Algo Design

Cours 5-2

Tas

Recherche en table

- Problème : Trouver une information associée à une « clé »
- Exemple : Annuaire, Bibliothèque, …
- **Fonctions :**
  - recherche,
  - insertion,
  - suppression.
- Quelle structure de données utiliser ?

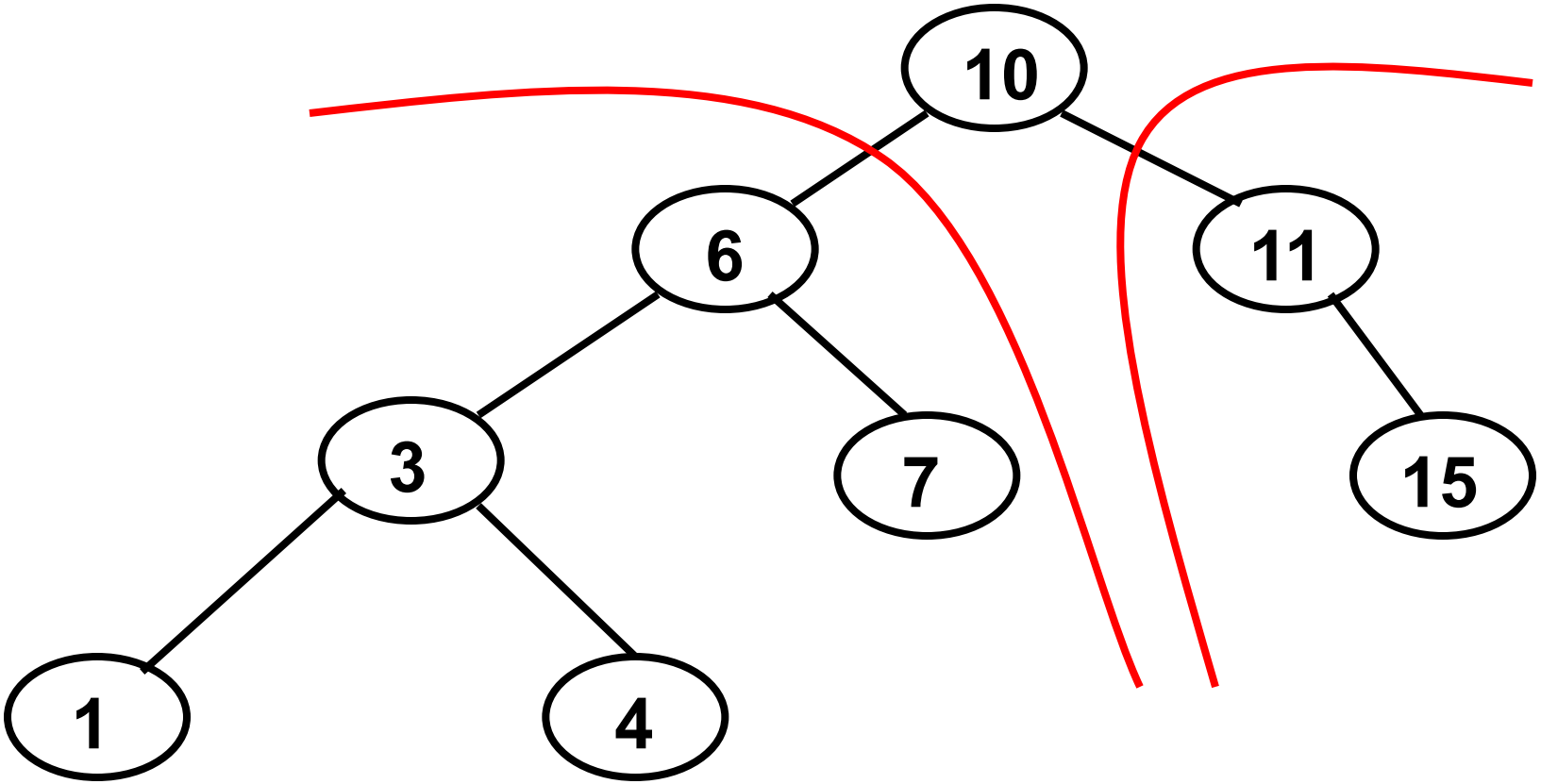
Solutions

- **Table à adressage direct**
  - stockage en $\Theta(m)$
  - opérations en $\Theta(1)$
- **Recherche séquentielle**
  - stockage en $\Theta(n)$
  - opérations en $\Theta(n)$
- **Recherche dichotomique**
  - Recherche en $\Theta(\log(n))$, autres en $\Theta(n)$

Arbre binaire de recherche

- Structure d'arbre binaire
- Chaque nœud contient une clé
- Toute clé d'un nœud de l'arbre est
  - supérieure à celles des descendants de son fils gauche
  - inférieure à celles des descendants de son fils droit

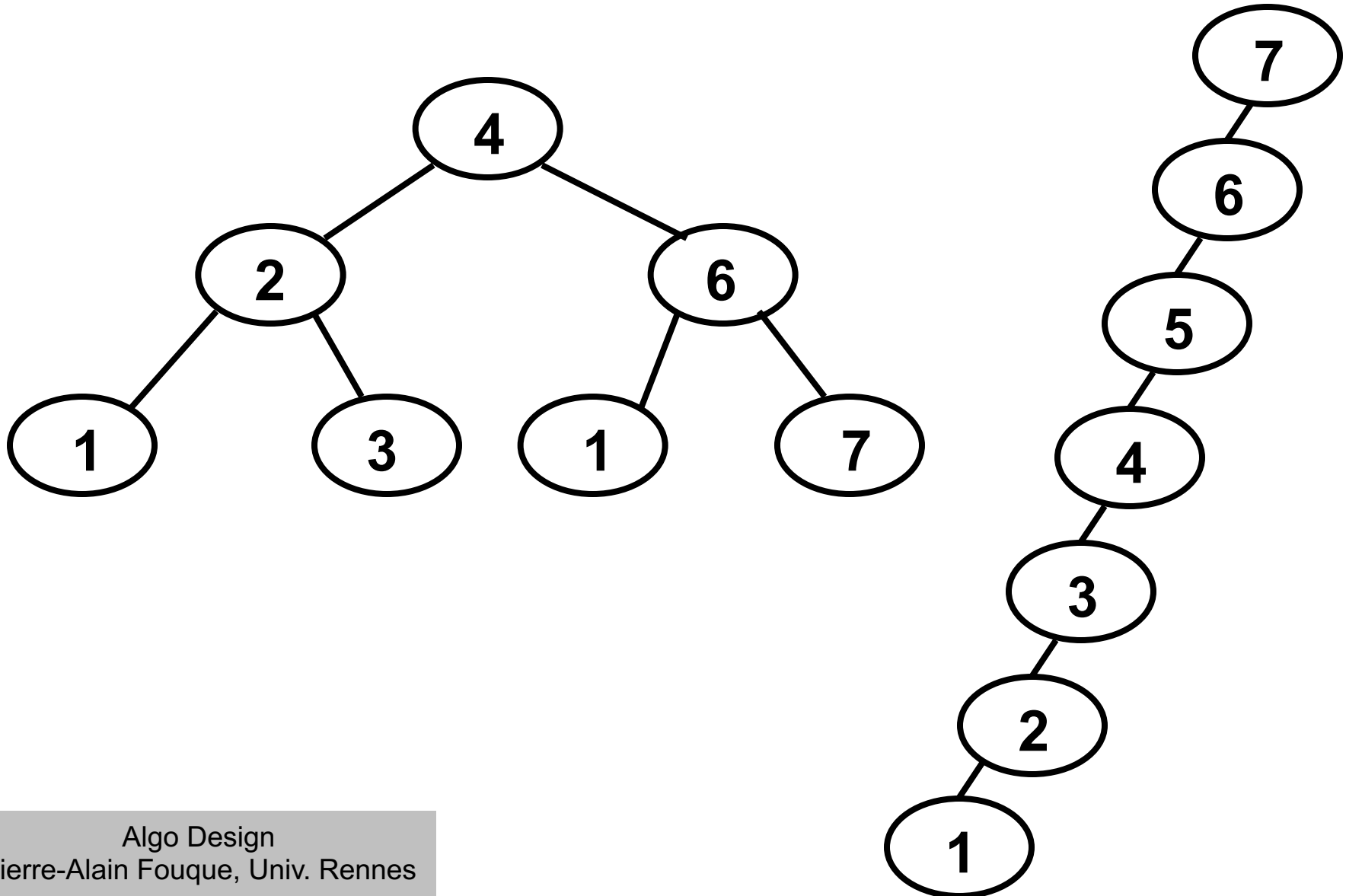
Exemple d'ABR



Complexité des ABR

- stockage en $\Theta(n)$
- recherche en $\Theta(\log(n))$
- insertion en $\Theta(\log(n))$
- suppression en $\Theta(\log(n))$
- **Solution au problème de la recherche en table**
- Dans le pire cas, les complexités sont mauvaises → structure d'arbre **équilibré**

ABRs contenant 1,2,...7



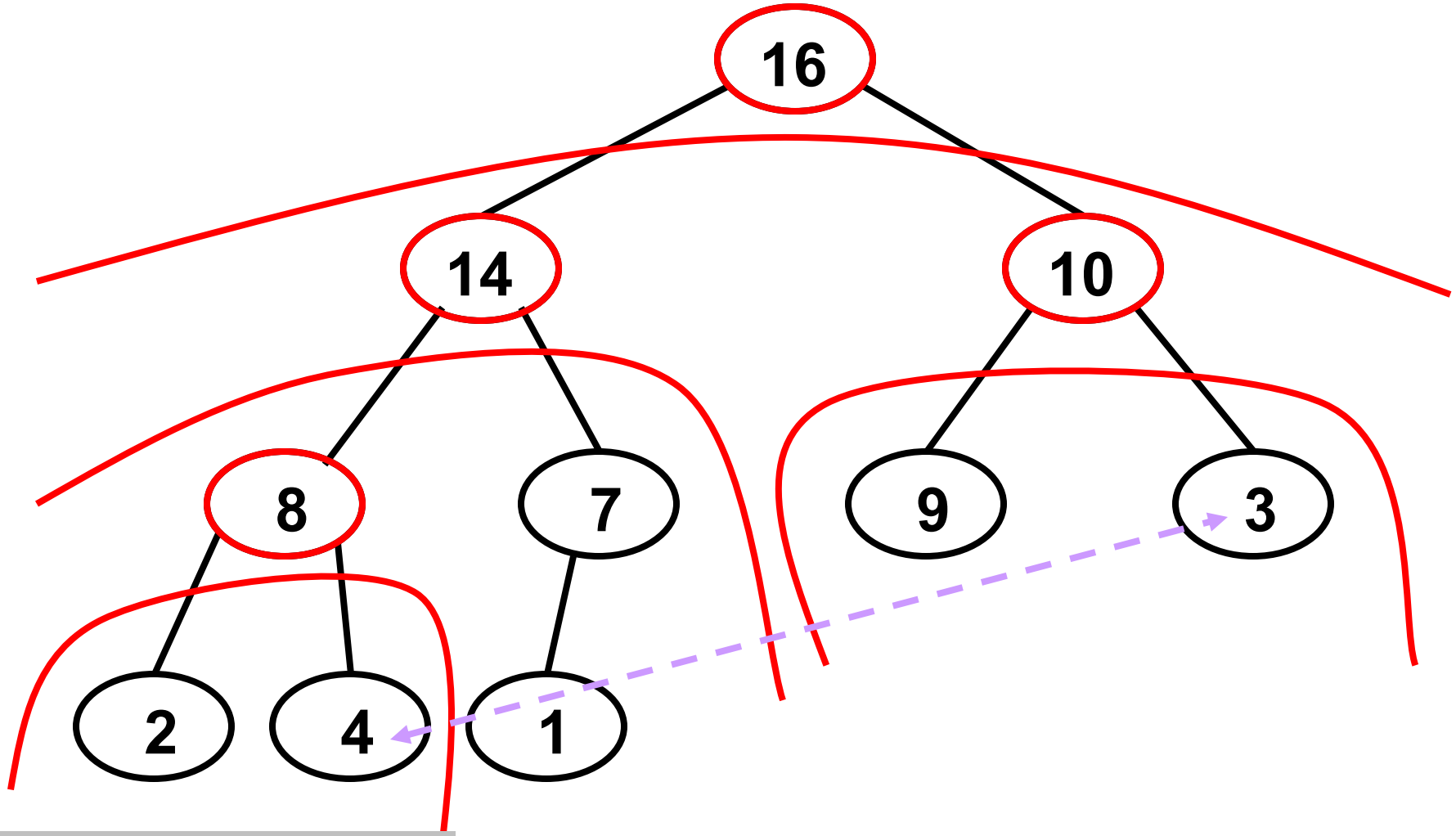
File de priorité

- Problème :
  quelle structure de donnée utiliser afin de réaliser efficacement
  - une fonction d'insertion d'un nouvel élément
  - une fonction qui retourne l'élément de plus haut rang
  - une fonction qui supprime l'élément de plus haut rang
- Application : gestion d'évènements

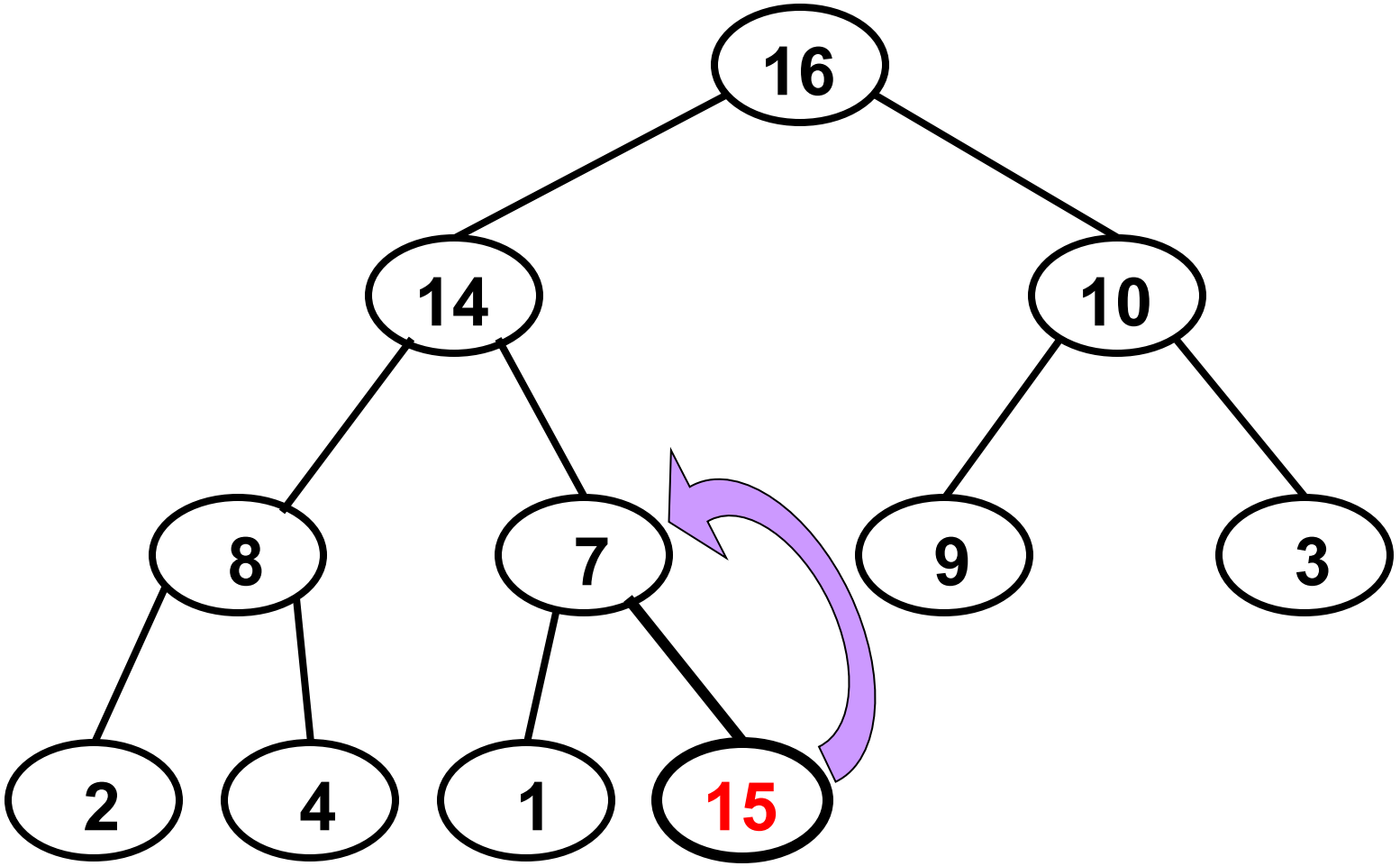
Structure de tas (heap)

- Arbre binaire **complet**
  - hauteur minimale
  - dernier niveau « tassé » à gauche
- **La clé de tout nœud est supérieure ou égale à celles de ses descendants**

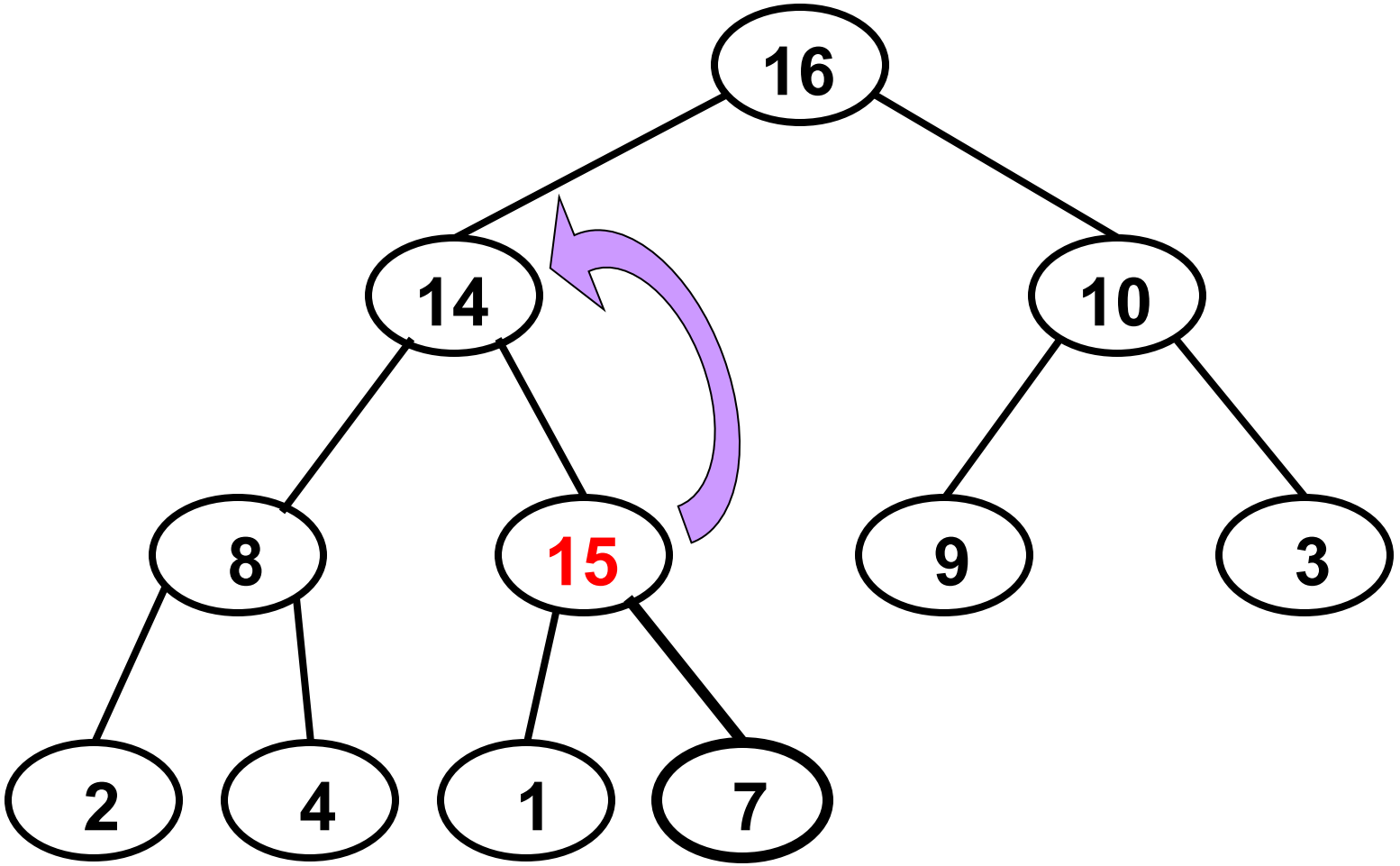
Exemple de tas



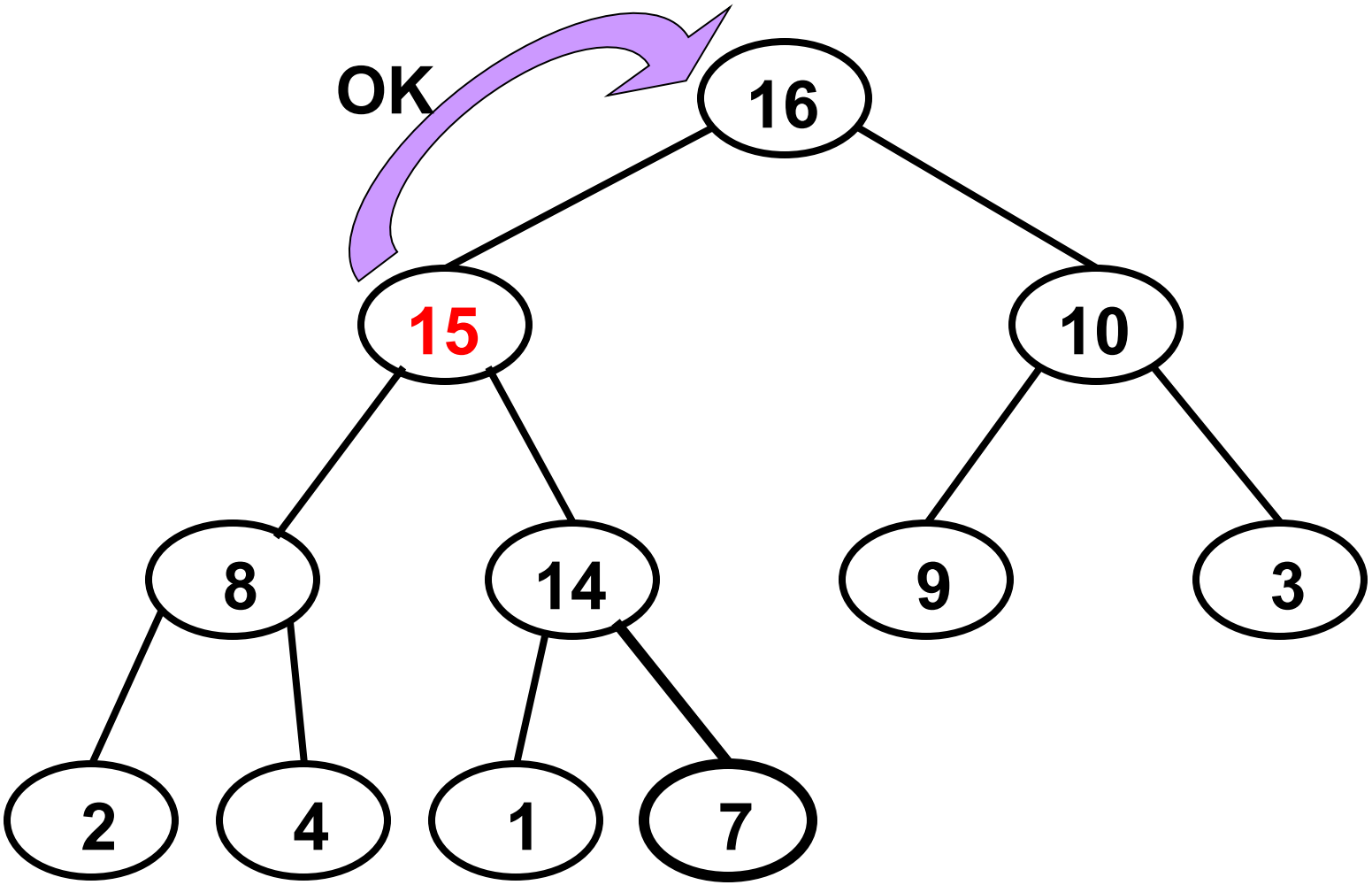
Insertion de 15



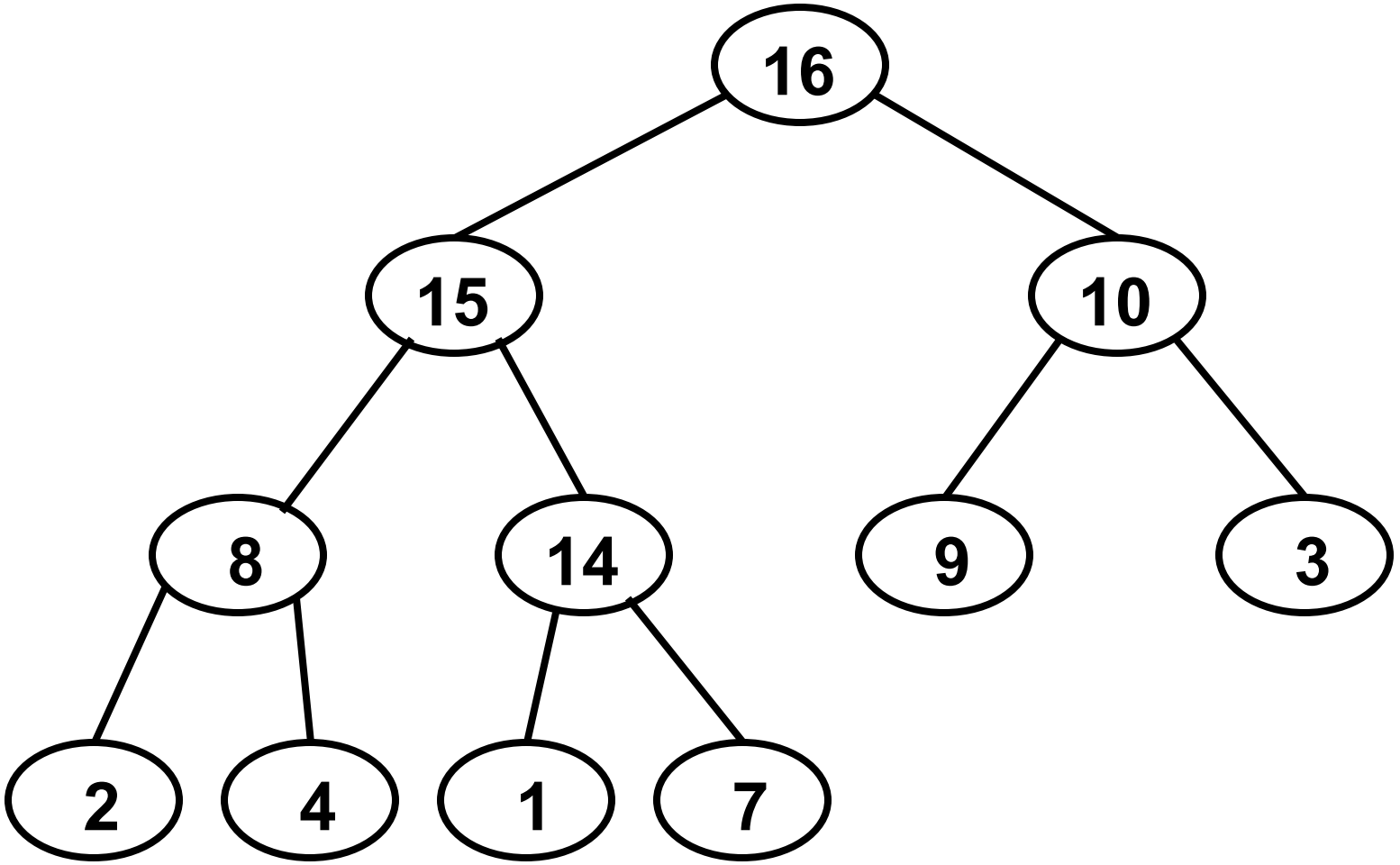
Insertion de 15



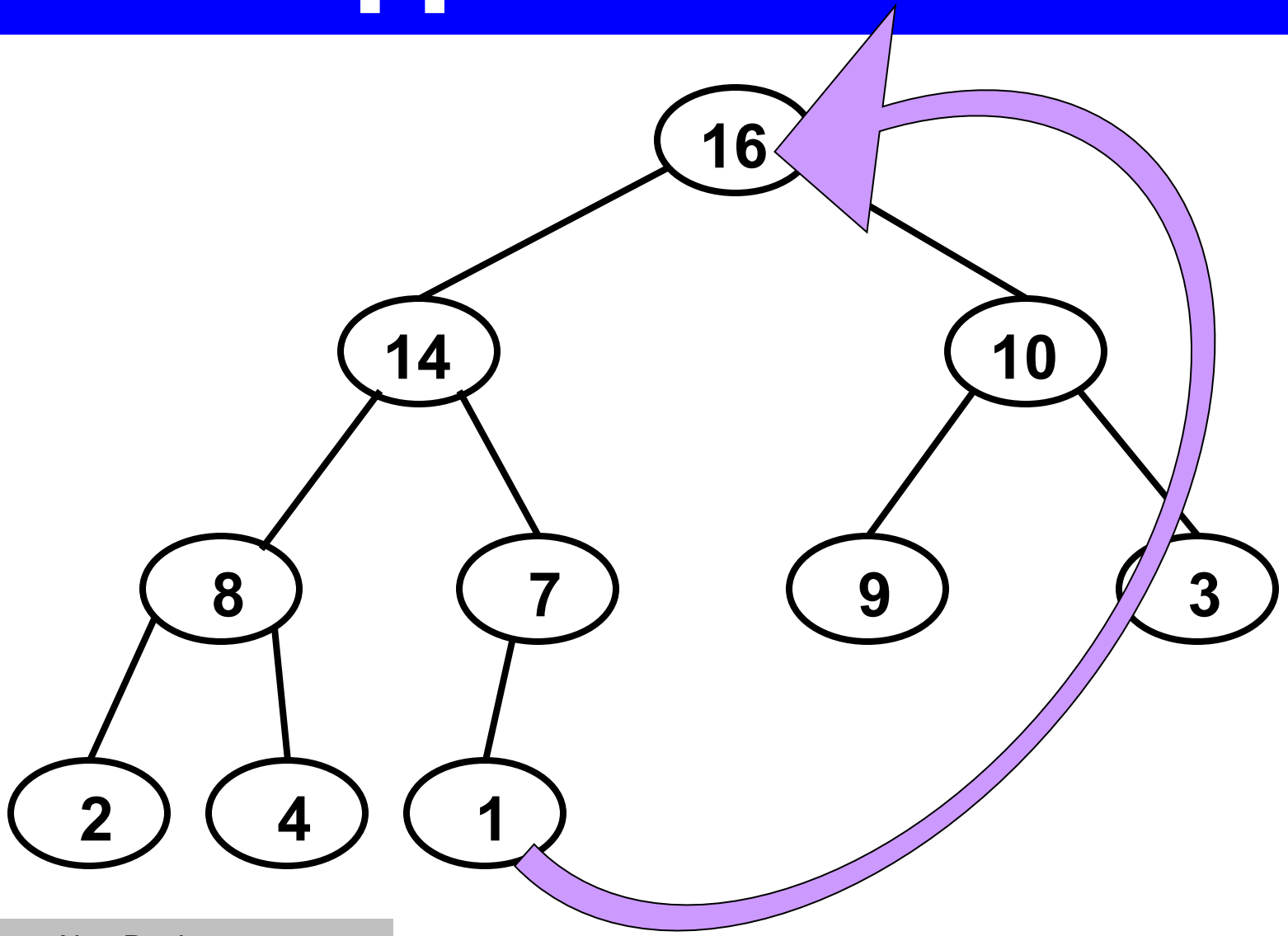
Insertion de 15



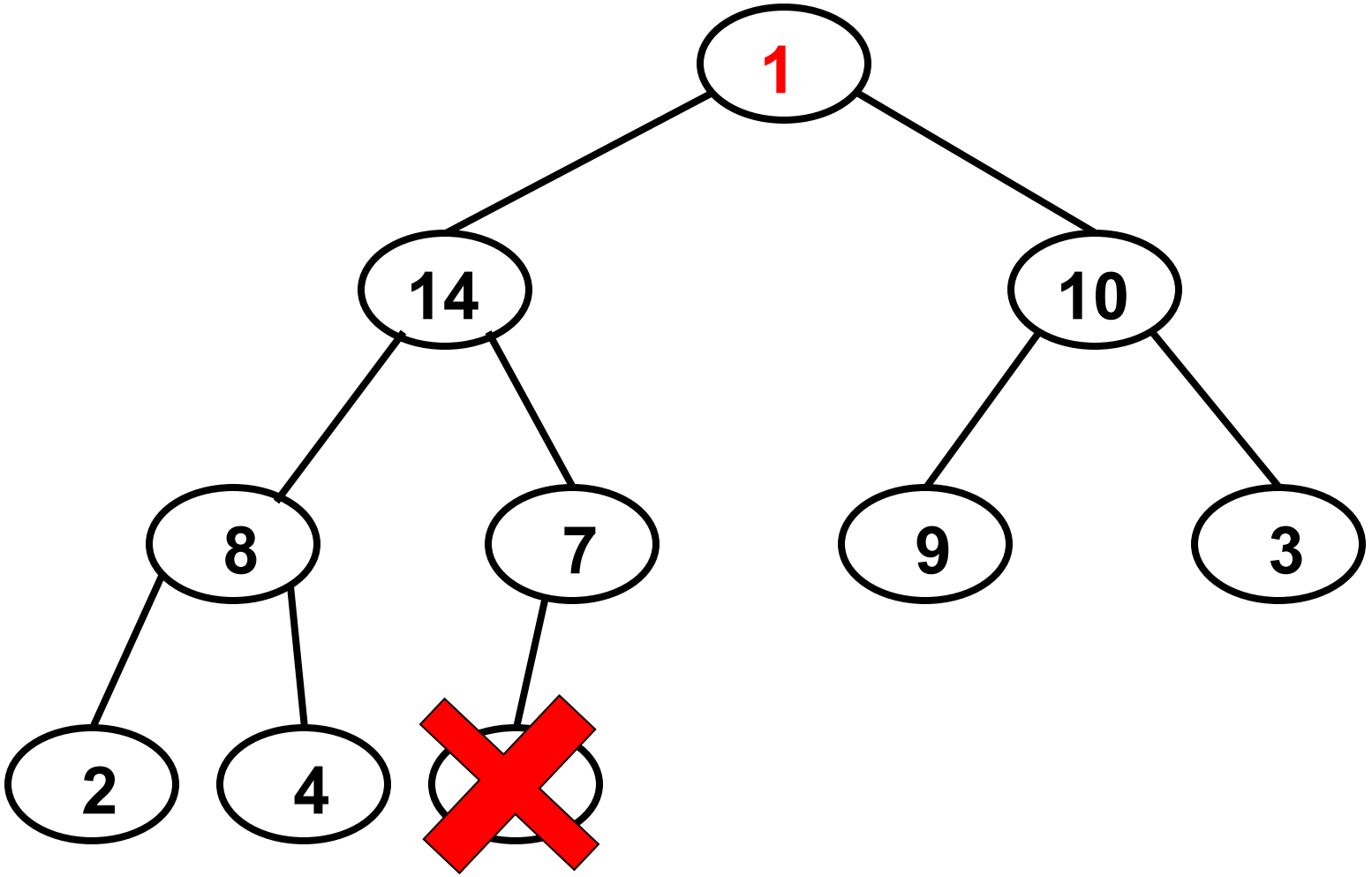
Insertion de 15



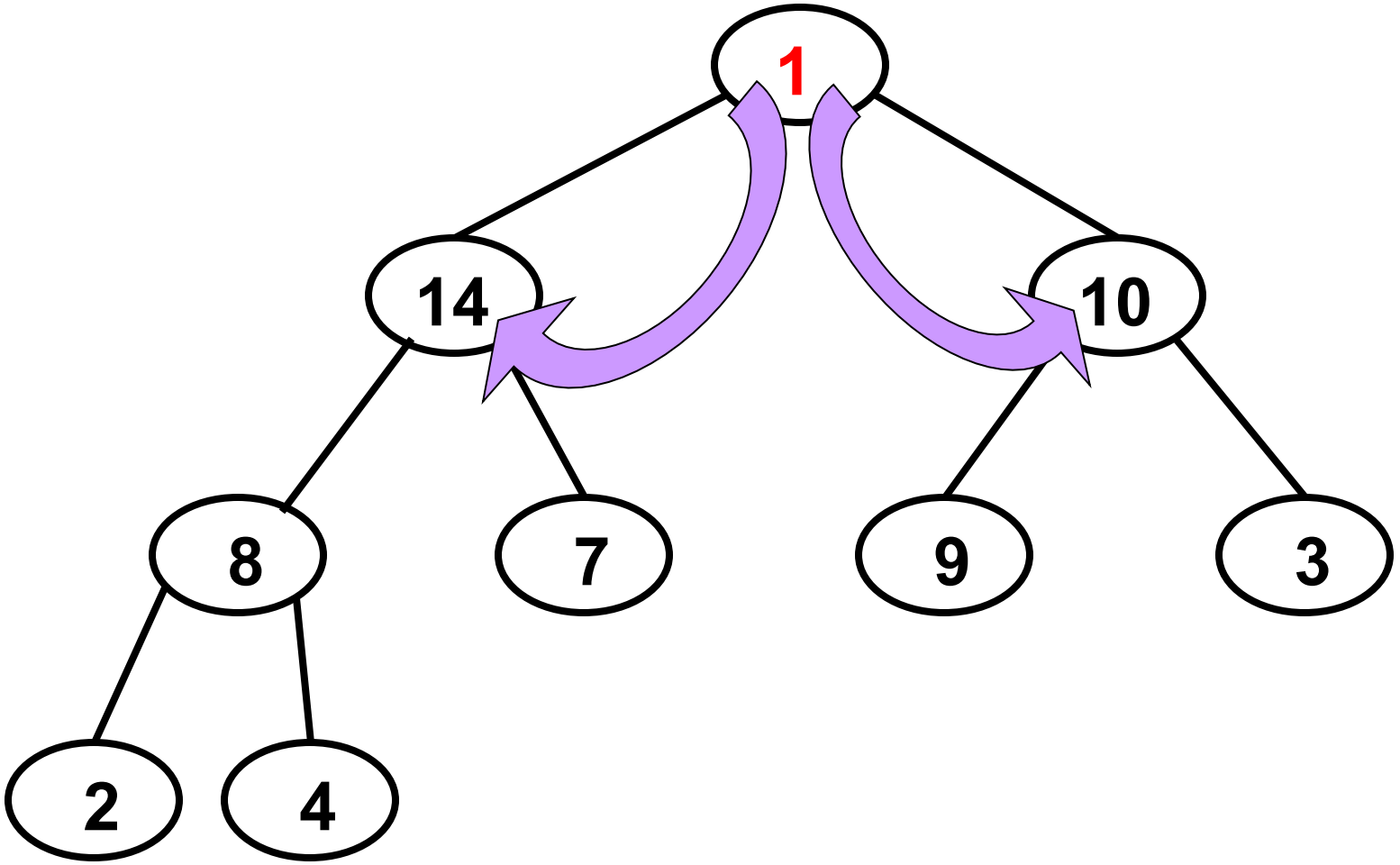
Suppression *racine*



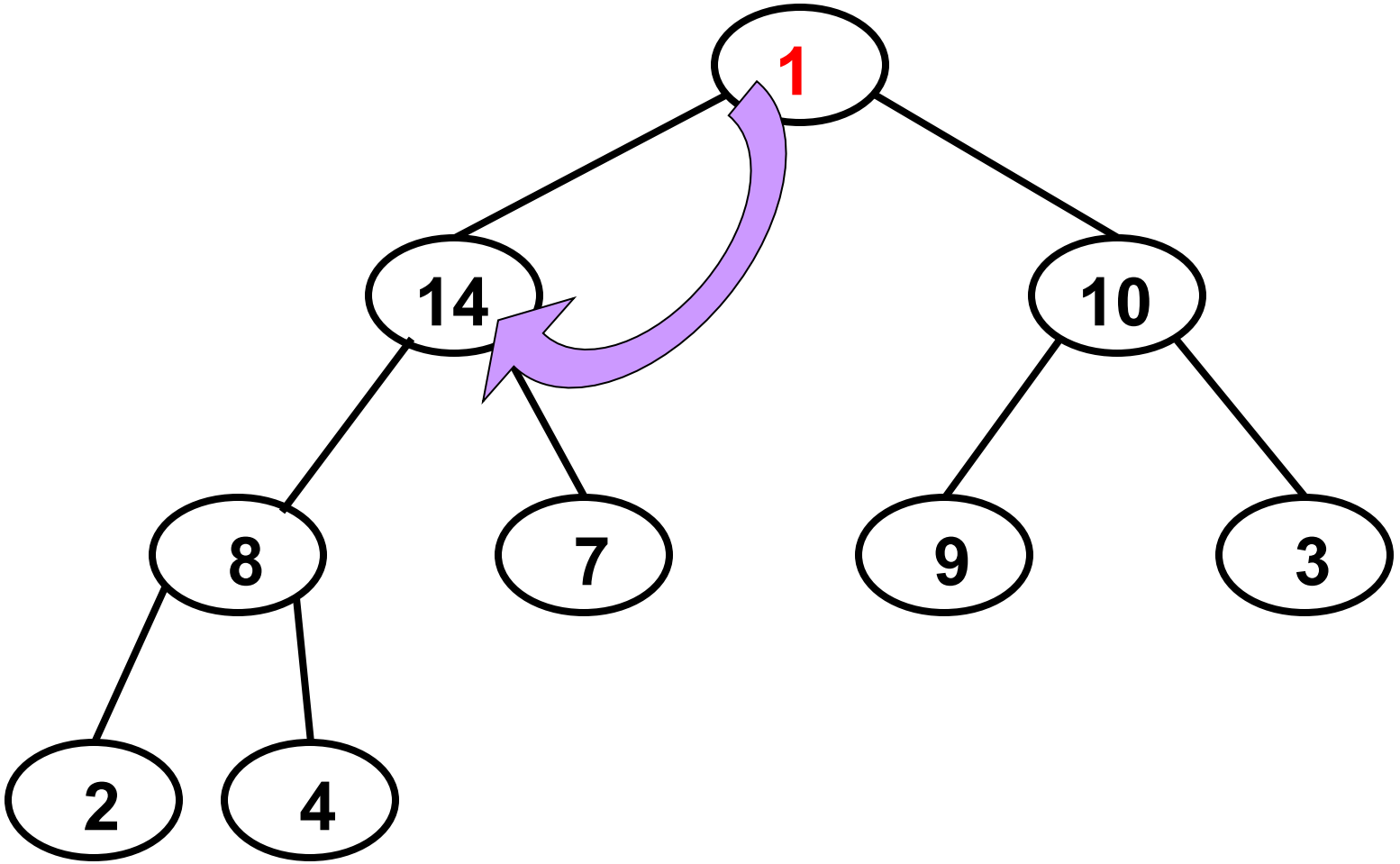
Suppression racine



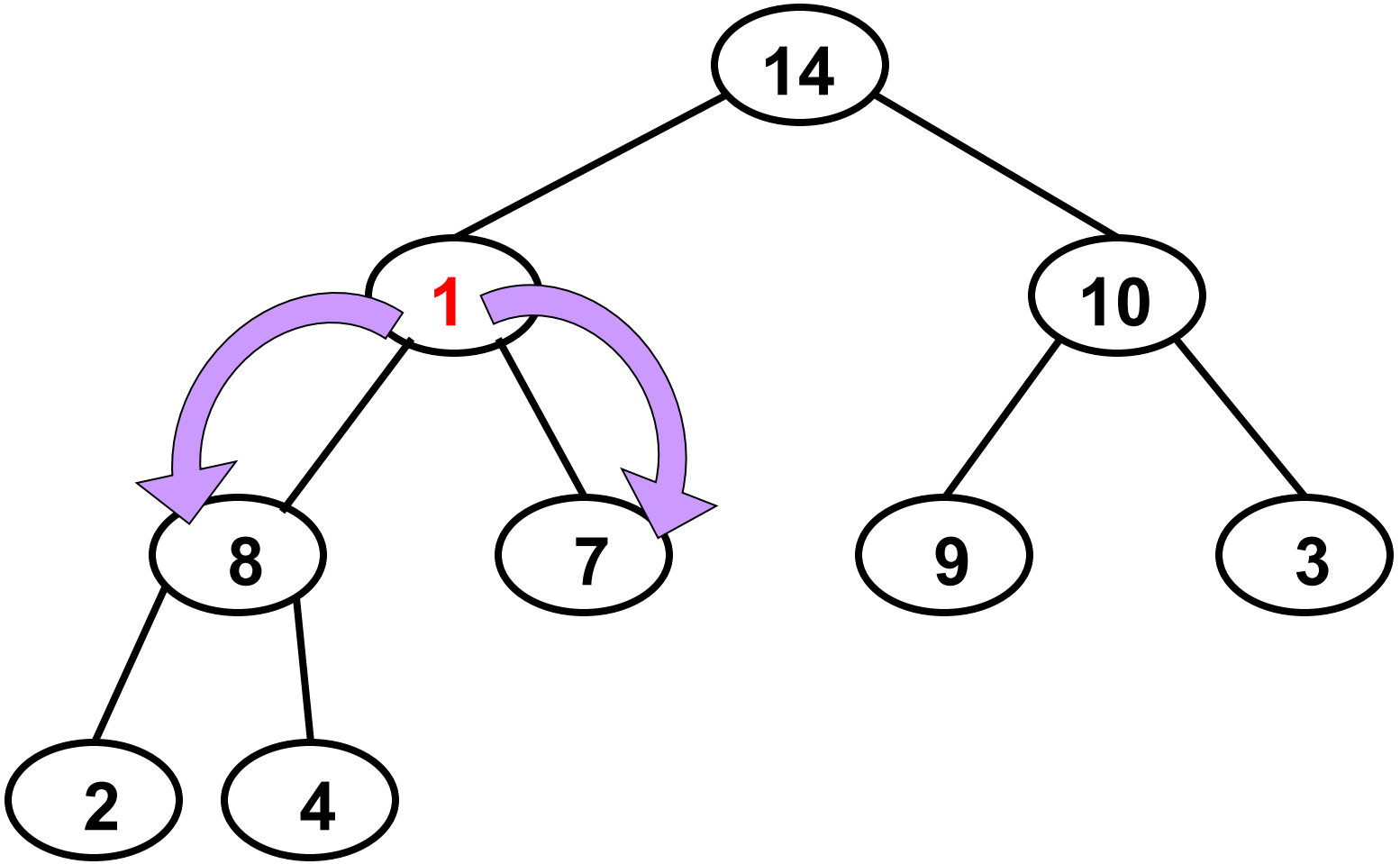
Suppression racine



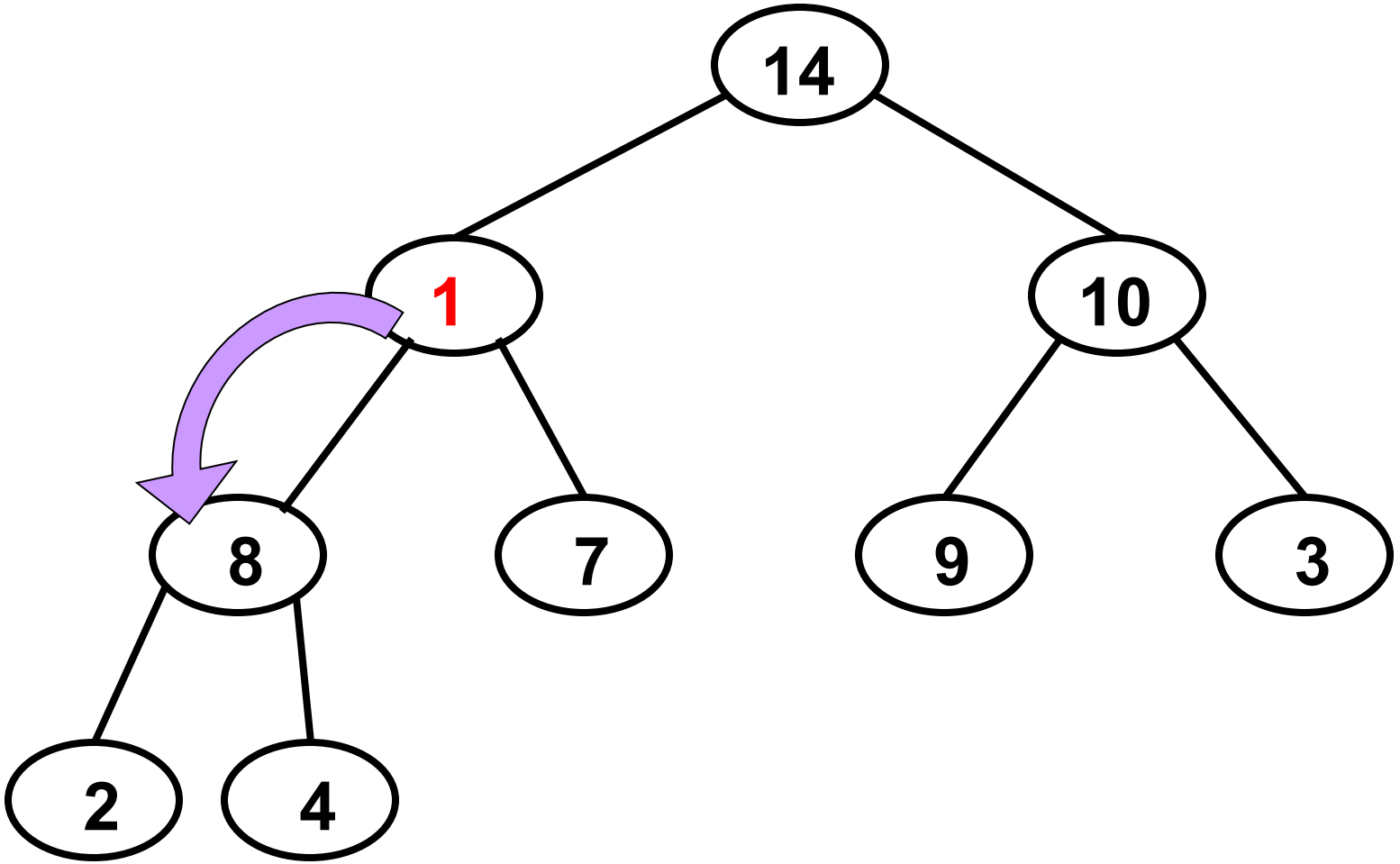
Suppression racine



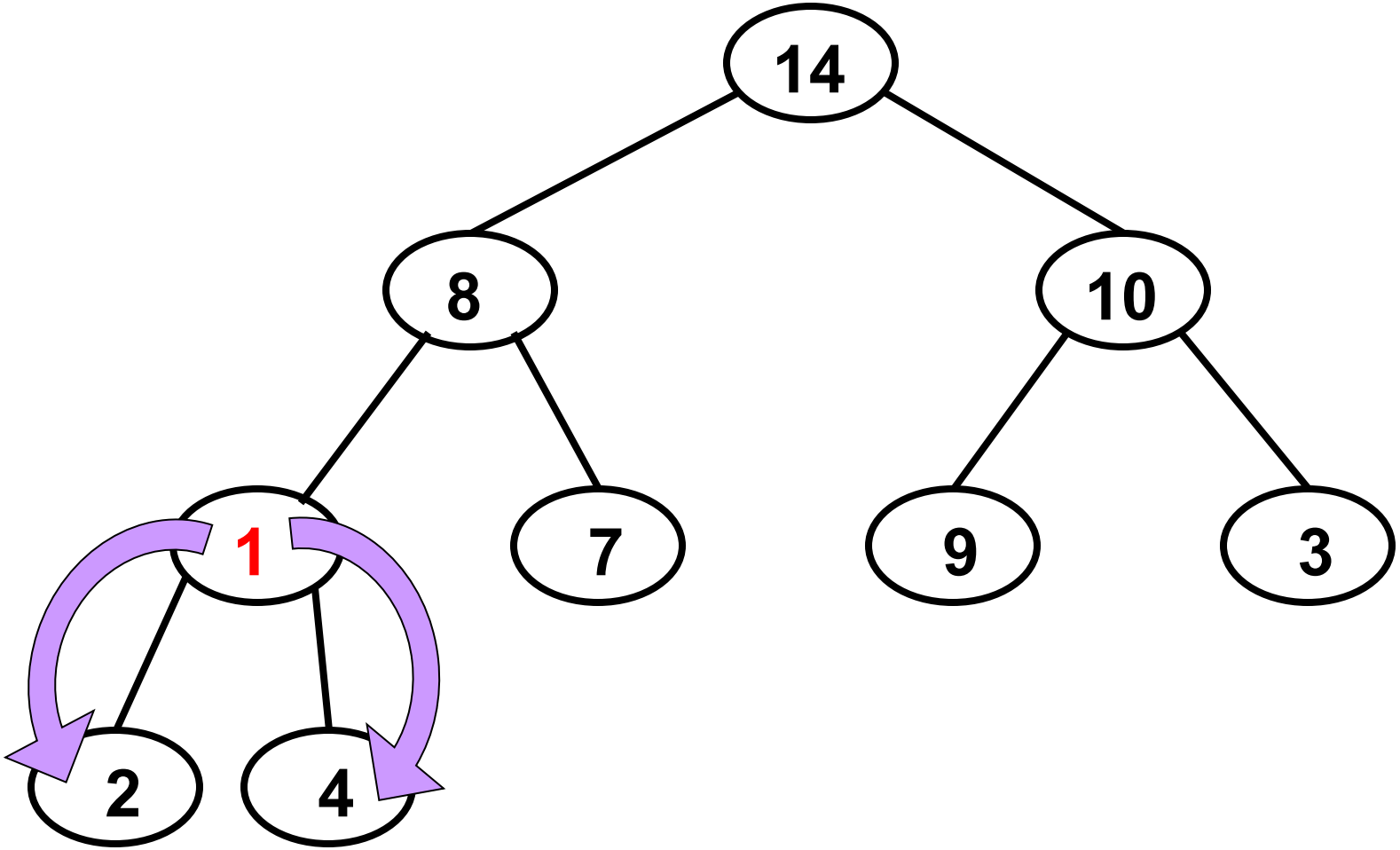
Suppression racine



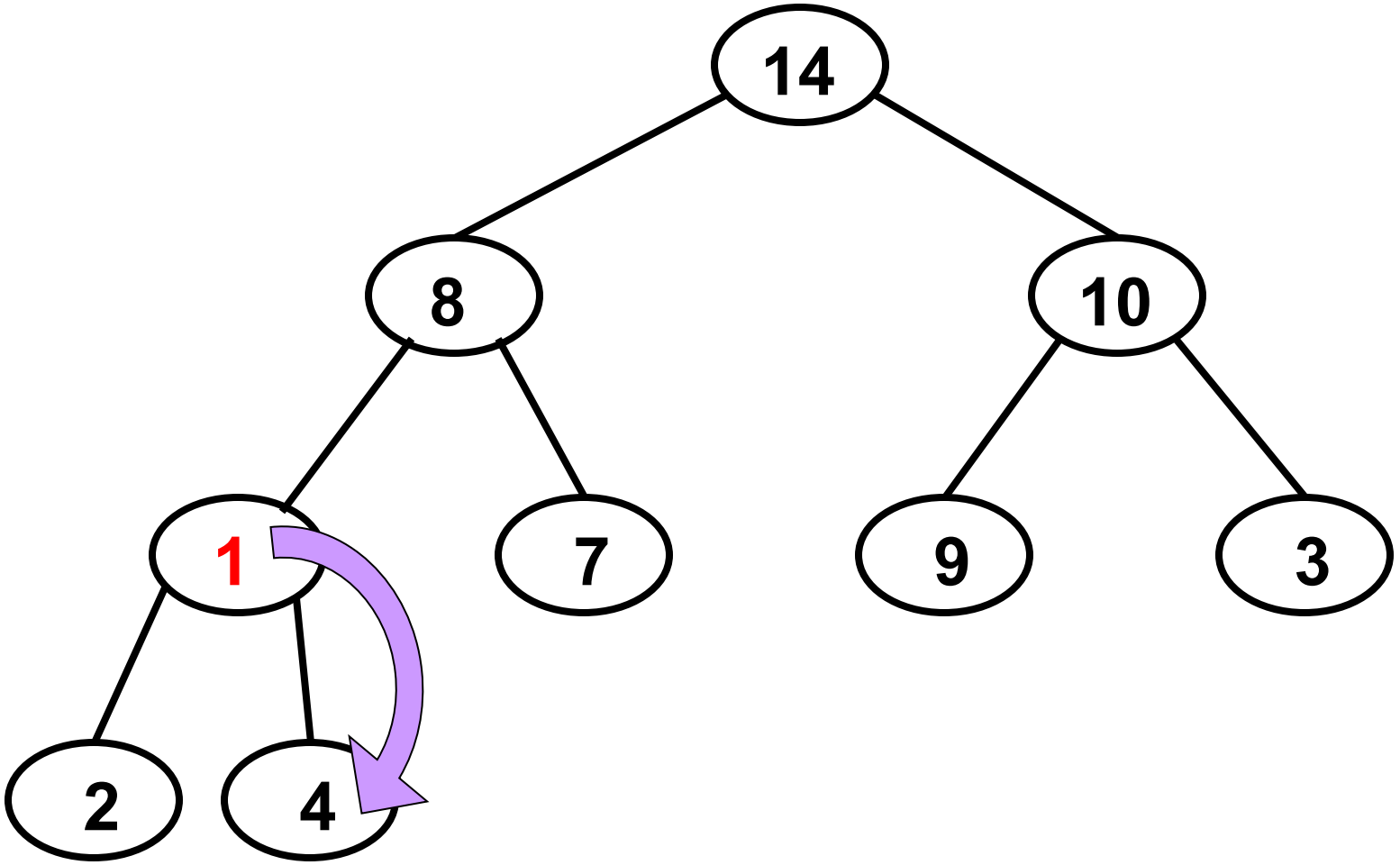
Suppression racine



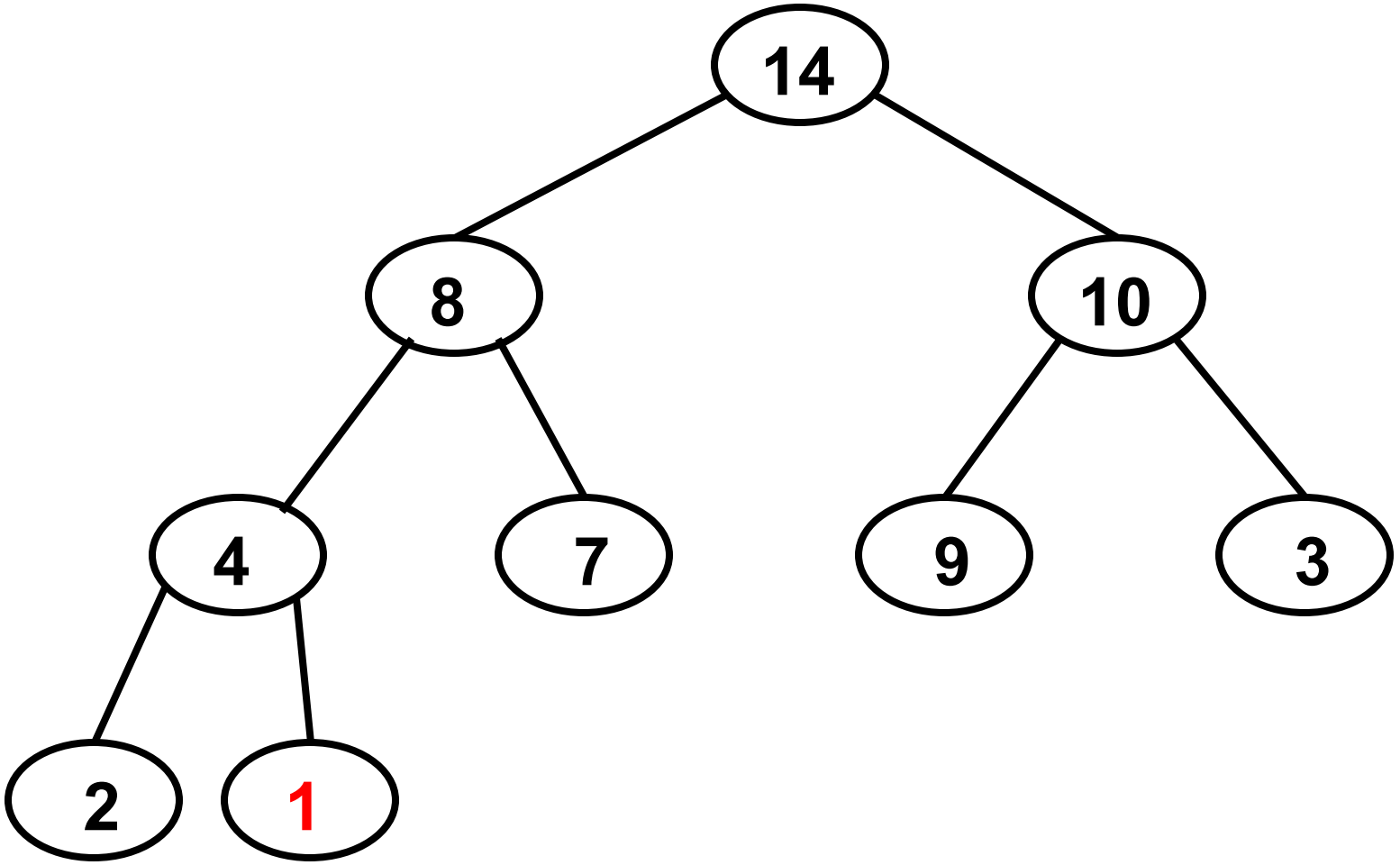
Suppression racine



Suppression racine



Suppression racine



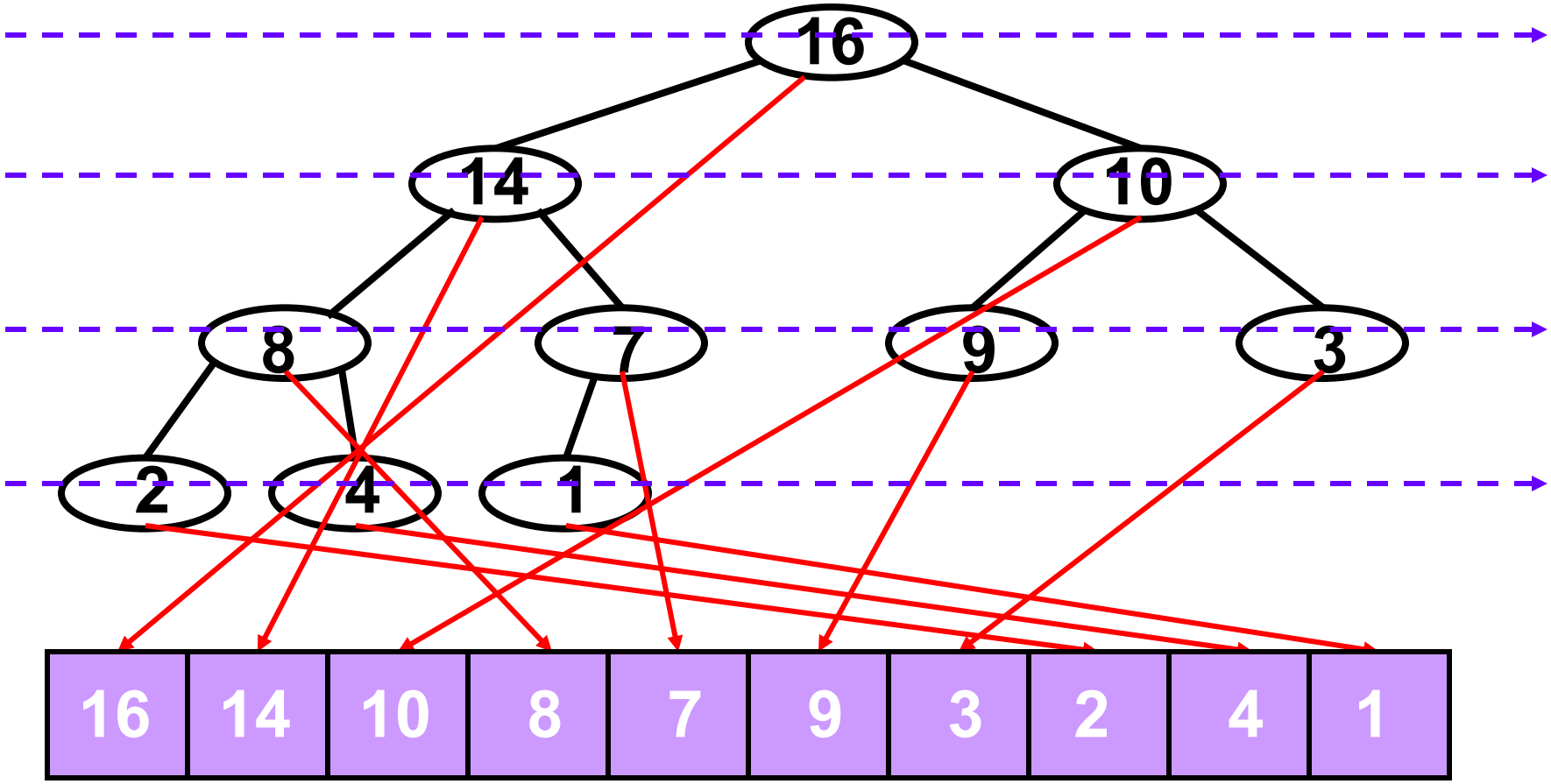
Complexité des tas

- Maximum
  Θ(1)
- Insertion
  Θ(log(n))
- Suppression
  Θ(log(n))
- Excellente solution car complexité bonne **et** valable dans le cas le pire

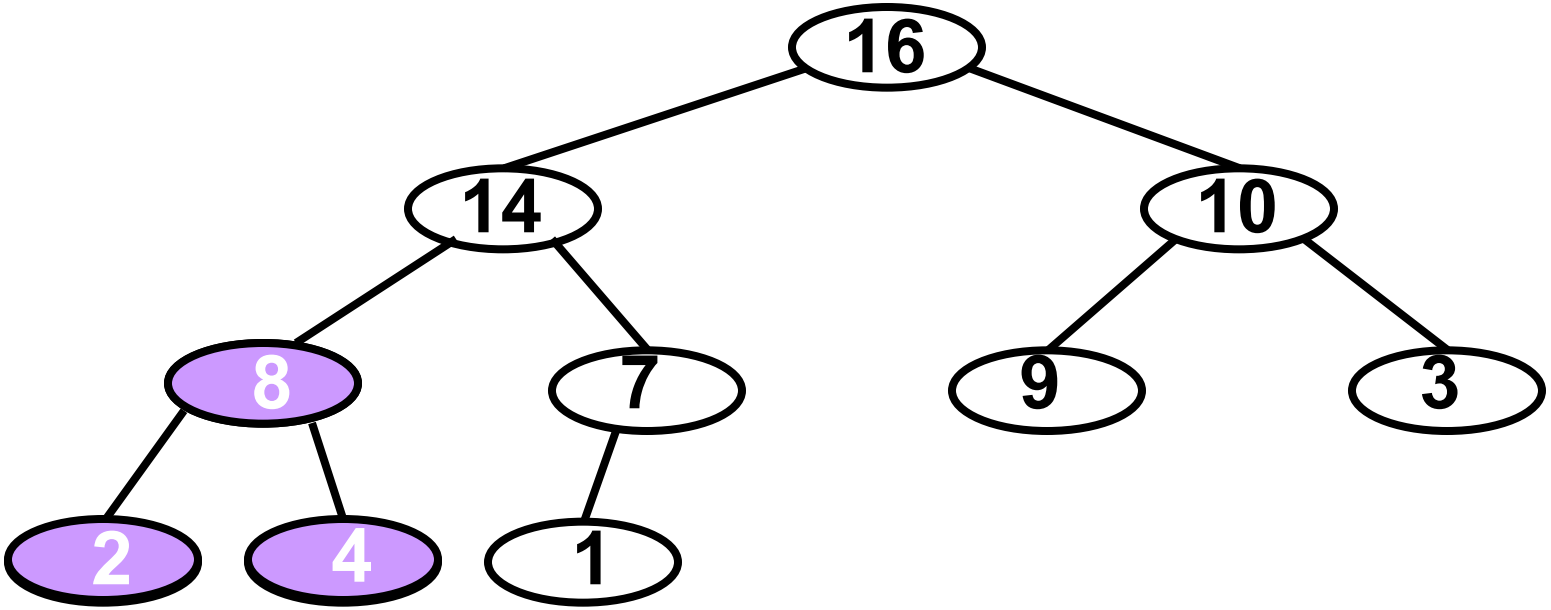
Implémentation des tas

- On utilise un simple **tableau**
- le fils gauche du nœud d'indice ***i*** se trouve dans la case d'indice ***2.i***
- le fils droit du nœud d'indice i se trouve dans la case d'indice ***2.i+1***
- le père du nœud d'indice i se trouve dans la case d'indice ***E(i/2)***

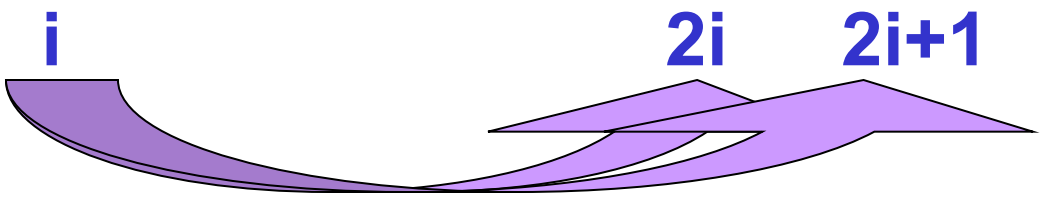
Implémentation des tas



Implémentation des tas



1	2	3	4	5	6	7	8	9	10
16	14	10	8	7	9	3	2	4	1



Tri par tas (heapsort)

- Insérer les ***n*** éléments dans un tas
  O(log(n)) x n
- répéter ***n*** fois
  - rechercher le maximum **Θ(1) x n**
  - supprimer le maximum **O(log(n)) x n**
- **Nouvel algorithme de tri en O(n.log(n))**
  même dans le pire cas
- Implémentation TP semaine prochaine