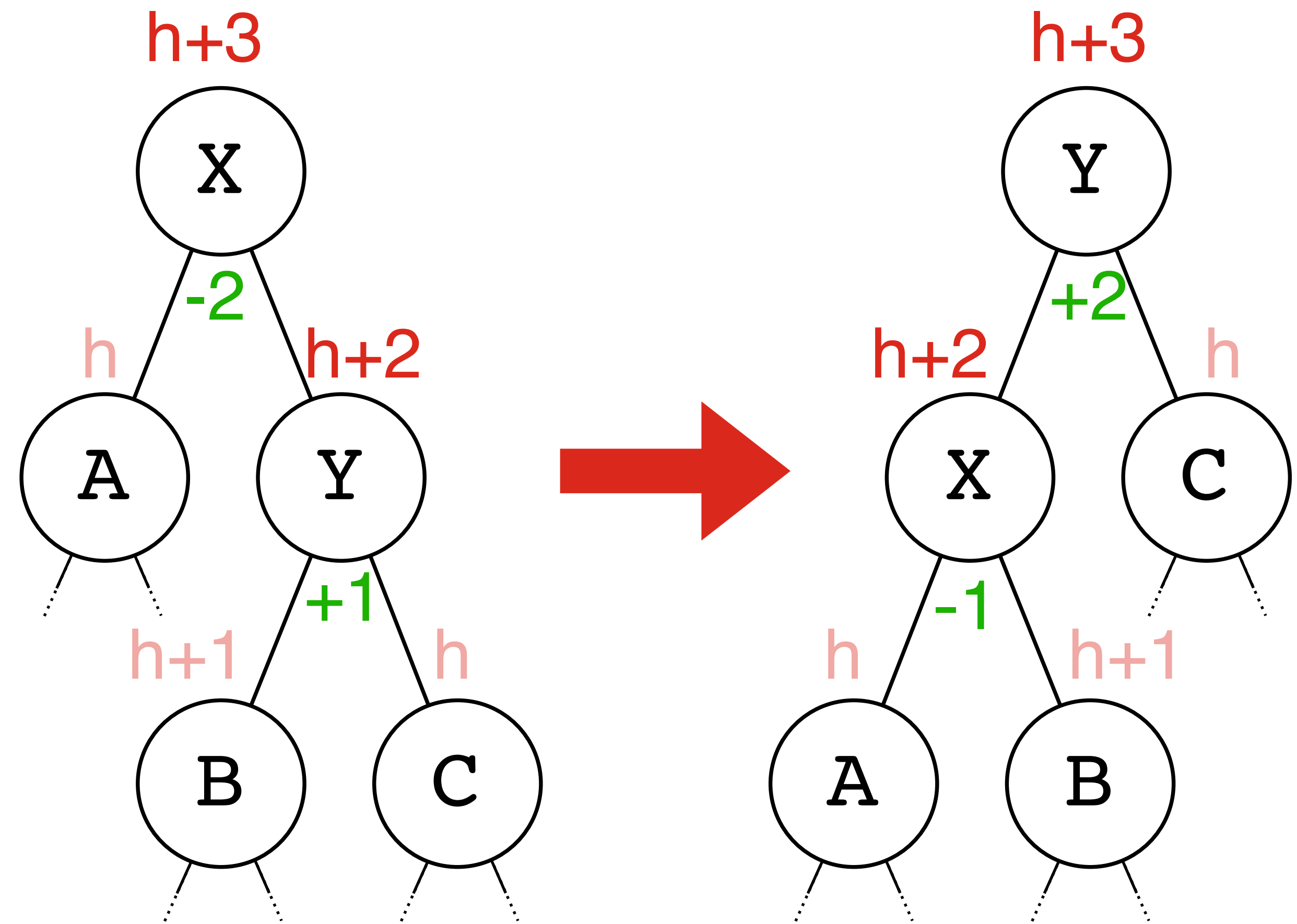


Cas 3: $\text{équilibre}(Y) = +1$

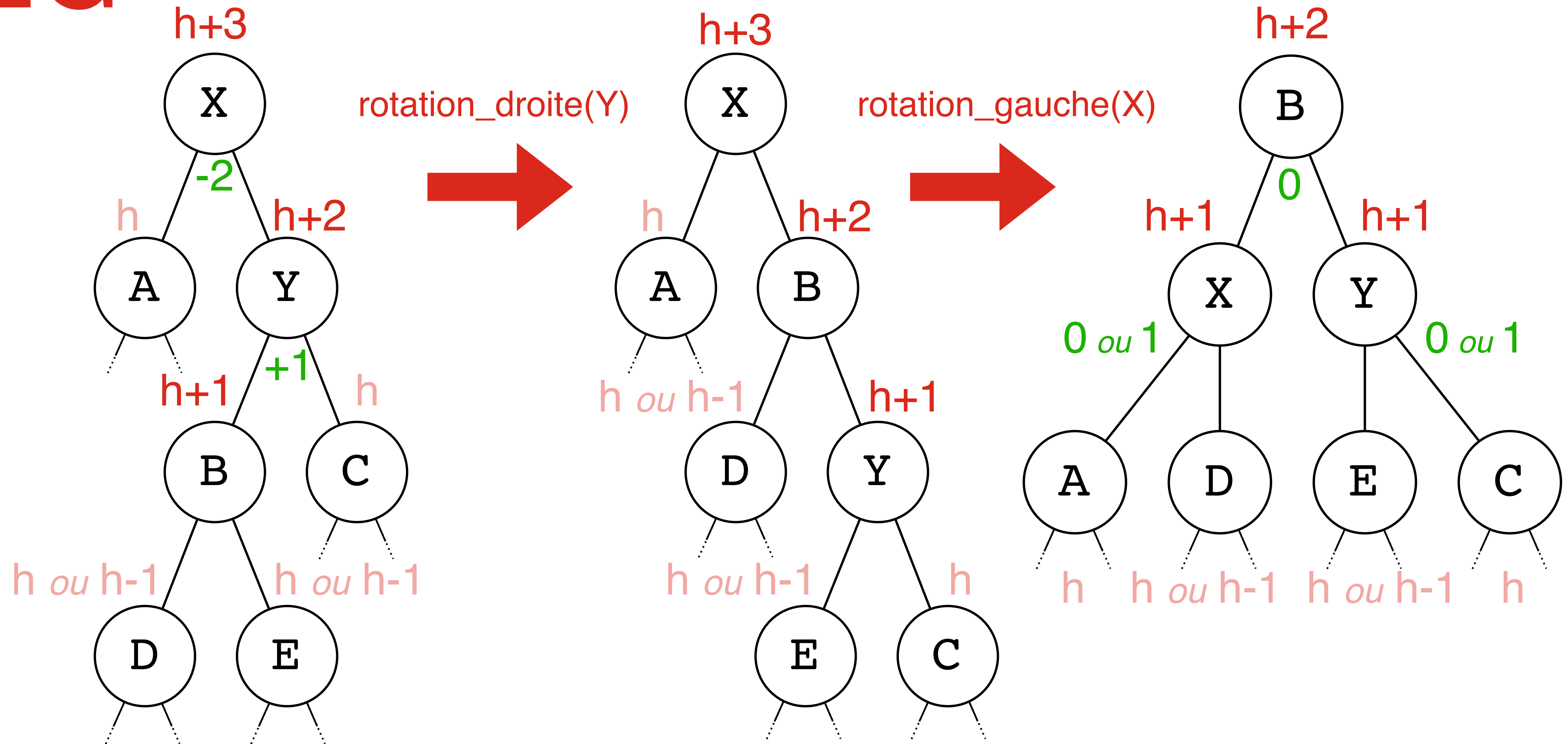


Après rotation,

- $\text{équilibre}(Y) = +2$
- la rotation simple ne résout pas le problème
- Solution : une rotation **droite** de **Y** avant la rotation **gauche** de **X**, dans ce cas 3 uniquement



Cas 3bis : double rotation

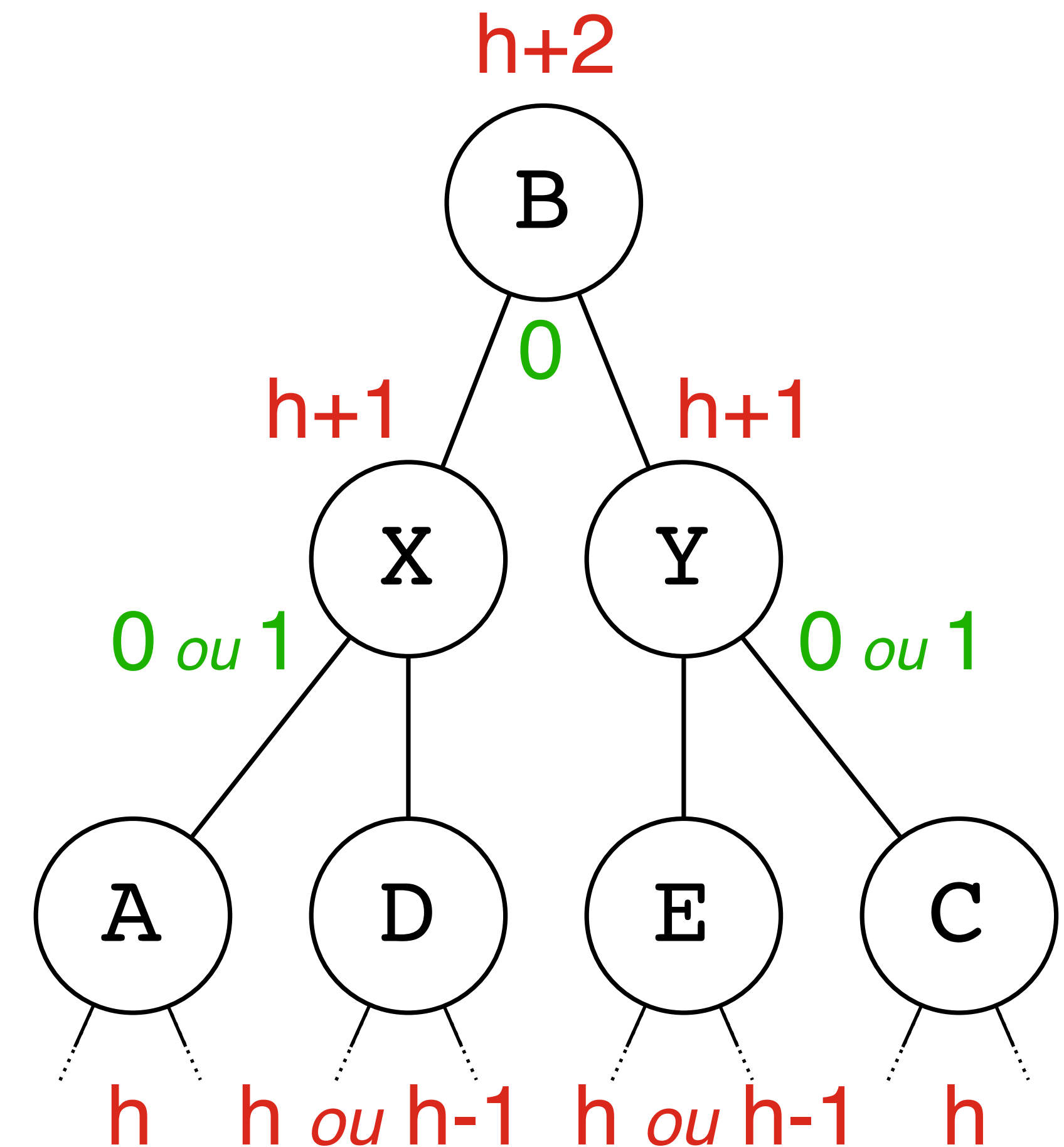


Cas 3bis : double rotation



Après double rotation...

- B , X , et Y sont équilibrés
- La hauteur globale a diminué de 1



En résumé, pour équilibrer



- Si le noeud r penche à droite ($\text{équilibre}(r) = -2$),
 - si $r.\text{droit}$ est équilibré ou penche à droite (cas 1 et 2), une rotation gauche suffit
 - si $r.\text{droit}$ penche à gauche (cas 3), il faut effectuer une rotation droite de $r.\text{droit}$ avant la rotation gauche de r

L'équilibrage conserve la hauteur de r si $r.\text{droit}$ était équilibré, la diminue de 1 sinon

- Si le noeud r penche à gauche, idem par symétrie

Algorithme d'équilibrage



```
fonction rétablir_équilibre (ref r)
    si r == ∅, retourner

    si équilibre(r) < -1,                // penche à droite
        si équilibre(r.droit) > 0,
            rotation_droite(r.droit)
        rotation_gauche(r)

    sinon, si équilibre(r) > 1,          // penche à gauche
        si équilibre(r.gauche) < 0,
            rotation_gauche(r.gauche)
        rotation_droite(r)

    sinon, calculer_hauteur(r)
```

Avec pour rappel ...



```

fonction hauteur (r)
    si r == ∅, retourner 0
    sinon,      retourner r.hauteur
    
```

```

fonction calculer_hauteur (r)
    si r != ∅,
        r.hauteur ← 1 +
        max(hauteur(r.gauche), hauteur(r.droit))
    
```

```

fonction équilibre (r)
    si r == ∅, retourner 0
    sinon, retourner
        hauteur(r.gauche) - hauteur(r.droit)
    
```

```

fonction rotation_gauche (ref r)
    t ← r.droit
    r.droit ← t.gauche
    t.gauche ← r
    r ← t
    calculer_hauteur(r.gauche)
    calculer_hauteur(r)
    
```

```

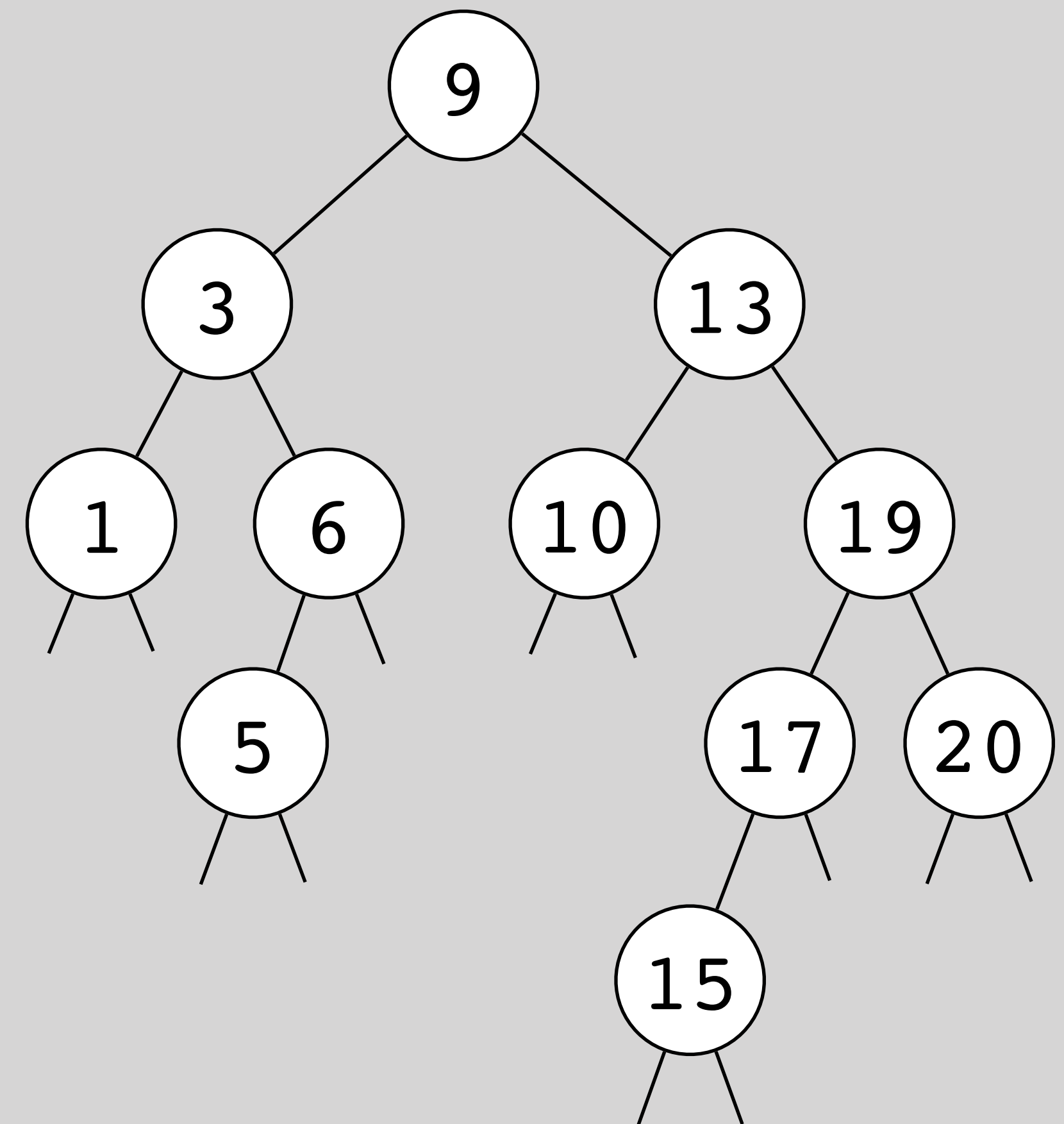
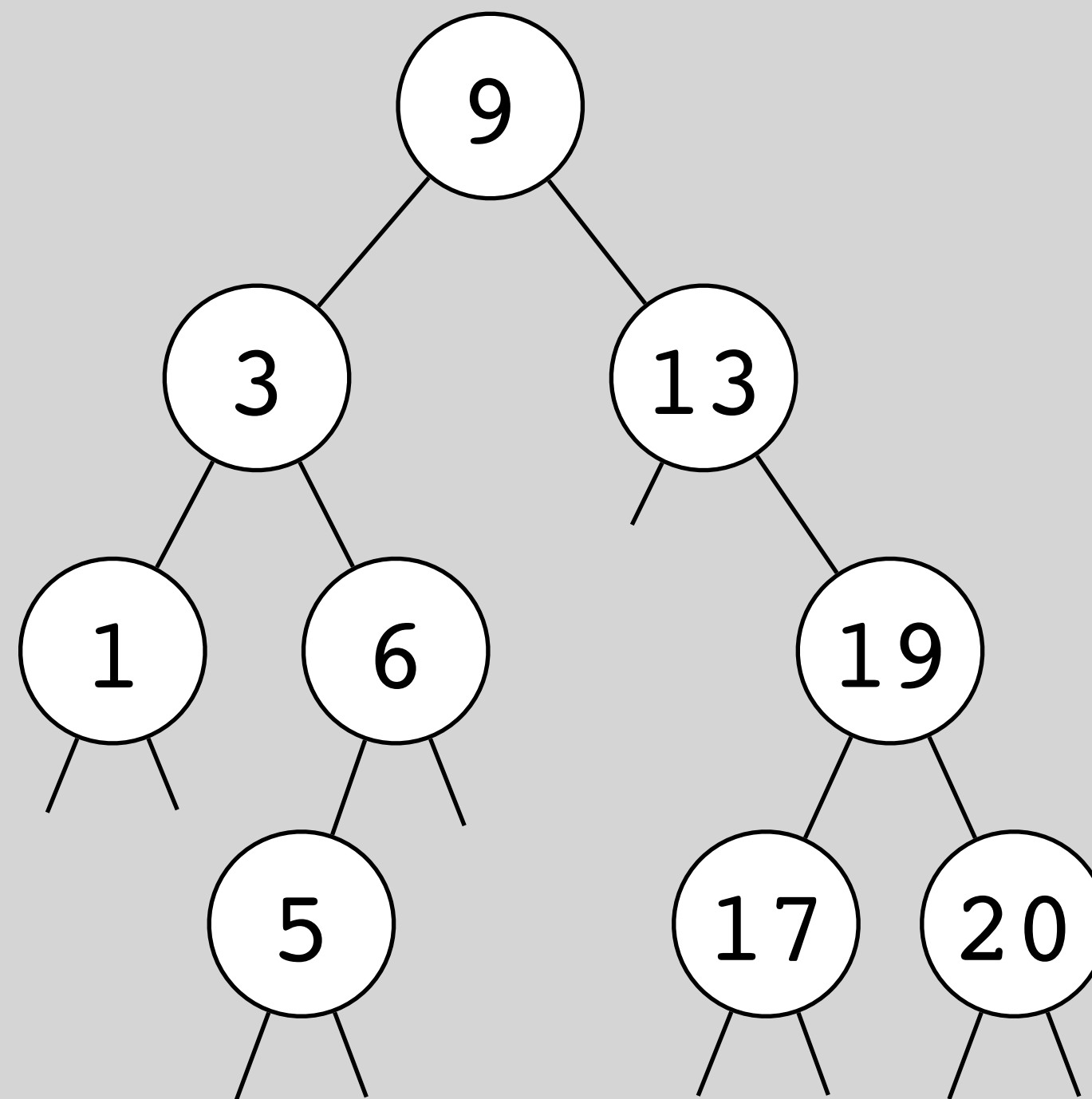
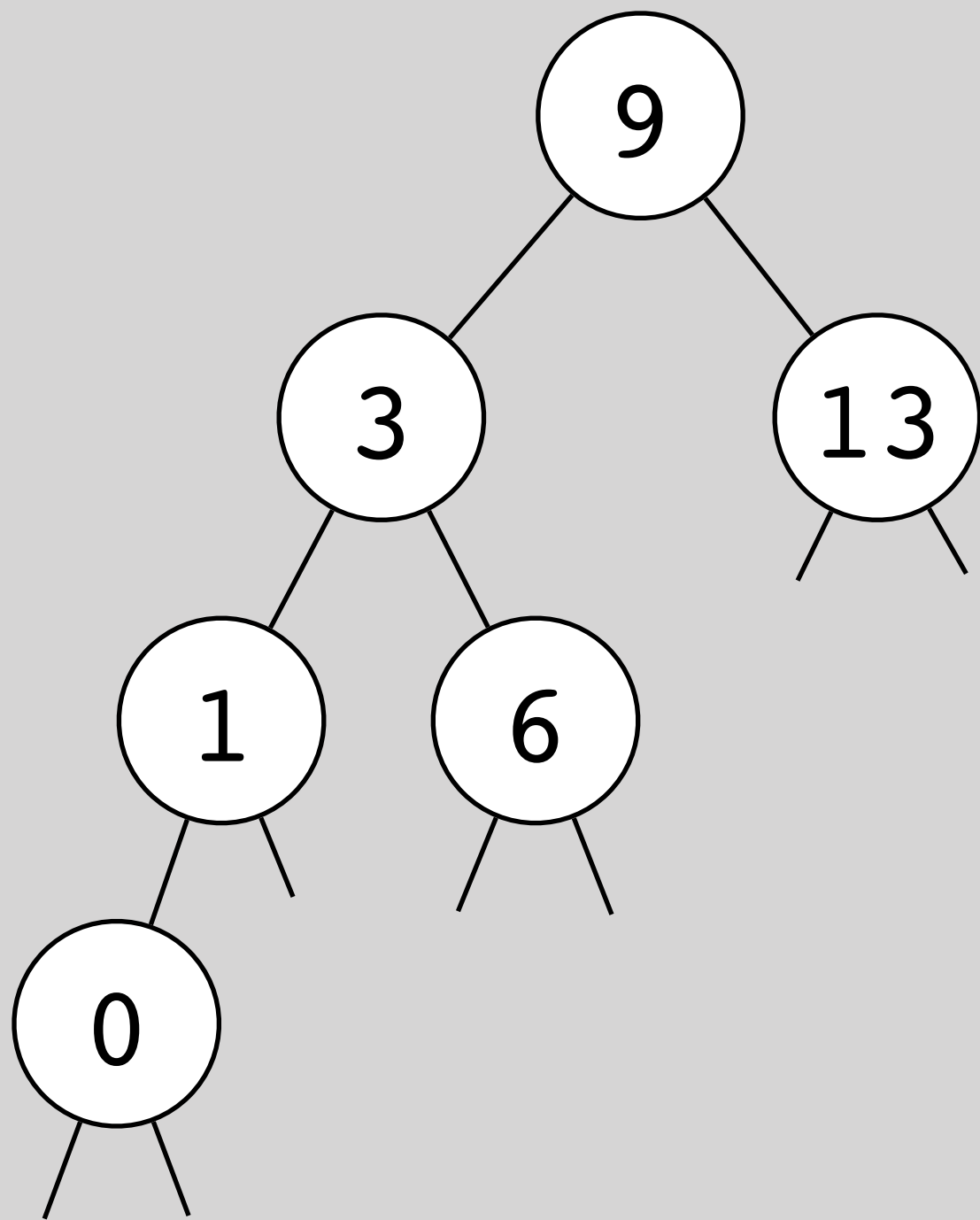
fonction rotation_droite (ref r)
    t ← r.gauche
    r.gauche ← t.droit
    t.droit ← r
    r ← t
    calculer_hauteur(r.droit)
    calculer_hauteur(r)
    
```

Exercice



Pour chaque arbre,

- Quel est le noeud déséquilibré ?
- Rééquilibrer l'arbre.

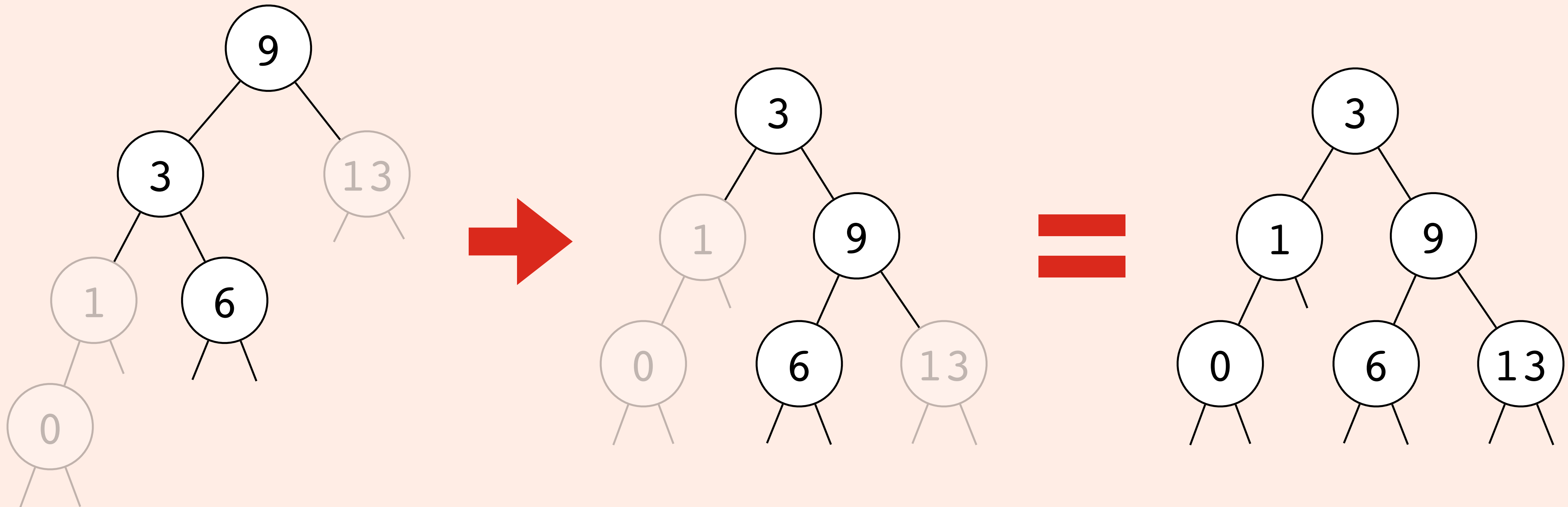


Solution



Pour chaque arbre,

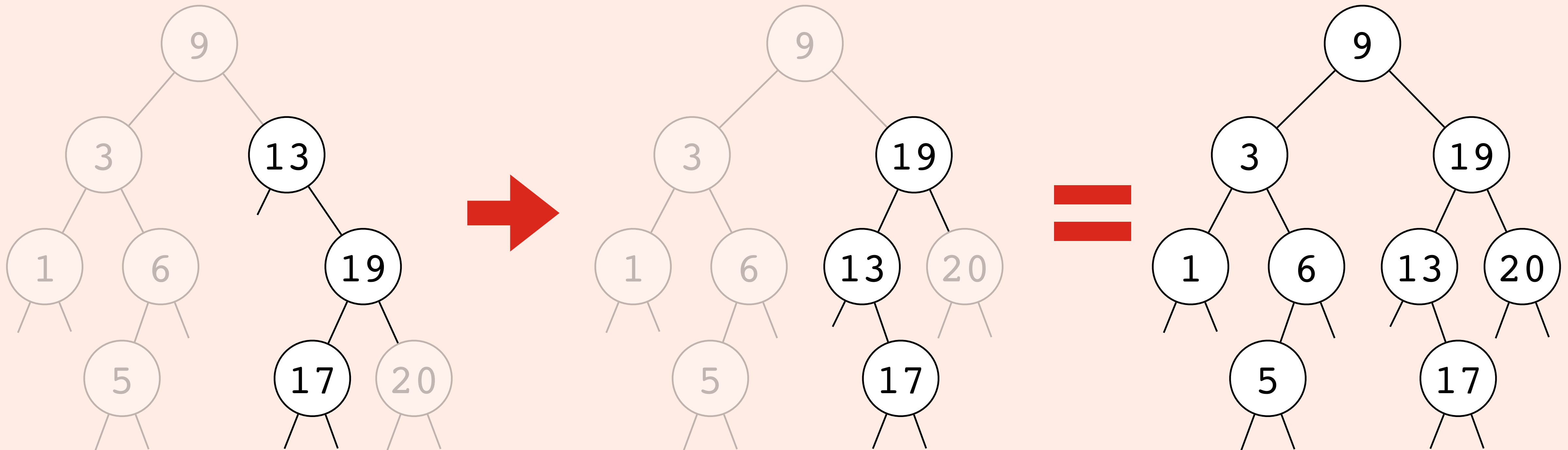
- Quel est le noeud déséquilibré ?
- 9 penche à gauche (équilibre de 2)
- Rééquilibrer l'arbre.





Pour chaque arbre,

- Quel est le noeud déséquilibré ?
- 13 penche à droite (équilibre de -2)
- Rééquilibrer l'arbre.

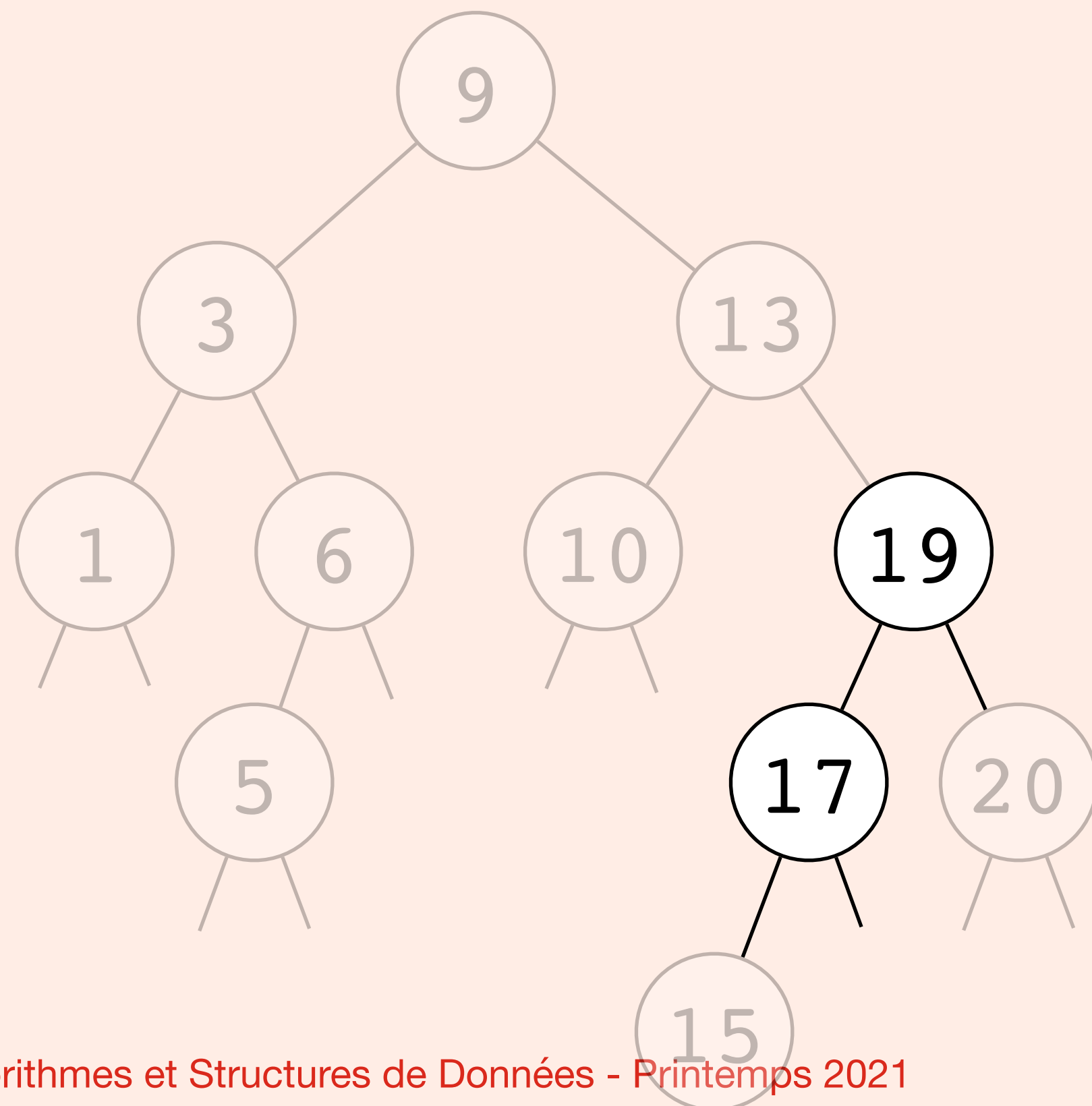


Solution

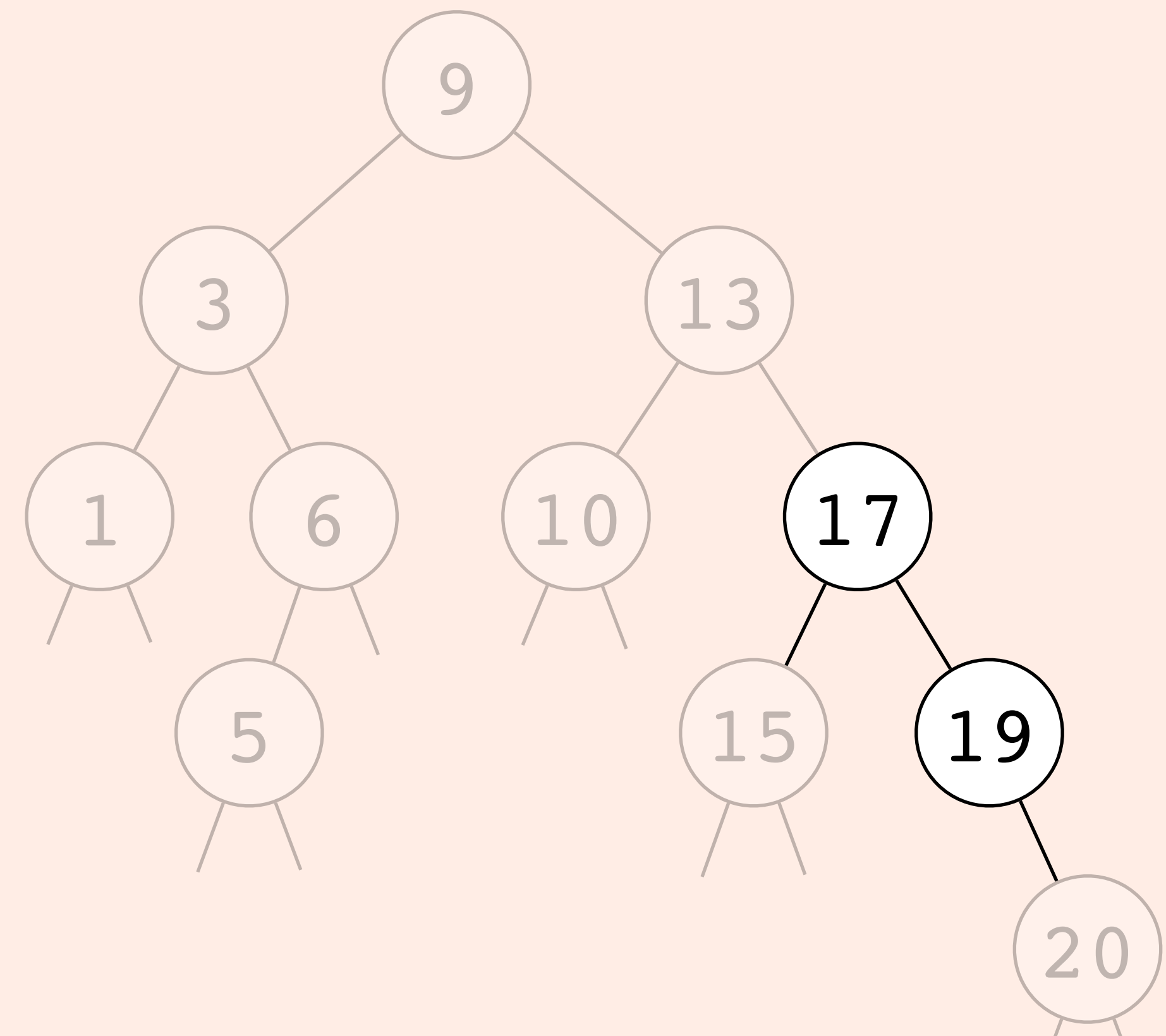
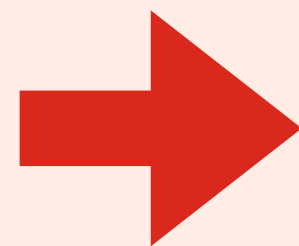


Pour chaque arbre,

- Quel est le noeud déséquilibré ?
- 13 penche à droite (équilibre de -2)
- Rééquilibrer l'arbre.
- Cas symétrique (cas 3) !



Première rotation
de l'enfant droit

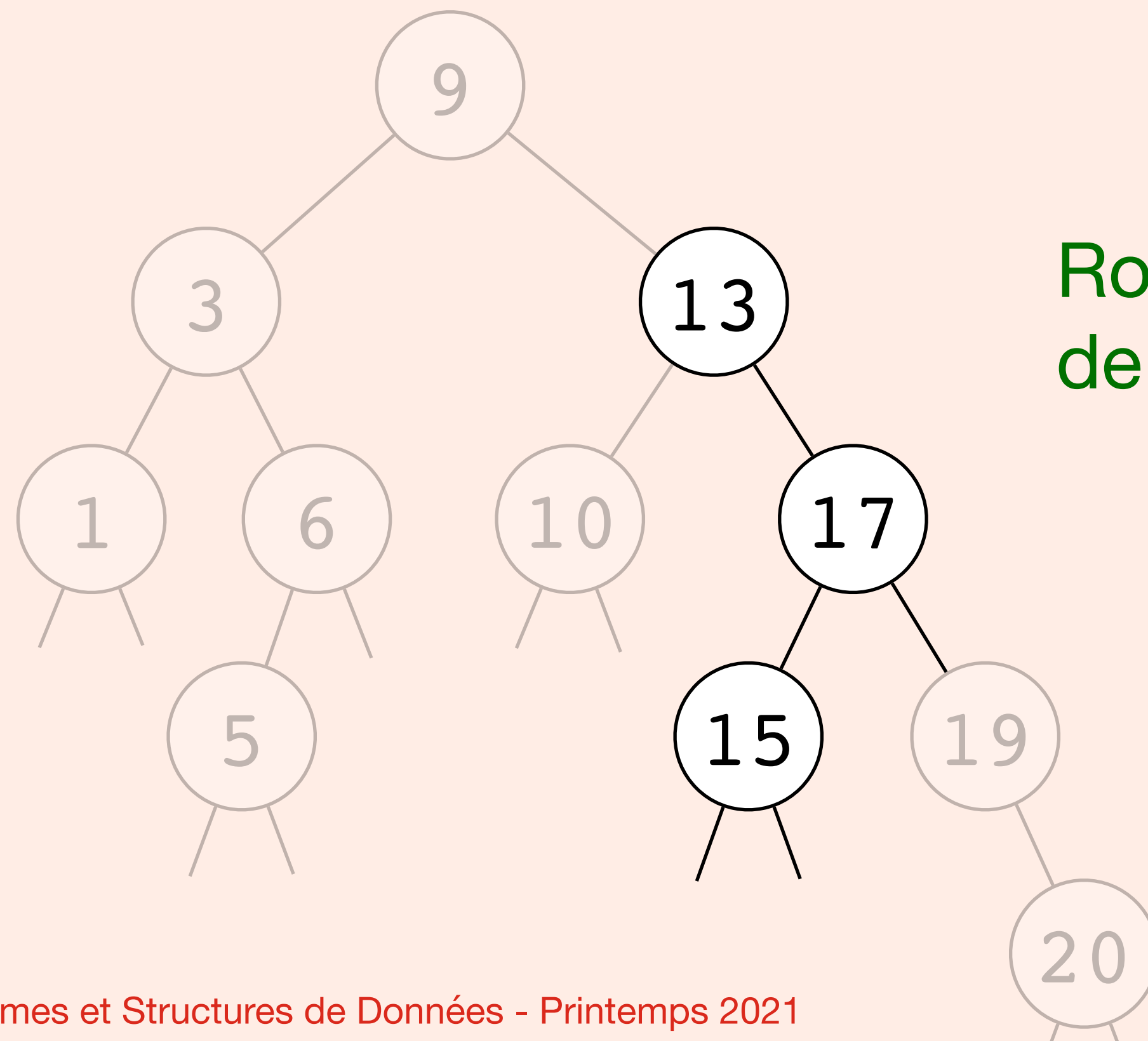


Solution (suite)

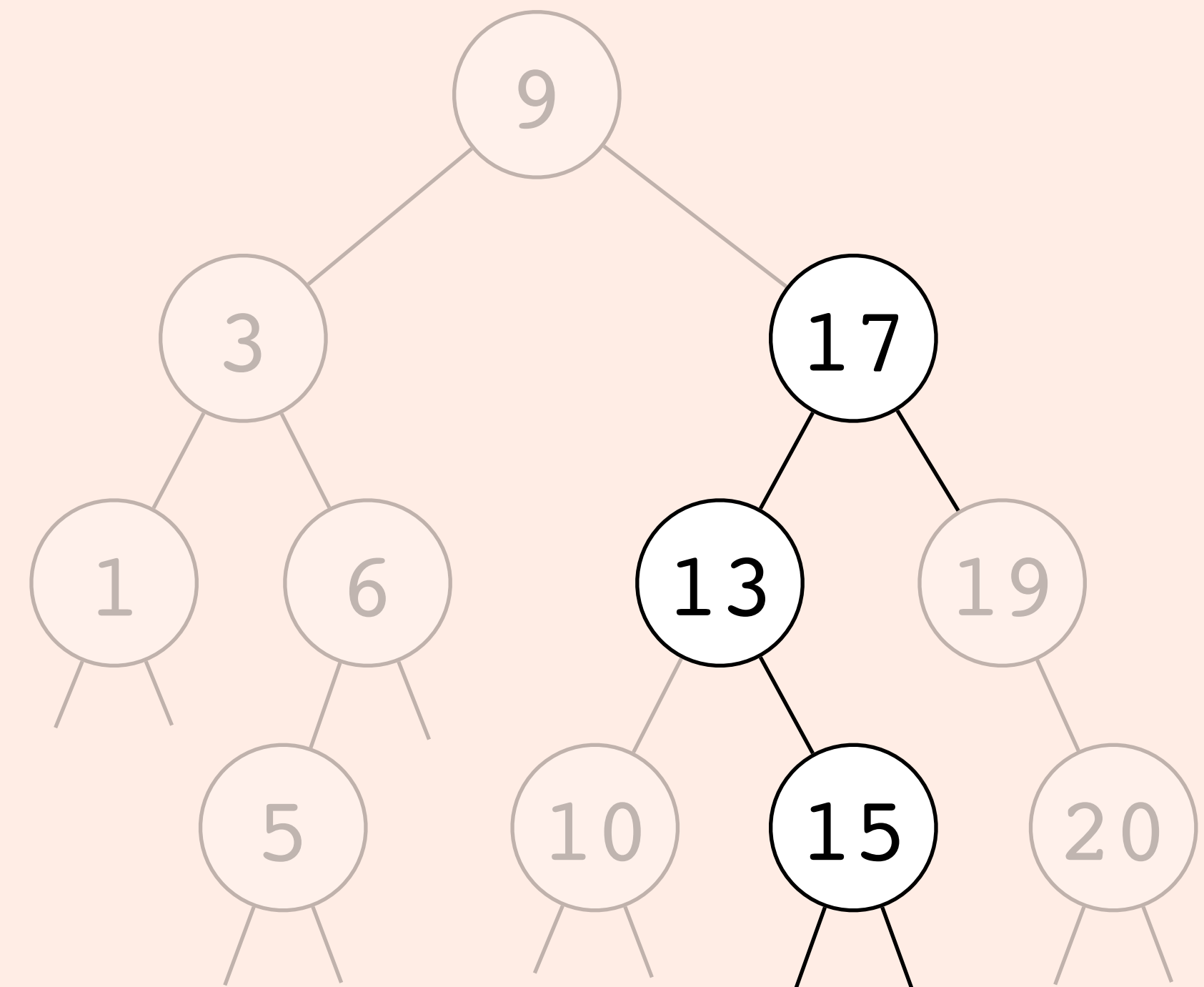
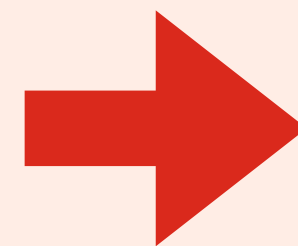


Pour chaque arbre,

- Quel est le noeud déséquilibré ?
- 13 penche à droite (équilibre de -2)
- Rééquilibrer l'arbre.
- Cas symétrique (cas 3) !



Rotation gauche
de 13





Le plus simple :

- code identique à celui de l'insertion dans un ABR
- avec appel à `rétablir_équilibre(r)` après un appel récursif

Plus efficace (*pas décrit ici*)

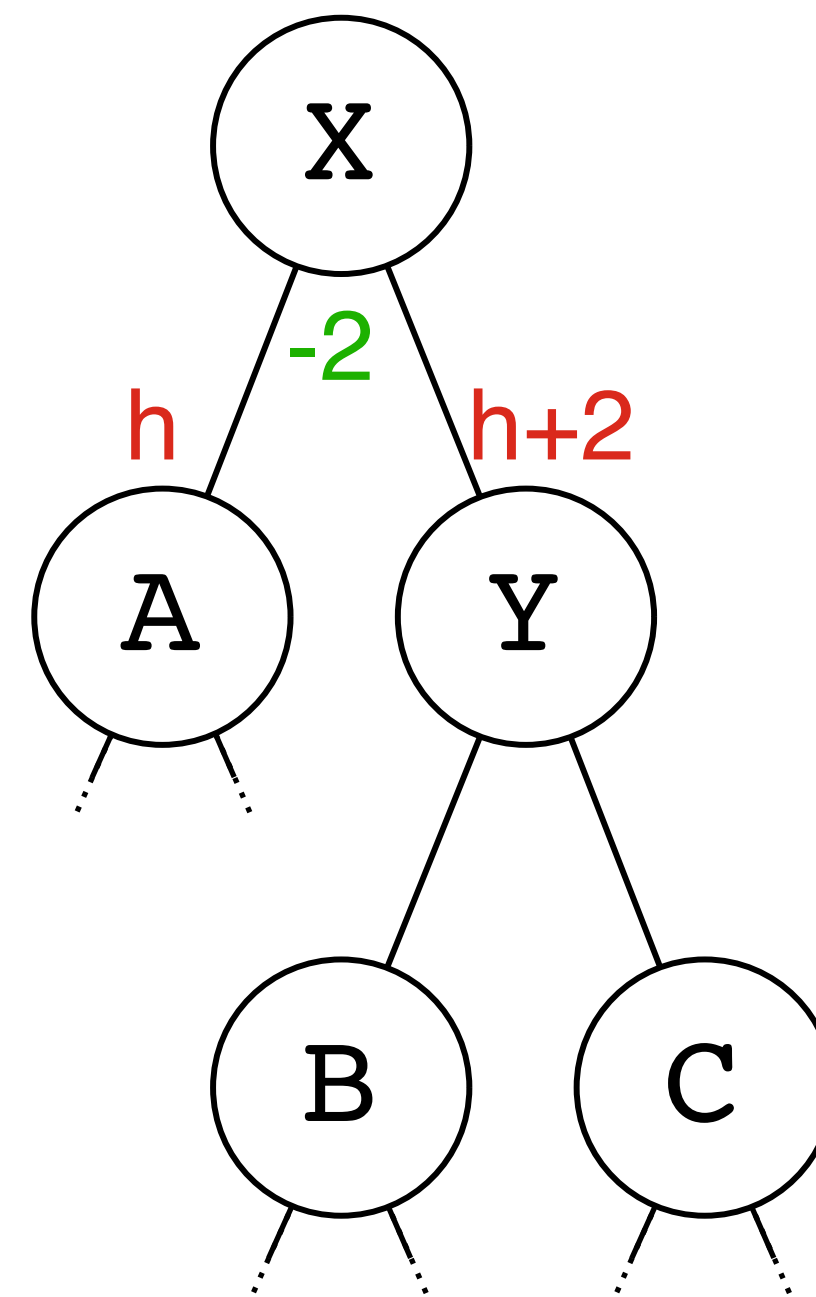
- Utiliser le fait qu'un seul **rétablissement** sera nécessaire
- Nécessite plus de communication entre noeud ou une approche itérative

```
fonction insérer (ref r, k)
    si r est ∅
        r ← nouveau noeud de clé k
    sinon, si k == r.clé
        k est déjà présent. L'action
        dépend du TDA mis en oeuvre
    sinon,
        si k < r.clé
            insérer(r.gauche, k)
        sinon // k > r.clé
            insérer(r.droite, k)
        rétablir_équilibre(r)
```

Pourquoi un seul rétablissement ?



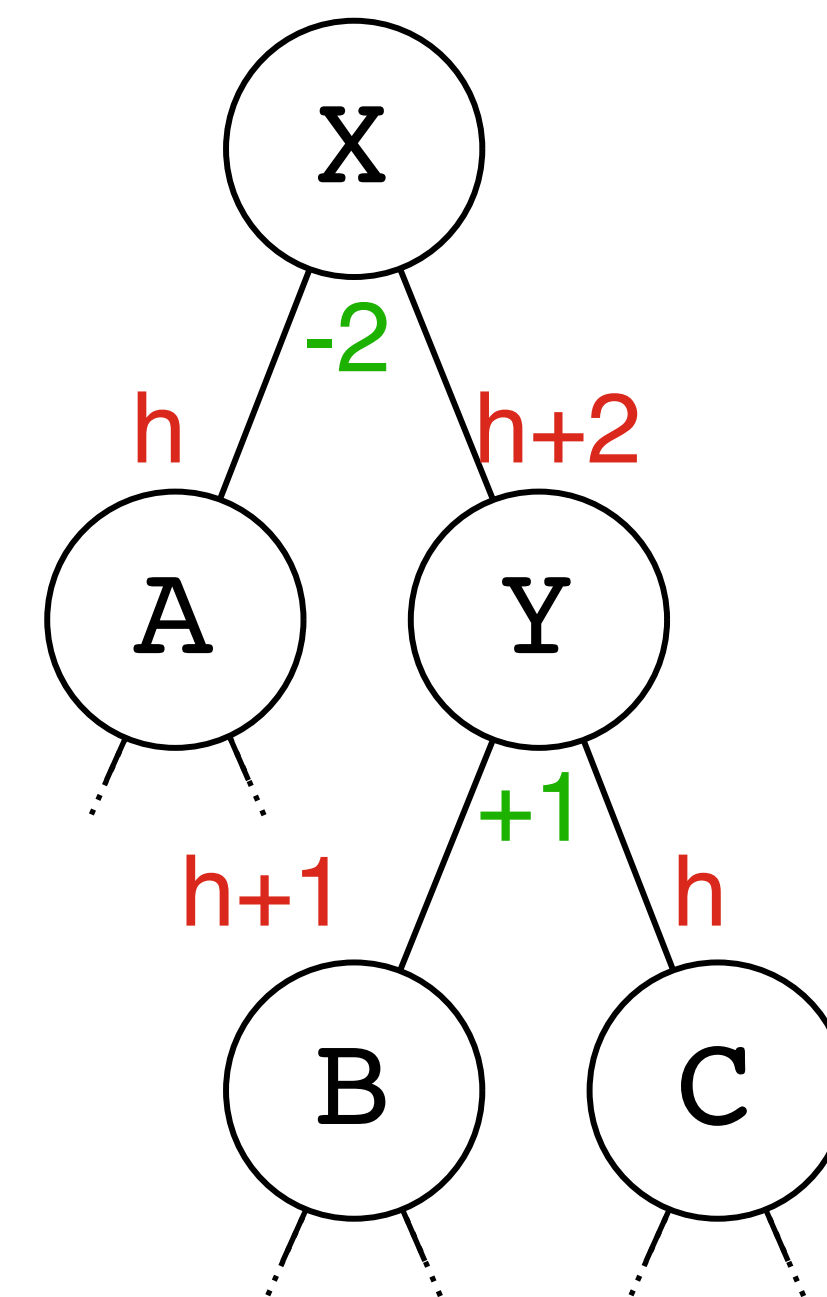
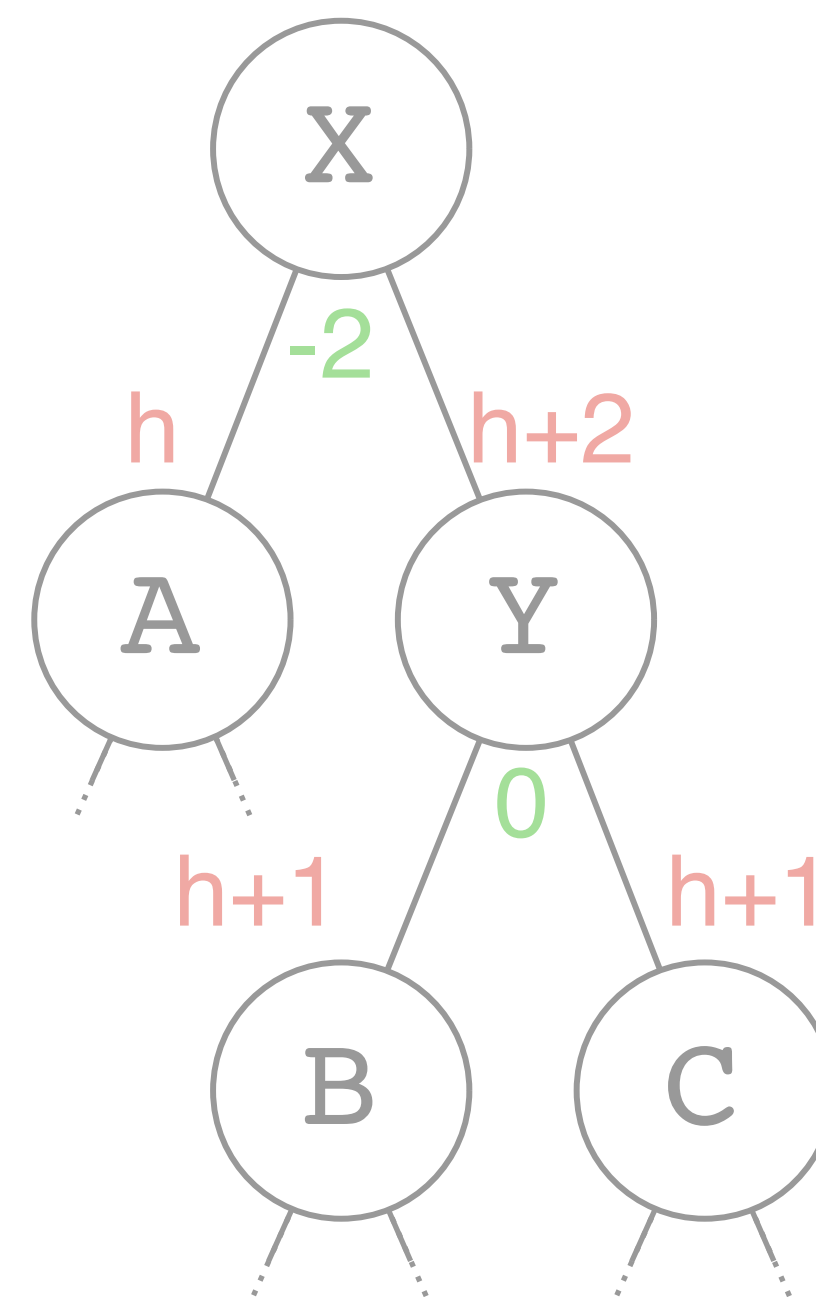
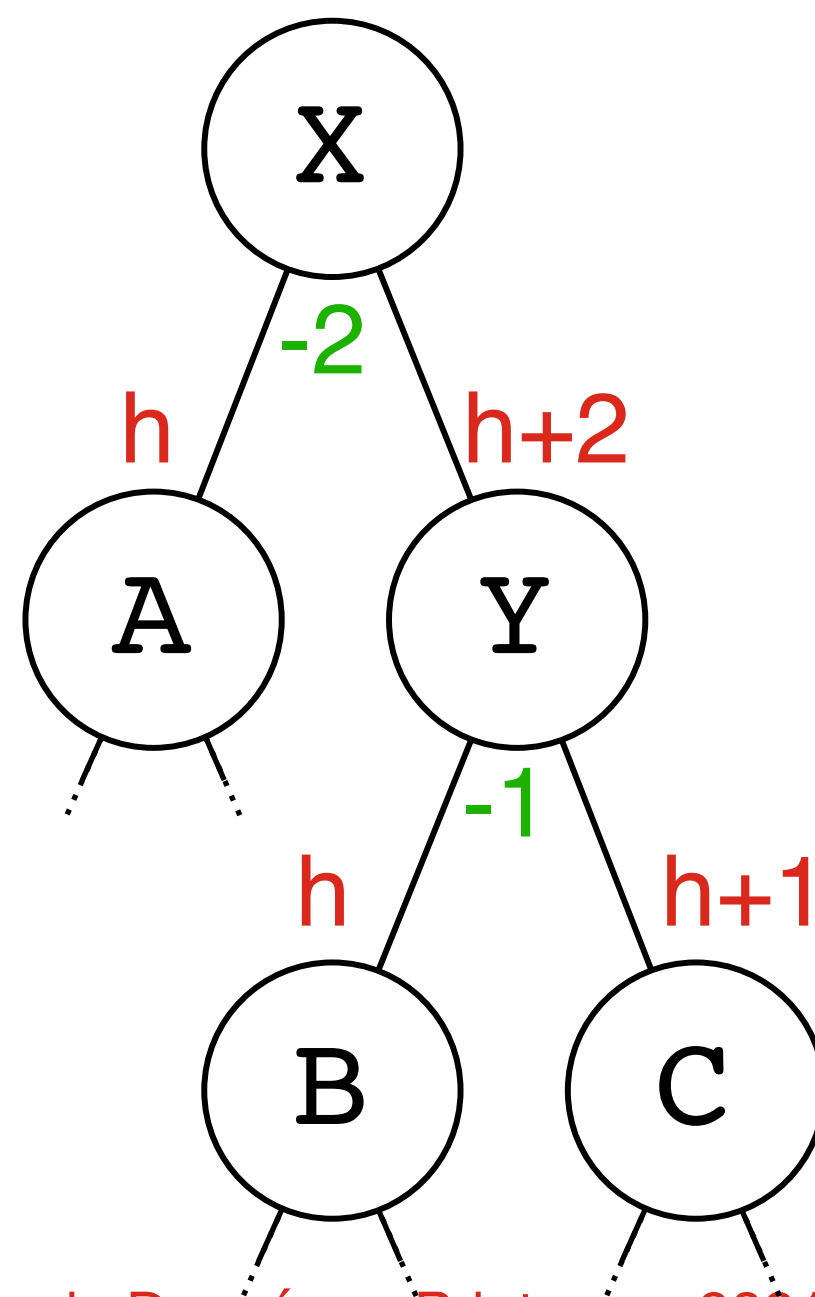
- Si le déséquilibre de X provient d'une insertion, elle a eu lieu dans les sous-arbres B ou C



Pourquoi un seul rétablissement ?



- Si le déséquilibre de X provient d'une insertion, elle a eu lieu dans les sous-arbres B ou C
Pourquoi ? *Parce que si elle avait eu lieu dans A, X aurait déjà été déséquilibré de -2 ou -3.*
- Seuls les cas 1 et 3 sont possibles.

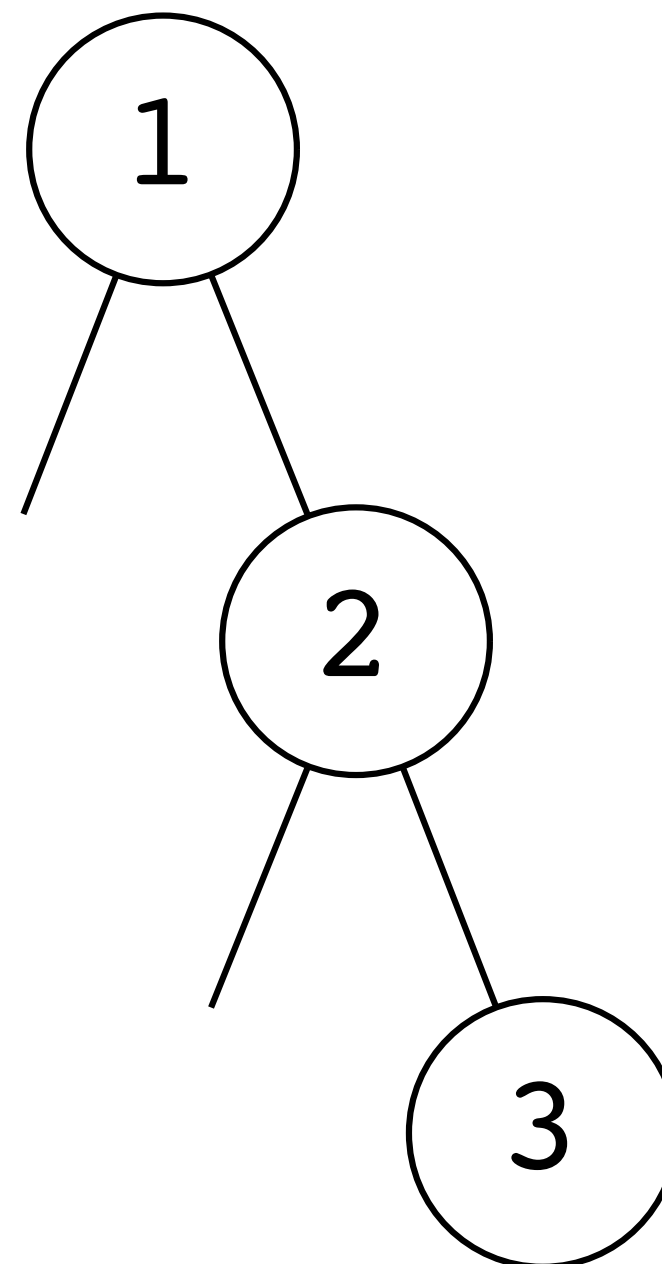
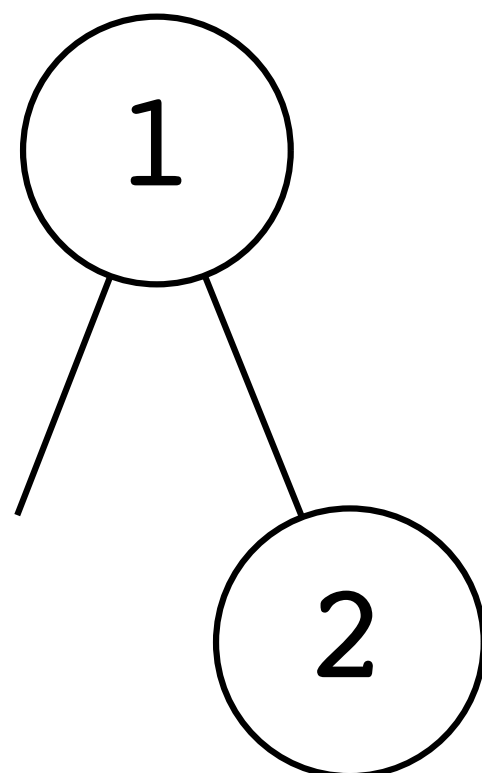
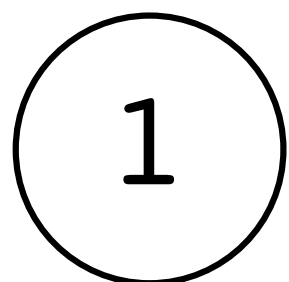


Pourquoi un seul rétablissement ?

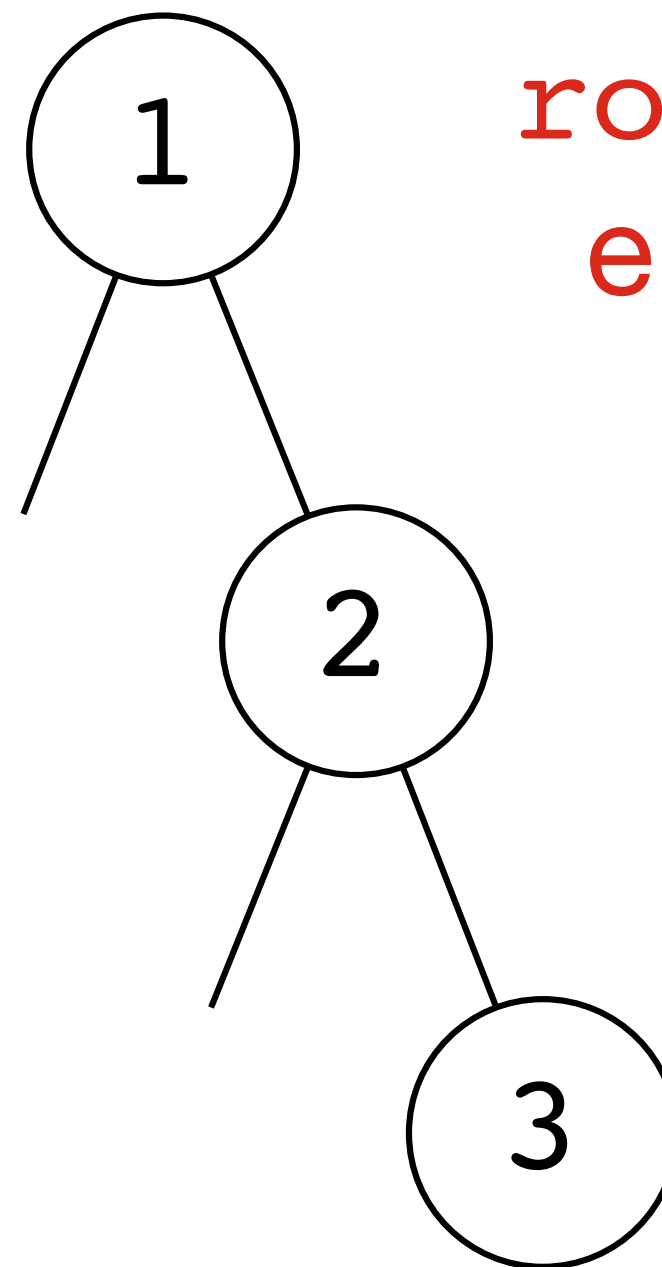
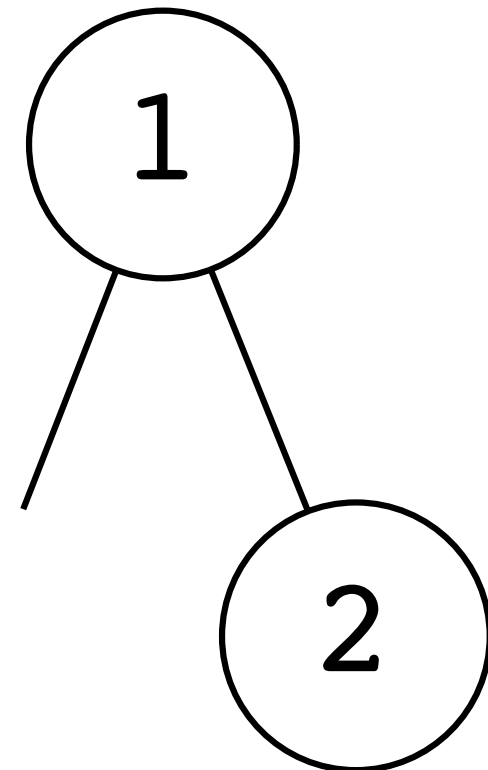
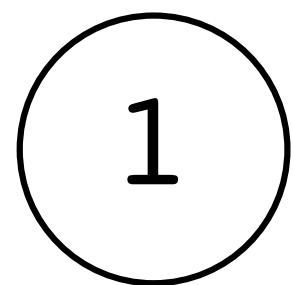


- Si le déséquilibre de X provient d'une insertion, elle a eu lieu dans les sous-arbres B ou C
Pourquoi ? *Parce que si elle avait eu lieu dans A, X aurait déjà été déséquilibré de -2 ou -3.*
- Seuls les cas 1 et 3 sont possibles.
Pourquoi ? *Parce que dans le cas 2, la hauteur de Y n'a pas pu changer dû à l'insertion.*
- L'équilibrage de X diminue donc sa hauteur de 1, i.e. rétablit sa hauteur avant insertion.

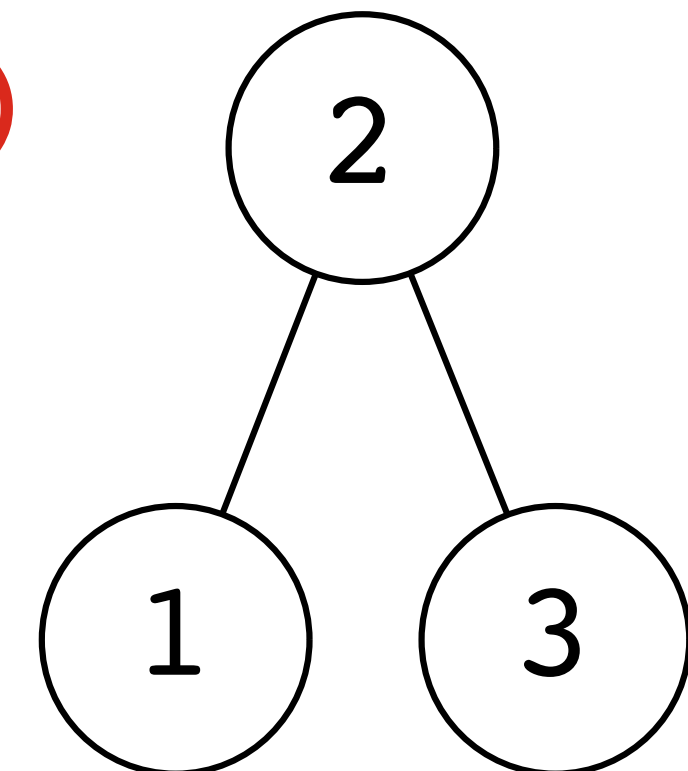
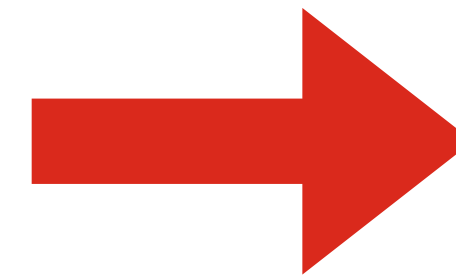
Exemple : insérer 1, 2, 3



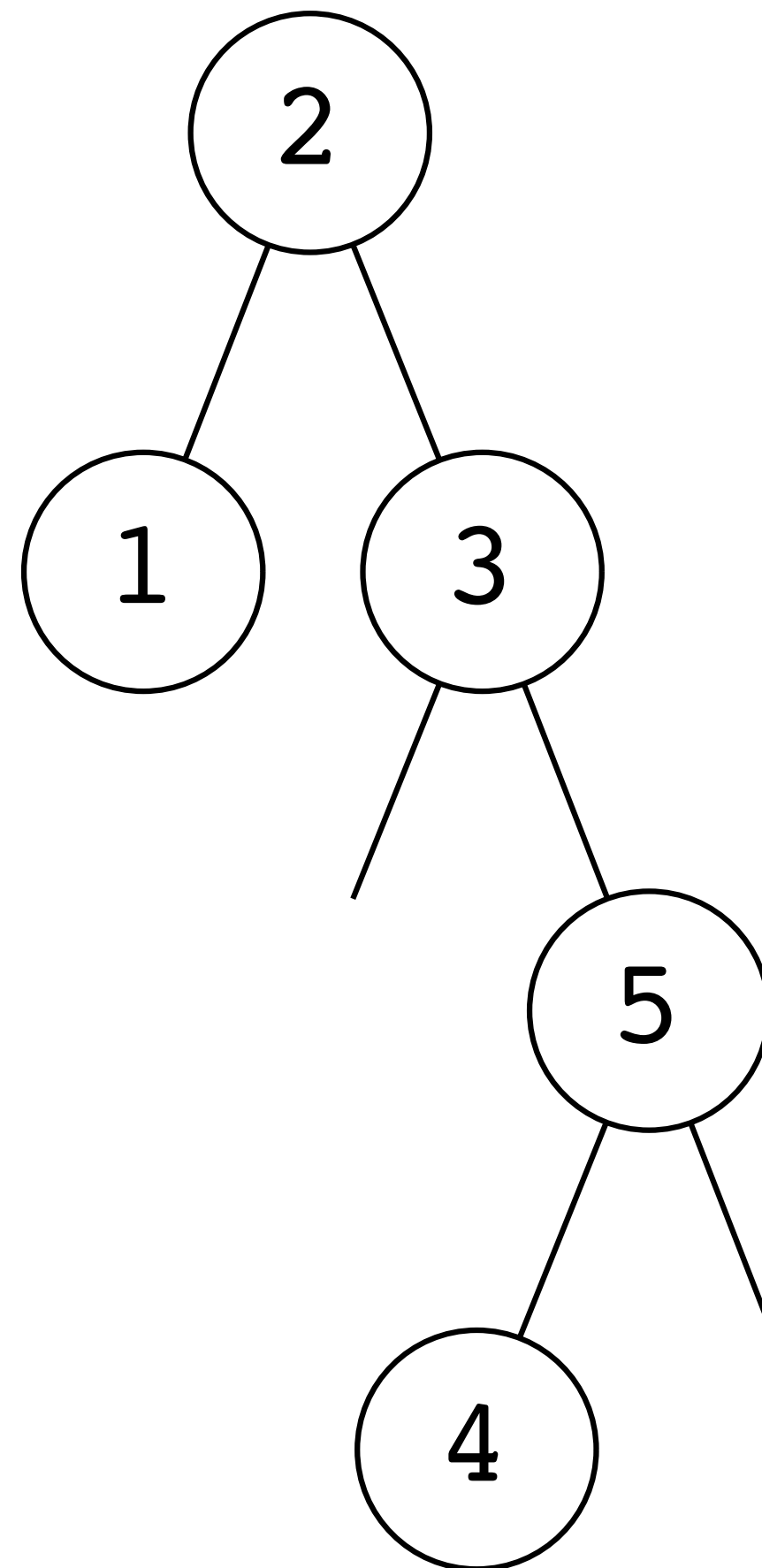
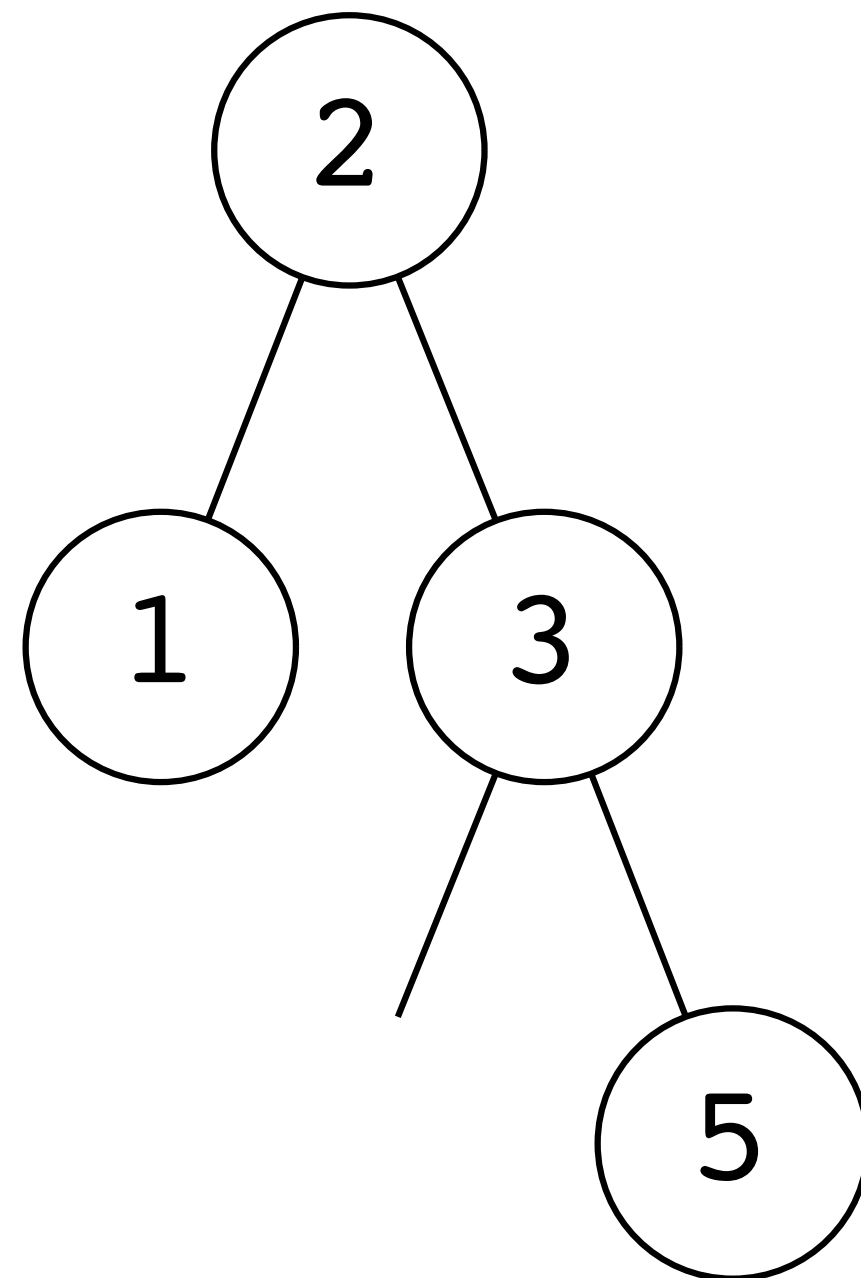
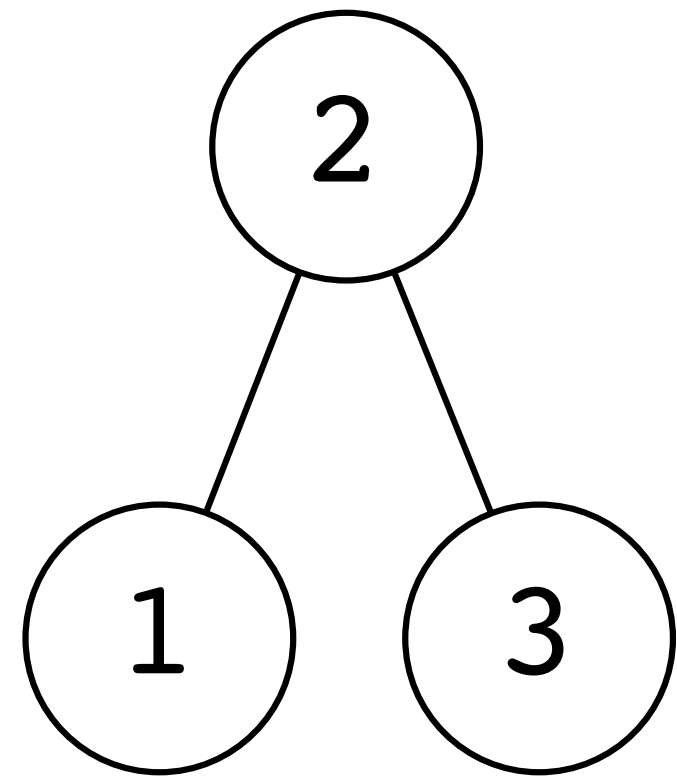
Exemple : insérer 1, 2, 3



rotation_gauche(1)
en sortie de récursion



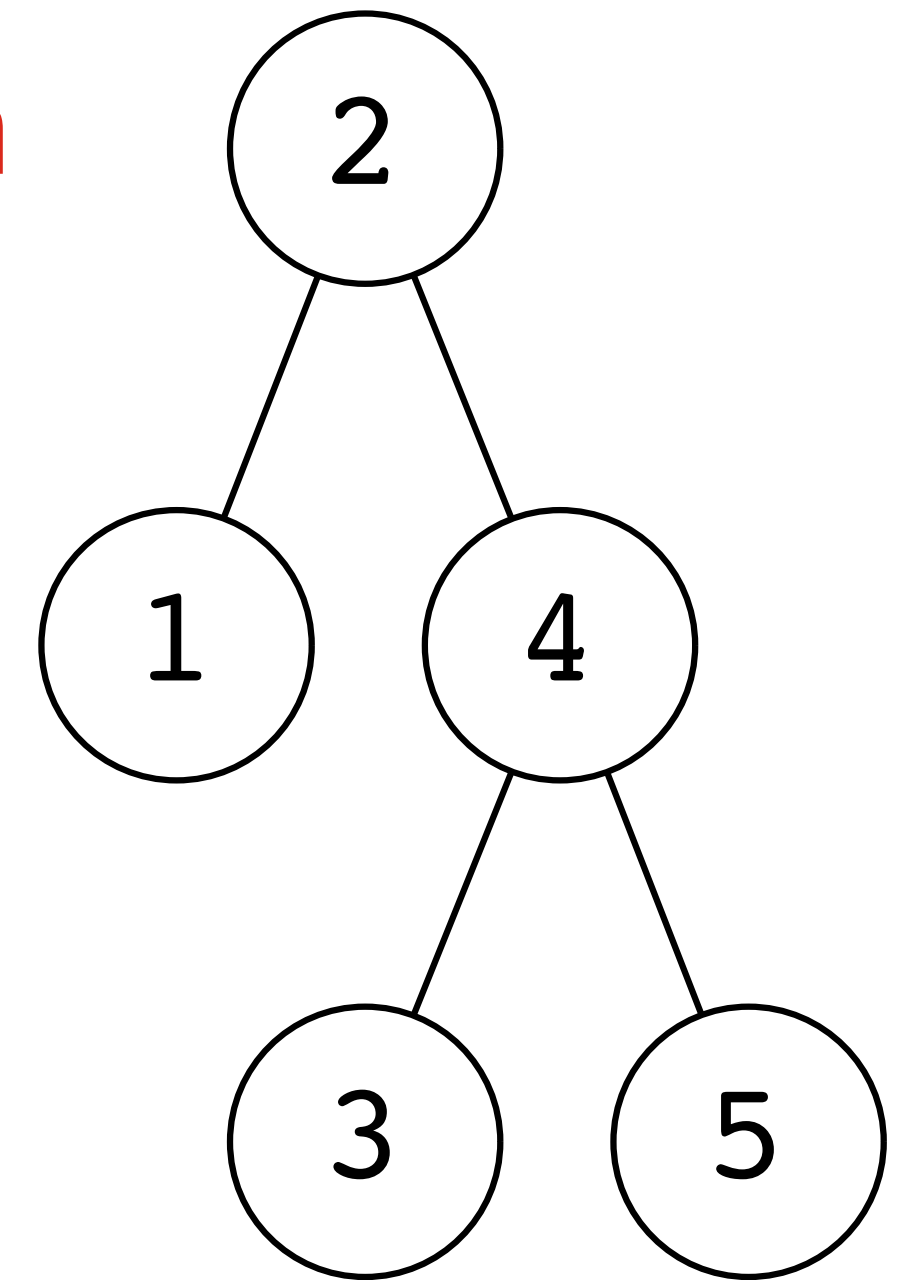
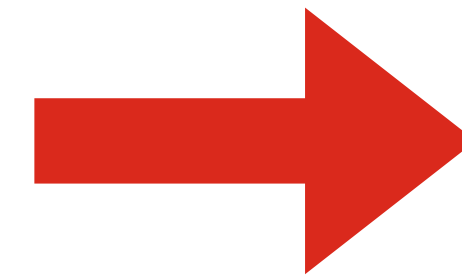
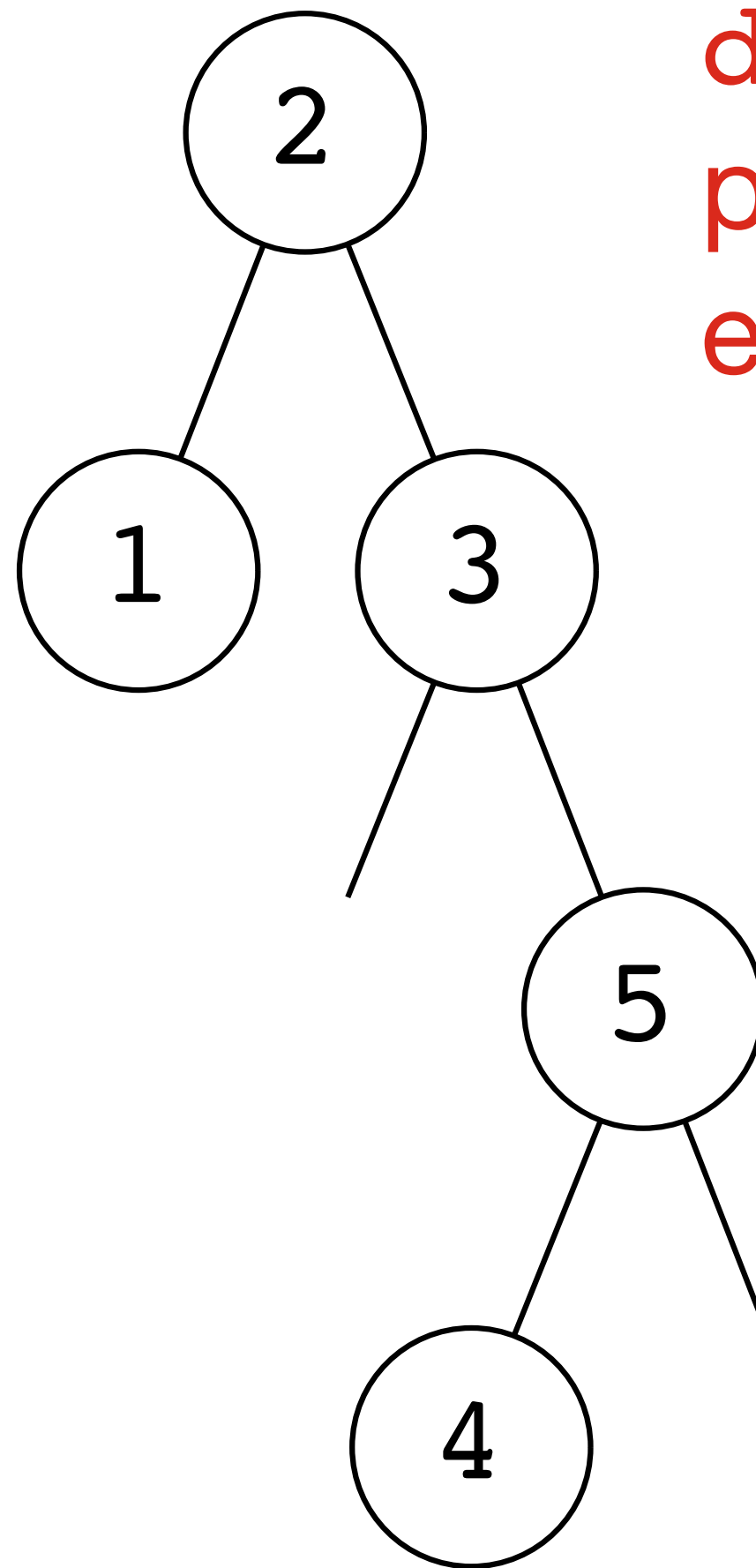
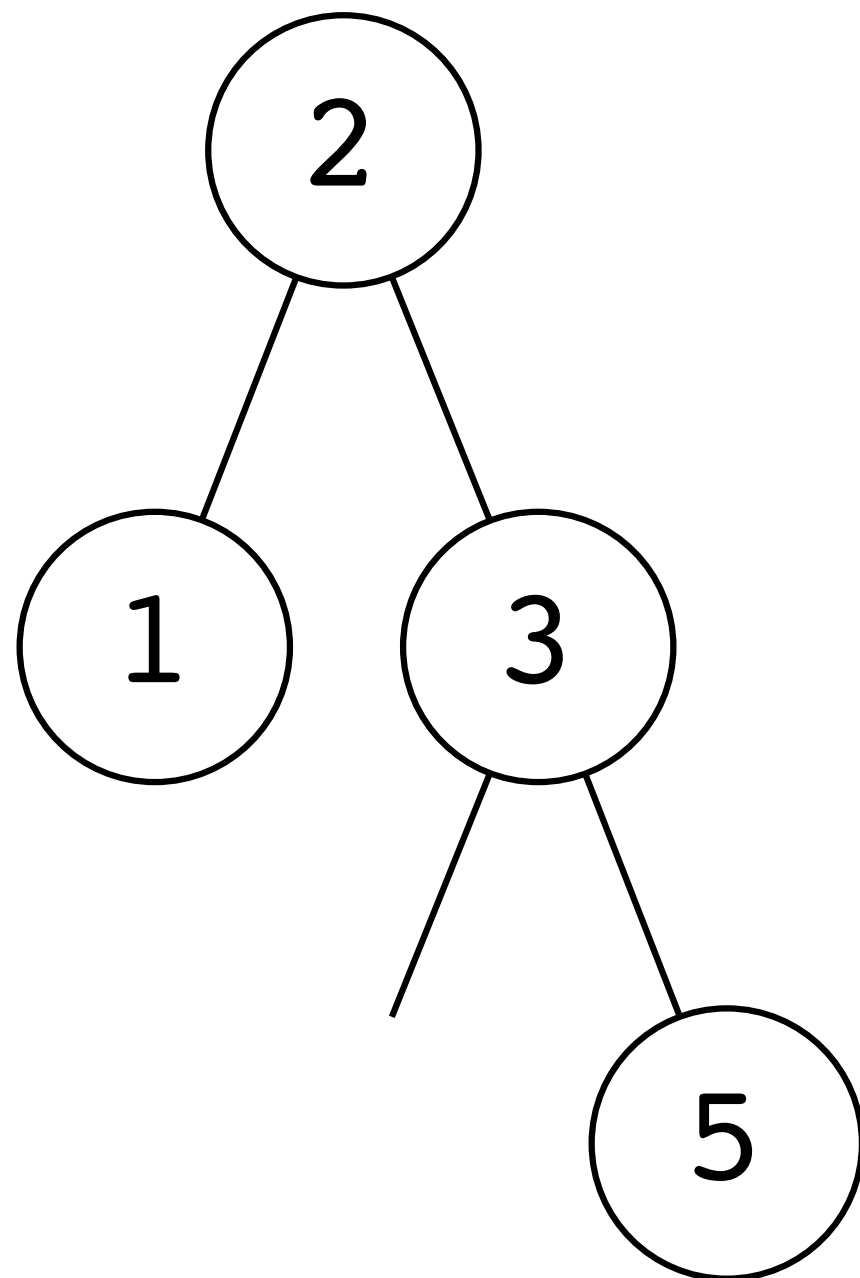
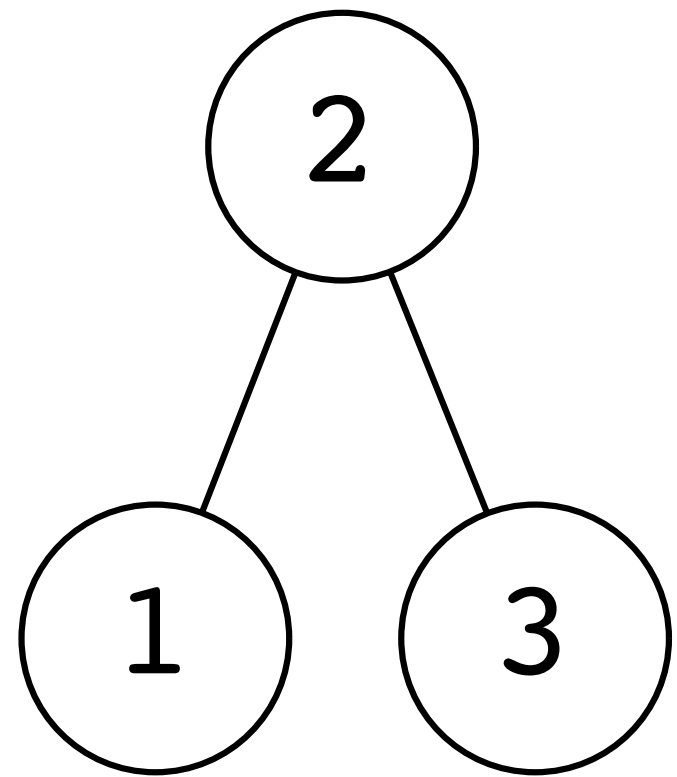
Exemple : insérer 5, 4



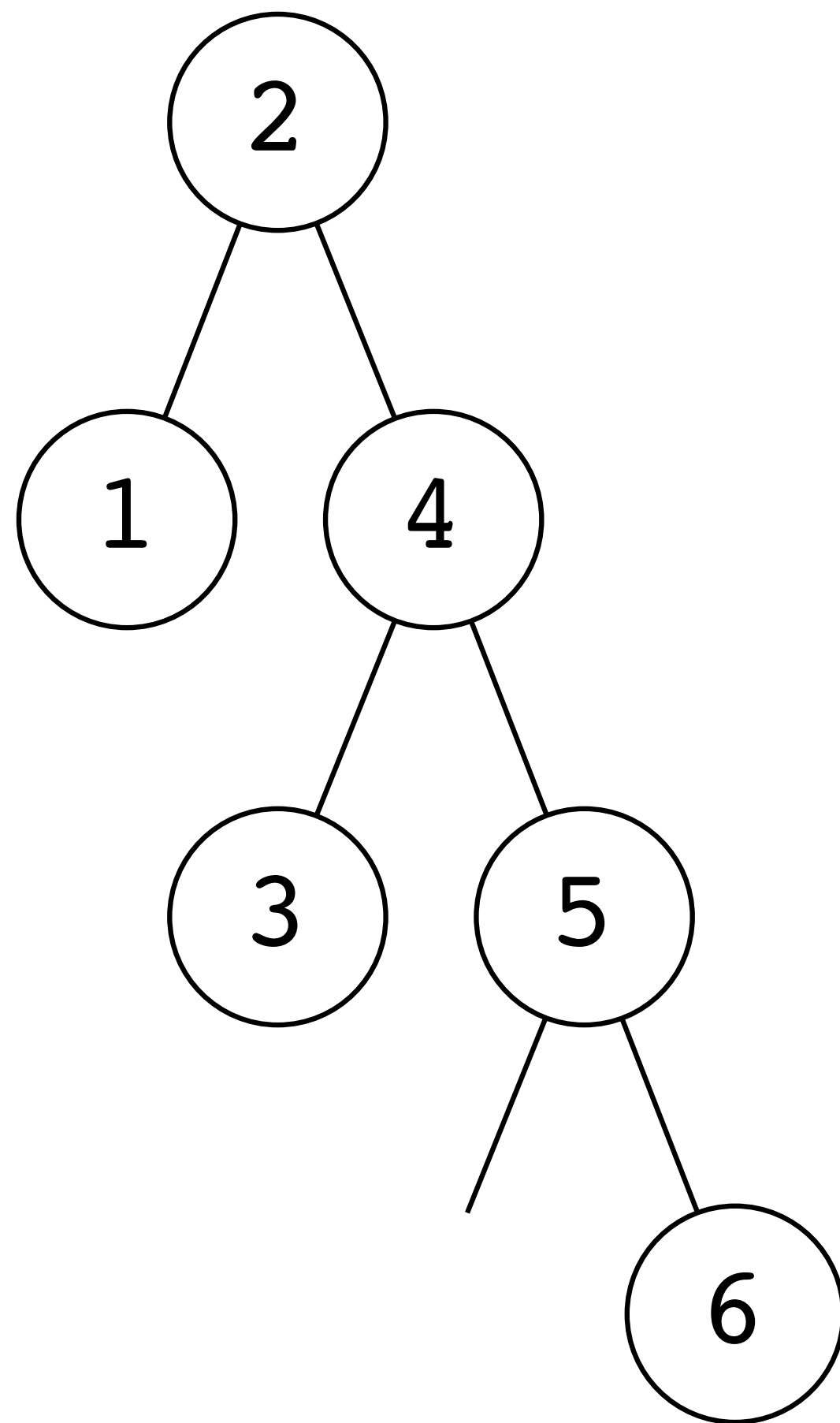
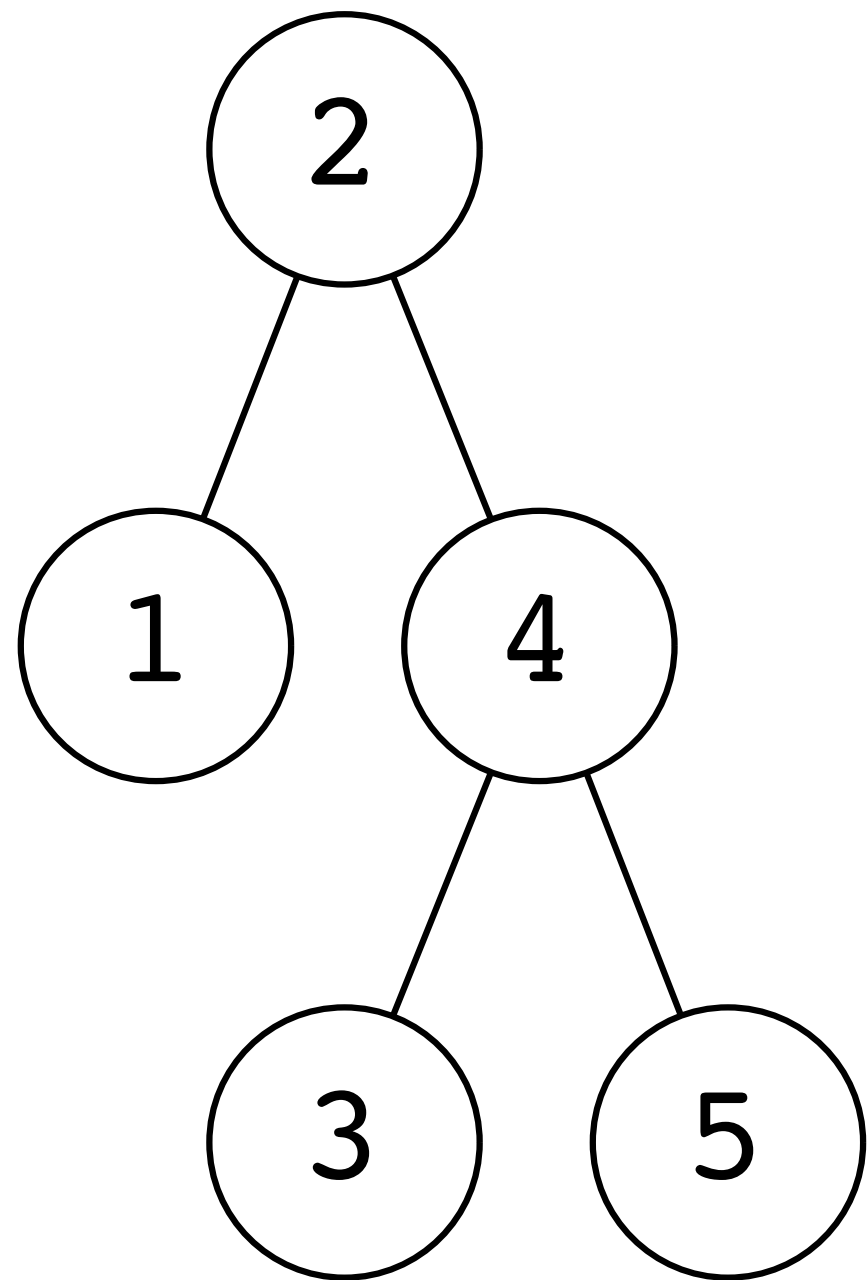


Exemple : insérer 5, 4

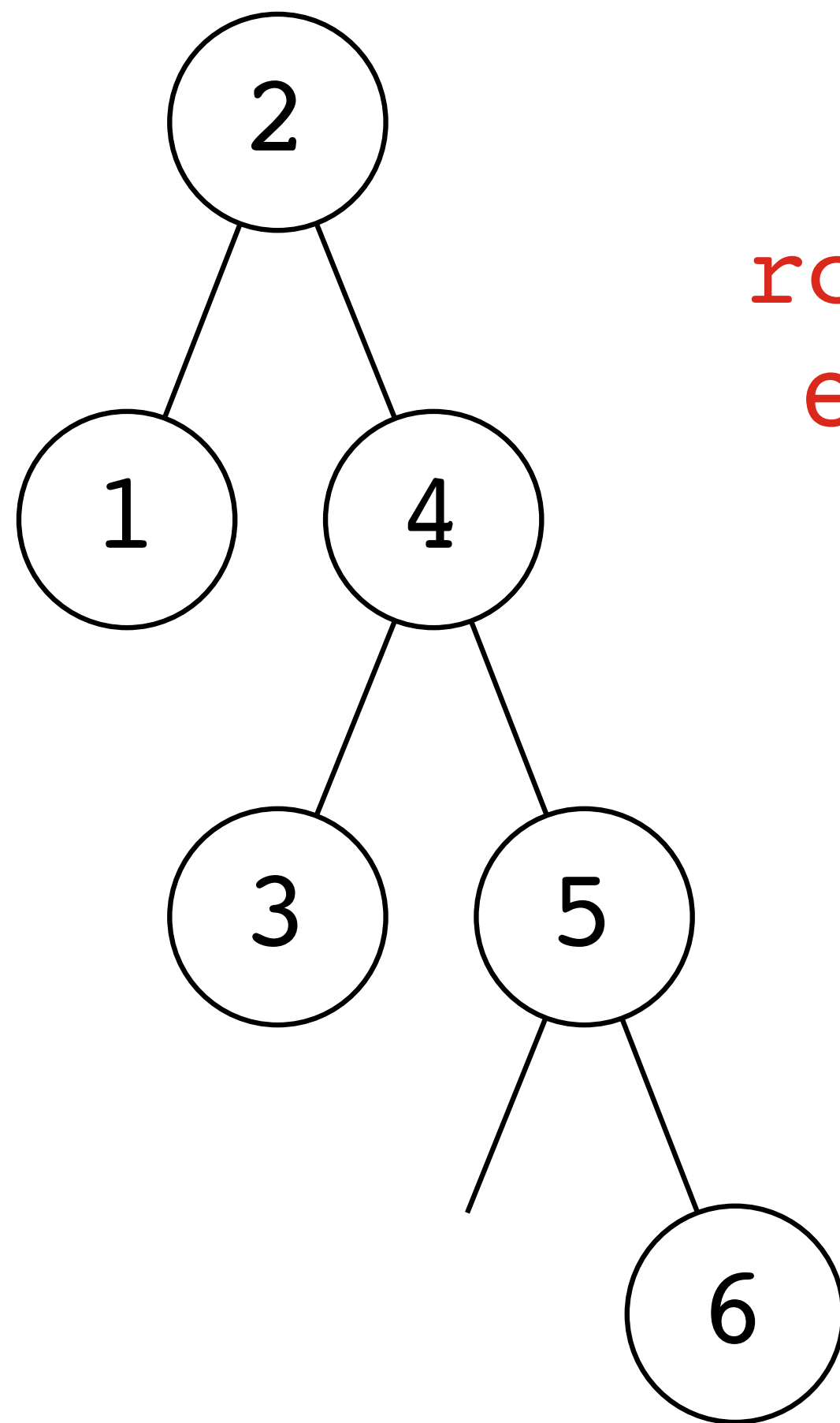
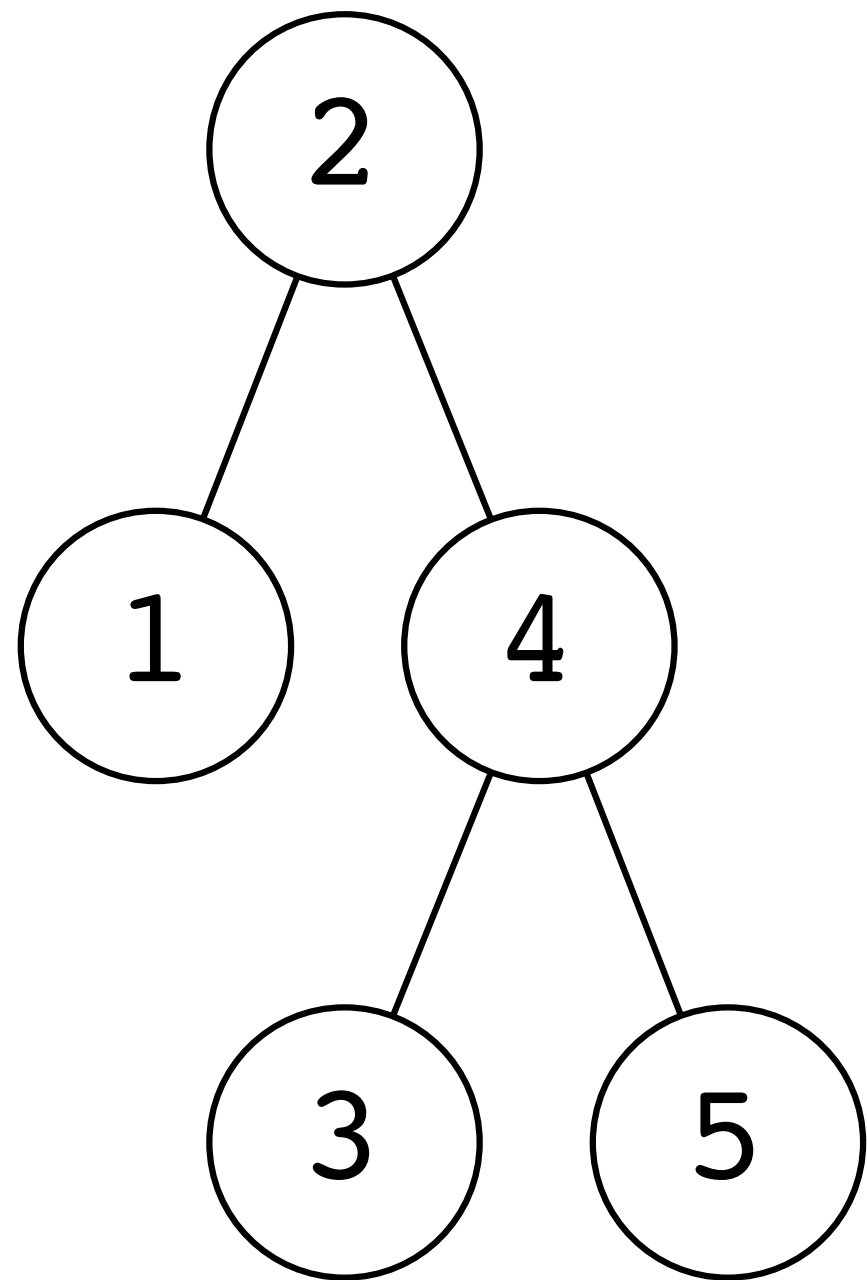
double rotation :
droite(5) puis gauche(3)
pour ré-équilibrer 3
en sortie de récursion



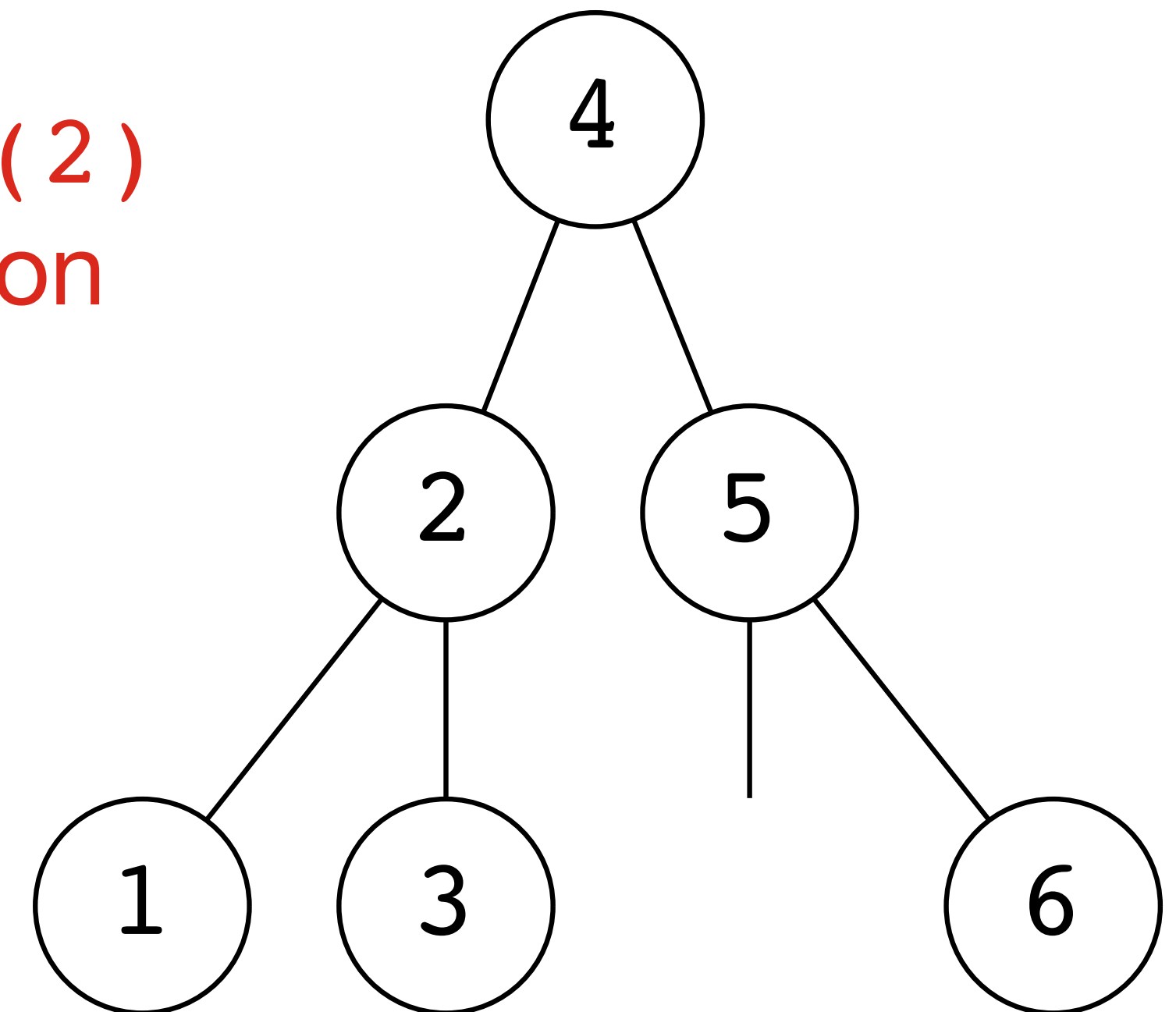
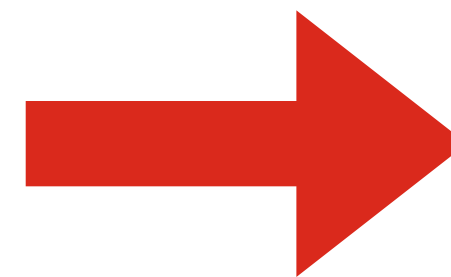
Exemple : insérer 6



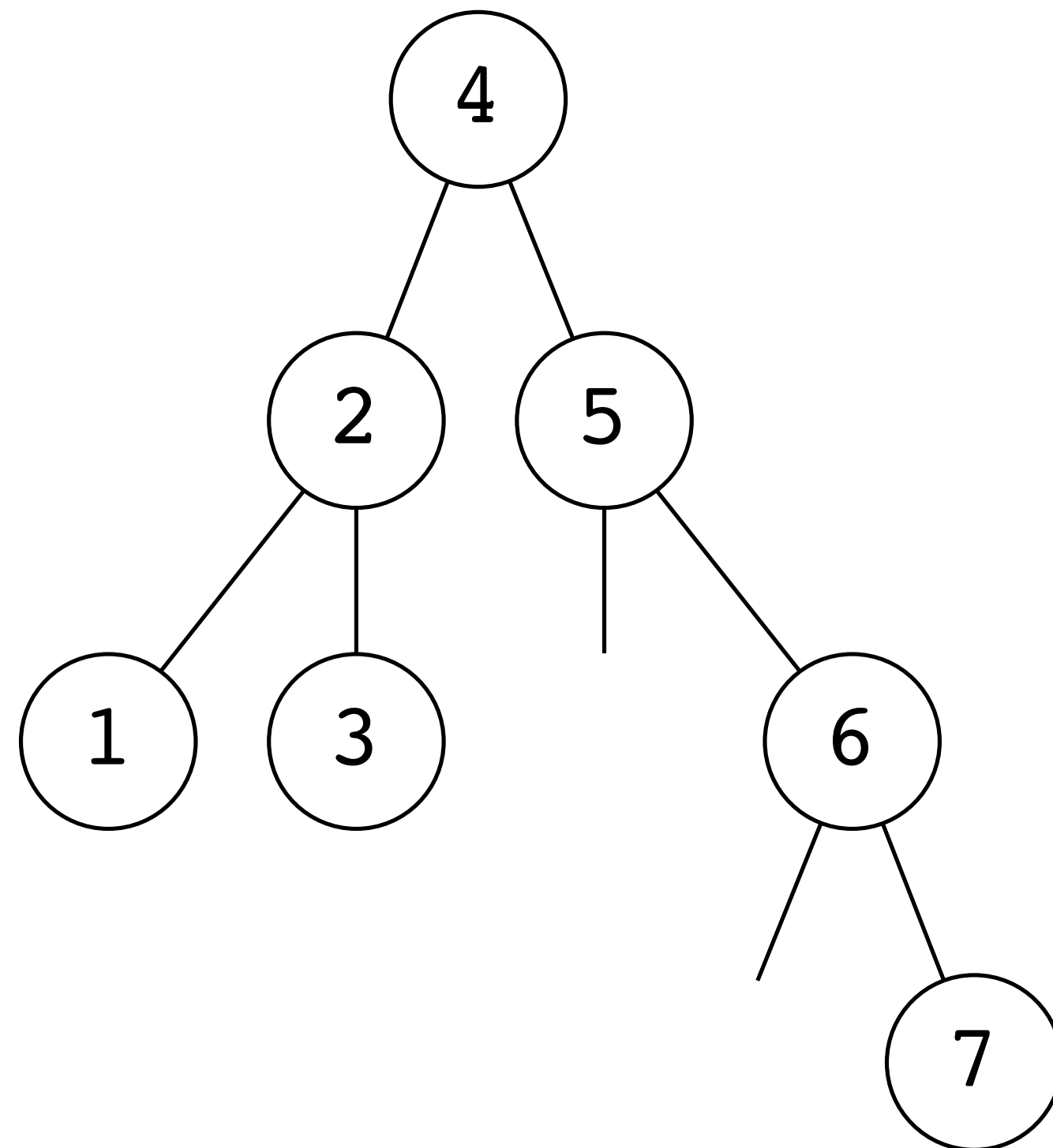
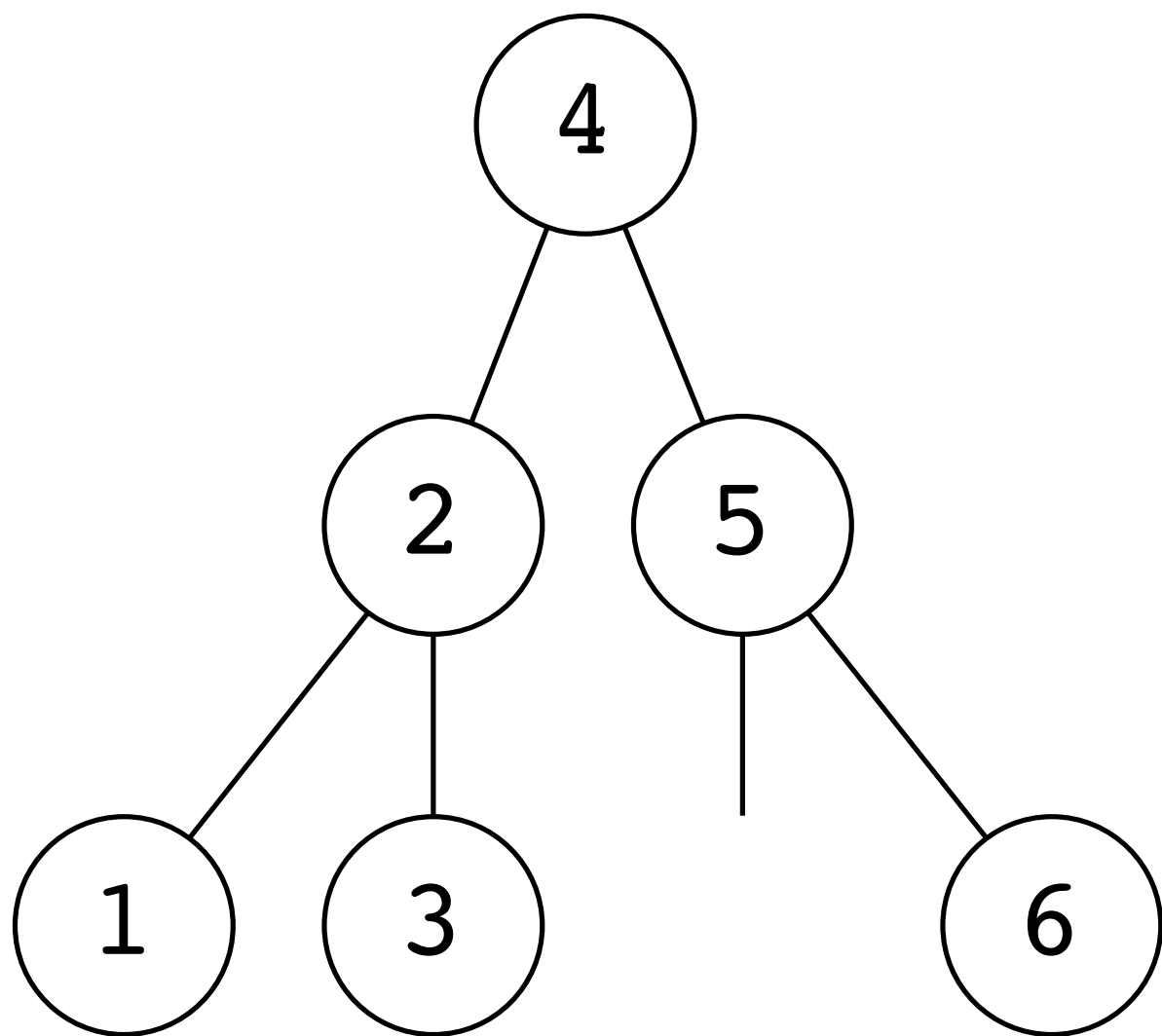
Exemple : insérer 6



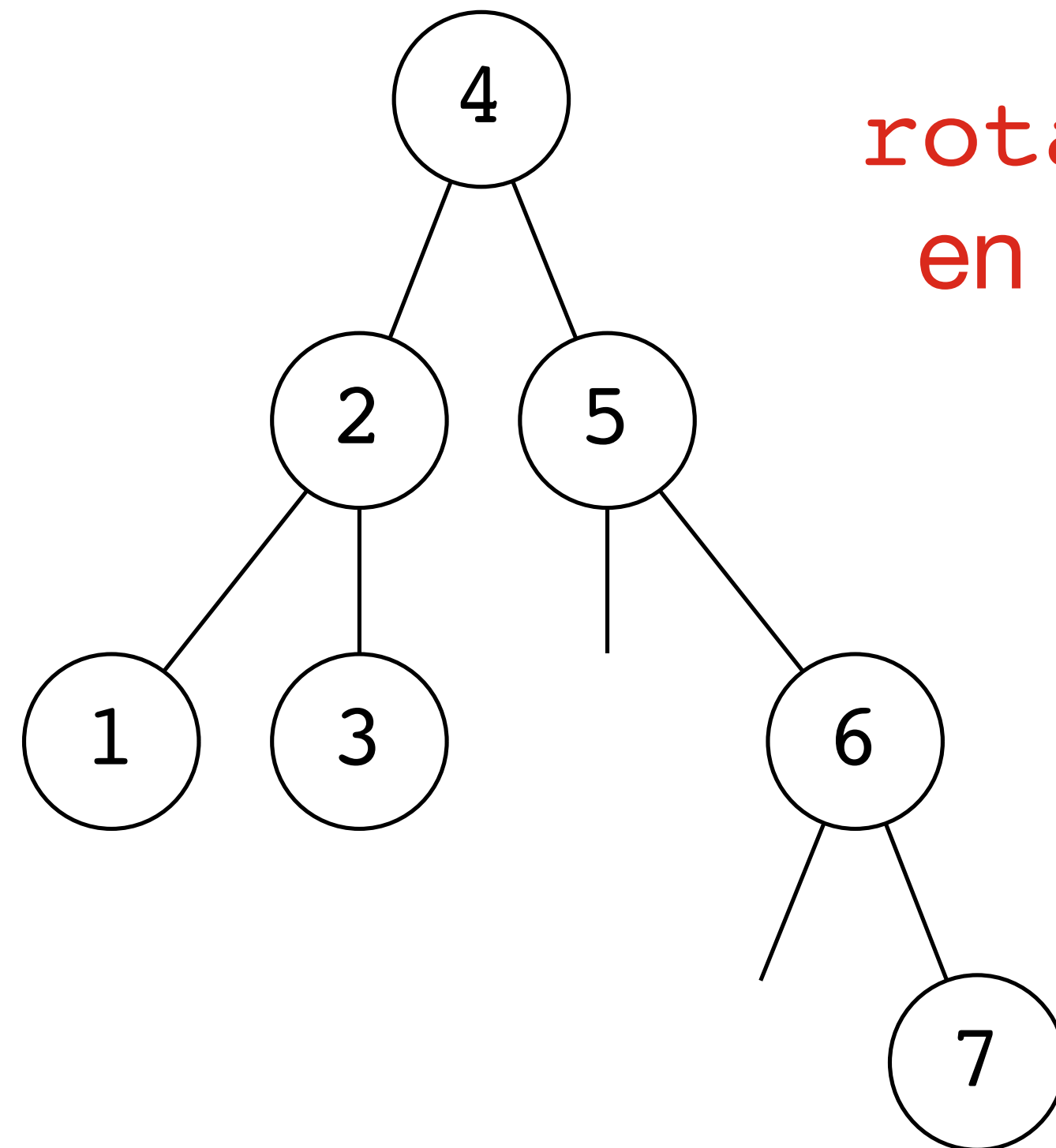
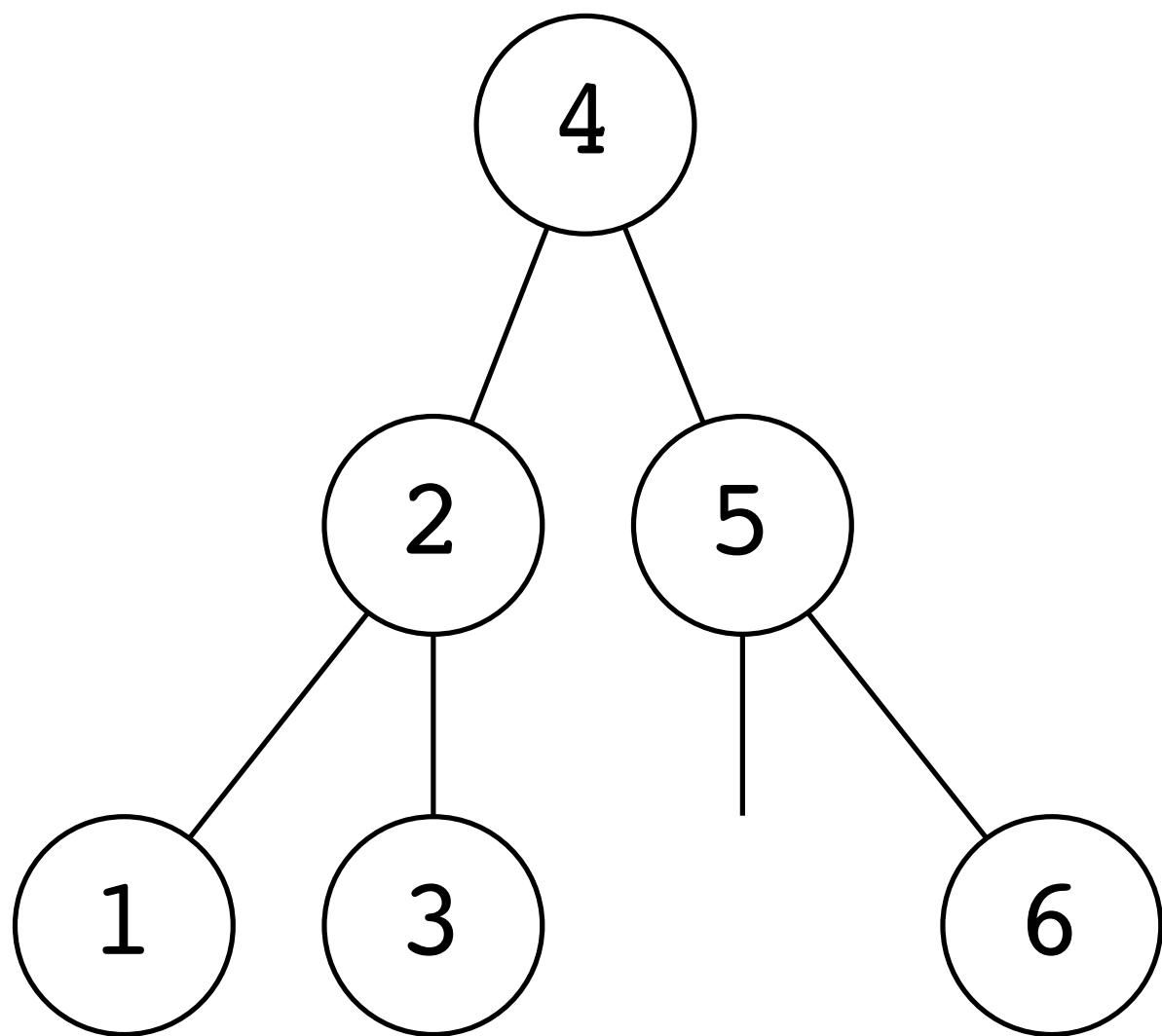
`rotation_gauche(2)`
en sortie de récursion



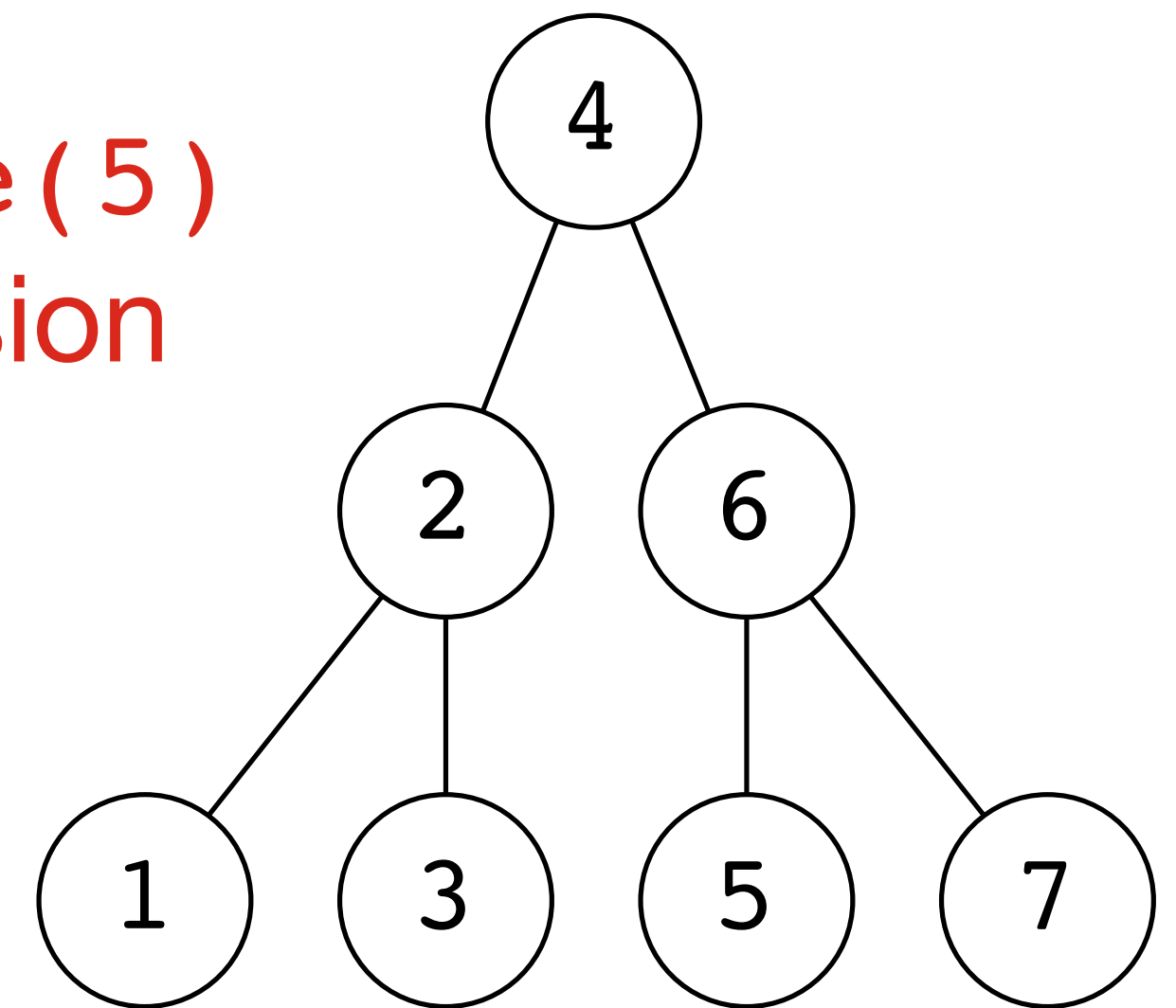
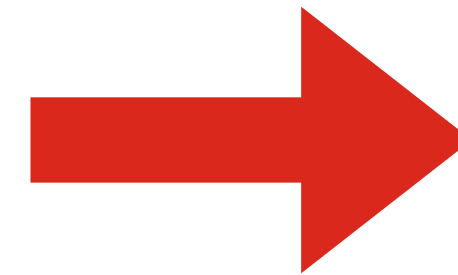
Exemple : insérer 7



Exemple : insérer 7



rotation_gauche(5)
en sortie de récursion



Suppression d'un arbre AVL



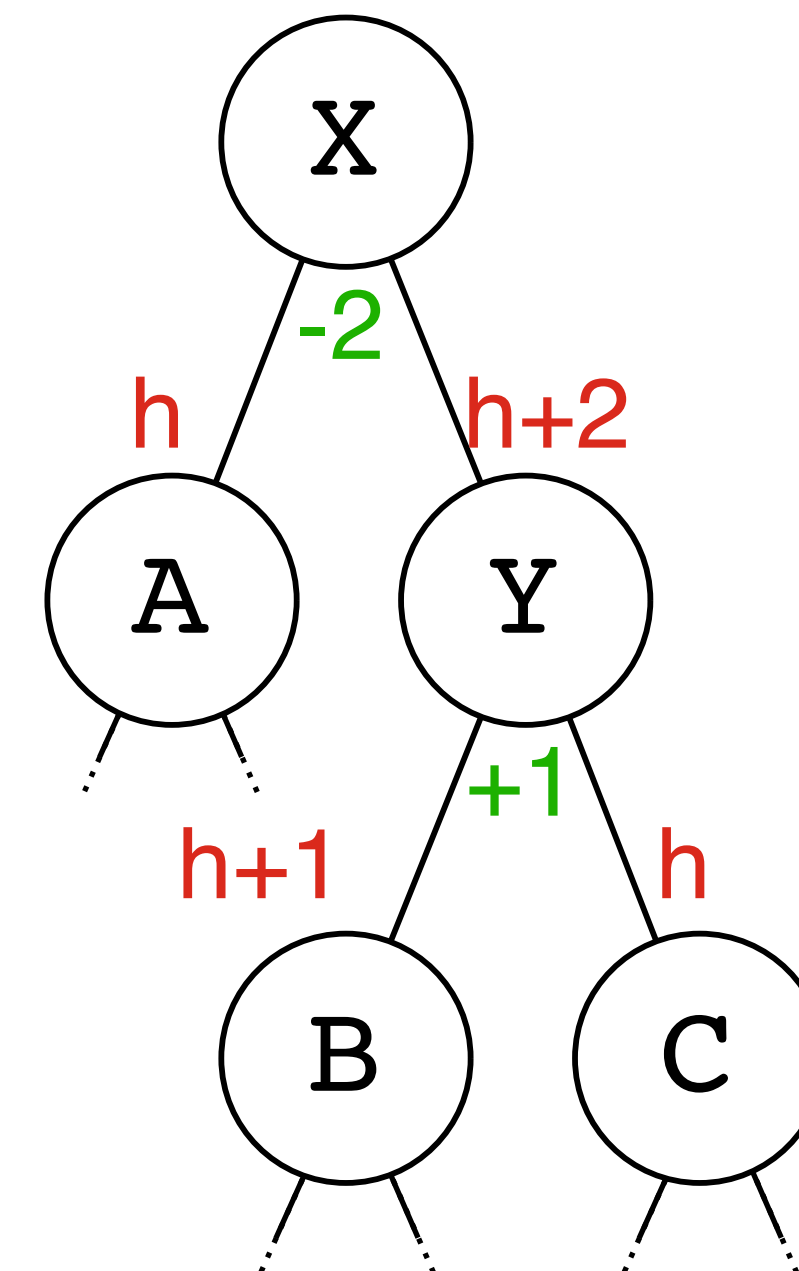
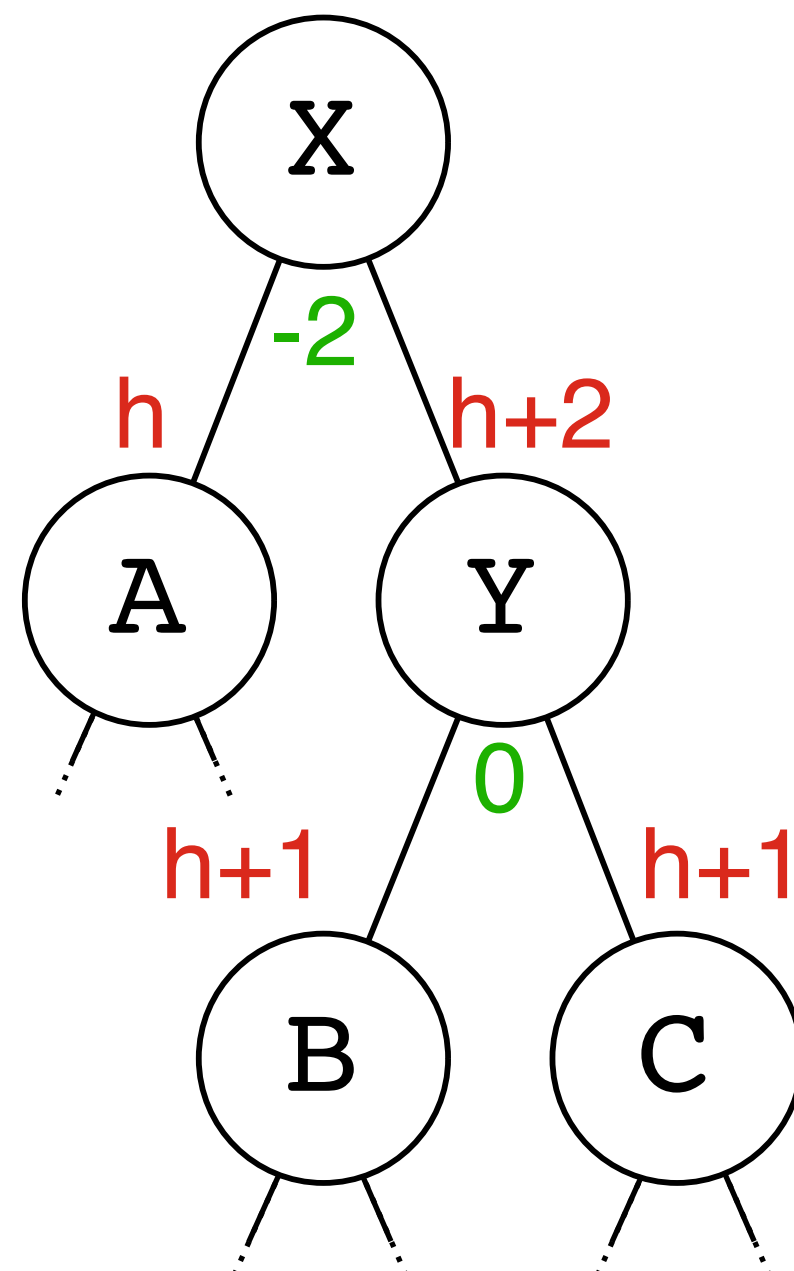
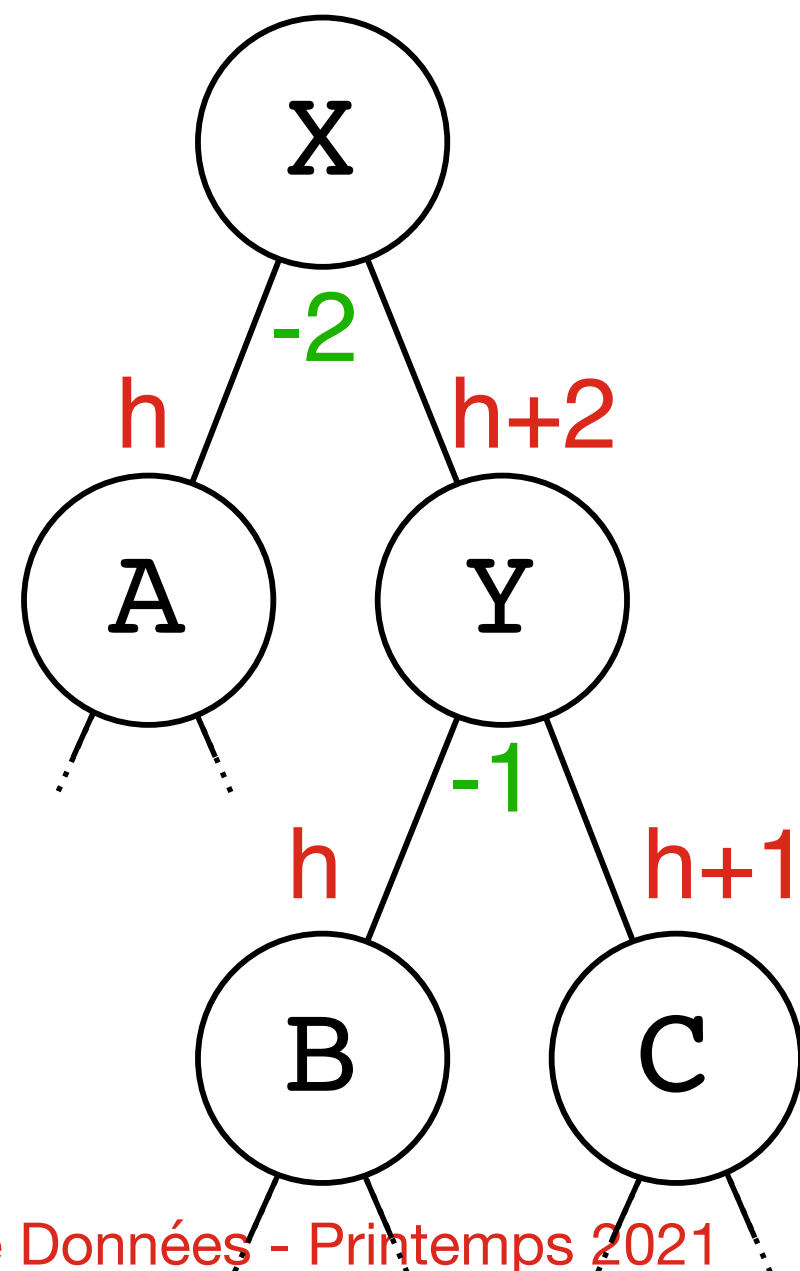
Comme pour l'insertion,

- code identique à celui de suppression dans un ABR
- avec appel à `rétablir_équilibre(r)` après un appel récursif

Suppression d'un arbre AVL



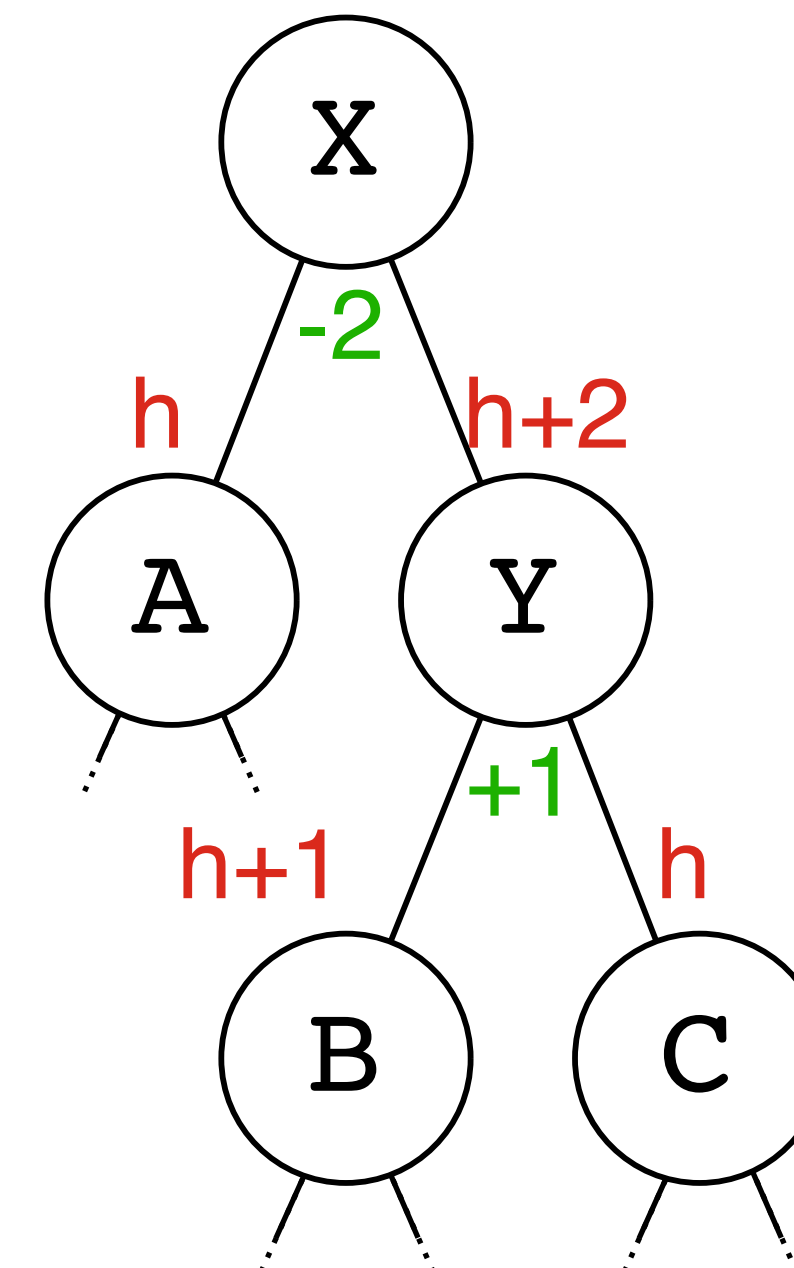
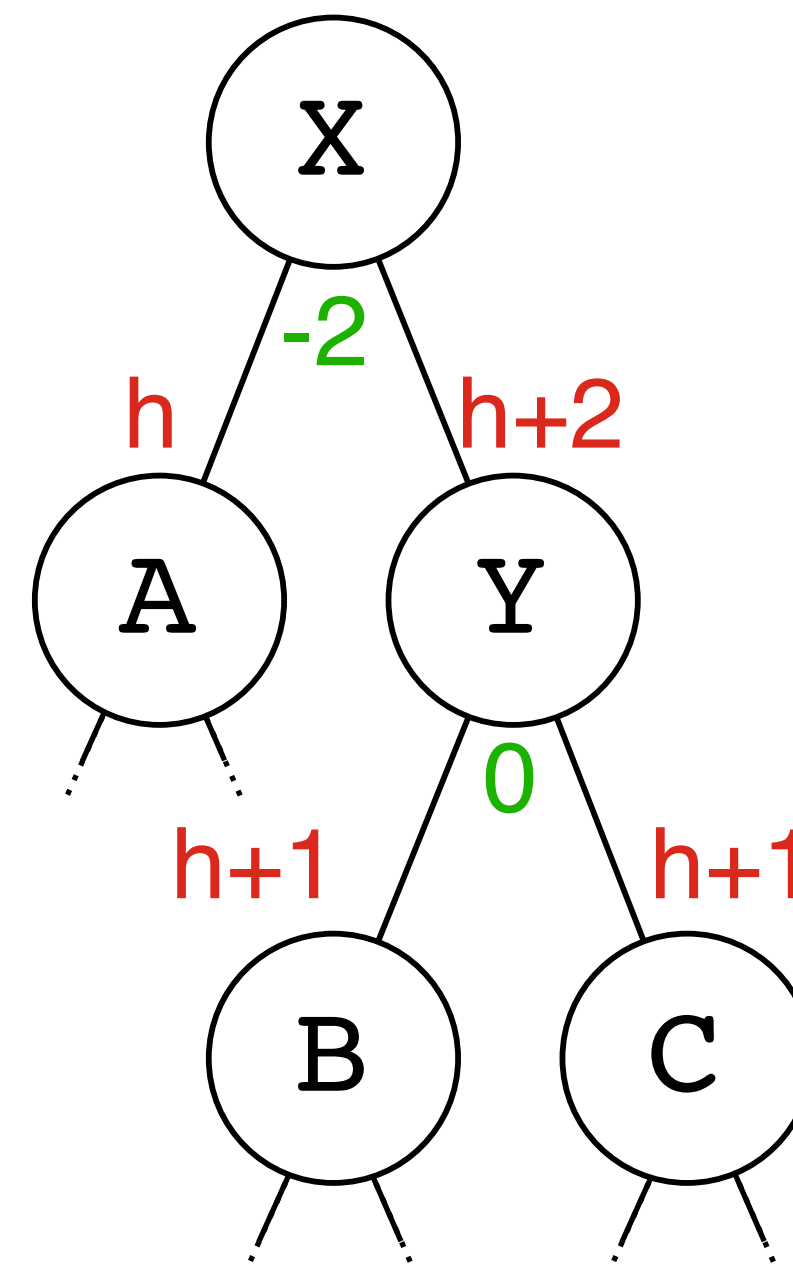
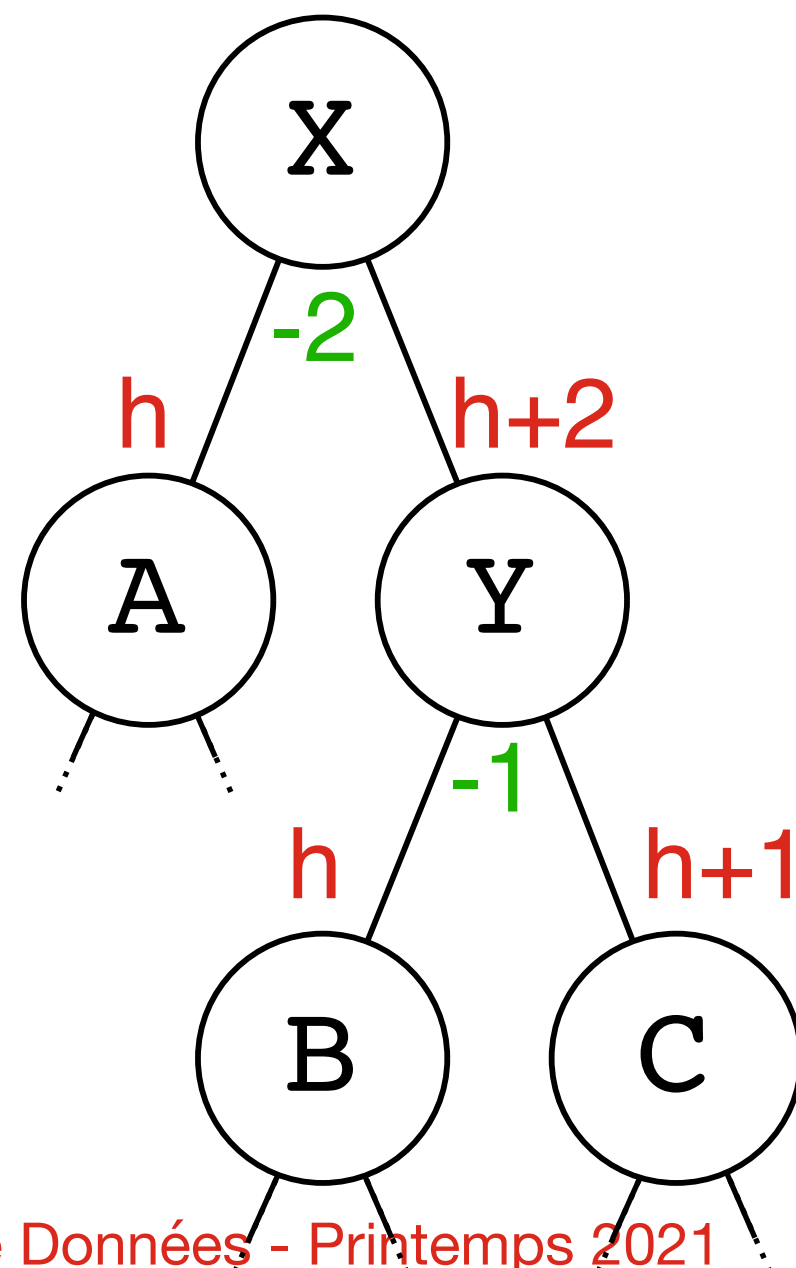
- Un déséquilibre ne survient que si l'élément a été supprimé du sous-arbre A.



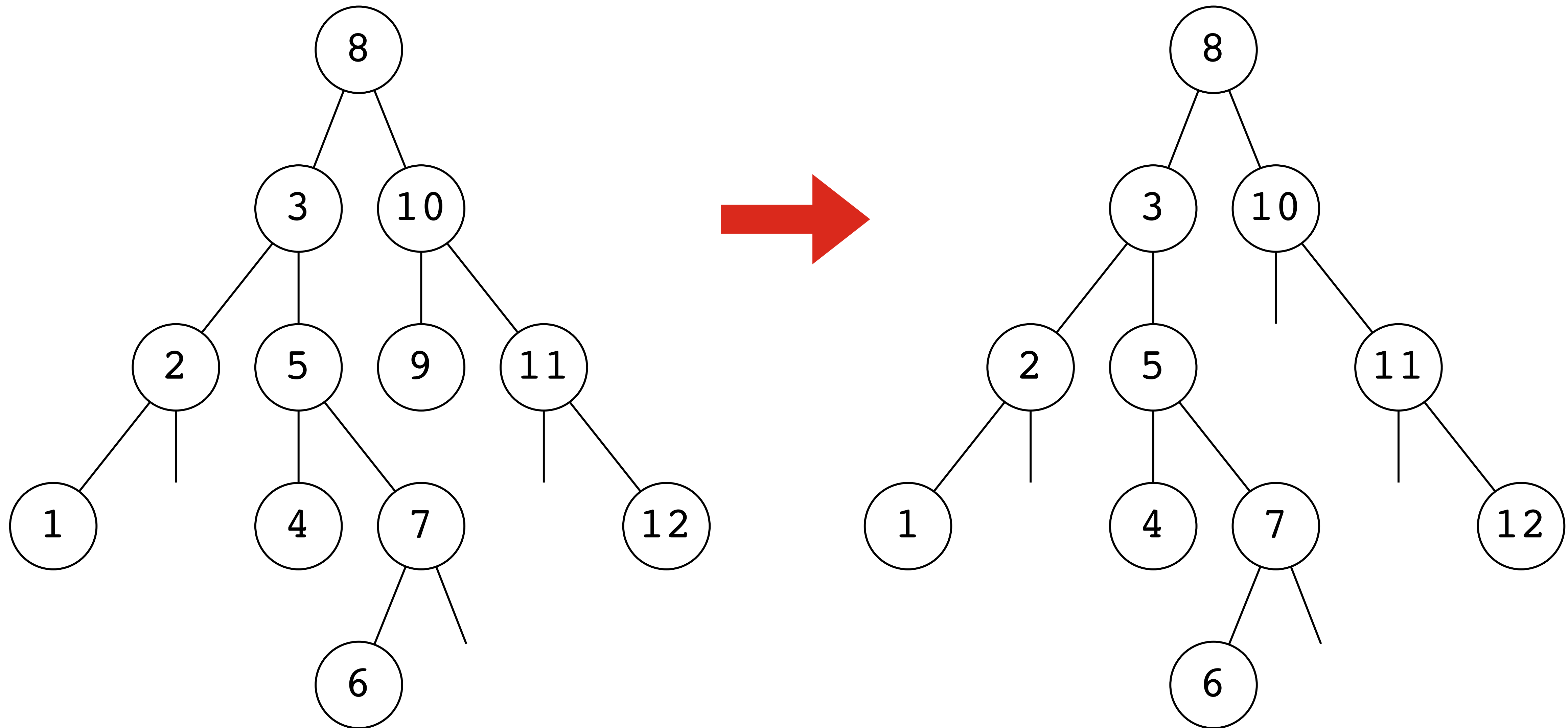
Suppression d'un arbre AVL



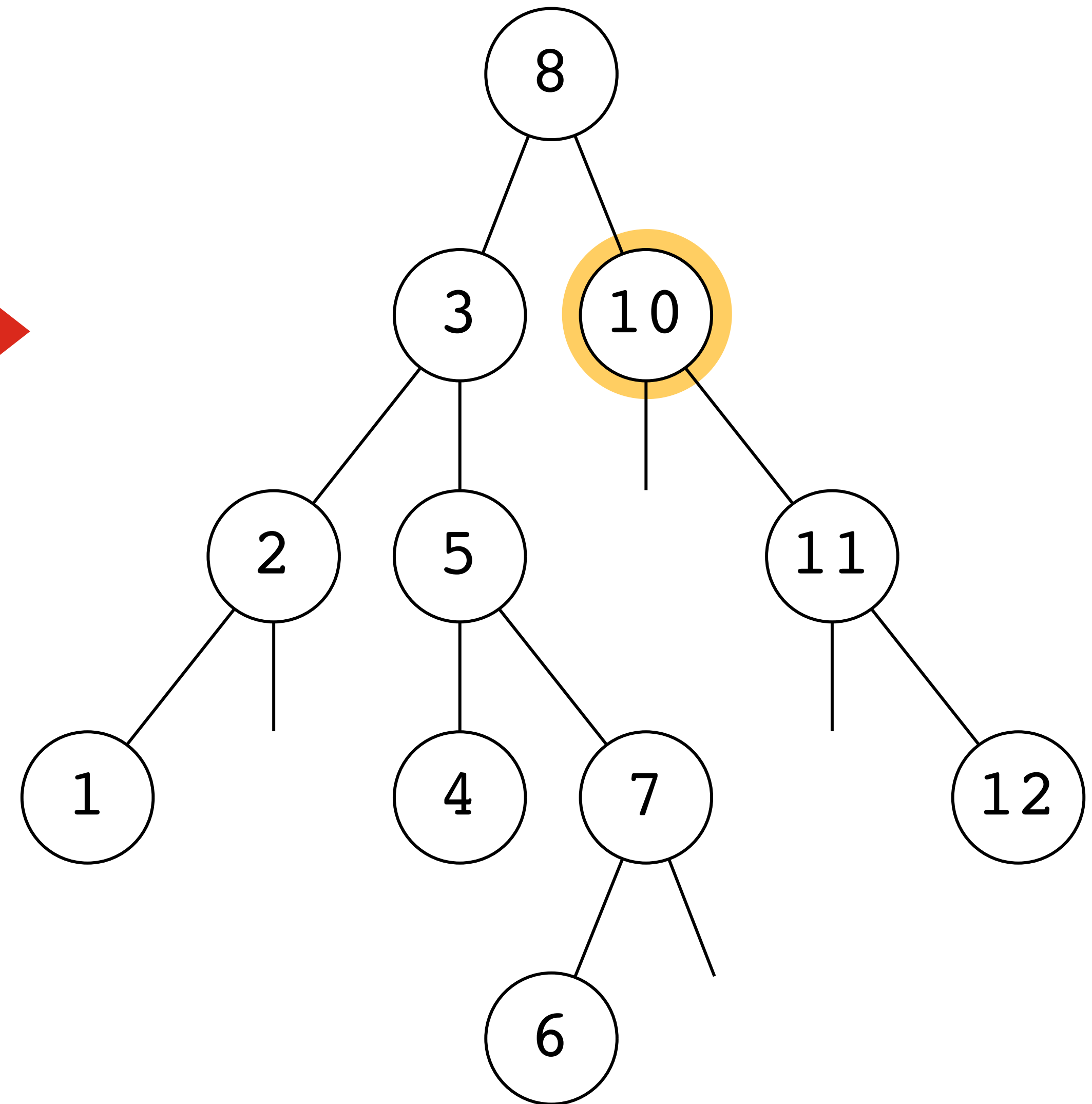
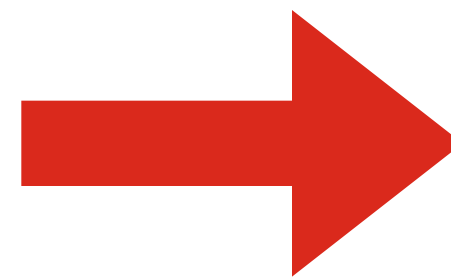
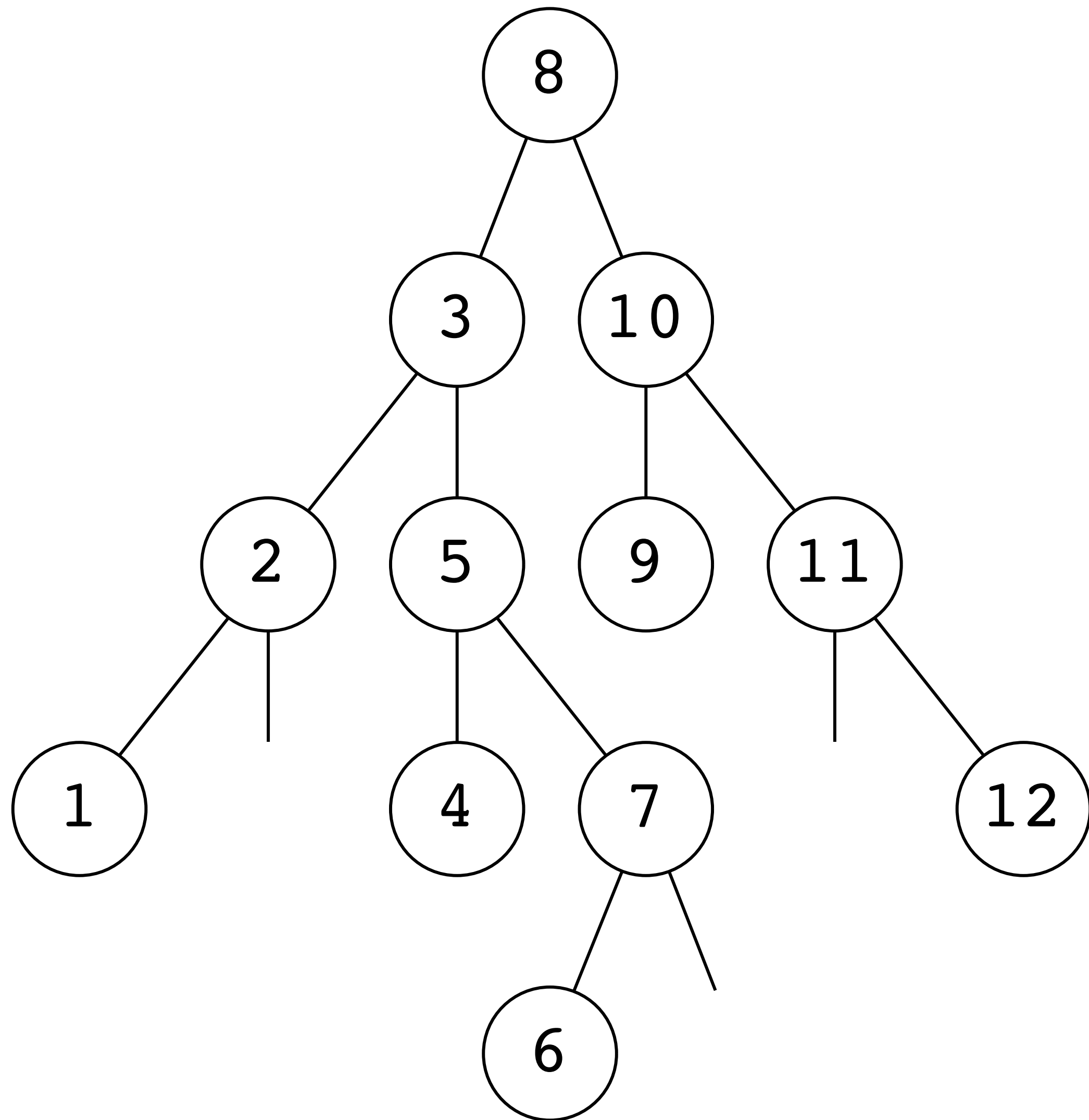
- Un déséquilibre ne survient que si l'élément a été supprimé du sous-arbre A.
Pourquoi ? Parce que si elle avait eu lieu dans B ou C, X aurait déjà été déséquilibré.
- Dans les cas 1 et 3, l'équilibrage diminue la hauteur de X de 1, ce qui peut propager le déséquilibre à son parent, qu'il faut alors rééquilibrer aussi.



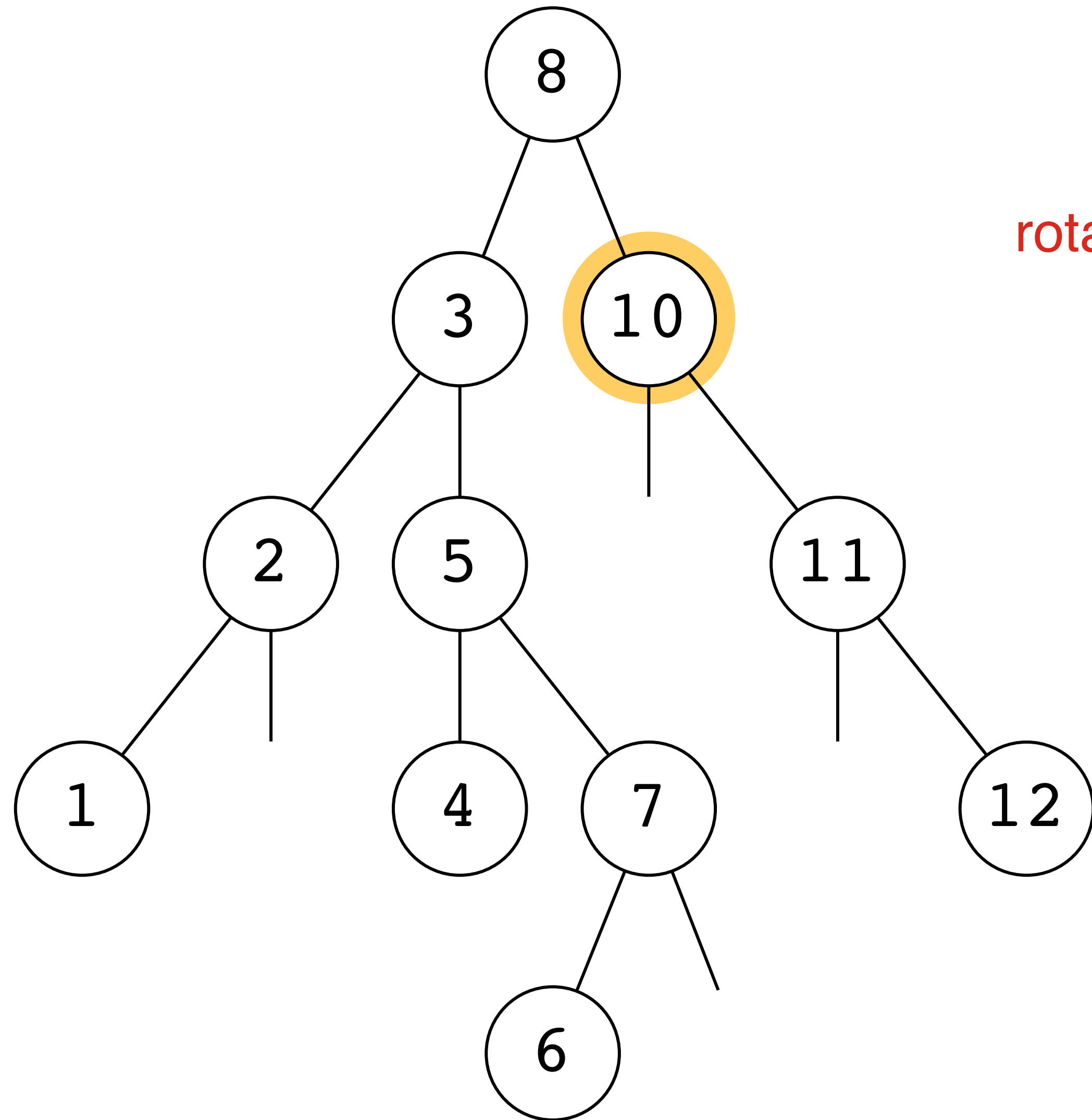
Exemple : supprimer 9



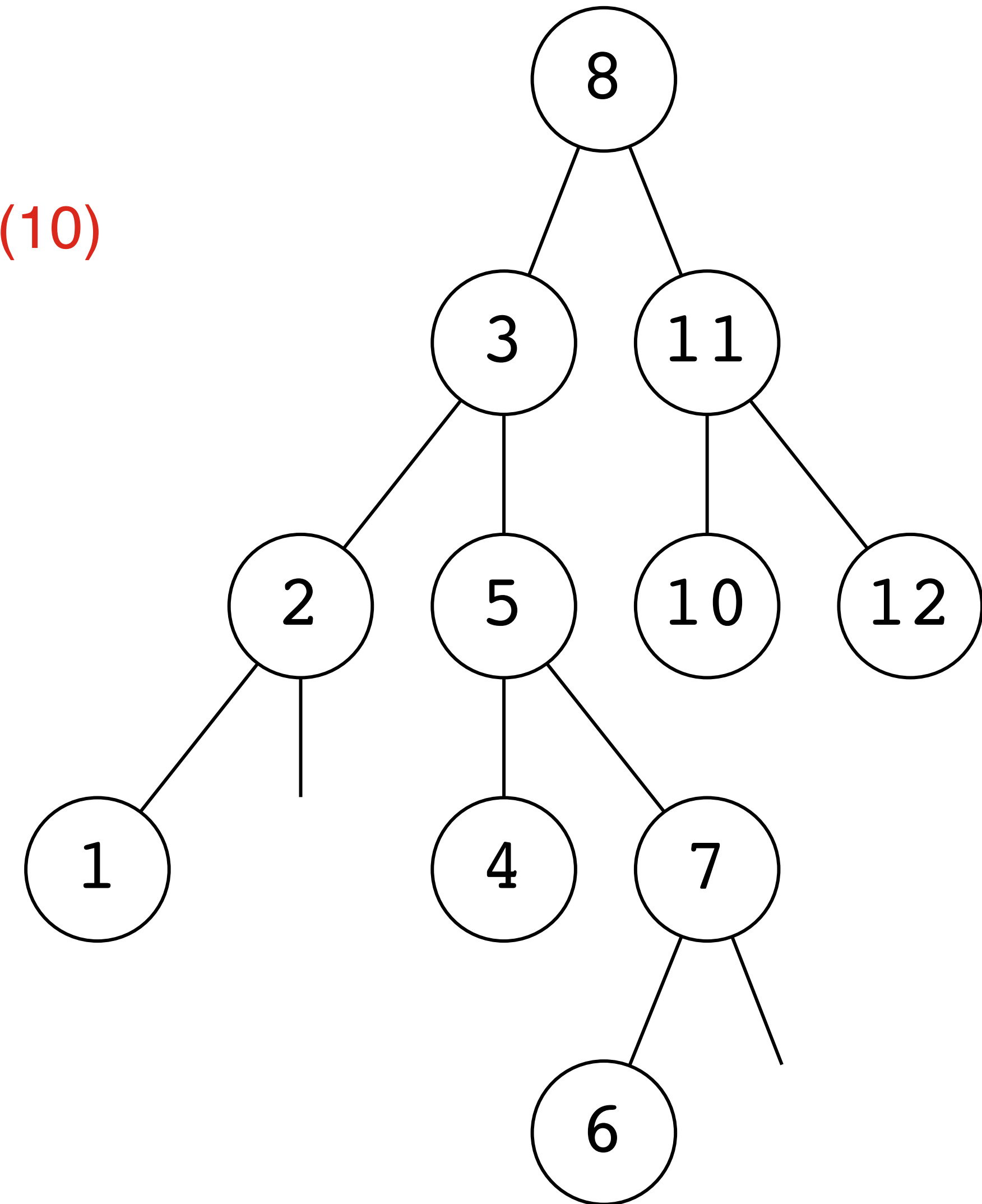
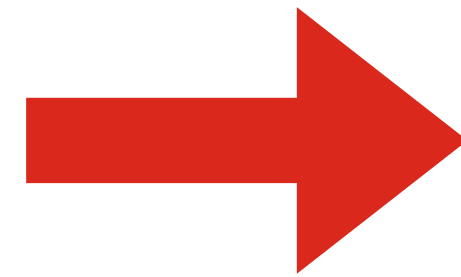
Exemple : supprimer 9



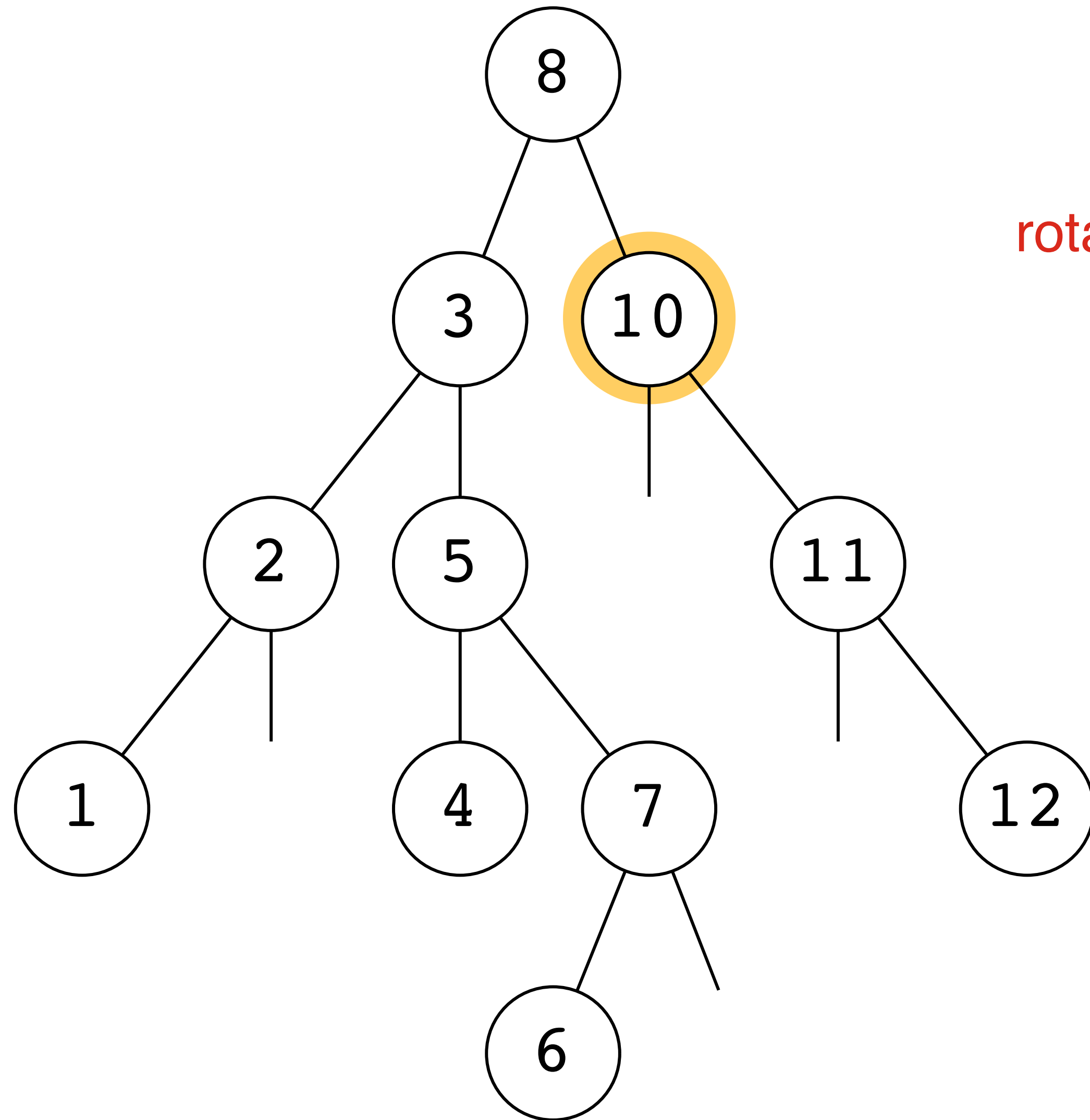
Déséquilibre en 10



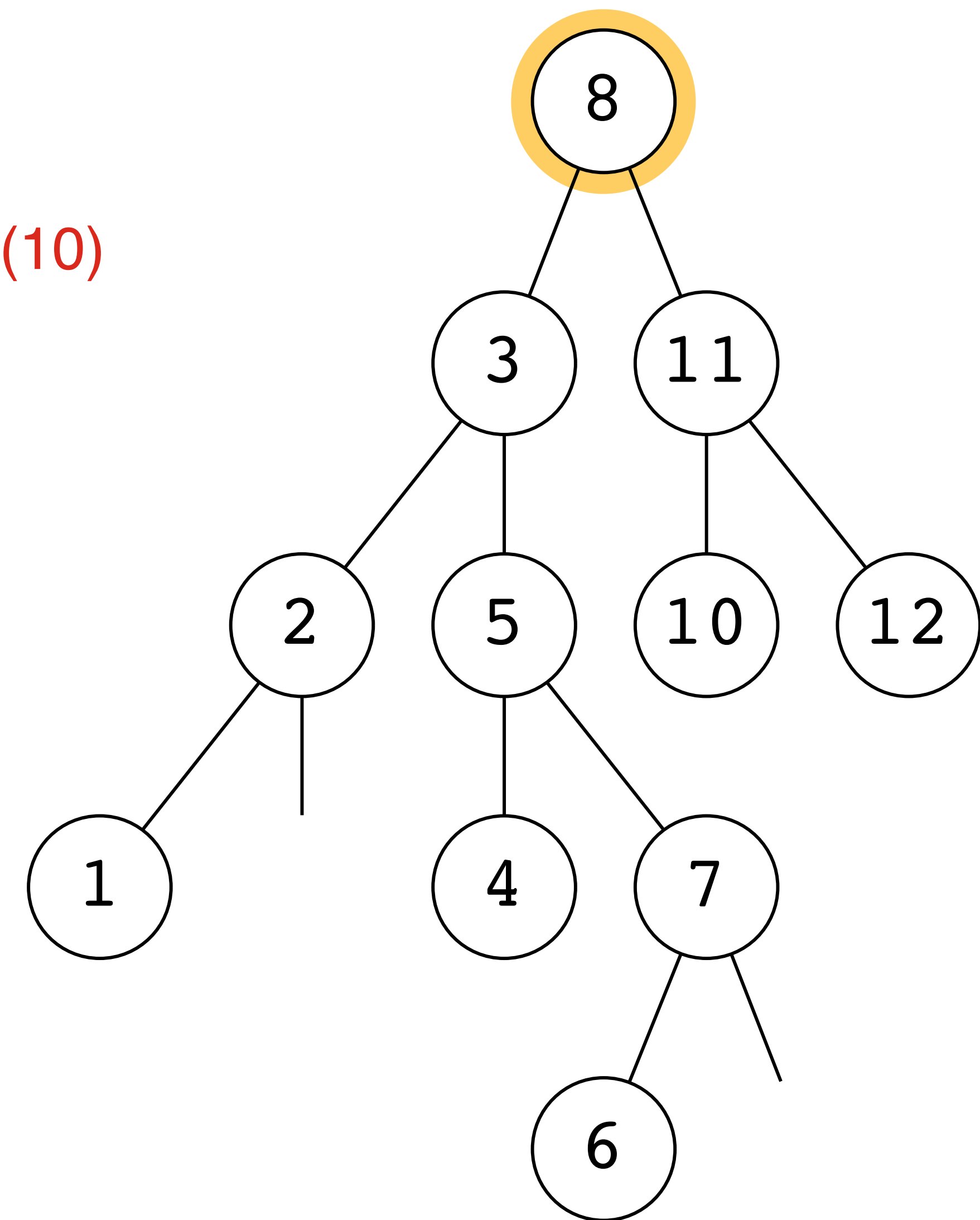
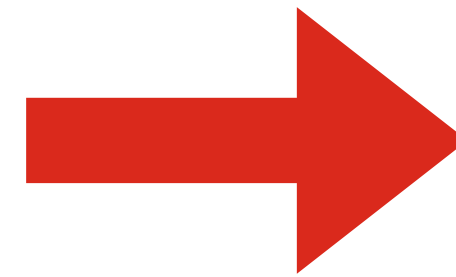
rotation_gauche(10)



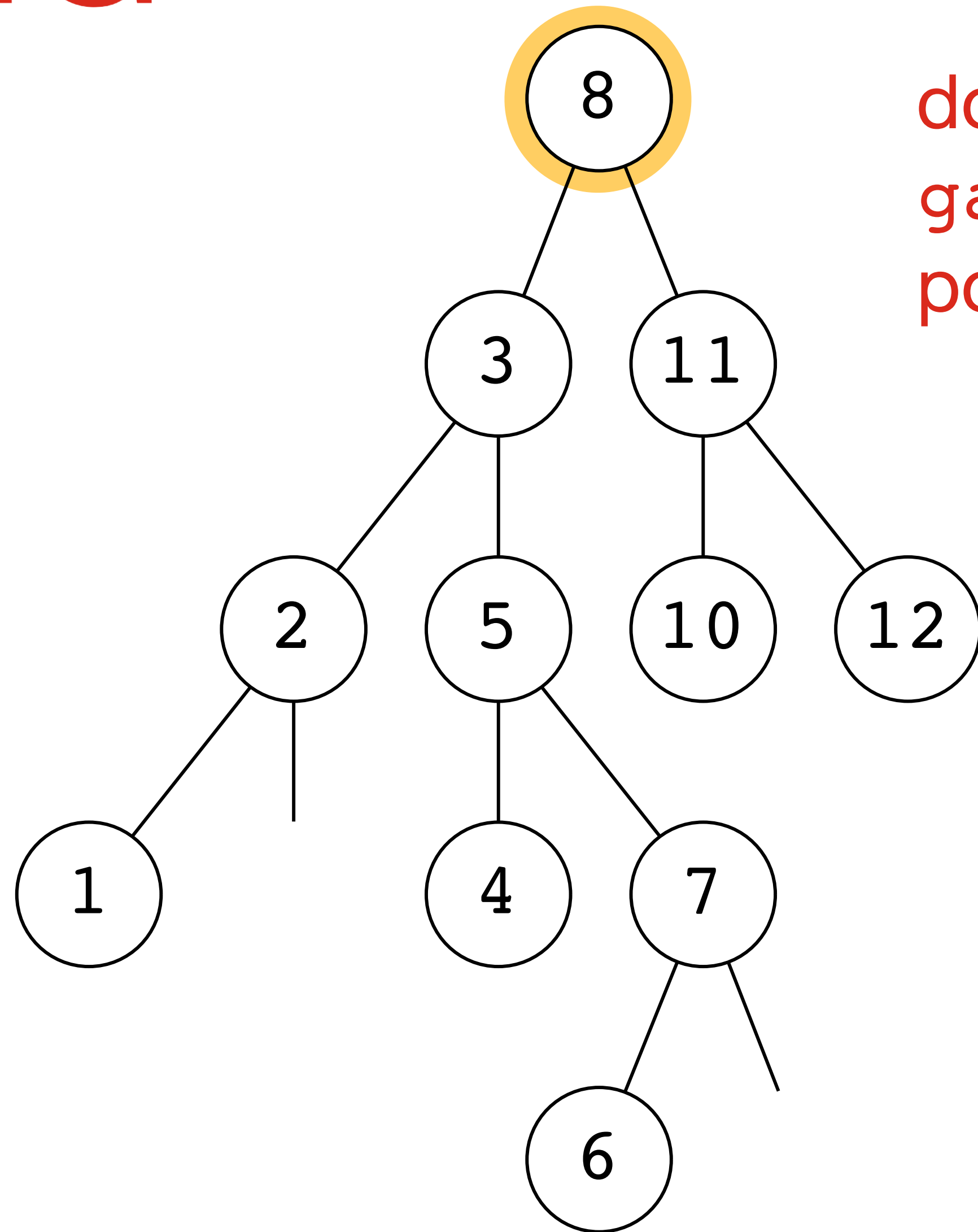
Déséquilibre en 10



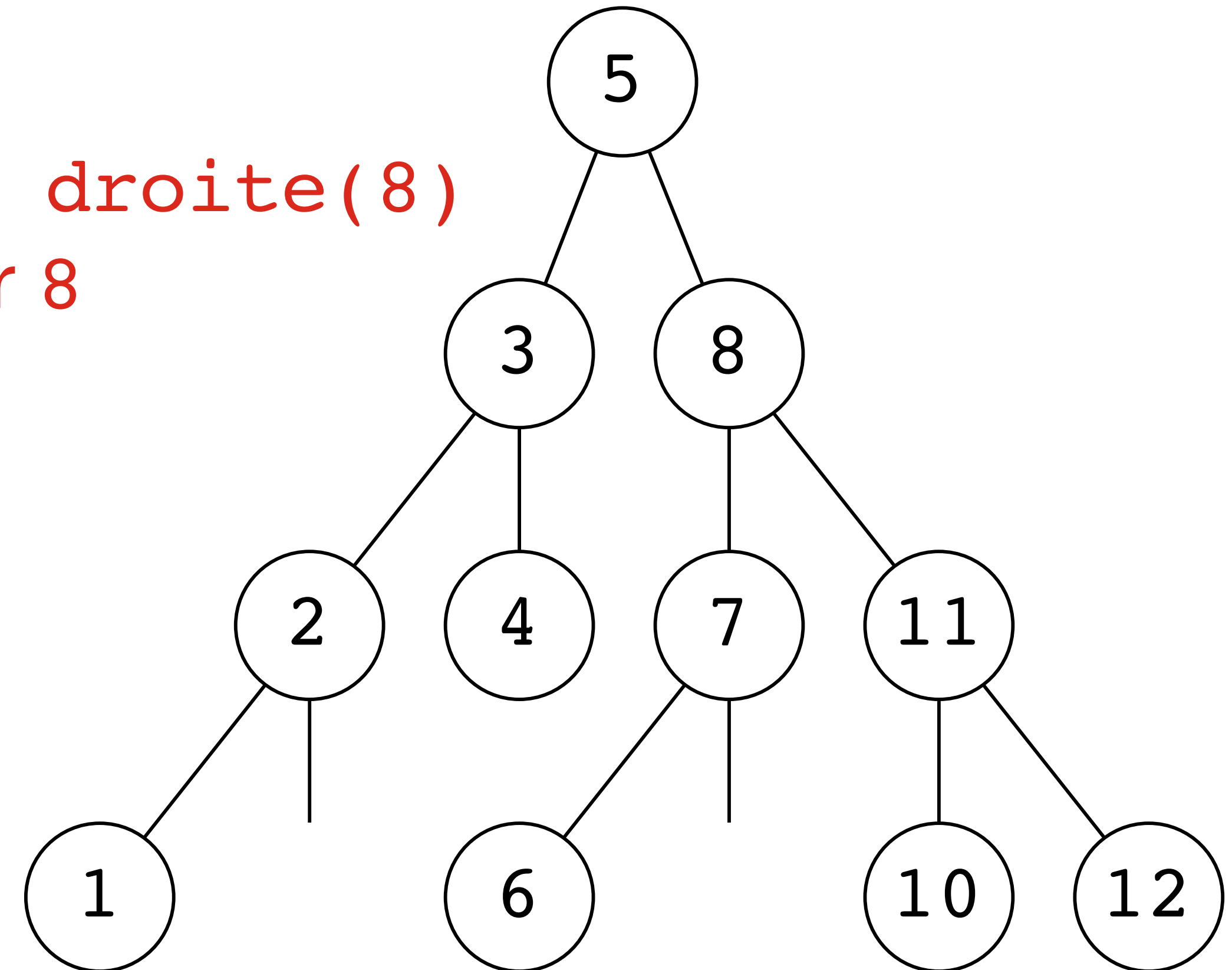
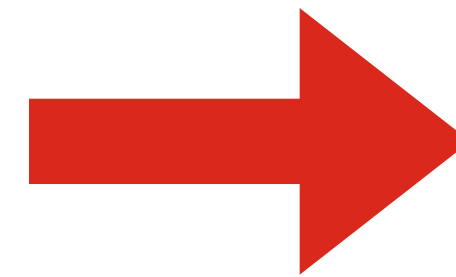
rotation_gauche(10)



Déséquilibre en 8



double rotation :
gauche (3) puis droite (8)
pour ré-équilibrer 8



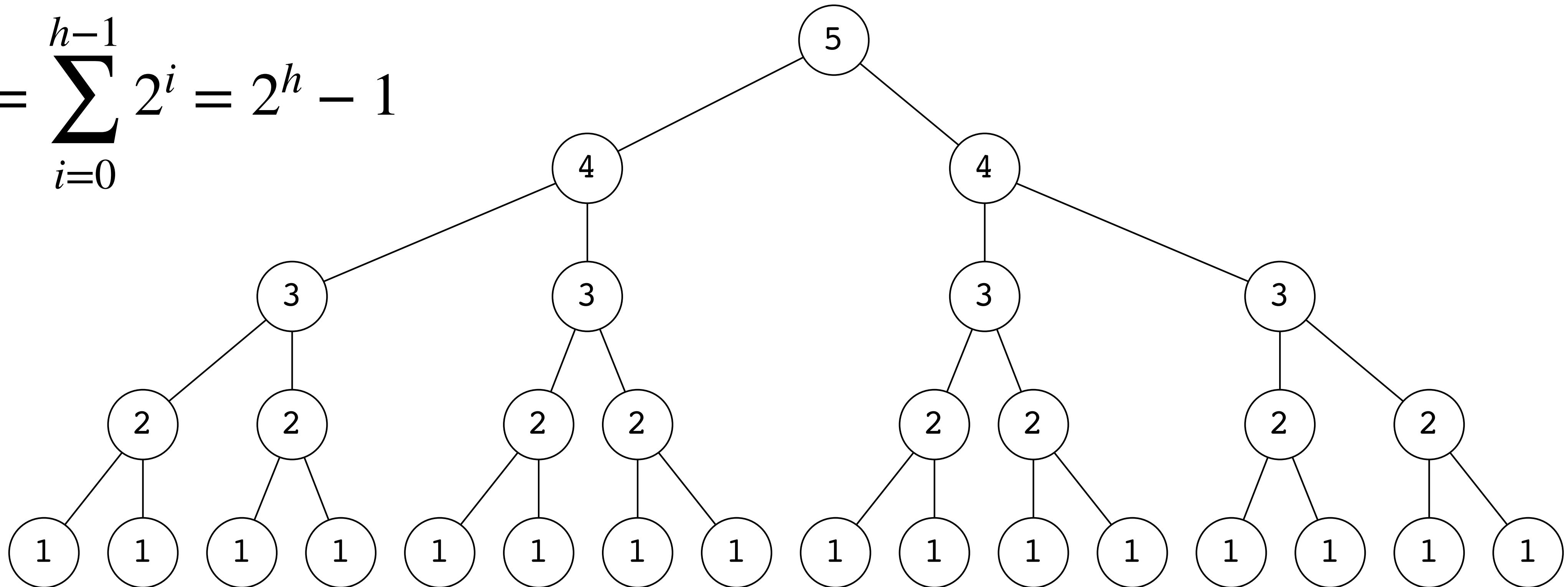


- Comme pour tout ABR, la complexité des opérations dépend de la profondeur du noeud cherché / inséré / supprimé
- Cette profondeur est bornée par la hauteur de l'arbre
- Etudions la relation $n(h)$ entre cette hauteur h et le nombre n d'éléments stockés dans l'arbre
 - Dans le meilleur cas
 - Dans le pire cas



- L'arbre binaire est complet, donc parfaitement équilibré en tous ses noeuds

$$n(h) = \sum_{i=0}^{h-1} 2^i = 2^h - 1$$

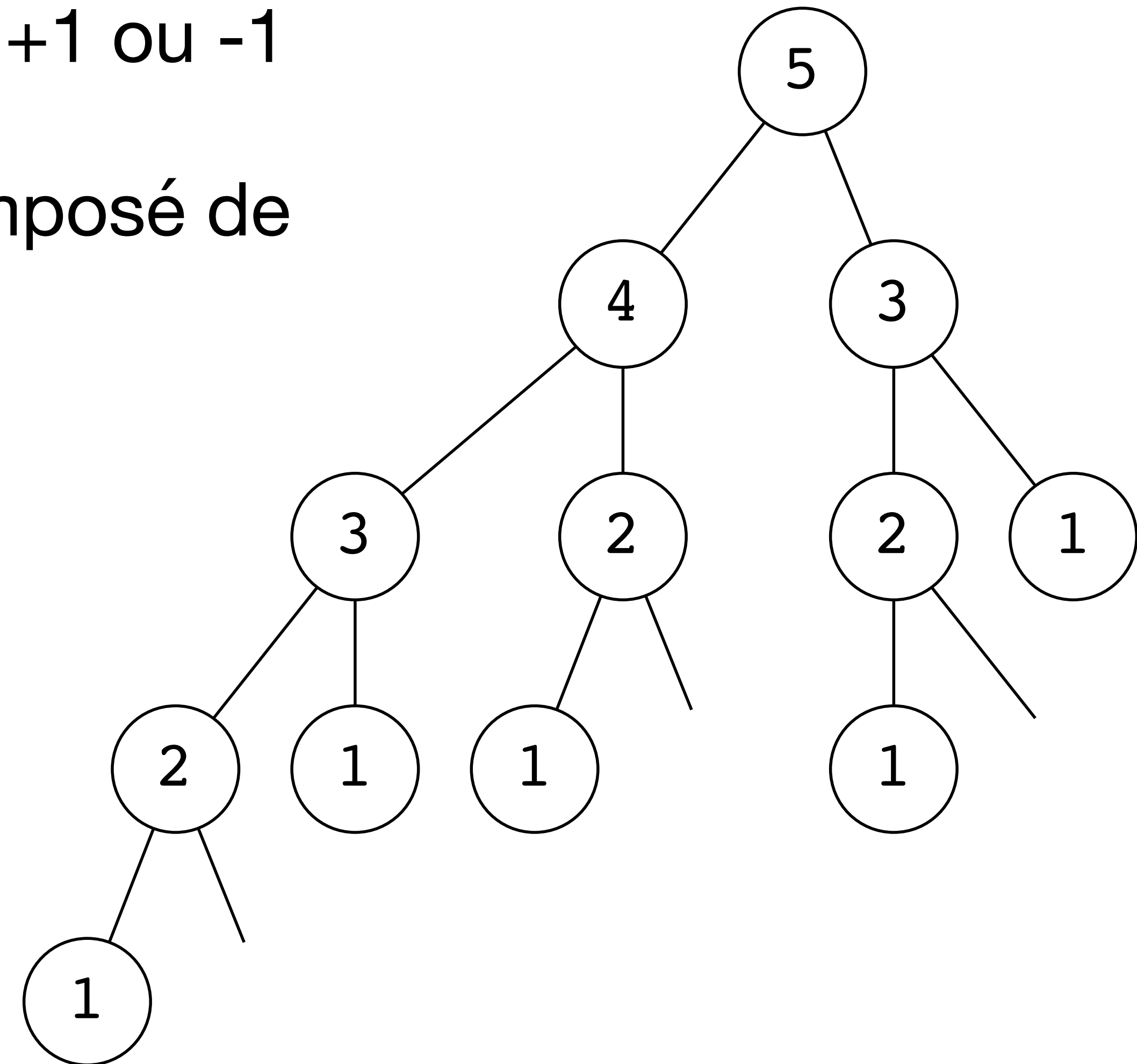


On utilise ici les hauteurs comme label des noeuds

Pire cas



- Tous les noeuds ont un déséquilibre de +1 ou -1
- Le pire arbre de hauteur h est donc composé de
 - Sa racine
 - Le pire arbre de hauteur $h-1$ d'un côté
 - Le pire arbre de hauteur $h-2$ de l'autre



On utilise ici les hauteurs comme label des noeuds

Pire cas (2)



- Le nombre $n(h)$ d'éléments du pire arbre AVL de hauteur h se calcule donc en résolvant la récurrence

$$n(0) = 0 \qquad n(1) = 1$$

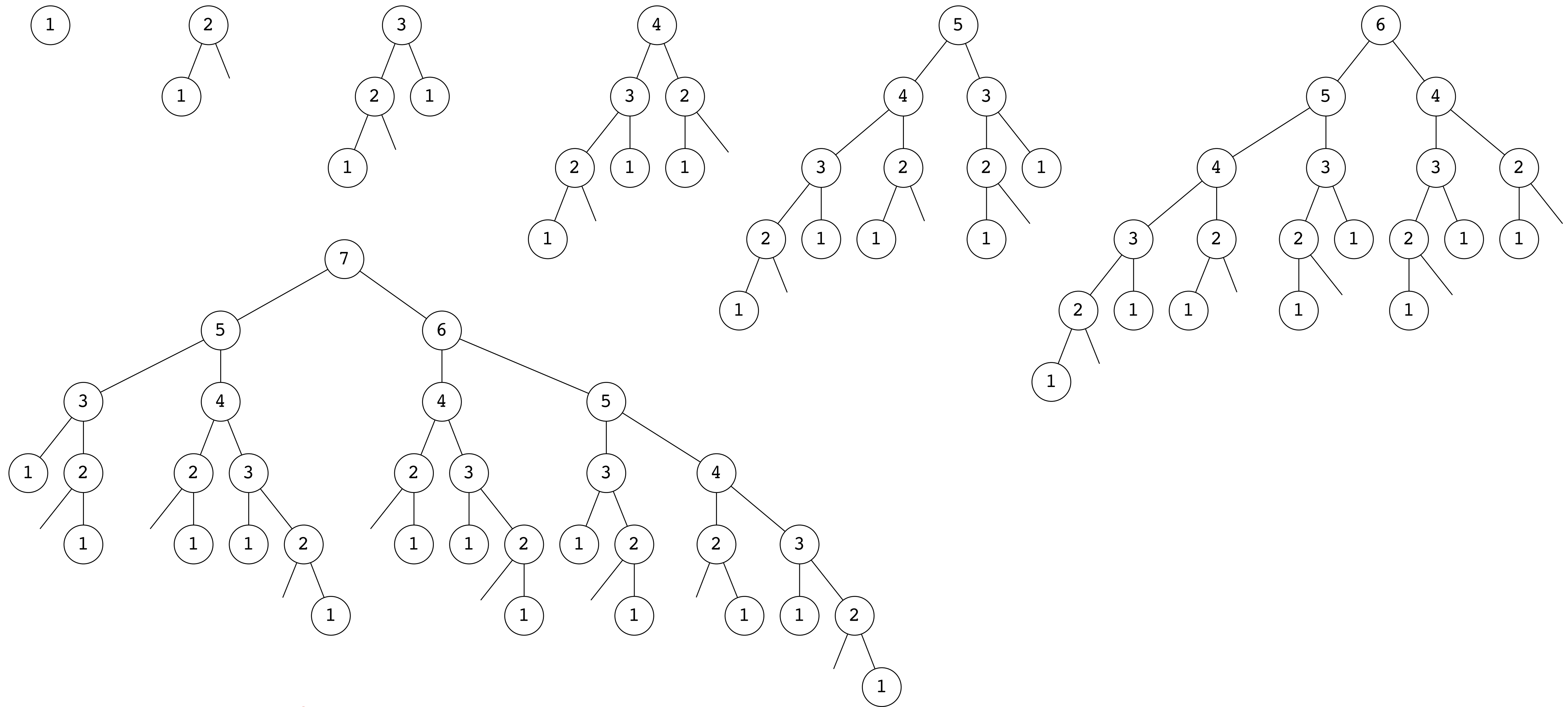
$$n(h) = 1 + n(h - 1) + n(h - 2)$$

- Les premières valeurs de cette suite sont 0, 1, 2, 4, 7, 12, 20, 33, 54, 88, ...
i.e. la suite des nombres de Fibonacci moins 1

- $n(h)$ croît donc comme ϕ^h avec $\phi = \frac{1 + \sqrt{5}}{2}$

- Réciproquement, $h(n) \approx 1.4404 \cdot \log_2(n)$

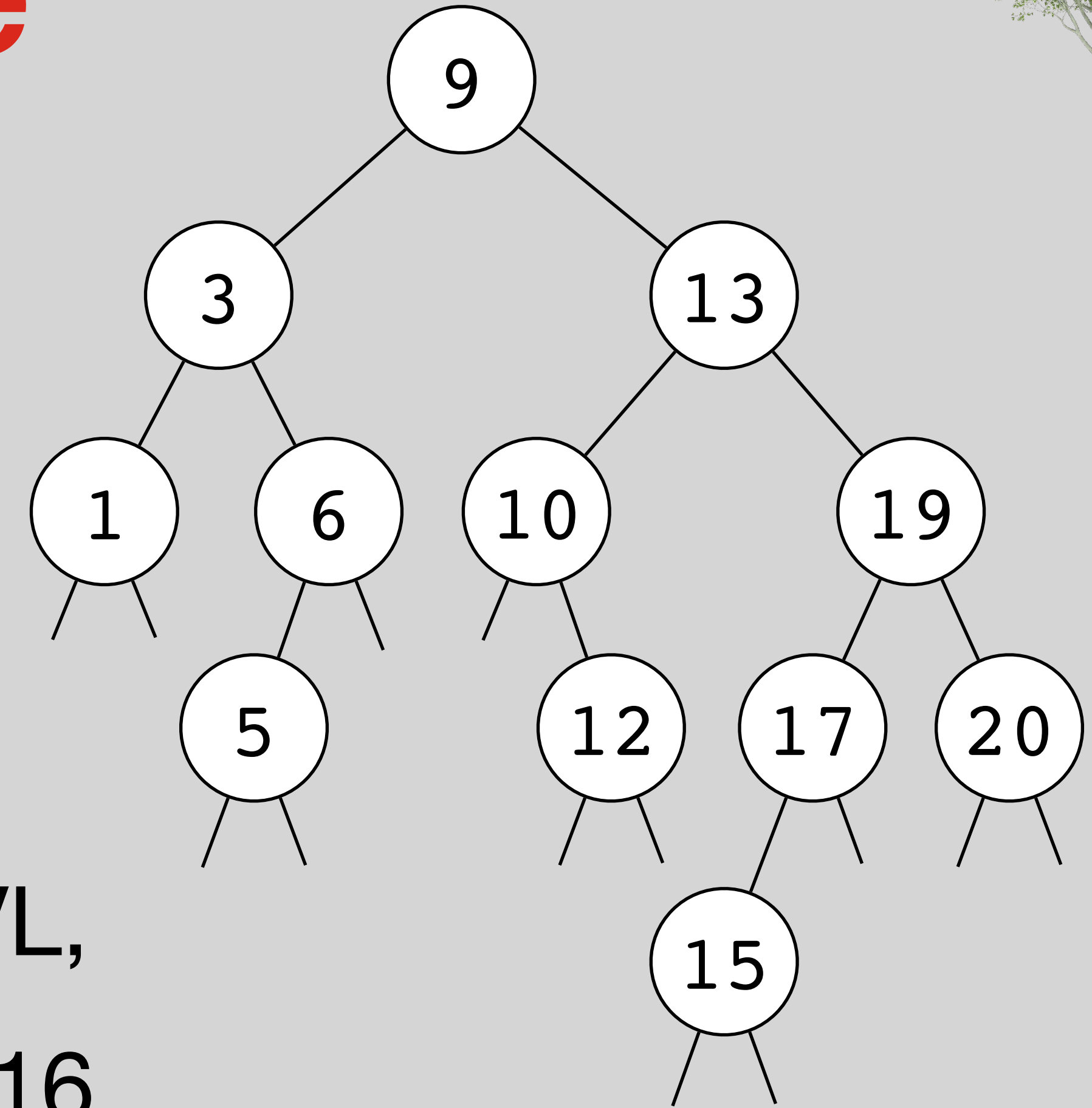
Les pires arbres AVL...



Exercice



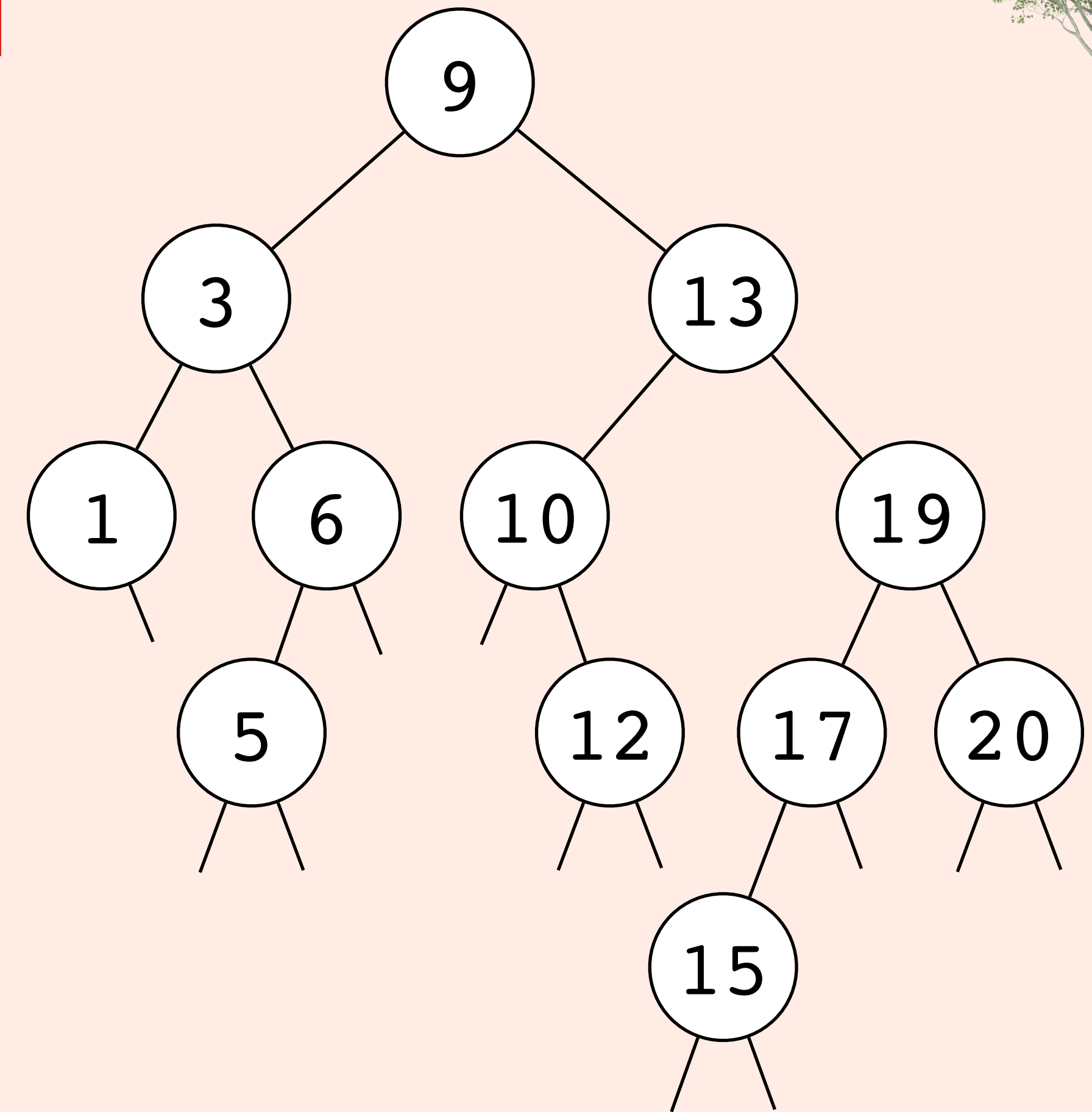
- Cet arbre est-il équilibré ?
- Cet arbre est-il un ABR ?
- Quel élément a pour rang 7 ?
- Quel est le rang de la clé 14 ?
- En considérant cet arbre comme un AVL,
 - Le dessiner après insertion d'une clé 16.
 - Le dessiner après suppression de la clé 10.
 - Le dessiner après suppression de la clé 1.



Solution

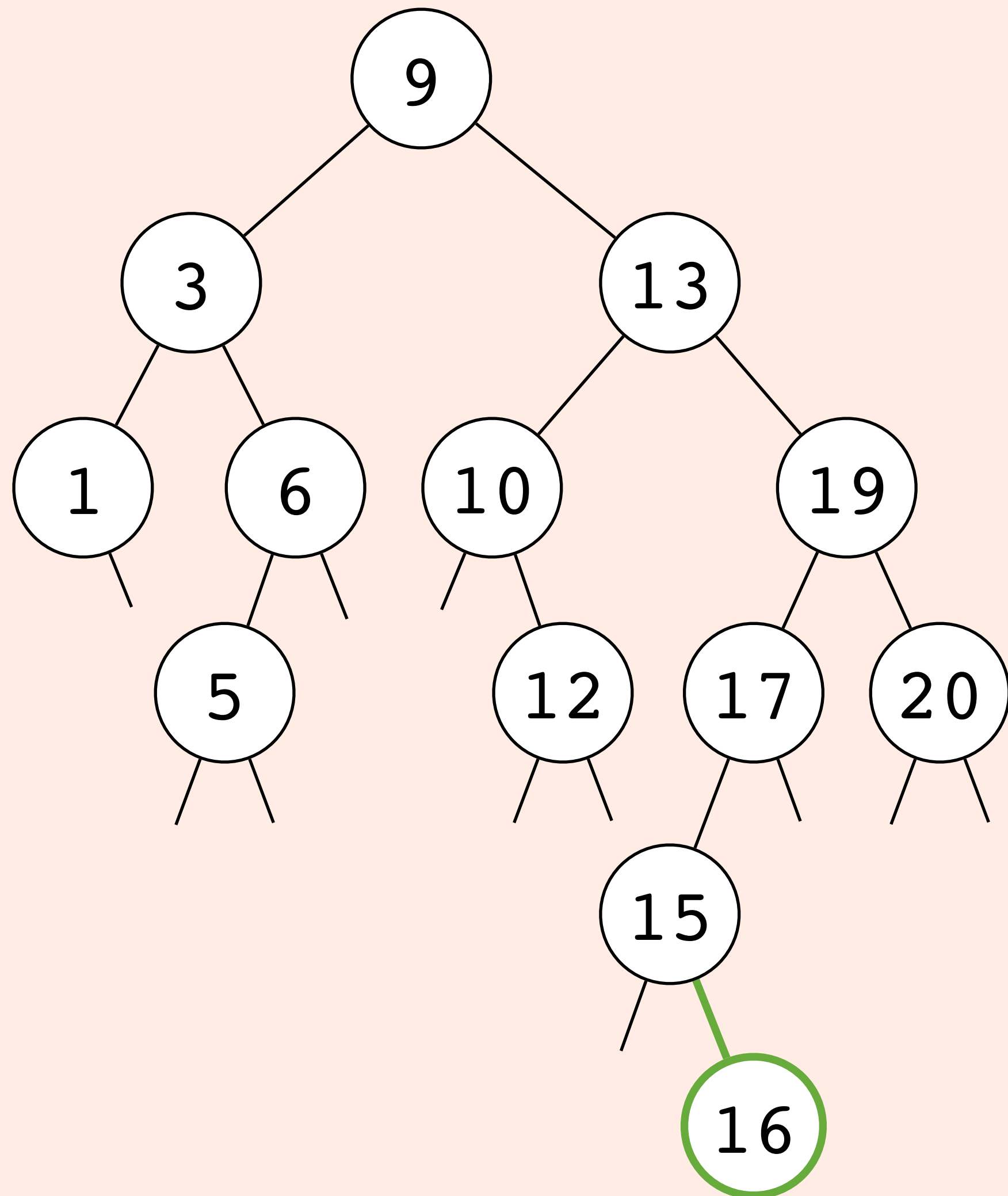


- Cet arbre est-il équilibré ? **Oui**
- Cet arbre est-il un ABR ? **Oui**
- Quel élément a pour rang 7 ? **13**
- Quel est le rang de la clé 14 ? **8**



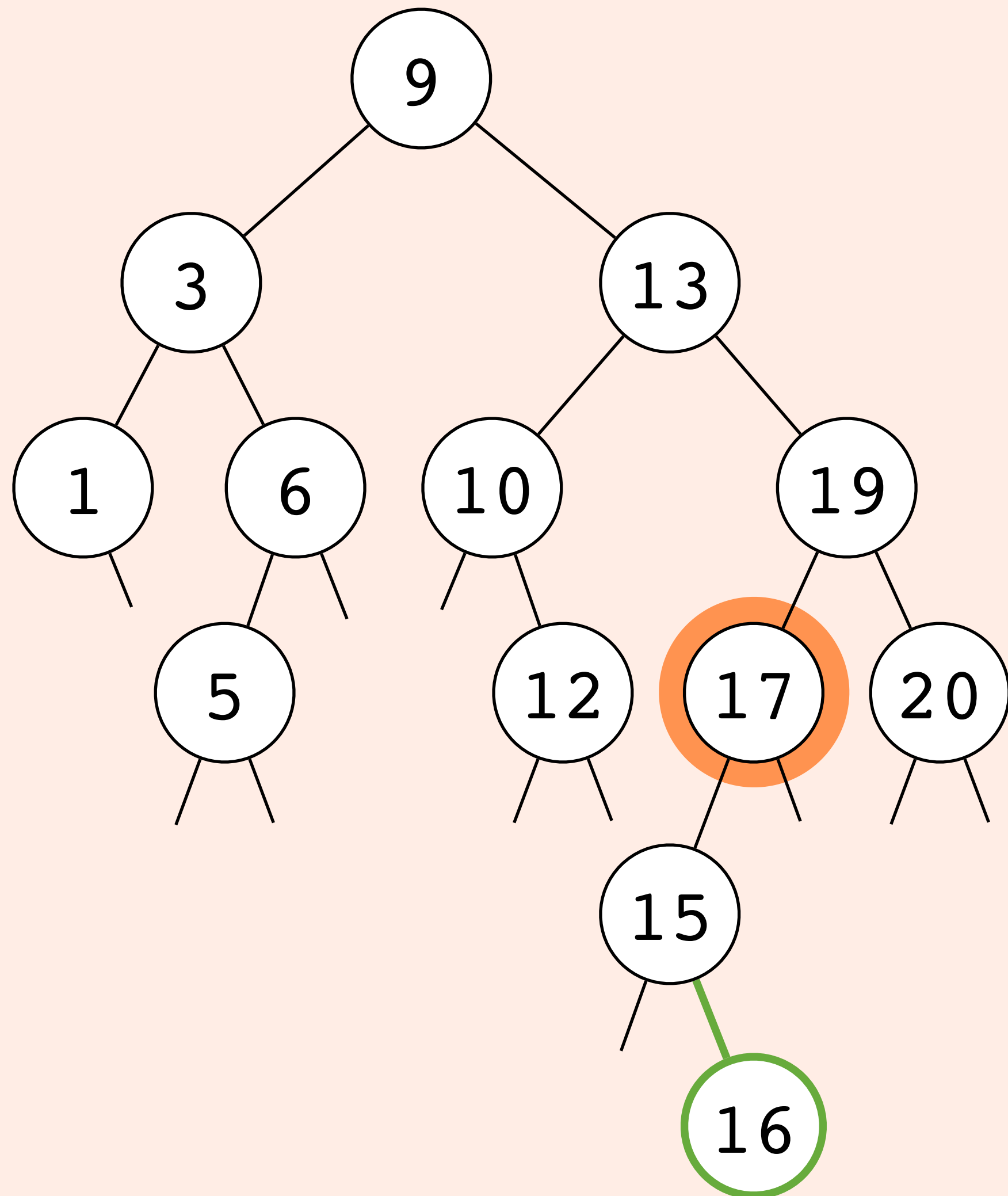
Solution

Insertion de 16



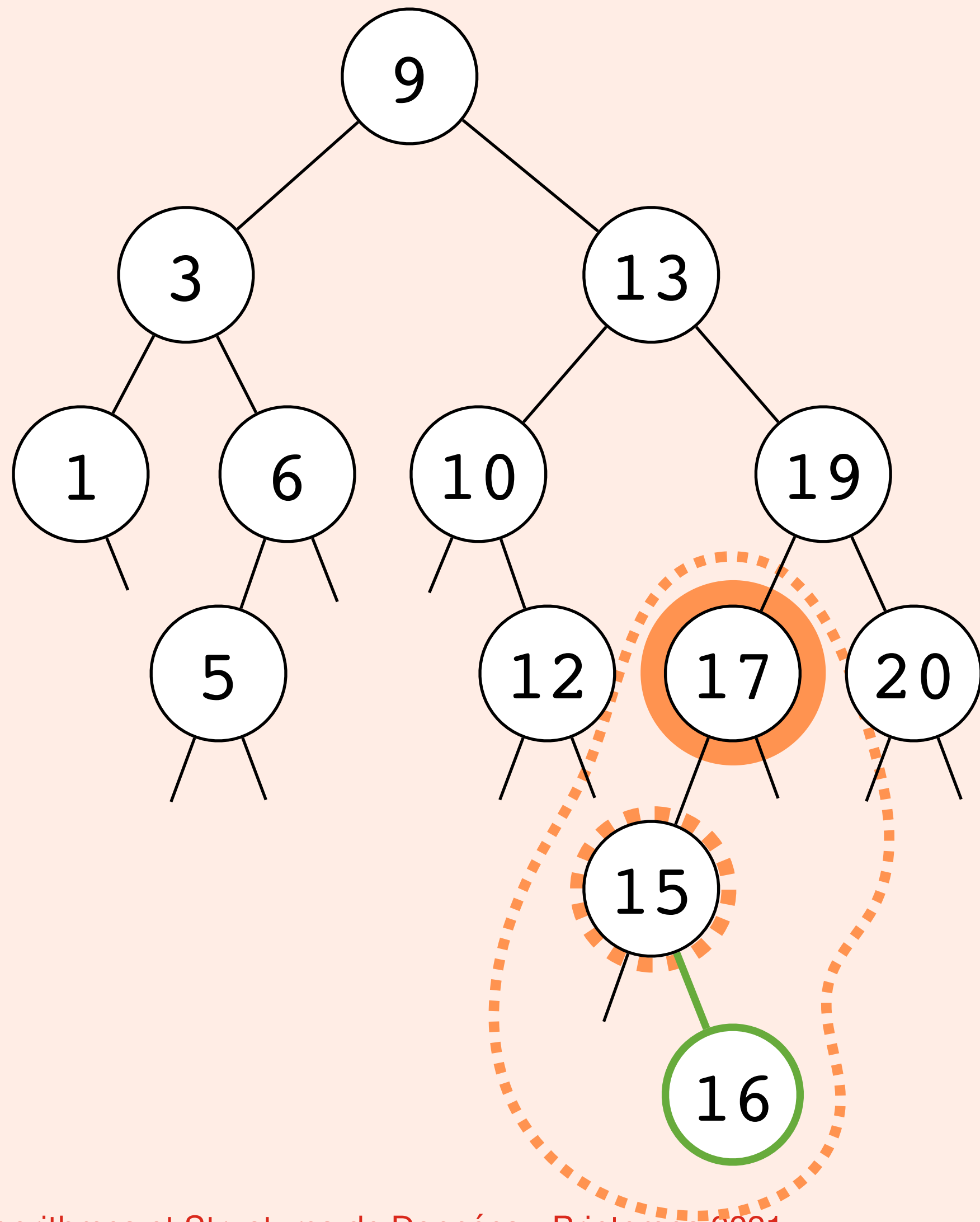
Solution

Insertion de 16



Solution

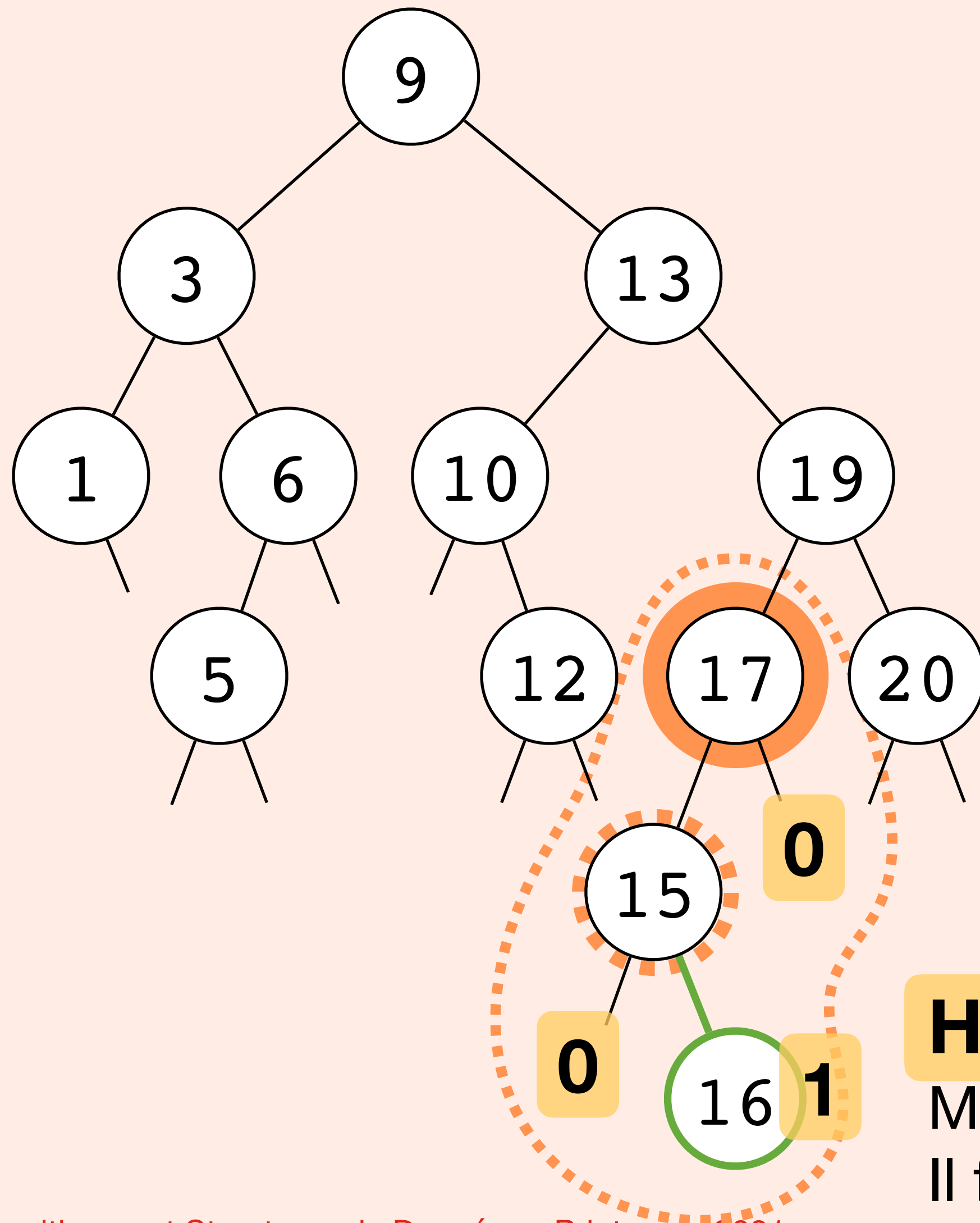
Insertion de 16





Solution

Insertion de 16



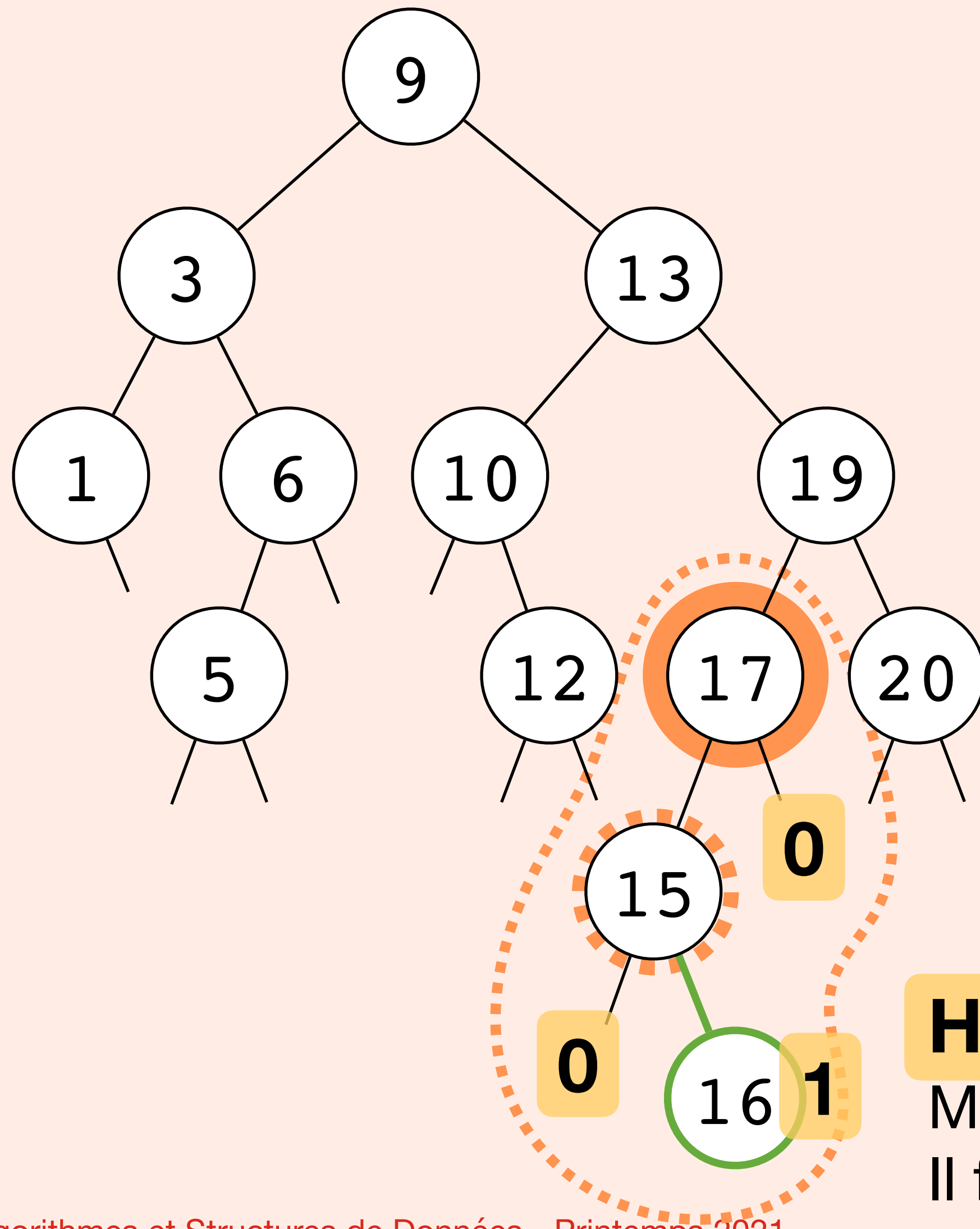
Hauteurs symétriques

Même profondeur à gauche et à droite.
Il faut donc faire une rotation double.

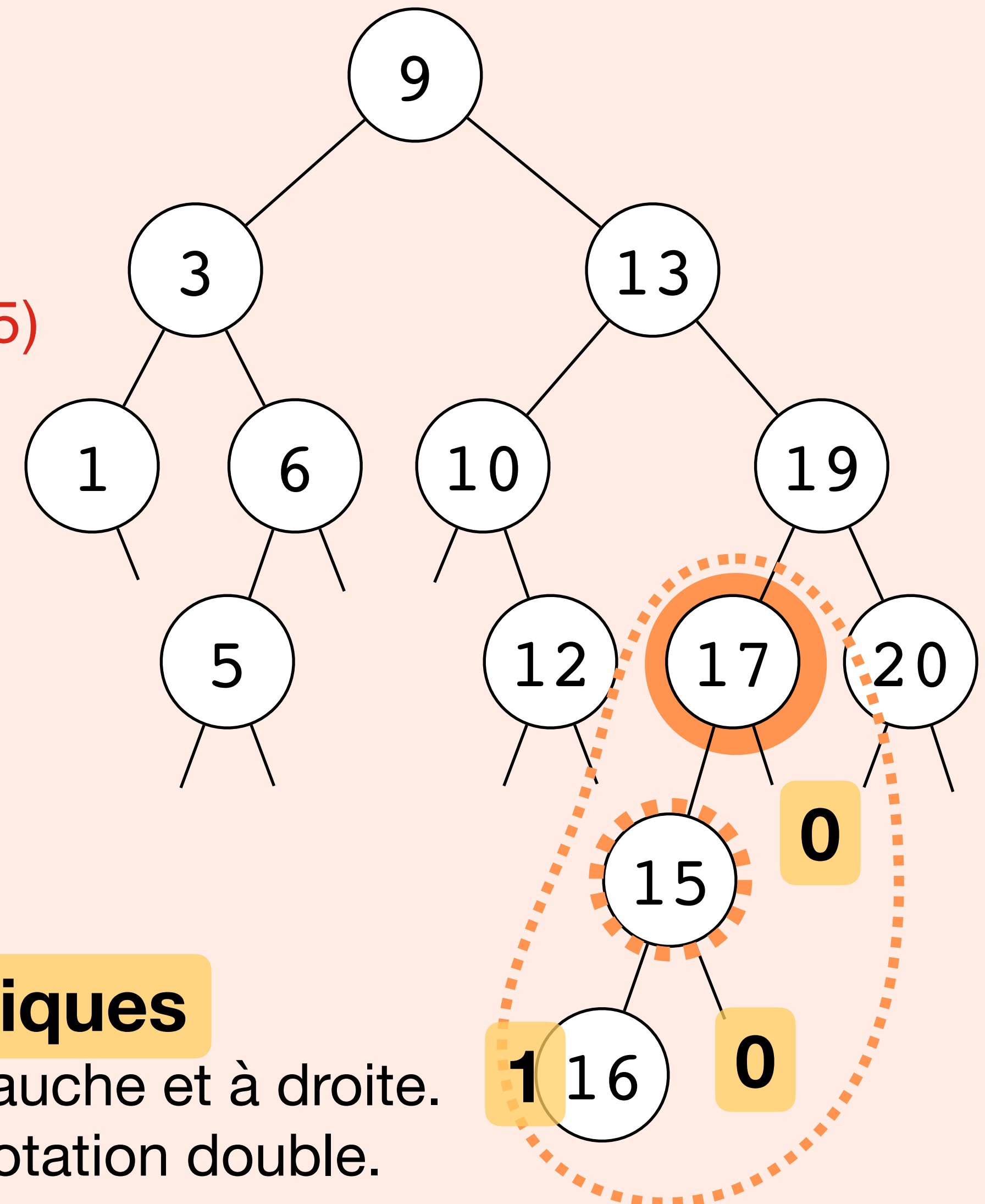
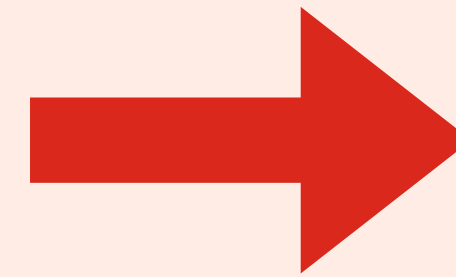


Solution

Insertion de 16



Première rotation
rotation_gauche(15)

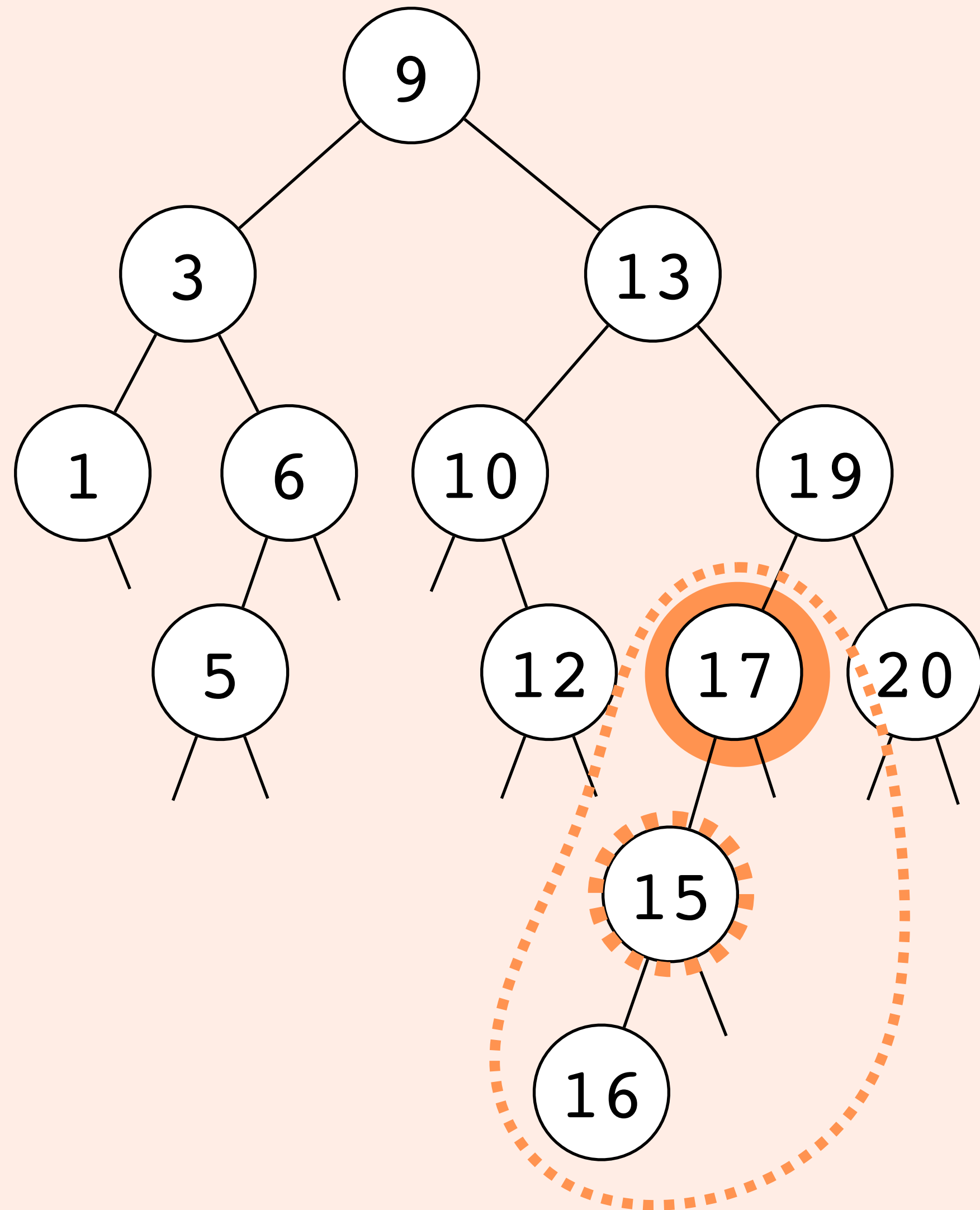


Hauteurs symétriques

Même profondeur à gauche et à droite.
Il faut donc faire une rotation double.

Solution

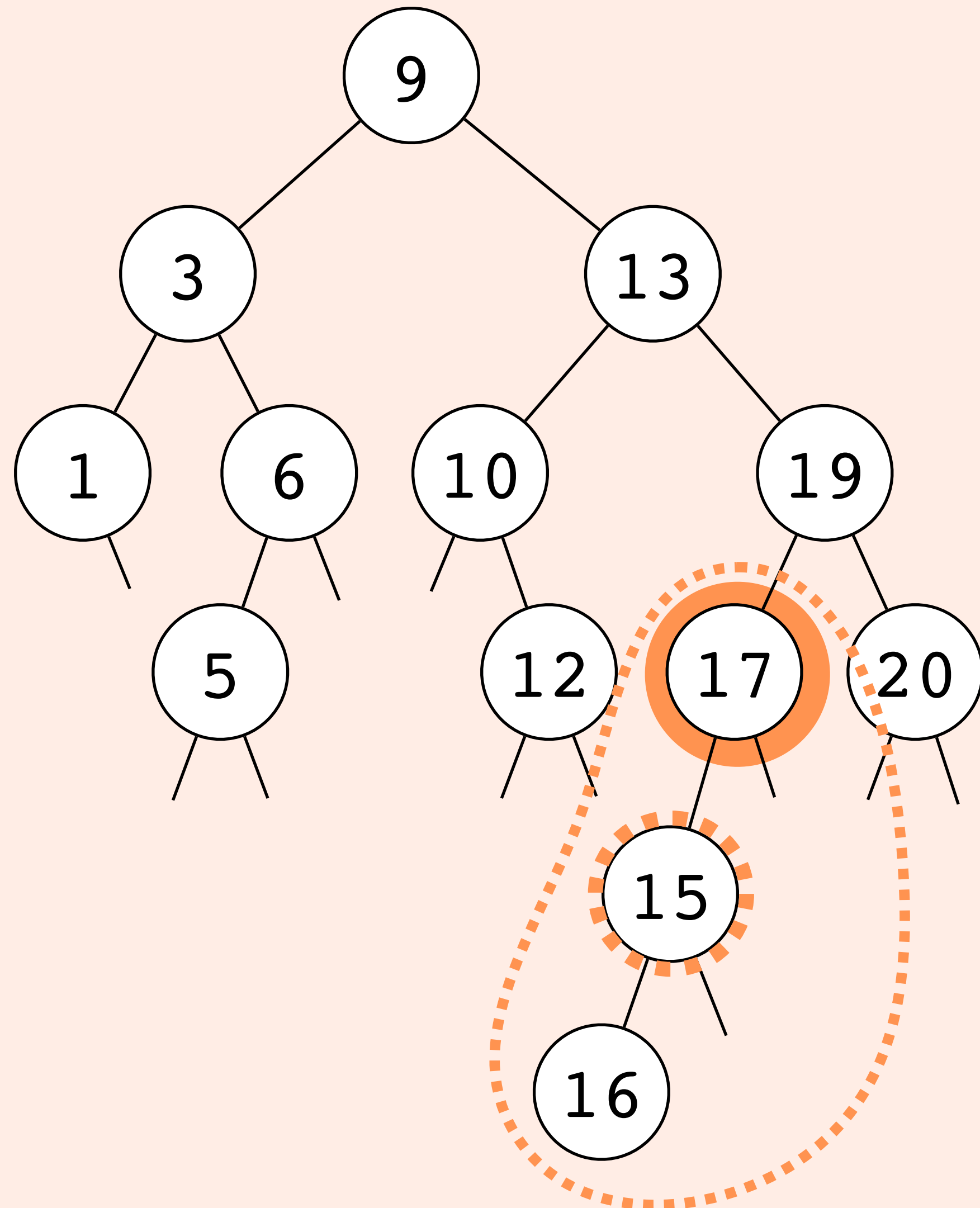
Insertion de 16



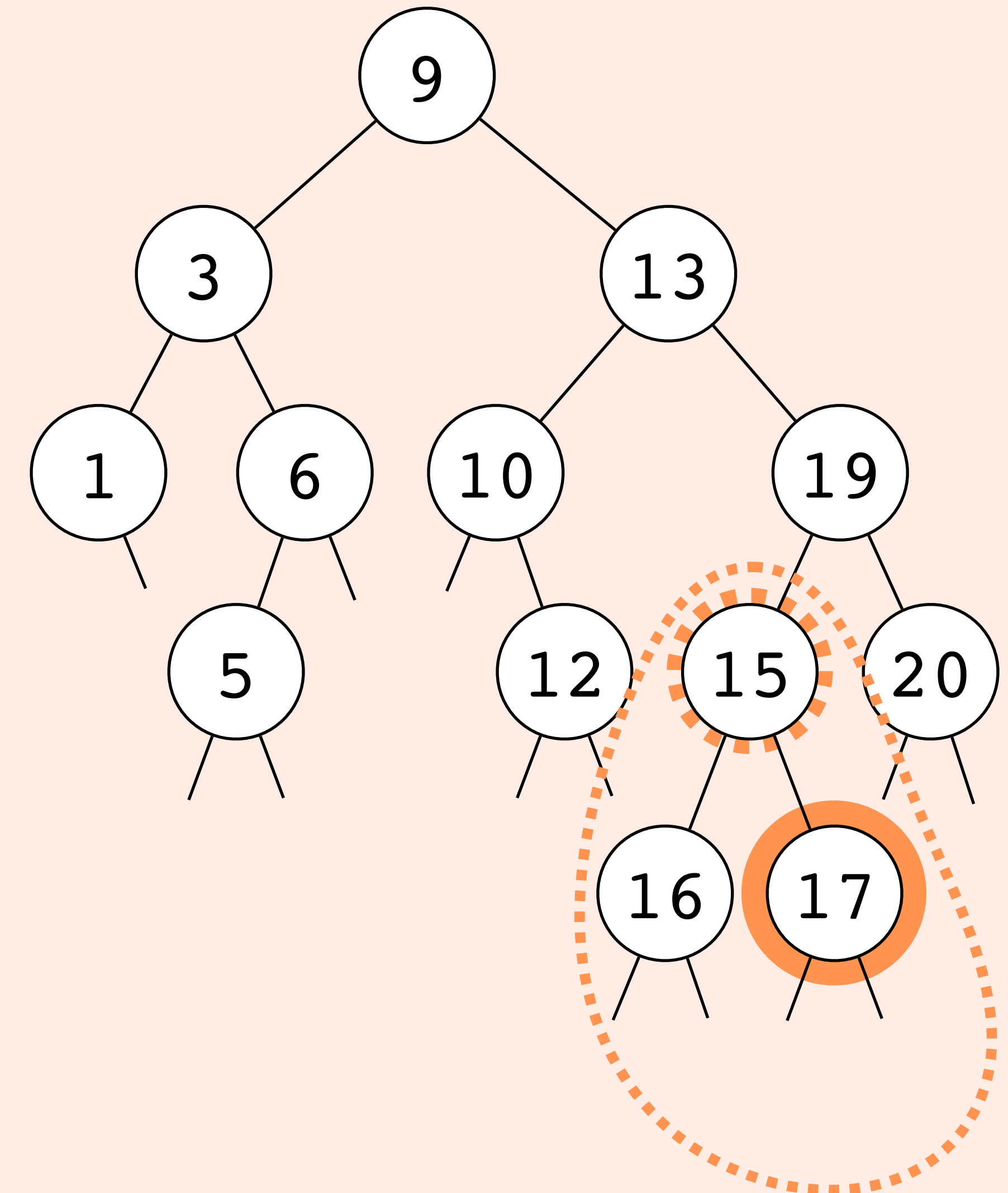
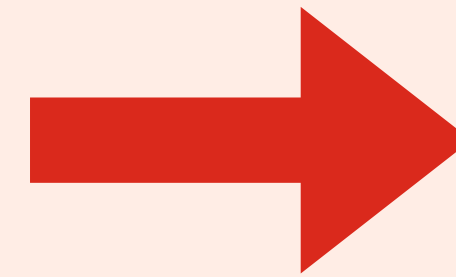


Solution

Insertion de 16



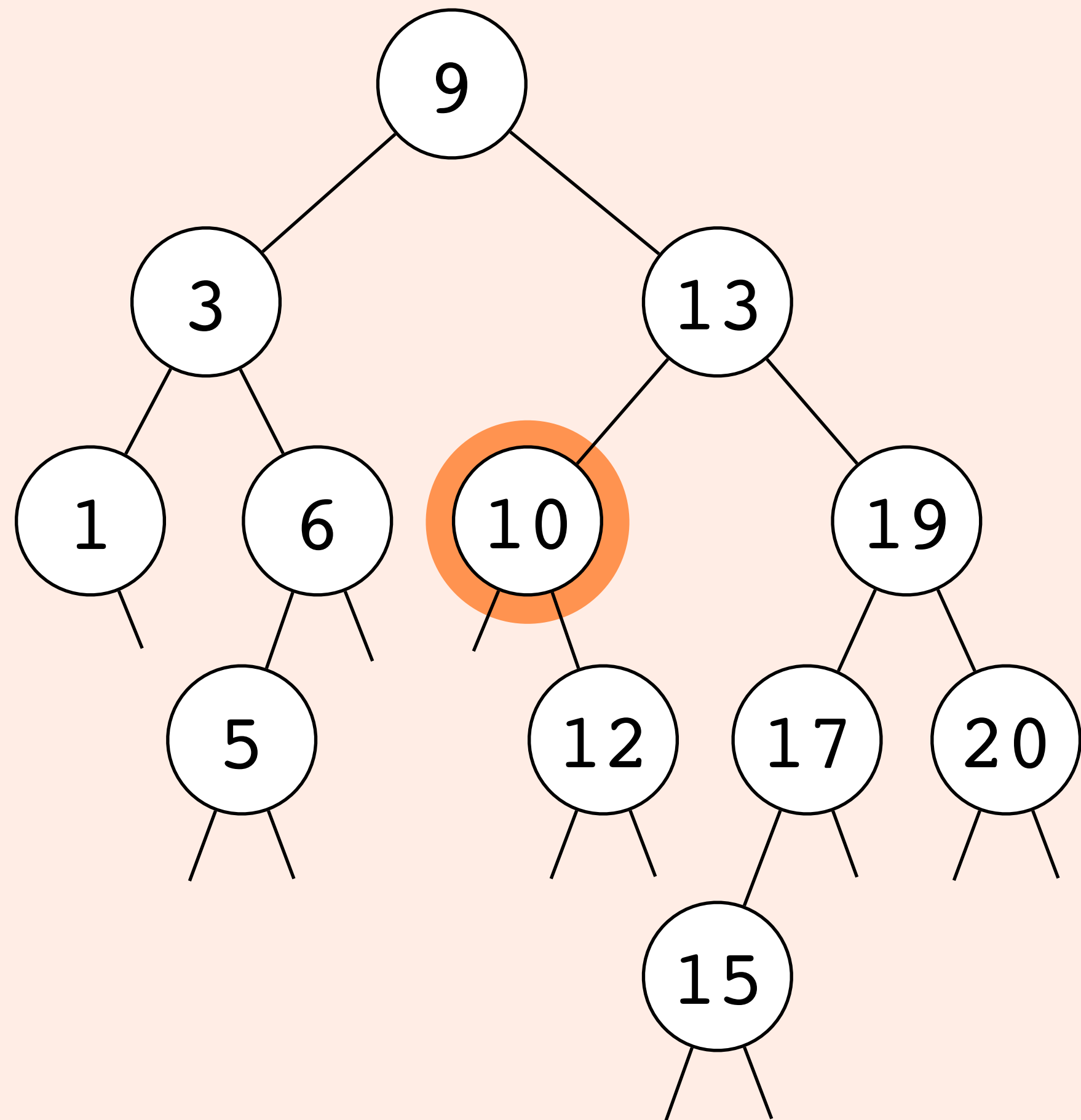
Seconde rotation
rotation_droite(17)



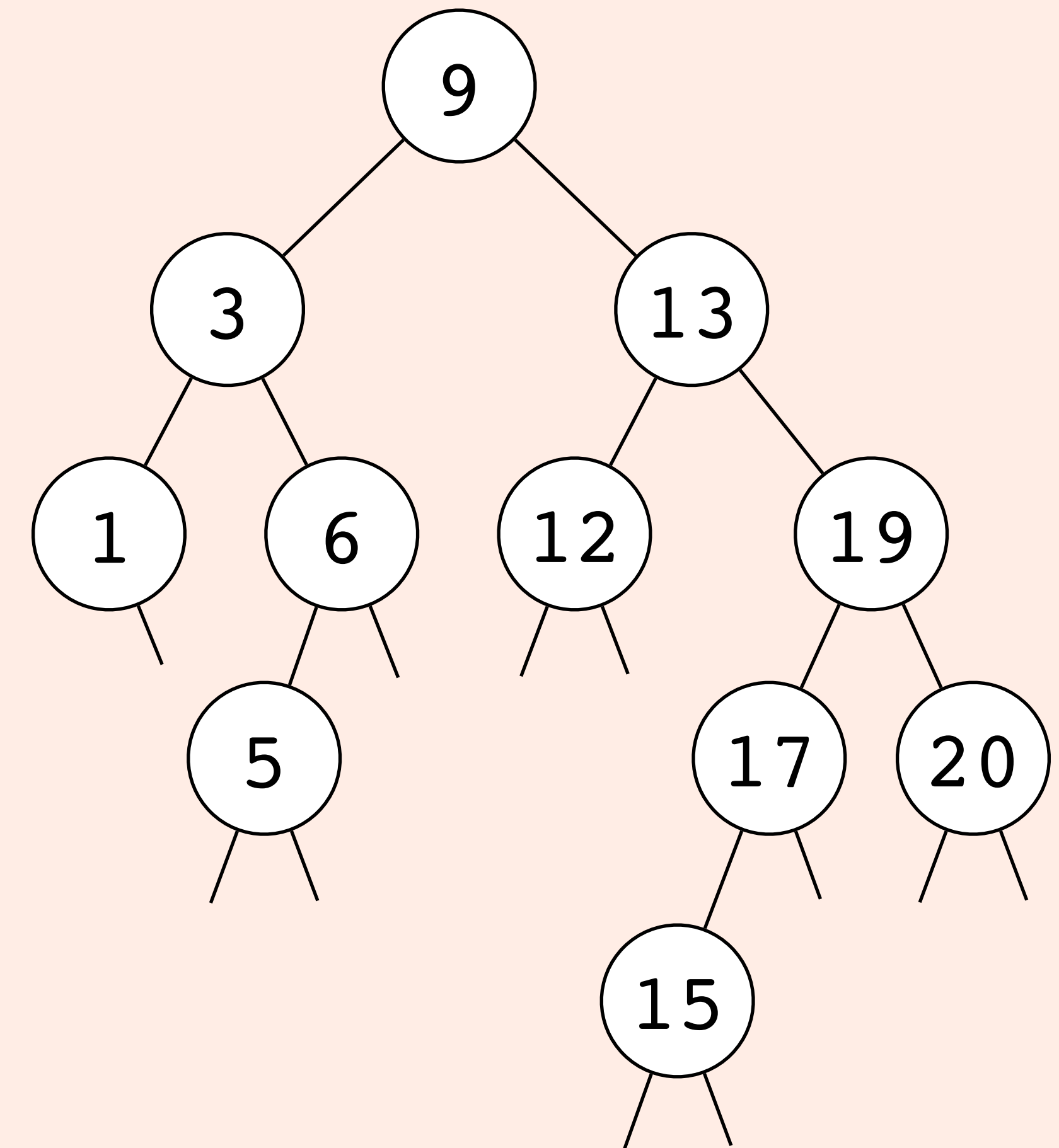
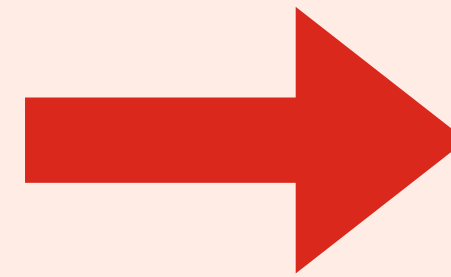


Solution

Suppression de 10



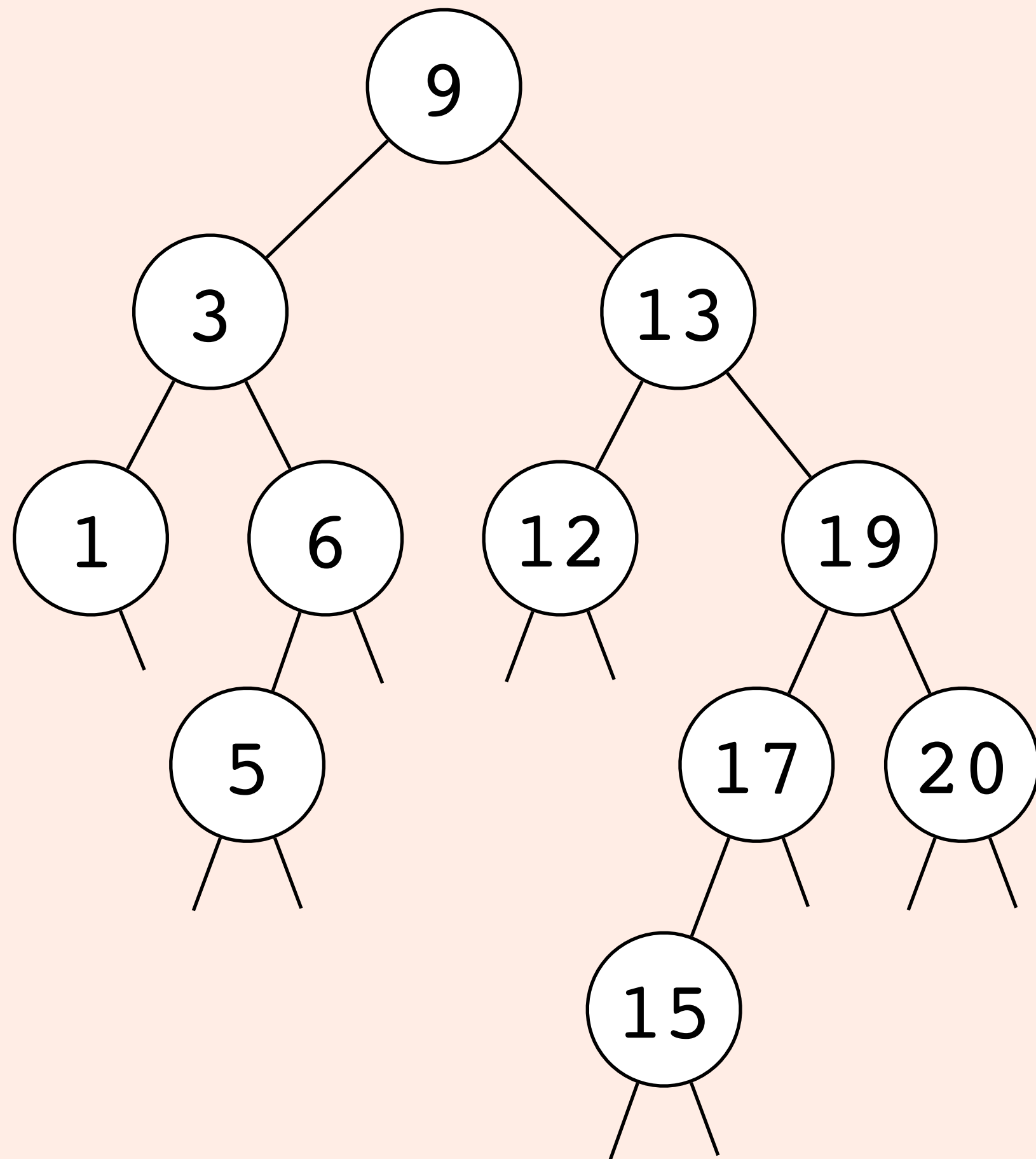
Suppression de l'ABR





Solution

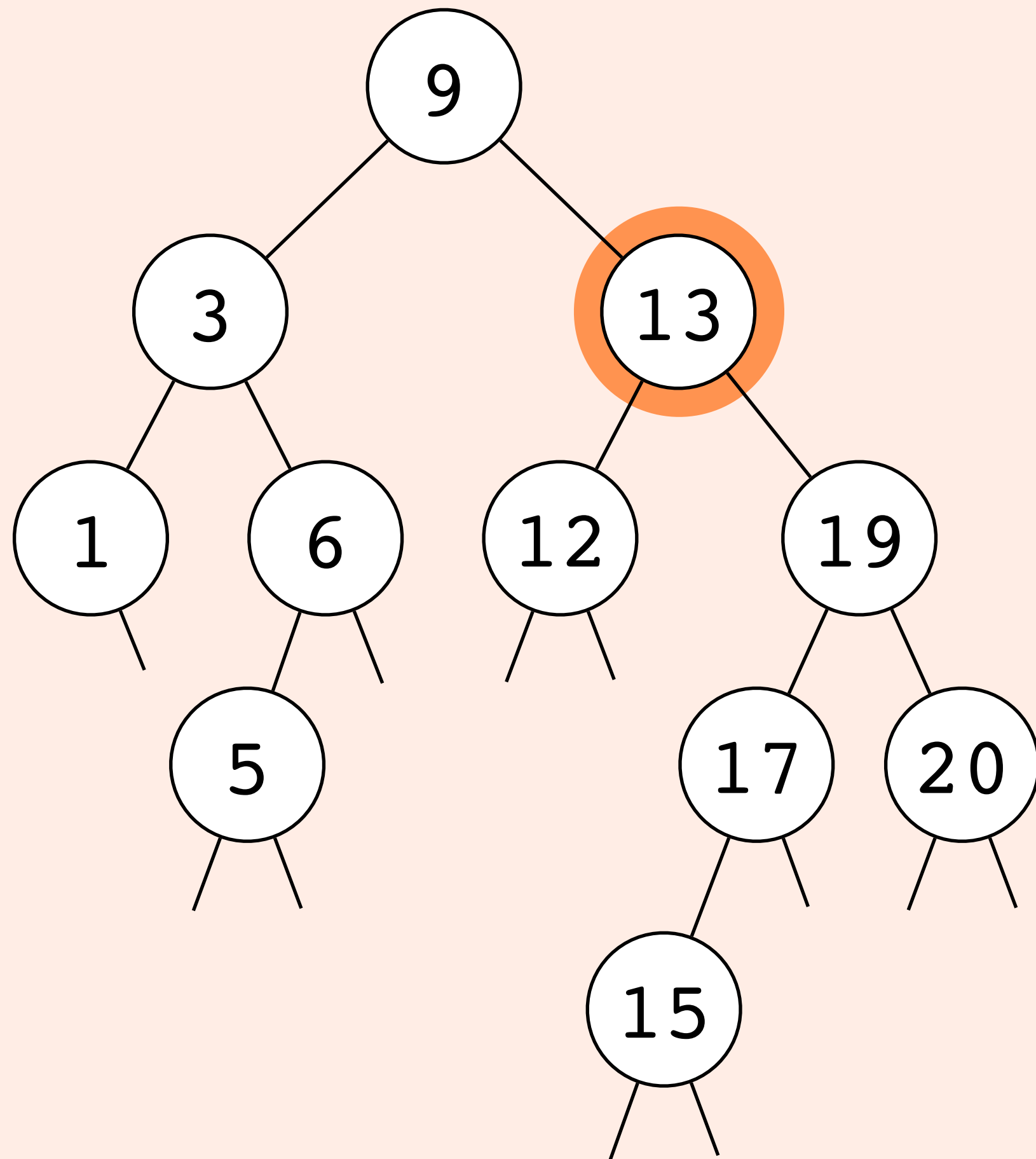
Suppression de 10





Solution

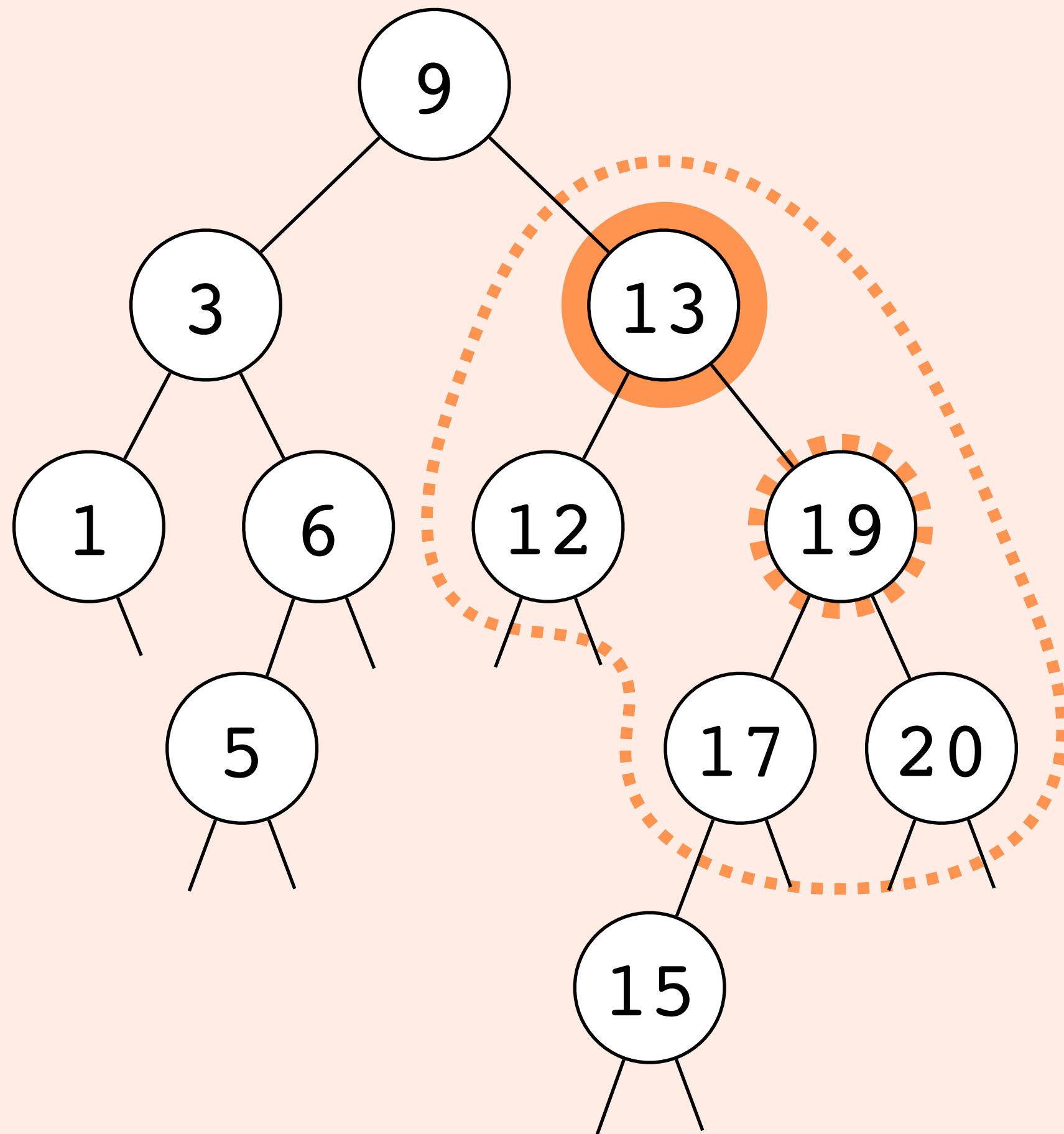
Suppression de 10





Solution

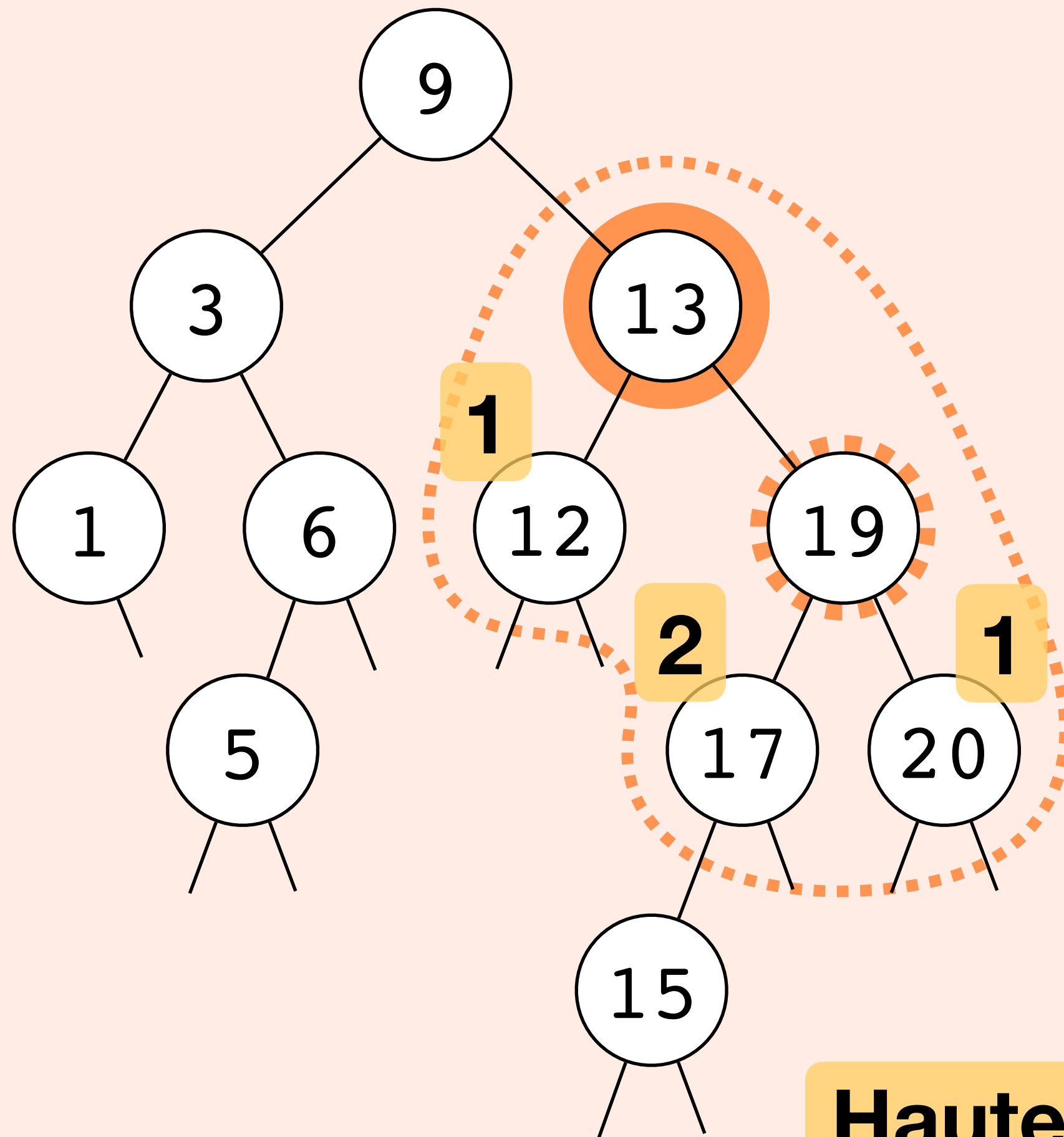
Suppression de 10





Solution

Suppression de 10



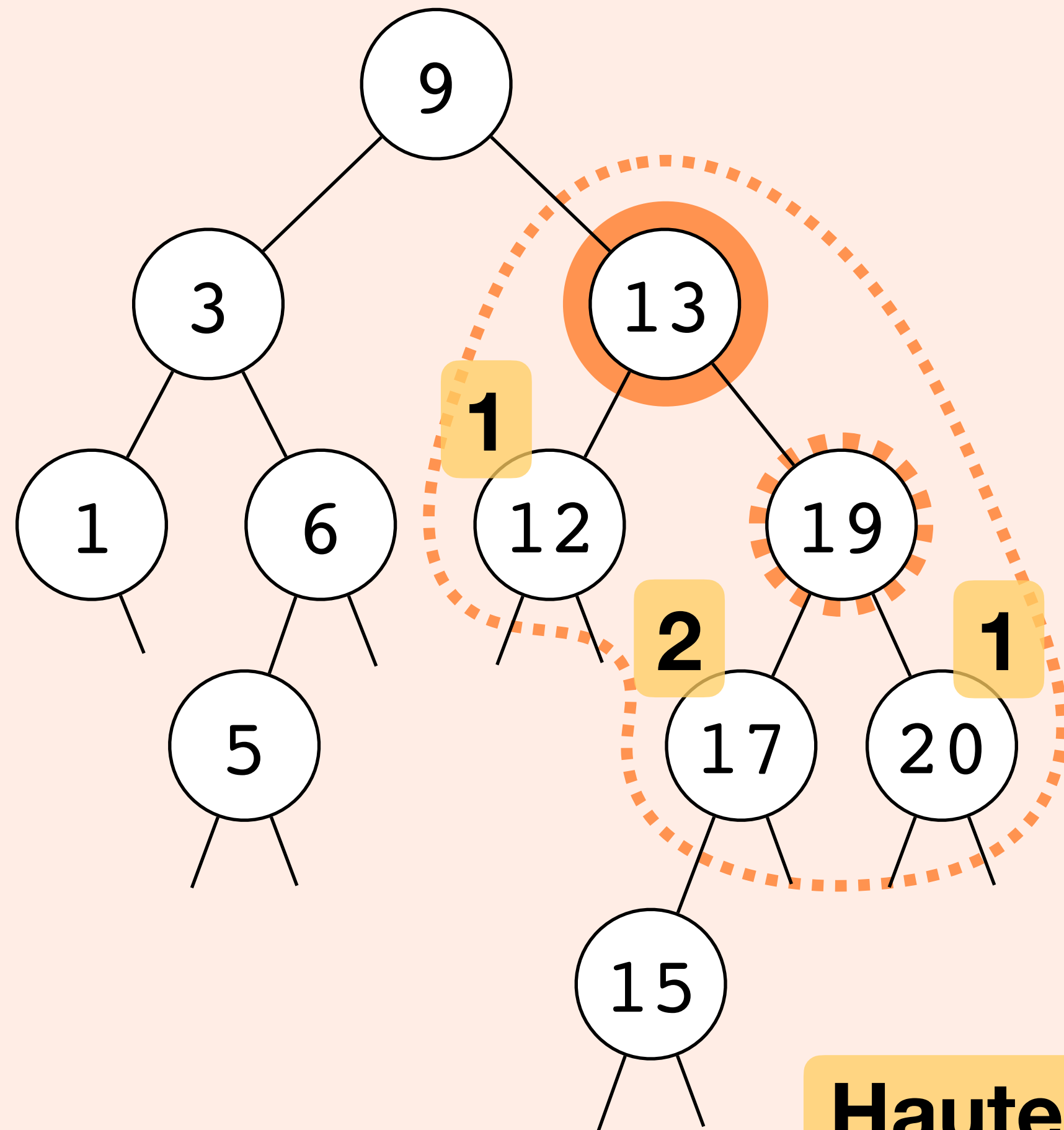
Hauteurs symétriques

Même profondeur à gauche et à droite.
Il faut donc faire une rotation double.

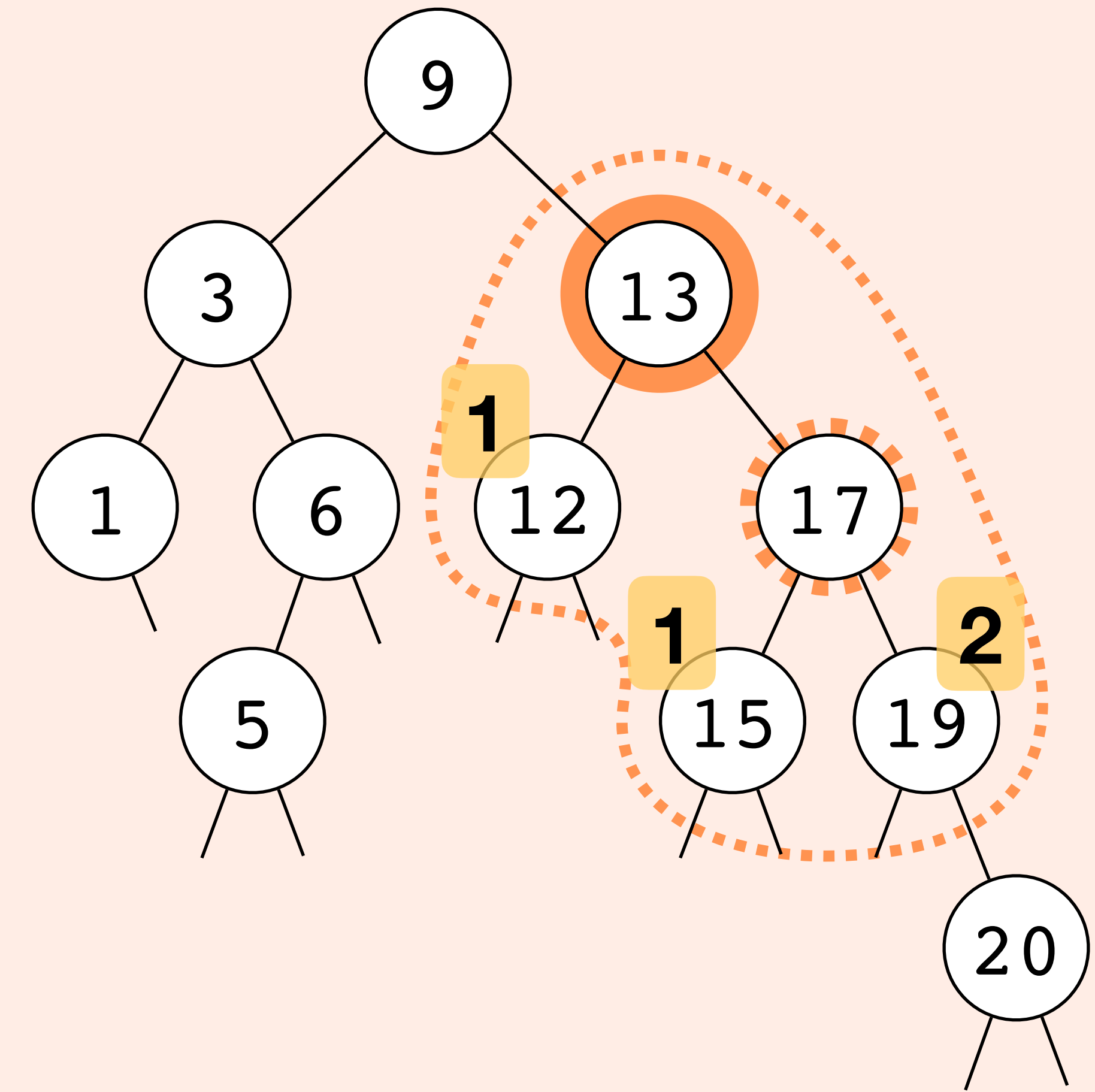


Solution

Suppression de 10



Première rotation
rotation_droite(19)



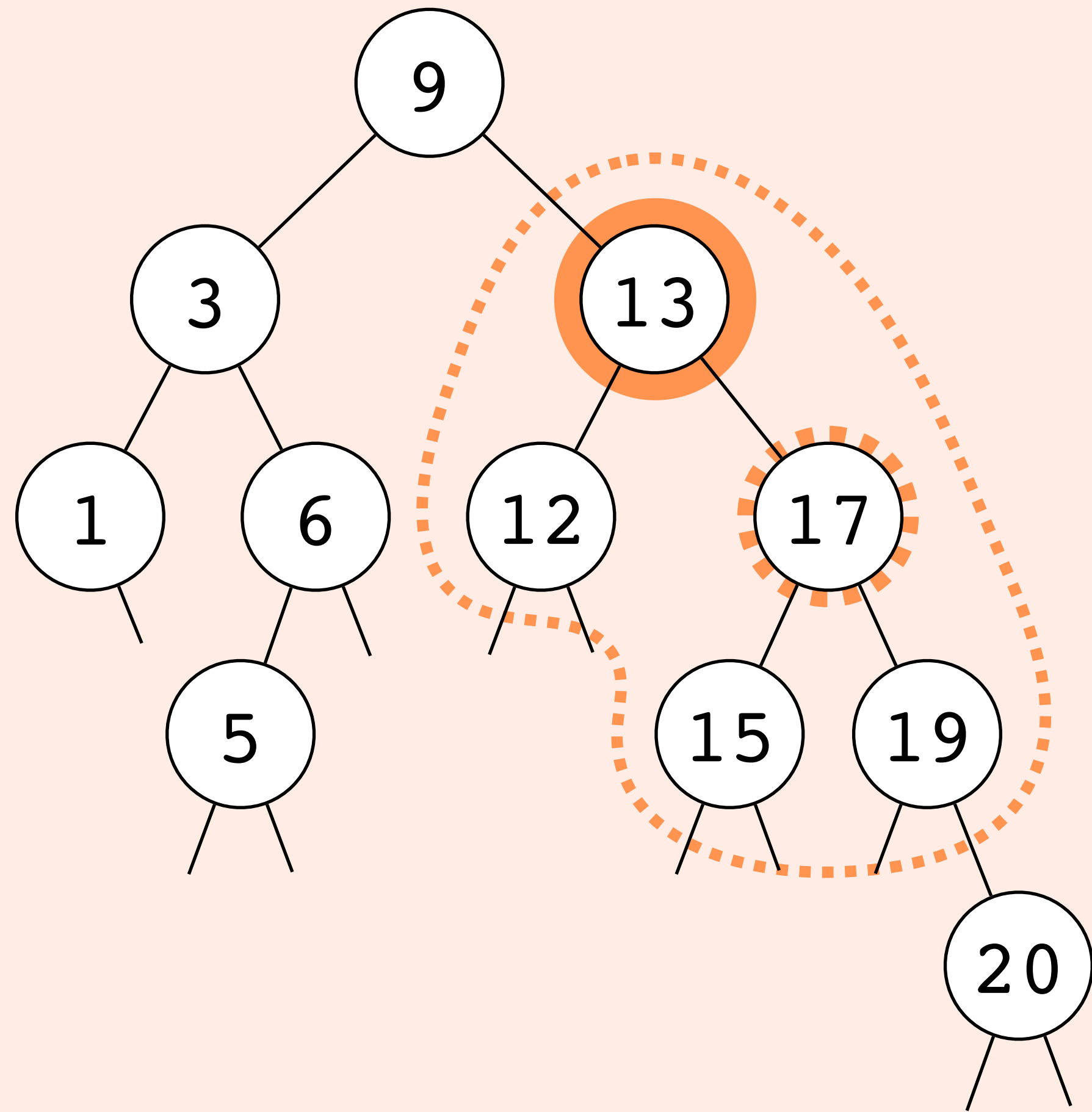
Hauteurs symétriques

Même profondeur à gauche et à droite.
Il faut donc faire une rotation double.



Solution

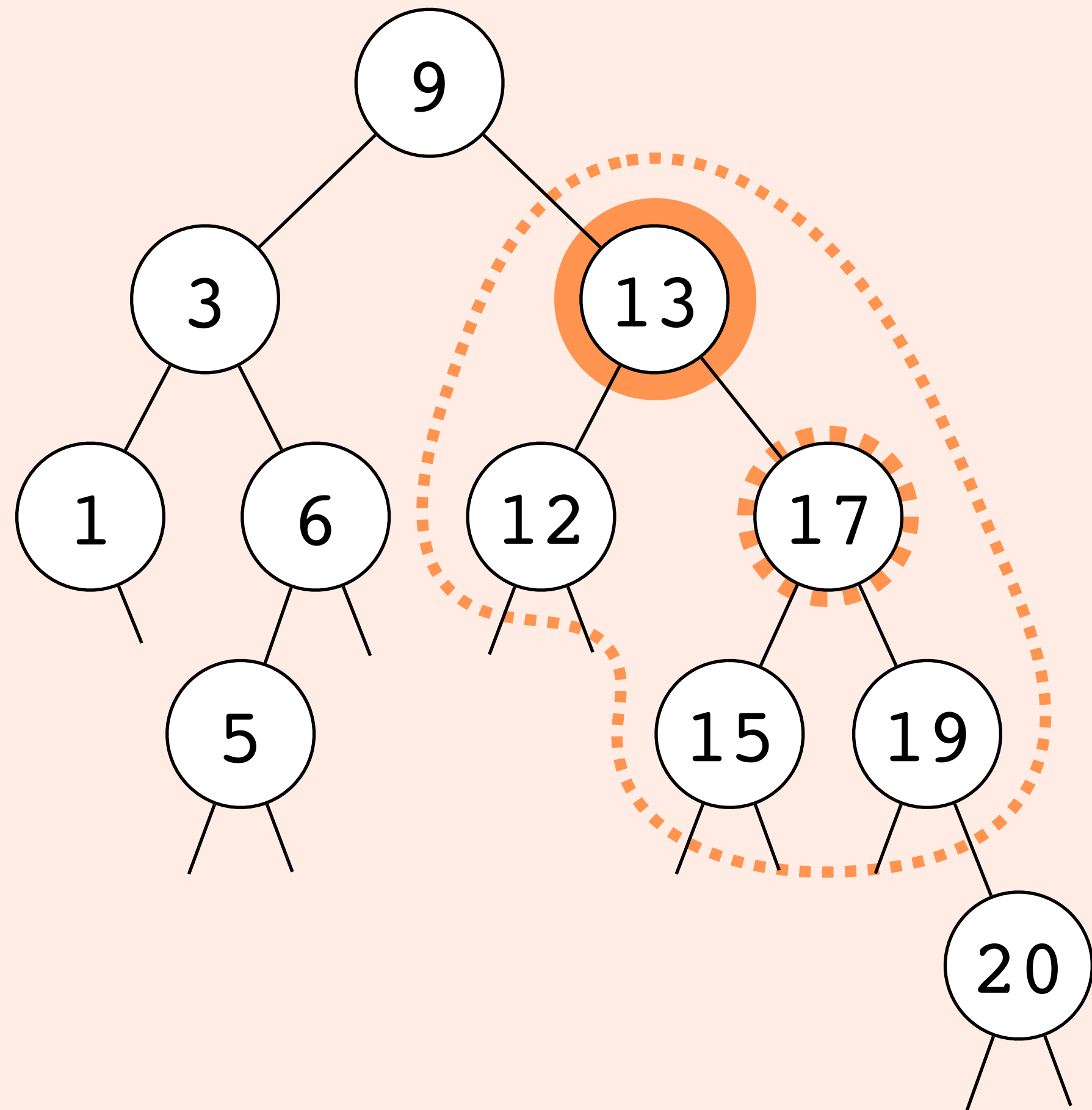
Suppression de 10



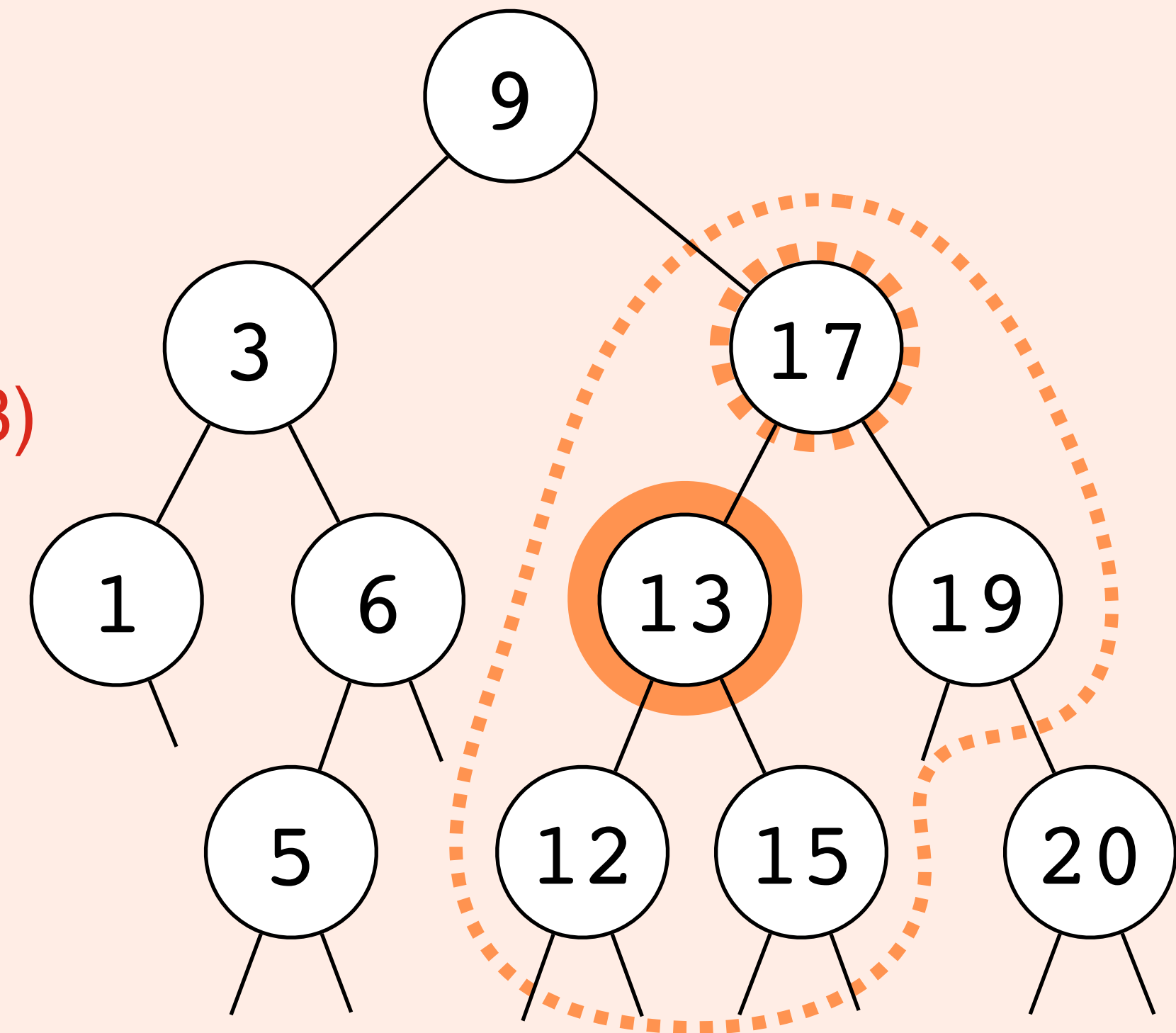
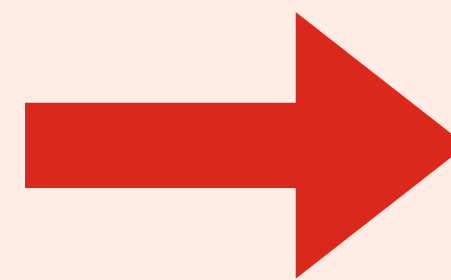


Solution

Suppression de 10



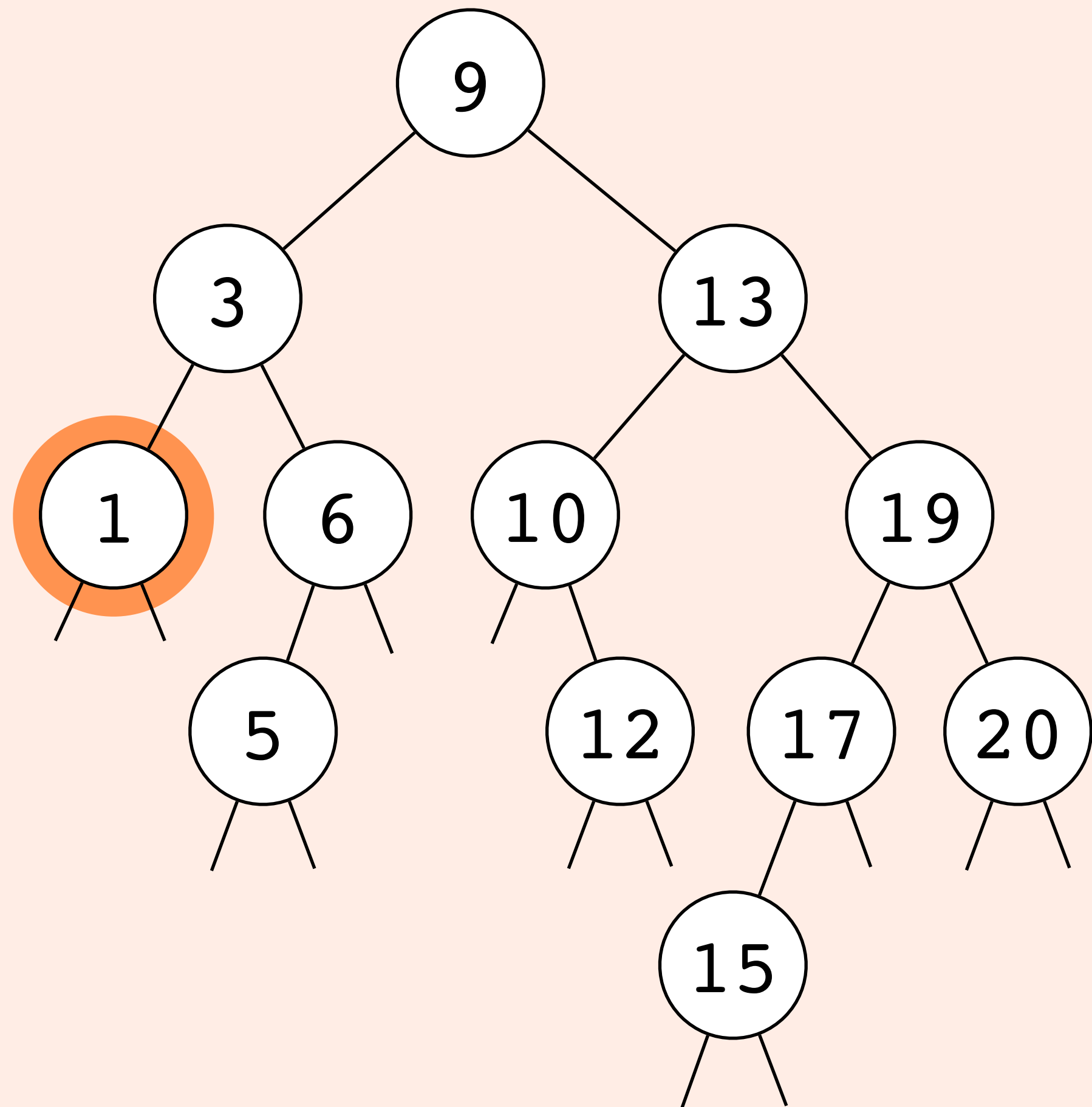
Seconde rotation
rotation_gauche(13)



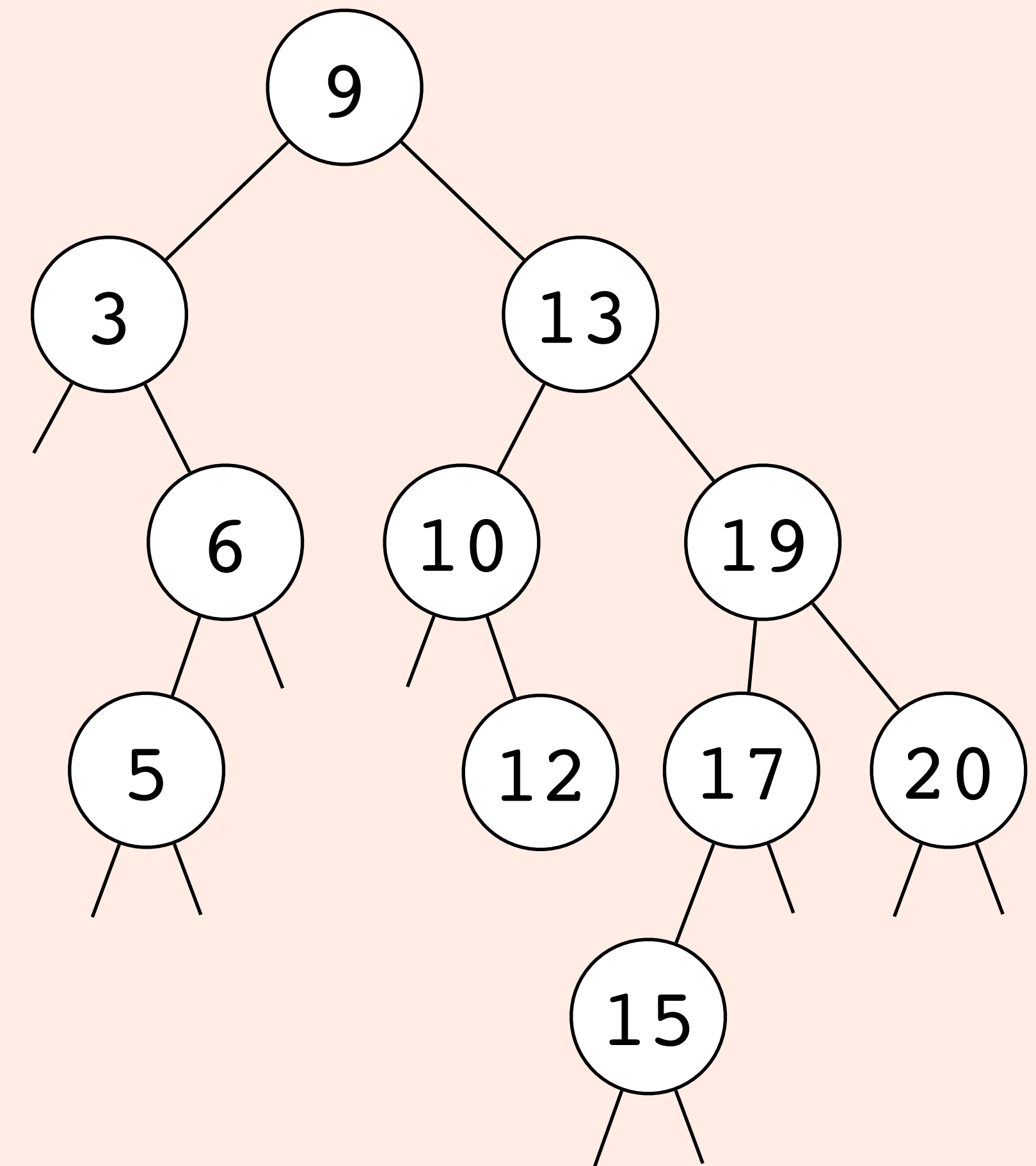
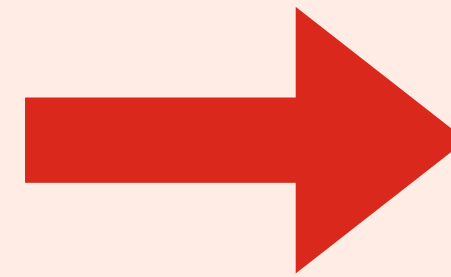


Solution

Suppression de 1

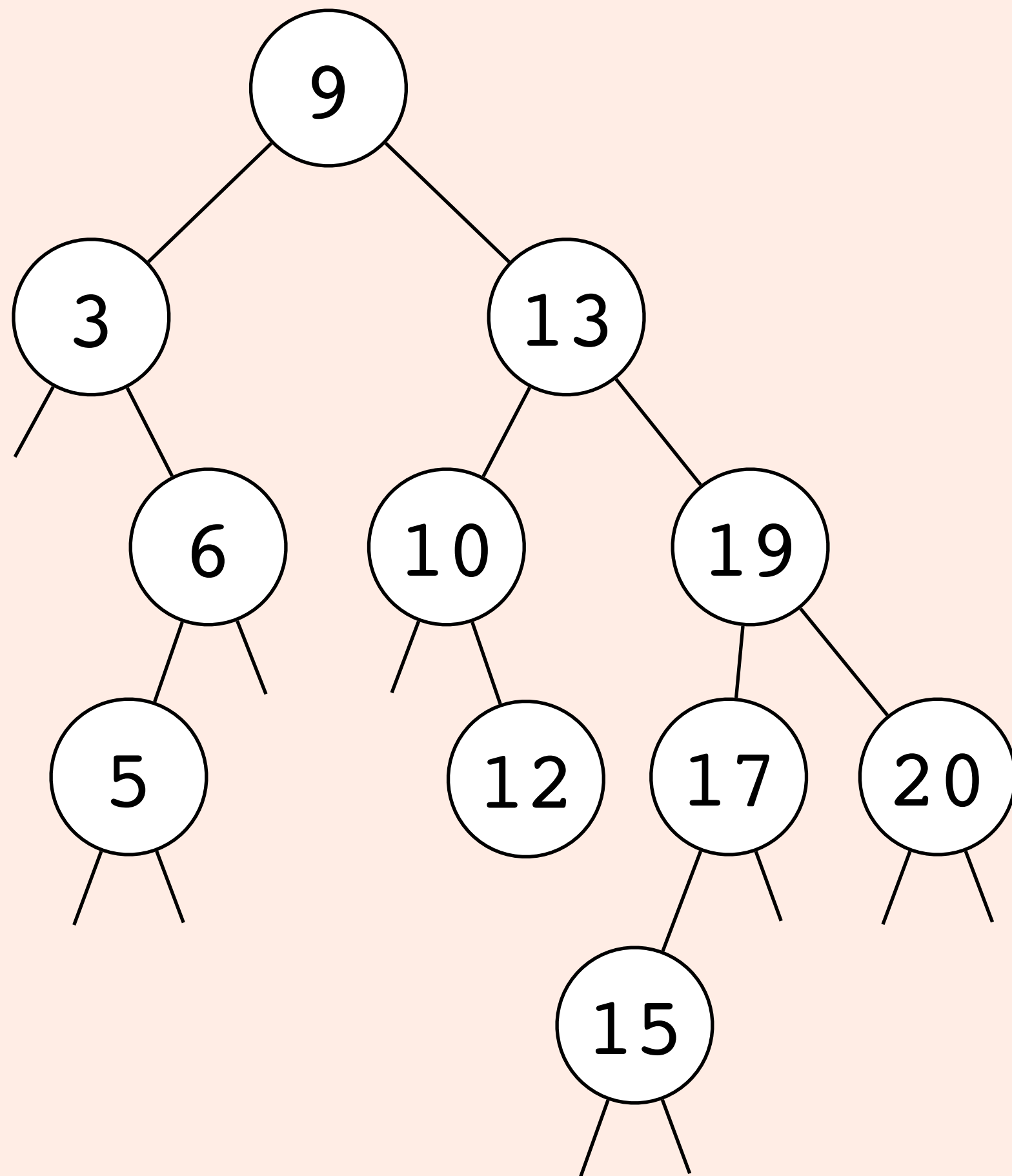


Suppression de l'arbre



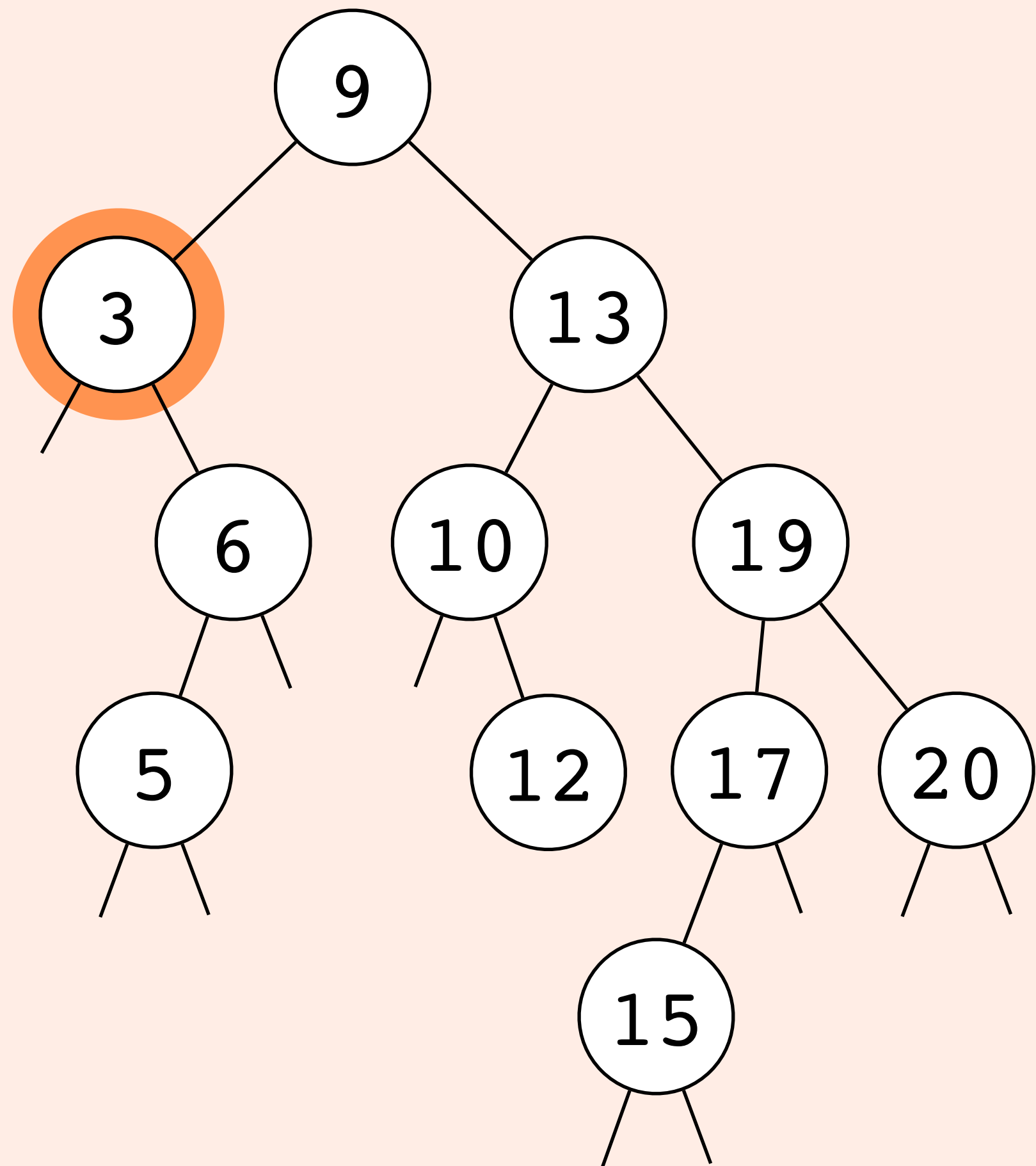
Solution

Suppression de 1



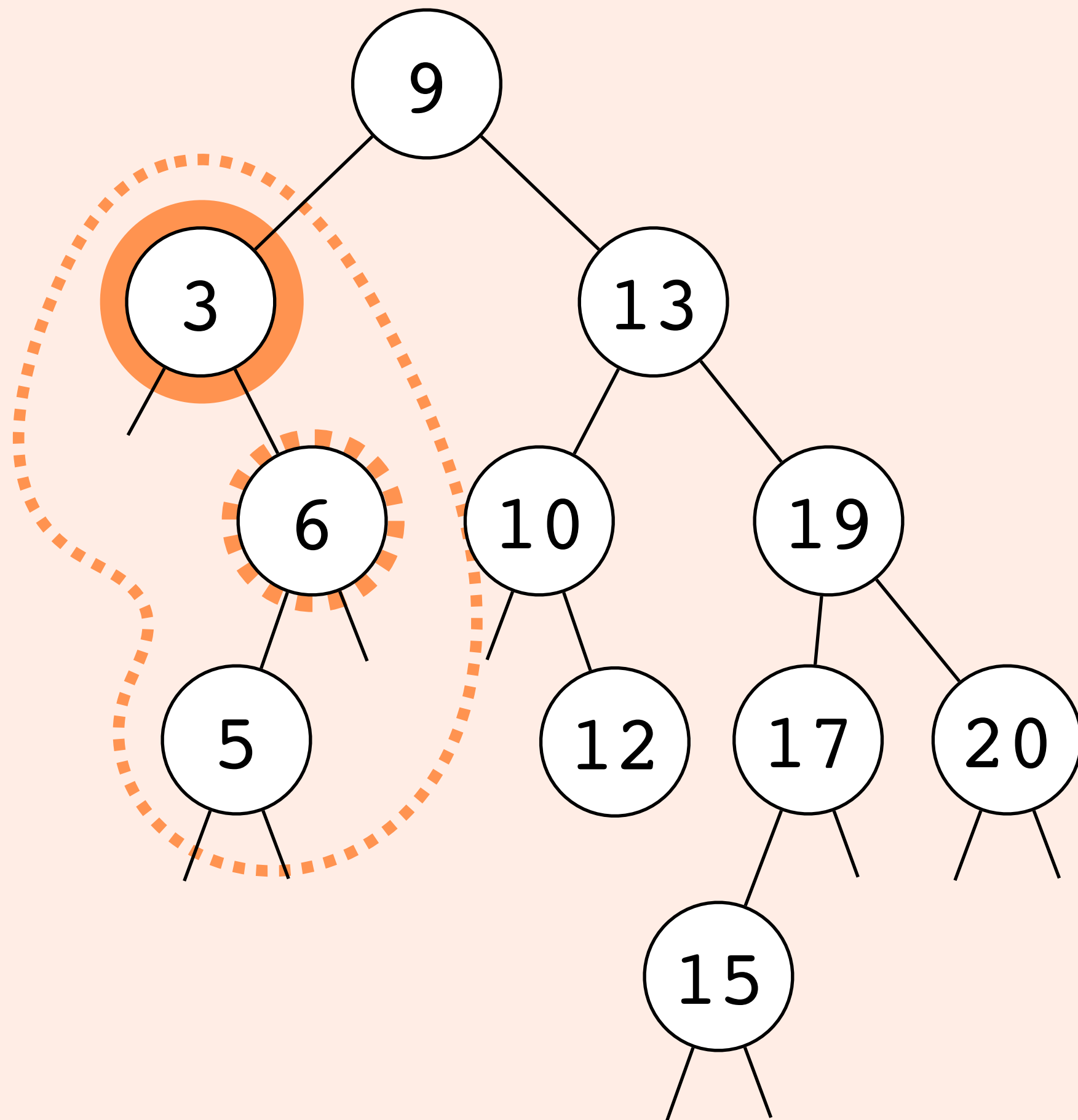
Solution

Suppression de 1



Solution

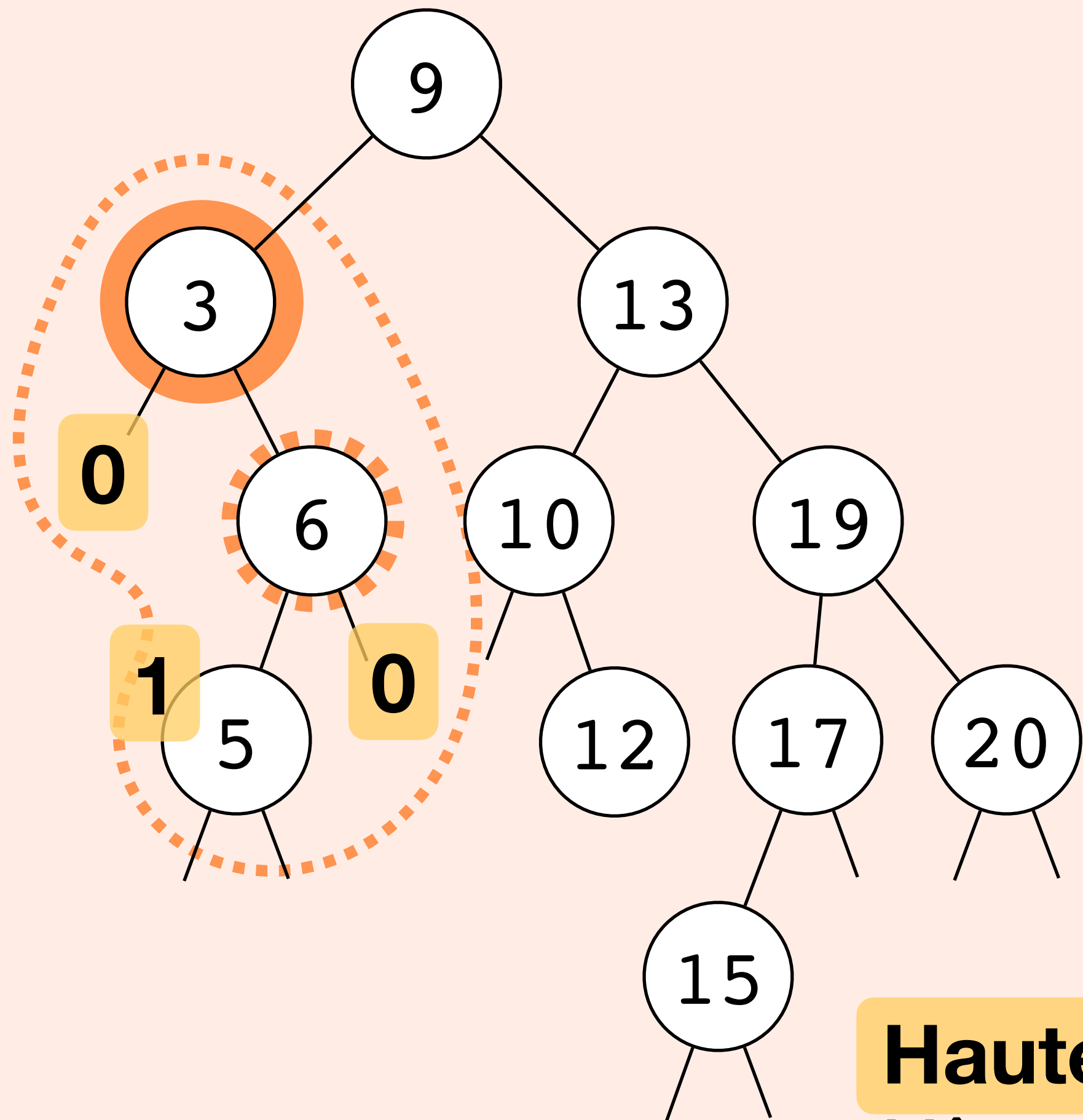
Suppression de 1





Solution

Suppression de 1



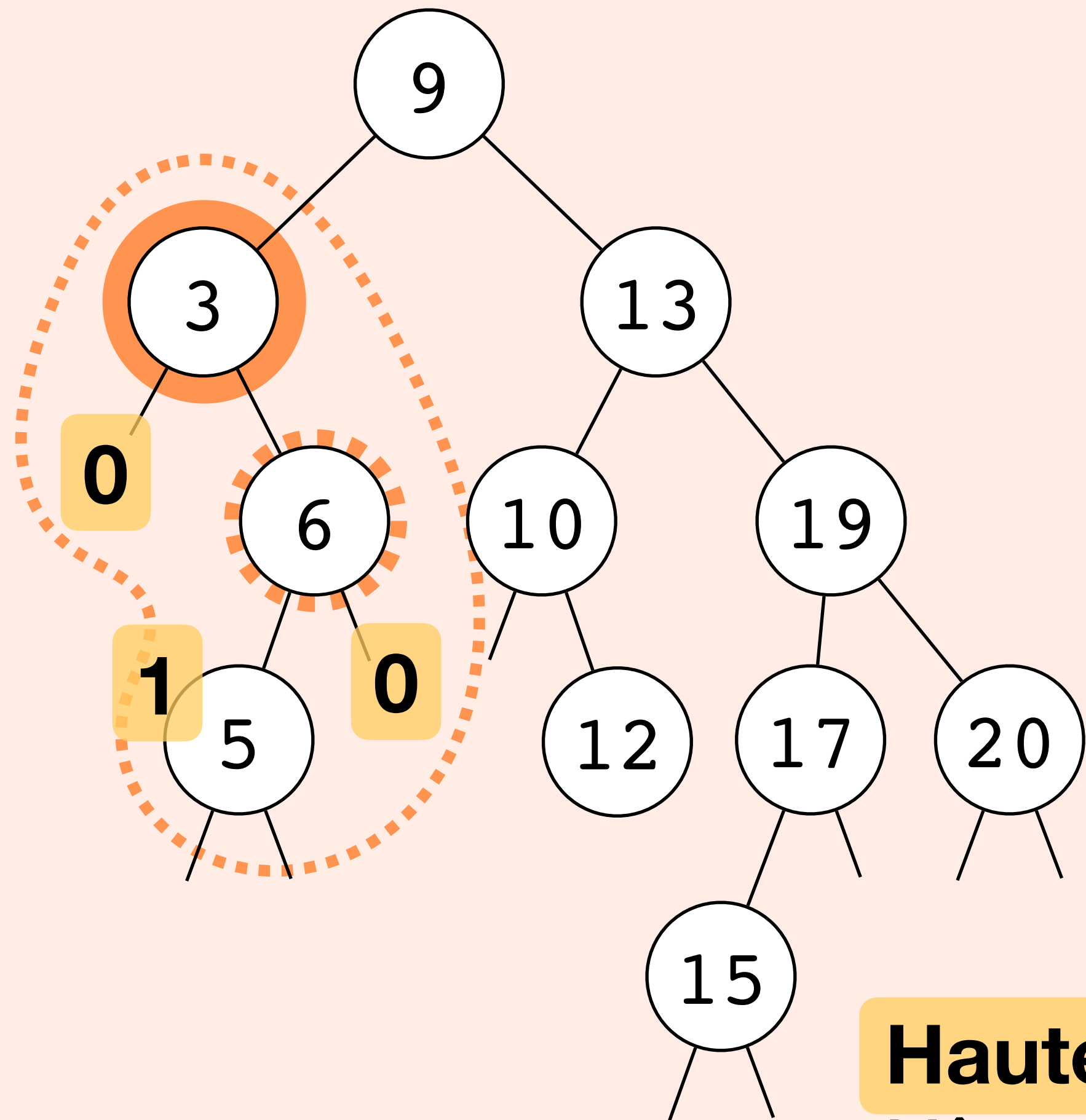
Hauteurs symétriques

Même profondeur à gauche et à droite.
Il faut donc faire une rotation double.

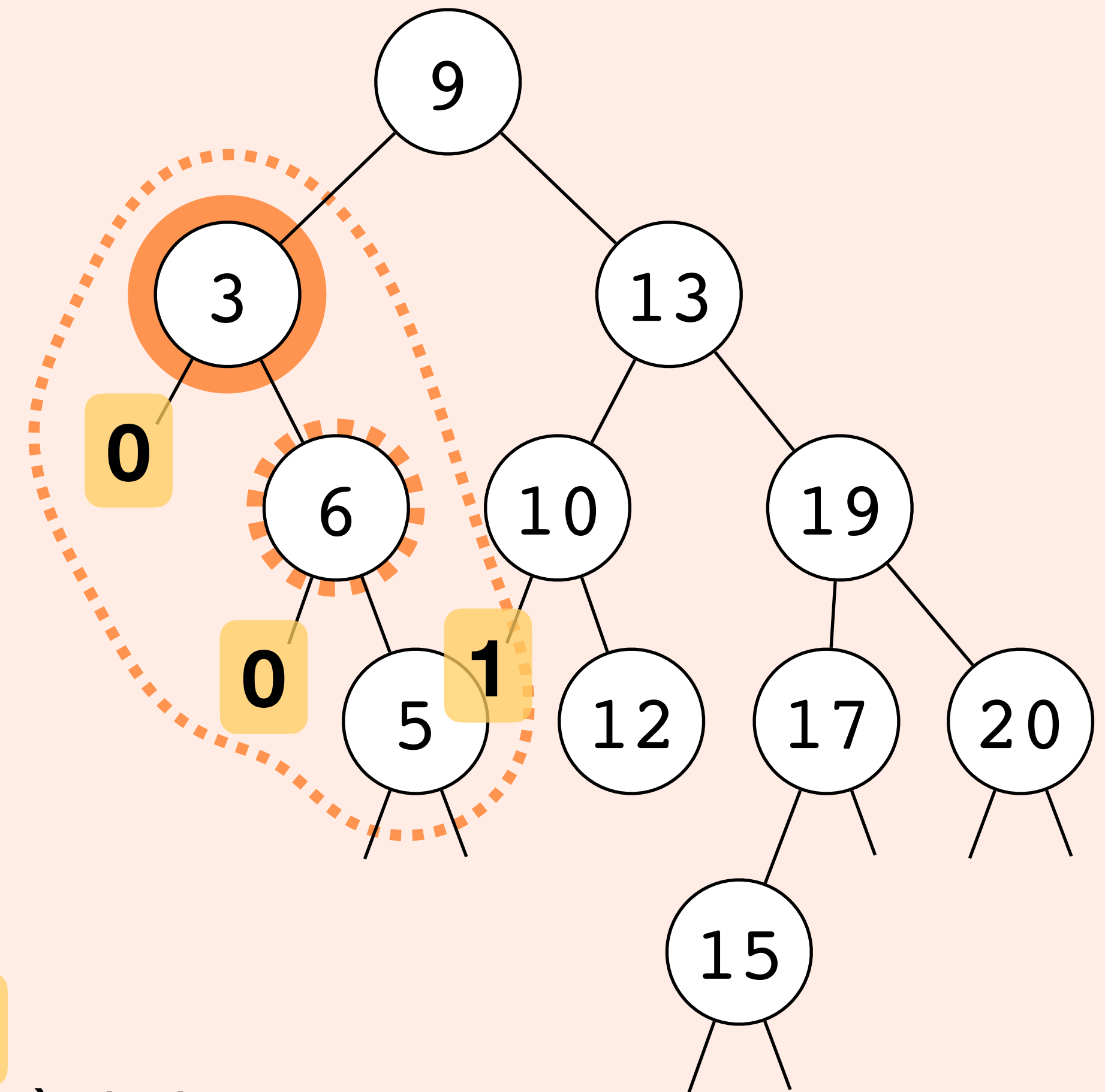


Solution

Suppression de 1



Première rotation
rotation_droite(6)



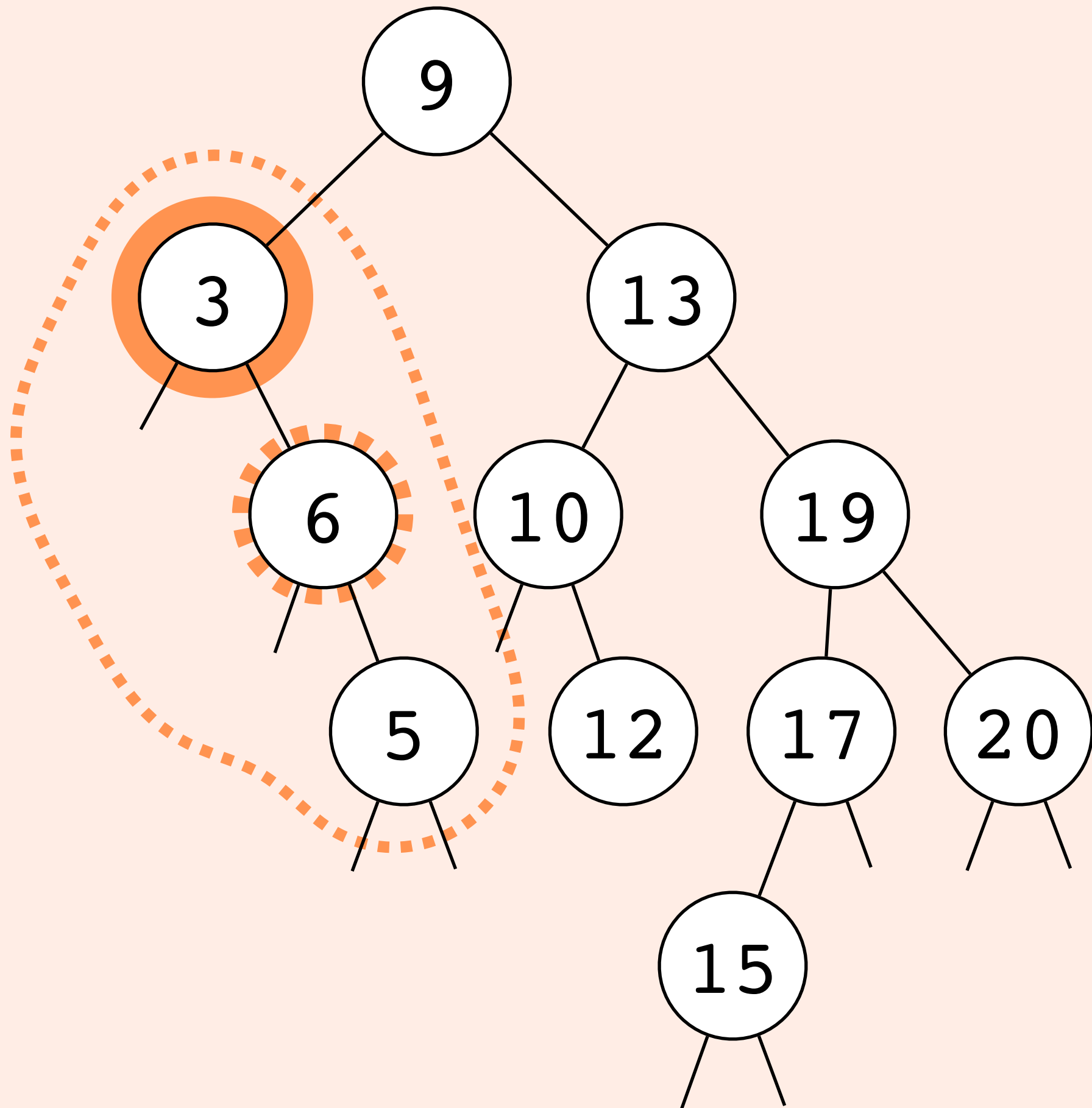
Hauteurs symétriques

Même profondeur à gauche et à droite.
Il faut donc faire une rotation double.



Solution

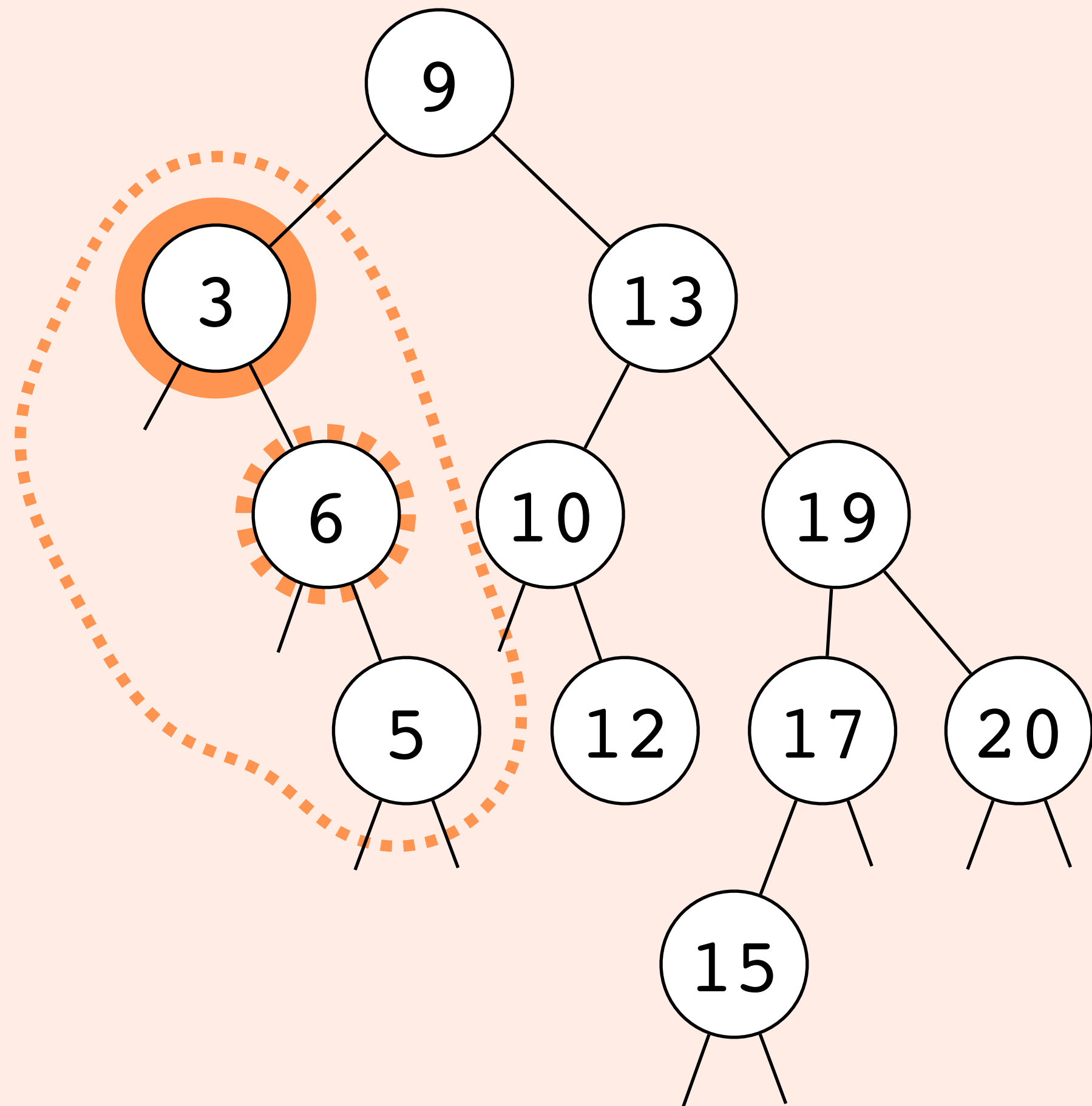
Suppression de 1



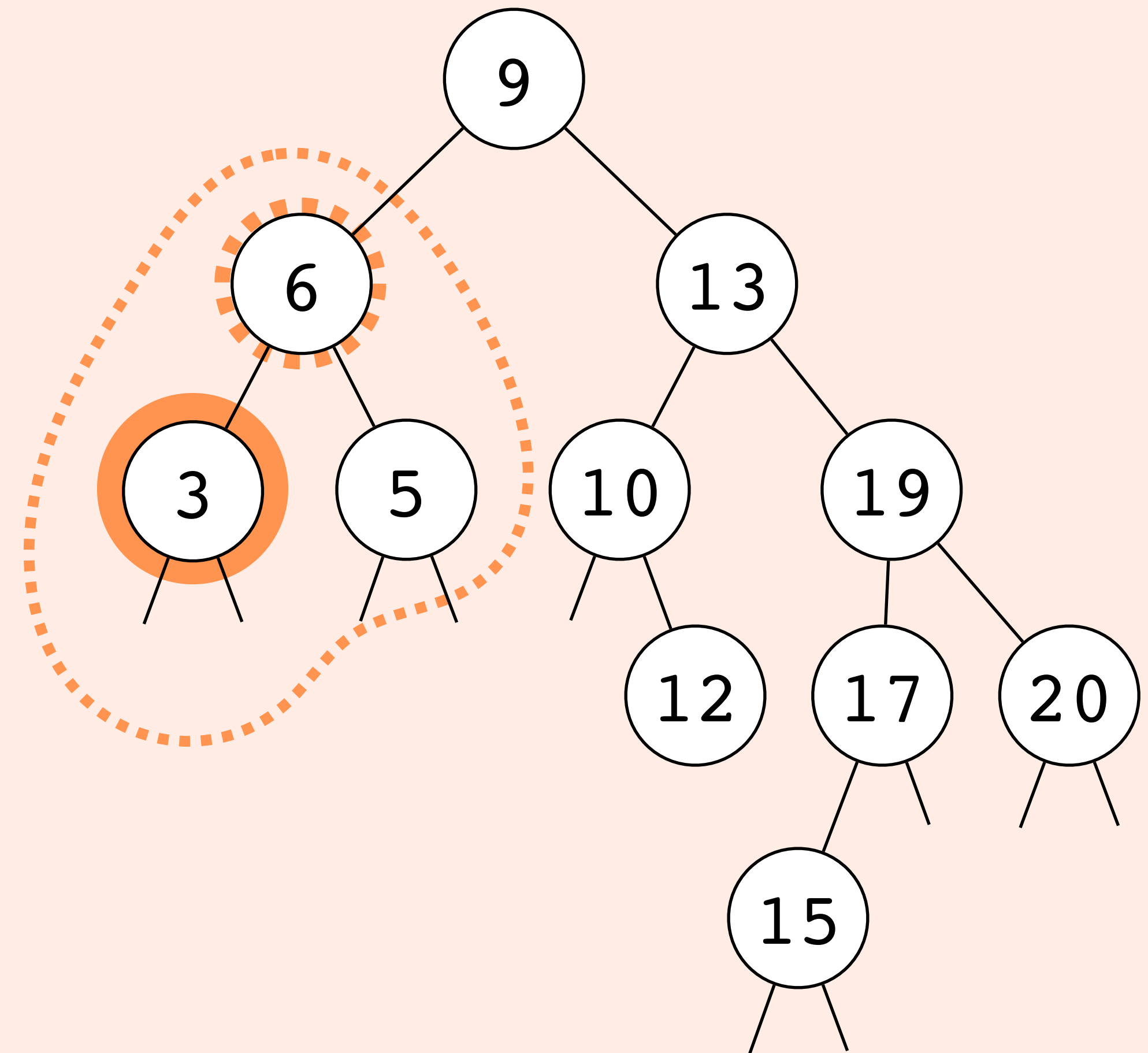
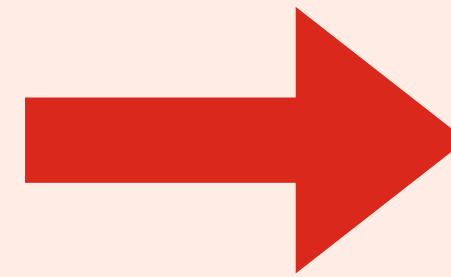


Solution

Suppression de 1



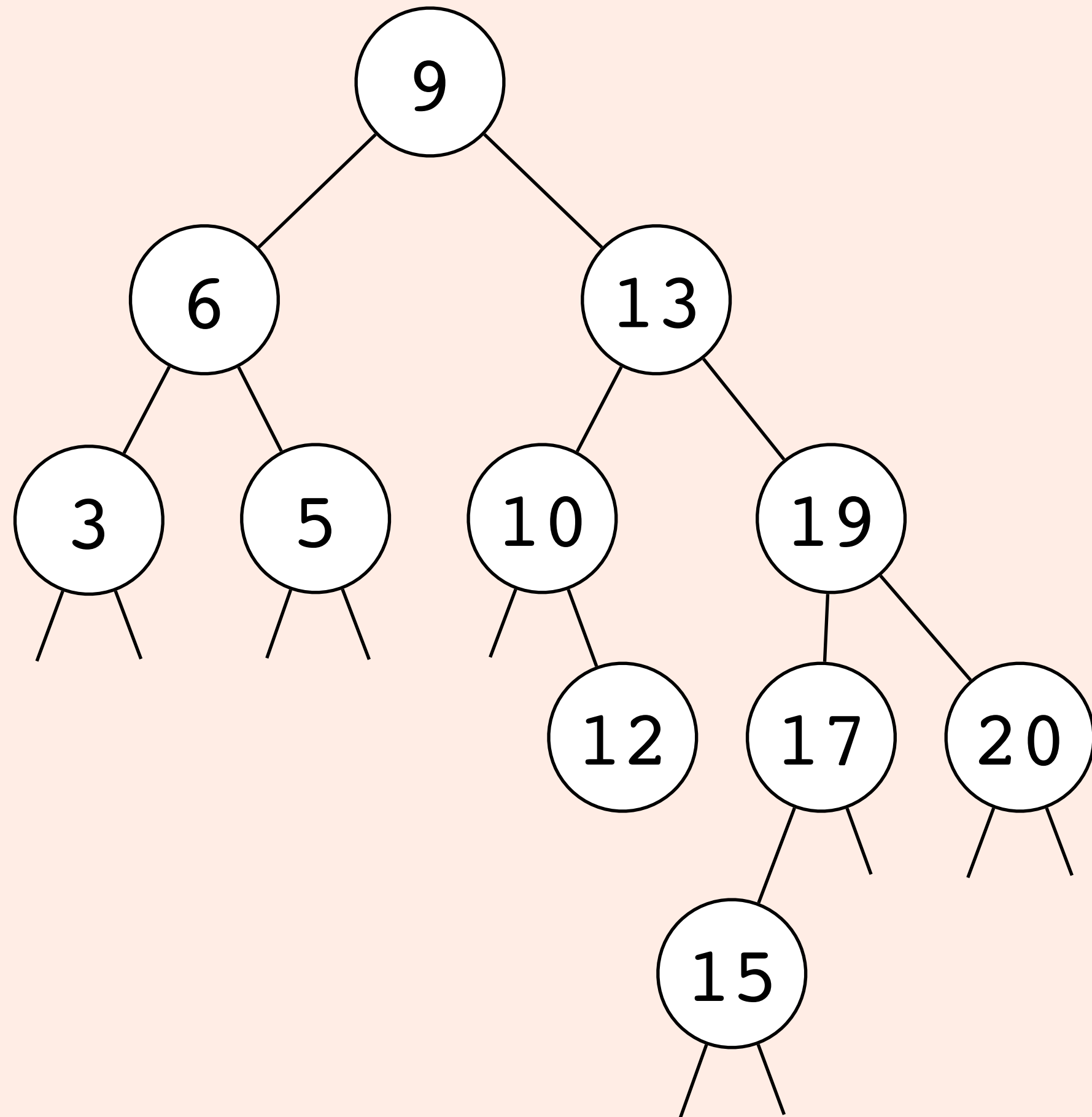
Seconde rotation
rotation_gauche(3)





Solution

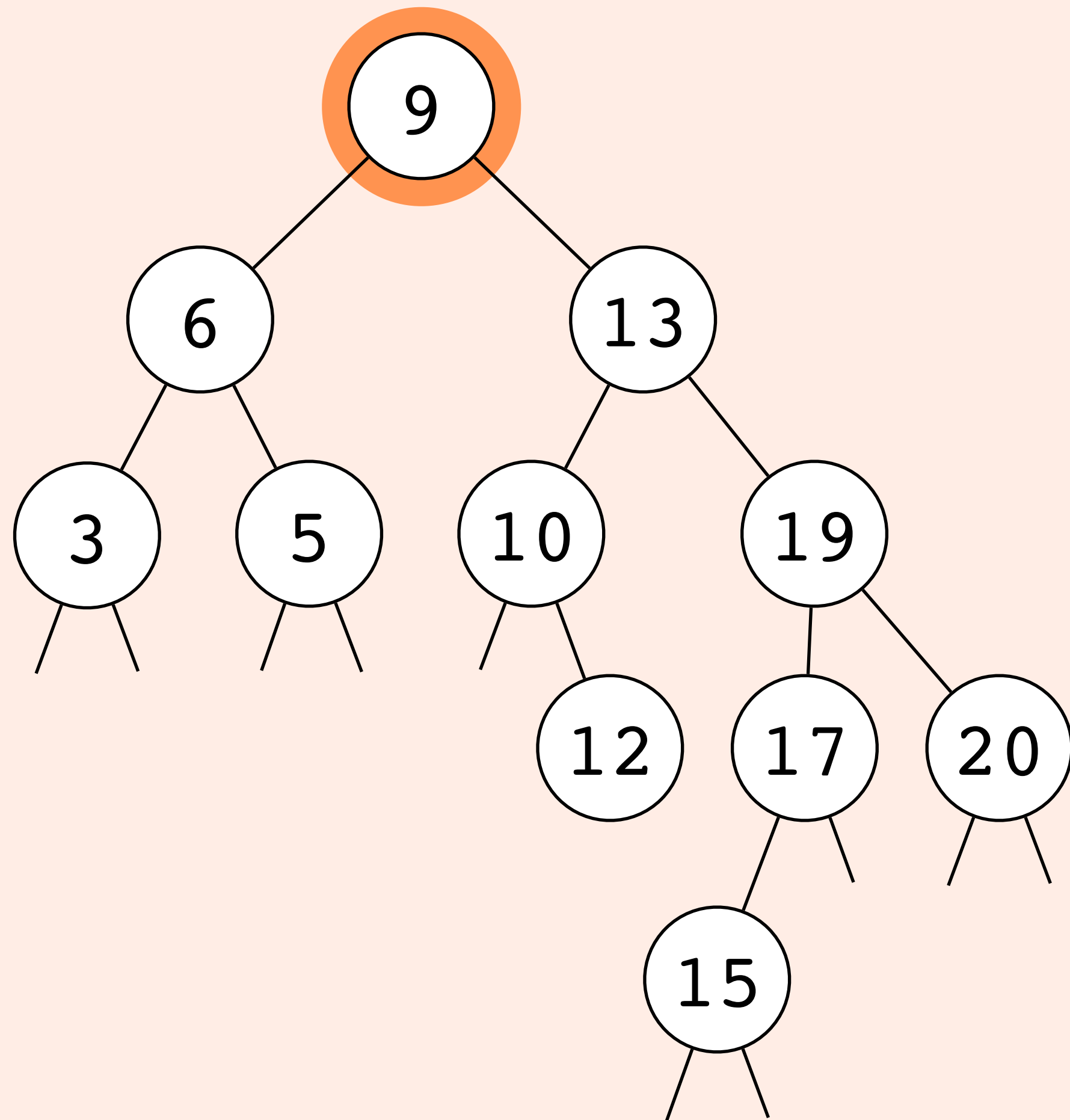
Suppression de 1





Solution

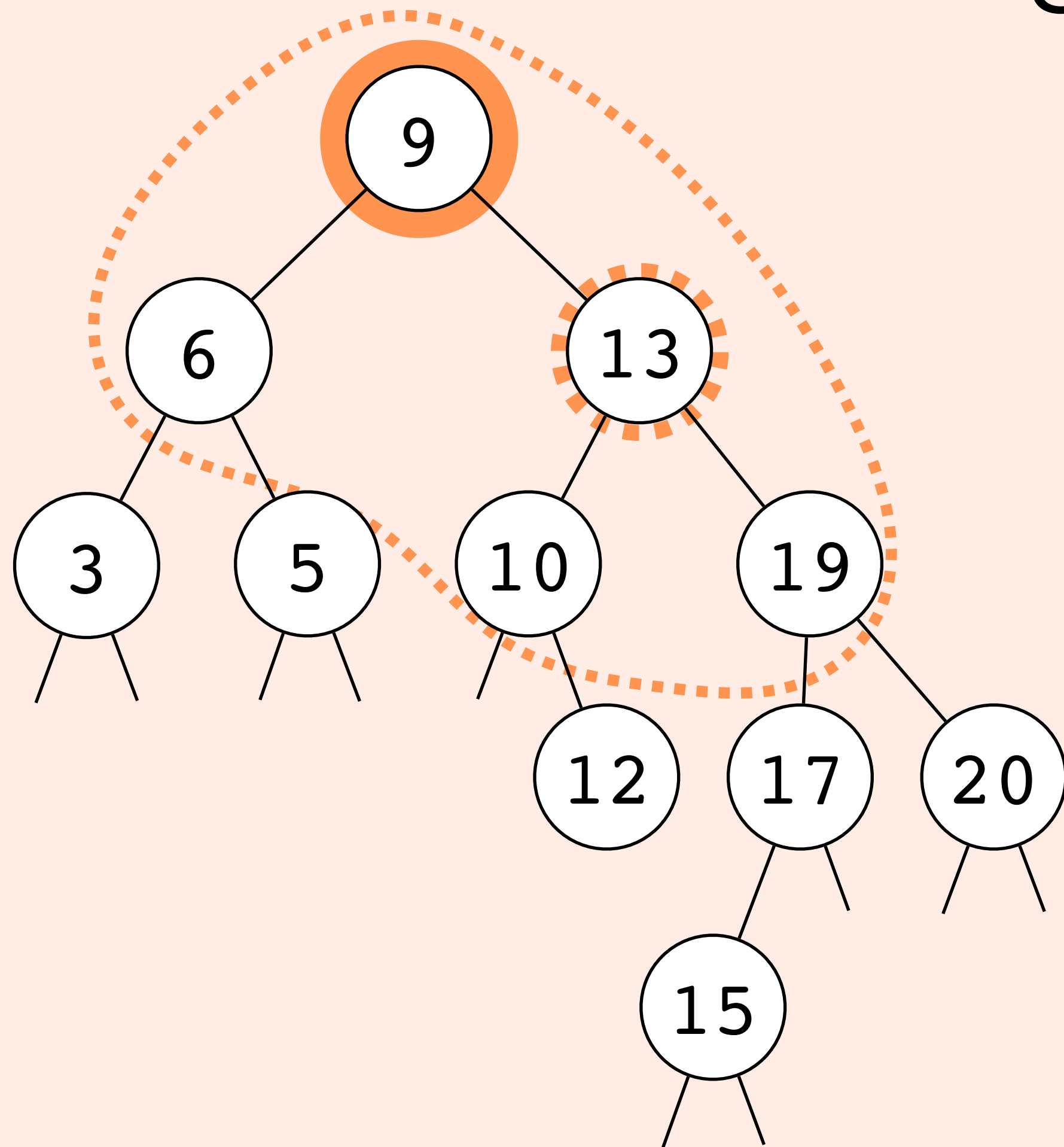
Suppression de 1





Solution

Suppression de 1





Solution

Suppression de 1

Déséquilibre sur 9
rotation_gauche(9)

