

Structures Linéaires



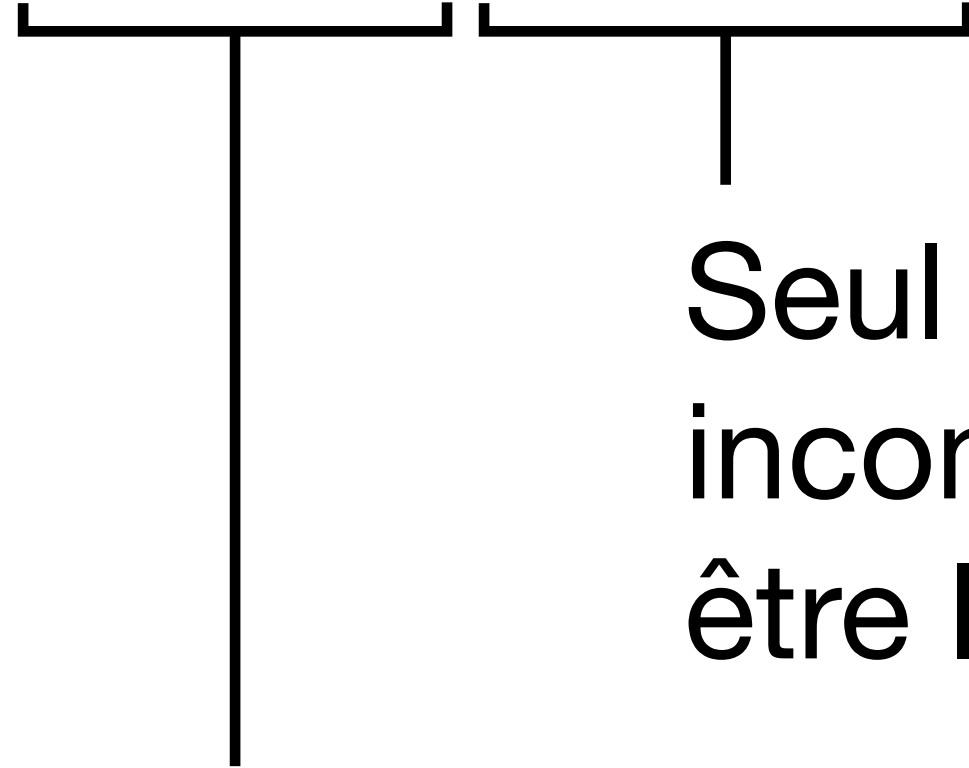


14. Tas : push & pop

Qu'est-ce qu'un tas ?

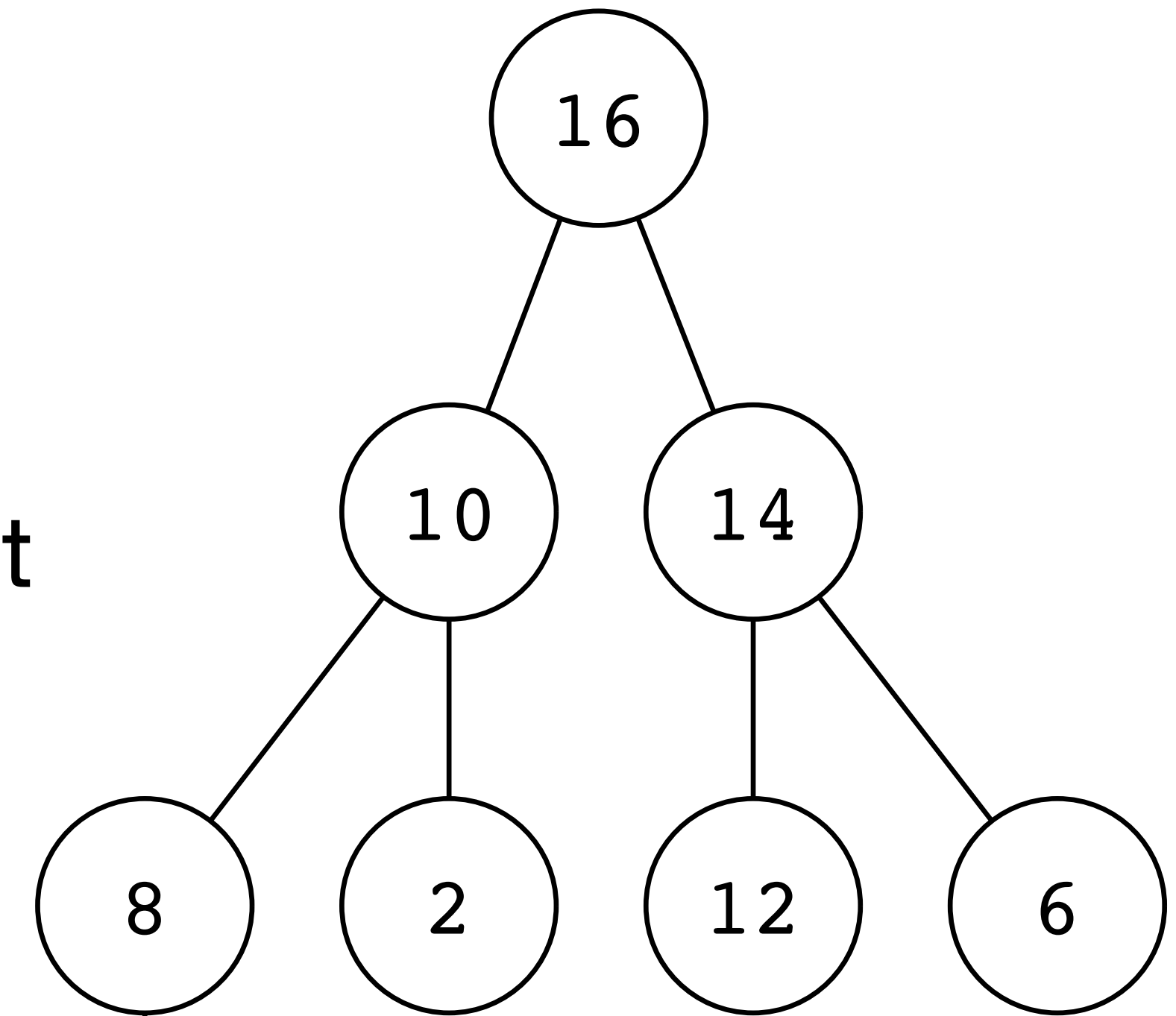


Un arbre binaire complet...



Seul le dernier niveau peut être incomplet, et ses noeuds doivent être le plus à gauche possible.

Chaque noeud a maximum 2 enfants



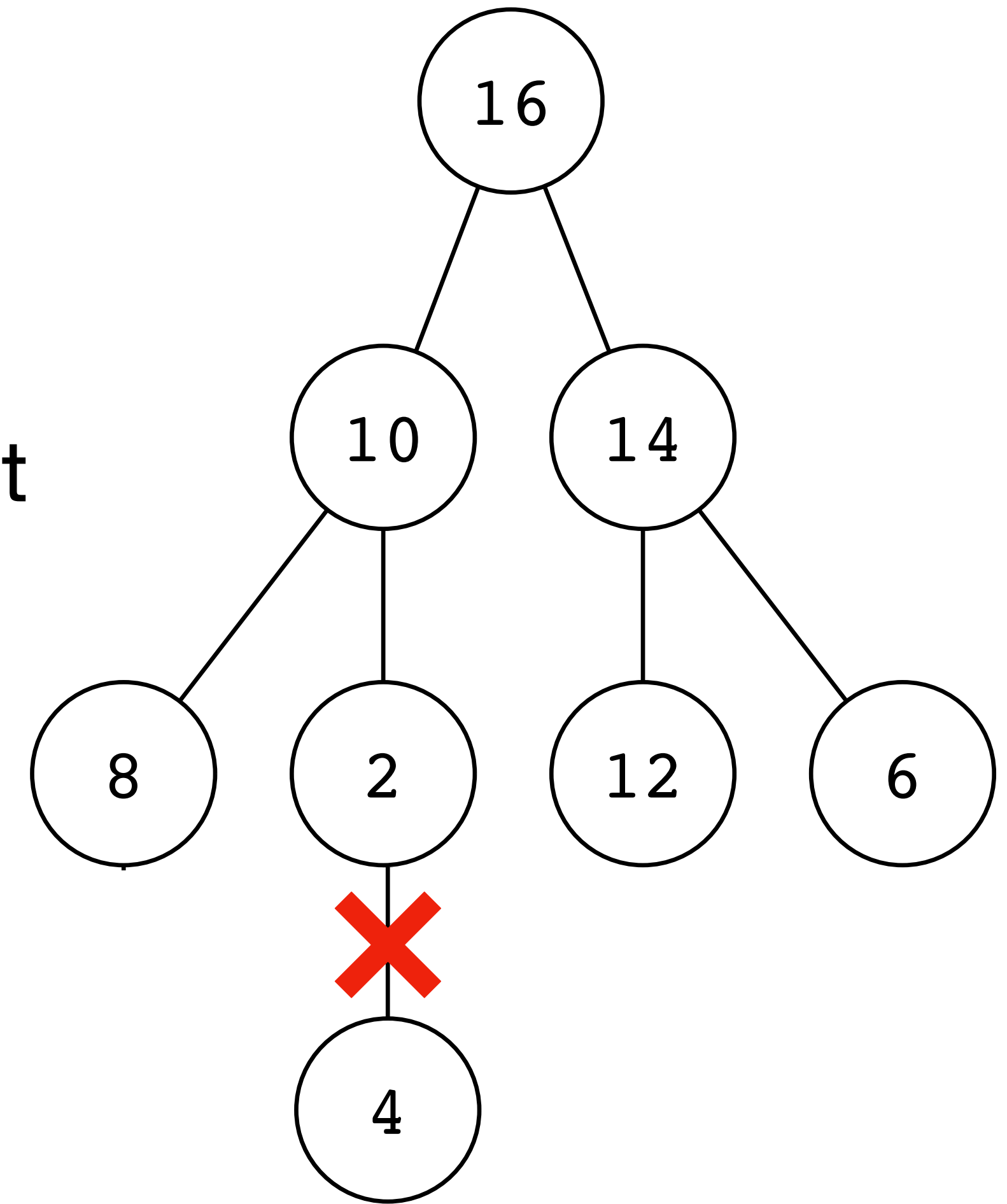
Qu'est-ce qu'un tas ?



Un arbre binaire complet...

Seul le dernier niveau peut être incomplet, et ses noeuds doivent être le plus à gauche possible.

Chaque noeud a maximum 2 enfants



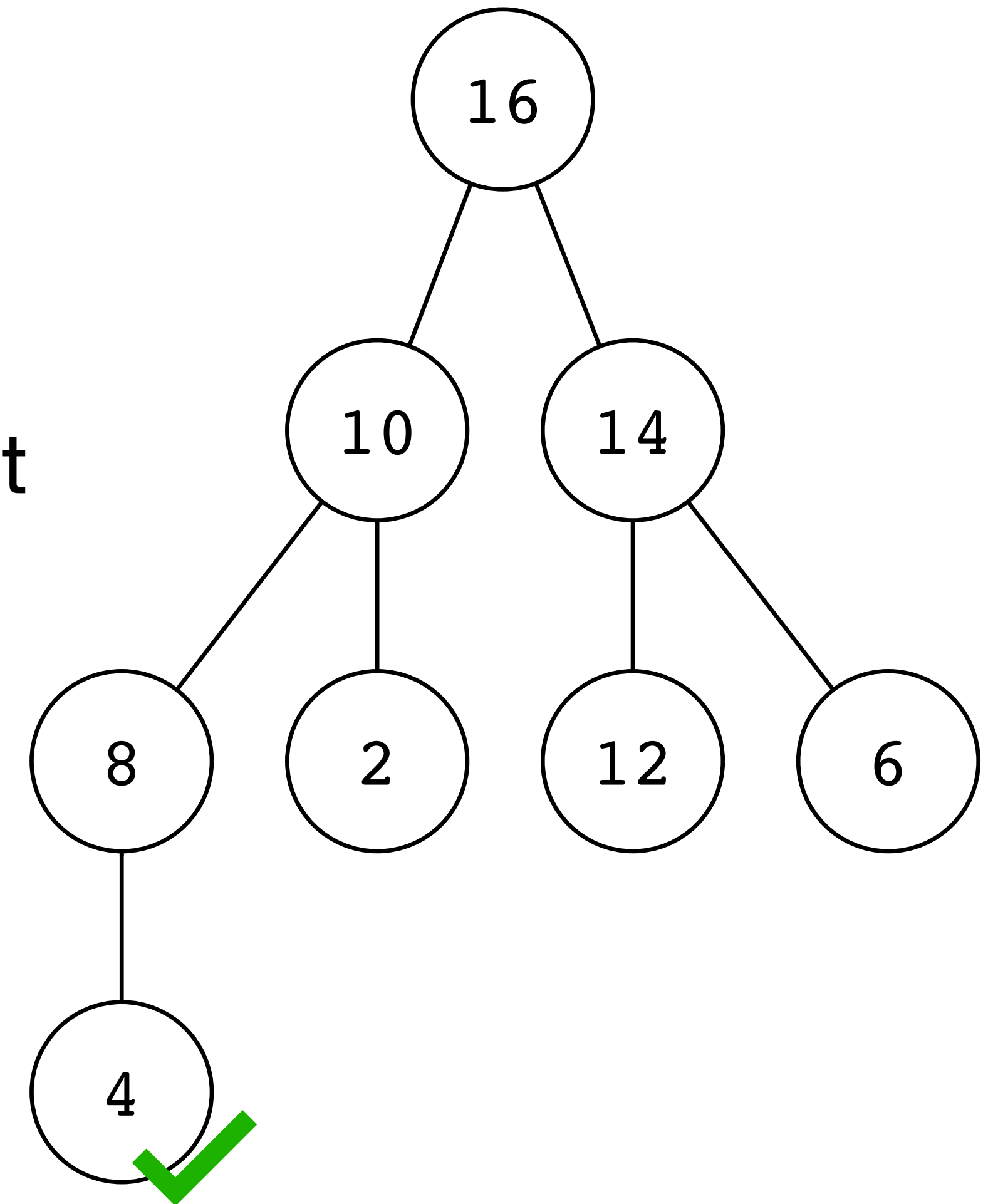
Qu'est-ce qu'un tas ?



Un arbre binaire complet...

Seul le dernier niveau peut être incomplet, et ses noeuds doivent être le plus à gauche possible.

Chaque noeud a maximum 2 enfants



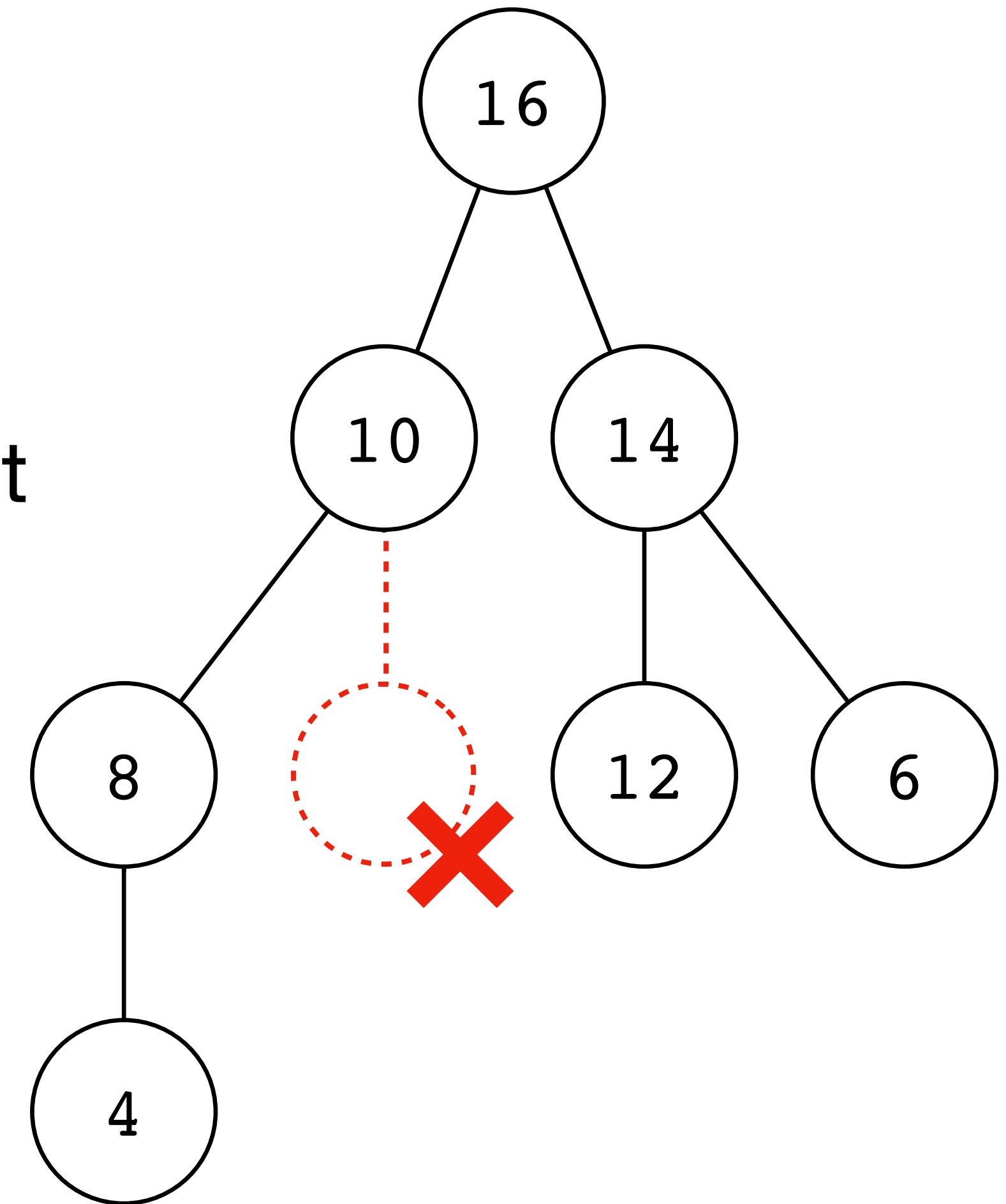
Qu'est-ce qu'un tas ?



Un arbre binaire complet...

Seul le dernier niveau peut être incomplet, et ses noeuds doivent être le plus à gauche possible.

Chaque noeud a maximum 2 enfants



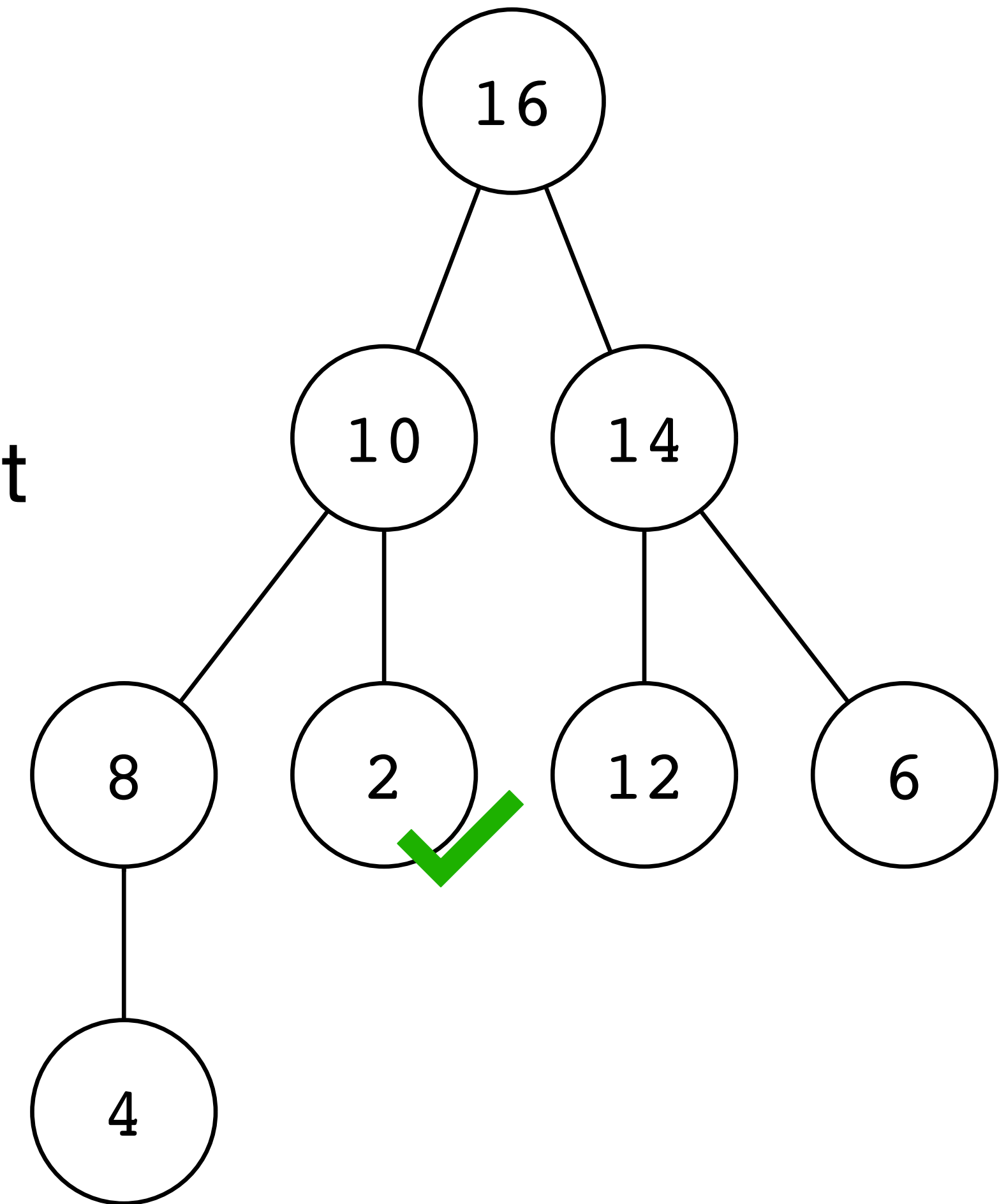
Qu'est-ce qu'un tas ?



Un arbre binaire complet...

Seul le dernier niveau peut être incomplet, et ses noeuds doivent être le plus à gauche possible.

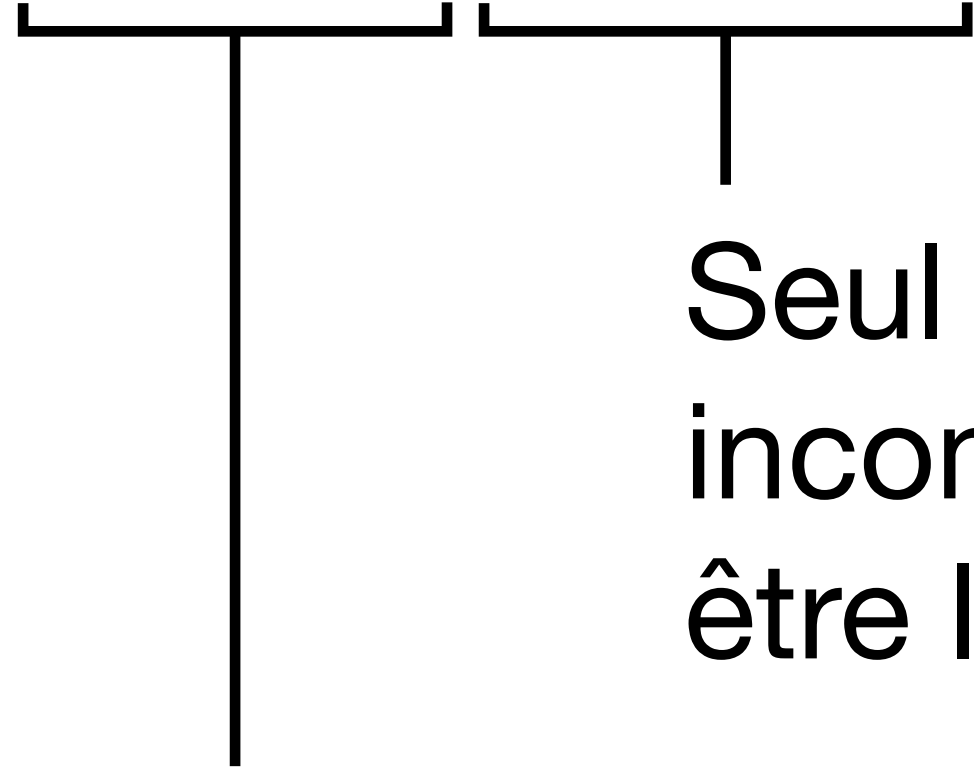
Chaque noeud a maximum 2 enfants



Qu'est-ce qu'un tas ?



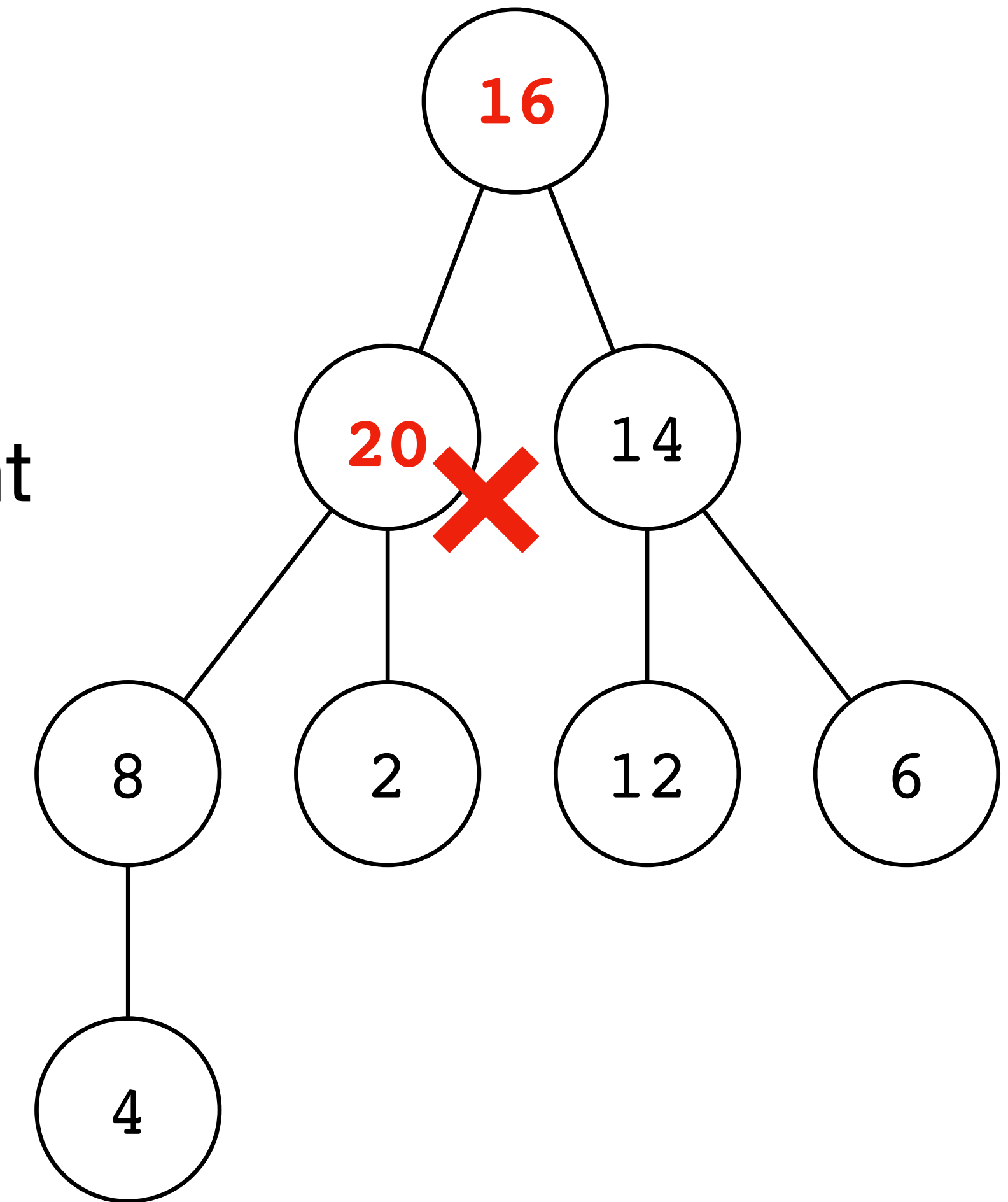
Un arbre binaire complet...



Seul le dernier niveau peut être incomplet, et ses noeuds doivent être le plus à gauche possible.

Chaque noeud a maximum 2 enfants

...tel que tout noeud contient une valeur plus grande que celle de ses enfants.



Qu'est-ce qu'un tas ?

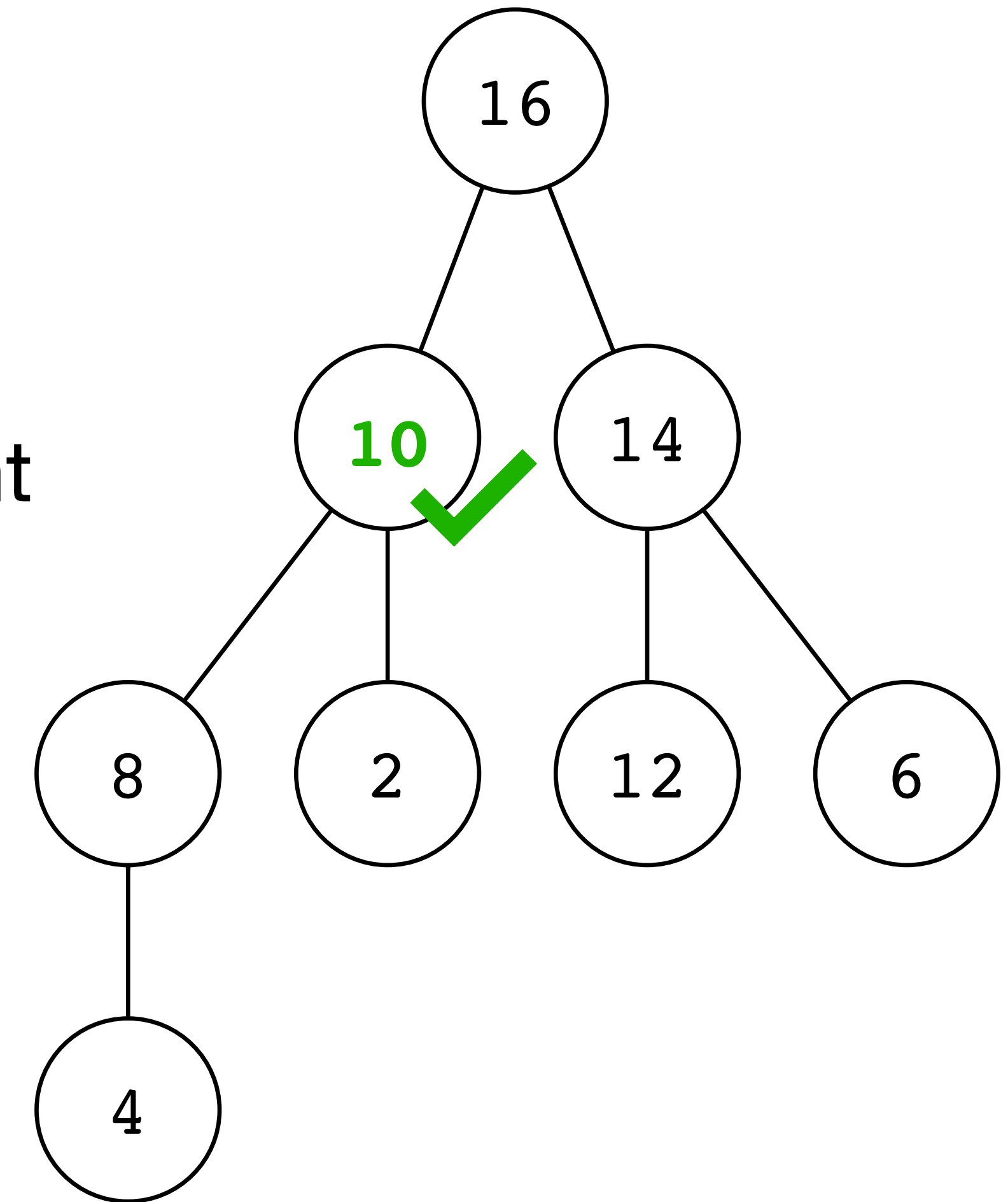


Un arbre binaire complet...

Seul le dernier niveau peut être incomplet, et ses noeuds doivent être le plus à gauche possible.

Chaque noeud a maximum 2 enfants

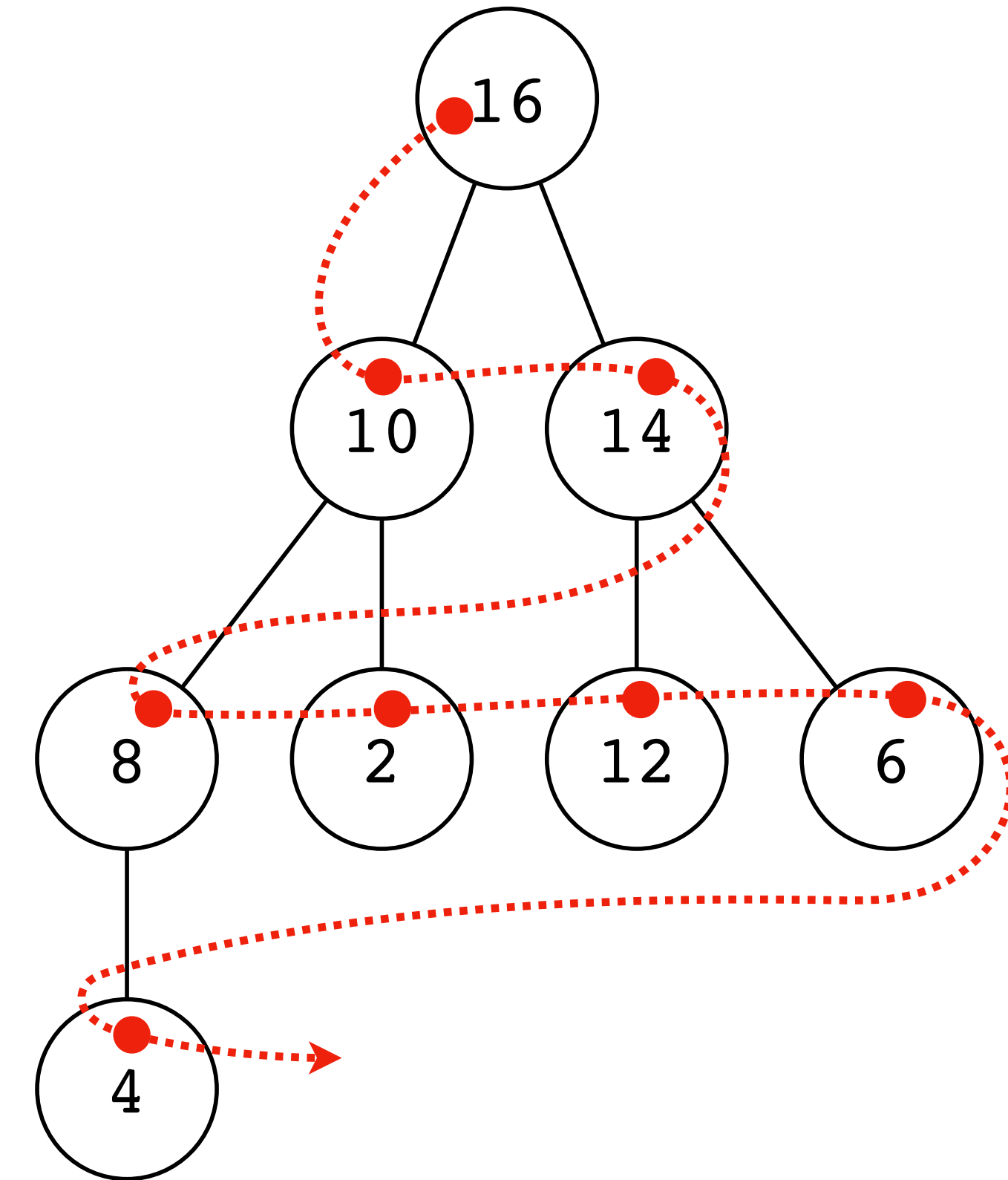
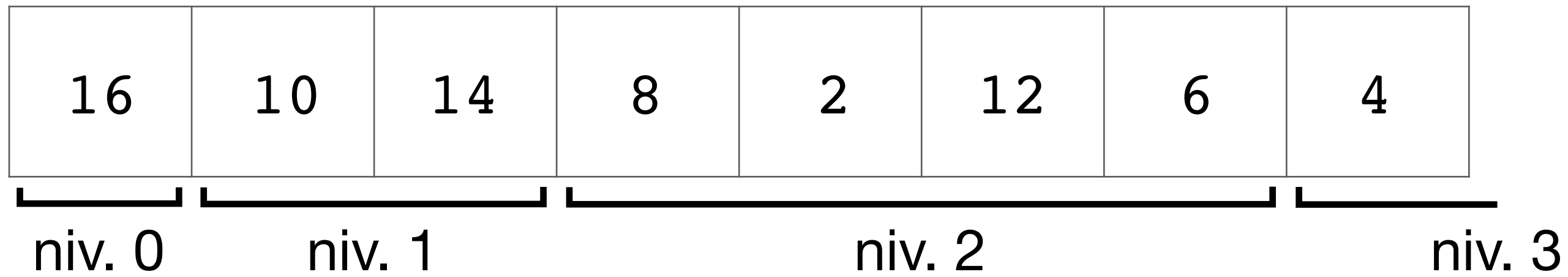
...tel que tout noeud contient une valeur plus grande que celle de ses enfants.



Représentation d'un tas



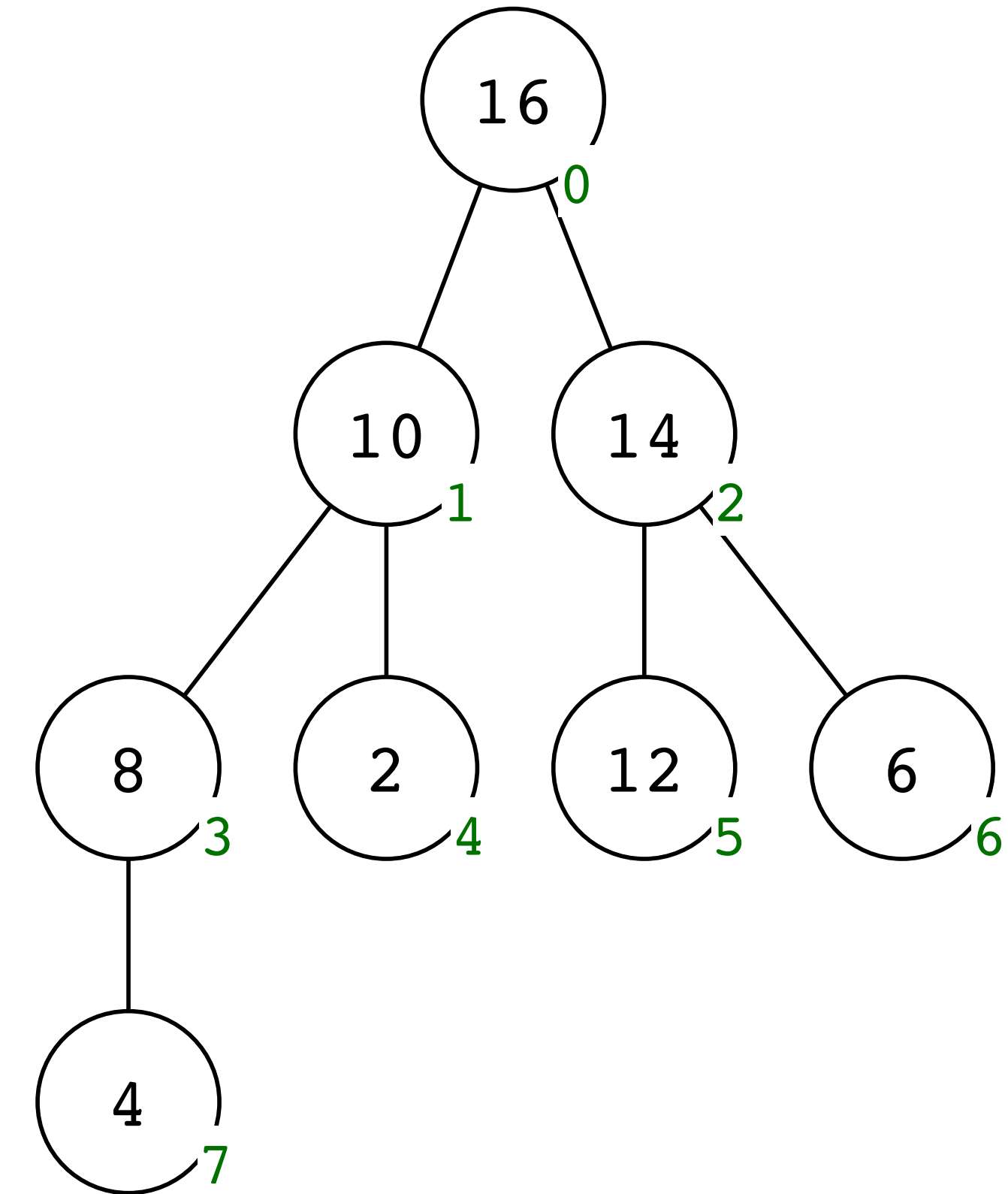
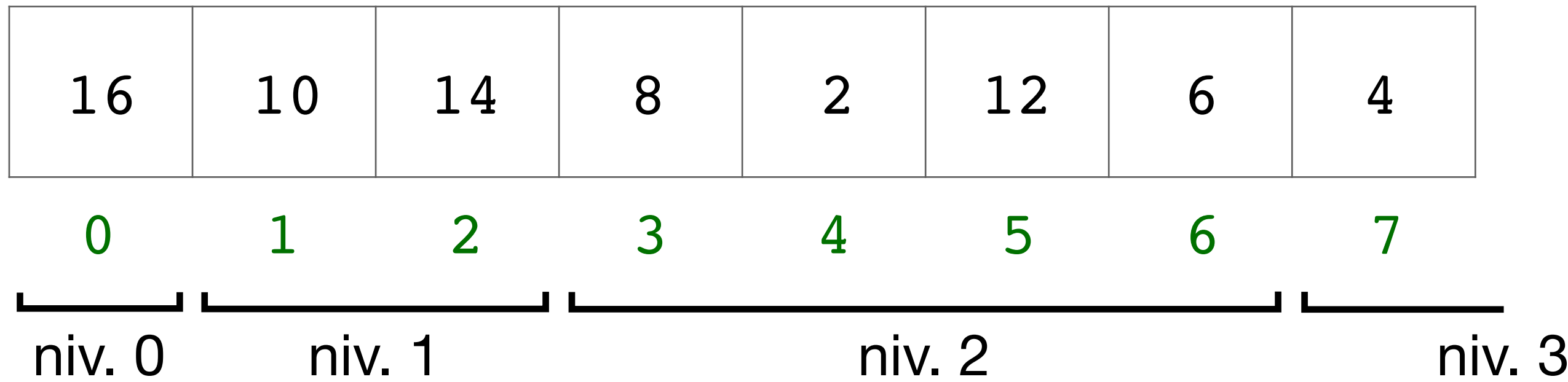
Dans un tableau



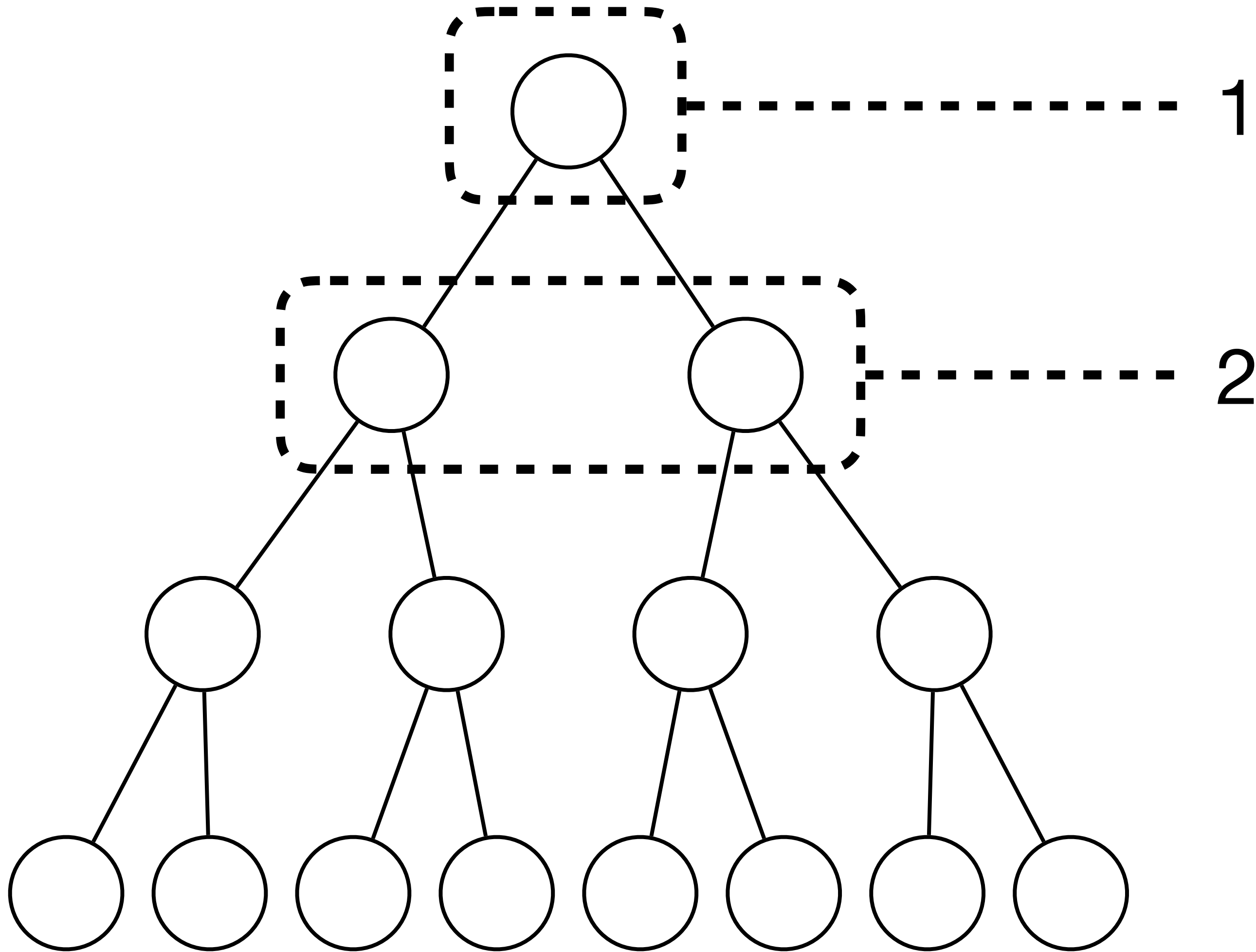
Représentation d'un tas



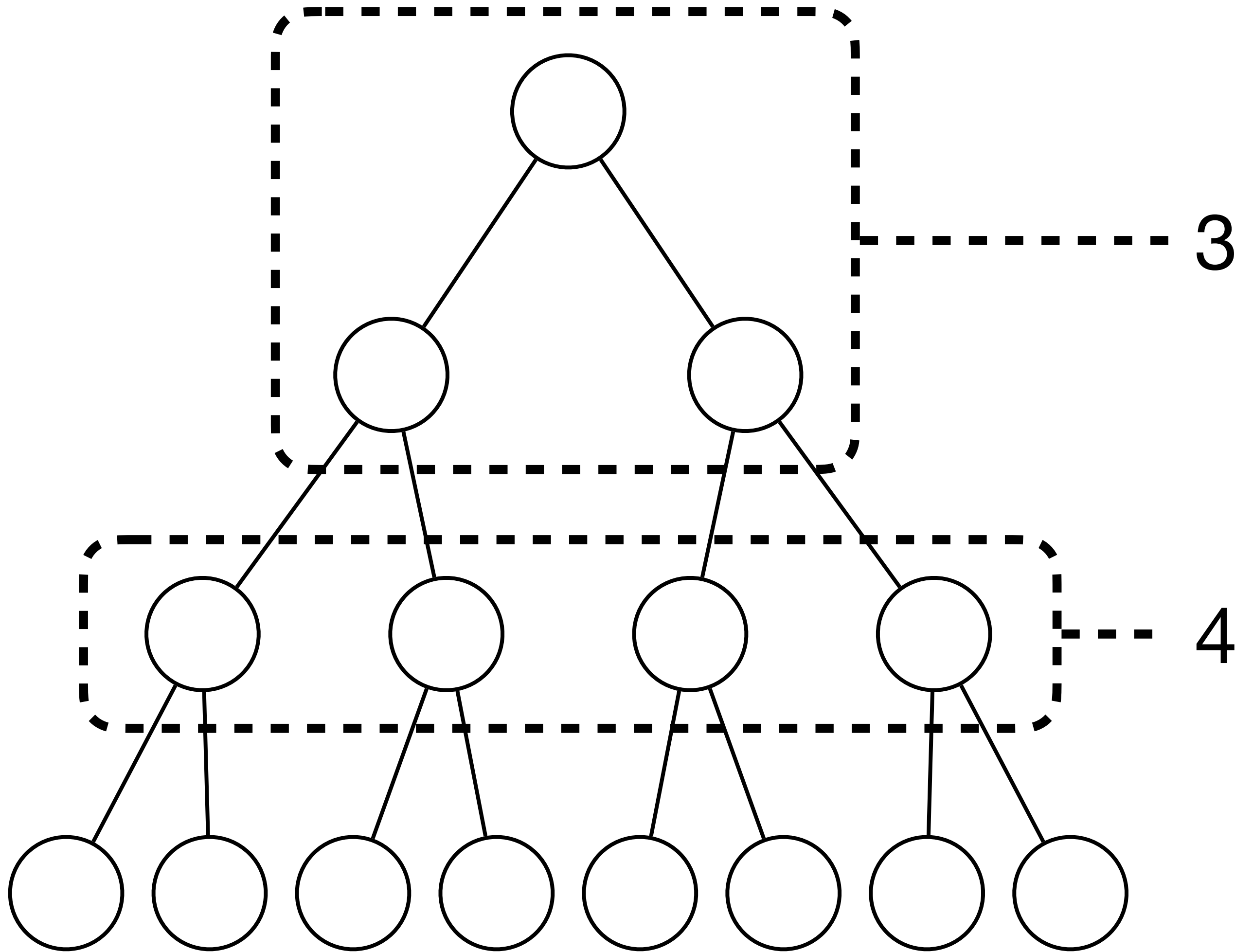
Dans un tableau



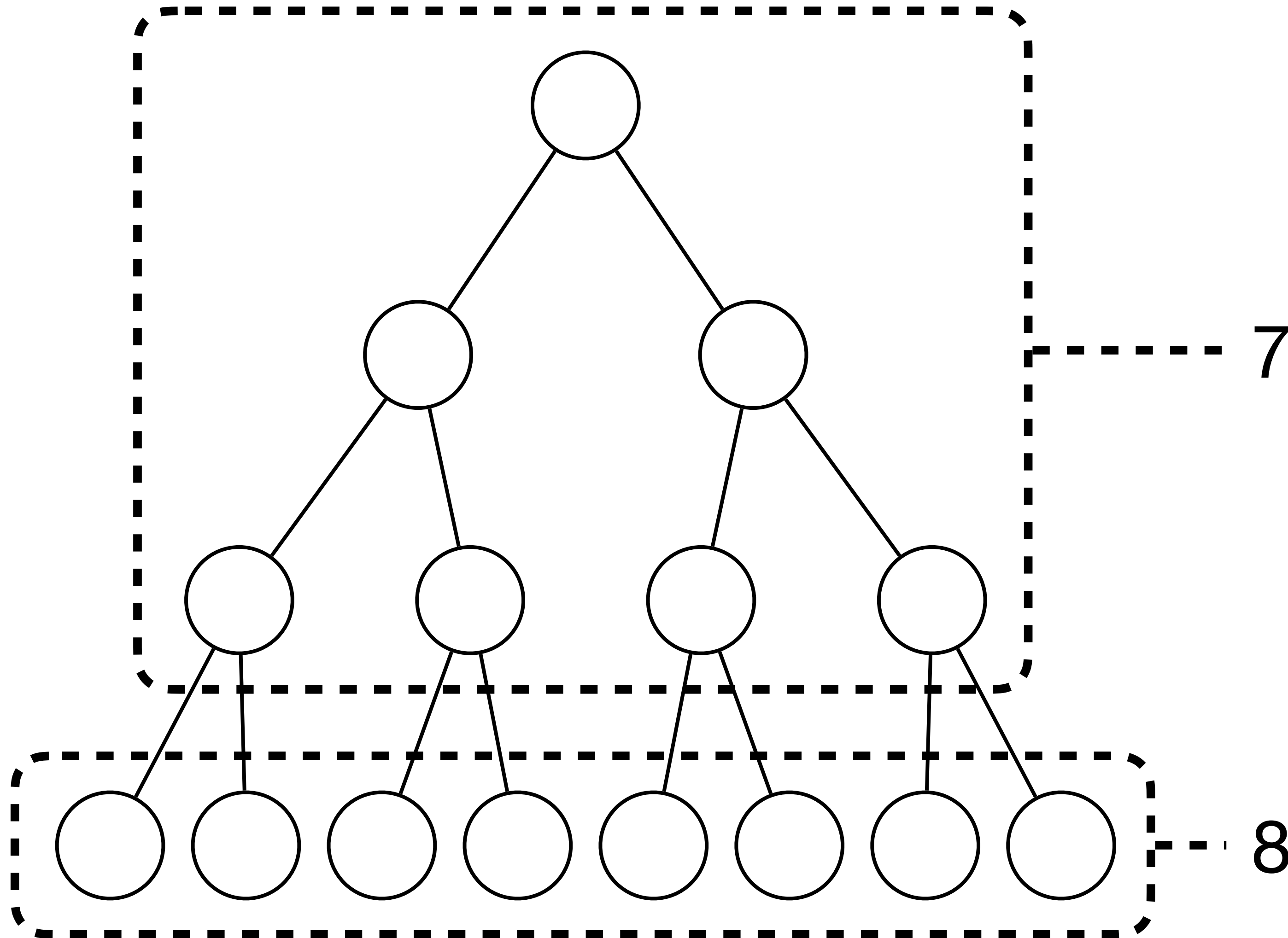
Remarque



Remarque



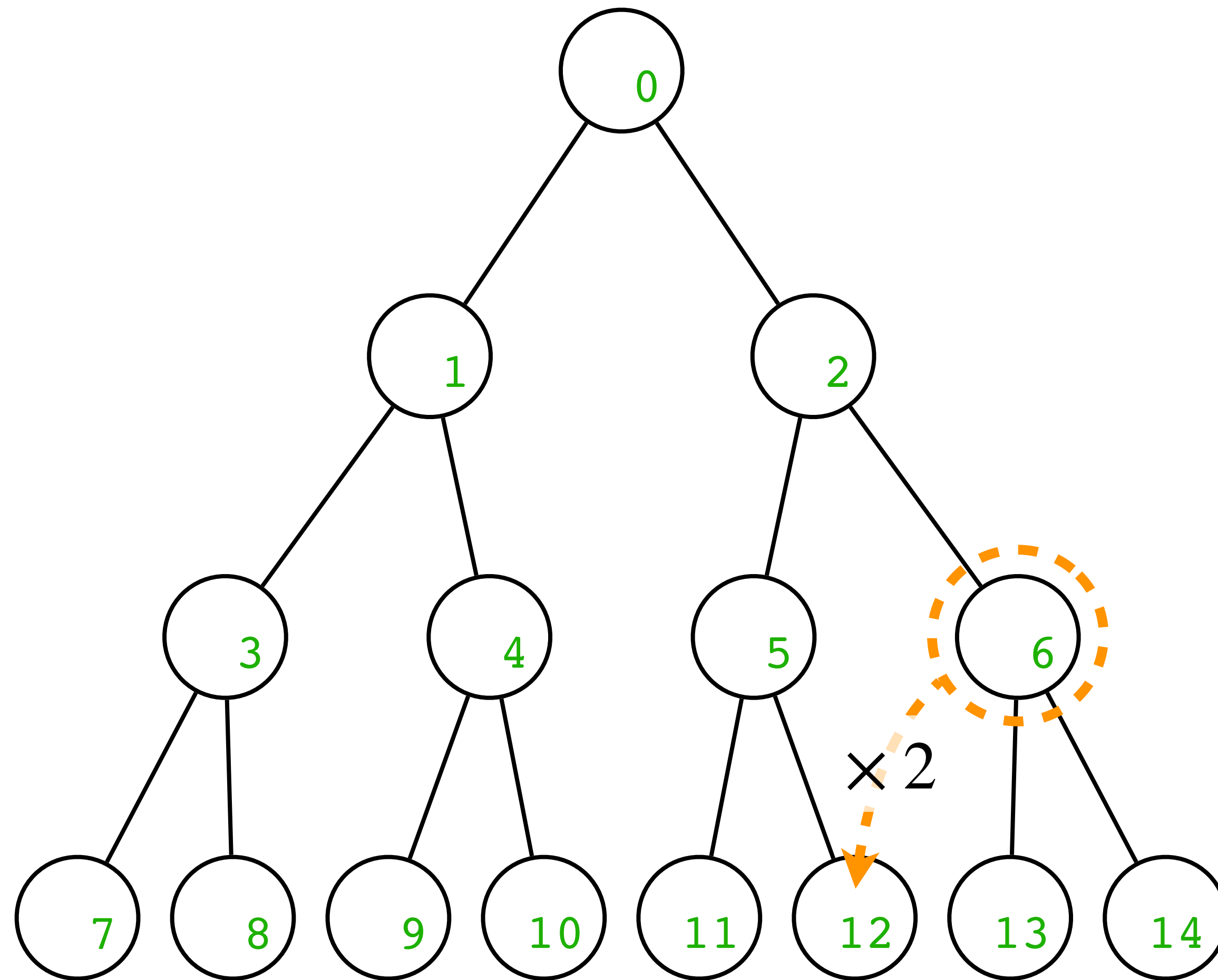
Remarque



En général :

Un arbre binaire avec k feuilles
a $k - 1$ noeuds intermédiaires.

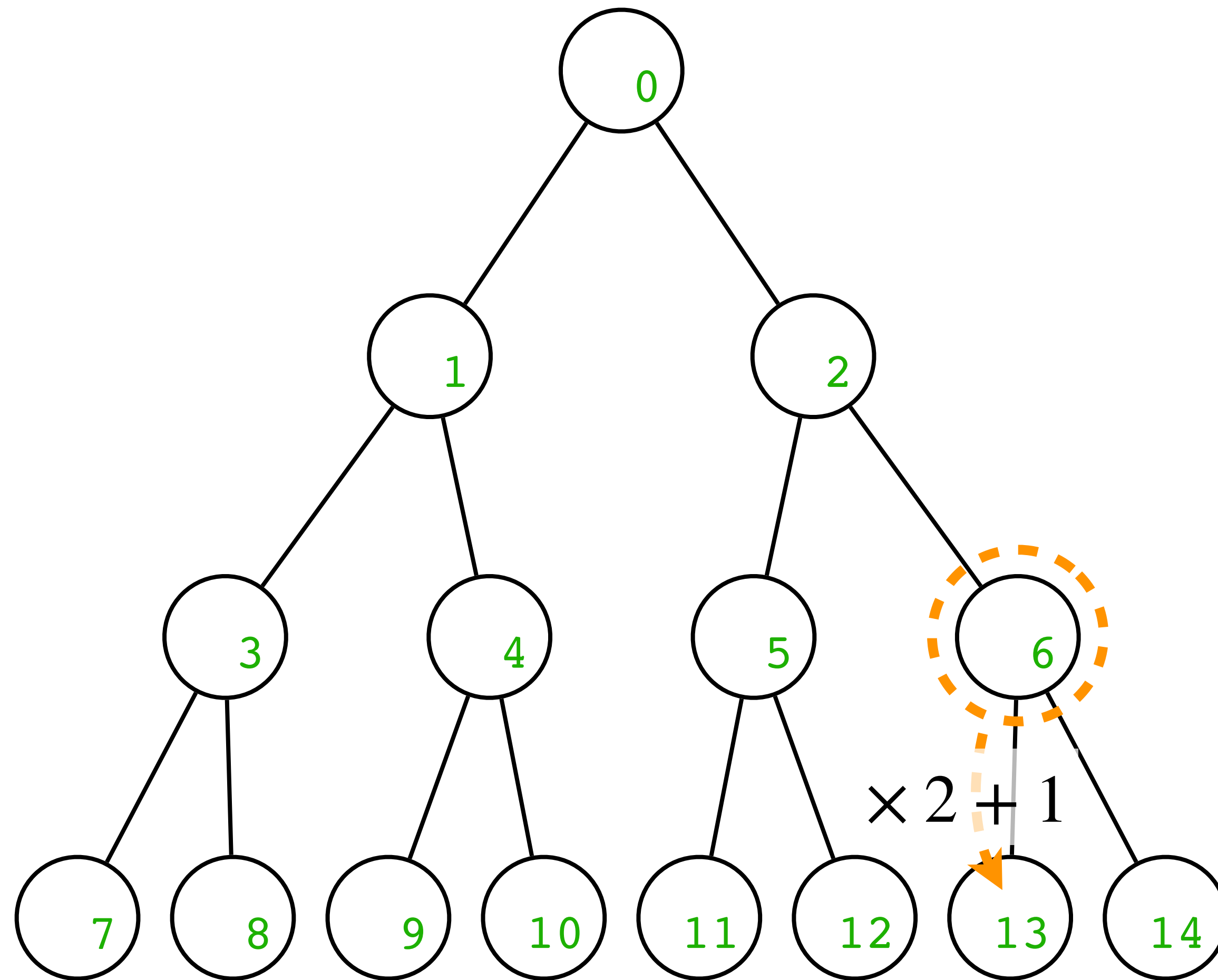
Remarque



En général :

Un arbre binaire avec k feuilles
a $k - 1$ noeuds intermédiaires.

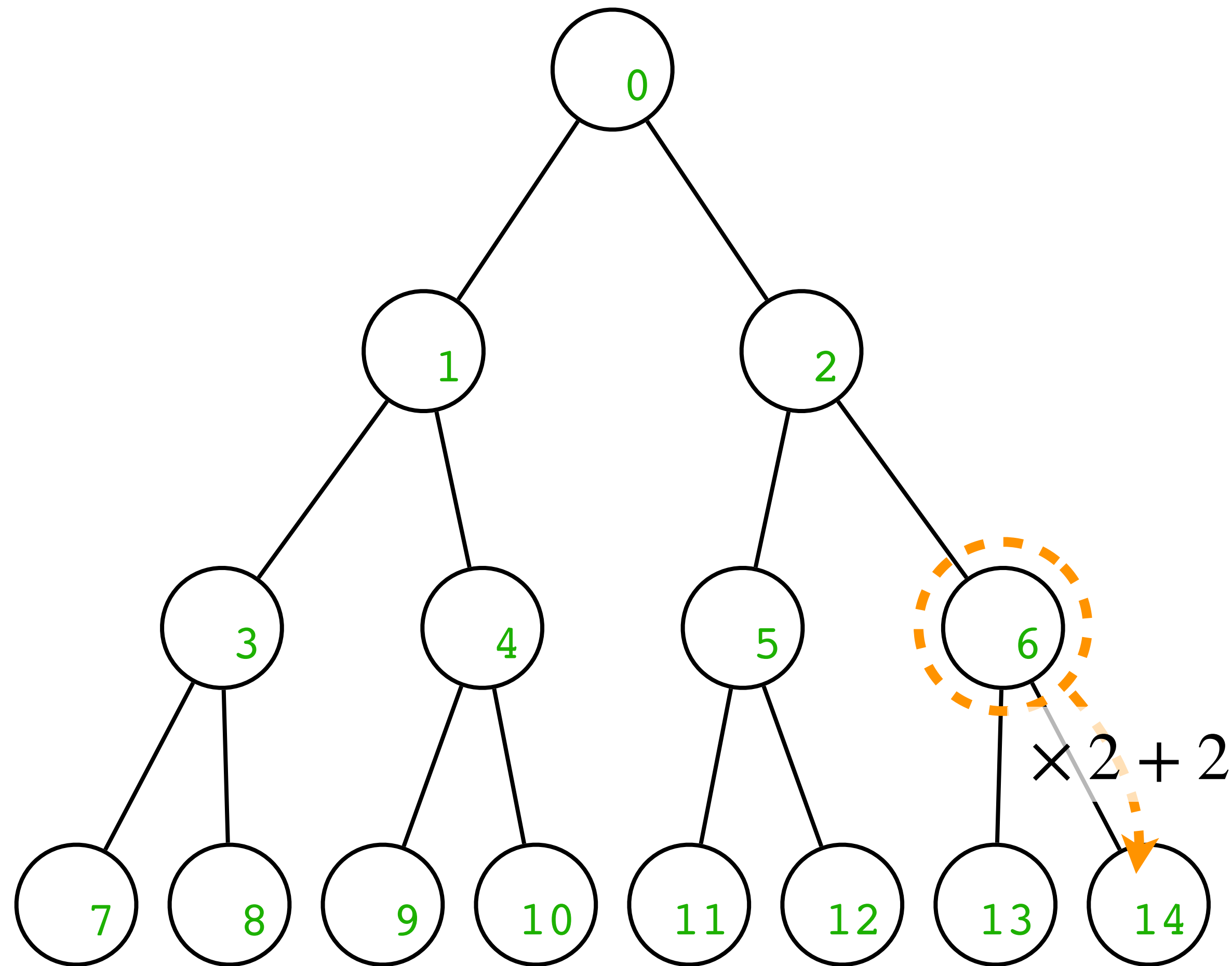
Remarque



En général :

Un arbre binaire avec k feuilles
a $k - 1$ noeuds intermédiaires.

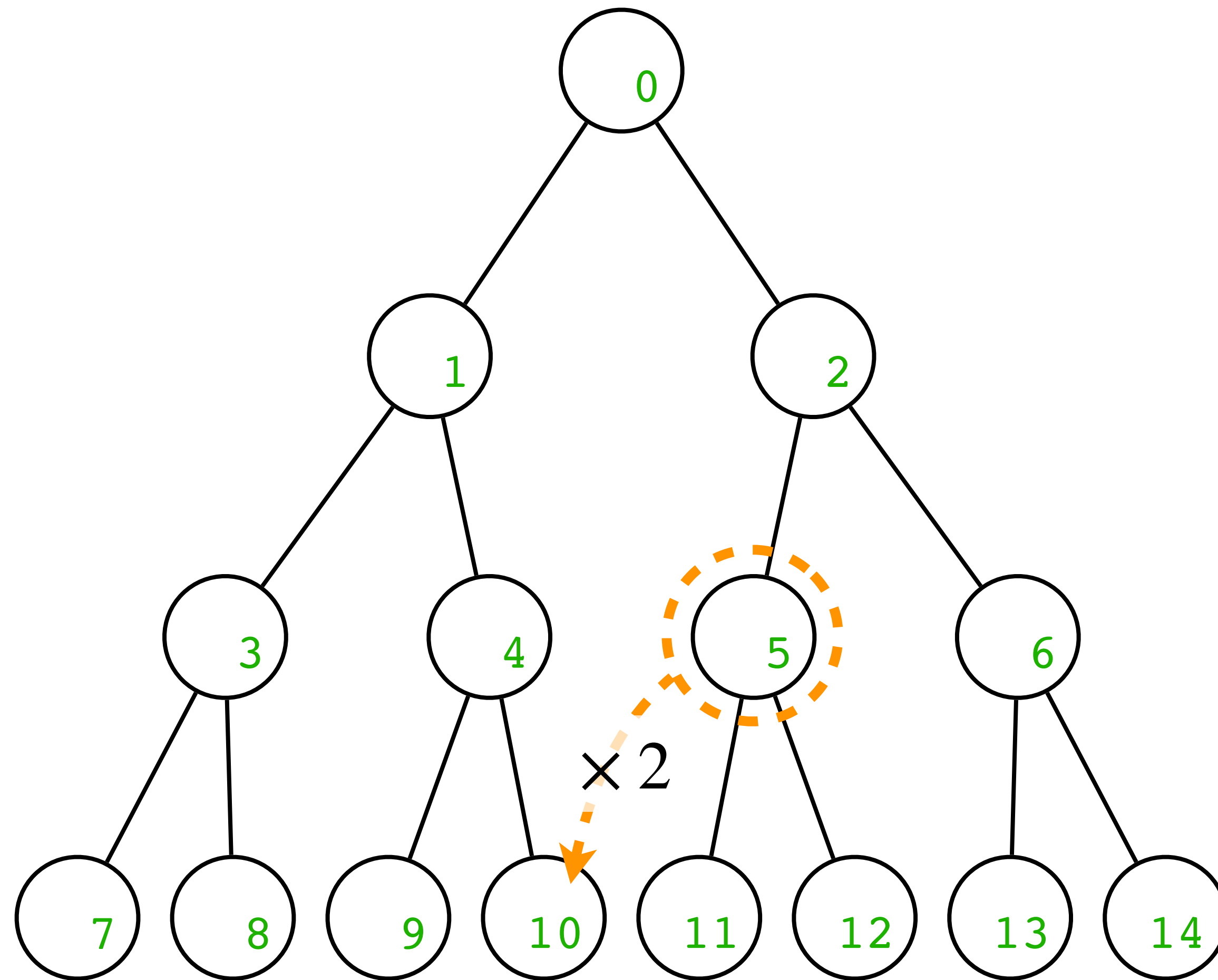
Remarque



En général :

Un arbre binaire avec k feuilles
a $k - 1$ noeuds intermédiaires.

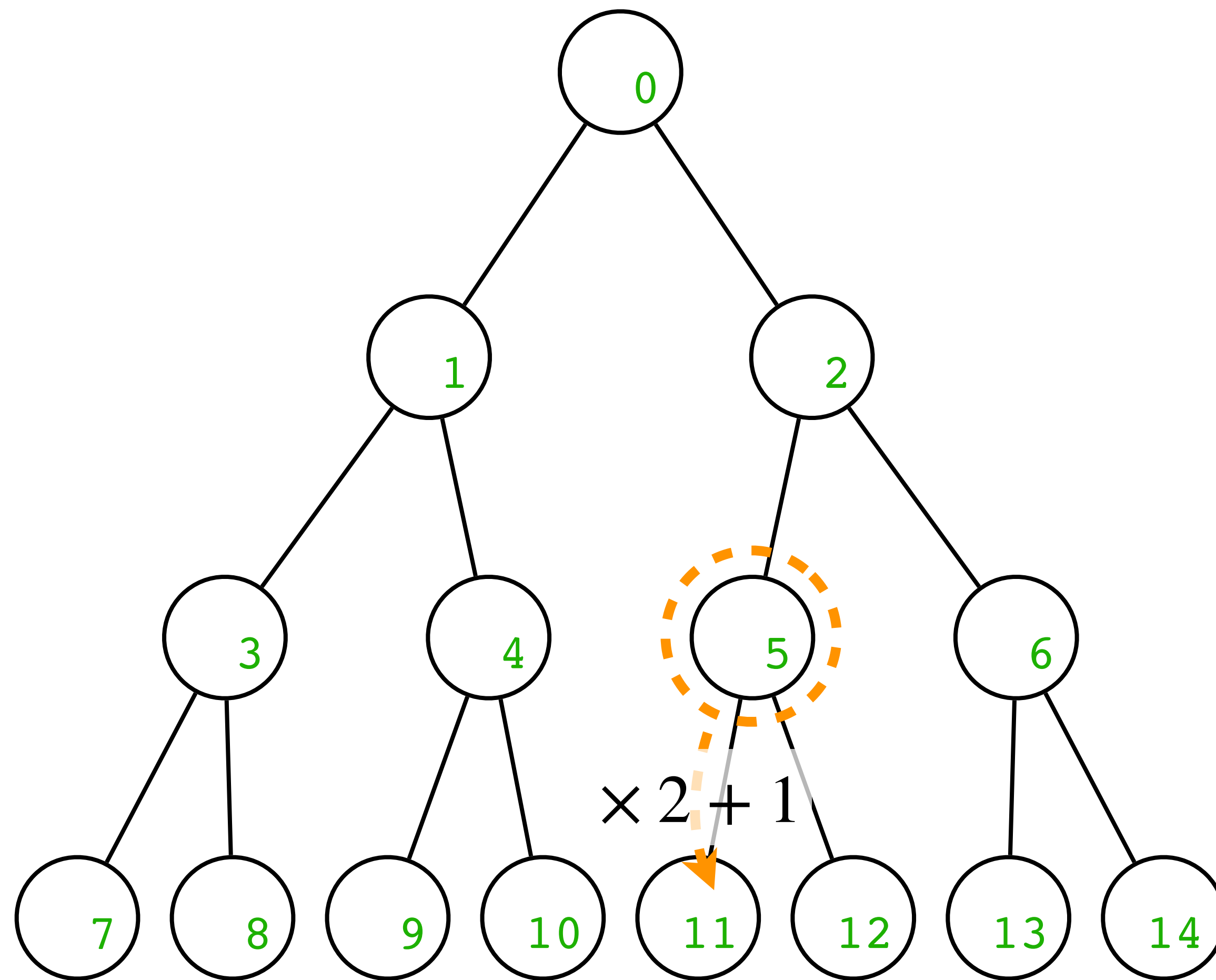
Remarque



En général :

Un arbre binaire avec k feuilles
a $k - 1$ noeuds intermédiaires.

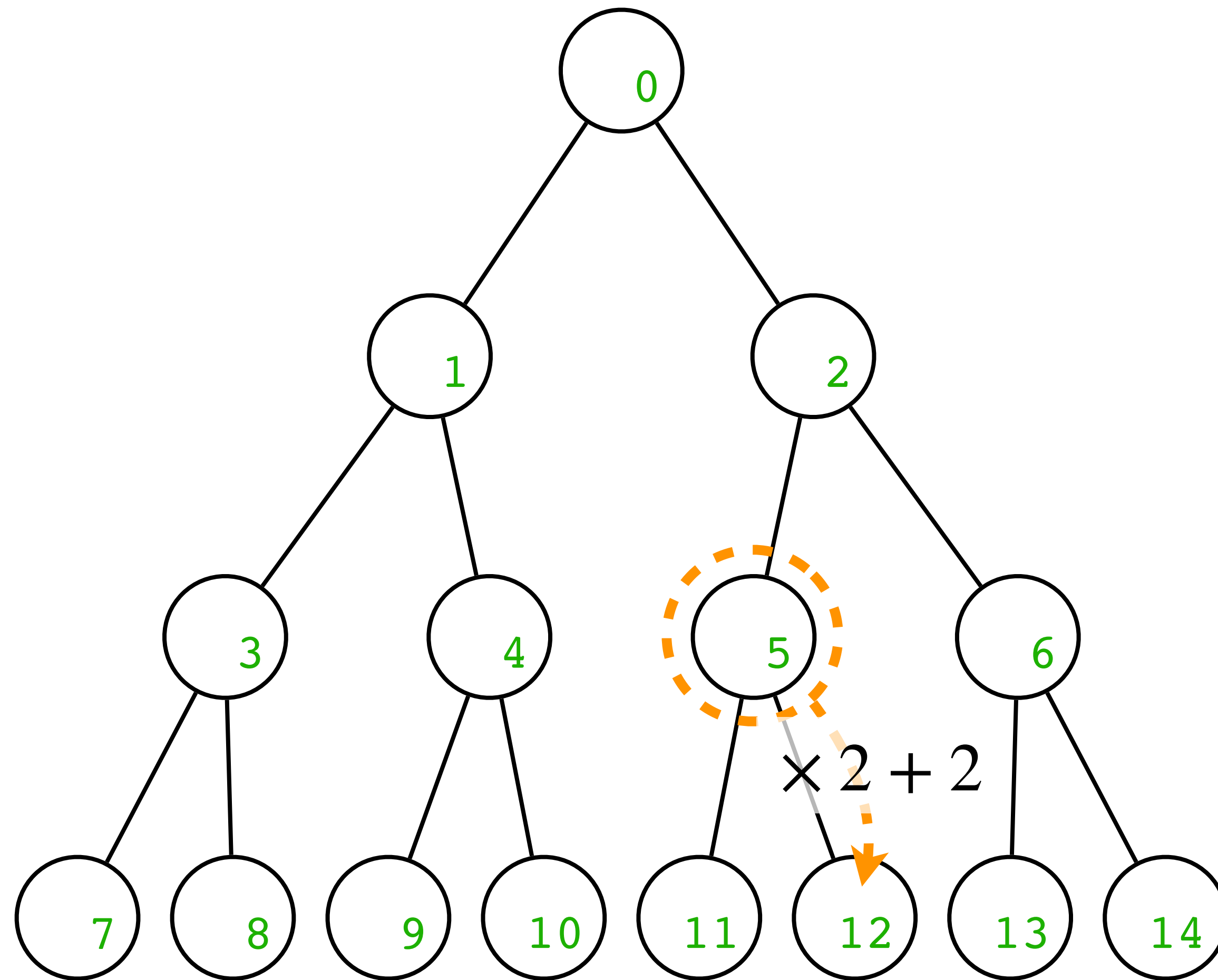
Remarque



En général :

Un arbre binaire avec k feuilles
a $k - 1$ noeuds intermédiaires.

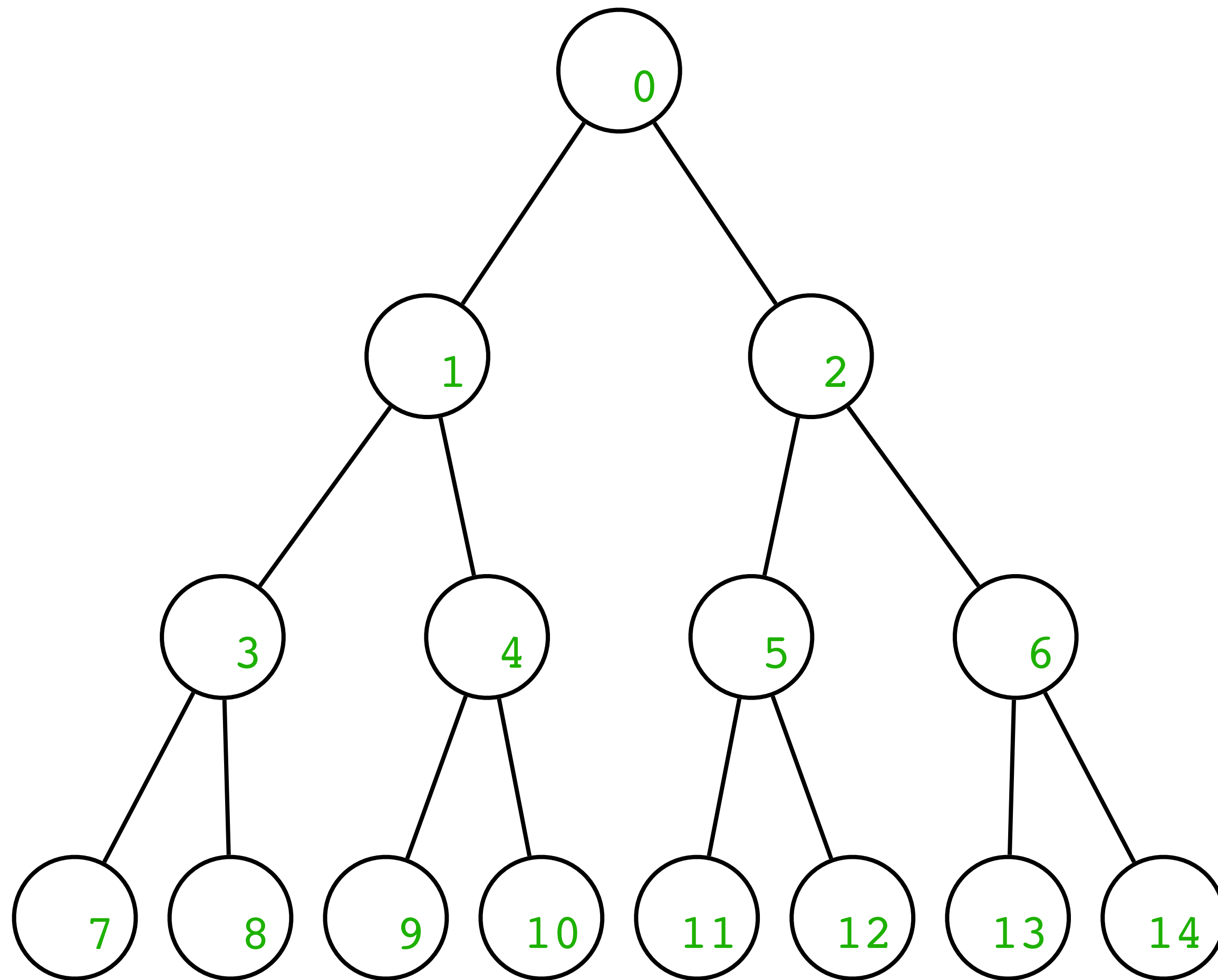
Remarque



En général :

Un arbre binaire avec k feuilles
a $k - 1$ noeuds intermédiaires.

Remarque



En général :

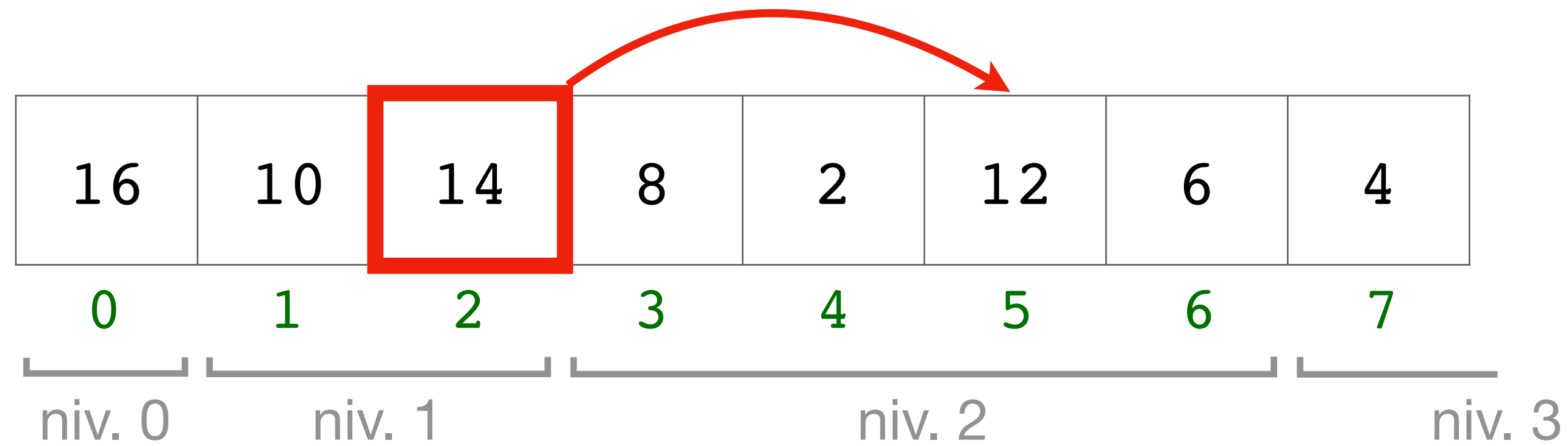
Un arbre binaire avec k feuilles
a $k - 1$ noeuds intermédiaires.

Donc :

$$2i + 1 = \text{enfant}_1(i)$$

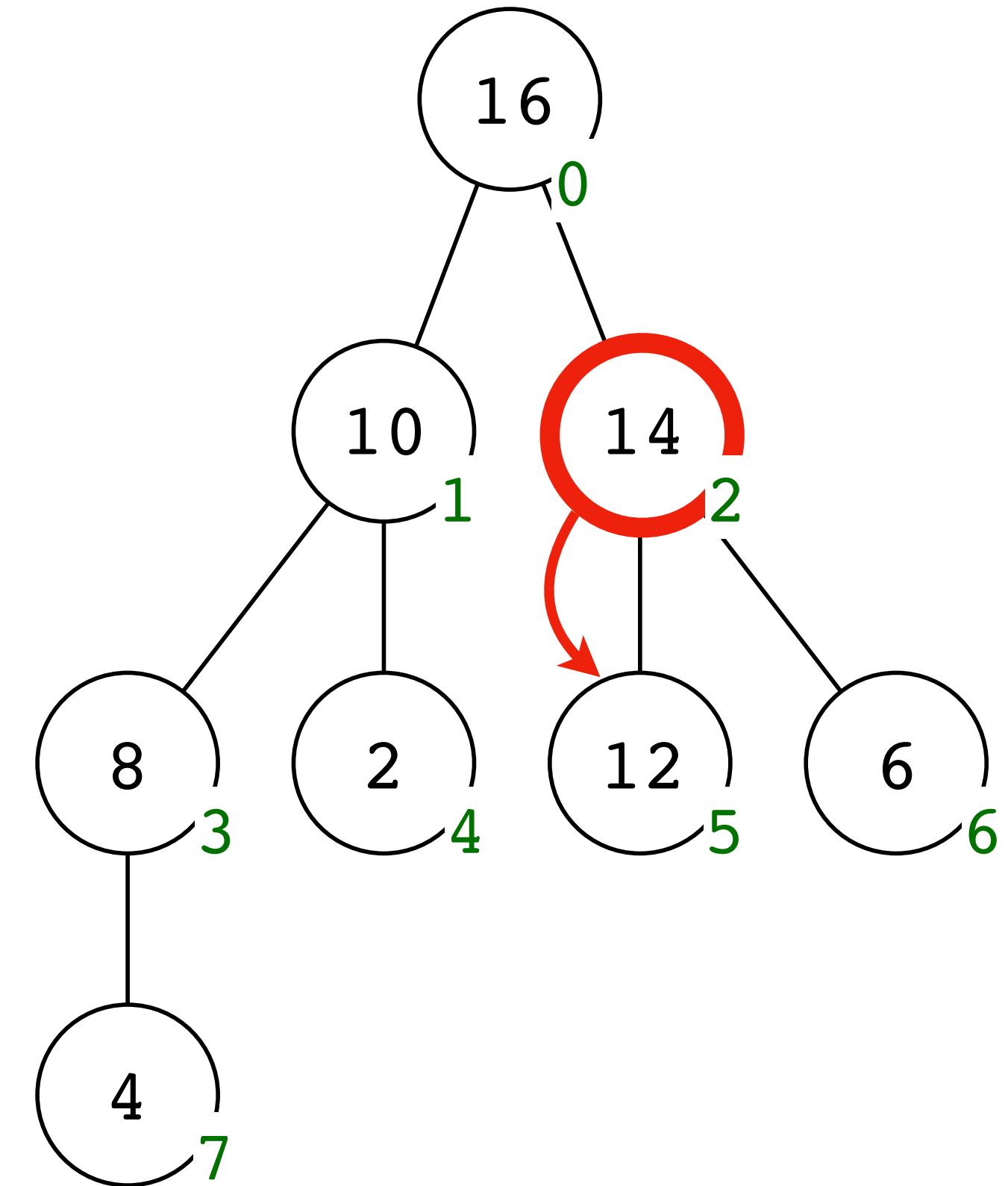
$$2i + 2 = \text{enfant}_2(i)$$

Indices et parenté

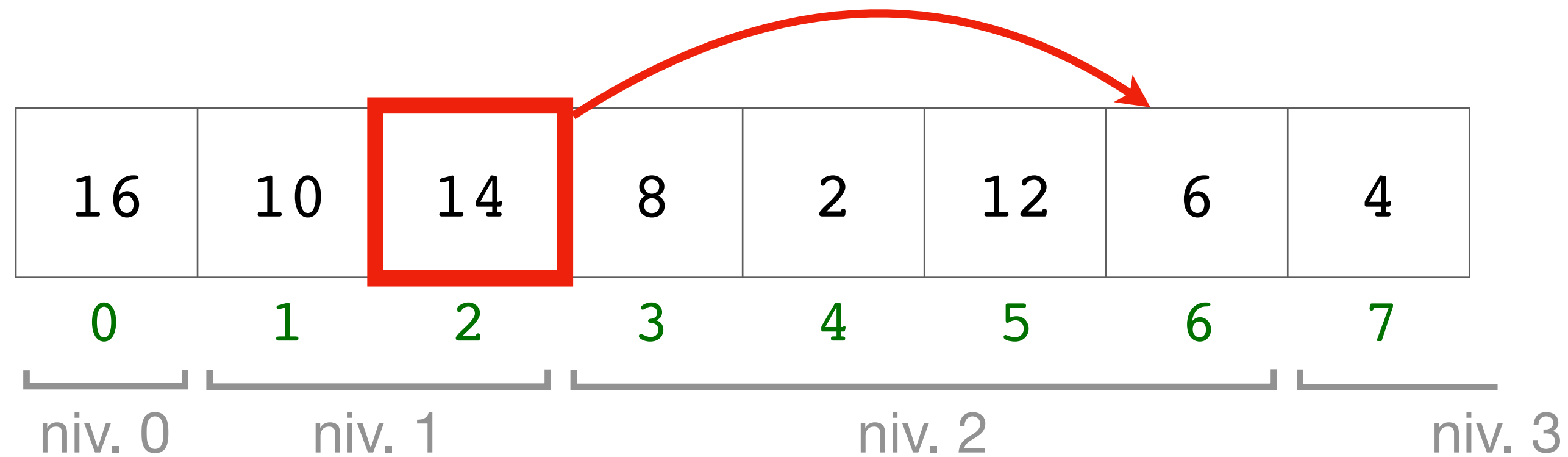


```

fonction el(i)
  retourner 2 * i + 1
  
```

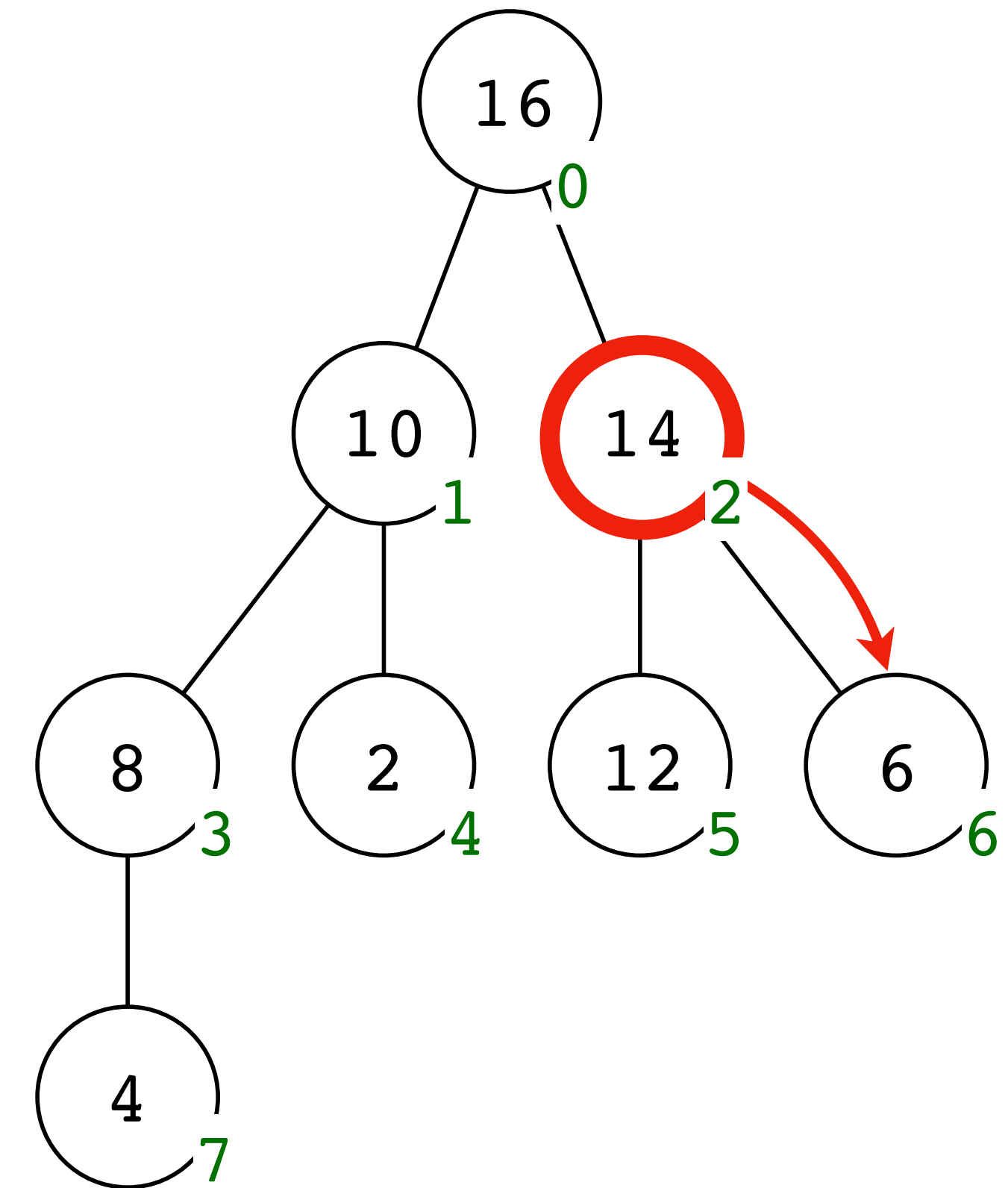


Indices et parenté

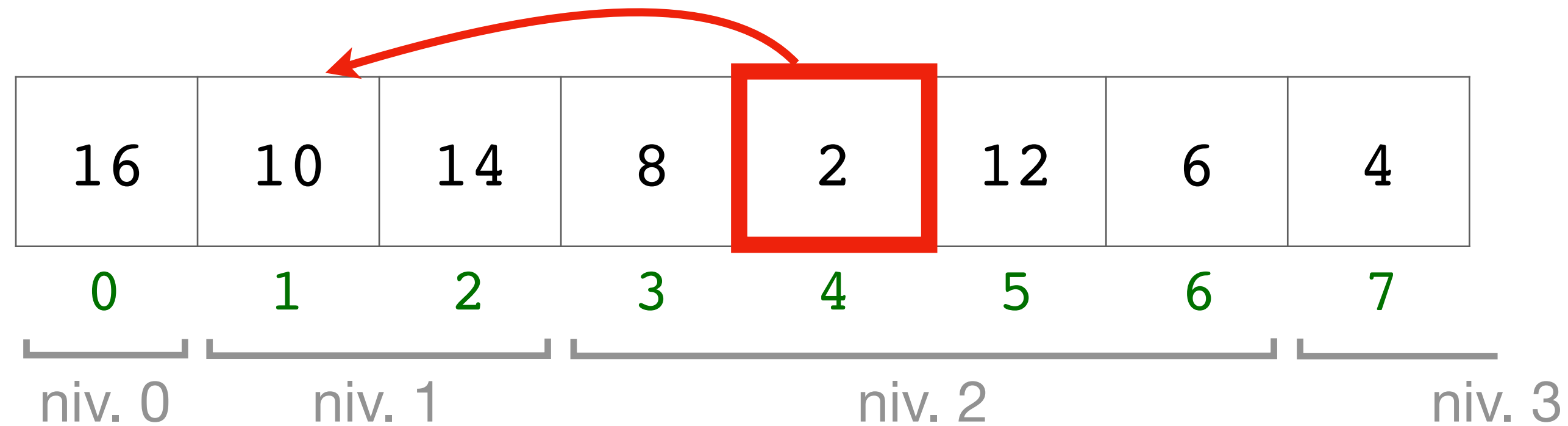


```
fonction e1(i)
  retourner 2 * i + 1
```

```
fonction e2(i)
  retourner 2 * i + 2
```



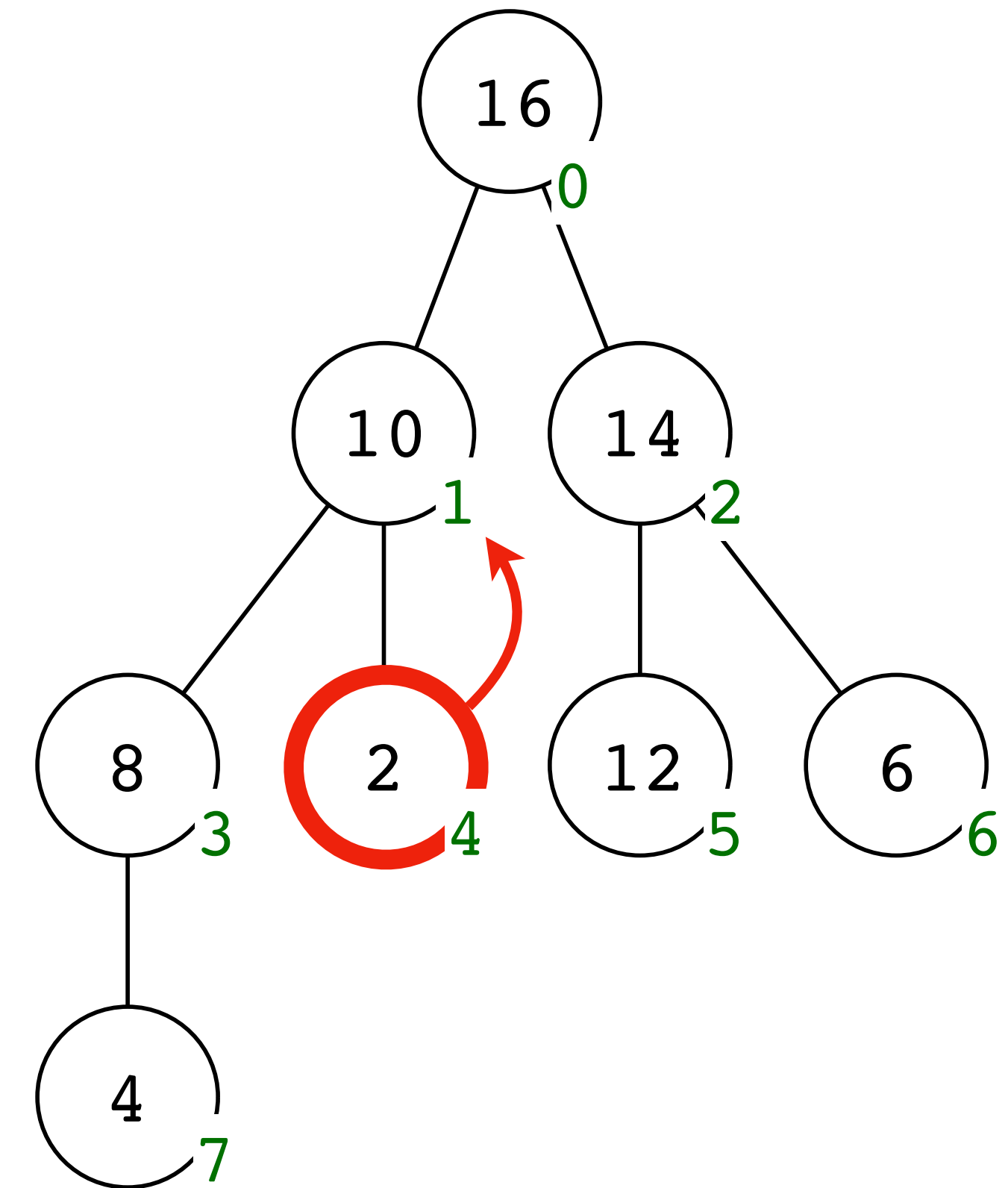
Indices et parenté



```
fonction e1(i)
    retourner 2 * i + 1
```

```
fonction e2(i)
    retourner 2 * i + 2
```

```
fonction parent(i)
    retourner (i - 1) / 2
    (division entière)
```



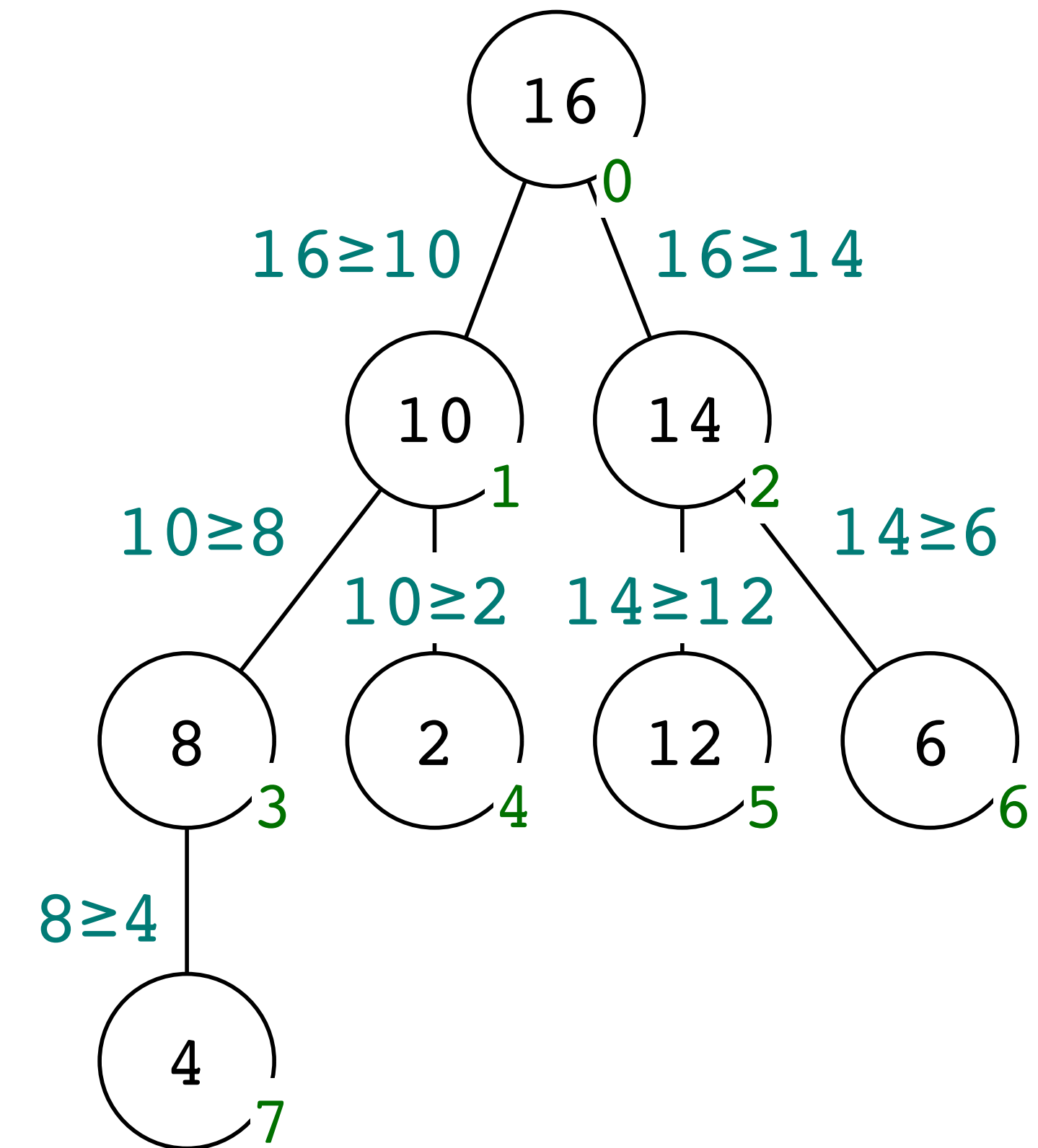
Condition d'ordre partiel



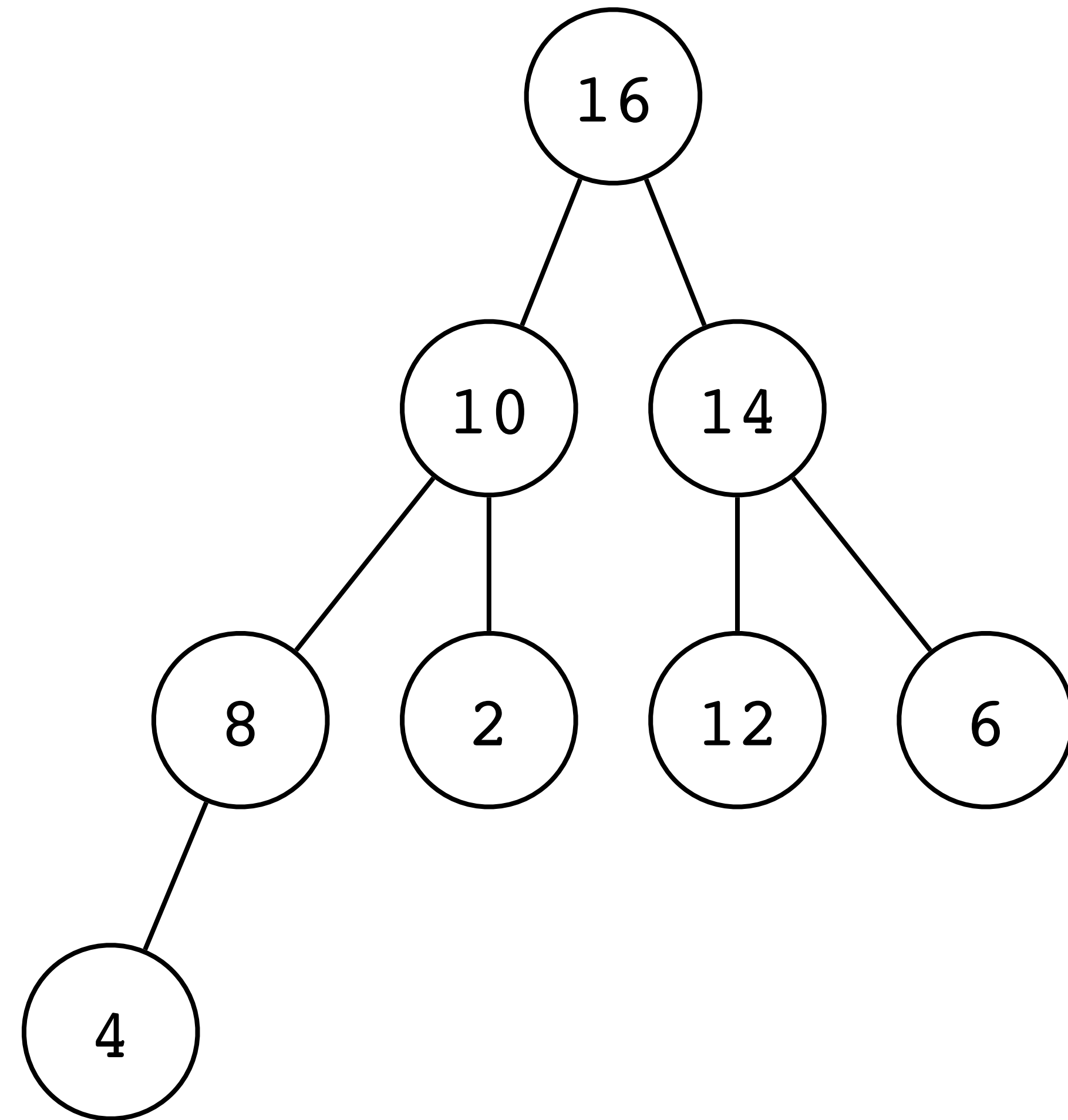
16	10	14	8	2	12	6	4
0	1	2	3	4	5	6	7

$$\forall i > 0, \\ T[\text{parent}(i)] \geq T[i]$$

*(“Pour tout index i strictement positif,
la valeur du parent de i est plus grande ou égale à celle de i ”)*



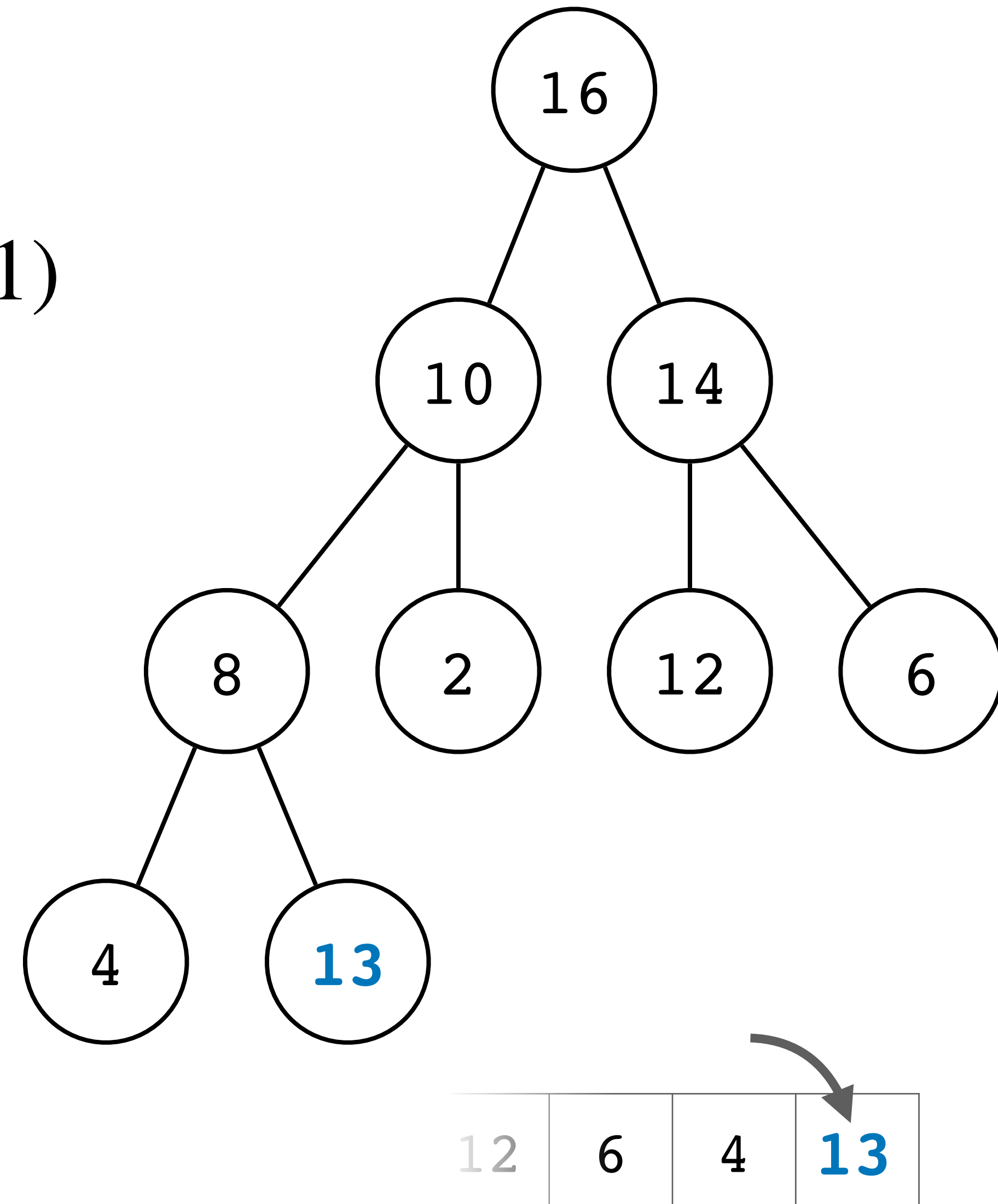
Insertion



Insertion



Étape 1 : Insertion en fin de tableau : $O(1)$

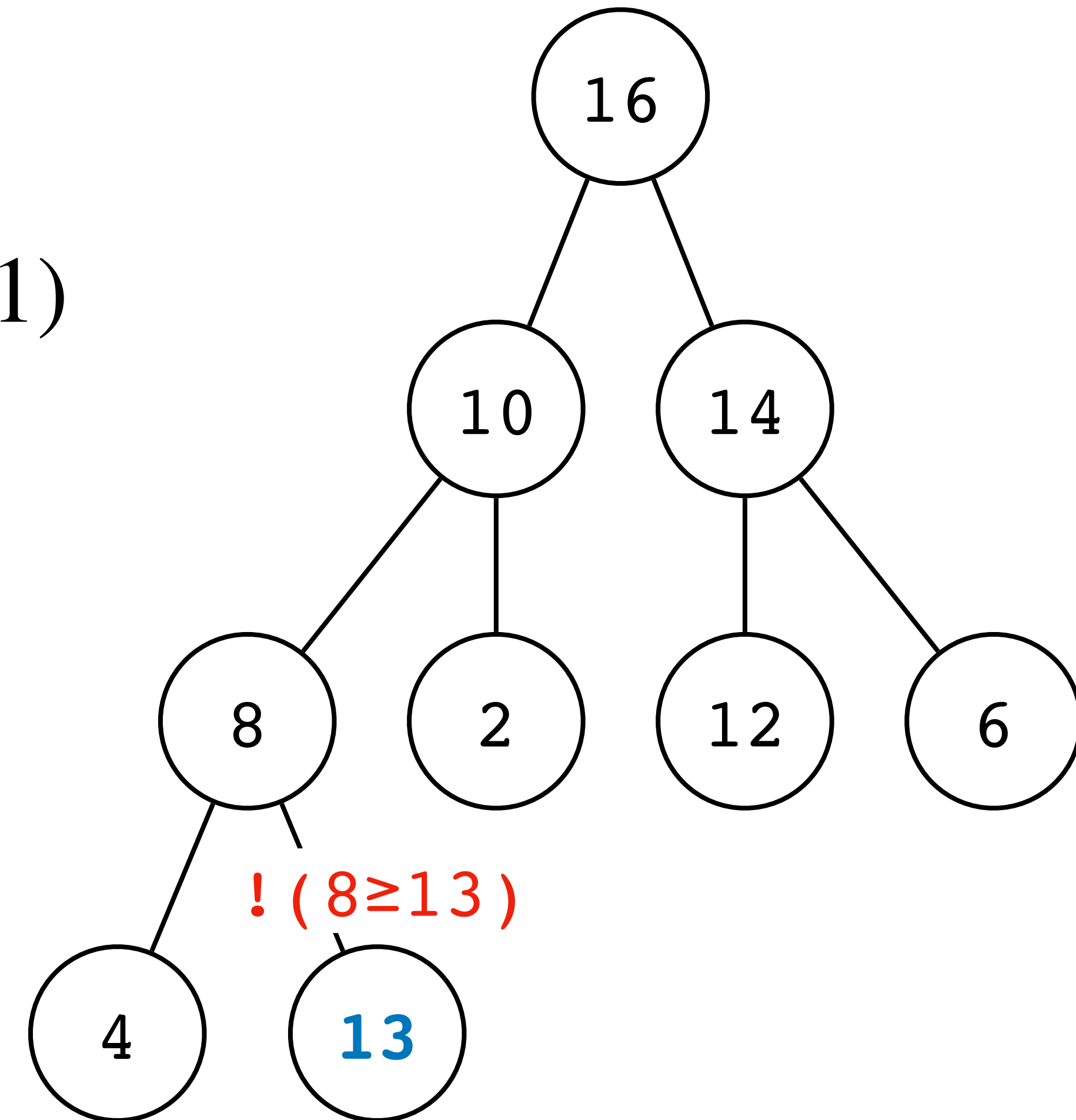


Insertion



Étape 1 : Insertion en fin de tableau : $O(1)$

Problème : *condition de tas perdue*



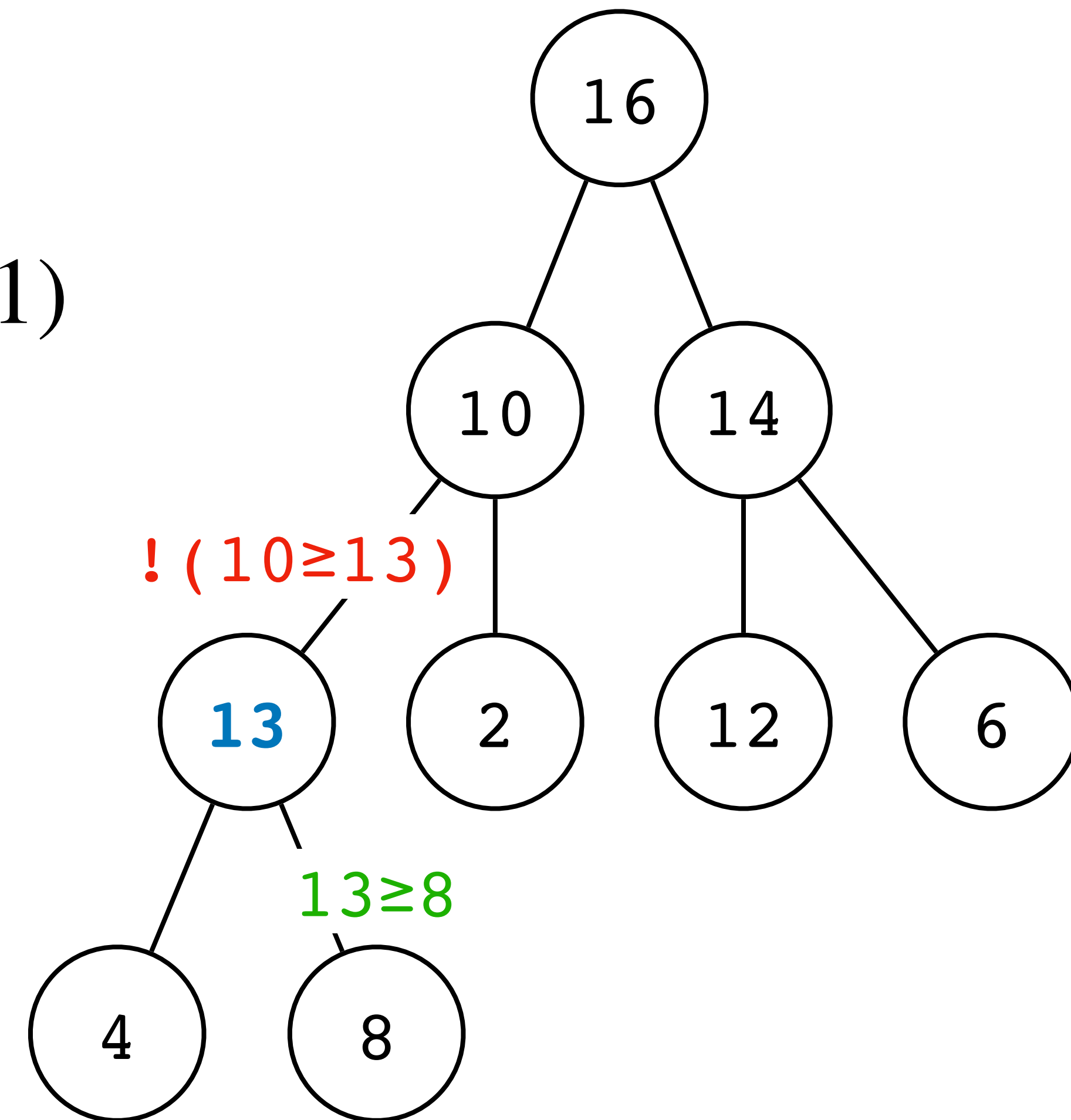
Insertion



Étape 1 : Insertion en fin de tableau : $O(1)$

Problème : *condition de tas perdue*

Étape 2 : Rétablissement de la condition
en remontant l'élément inséré :



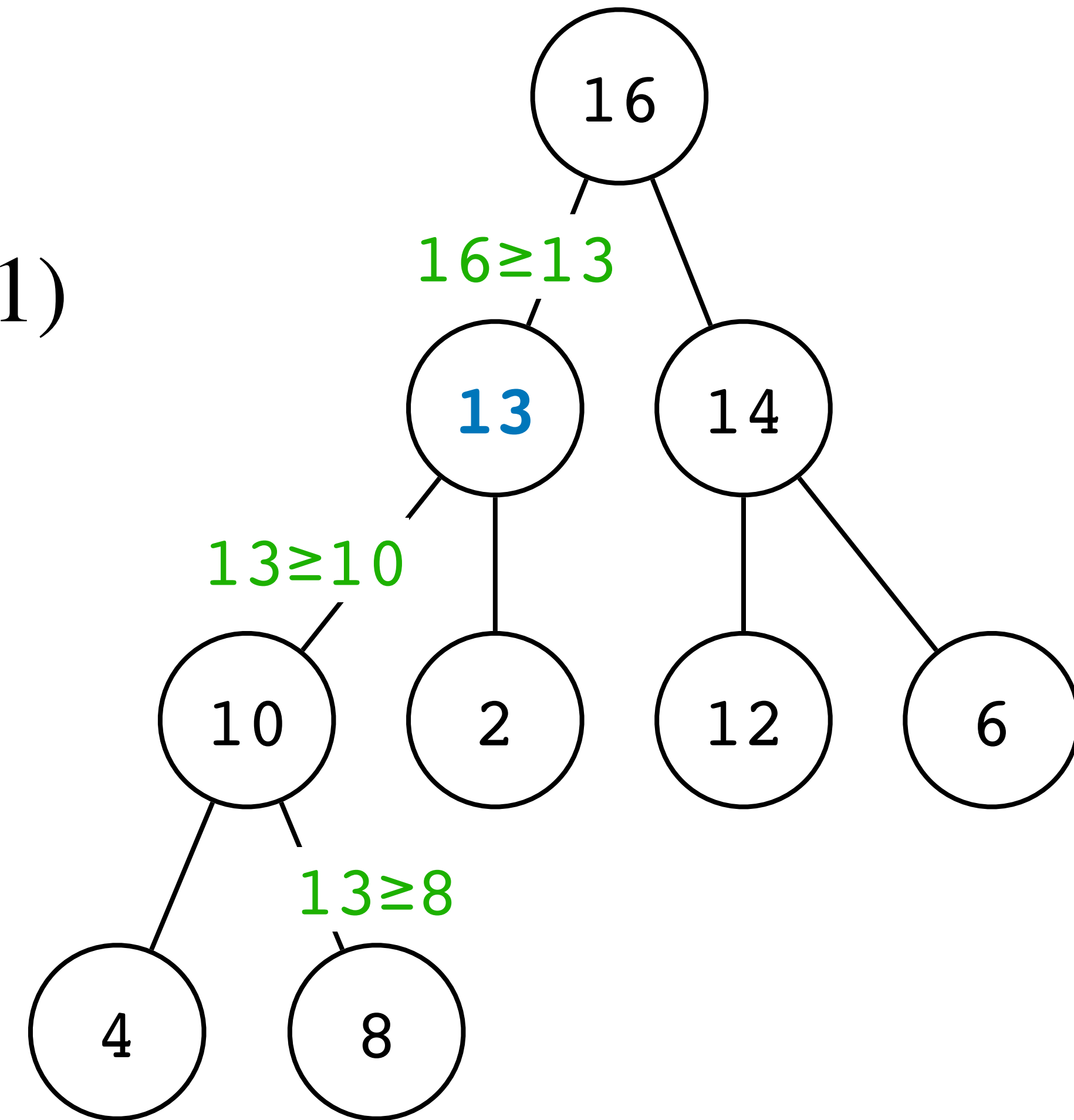
Insertion



Étape 1 : Insertion en fin de tableau : $O(1)$

Problème : *condition de tas perdue*

Étape 2 : Rétablissement de la condition
en remontant l'élément inséré :



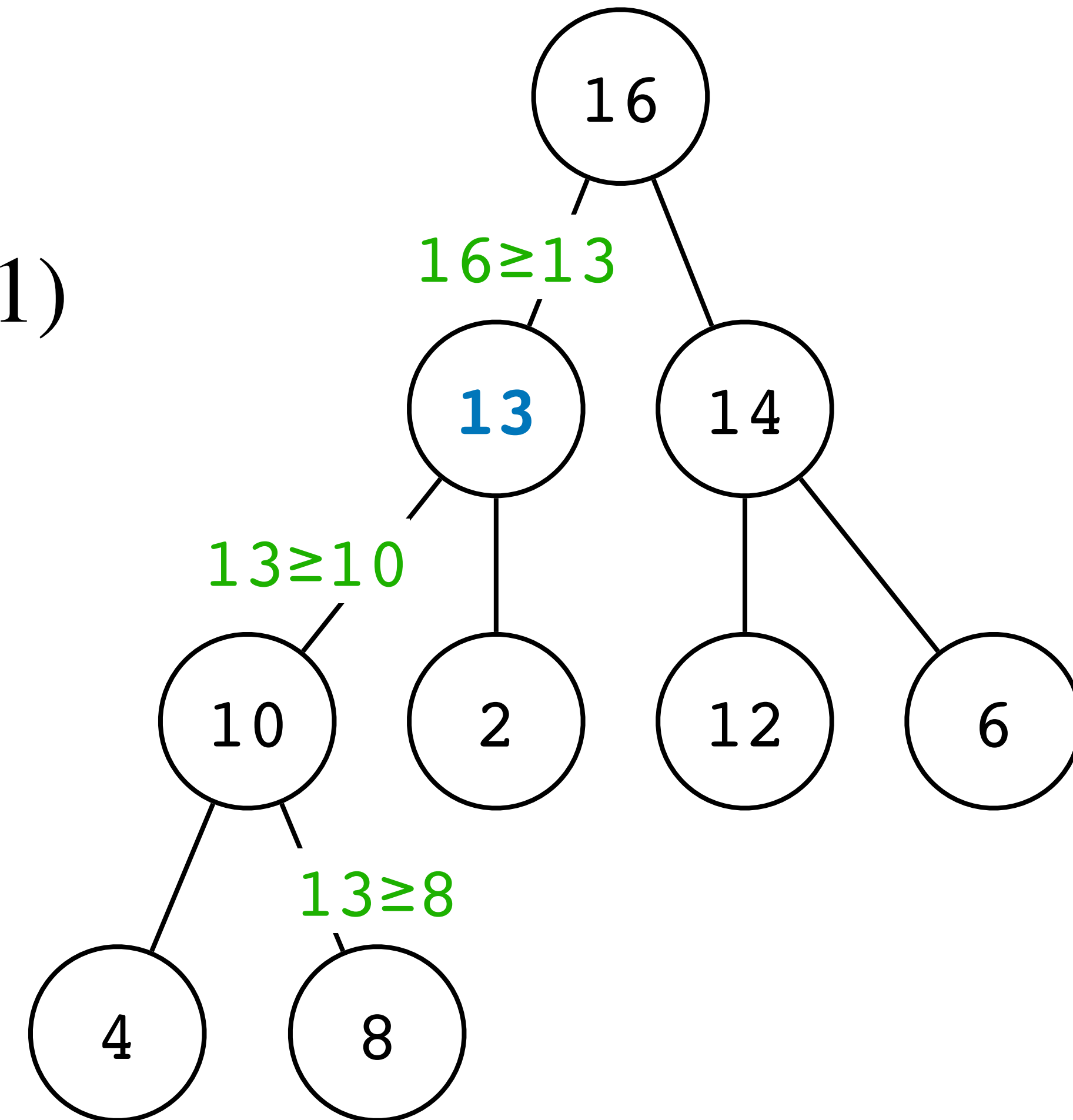
Insertion



Étape 1 : Insertion en fin de tableau : $O(1)$

Problème : condition de tas perdue

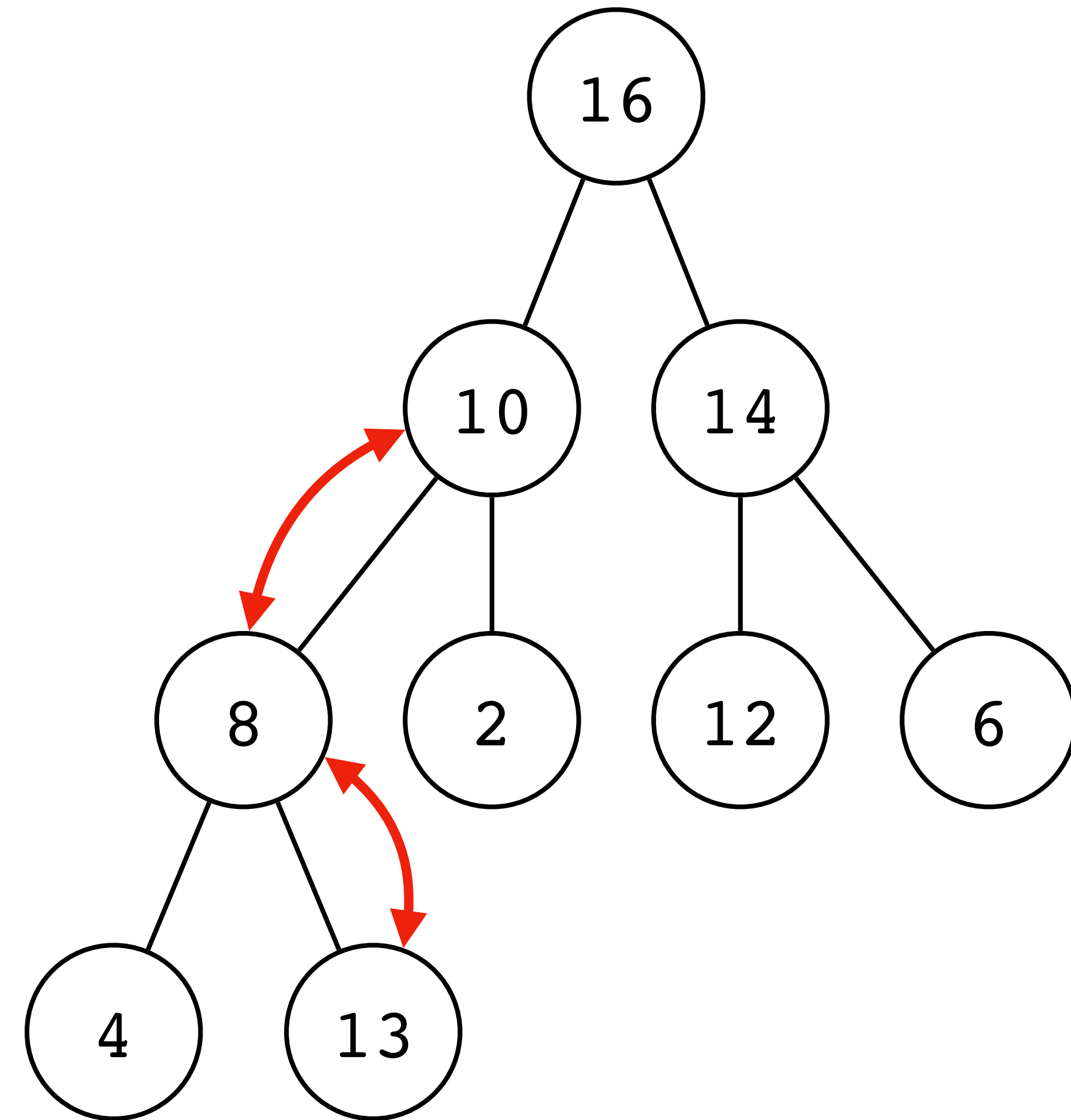
Étape 2 : Rétablissement de la condition
en remontant l'élément inséré :
 $O(\log(n))$



Remonter : algorithme



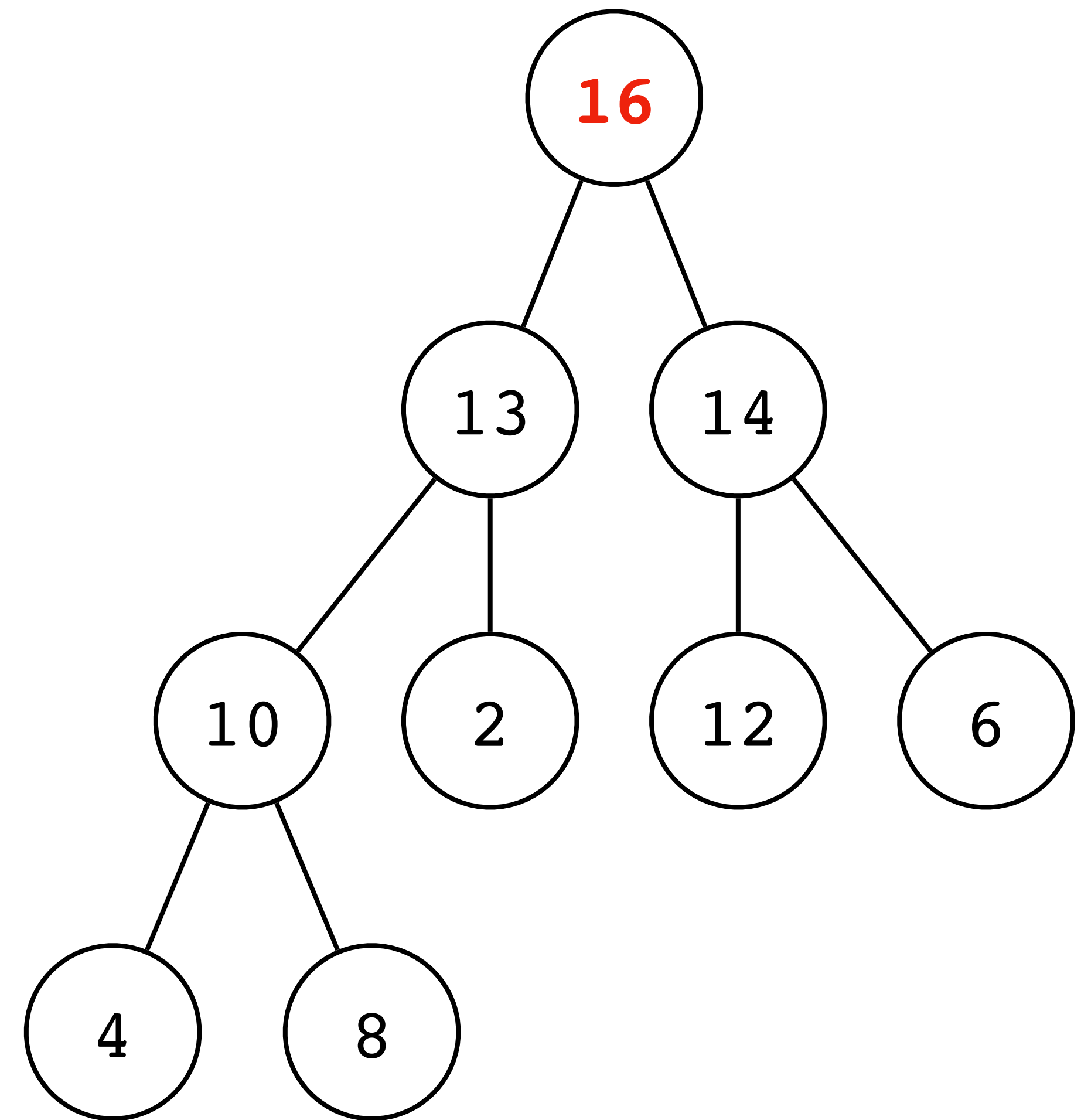
```
fonction remonter(T,i)
    tant que i > 0 et T[parent(i)] < T[i]
        échanger T[i], T[parent(i)]
        i ← parent(i)
```



Suppression du sommet



Suppression compliquée en sommet...
...plus simple en fin :

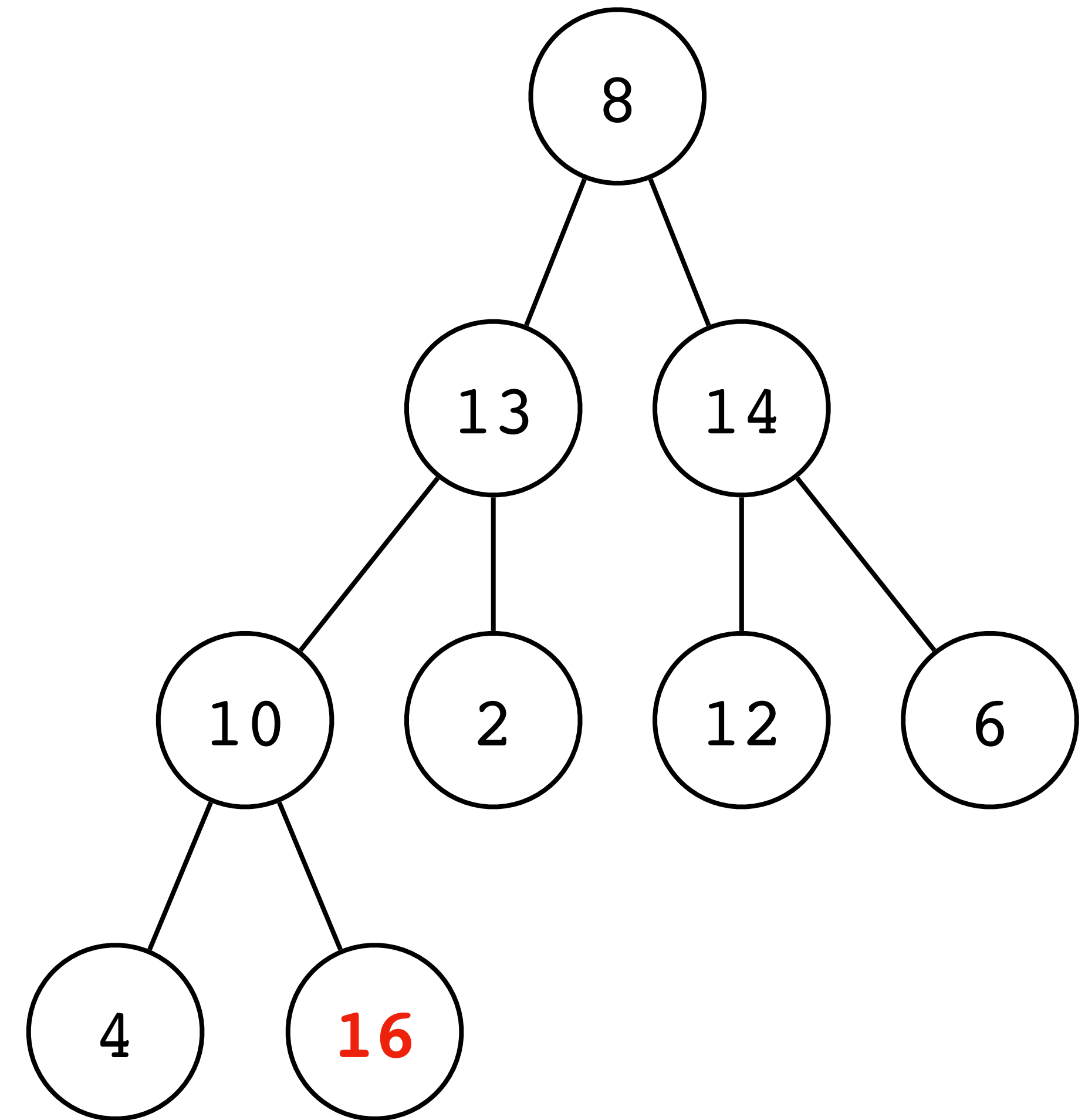


Suppression du sommet



Suppression compliquée en sommet...
...plus simple en fin :

Étape 1 : Permutation premier/dernier



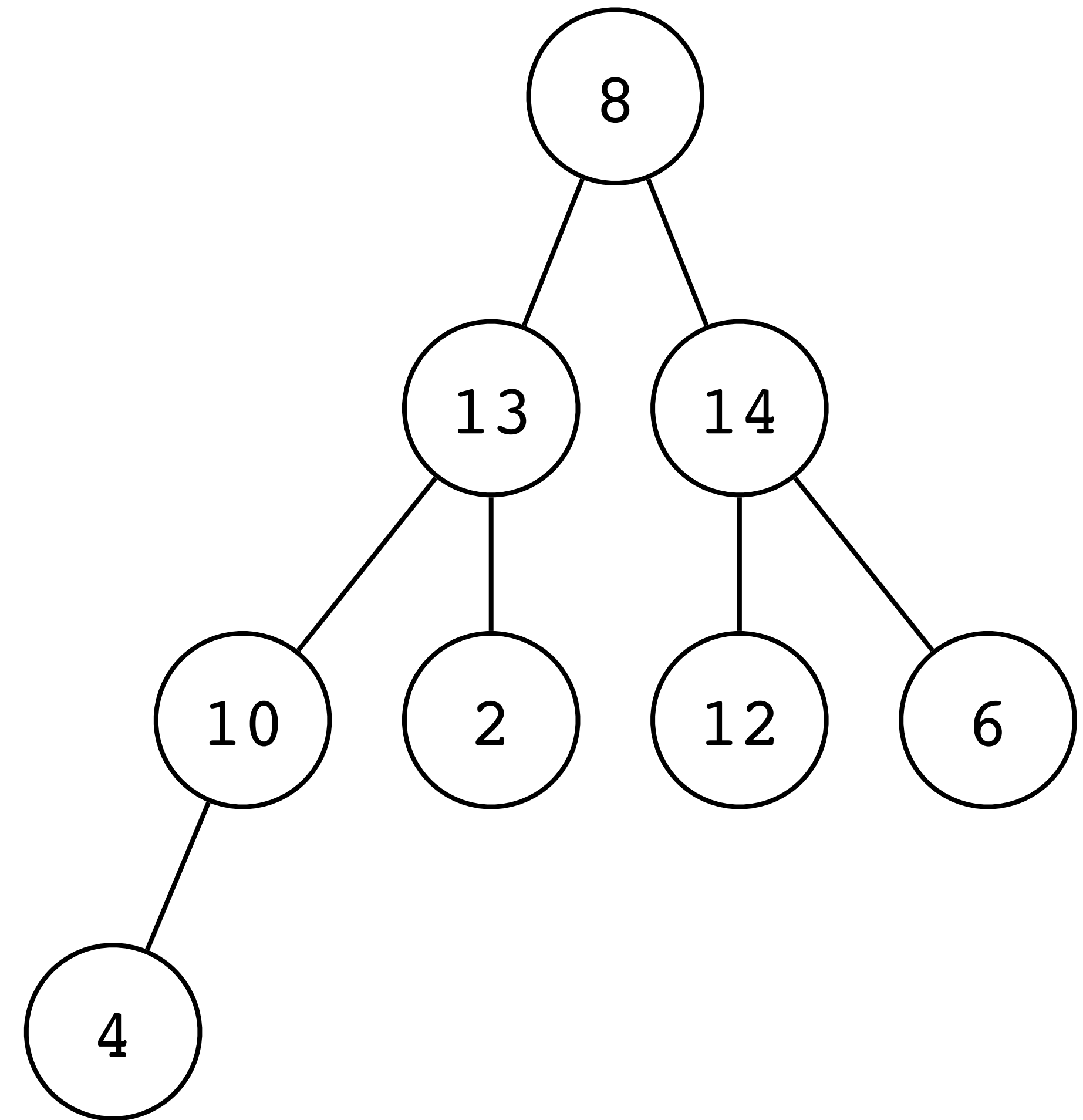
Suppression du sommet



Suppression compliquée en sommet...
...plus simple en fin :

Étape 1 : Permutation premier/dernier

Étape 2 : Suppression du dernier



Suppression du sommet

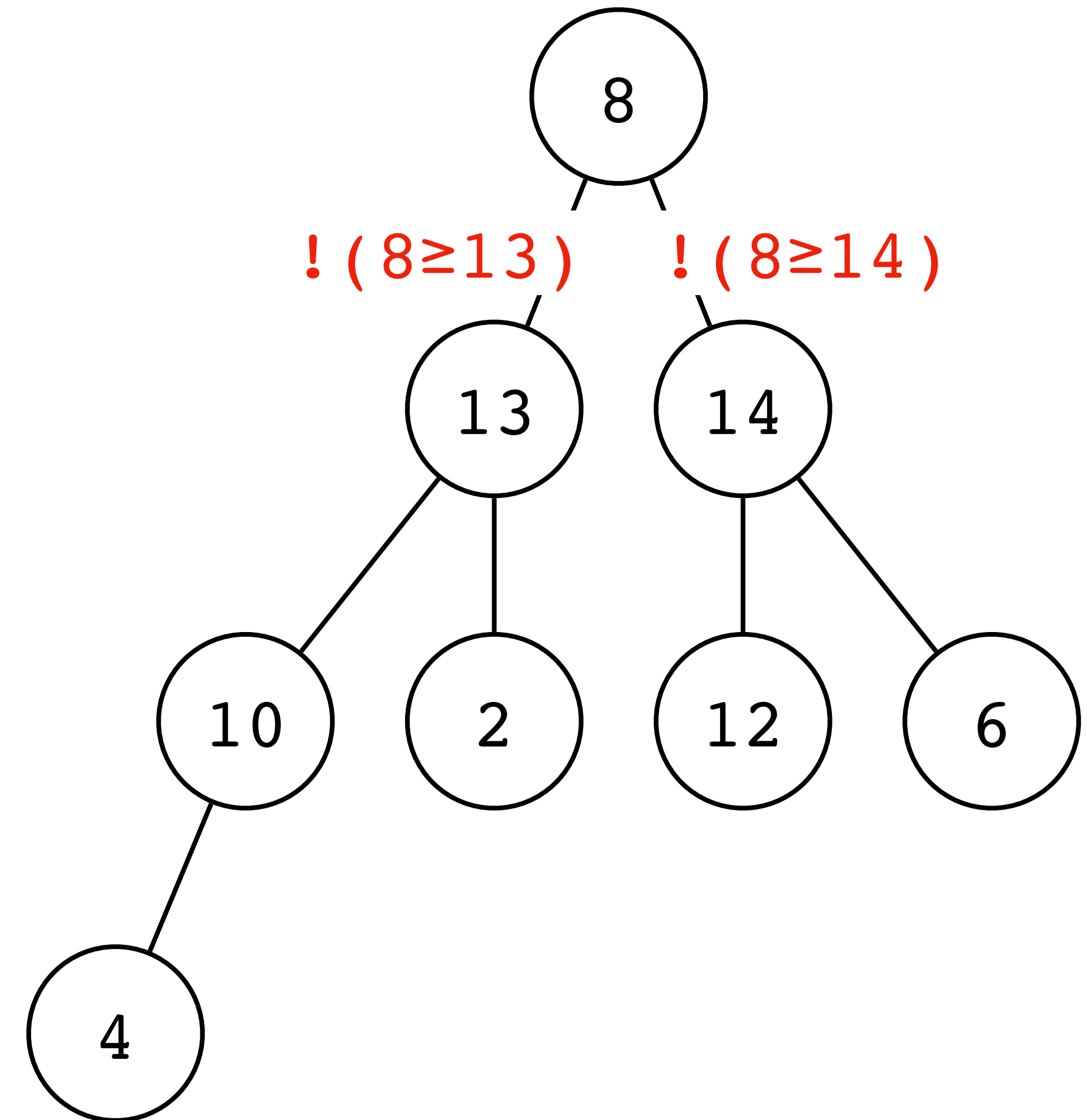


Suppression compliquée en sommet...
...plus simple en fin :

Étape 1 : Permutation premier/dernier

Étape 2 : Suppression du dernier

Problème : *Condition de tas perdue*



Suppression du sommet



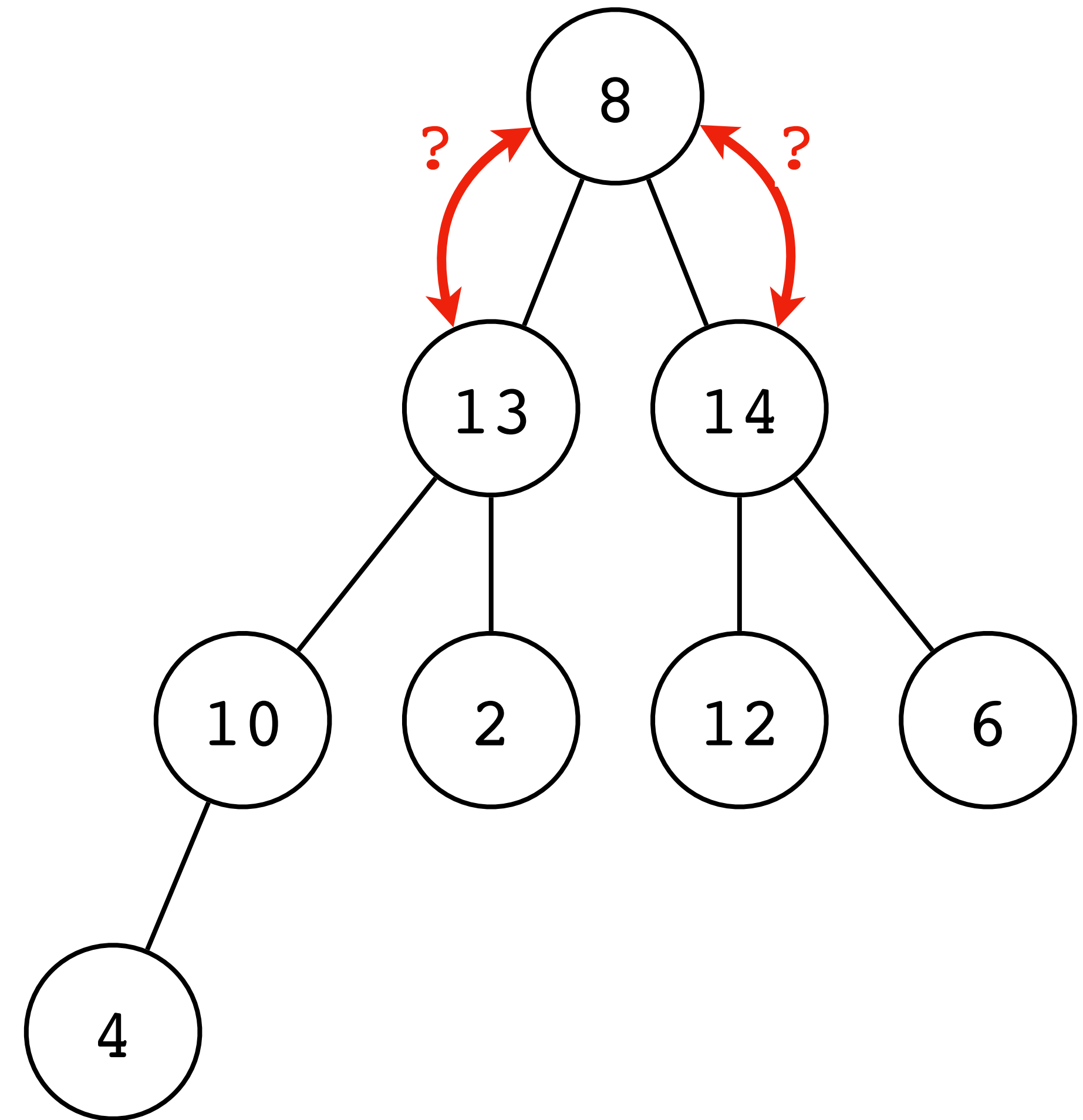
Suppression compliquée en sommet...
...plus simple en fin :

Étape 1 : Permutation premier/dernier

Étape 2 : Suppression du dernier

Problème : *Condition de tas perdue*

Étape 3 : Rétablissement de la condition
en descendant la racine



Suppression du sommet



Suppression compliquée en sommet...
...plus simple en fin :

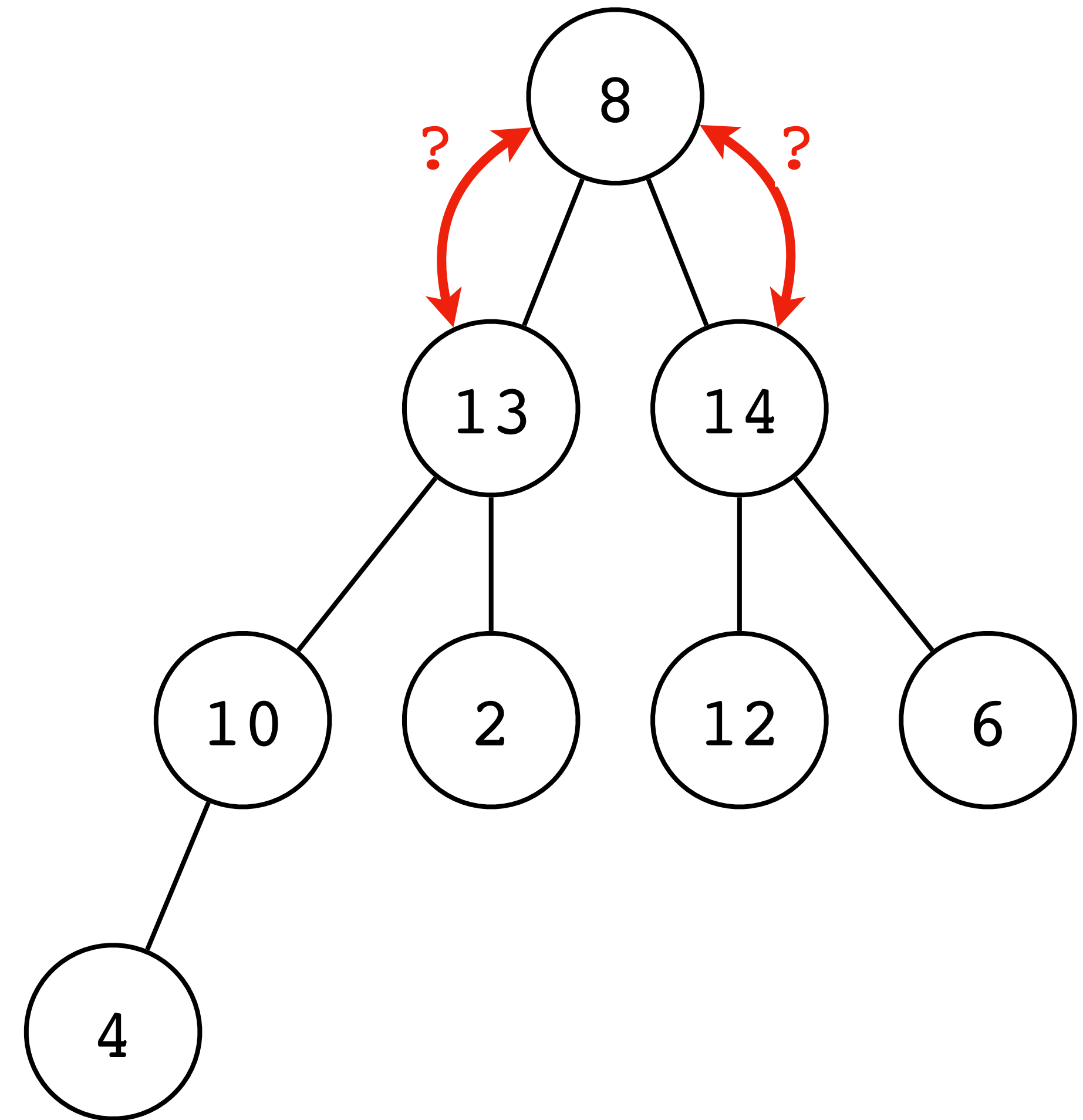
Étape 1 : Permutation premier/dernier

Étape 2 : Suppression du dernier

***Problème** : Condition de tas perdue*

Étape 3 : Rétablissement de la condition
en descendant la racine
du côté de plus grand enfant.

*Pourquoi ? Parce que si c'est le plus petit qui devient parent,
le second enfant sera plus grand que son parent.*



Suppression du sommet



Suppression compliquée en sommet...
...plus simple en fin :

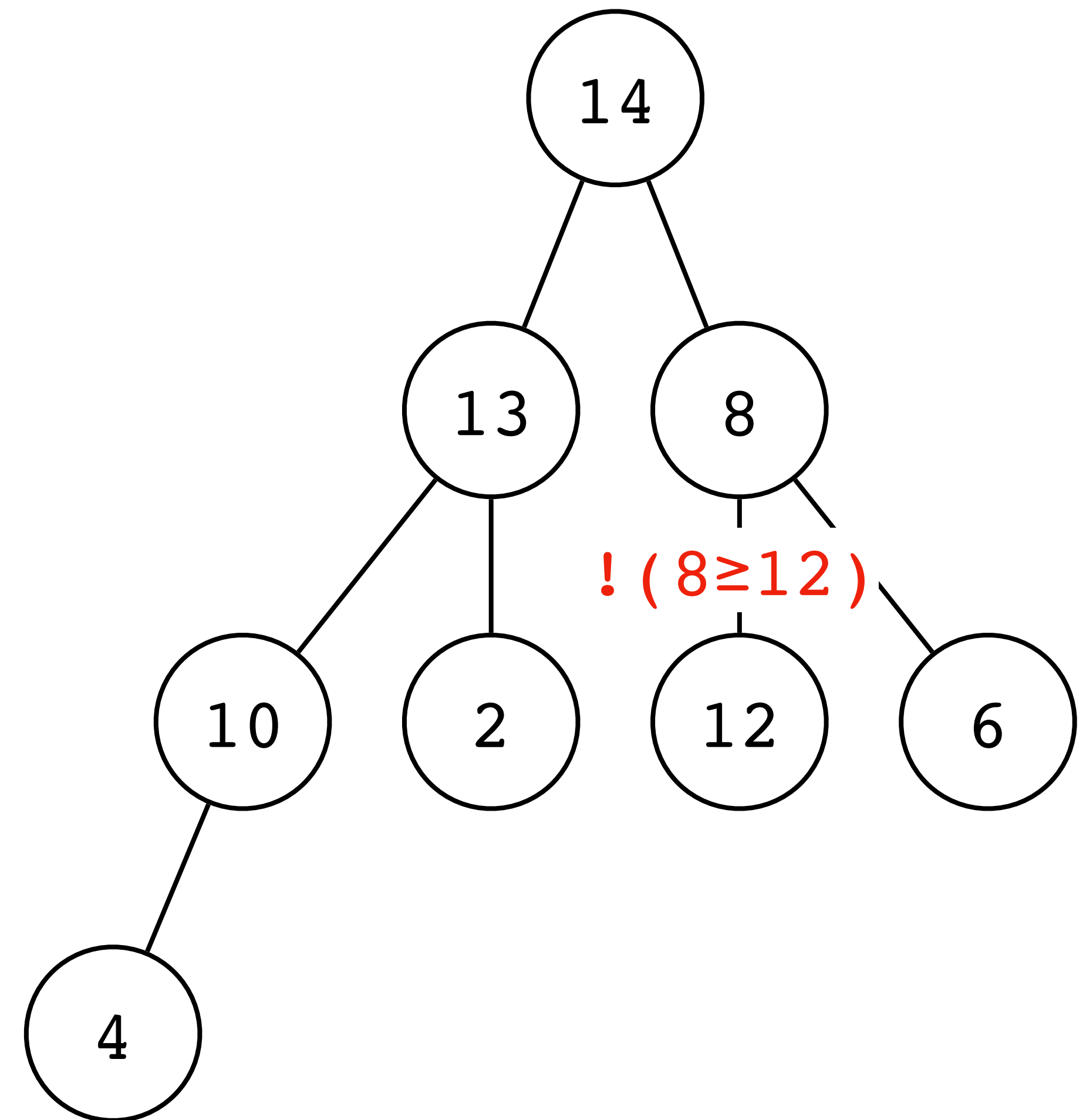
Étape 1 : Permutation premier/dernier

Étape 2 : Suppression du dernier

Problème : *Condition de tas perdue*

Étape 3 : Rétablissement de la condition
en descendant la racine
du côté de plus grand enfant.

*Pourquoi ? Parce que si c'est le plus petit qui devient parent,
le second enfant sera plus grand que son parent.*



Suppression du sommet



Suppression compliquée en sommet...
...plus simple en fin :

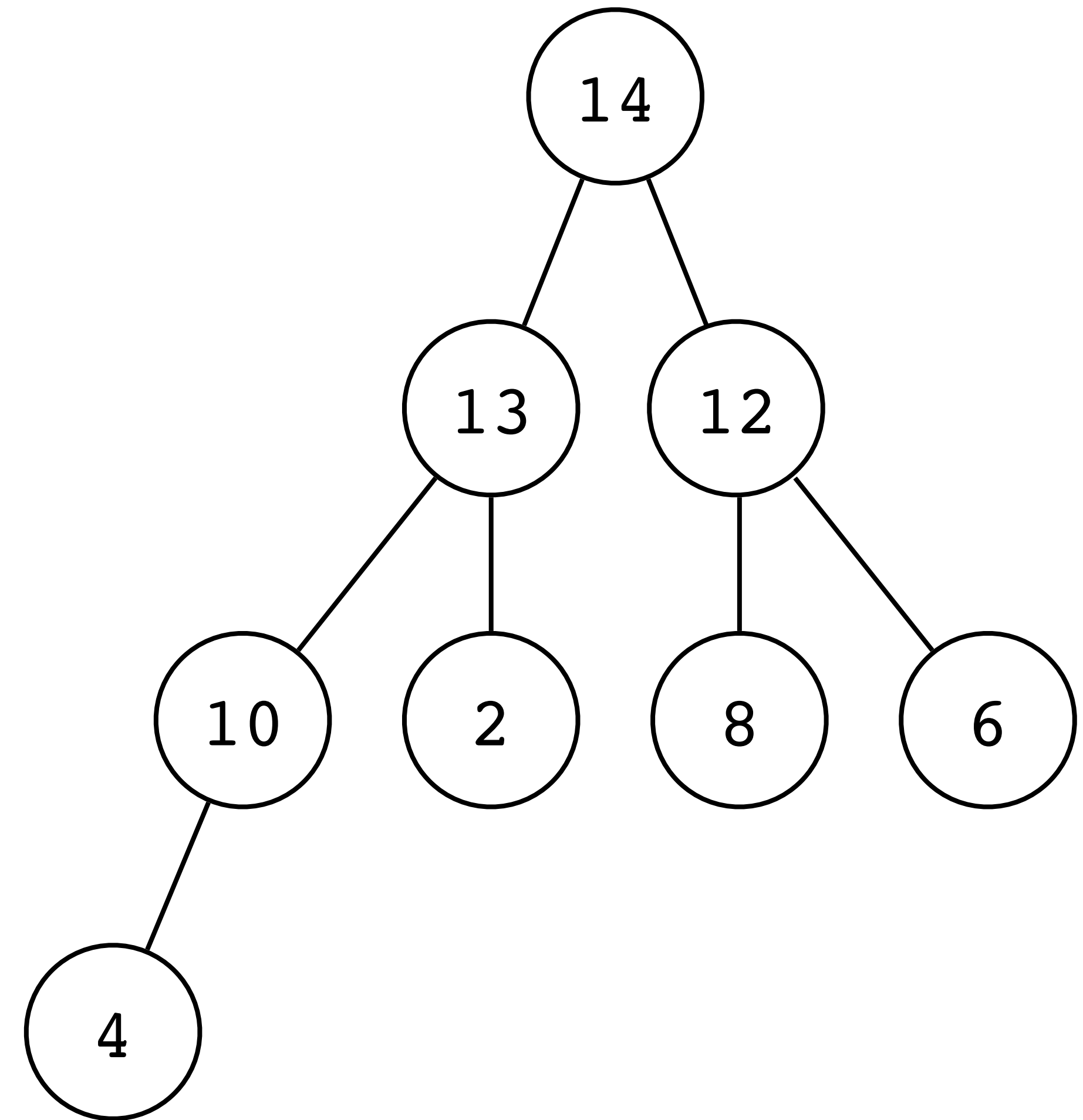
Étape 1 : Permutation premier/dernier

Étape 2 : Suppression du dernier

***Problème** : Condition de tas perdue*

Étape 3 : Rétablissement de la condition
en descendant la racine
du côté de plus grand enfant.

*Pourquoi ? Parce que si c'est le plus petit qui devient parent,
le second enfant sera plus grand que son parent.*

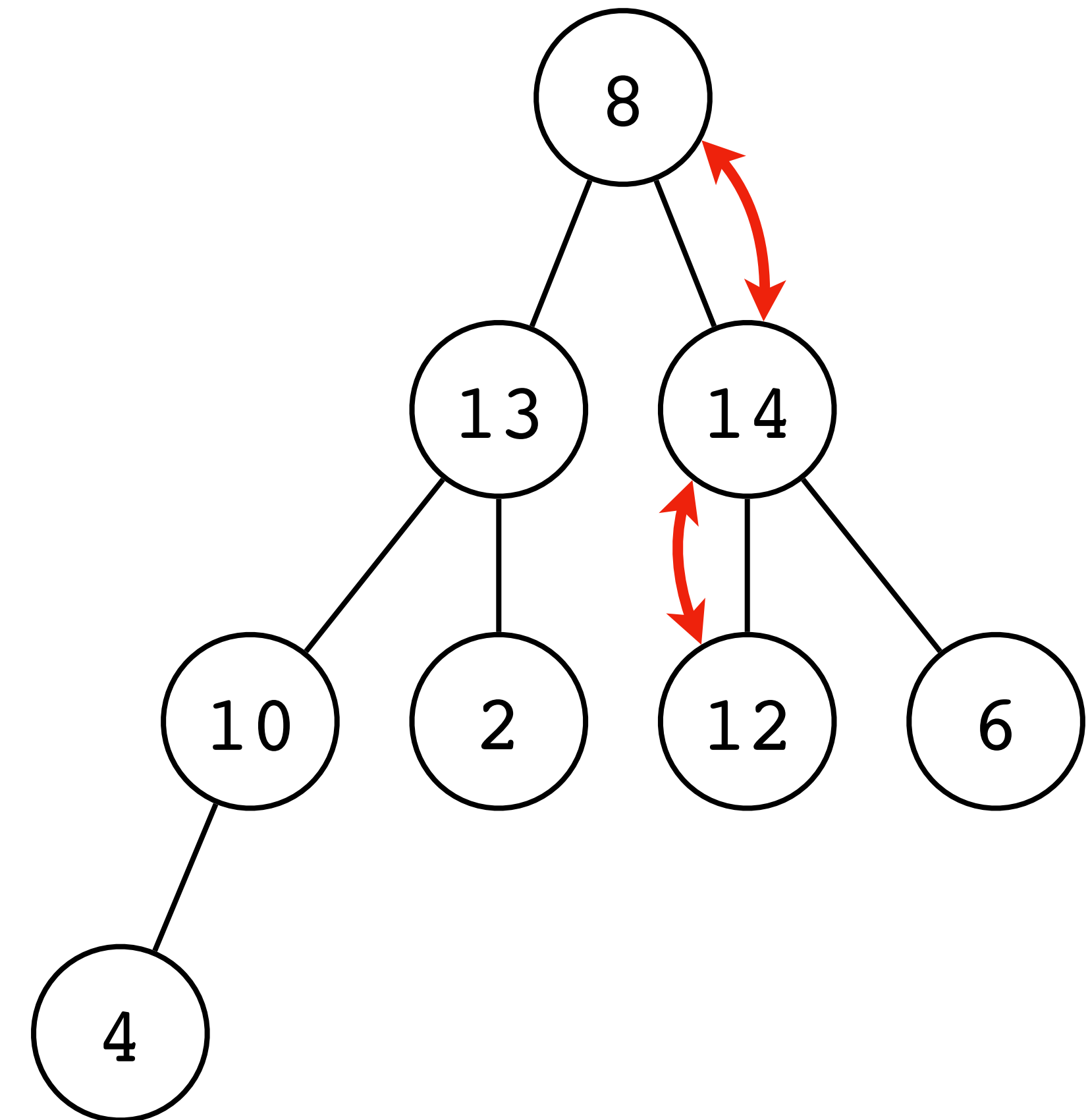


Descendre (sink)



```
fonction pge(T,i)      (plus grand enfant)  
    si e2(i) < taille(T) et T[e2(i)] > T[e1(i)]  
        retourner e2(i)  
    sinon, retourner e1(i)
```

```
fonction descendre(T,i)  
    tant que e1(i) < taille(T) et T[pge(T,i)] > T[i]  
        échanger T[pge(T,i)], T[i]  
        i ← pge(T,i)
```





15. Tas : make & sort

Créer un tas



- Comment ré-organiser en tas les éléments d'un tableau ?
- Approche naïve : insérer les éléments un à un

```
fonction créer_tas(T)  
  pour i de 1 à taille(T)-1  
    remonter(T, i)
```

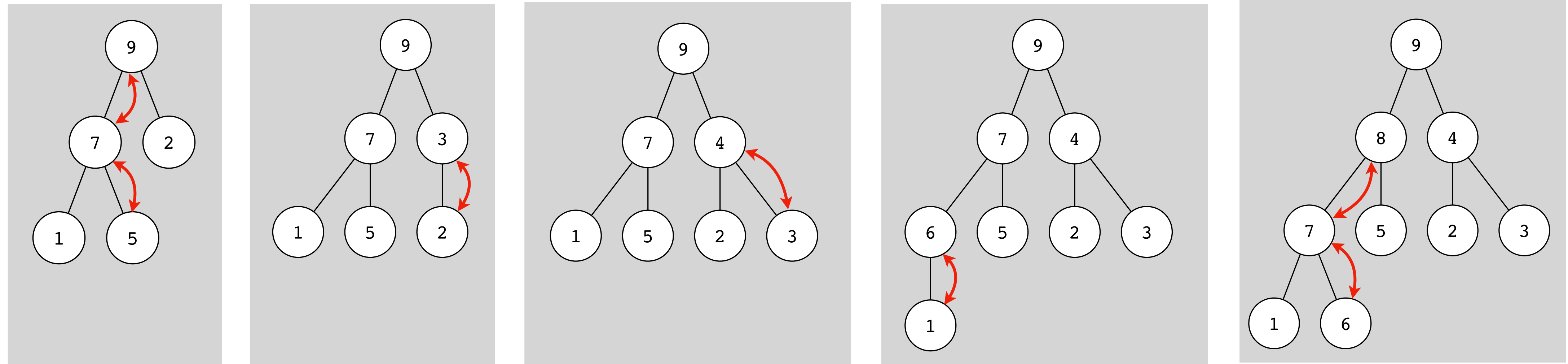
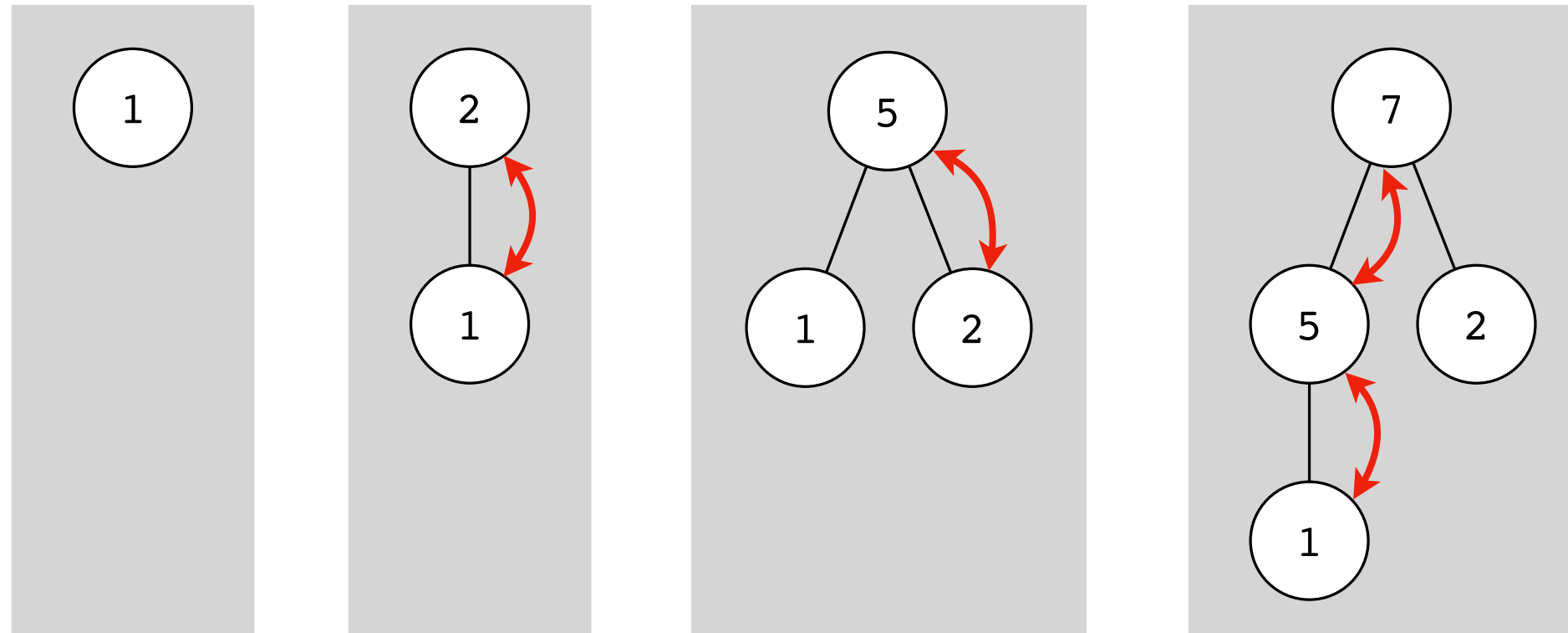
- Complexité : $O(n \log(n))$

Illustration approche naïve



[1, 2, 5, 7, 9, 3, 4, 6, 8]

[1, 2, 5, 7, 9, 3, 4, 6, 8]	2 ↔ 1
[2, 1, 5, 7, 9, 3, 4, 6, 8]	5 ↔ 2
[5, 1, 2, 7, 9, 3, 4, 6, 8]	7 ↔ 1, 7 ↔ 5
[7, 5, 2, 1, 9, 3, 4, 6, 8]	9 ↔ 5, 9 ↔ 7
[9, 7, 2, 1, 5, 3, 4, 6, 8]	3 ↔ 2
[9, 7, 3, 1, 5, 2, 4, 6, 8]	4 ↔ 3
[9, 7, 4, 1, 5, 2, 3, 6, 8]	6 ↔ 1
[9, 7, 4, 6, 5, 2, 3, 1, 8]	8 ↔ 6, 8 ↔ 7
[9, 8, 4, 7, 5, 2, 3, 1, 6]	

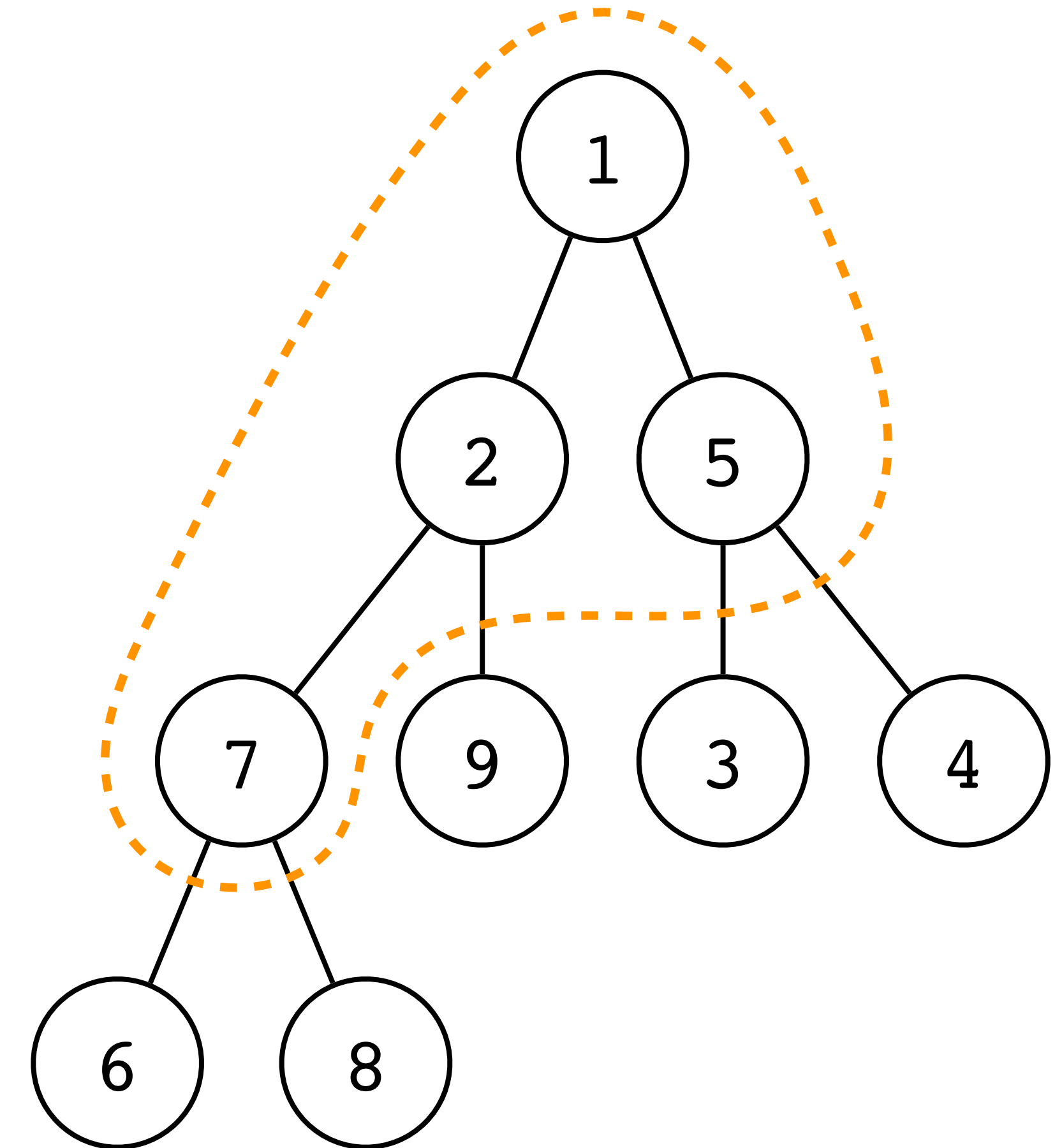


[1, 2, 5, 7, 9, 3, 4, 6, 8] : forme un tas

Créer un tas - algorithme en $O(n)$



Parcourir tous les noeuds ayant un enfant,
du dernier au premier.



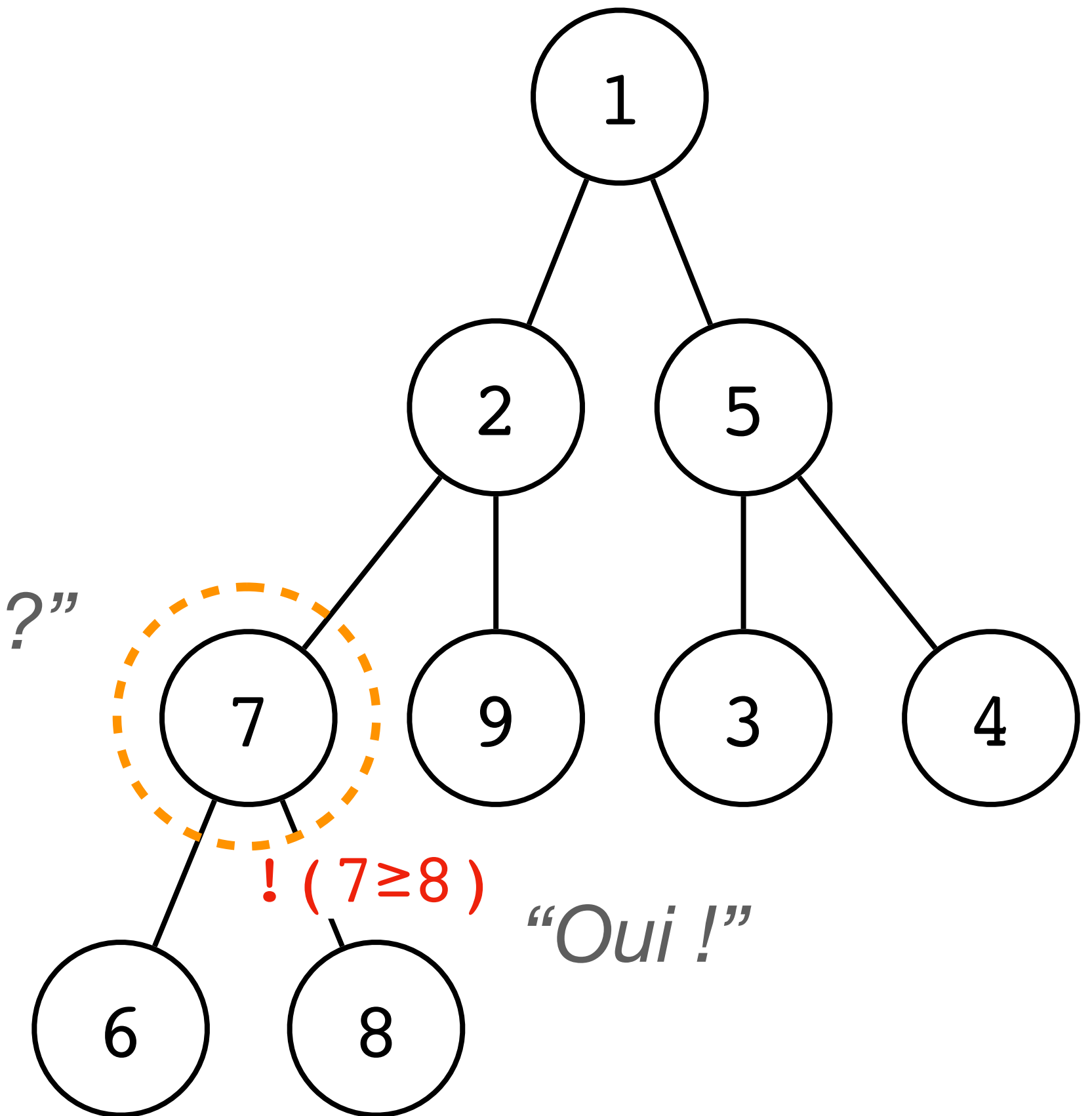
Créer un tas - algorithme en $O(n)$



Parcourir tous les noeuds ayant un enfant,
du dernier au premier.

Pour chaque élément,
le descendre si nécessaire.

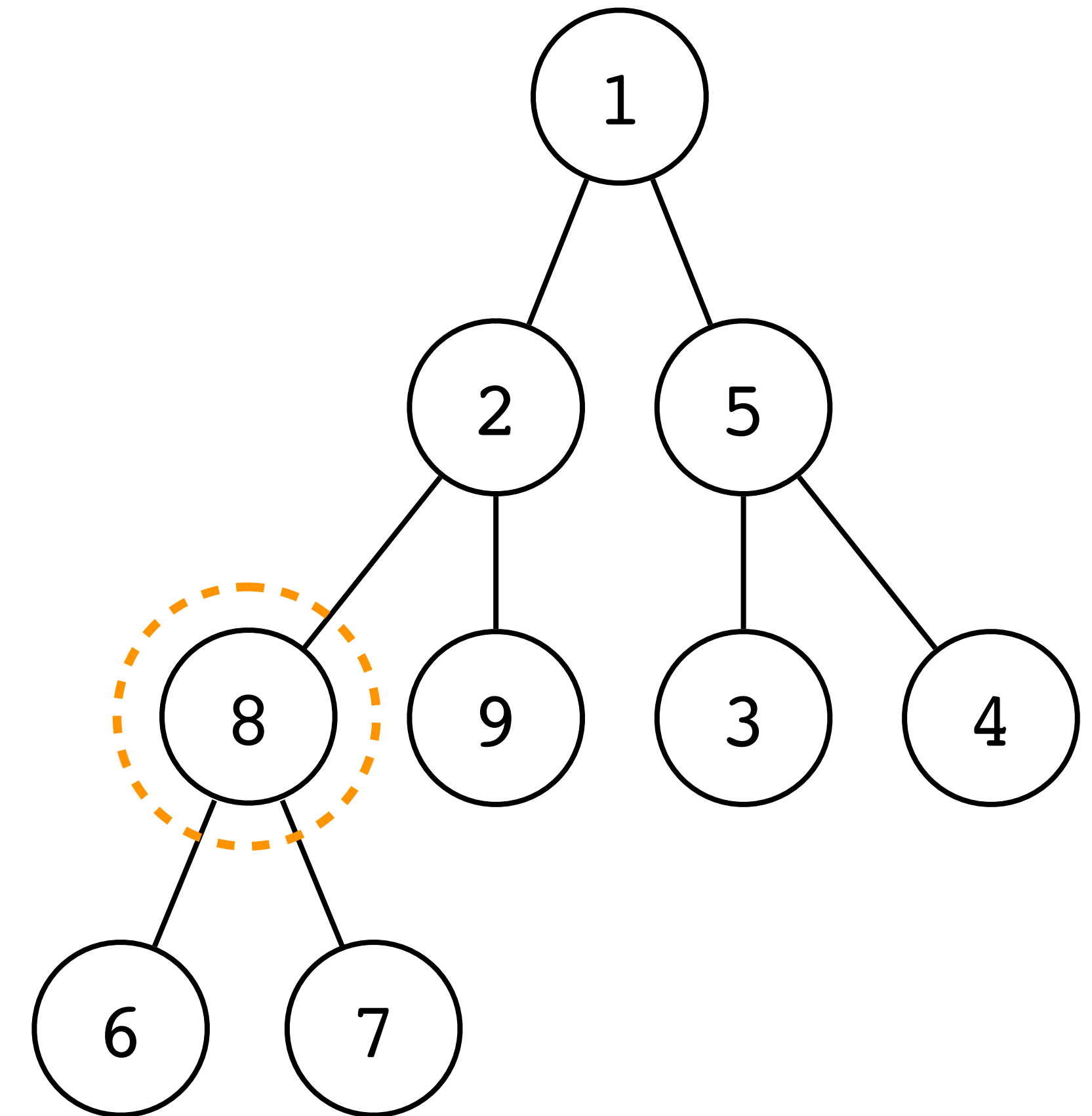
“doit descendre ?”



Créer un tas - algorithme en $O(n)$ 

Parcourir tous les noeuds ayant un enfant,
du dernier au premier.

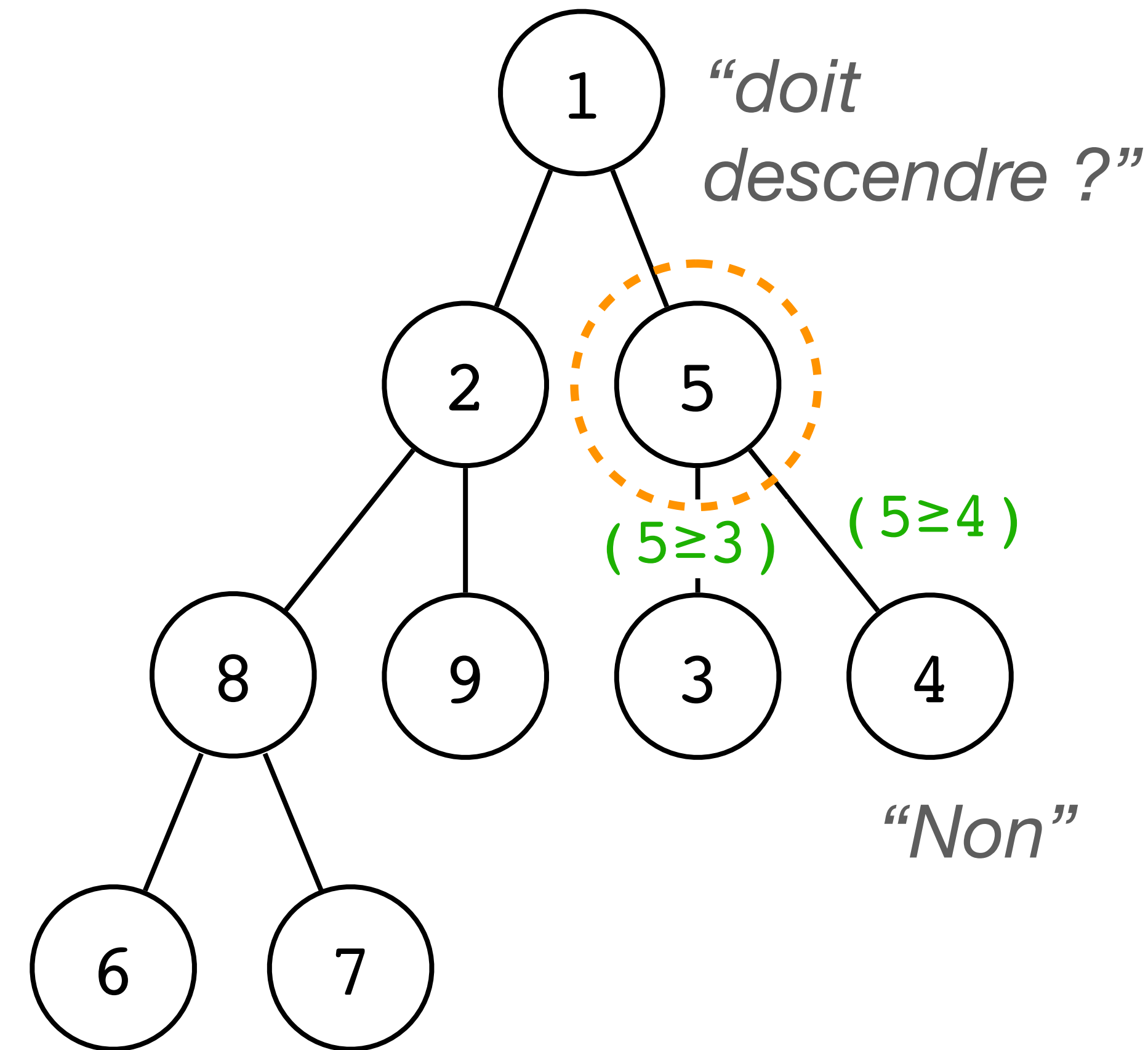
Pour chaque élément,
le descendre si nécessaire.



Créer un tas - algorithme en $O(n)$ 

Parcourir tous les noeuds ayant un enfant,
du dernier au premier.

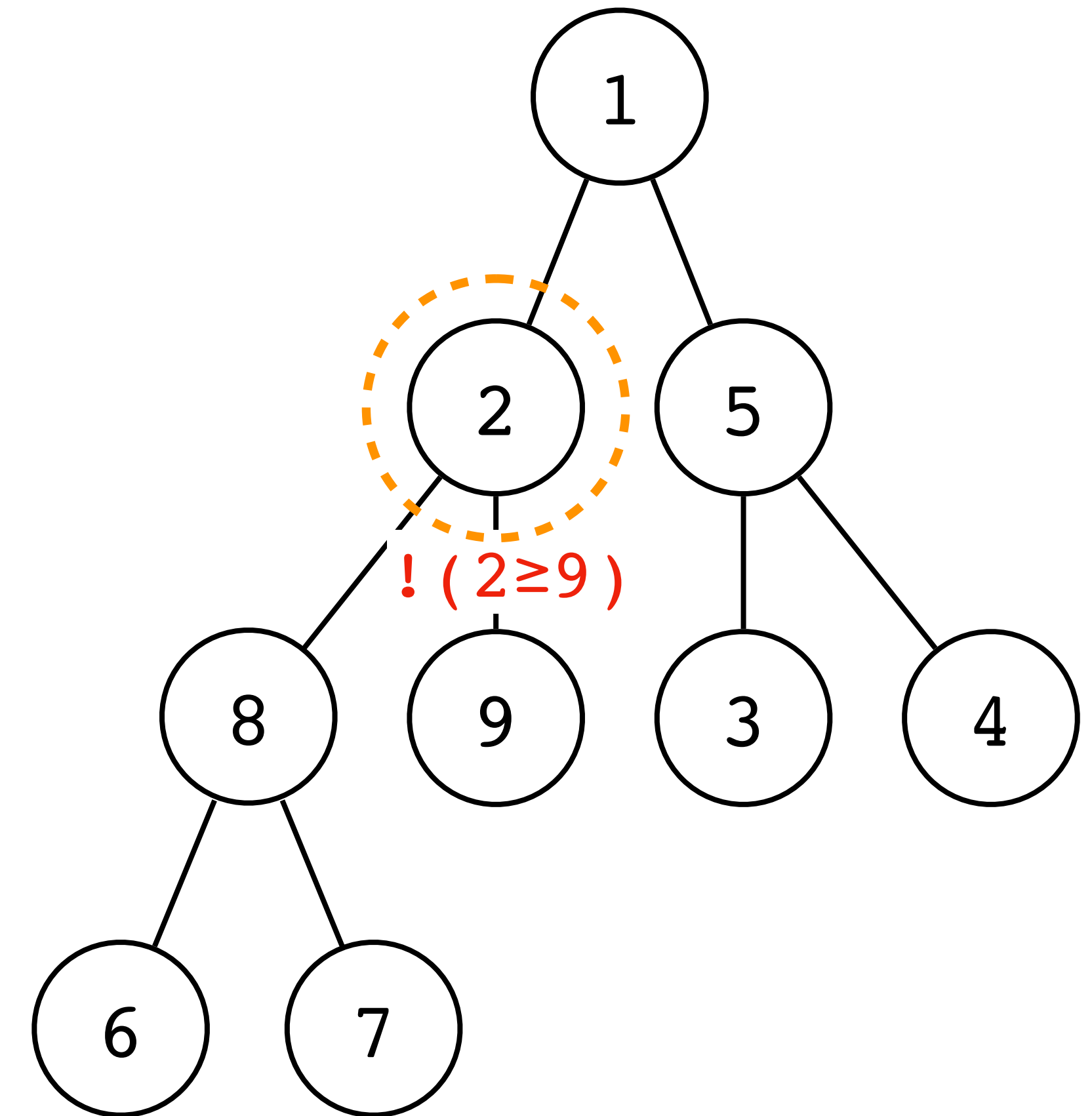
Pour chaque élément,
le descendre si nécessaire.



Créer un tas - algorithme en $O(n)$ 

Parcourir tous les noeuds ayant un enfant,
du dernier au premier.

Pour chaque élément,
le descendre si nécessaire.

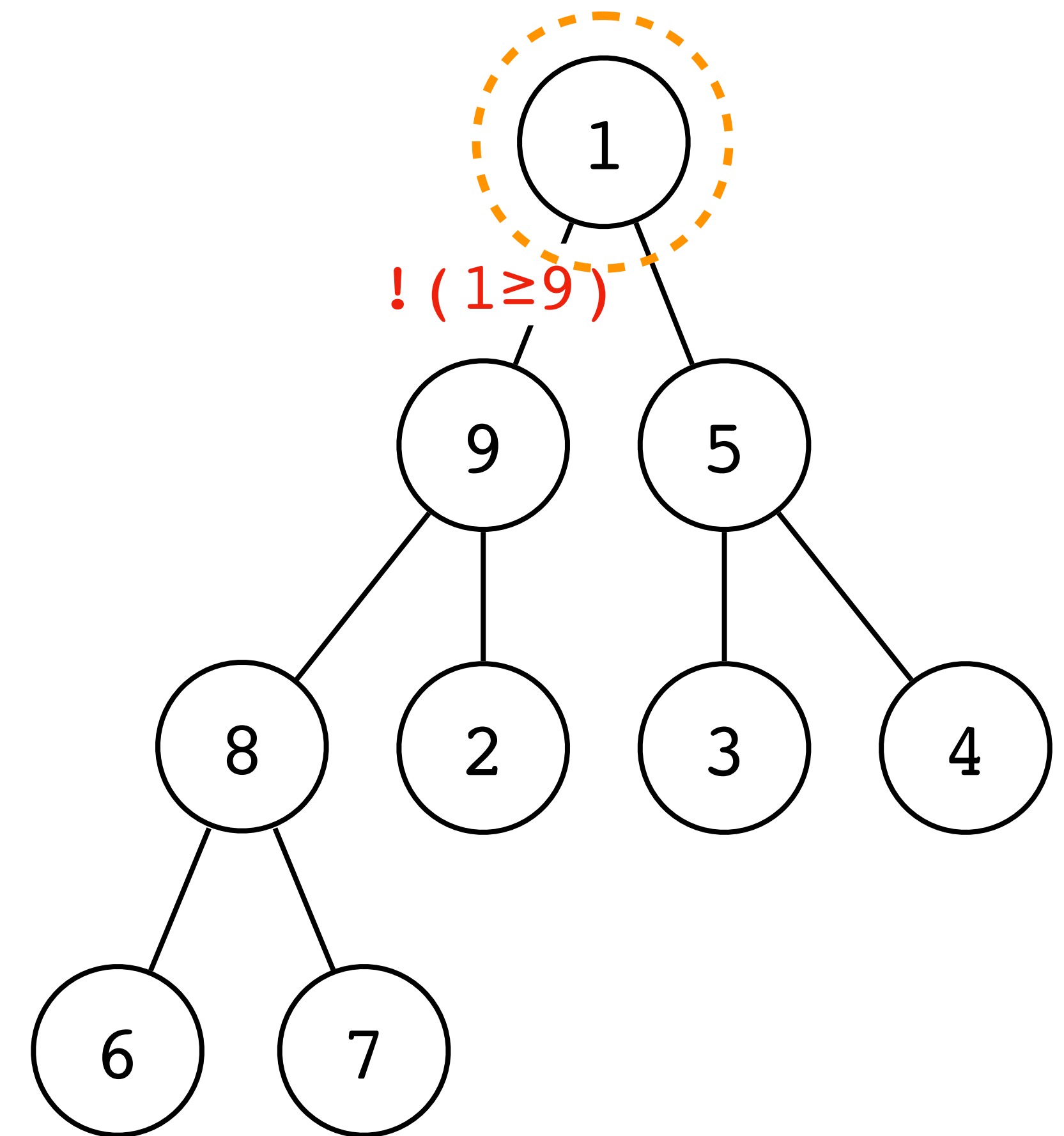


Créer un tas - algorithme en $O(n)$



Parcourir tous les noeuds ayant un enfant,
du dernier au premier.

Pour chaque élément,
le descendre si nécessaire.



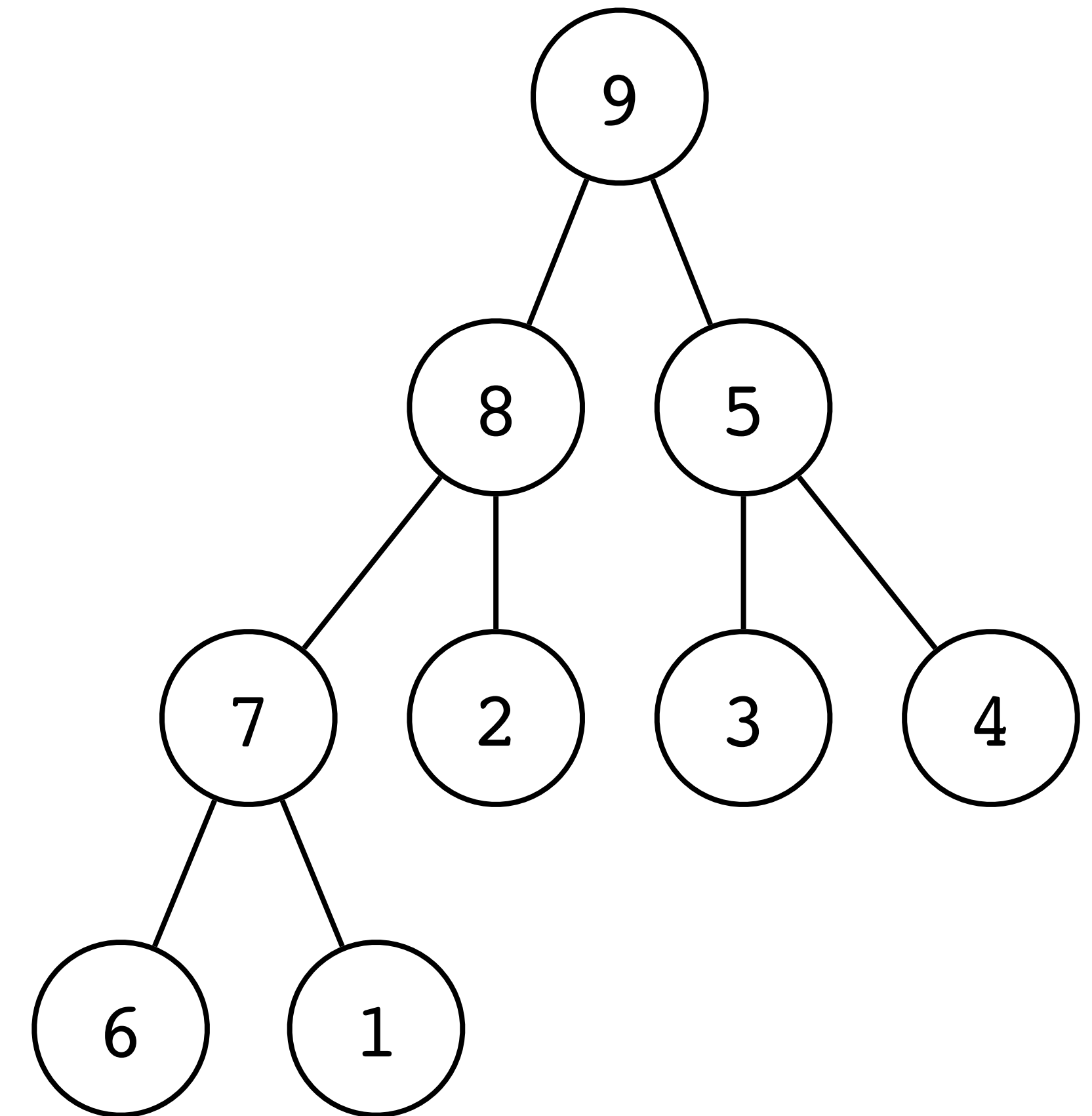
Créer un tas - algorithme en $O(n)$ 

Parcourir tous les noeuds ayant un enfant,
du dernier au premier.

Pour chaque élément,
le descendre si nécessaire.

*Premier noeud
ayant un enfant.*

```
fonction créer_tas(T)
  p ← parent(taille(T)-1)
  pour i décroissant de p à 0
    descendre(T, i)
```



Créer un tas - Complexité



```

fonction créer_tas(T)
  p ← parent(taille(T)-1)
  pour i décroissant de p à 0
    descendre(T, i)
  
```

$$O\left(\log_2(n/2) \cdot 1 + \log_2(n/4) \cdot 2 + \dots + 3 \cdot \frac{n}{16} + 2 \cdot \frac{n}{8} + 1 \cdot \frac{n}{4}\right)$$

$$= O\left(\sum_{k=1}^{\log_2(n/2)} k \cdot \frac{n}{2^{k+1}}\right) = O(n) \quad \text{Car } \sum_{k=1}^{\infty} \frac{k}{2^k} = 2$$

Descente Noeuds

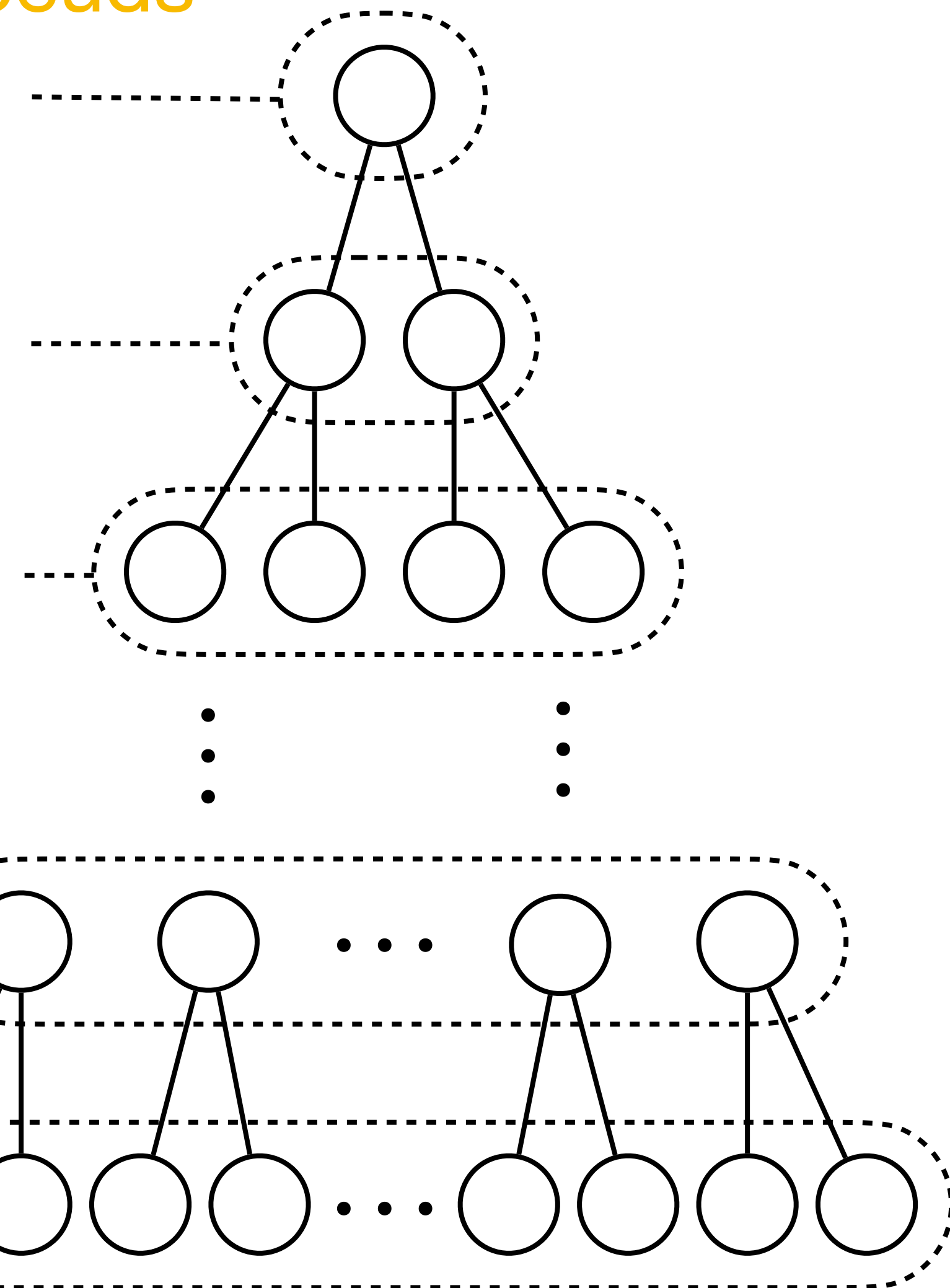
$$O\left(\log_2(n/2) \cdot 1\right)$$

$$O\left(\log_2(n/4) \cdot 2\right)$$

$$O\left(\log_2(n/8) \cdot 4\right)$$

$$O\left(1 \cdot \frac{n}{4}\right)$$

$$O\left(0 \cdot \frac{n}{2}\right)$$



Tri par tas



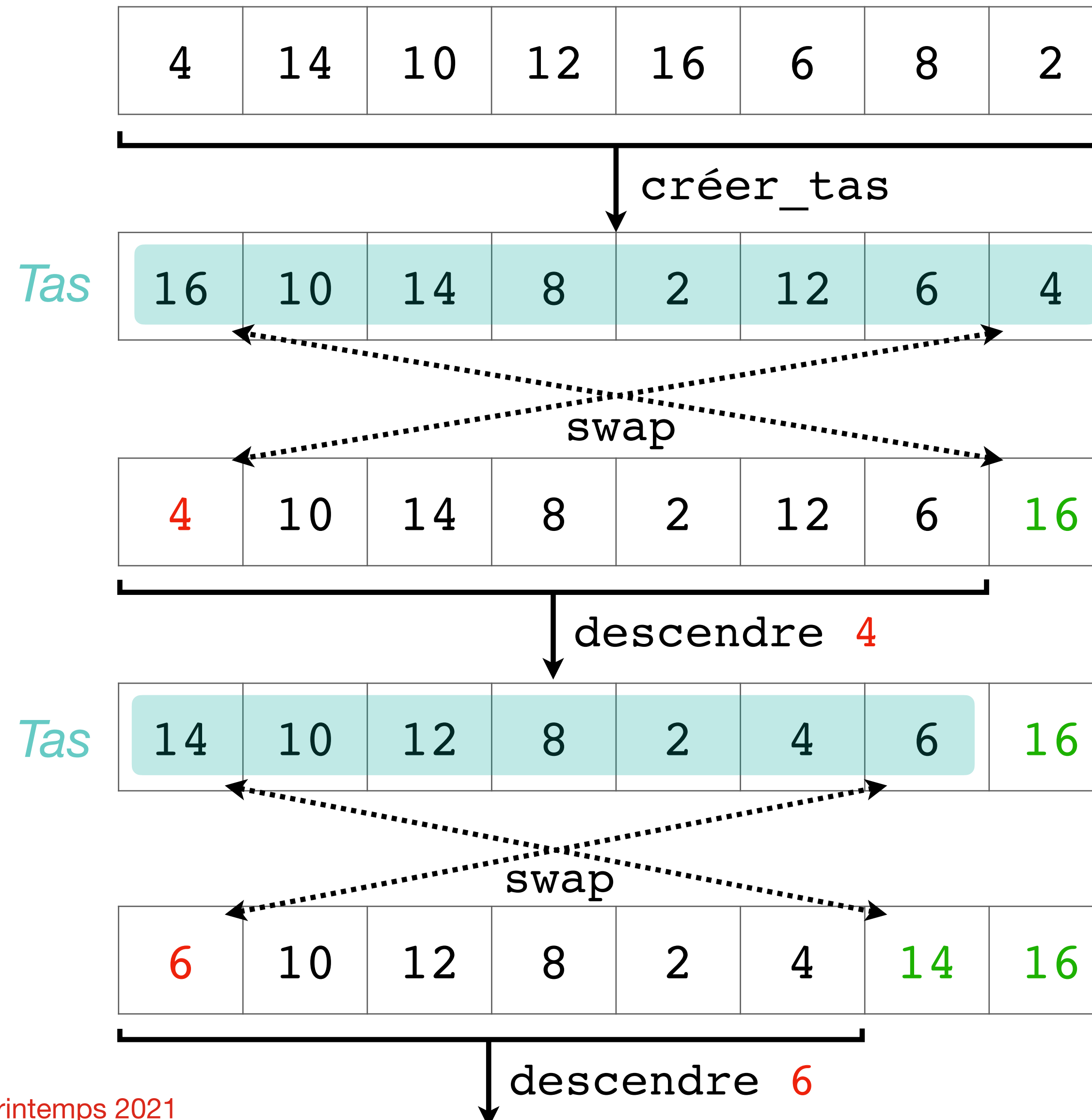
Peut-on trier un tableau à l'aide des tas ?

Un tas un est tableau partiellement trié...
...mais que très partiellement.

Par contre, un tas stocke le max en racine !

Peut donc être utilisé pour un genre de tri par sélection.

Tri par tas



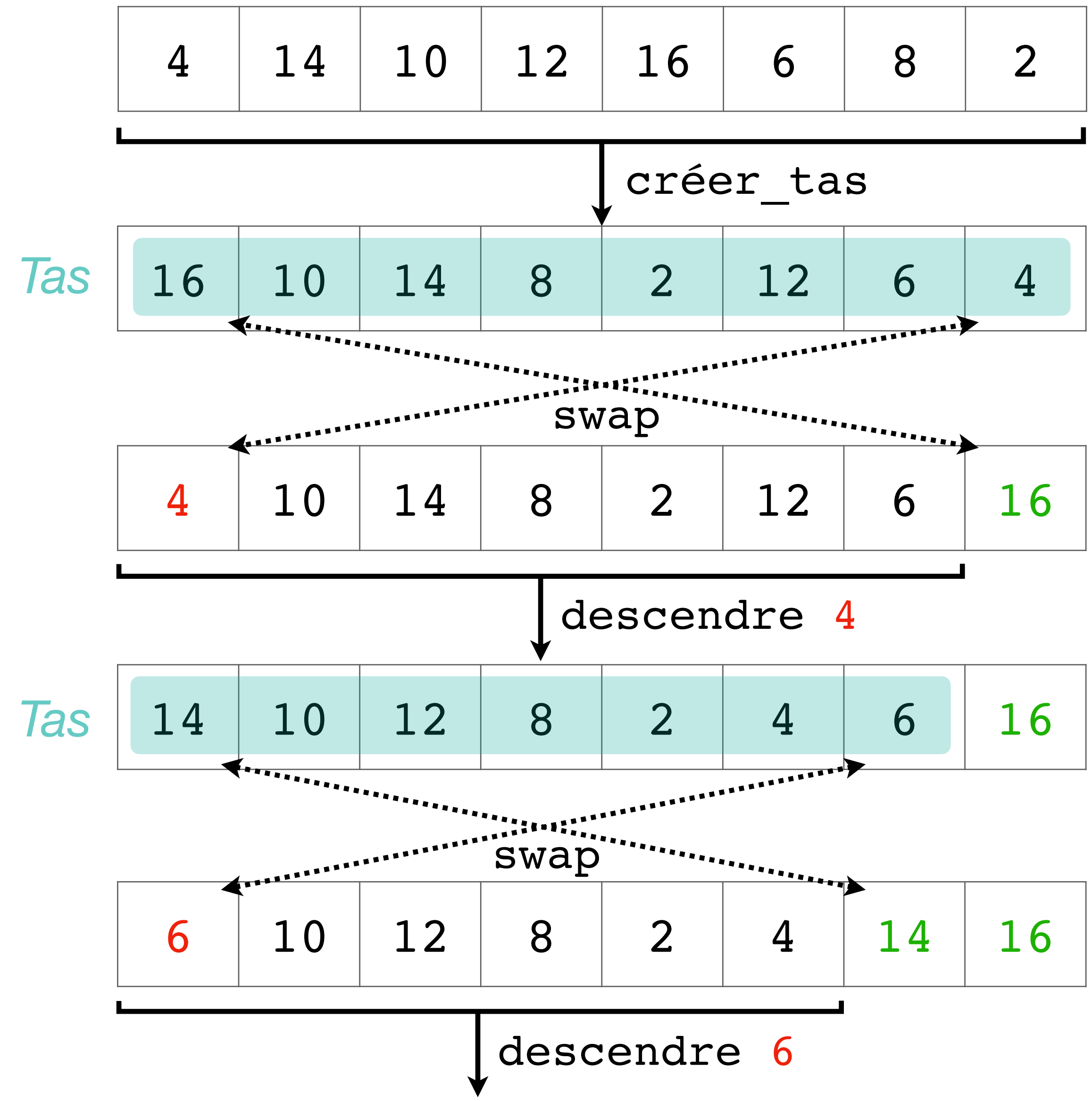
Tri par tas



- Tri par sélection du maximum en maintenant la partie gauche en tas
 - initialement avec créer_tas
 - puis en faisant descendre la racine

Complexité

- créer_tas en $O(n)$
- descendre en $O(\log_2(n))$
 - effectué $O(n)$ fois
- Donc $O(n \log(n))$



Tri par tas

- Tri par sélection du maximum en maintenant la partie gauche en tas
 - initialement avec créer_tas
 - puis en faisant descendre la racine

Complexité

- créer_tas en $O(n)$
- descendre en $O(\log_2(n))$
 - effectué $O(n)$ fois
- Donc $O(n \log(n))$



```
fonction tri_par_tas(T):  
    N ← taille(T)  
    créer_tas(T)  
    pour k décroissant de N-1 à 1  
        échanger T[k], T[0]  
        descendre(T, 0, k)
```

```
fonction descendre(T, i, k)  
    tant que e1(i) < k  
        et T[pge(T, i, k)] > T[i]  
        échanger T[pge(T, i, k)], T[i]  
        i ← pge(T, i, k)
```

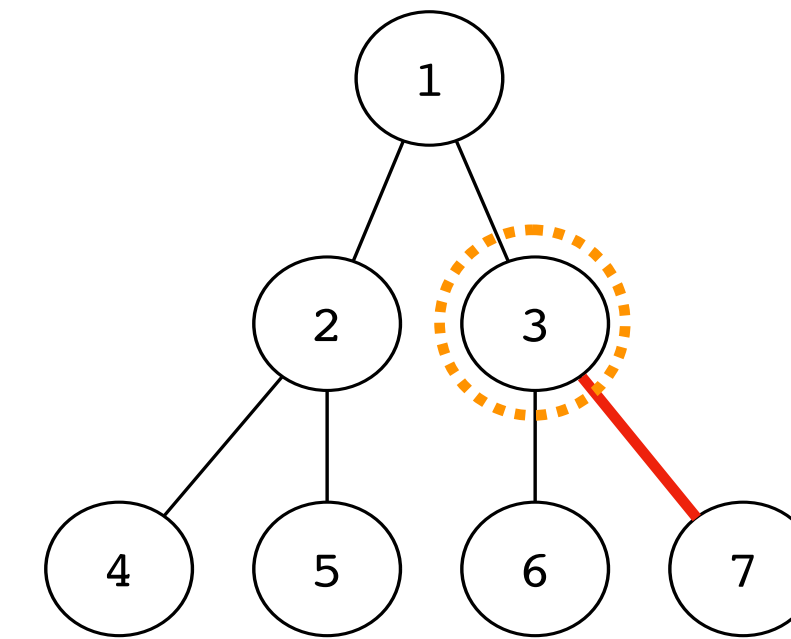
```
fonction pge(T, i, k)  
    si e2(i) < k et T[e2(i)] > T[e1(i)]  
        retourner e2(i)  
    sinon, retourner e1(i)
```

Exemple

Créer le tas

[1, 2, 3, 4, 5, 6, 7]

Echanges



Exemple

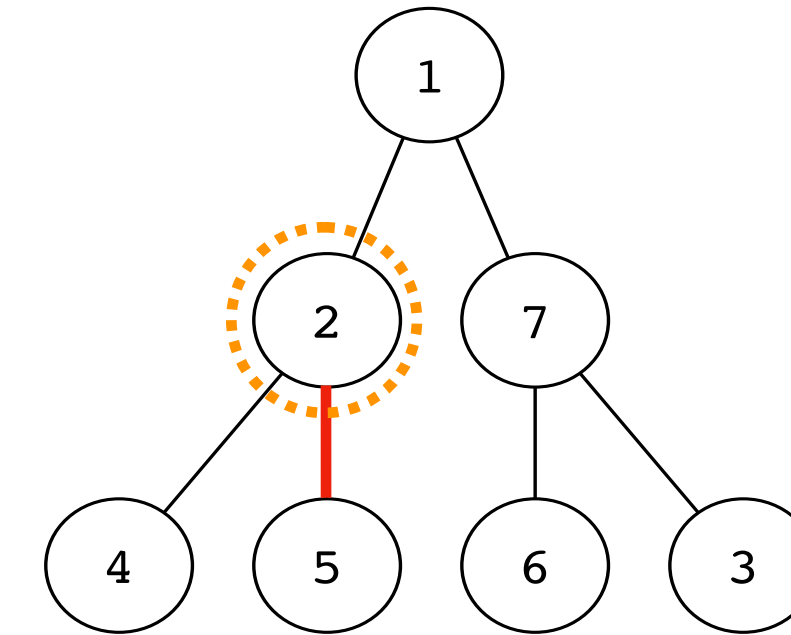
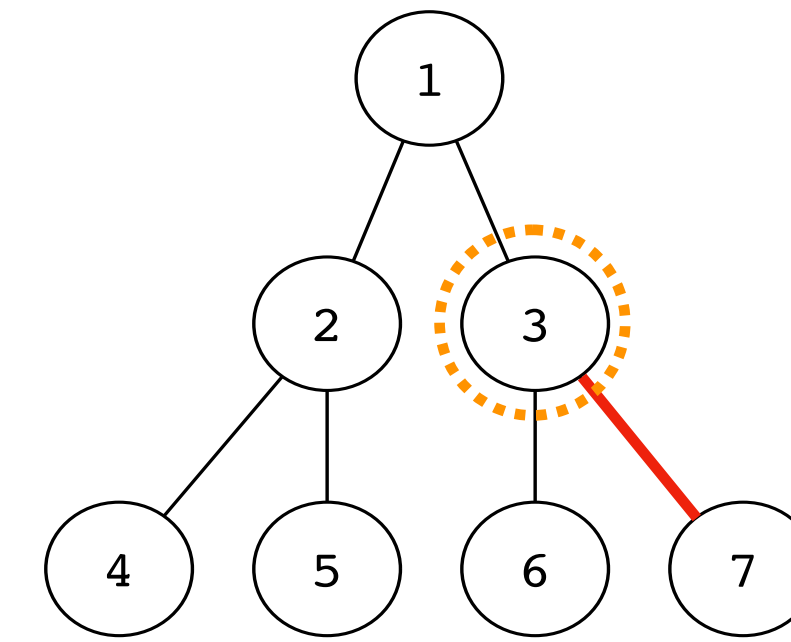
Créer le tas

[1, 2, 3, 4, 5, 6, 7]

[1, 2, 7, 4, 5, 6, 3]

Echanges

3 ↔ 7



Exemple

Créer le tas

[1, 2, 3, 4, 5, 6, 7]

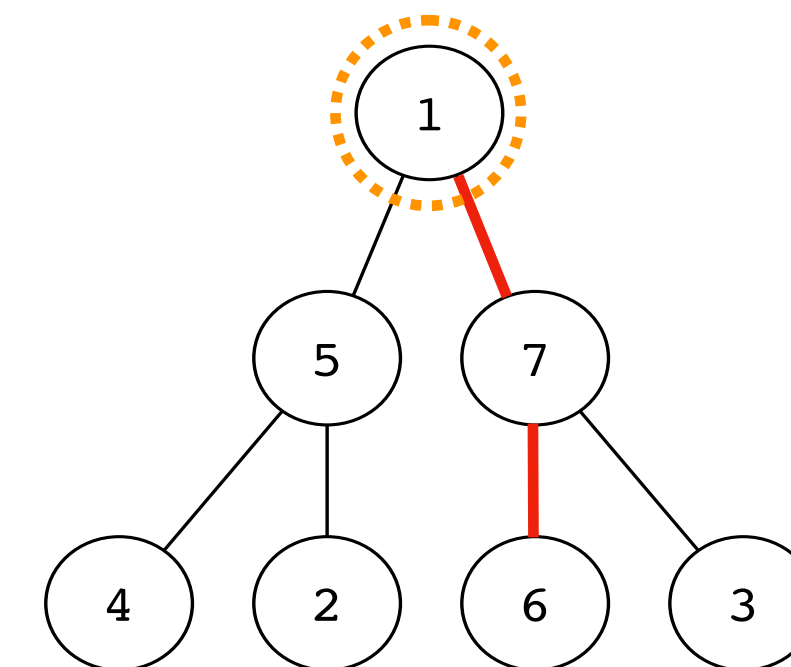
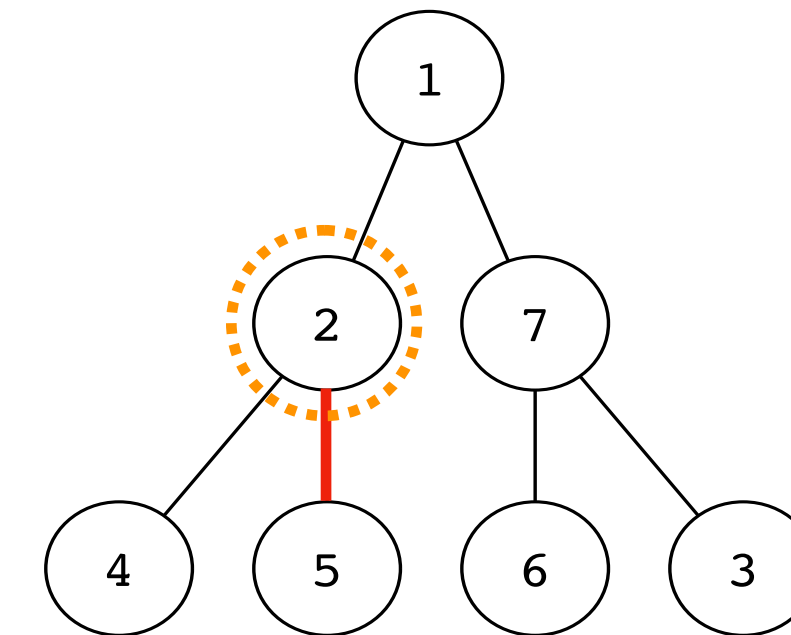
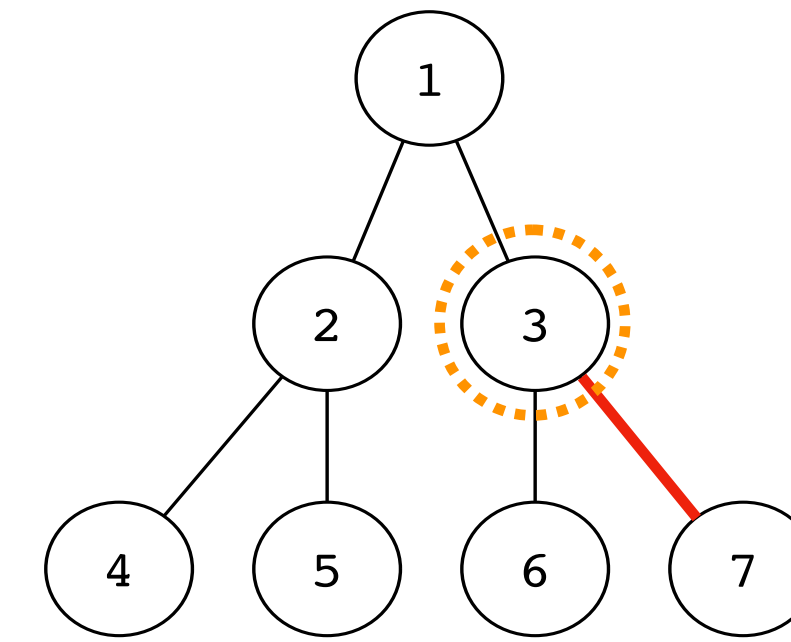
[1, 2, 7, 4, 5, 6, 3]

[1, 5, 7, 4, 2, 6, 3]

Echanges

3 ↔ 7

2 ↔ 5



Exemple

Créer le tas

[1, 2, 3, 4, 5, 6, 7]

[1, 2, 7, 4, 5, 6, 3]

[1, 5, 7, 4, 2, 6, 3]

Tas

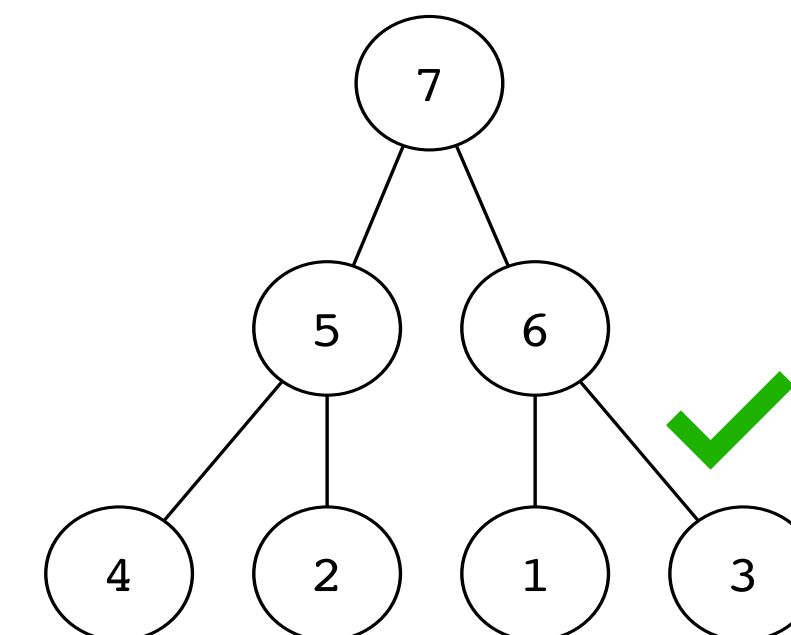
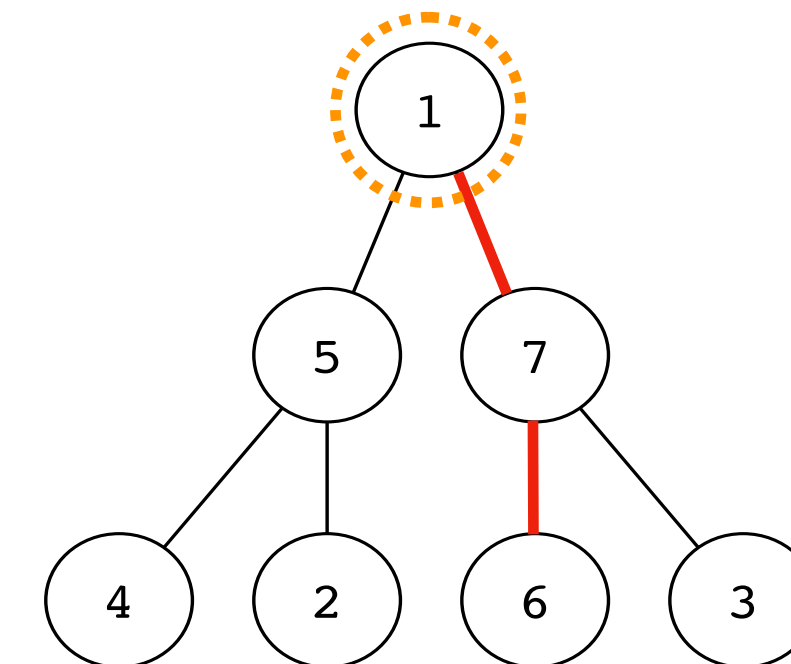
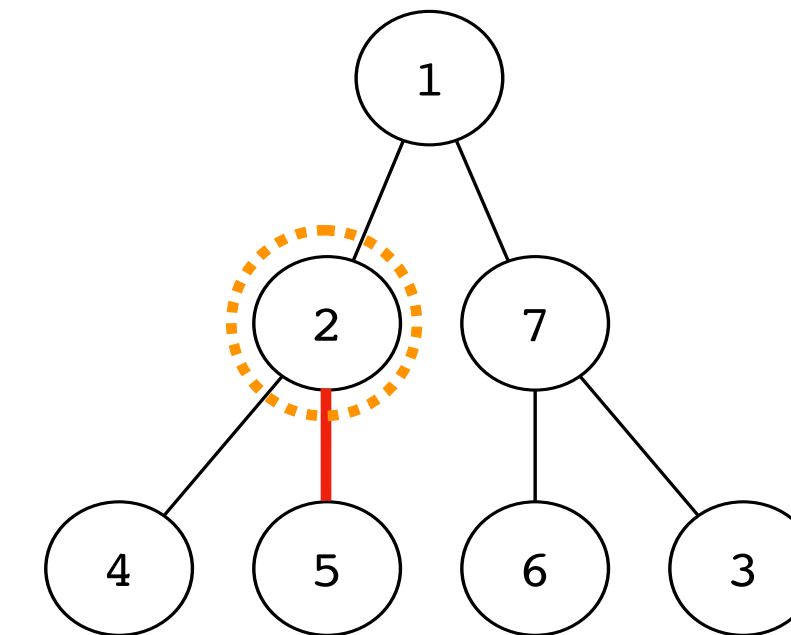
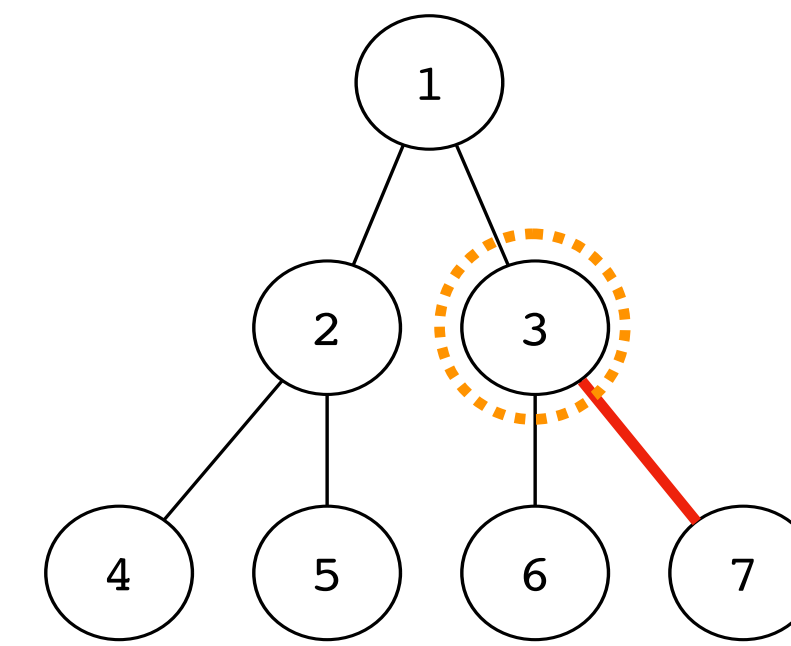
[7, 5, 6, 4, 2, 1, 3]

Echanges

3 ↔ 7

2 ↔ 5

1 ↔ 7, 1 ↔ 6



Exemple

Créer le tas

[1, 2, 3, 4, 5, 6, 7]

[1, 2, 7, 4, 5, 6, 3]

[1, 5, 7, 4, 2, 6, 3]

Tas

[7, 5, 6, 4, 2, 1, 3]

Boucle sur k

[6, 5, 3, 4, 2, 1, 7]

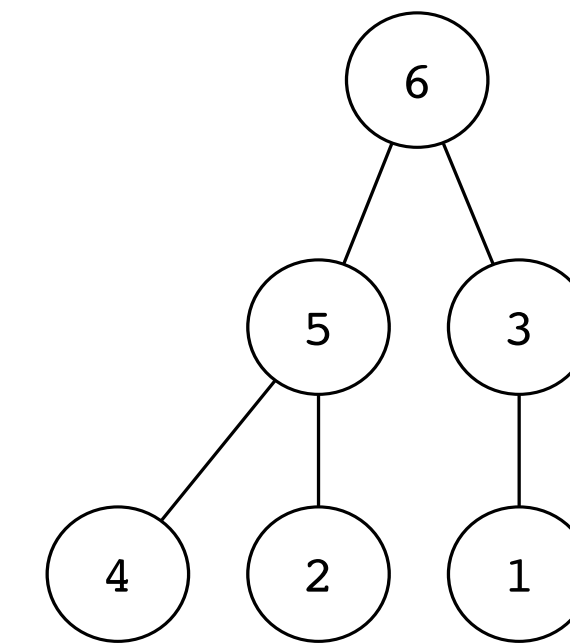
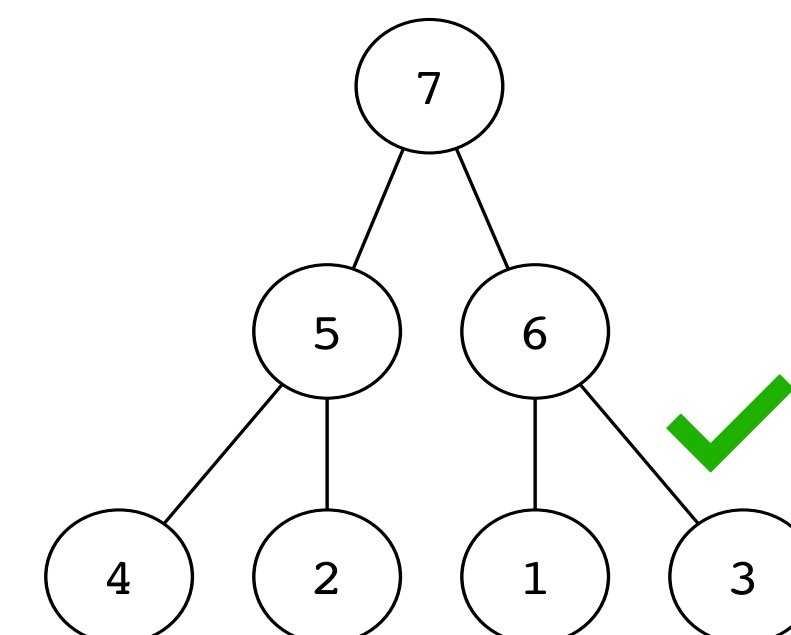
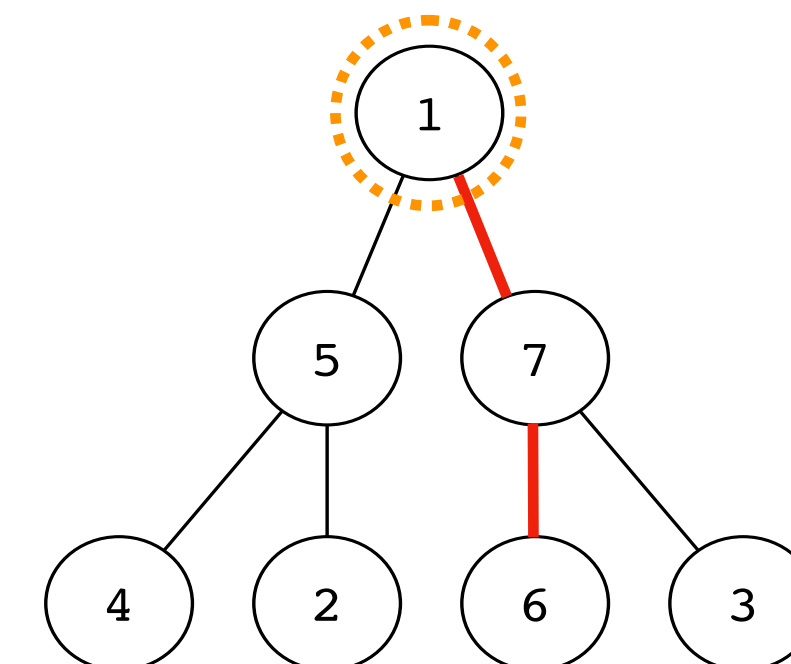
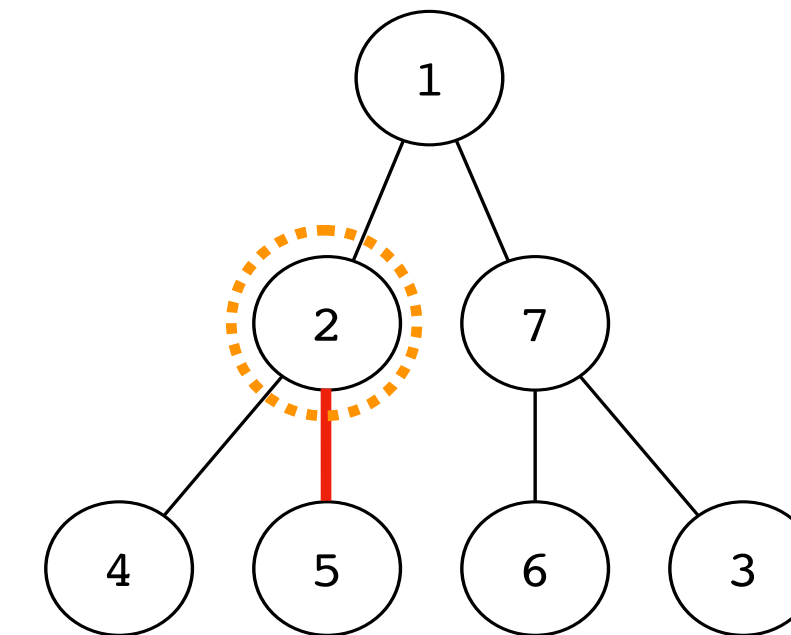
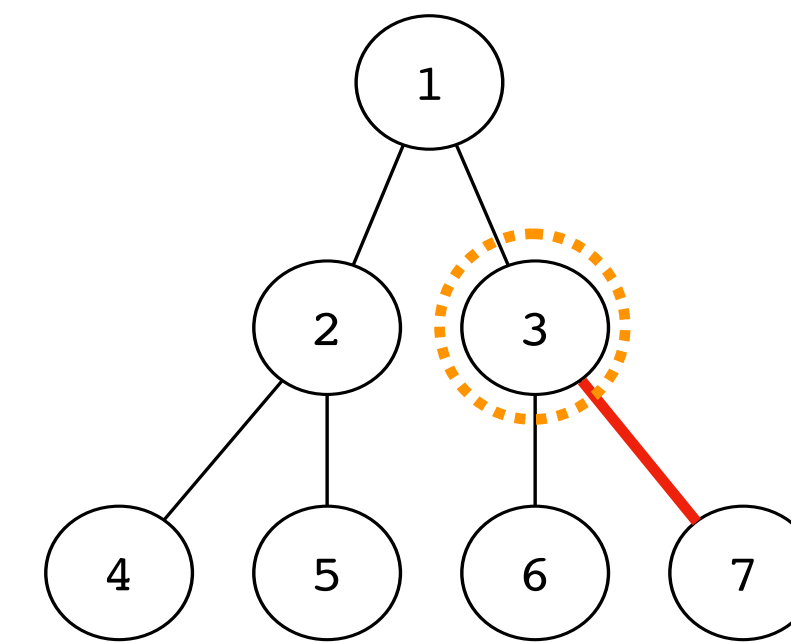
Echanges

3 ↔ 7

2 ↔ 5

1 ↔ 7, 1 ↔ 6

7 ↔ 3, 3 ↔ 6



Exemple

Créer le tas

[1, 2, 3, 4, 5, 6, 7]

[1, 2, 7, 4, 5, 6, 3]

[1, 5, 7, 4, 2, 6, 3]

Tas

[7, 5, 6, 4, 2, 1, 3]

Boucle sur k

[6, 5, 3, 4, 2, 1, 7]

[5, 4, 3, 1, 2, 6, 7]

Echanges

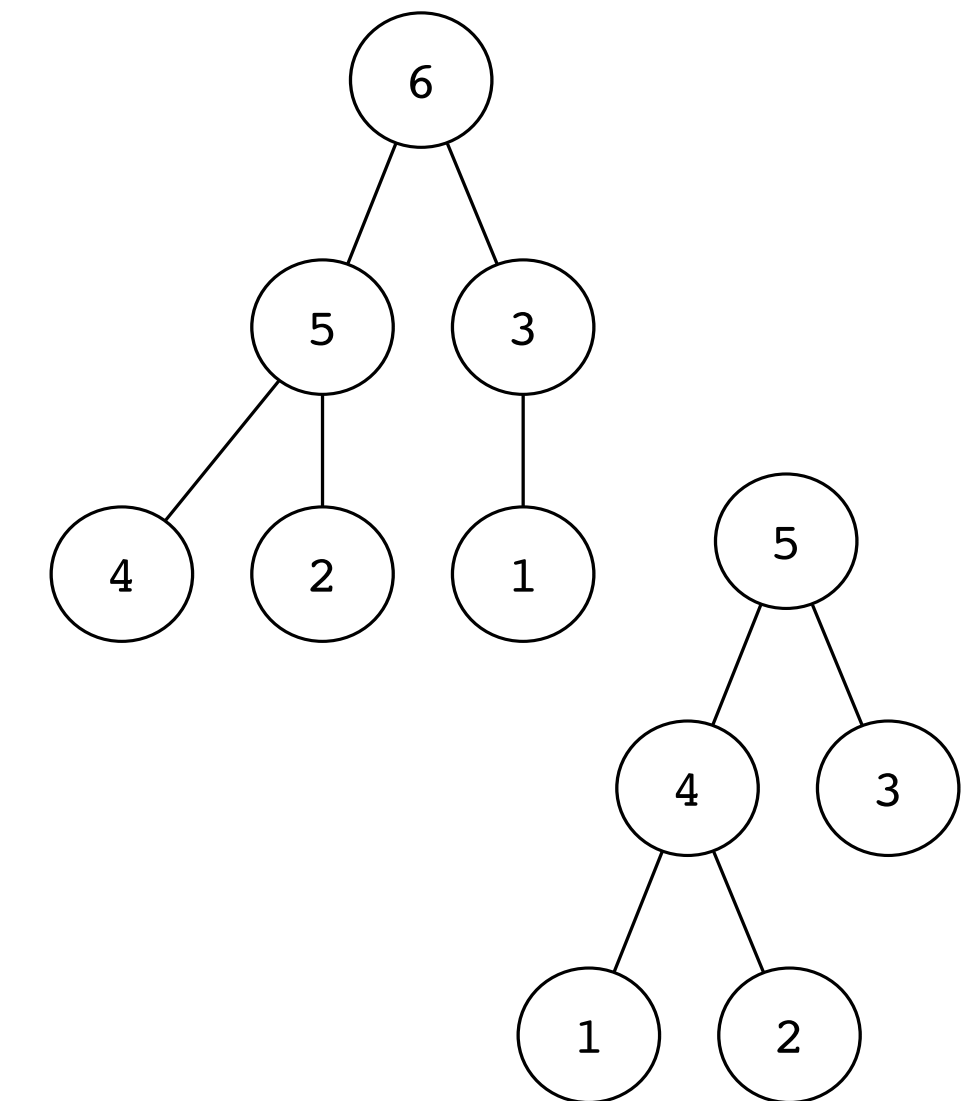
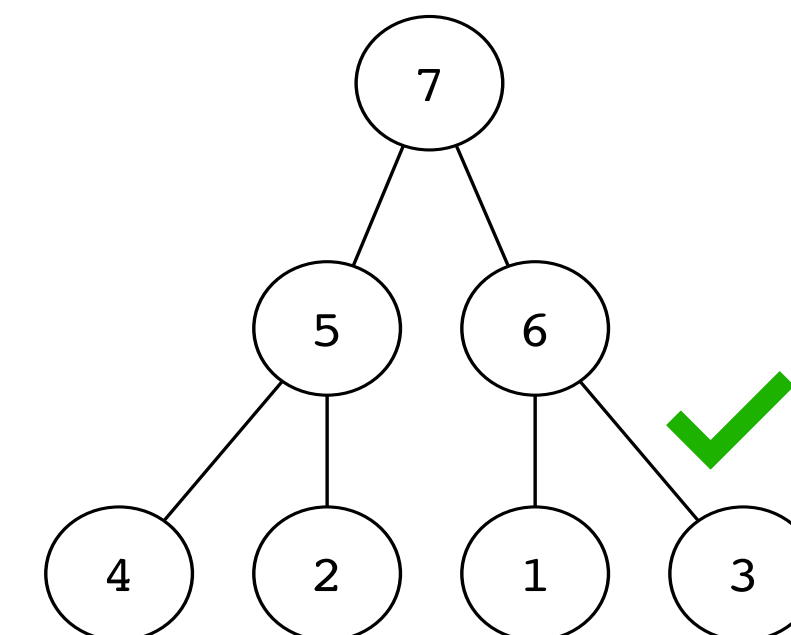
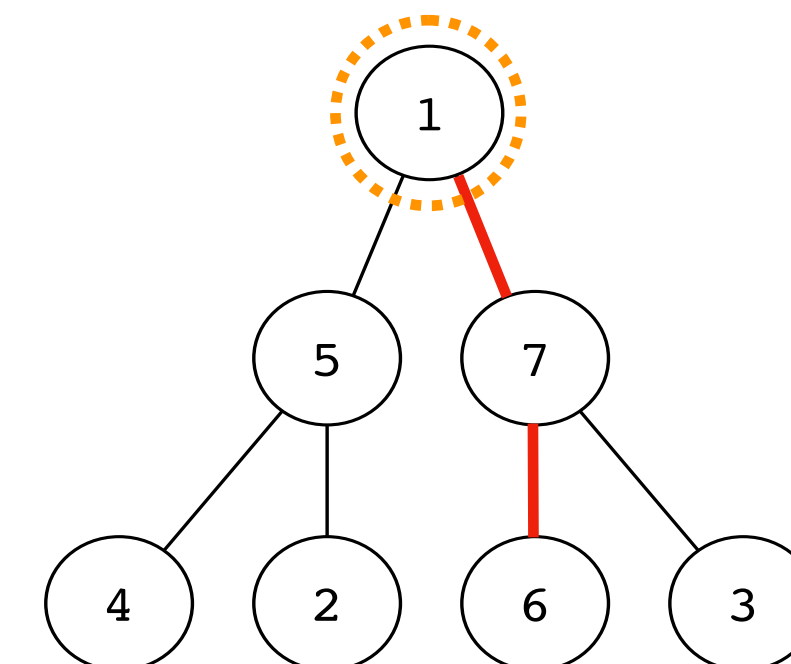
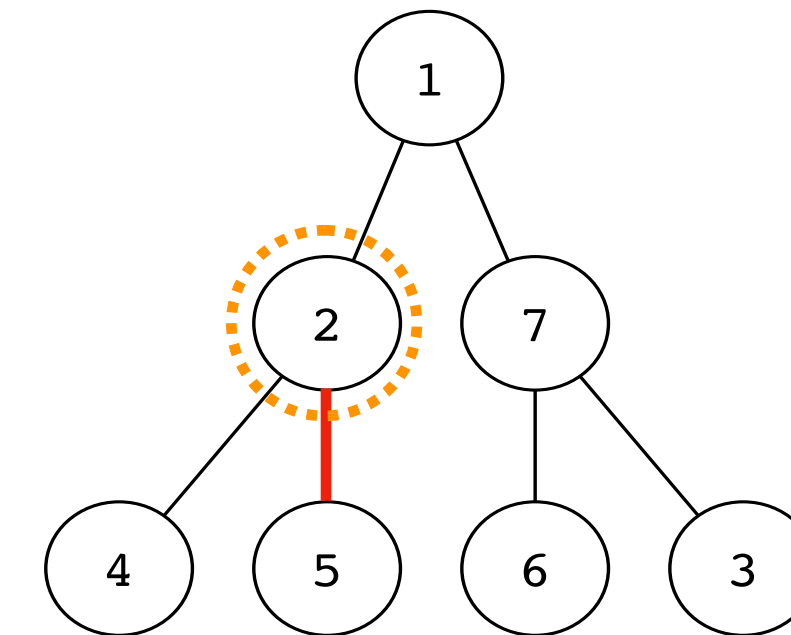
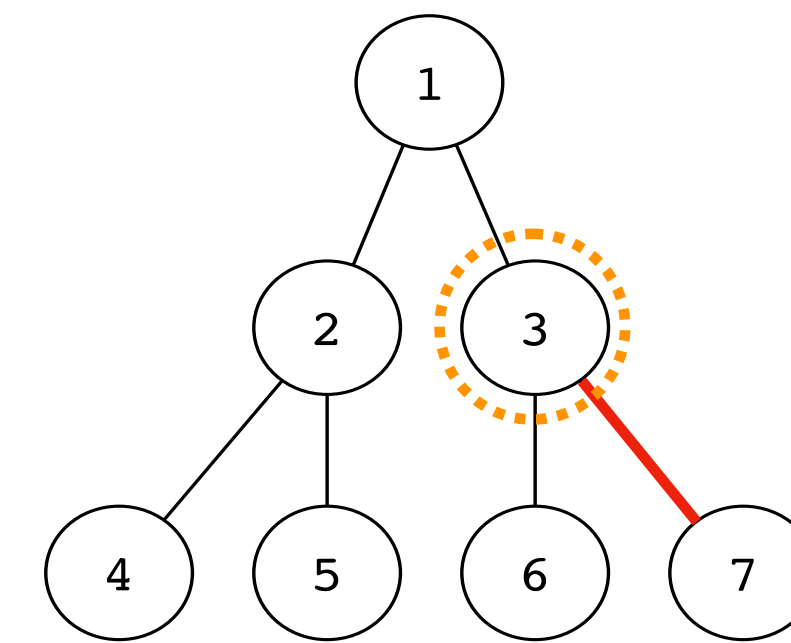
3 ↔ 7

2 ↔ 5

1 ↔ 7, 1 ↔ 6

7 ↔ 3, 3 ↔ 6

6 ↔ 1, 1 ↔ 5, 1 ↔ 4



Exemple

Créer le tas

[1, 2, 3, 4, 5, 6, 7]

[1, 2, 7, 4, 5, 6, 3]

[1, 5, 7, 4, 2, 6, 3]

Tas

[7, 5, 6, 4, 2, 1, 3]

Boucle sur k

[6, 5, 3, 4, 2, 1, 7]

[5, 4, 3, 1, 2, 6, 7]

[4, 2, 3, 1, 5, 6, 7]

Echanges

3 ↔ 7

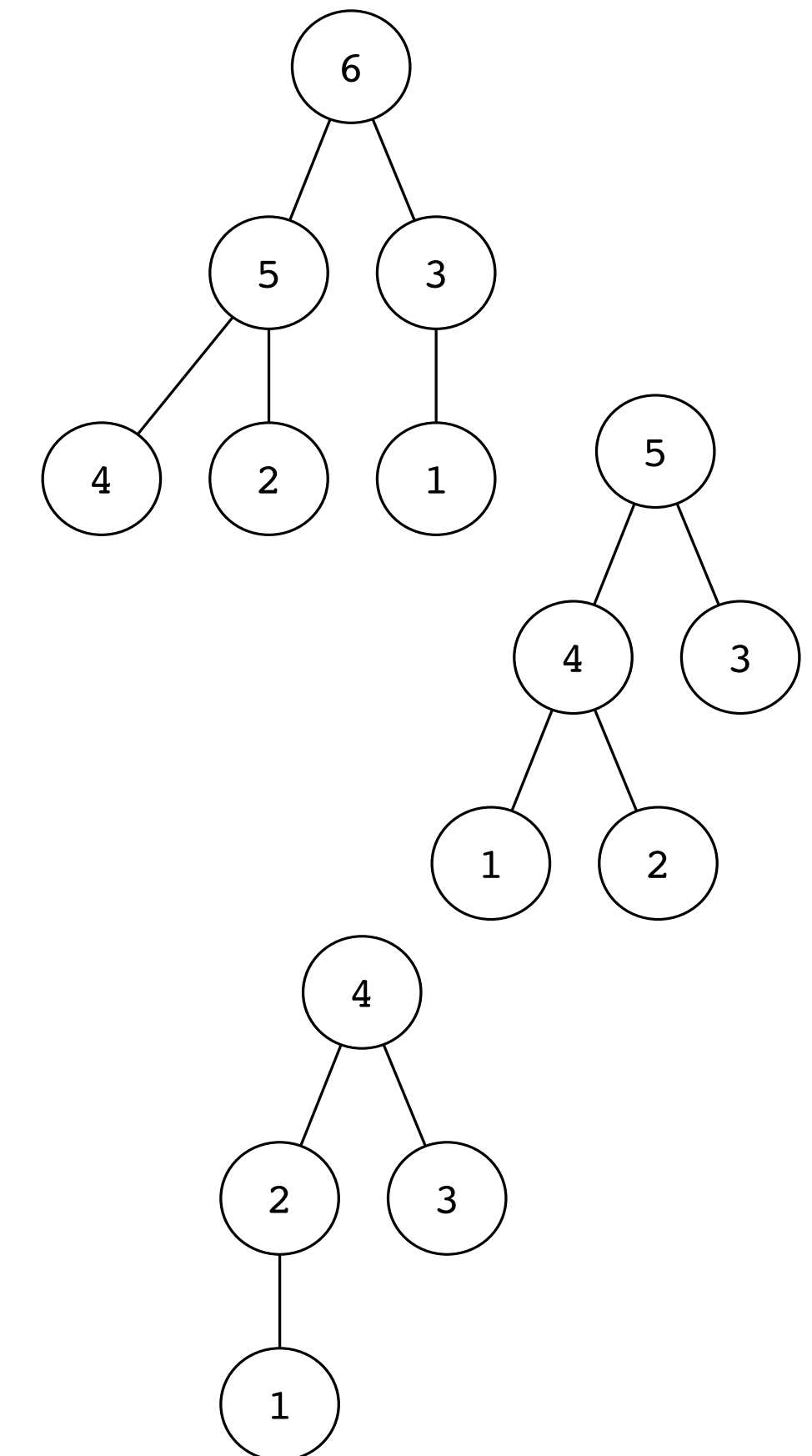
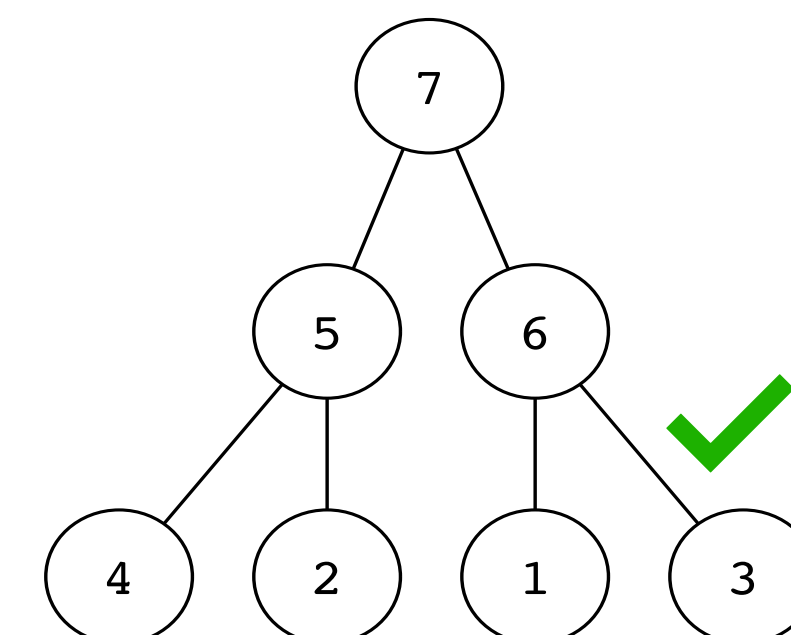
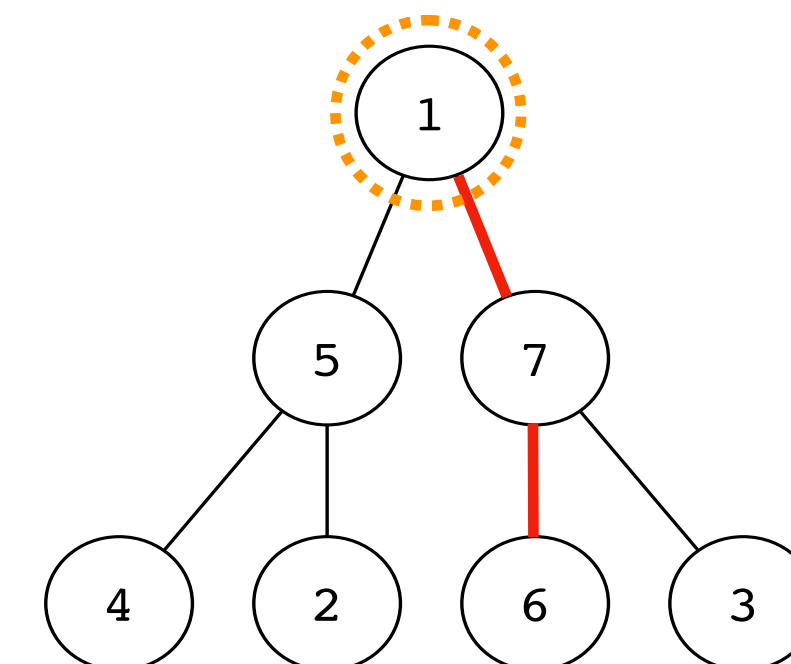
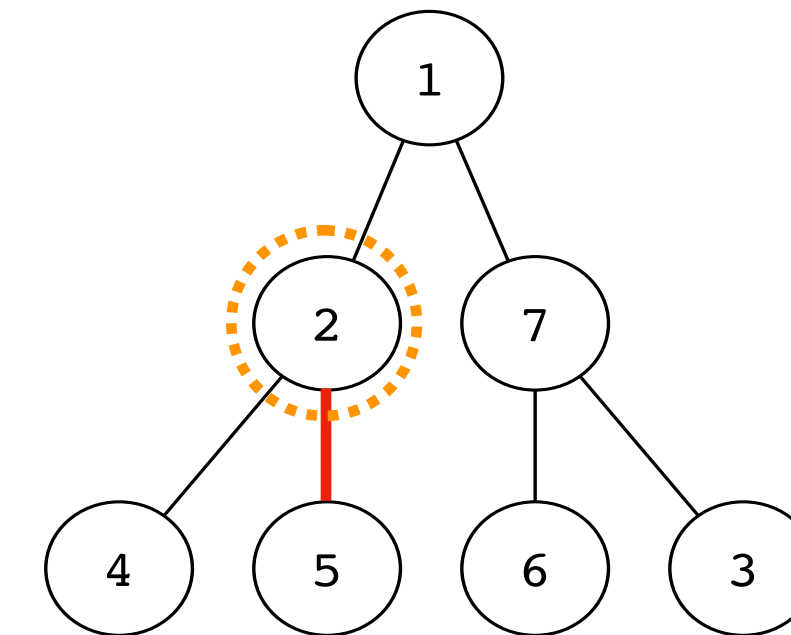
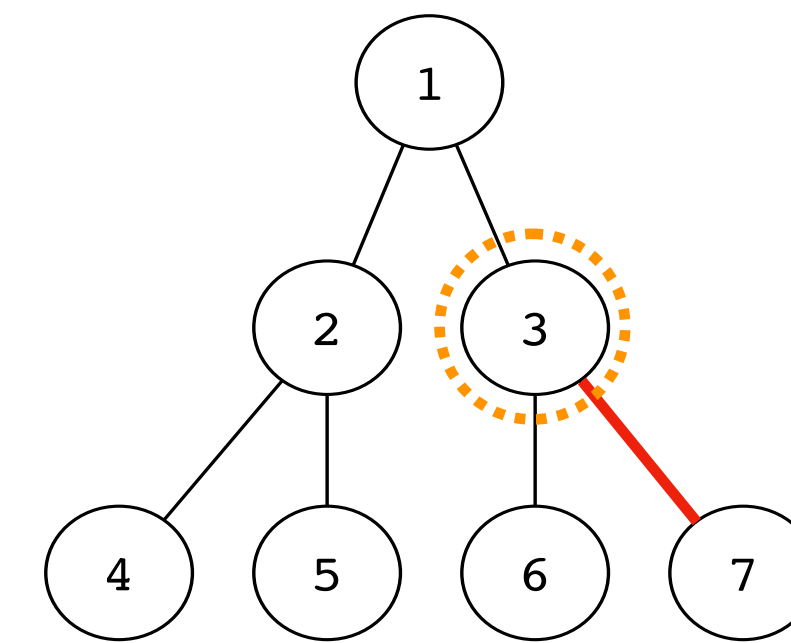
2 ↔ 5

1 ↔ 7, 1 ↔ 6

7 ↔ 3, 3 ↔ 6

6 ↔ 1, 1 ↔ 5, 1 ↔ 4

5 ↔ 2, 2 ↔ 4



Exemple

Créer le tas

[1, 2, 3, 4, 5, 6, 7]

[1, 2, 7, 4, 5, 6, 3]

[1, 5, 7, 4, 2, 6, 3]

Tas

[7, 5, 6, 4, 2, 1, 3]

Boucle sur k

[6, 5, 3, 4, 2, 1, 7]

[5, 4, 3, 1, 2, 6, 7]

[4, 2, 3, 1, 5, 6, 7]

[3, 2, 1, 4, 5, 6, 7]

Echanges

3 ↔ 7

2 ↔ 5

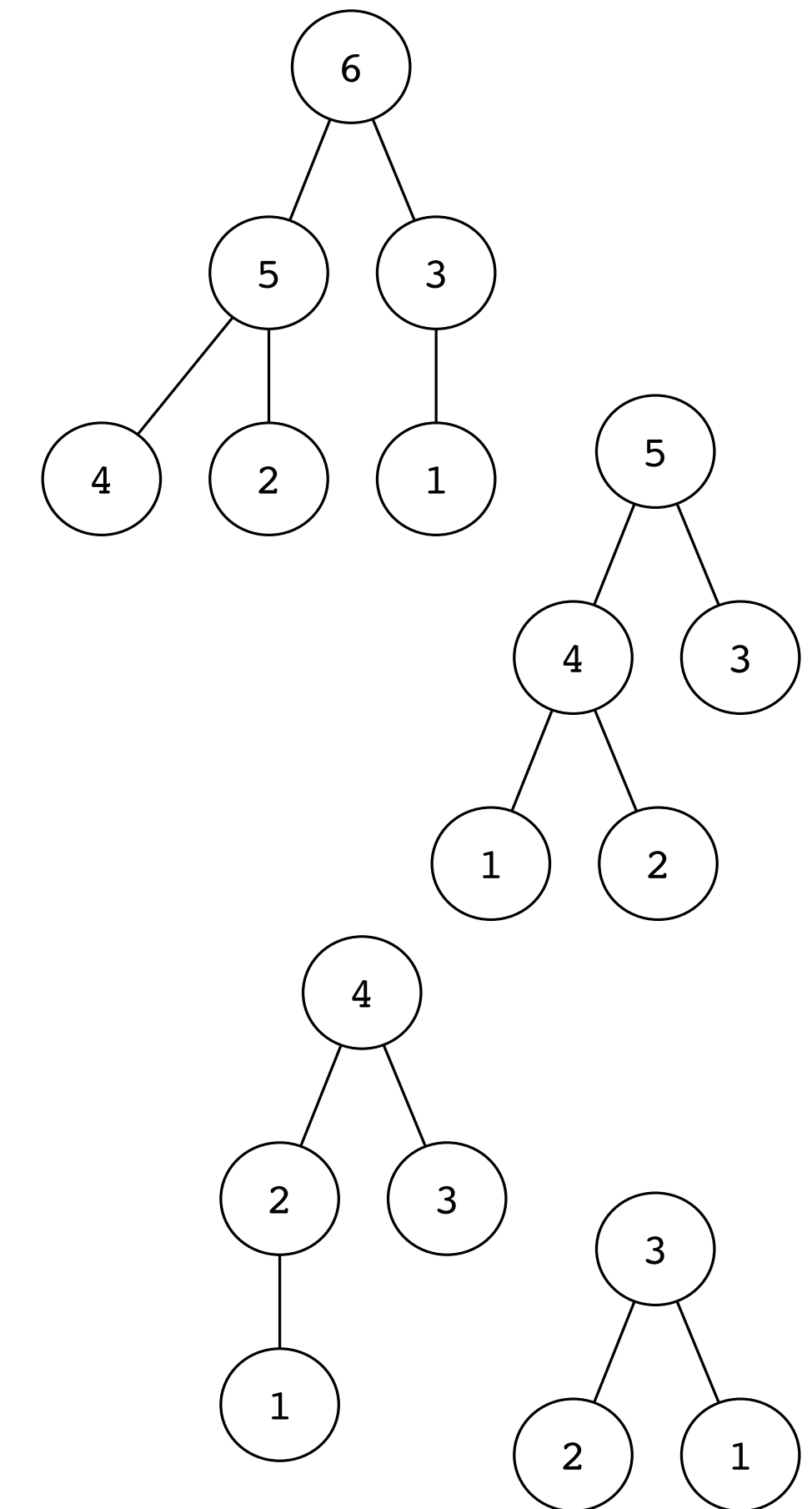
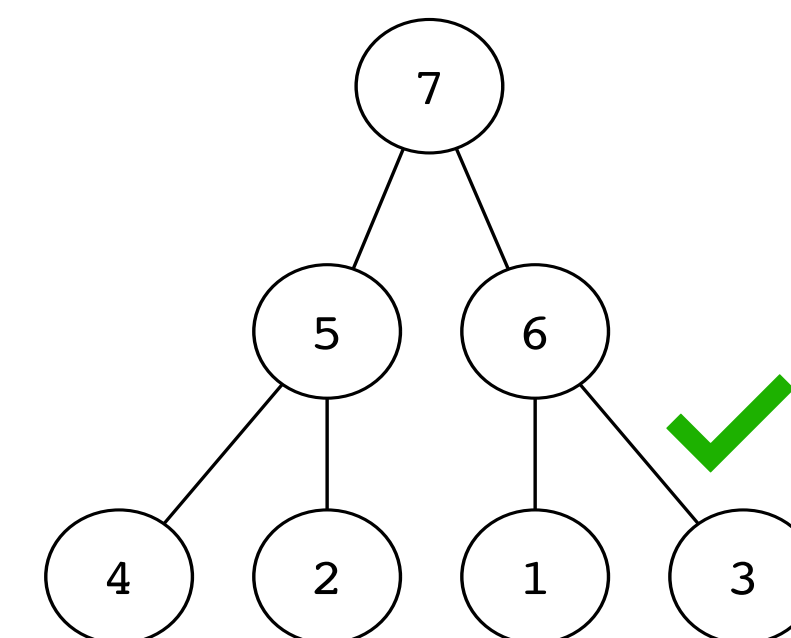
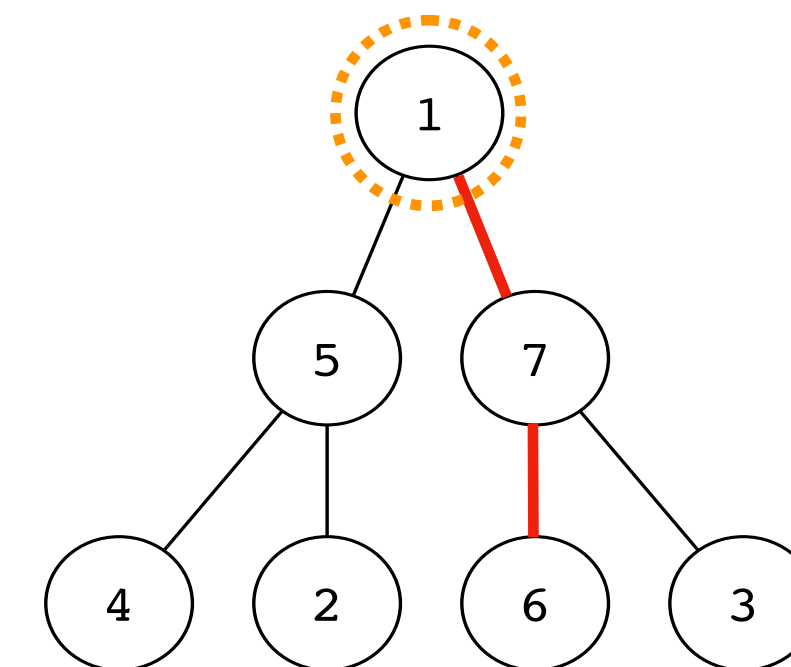
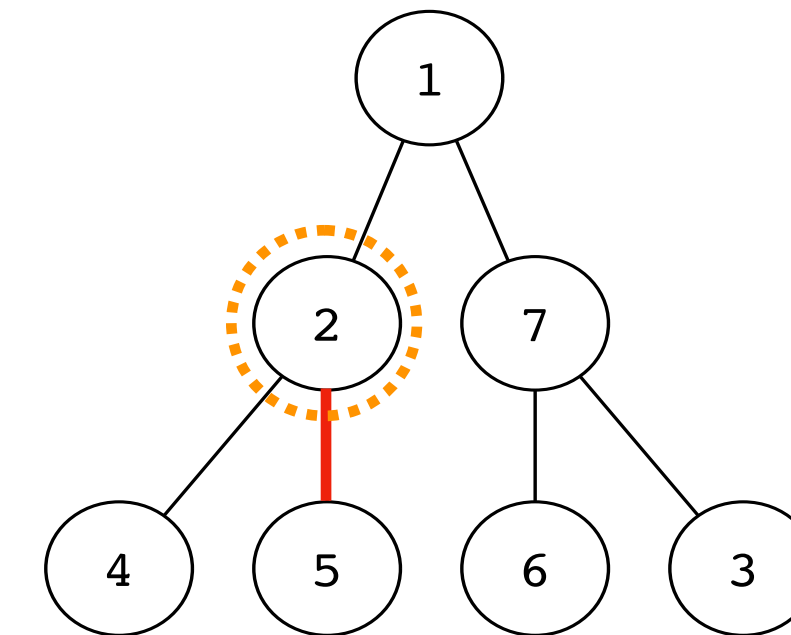
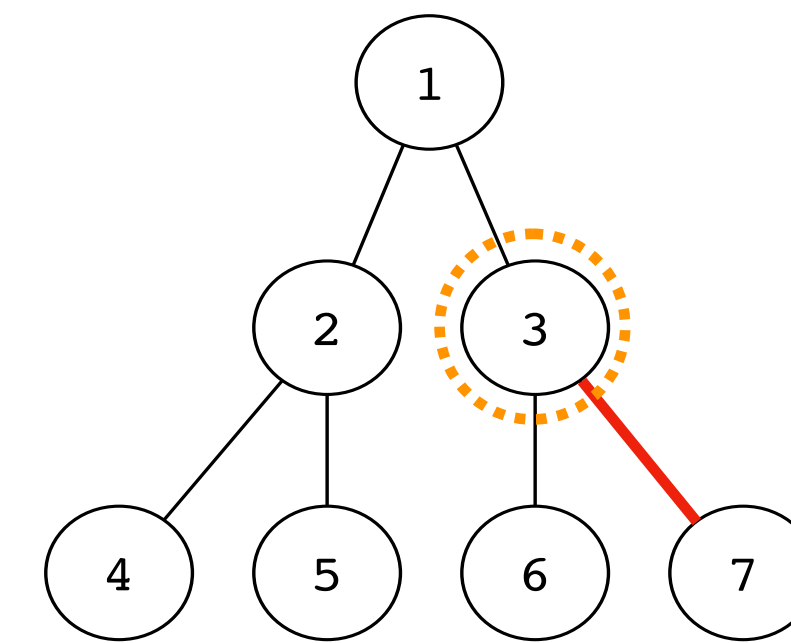
1 ↔ 7, 1 ↔ 6

7 ↔ 3, 3 ↔ 6

6 ↔ 1, 1 ↔ 5, 1 ↔ 4

5 ↔ 2, 2 ↔ 4

4 ↔ 1, 1 ↔ 3



Exemple

Créer le tas

[1, 2, 3, 4, 5, 6, 7]

[1, 2, 7, 4, 5, 6, 3]

[1, 5, 7, 4, 2, 6, 3]

Tas

[7, 5, 6, 4, 2, 1, 3]

Boucle sur k

[6, 5, 3, 4, 2, 1, 7]

[5, 4, 3, 1, 2, 6, 7]

[4, 2, 3, 1, 5, 6, 7]

[3, 2, 1, 4, 5, 6, 7]

[2, 1, 3, 4, 5, 6, 7]

Echanges

3 ↔ 7

2 ↔ 5

1 ↔ 7, 1 ↔ 6

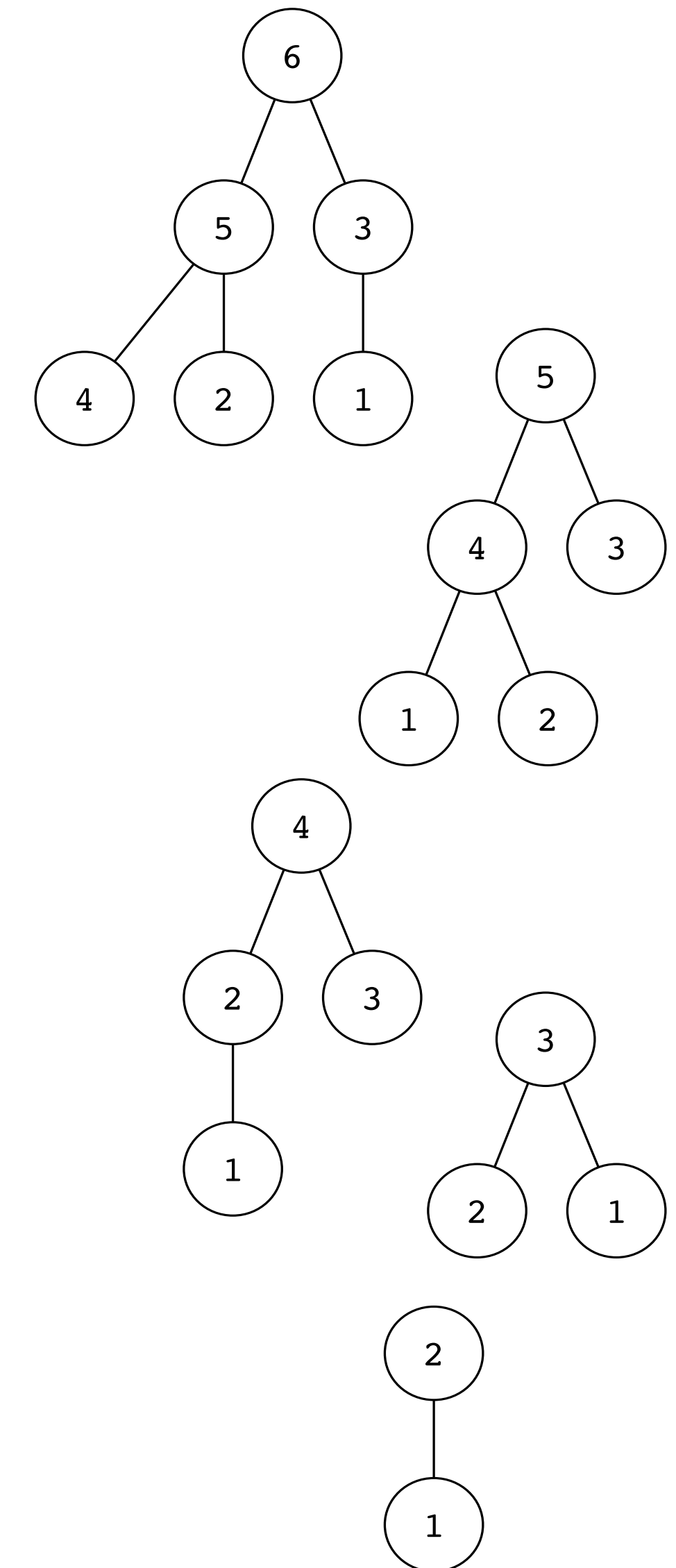
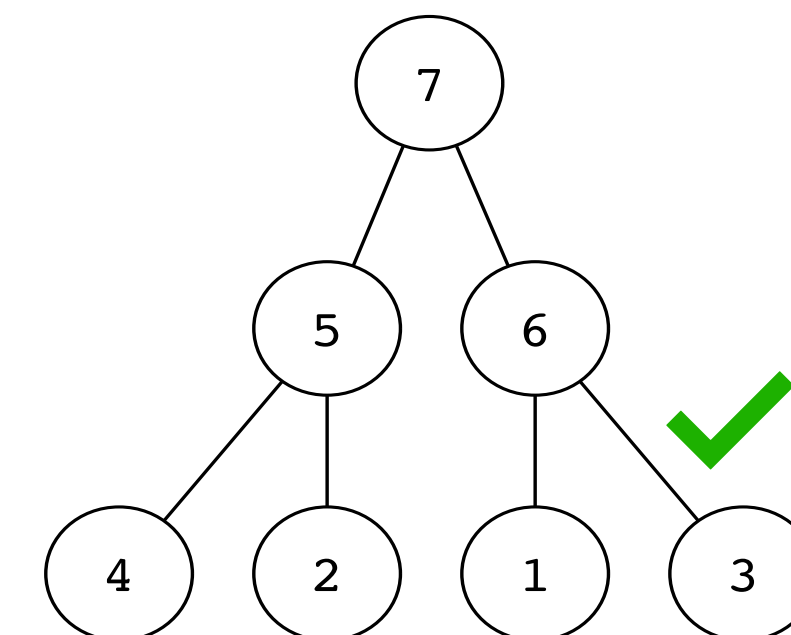
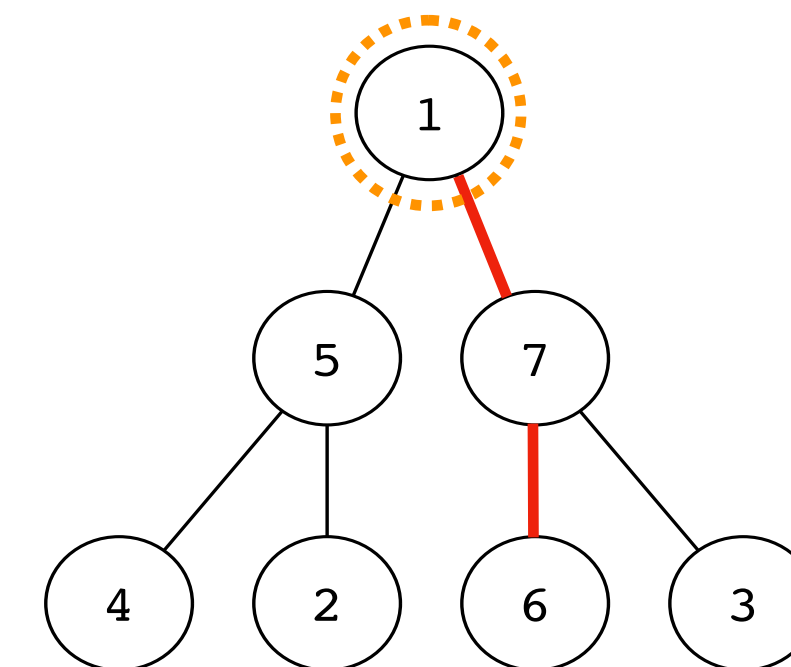
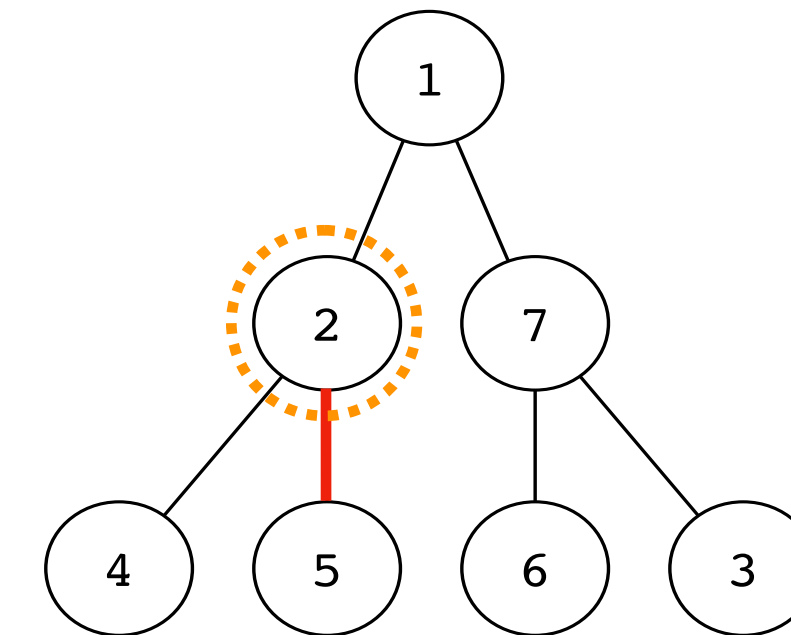
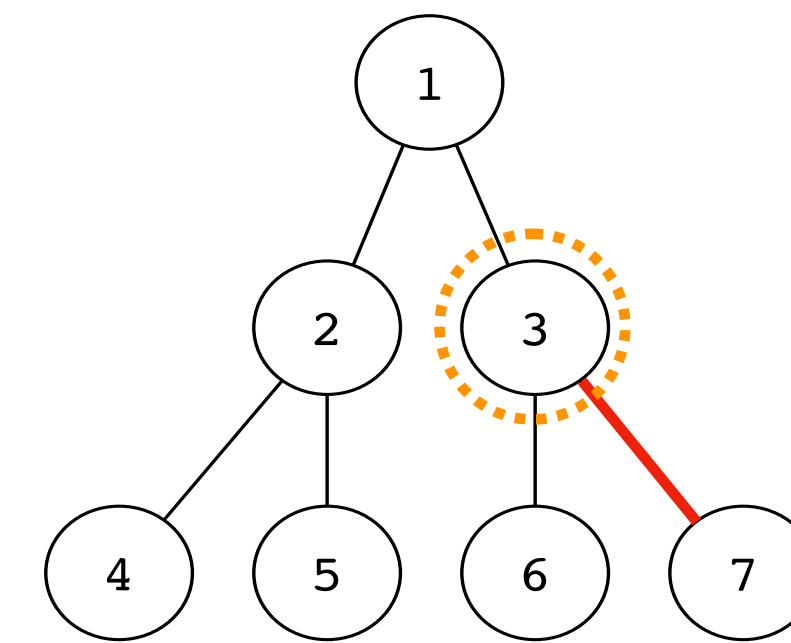
7 ↔ 3, 3 ↔ 6

6 ↔ 1, 1 ↔ 5, 1 ↔ 4

5 ↔ 2, 2 ↔ 4

4 ↔ 1, 1 ↔ 3

3 ↔ 1, 1 ↔ 2



Exemple

Créer le tas

[1, 2, 3, 4, 5, 6, 7]

[1, 2, 7, 4, 5, 6, 3]

[1, 5, 7, 4, 2, 6, 3]

Tas

[7, 5, 6, 4, 2, 1, 3]

Boucle sur k

[6, 5, 3, 4, 2, 1, 7]

[5, 4, 3, 1, 2, 6, 7]

[4, 2, 3, 1, 5, 6, 7]

[3, 2, 1, 4, 5, 6, 7]

[2, 1, 3, 4, 5, 6, 7]

[1, 2, 3, 4, 5, 6, 7]

Echanges

3 ↔ 7

2 ↔ 5

1 ↔ 7, 1 ↔ 6

7 ↔ 3, 3 ↔ 6

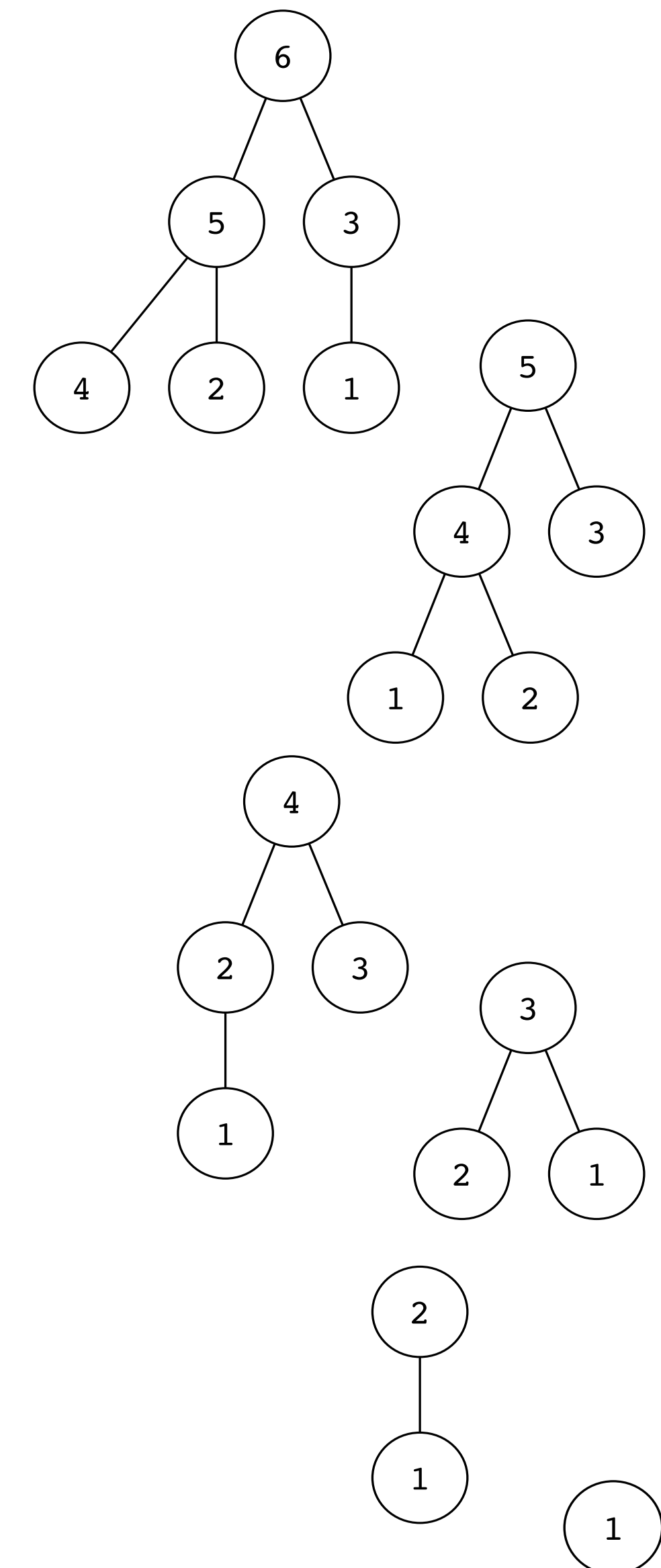
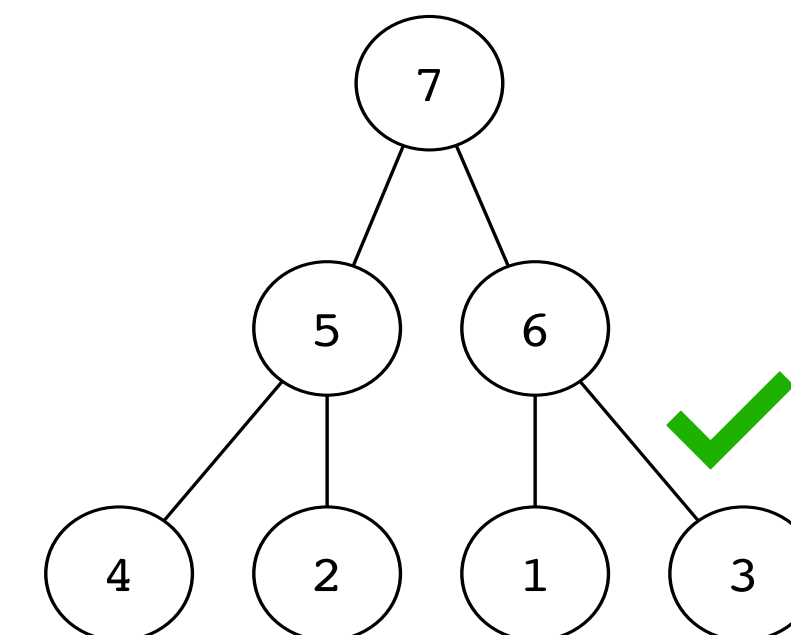
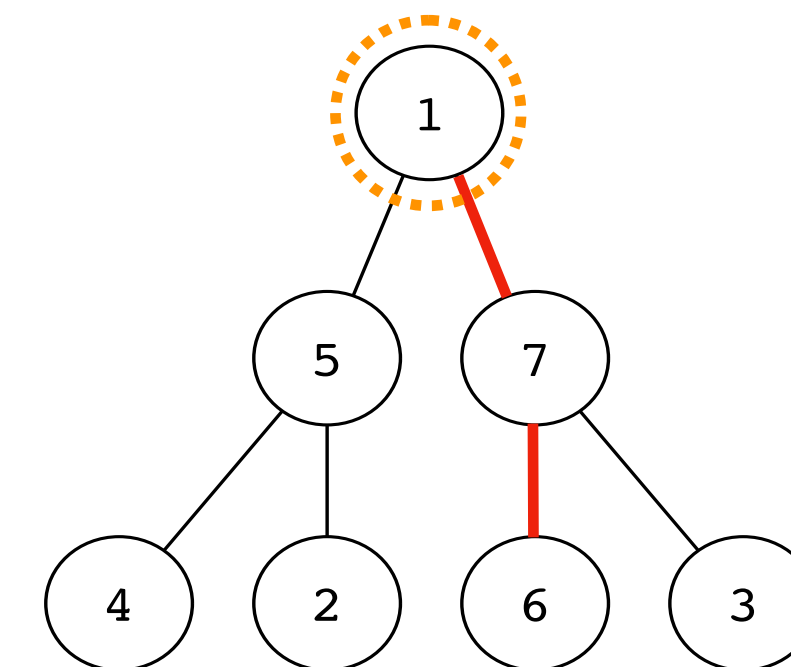
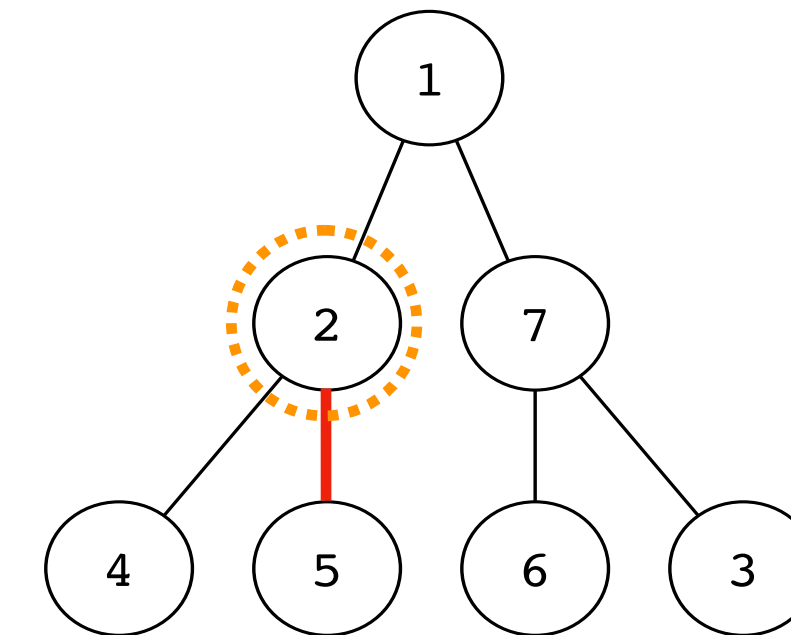
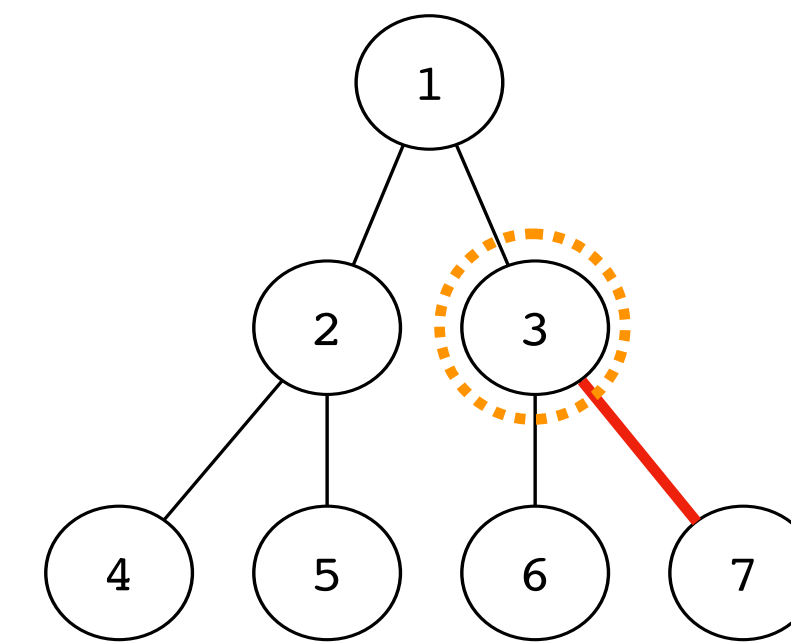
6 ↔ 1, 1 ↔ 5, 1 ↔ 4

5 ↔ 2, 2 ↔ 4

4 ↔ 1, 1 ↔ 3

3 ↔ 1, 1 ↔ 2

2 ↔ 1



En résumé



- Pour créer un tas depuis un tableau non trié
 - En montant les éléments, de 1 à $n-1$ en $O(n \log(n))$, ou
 - En descendant les éléments, de `parent` ($n-1$) à 0 en $O(n)$
- Pour trier par tas un tableau
 - Créer un tas, puis trier, par sélection du maximum, en maintenant la condition de tas pour le début du tableau
 - Complexité en $O(n \log(n))$

Exercices pour la prochaine fois



(Temps estimé : 5 min)

- A Pour chaque tableau ci-dessous, est-ce un tas ?
Si oui, le représenter sous forme d'arbre.

2	3	4	5	1
---	---	---	---	---

5	2	5	1	2
---	---	---	---	---

5	4	3	2	1
---	---	---	---	---

- B Insérer 6 dans le tas suivant.

5	4	5	3	1	4
---	---	---	---	---	---

- C Supprimer le premier 5 du tas suivant.

5	5	3	4	4	2	1	2
---	---	---	---	---	---	---	---

- D Complétez ces phrases de tête :

Pour créer un tas en $O(n)$,

on

	descend	les noeuds de	n-1 à 0
			$(n-1)/2$ à 0
			0 à n-1
			0 à $(n-1)/2$

 remonte

Pour trier un tas en $O(n \log(n))$,

on

	descend	les noeuds de	n-1 à 0
			$(n-1)/2$ à 0
			0 à n-1
			0 à $(n-1)/2$

 remonte