Module EA4 – Éléments d'Algorithmique

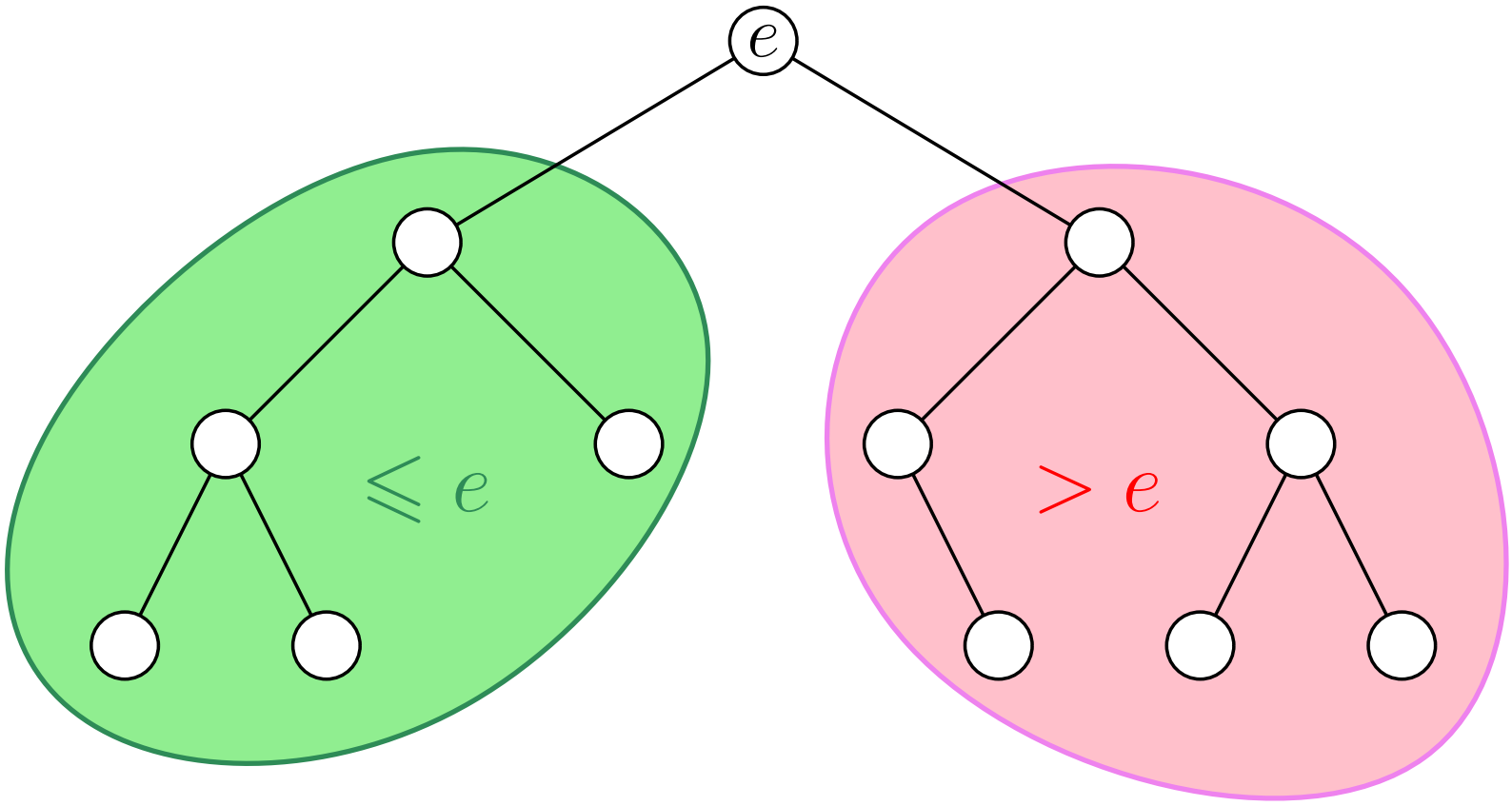
Dominique Poulalhon
dominique.poulalhon@liafa.univ-paris-diderot.fr

Université Paris Diderot
L2 Informatique, Math-Info et EIDD
Année universitaire 2013-2014

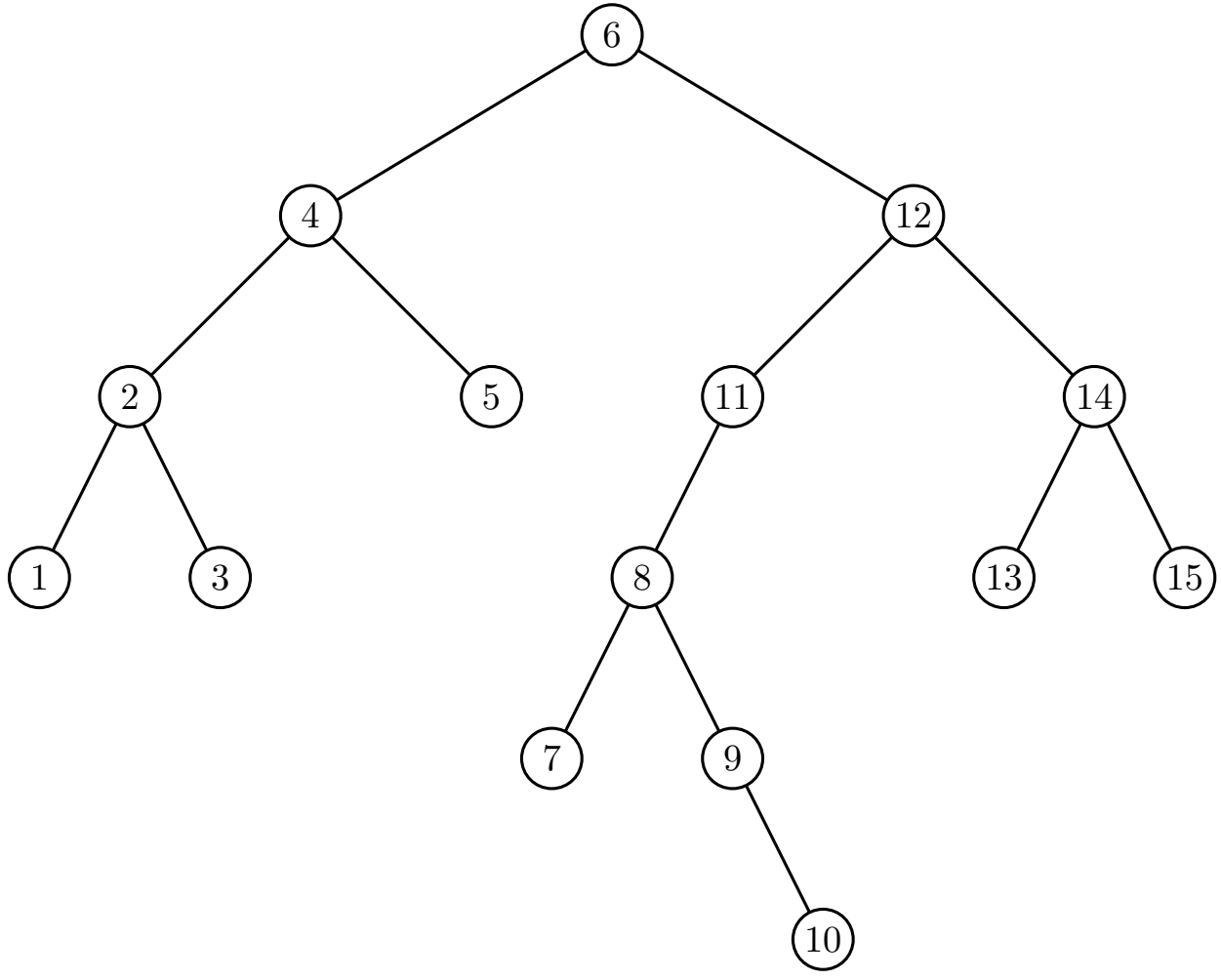
Arbre binaire de recherche (ABR)

en chaque nœud, l'étiquette est comprise entre

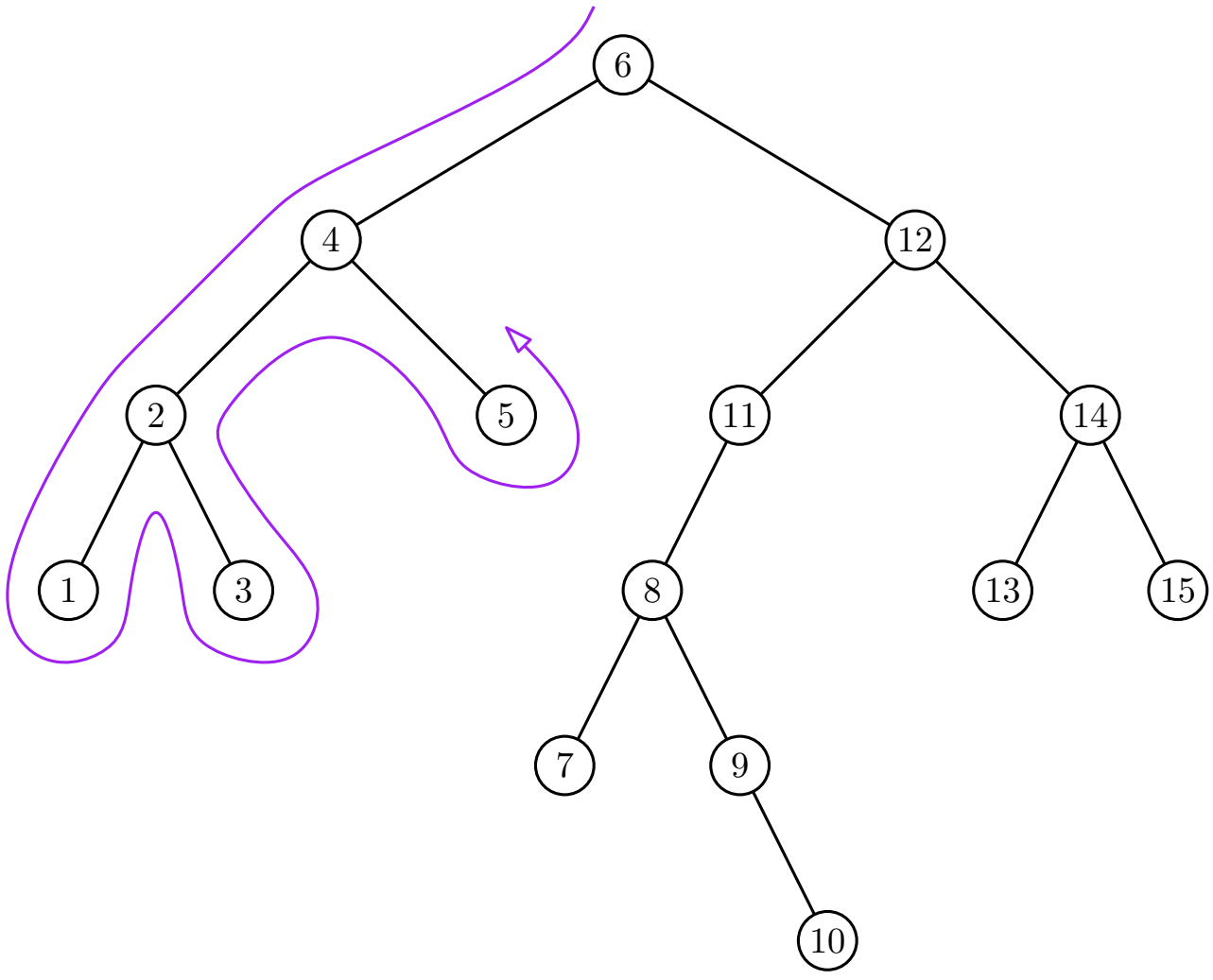
- les étiquettes du sous-arbre gauche (plus petites) et
- celles du sous-arbre droit (plus grandes)



Exemple d'ABR



Exemple d'ABR



Ordre dans un ABR

```
def liste_triee(noeud) :
  res = []
  if noeud != None :
    res = liste_triee(gauche(noeud))
    res += [ etiquette(noeud) ]
    res += liste_triee(droit(noeud))
  return res
```

Ordre dans un ABR

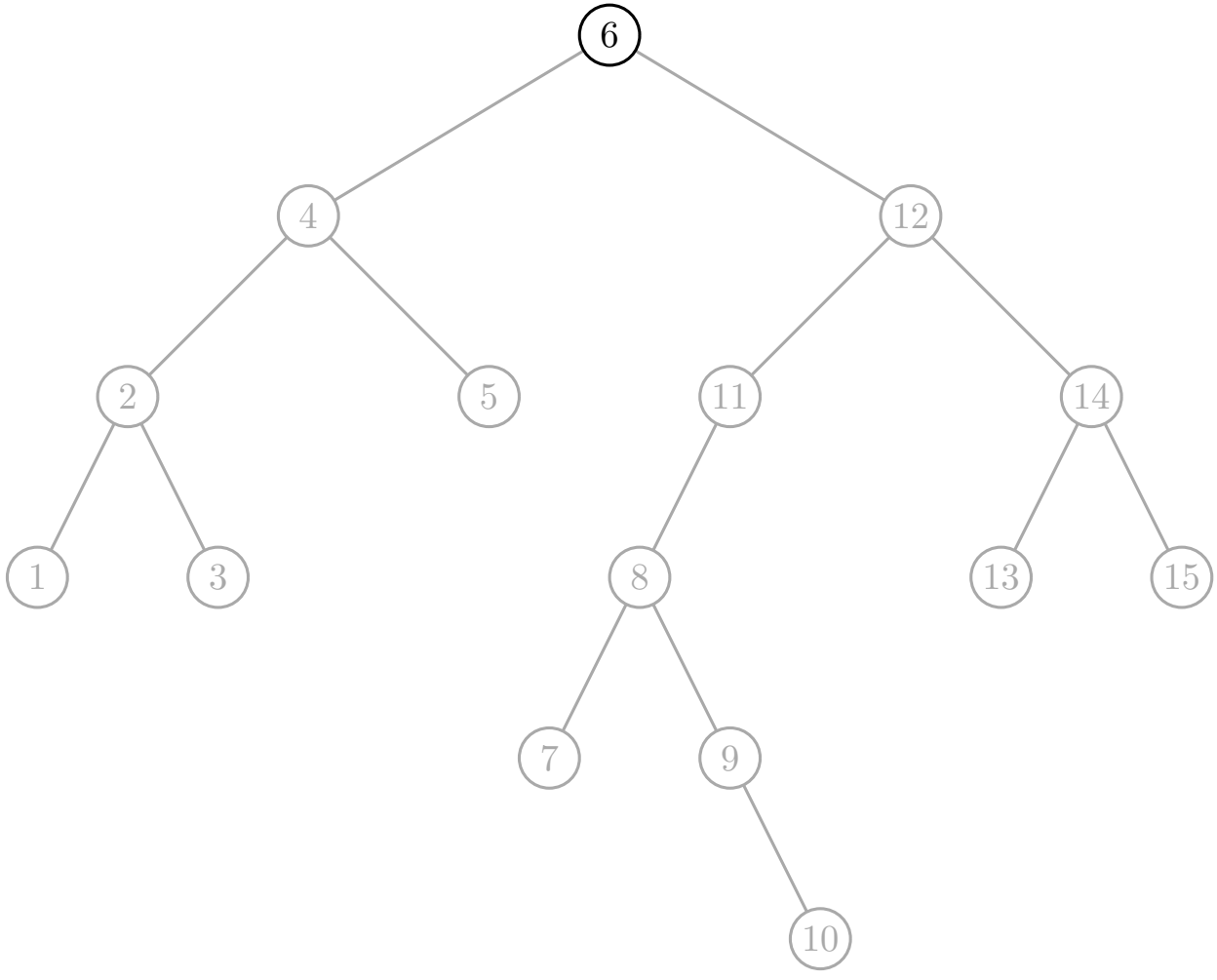
```
def liste_triee(noeud) :
  res = []
  if noeud != None :
    res = liste_triee(gauche(noeud))
    res += [ etiquette(noeud) ]
    res += liste_triee(droit(noeud))
  return res
```

Théorème

le parcours infixe d'un ABR à n nœuds produit la liste triée de ses éléments en temps $\Theta(n)$.

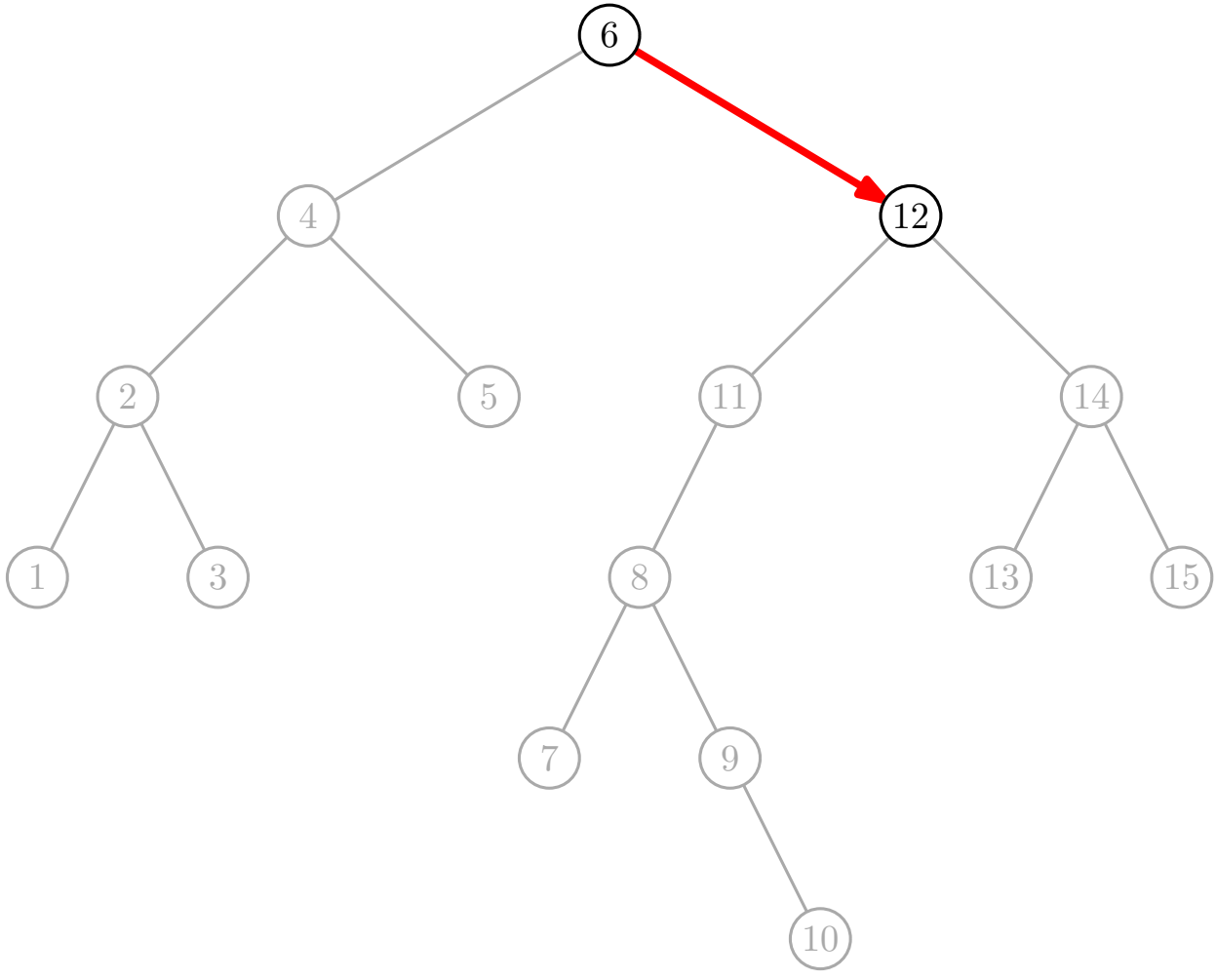
Recherche dans un ABR

Exemple – recherche de 9 :



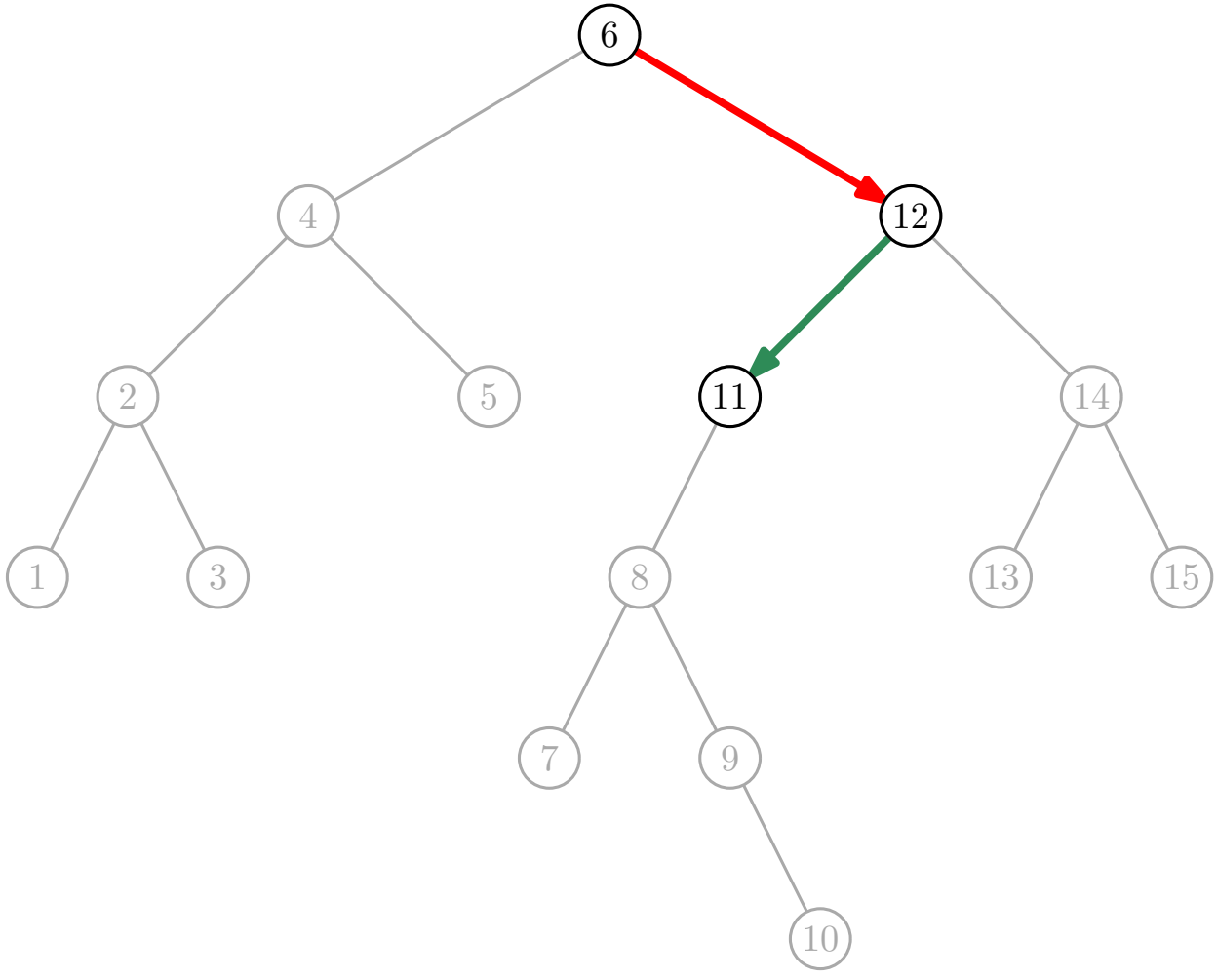
Recherche dans un ABR

Exemple – recherche de 9 :



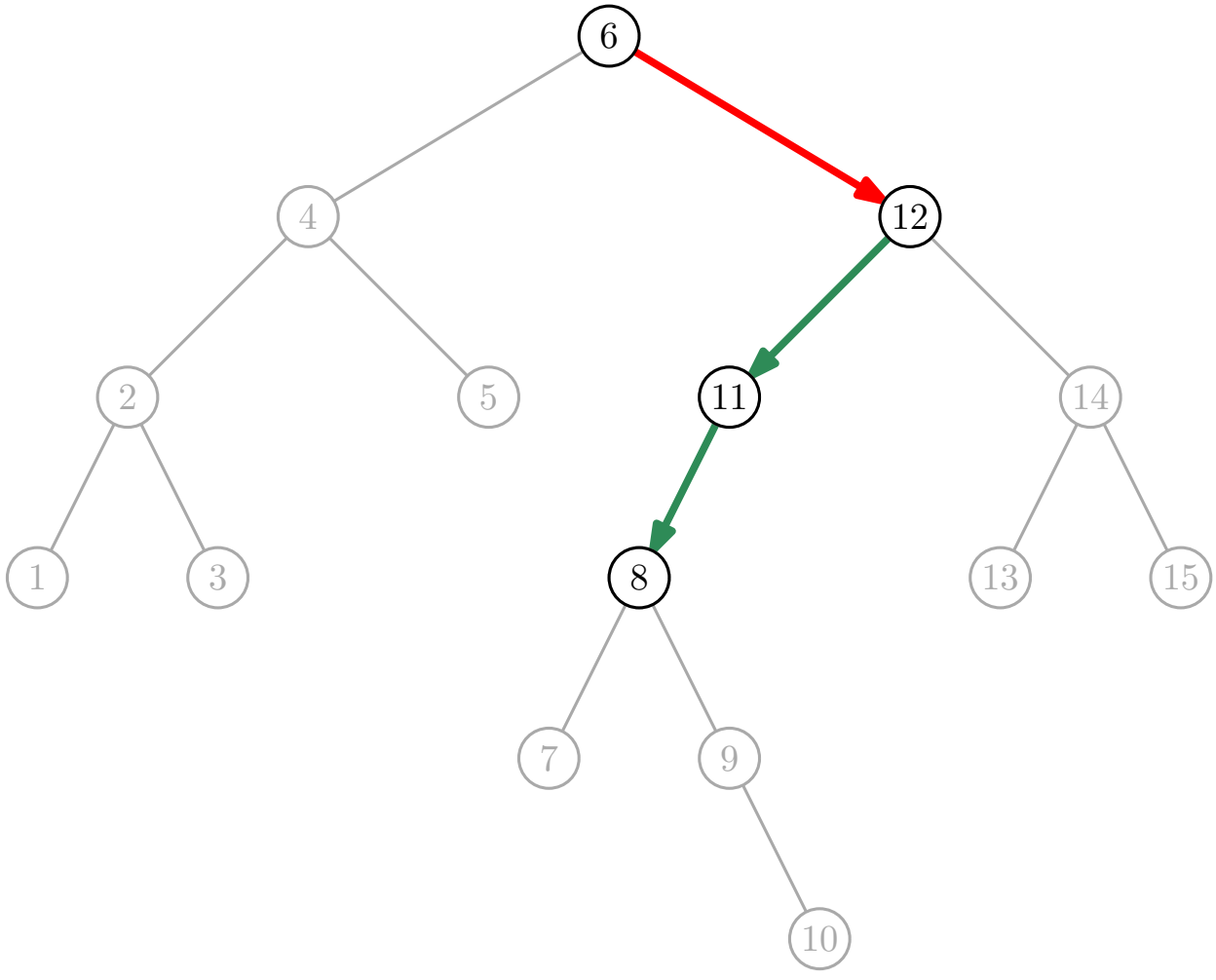
Recherche dans un ABR

Exemple – recherche de 9 :



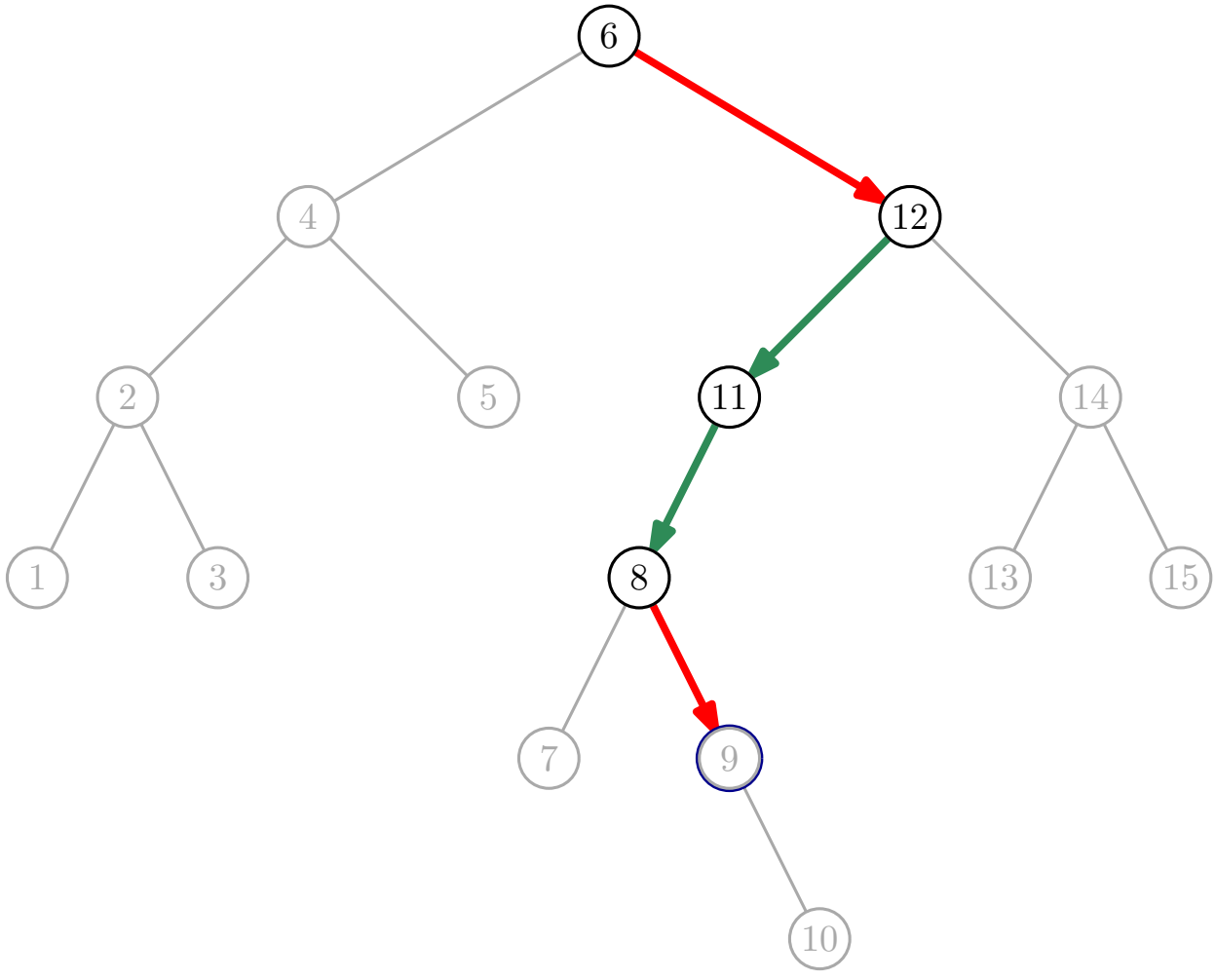
Recherche dans un ABR

Exemple – recherche de 9 :



Recherche dans un ABR

Exemple – recherche de 9 :



Recherche dans un ABR

```
def recherche(noeud, x) : # version récursive
  if noeud == None : return None
  if etiquette(noeud) == x : return noeud
  if etiquette(noeud) > x : return recherche(gauche(noeud))
  return recherche(droit(noeud))
```

Recherche dans un ABR

```
def recherche(noeud, x) : # version récursive
  if noeud == None : return None
  if etiquette(noeud) == x : return noeud
  if etiquette(noeud) > x : return recherche(gauche(noeud))
  return recherche(droit(noeud))
```

Théorème

`recherche(r, x)` effectue la recherche d'un élément `x` dans l'ABR de racine `r` en temps $\Theta(h)$ au pire, où h est la hauteur de l'ABR.

Cas particuliers : minimum/maximum

```
def minimum(noeud) : # version récursive
  if gauche(noeud) == None : return noeud
  return minimum(gauche(noeud))
```

Cas particuliers : minimum/maximum

```
def minimum(noeud) : # version récursive
  if gauche(noeud) == None : return noeud
  return minimum(gauche(noeud))

def minimum(noeud) : # version itérative
  while gauche(noeud) != None :
    noeud = gauche(noeud)
  return noeud
```

Cas particuliers : minimum/maximum

```
def minimum(noeud) : # version récursive
  if gauche(noeud) == None : return noeud
  return minimum(gauche(noeud))

def minimum(noeud) : # version itérative
  while gauche(noeud) != None :
    noeud = gauche(noeud)
  return noeud
```

Théorème

minimum(r) détermine le plus petit élément dans l'ABR de racine r en temps $O(h)$*, où* h *est la hauteur de l'ABR.*

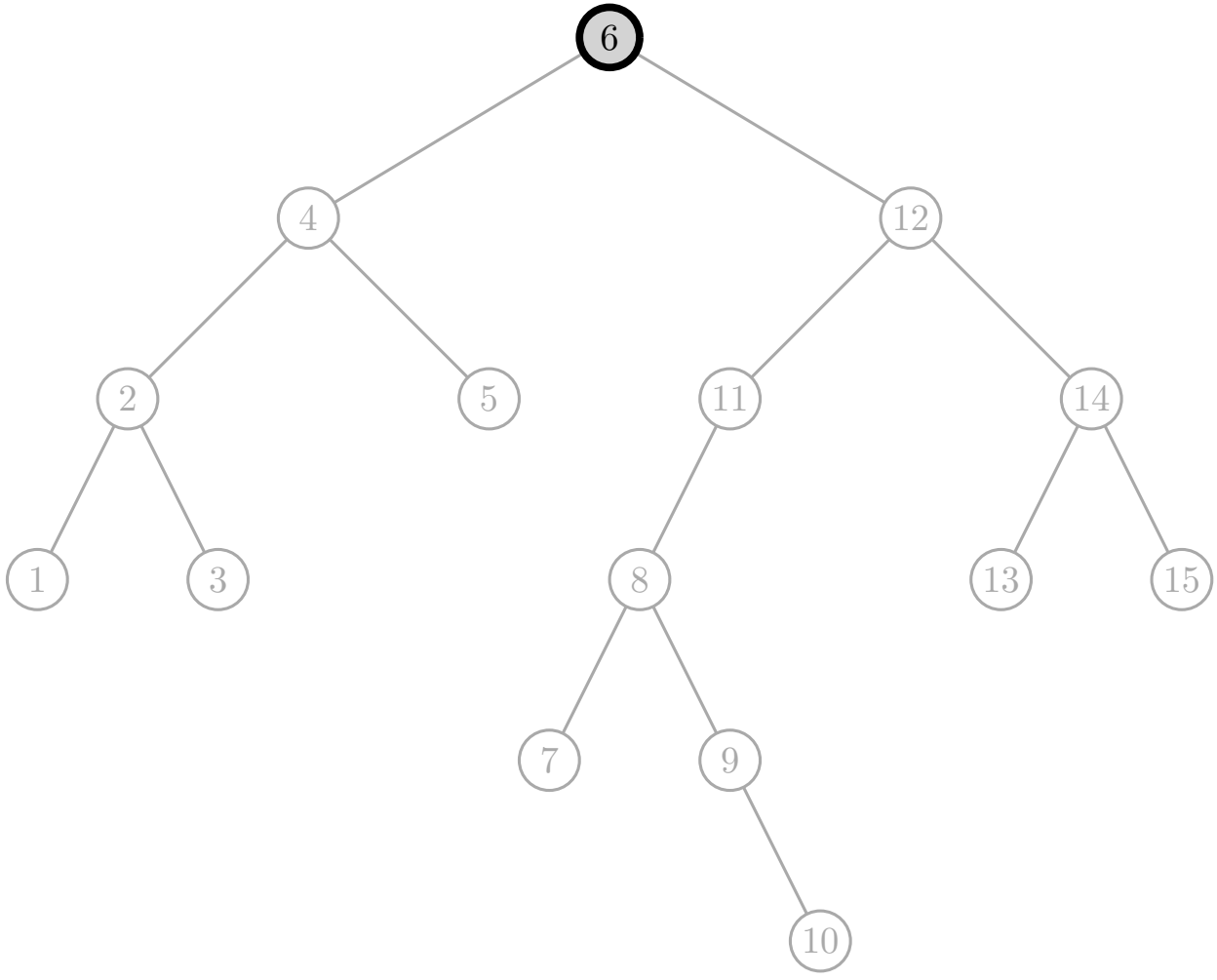
Successeur d'un élément

successeur(n)

étant donné un nœud `n` d'un ABR, d'étiquette `e`, déterminer le nœud de l'arbre ayant la plus petite étiquette supérieure à `e`.

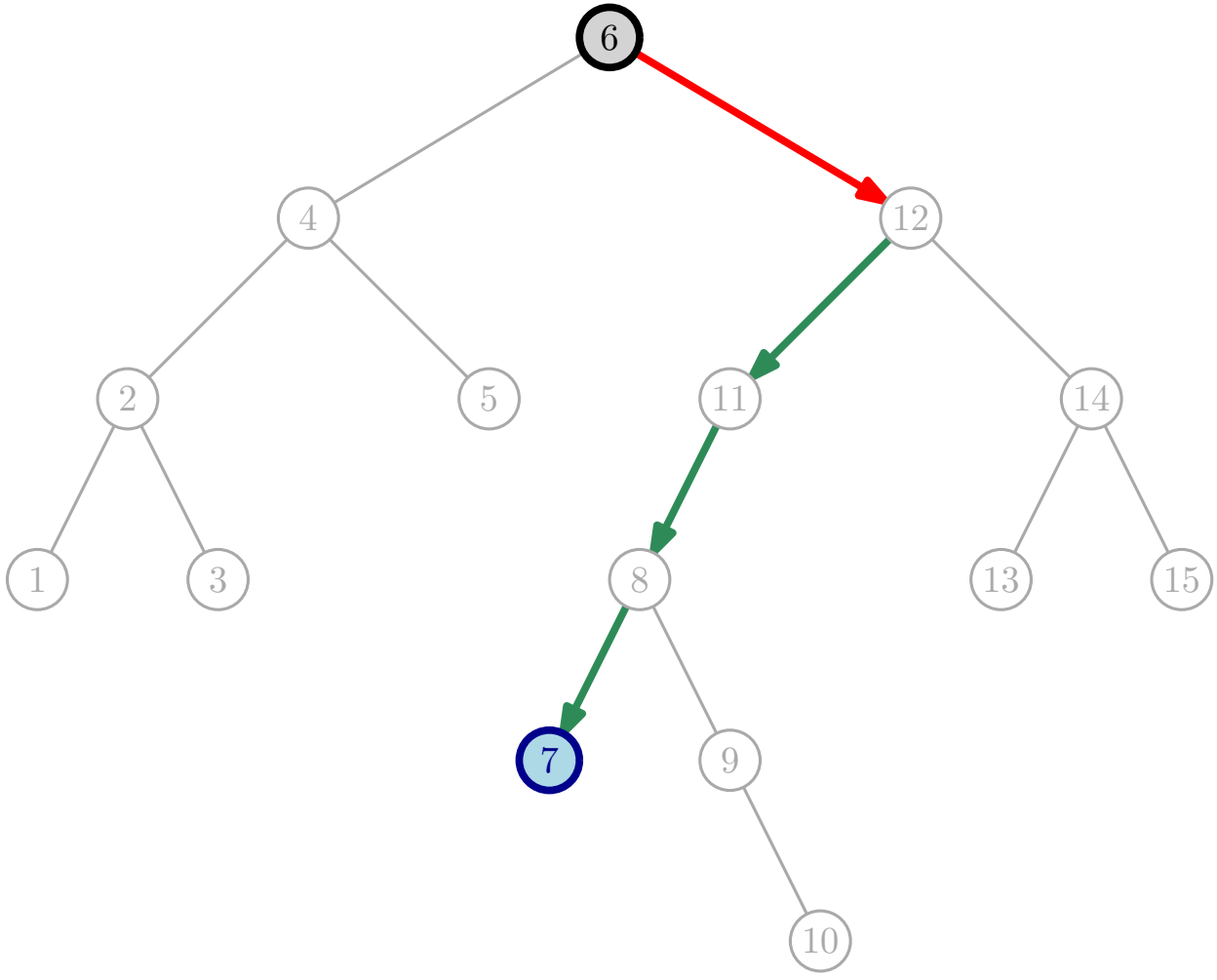
Successeur d'un élément

si le nœud a un fils droit



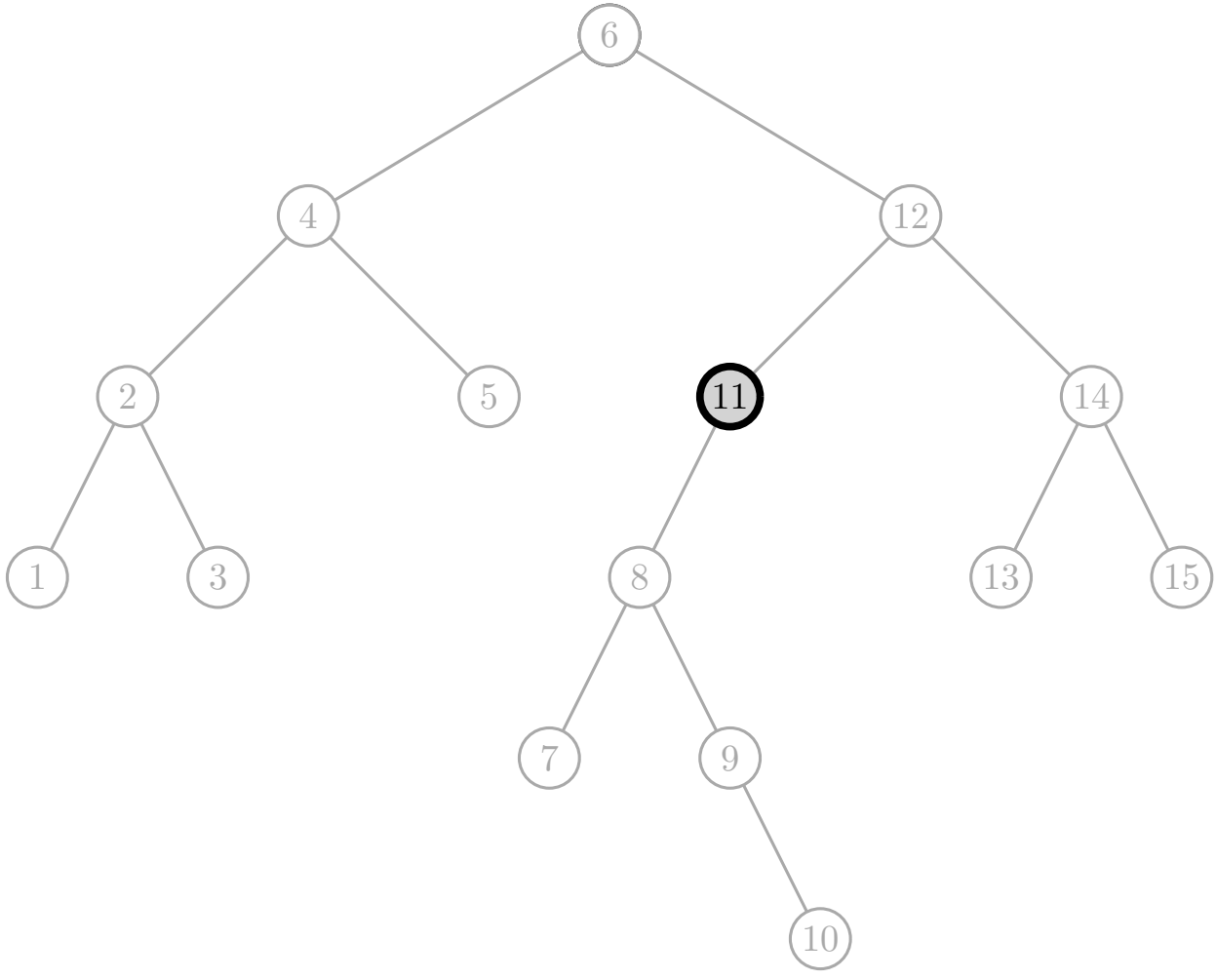
Successeur d'un élément

si le nœud a un fils droit



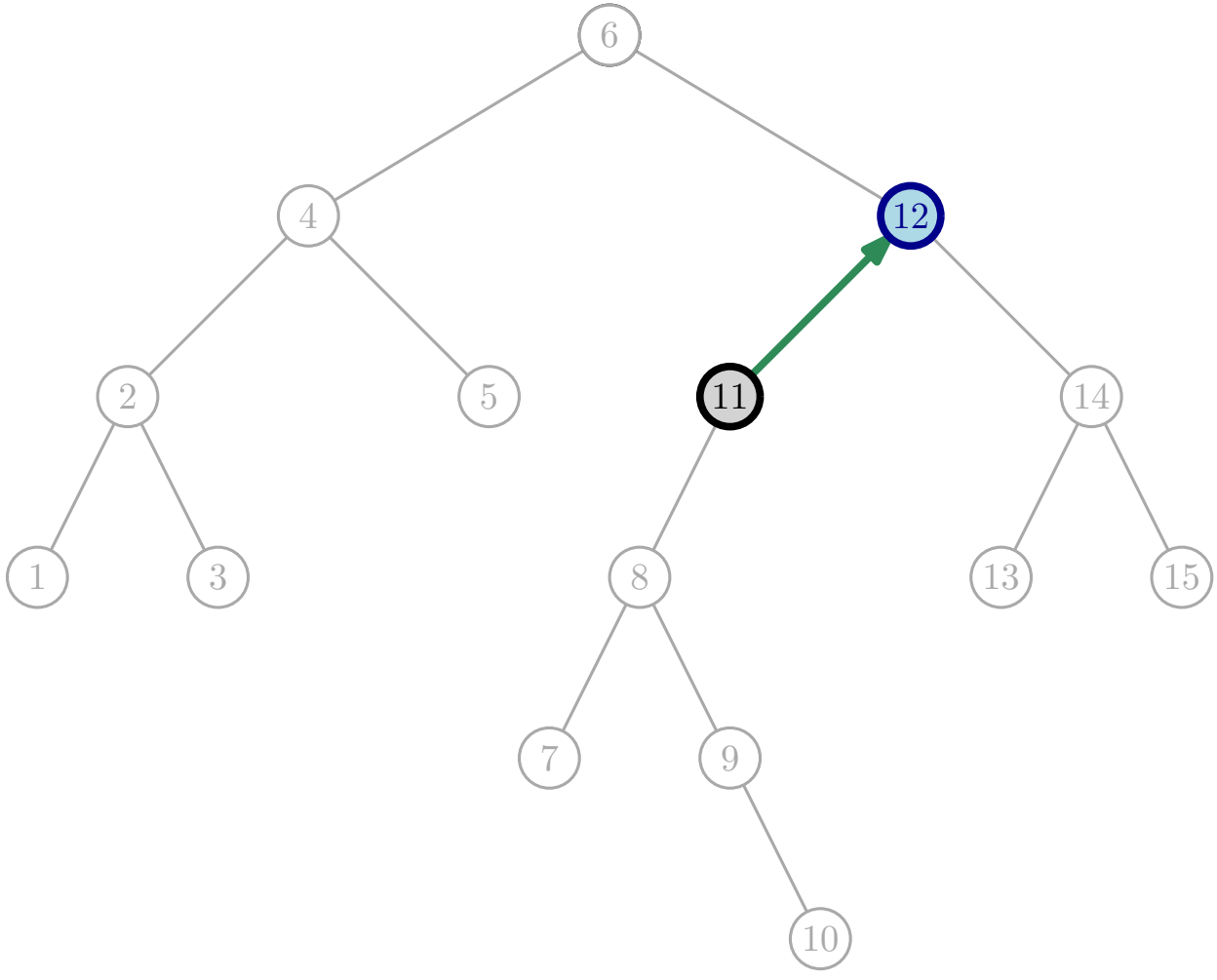
Successeur d'un élément

si le nœud n'a pas de fils droit



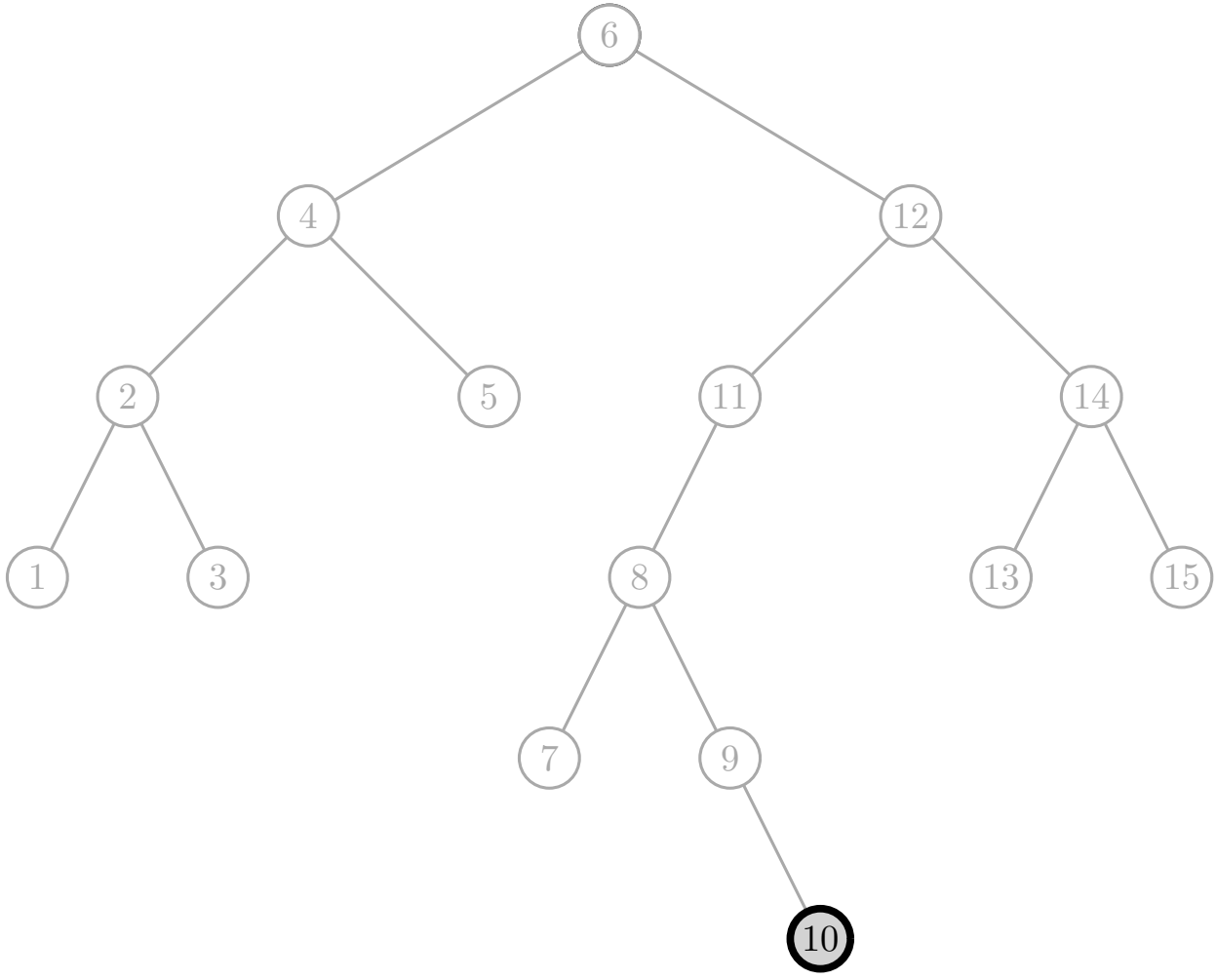
Successeur d'un élément

si le nœud n'a pas de fils droit



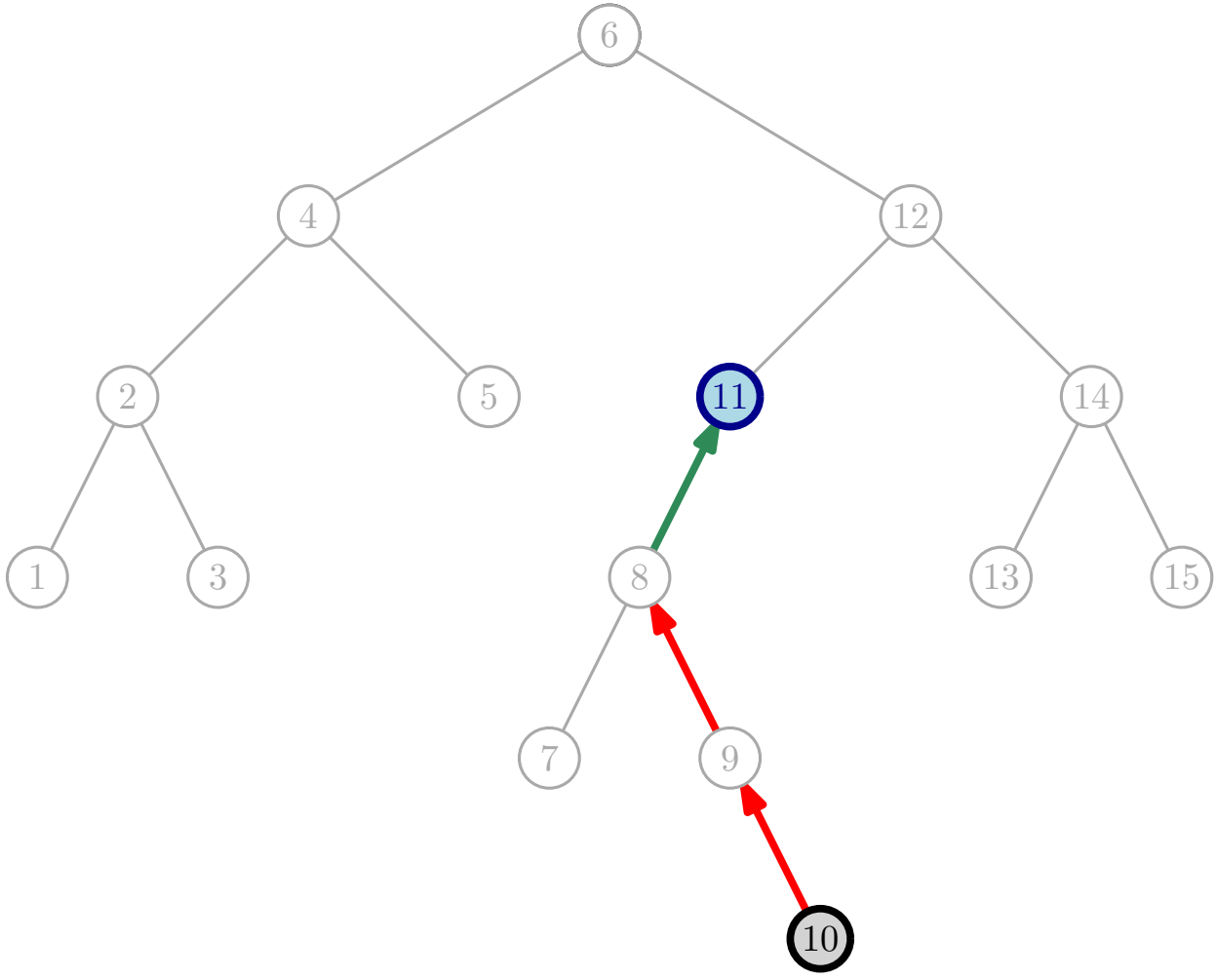
Successeur d'un élément

si le nœud n'a pas de fils droit



Successeur d'un élément

si le nœud n'a pas de fils droit



Successeur d'un élément

```
def successeur(noeud) :
  if droit(noeud) != None :
    return minimum(droit(noeud))
  while pere(noeud) != None and est_fils_droit(noeud) :
    noeud = pere(noeud)
  return pere(noeud)
```

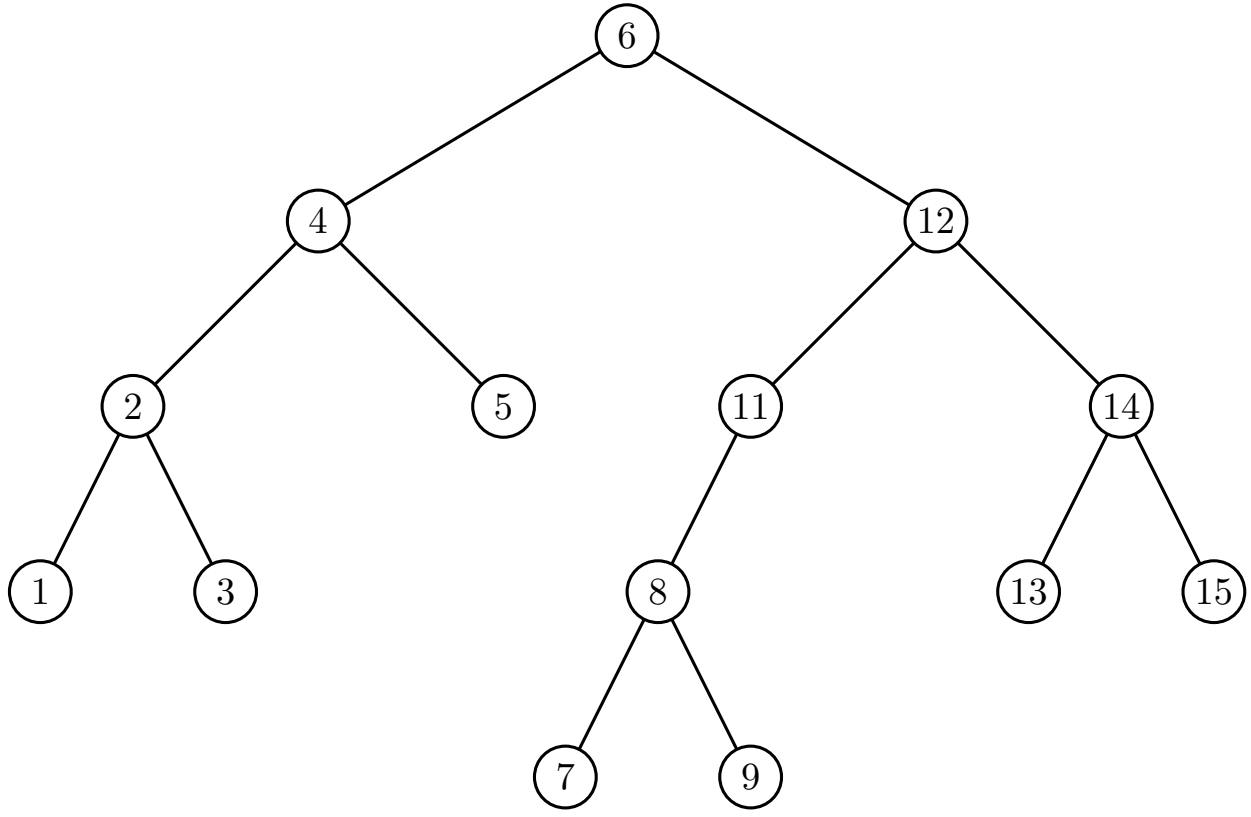
Successeur d'un élément

```
def successeur(noeud) :
  if droit(noeud) != None :
    return minimum(droit(noeud))
  while pere(noeud) != None and est_fils_droit(noeud) :
    noeud = pere(noeud)
  return pere(noeud)
```

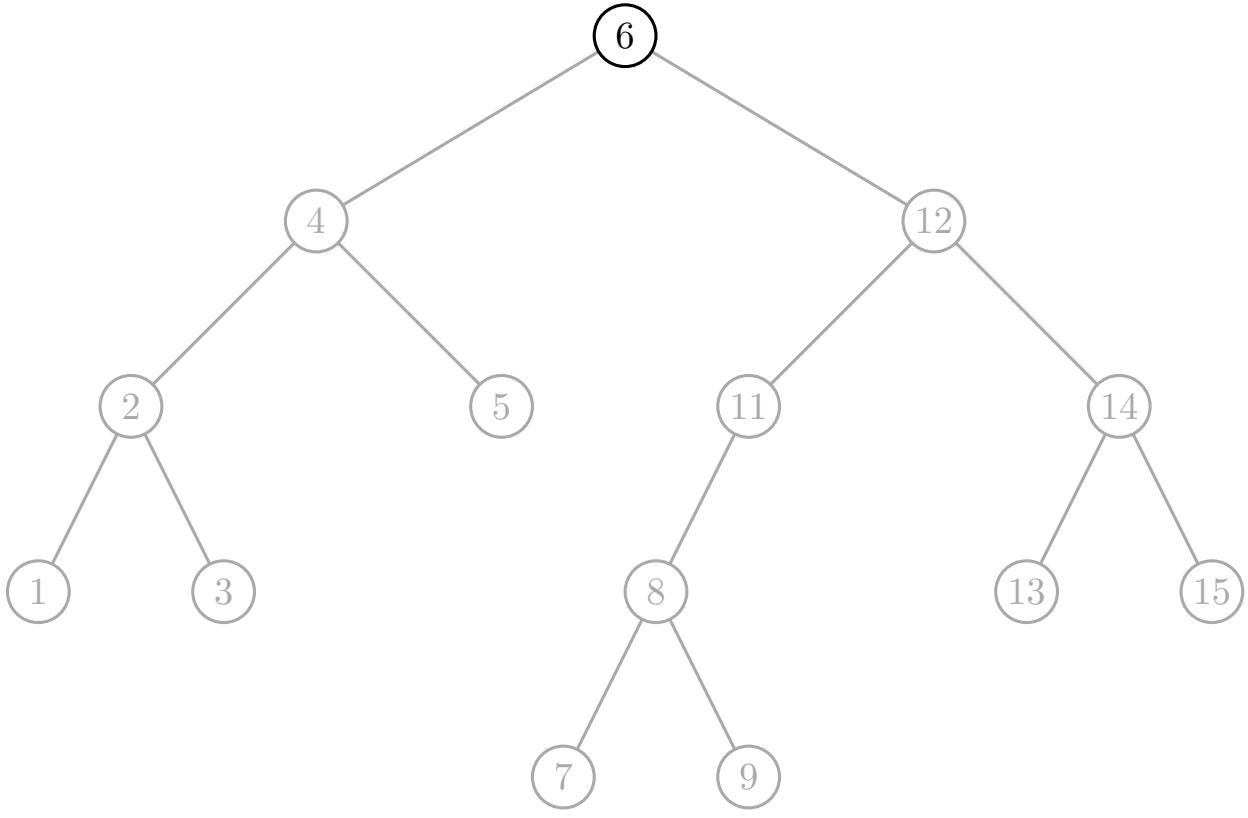
Théorème

successeur(noeud) détermine le successeur d'un noeud d'un ABR en temps $\Theta(h)$ au pire, où h est la hauteur de l'ABR.

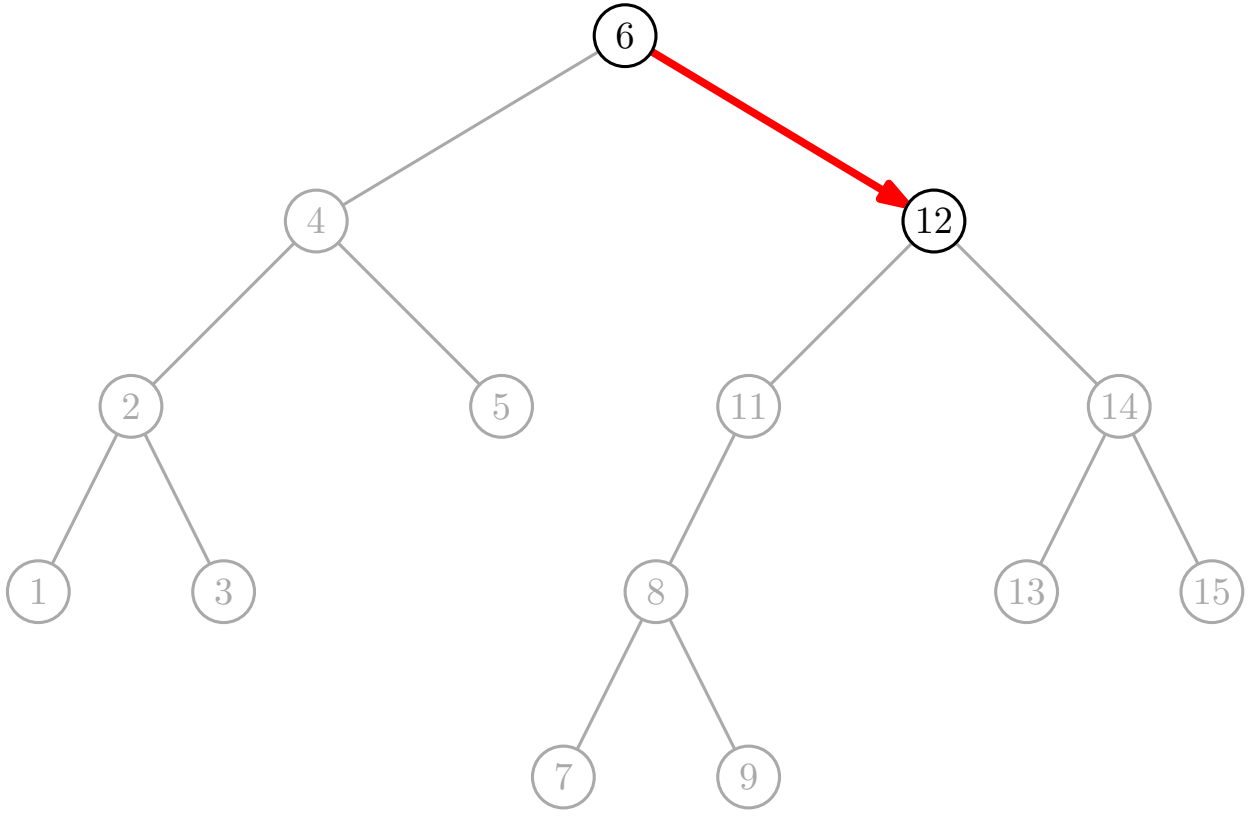
Insertion dans un ABR



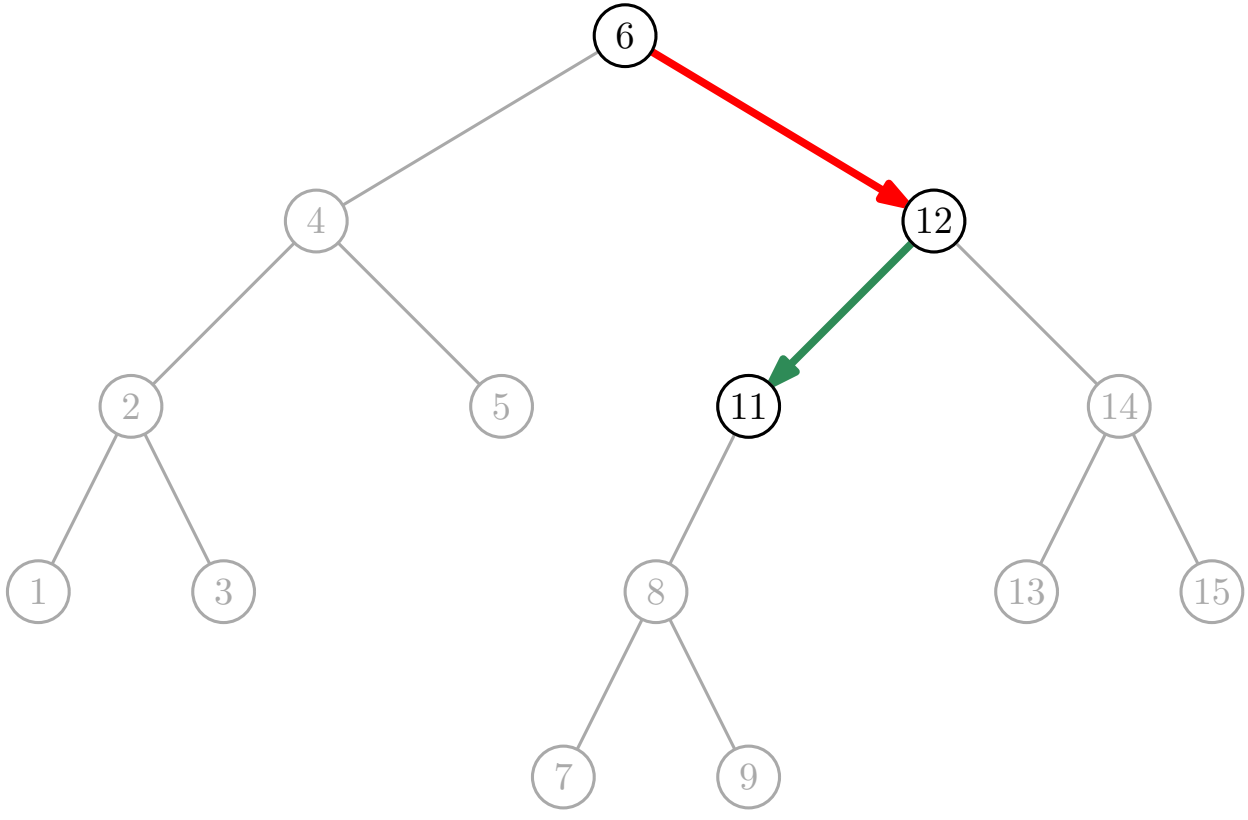
Insertion dans un ABR



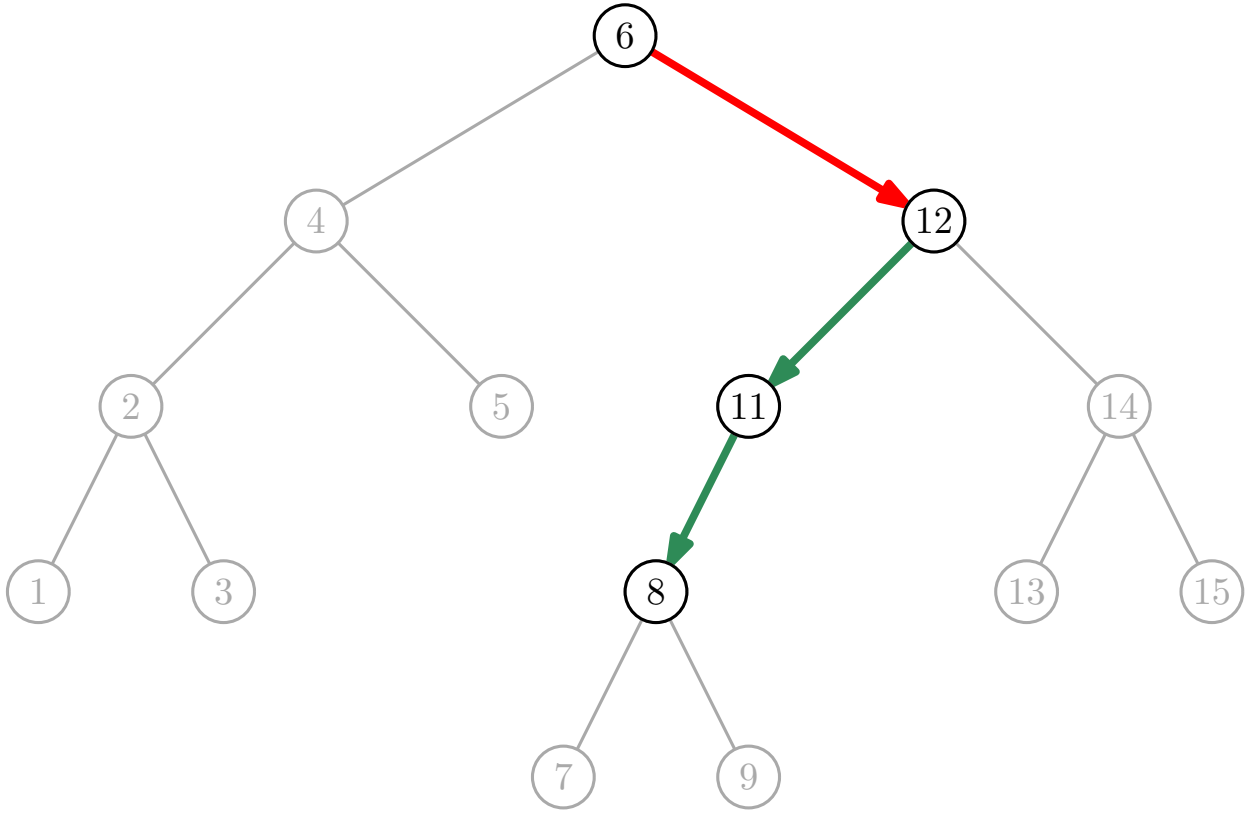
Insertion dans un ABR



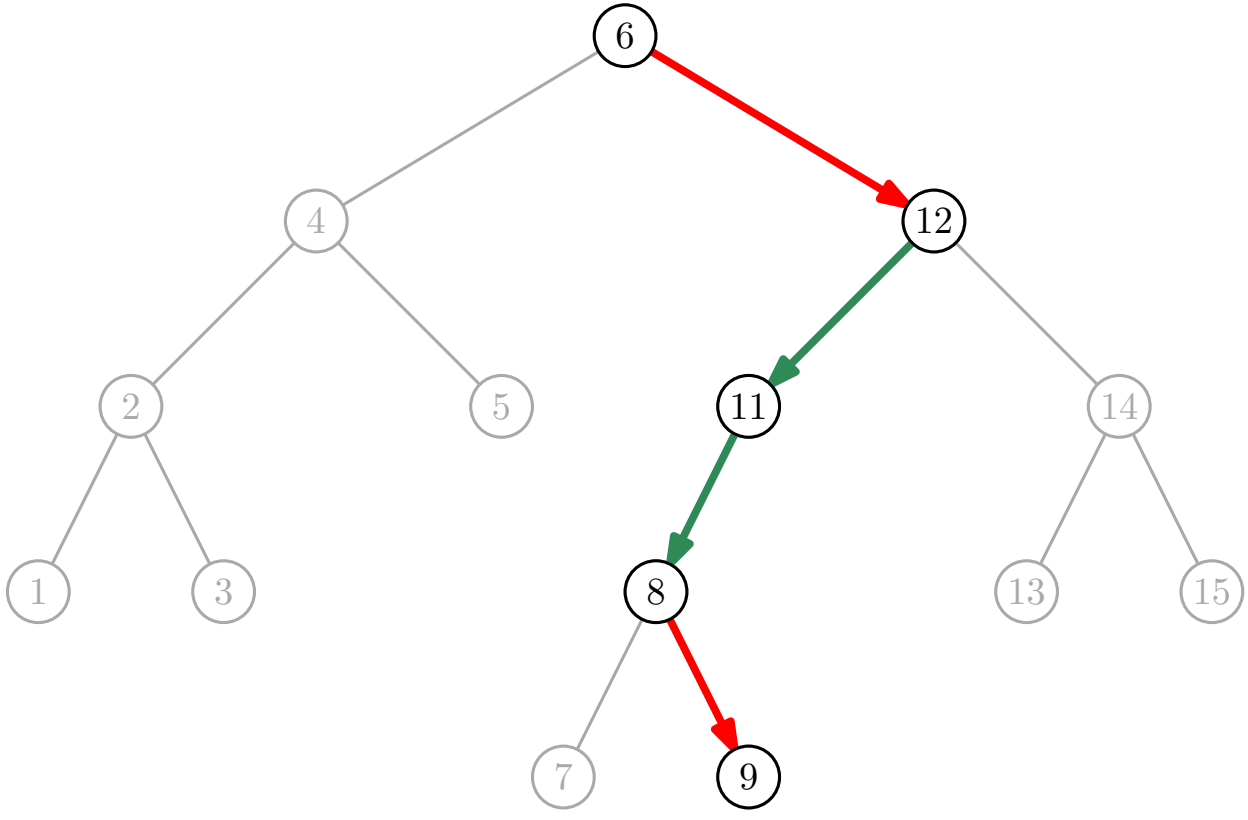
Insertion dans un ABR



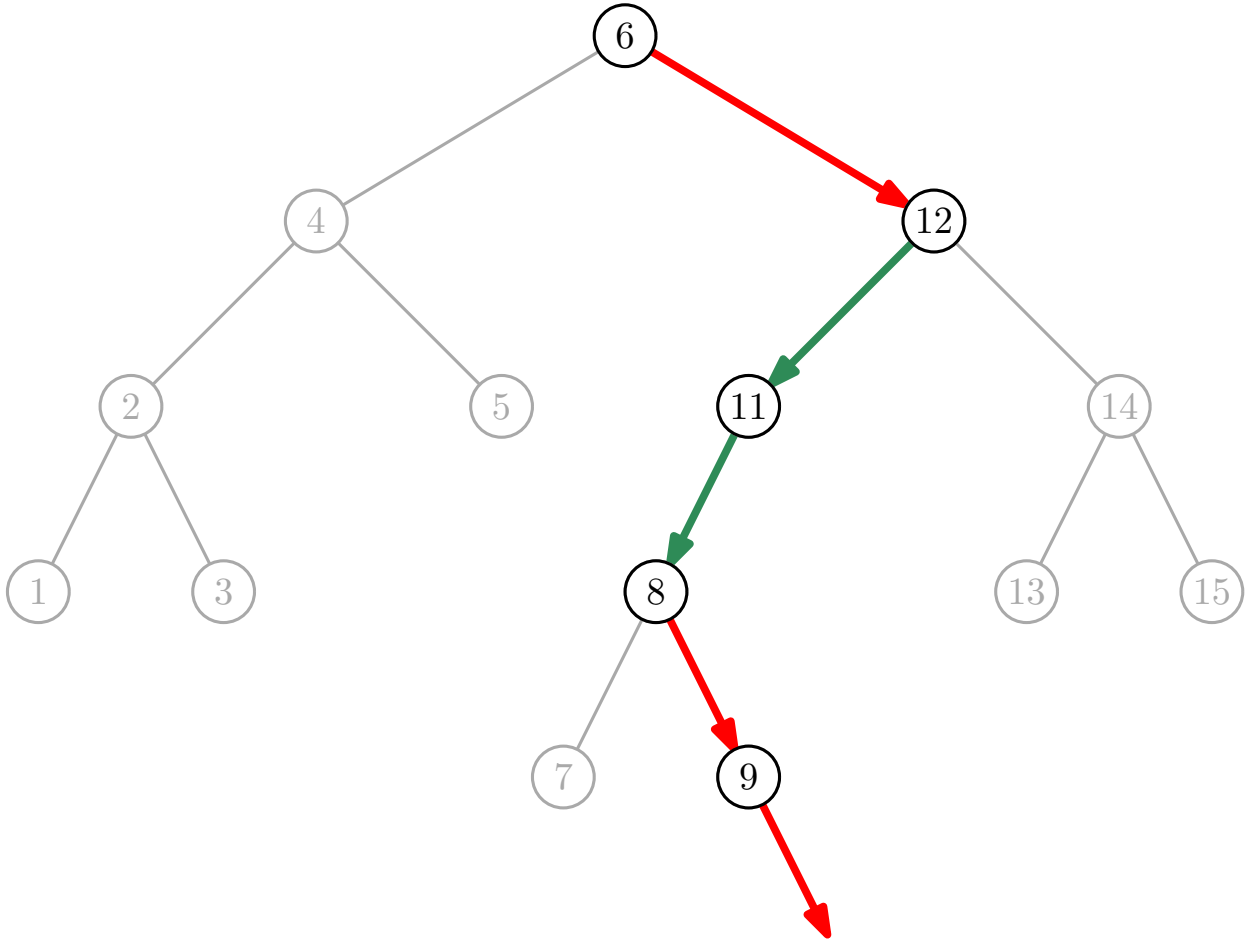
Insertion dans un ABR



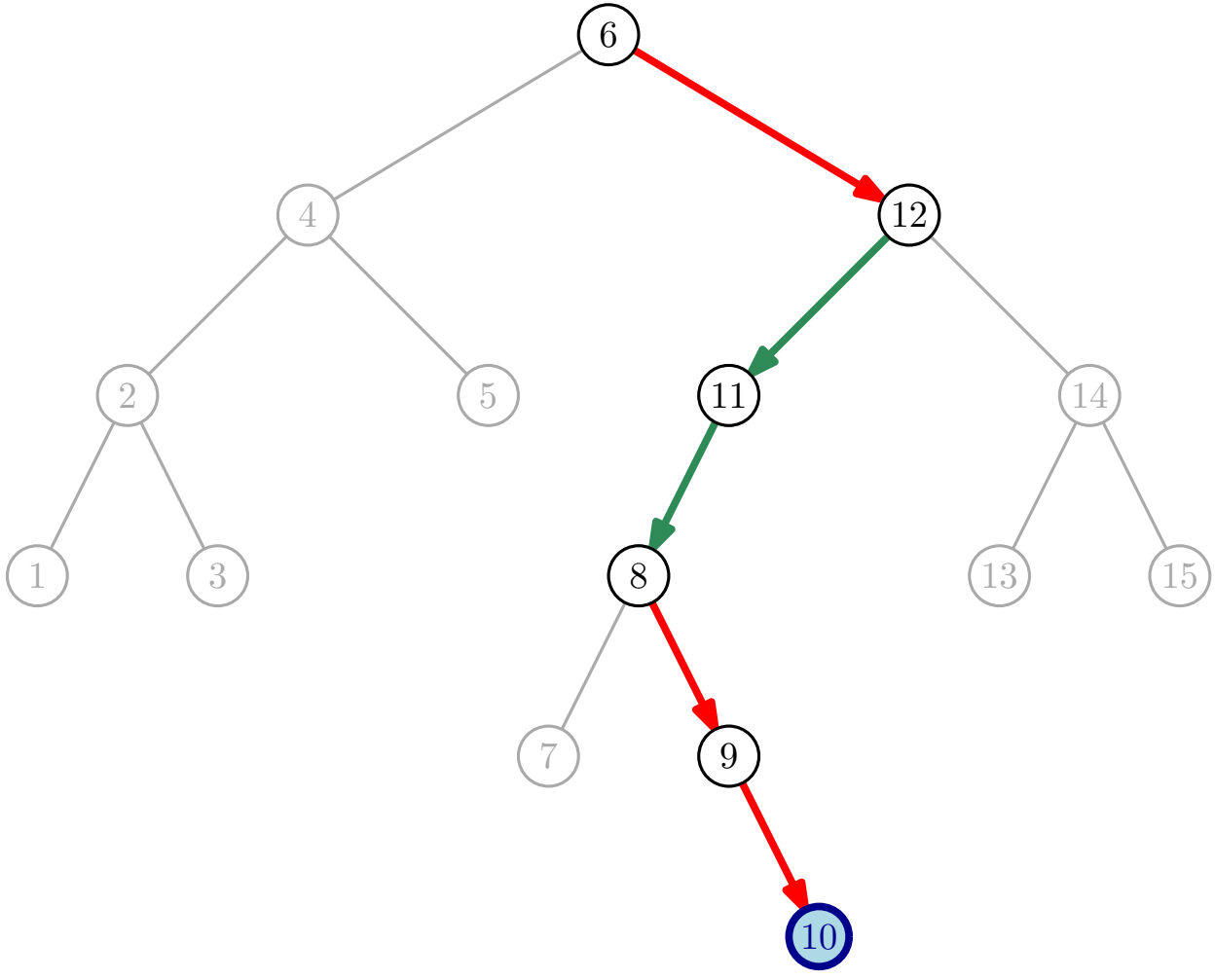
Insertion dans un ABR



Insertion dans un ABR



Insertion dans un ABR

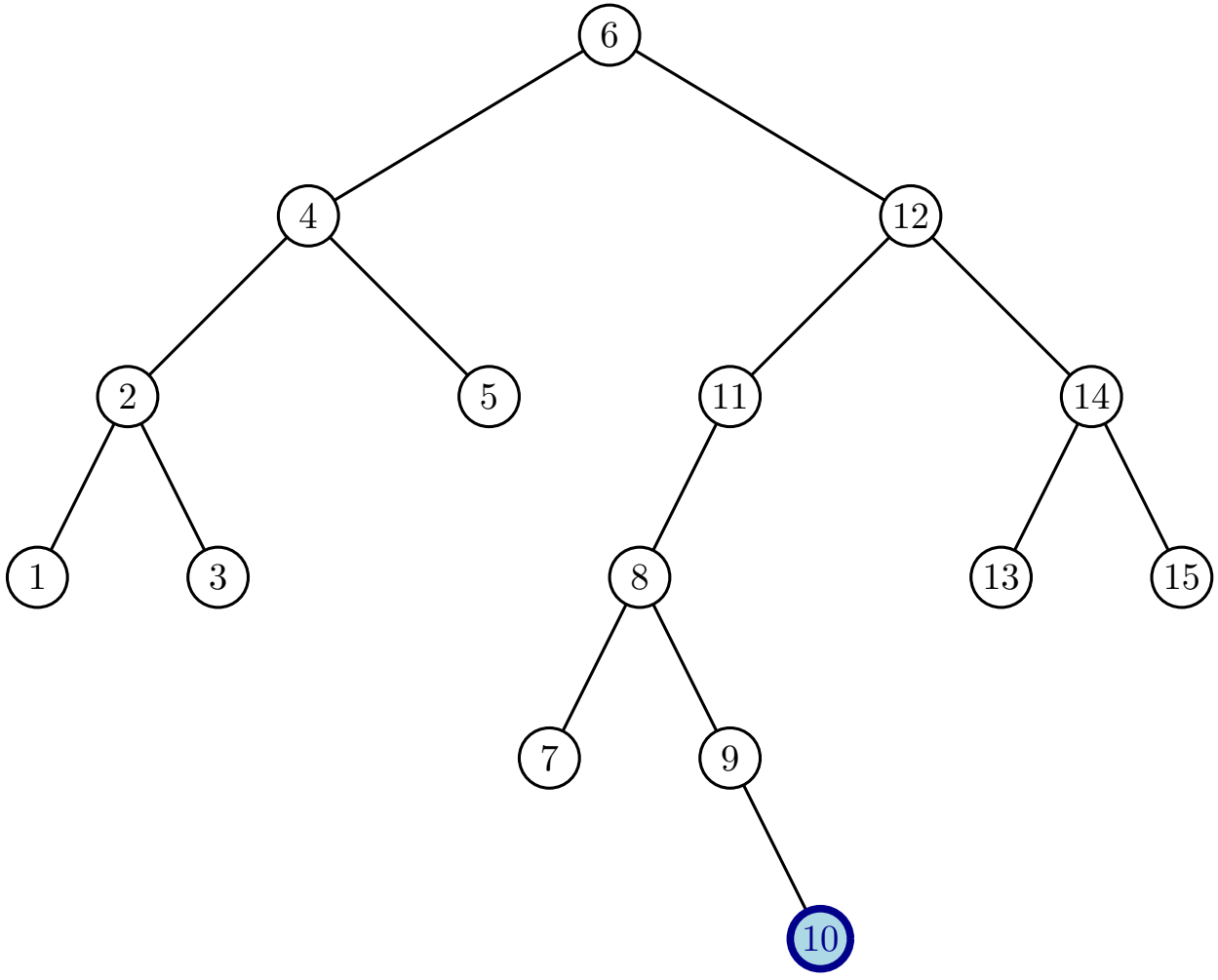


Théorème

L'insertion d'un nouvel élément dans un ABR de hauteur h *peut se faire en temps* $\Theta(h)$ *au pire.*

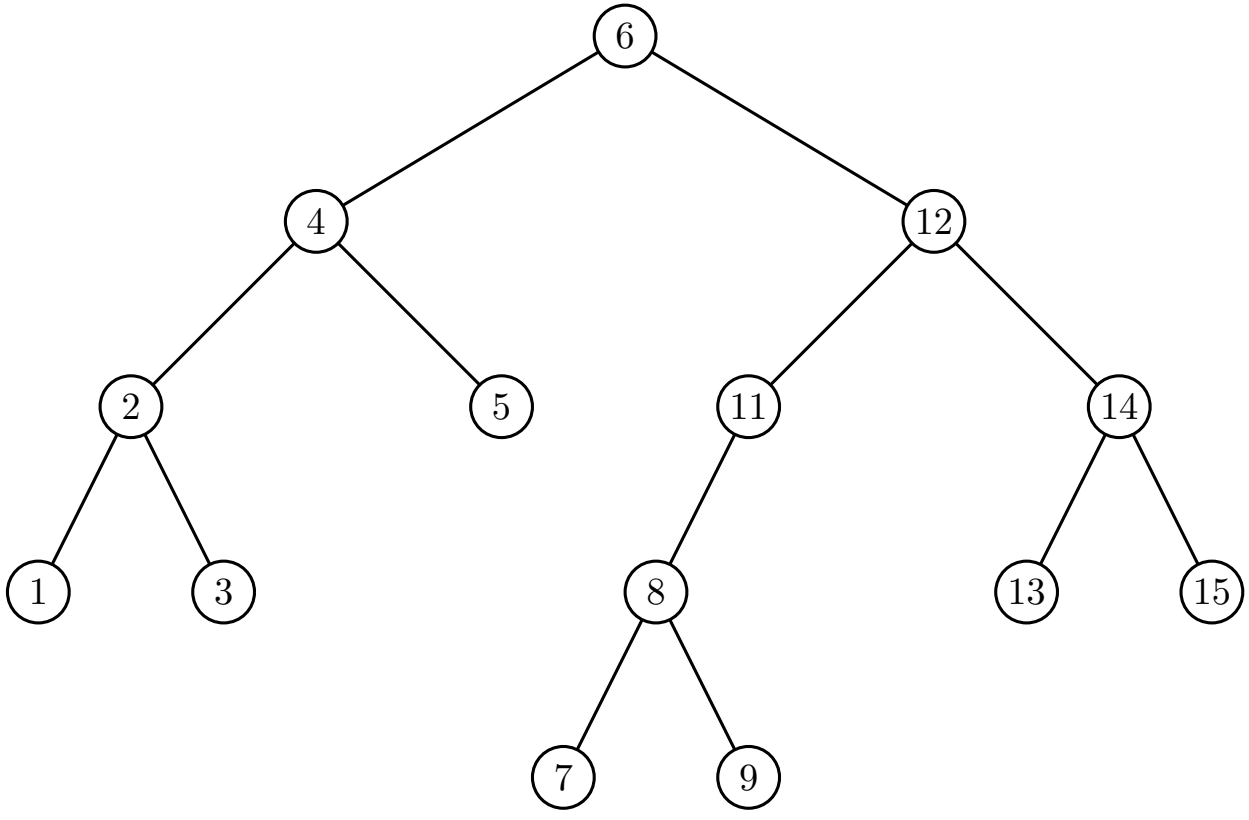
Suppression dans un ABR

si le nœud à supprimer n'a pas d'enfant



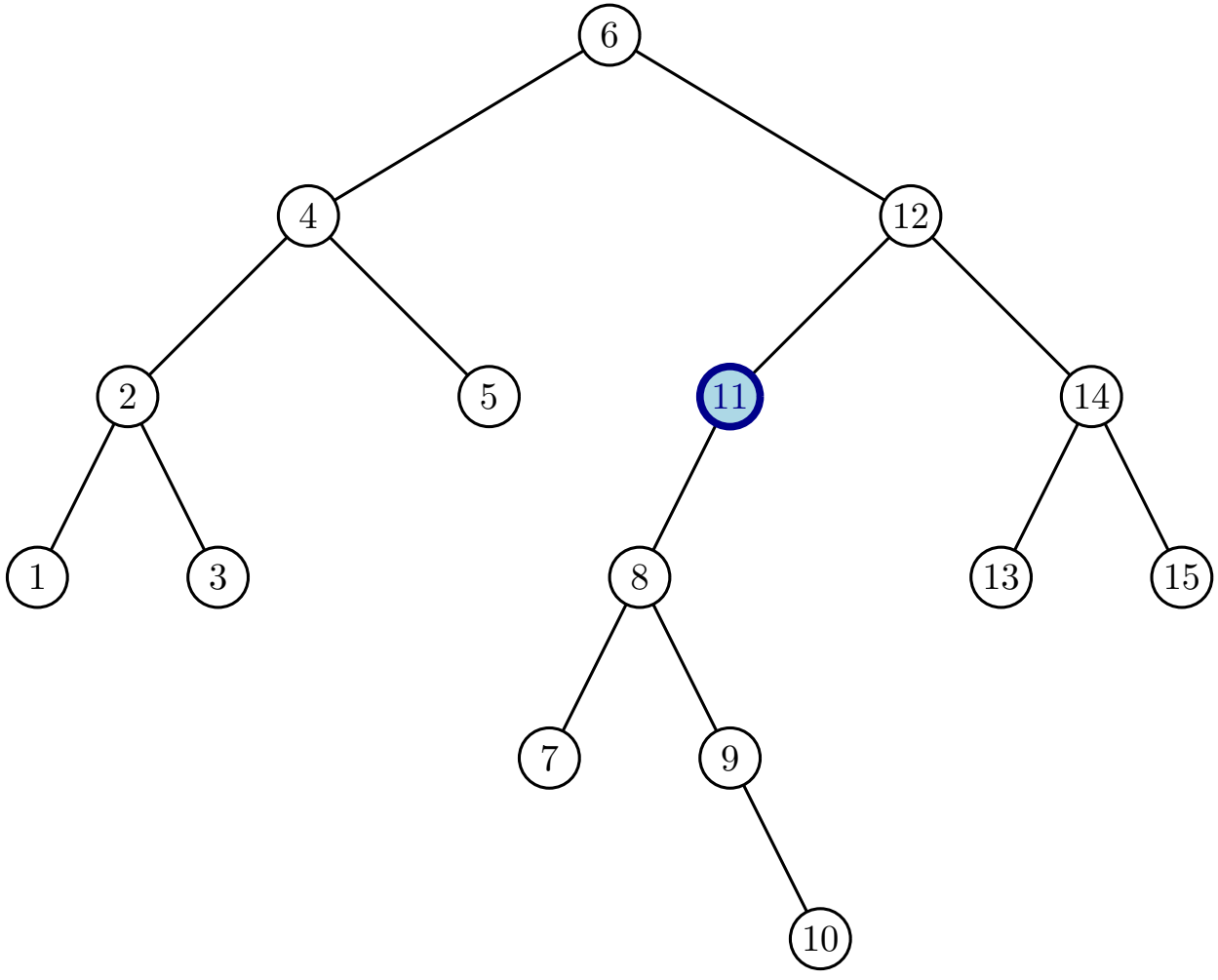
Suppression dans un ABR

si le nœud à supprimer n'a pas d'enfant



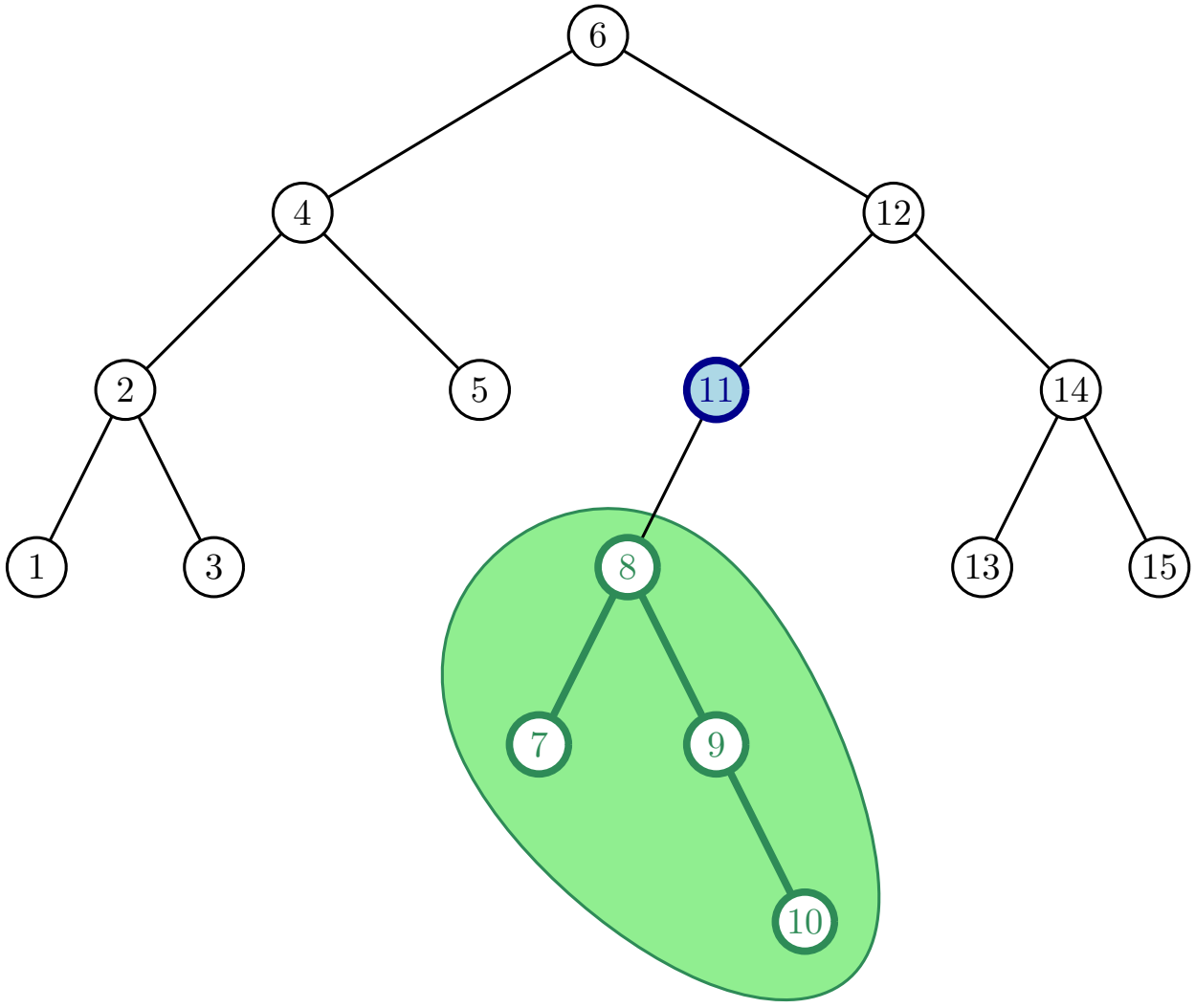
Suppression dans un ABR

si le nœud à supprimer n'a qu'un enfant



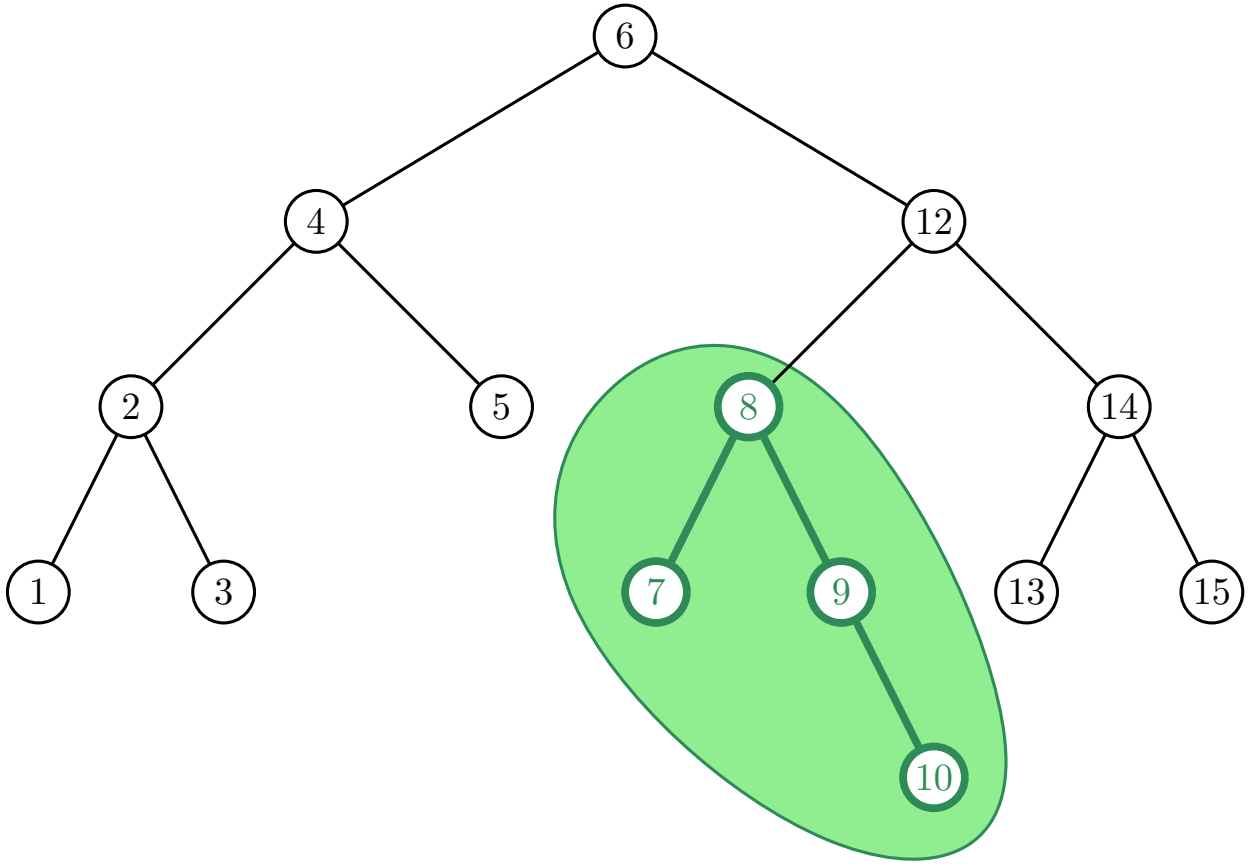
Suppression dans un ABR

si le nœud à supprimer n'a qu'un enfant



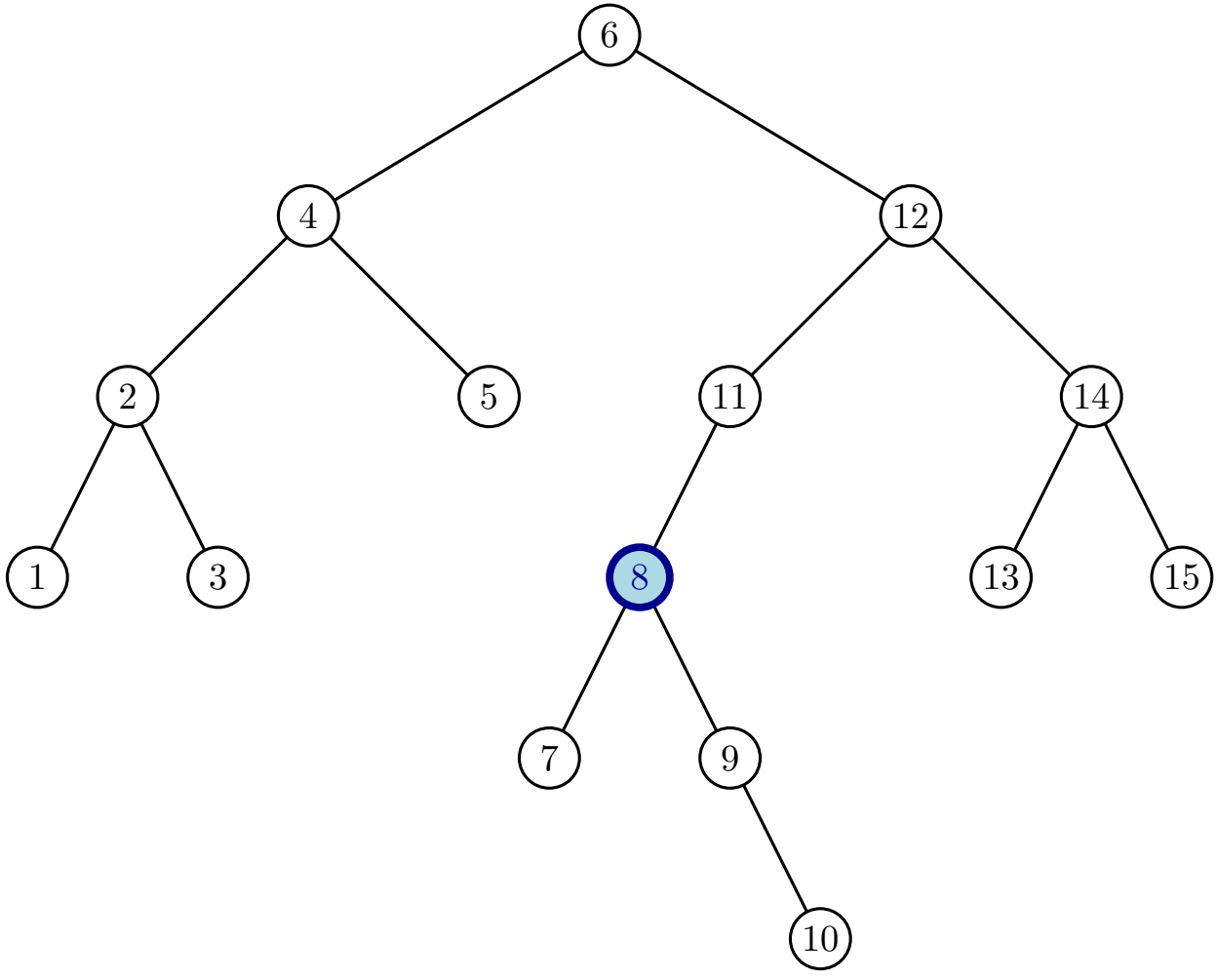
Suppression dans un ABR

si le nœud à supprimer n'a qu'un enfant



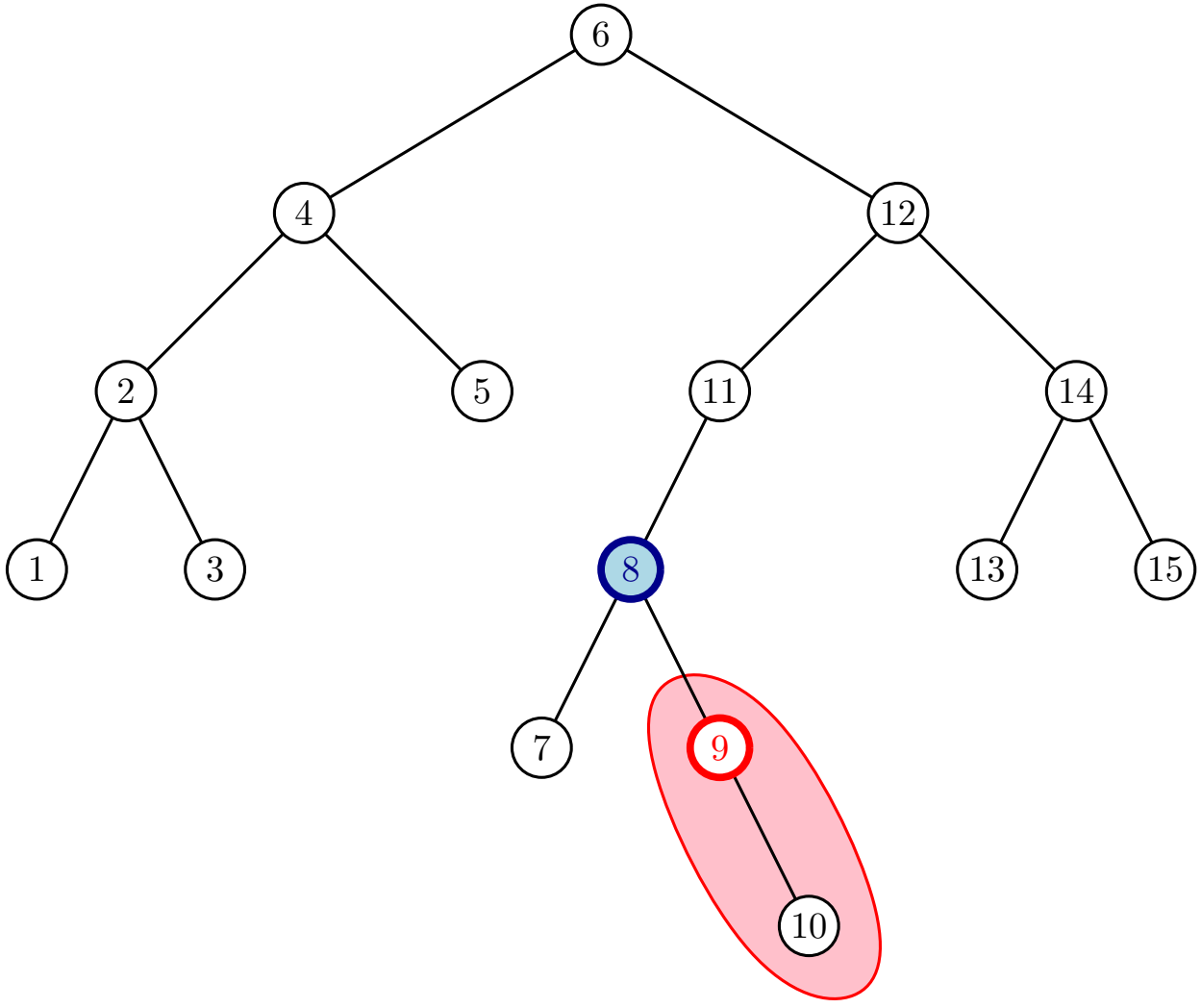
Suppression dans un ABR

sinon : remplacer le nœud à supprimer par son successeur



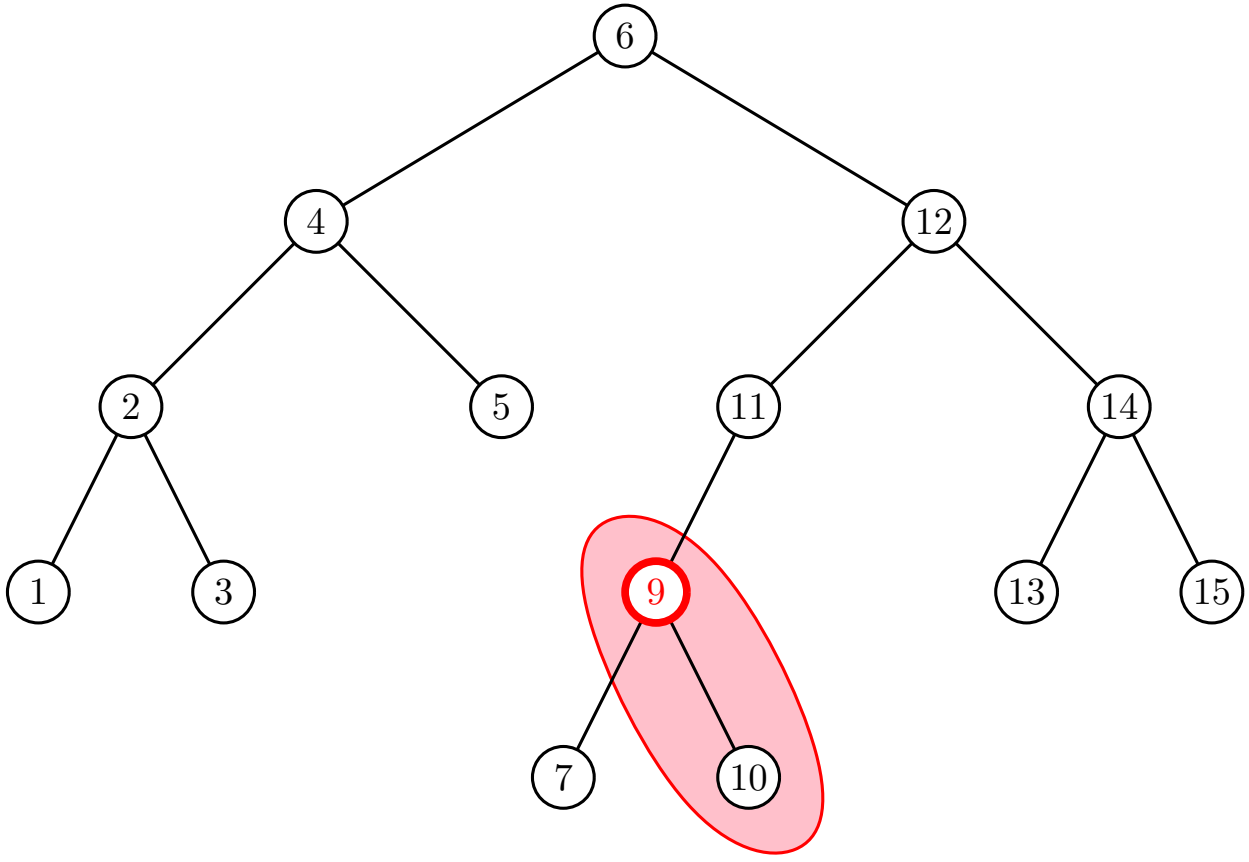
Suppression dans un ABR

sinon : remplacer le nœud à supprimer par son successeur



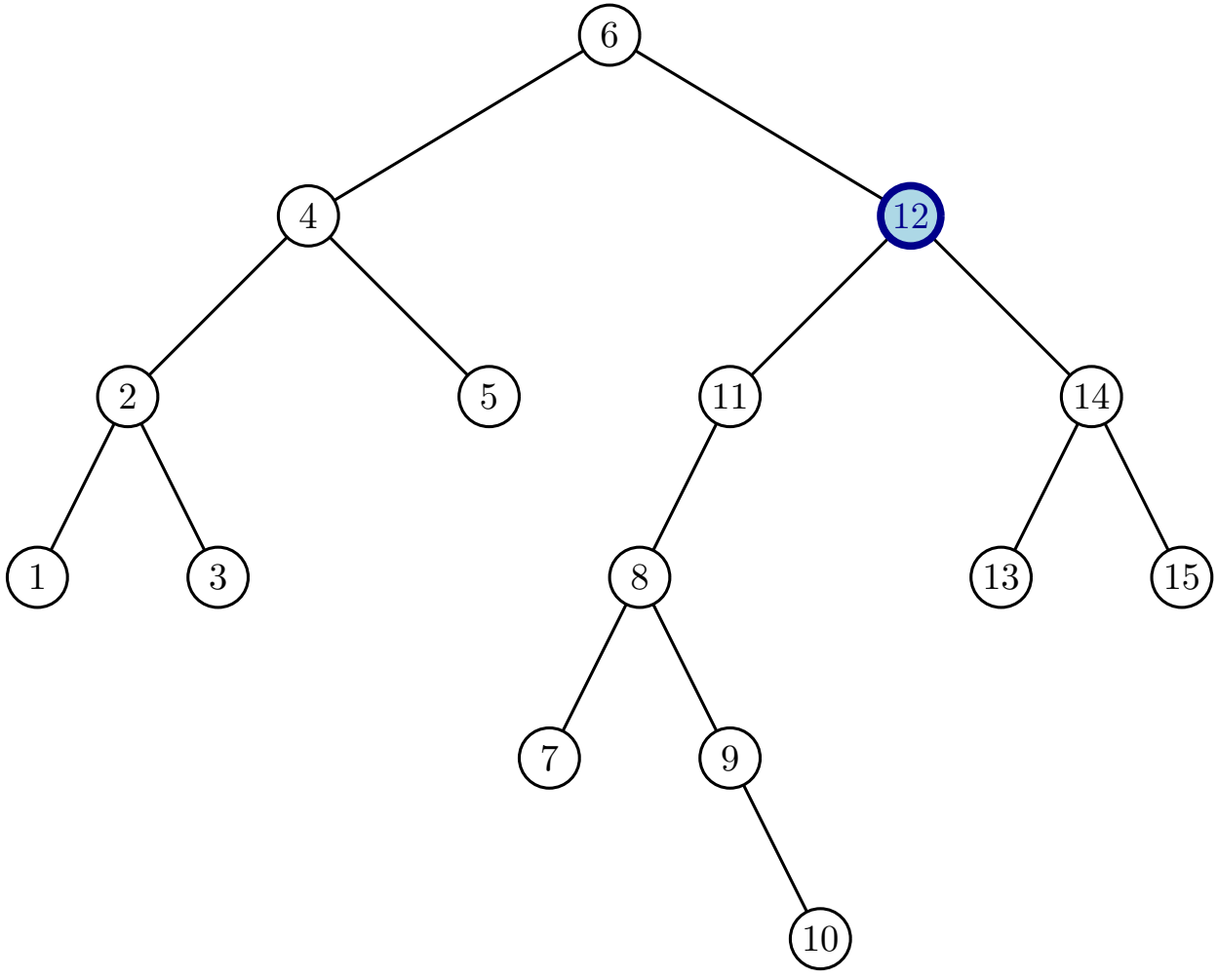
Suppression dans un ABR

sinon : remplacer le nœud à supprimer par son successeur



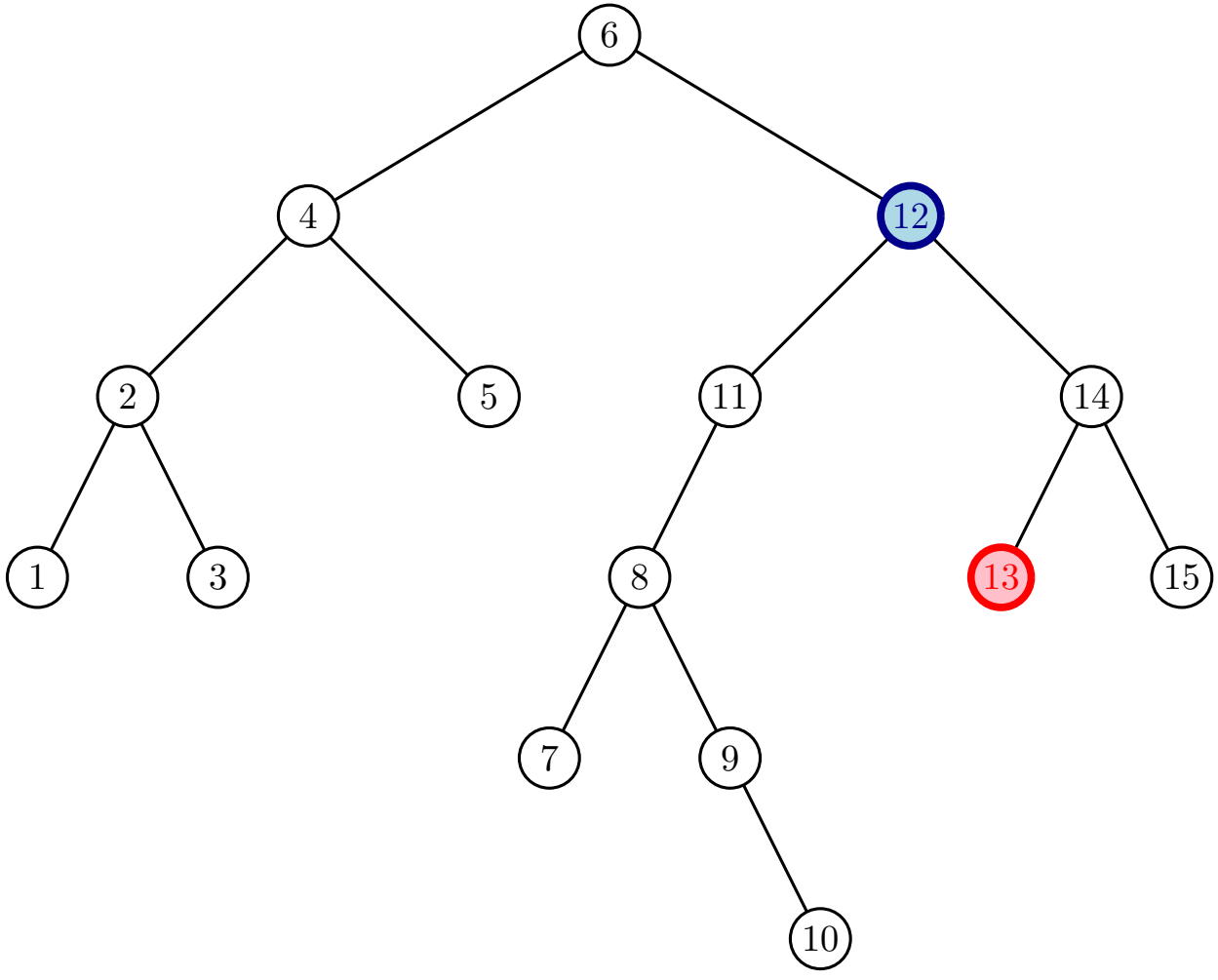
Suppression dans un ABR

sinon : remplacer le nœud à supprimer par son successeur



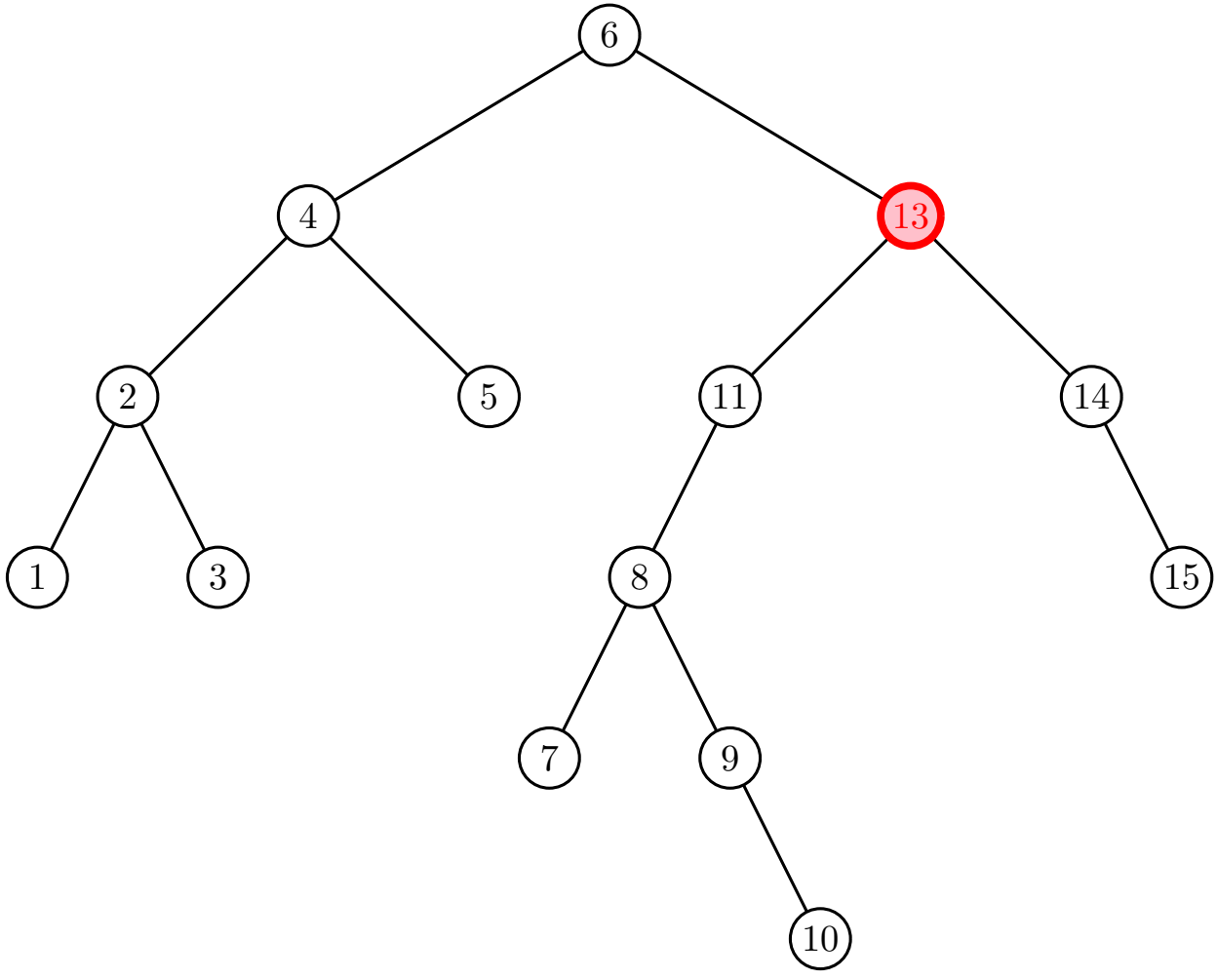
Suppression dans un ABR

sinon : remplacer le nœud à supprimer par son successeur



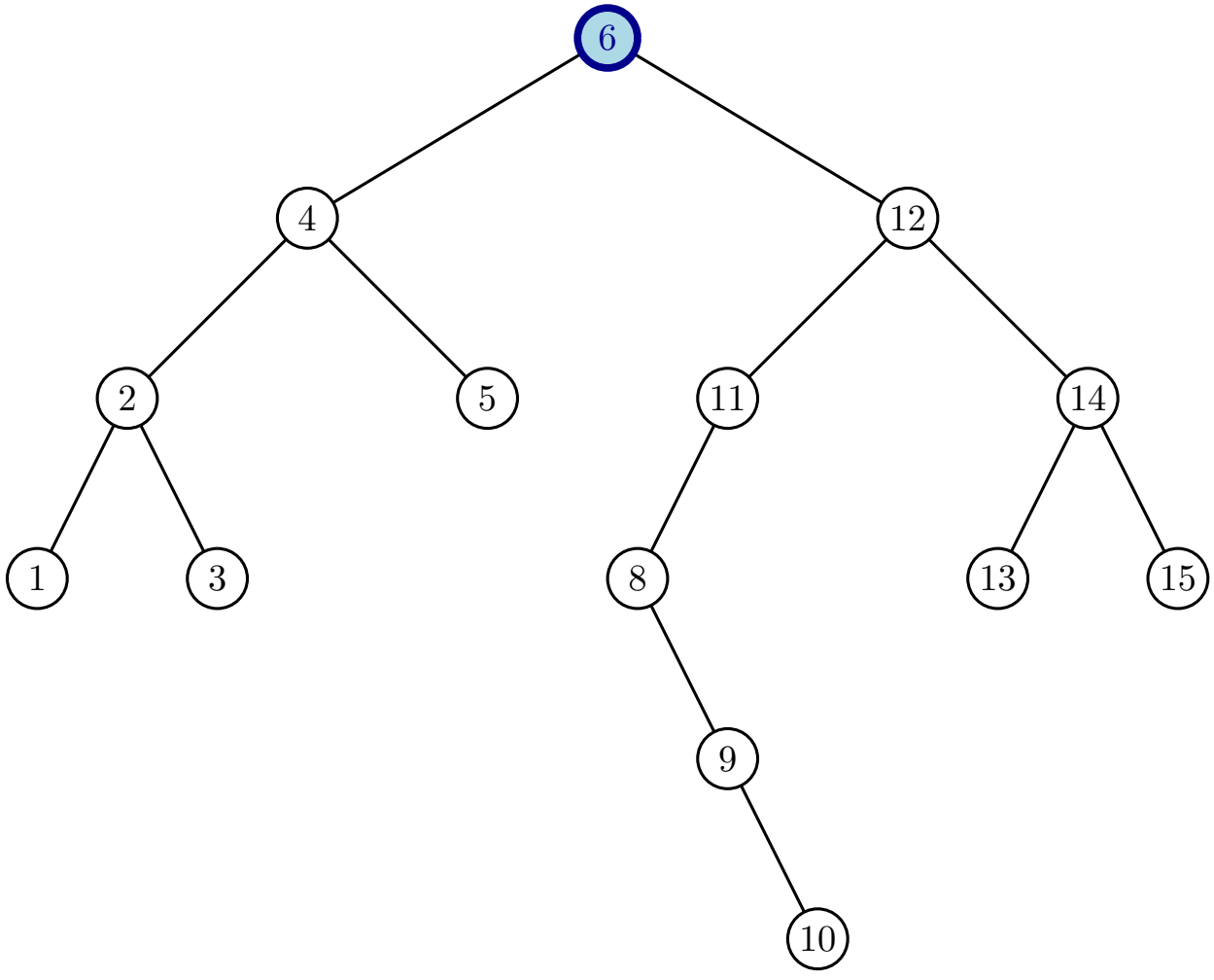
Suppression dans un ABR

sinon : remplacer le nœud à supprimer par son successeur



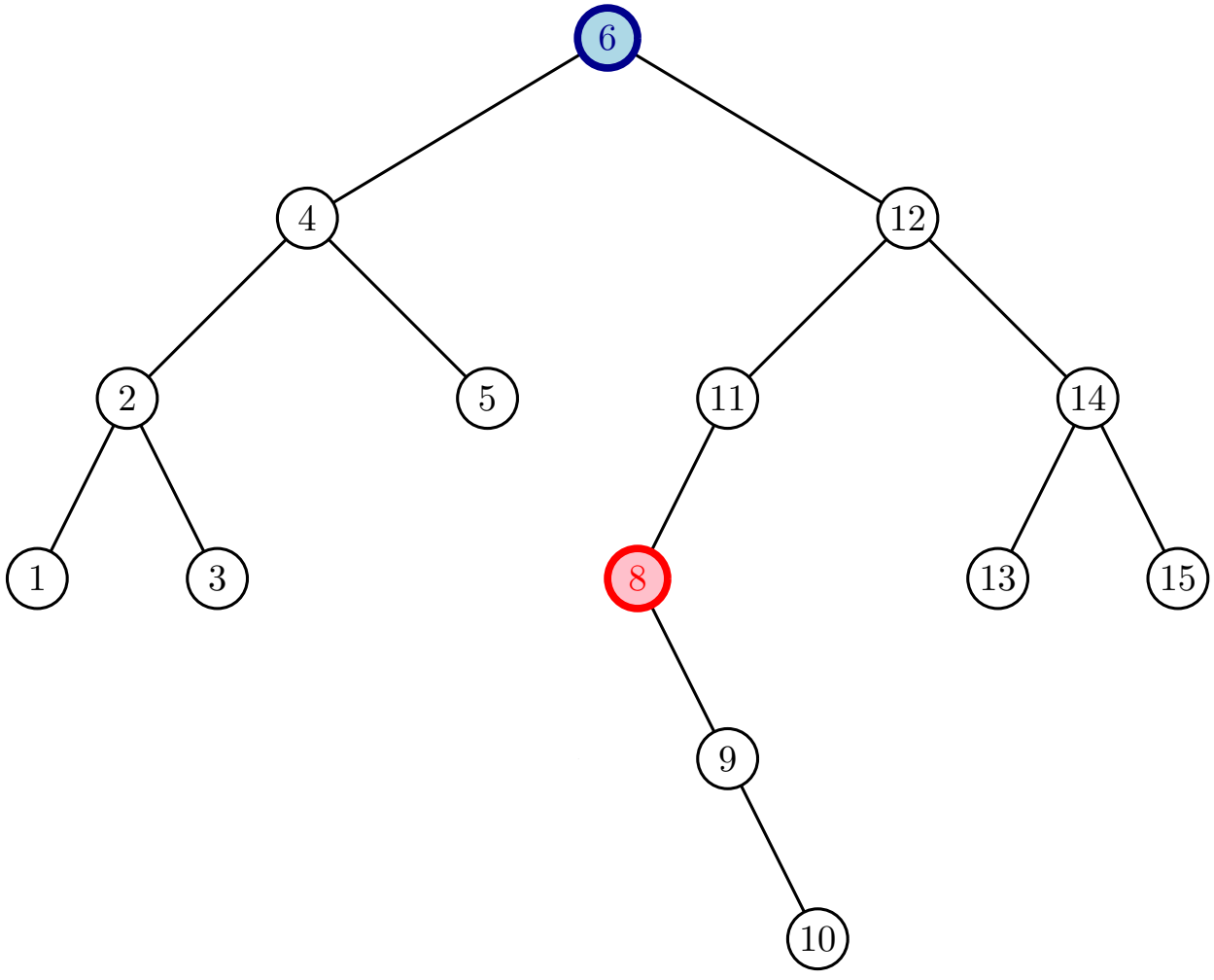
Suppression dans un ABR

sinon : remplacer le nœud à supprimer par son successeur



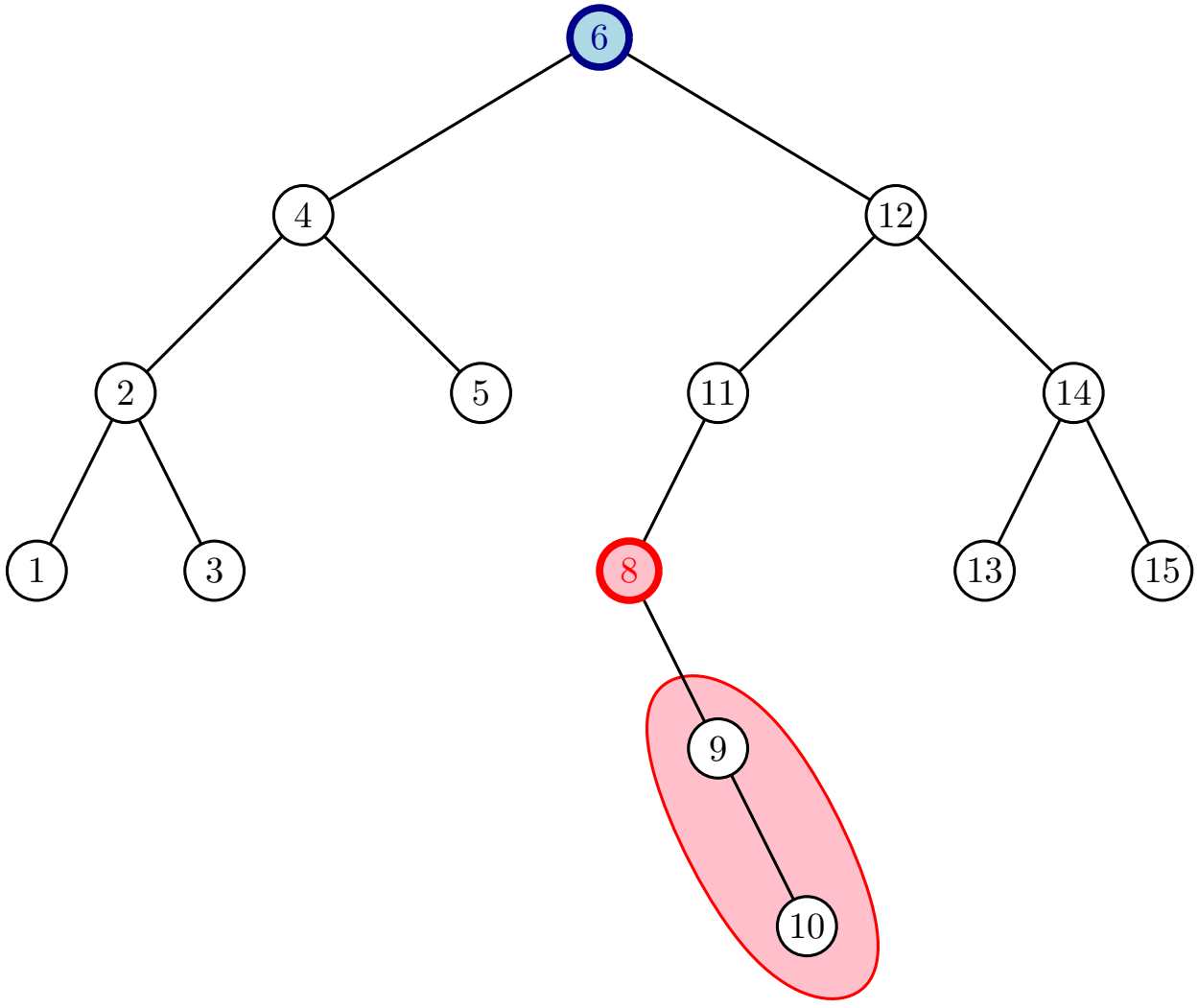
Suppression dans un ABR

sinon : remplacer le nœud à supprimer par son successeur



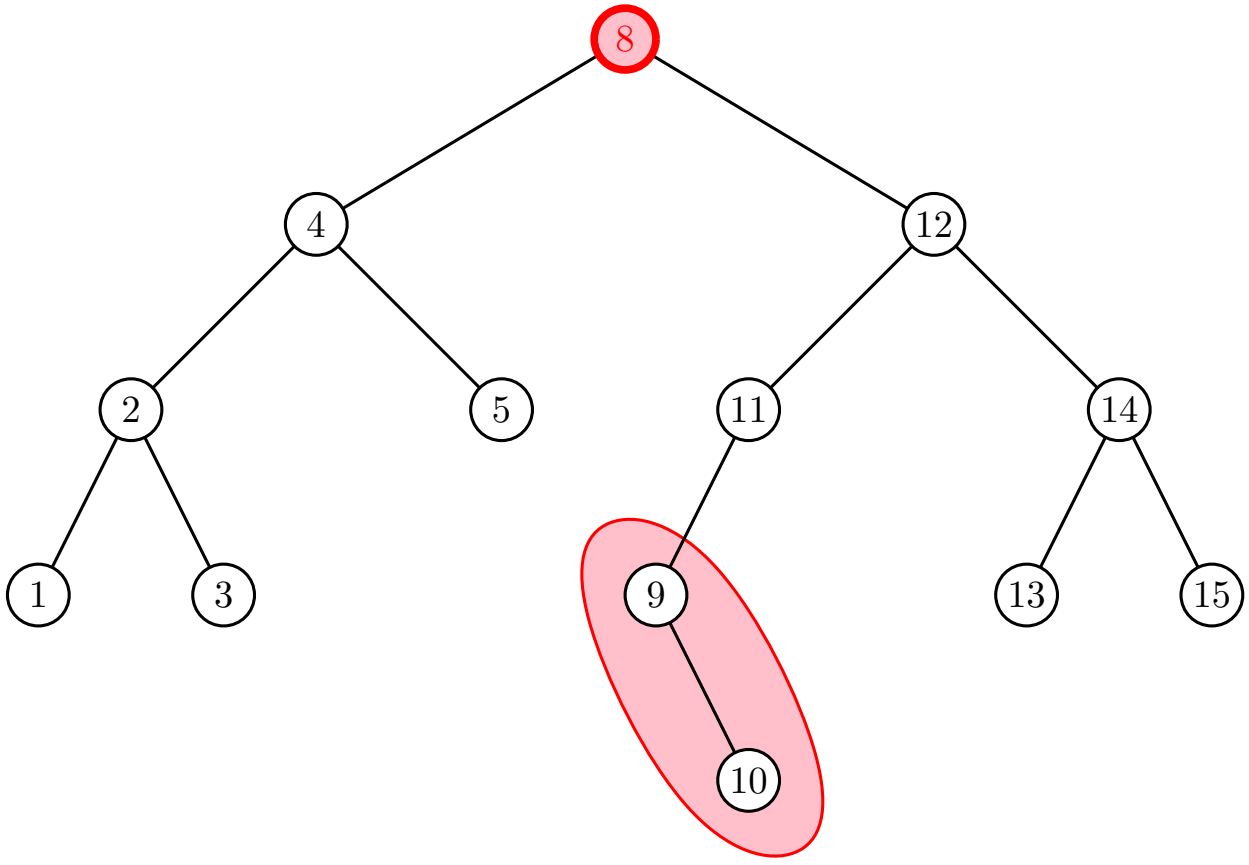
Suppression dans un ABR

sinon : remplacer le nœud à supprimer par son successeur



Suppression dans un ABR

sinon : remplacer le nœud à supprimer par son successeur



Suppression dans un ABR

cas d'une feuille : suppression simple

cas d'un nœud à un seul fils : l'autre fils remonte d'un niveau

cas où le successeur est le fils droit : le fils droit remonte d'un niveau et adopte son frère

autres cas : le noeud est remplacé par son successeur, dont l'unique fils (droit) remonte d'un niveau

Suppression dans un ABR

cas d'une feuille : suppression simple

cas d'un nœud à un seul fils : l'autre fils remonte d'un niveau

cas où le successeur est le fils droit : le fils droit remonte d'un niveau et adopte son frère

autres cas : le noeud est remplacé par son successeur, dont l'unique fils (droit) remonte d'un niveau

remarque : la même manipulation peut être faite avec le prédécesseur plutôt que le successeur

Suppression dans un ABR

cas d'une feuille : suppression simple

cas d'un nœud à un seul fils : l'autre fils remonte d'un niveau

cas où le successeur est le fils droit : le fils droit remonte d'un niveau et adopte son frère

autres cas : le noeud est remplacé par son successeur, dont l'unique fils (droit) remonte d'un niveau

Théorème

la suppression d'un nœud d'un ABR de hauteur h *se fait en temps* $\Theta(h)$ *au pire.*

Hauteur d'un ABR

la hauteur $h(A)$ d'un arbre binaire A à n nœuds vérifie :

$$\log n \leqslant h(A) \leqslant n - 1$$

Hauteur d'un ABR

la hauteur $h(A)$ d'un arbre binaire A à n nœuds vérifie :

$$\log n \leqslant h(A) \leqslant n - 1$$

chaque arbre binaire à n nœuds admet un unique étiquetage par $\{1, \ldots, n\}$ respectant les contraintes d'un ABR

Hauteur d'un ABR

la hauteur $h(A)$ d'un arbre binaire A à n nœuds vérifie :

$$\log n \leqslant h(A) \leqslant n - 1$$

chaque arbre binaire à n nœuds admet un unique étiquetage par $\{1, \ldots, n\}$ respectant les contraintes d'un ABR

Théorème (admis)

la hauteur moyenne d'un arbre binaire choisi uniformément parmi les arbres binaires à n nœuds est en $\Theta(\sqrt{n})$.

Hauteur d'un ABR

la hauteur $h(A)$ d'un arbre binaire A à n nœuds vérifie :

$$\log n \leqslant h(A) \leqslant n - 1$$

chaque arbre binaire à n nœuds admet un unique étiquetage par $\{1, \ldots, n\}$ respectant les contraintes d'un ABR

Théorème (admis)

la hauteur moyenne d'un arbre binaire choisi uniformément parmi les arbres binaires à n nœuds est en $\Theta(\sqrt{n})$.

et pourtant...

Théorème

la hauteur moyenne d'un ABR construit par l'insertion des entiers $1, \ldots, n$ dans un ordre aléatoire est en $\Theta(\log n)$.