# 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

# Quelle structure de données pour représenter un ensemble ?

# Quelle structure de données pour représenter un ensemble ?

Solution 1 : une liste de ses éléments, sans doublon

# Quelle structure de données pour représenter un ensemble ?

Solution 1 : une liste de ses éléments, sans doublon

| | tableau | | liste chaînée | |
|---|---|---|---|---|
| | non trié | trié | non triée | triée |
| recherche | $\Theta(n)$ | $\Theta(\log n)$ | $\Theta(n)$ | $\Theta(n)$ |

# Quelle structure de données pour représenter un ensemble ?

**Solution 1 : une liste de ses éléments, sans doublon**

| | tableau | | liste chaînée | |
|---|---|---|---|---|
| | non trié | trié | non triée | triée |
| recherche | $\Theta(n)$ | $\Theta(\log n)$ | $\Theta(n)$ | $\Theta(n)$ |
| insertion | $+\,\Theta(1)$ | $\Theta(n)$ | $+\,\Theta(1)$ | $+\,\Theta(1)$ |
| suppression | $\Theta(n)$ | $\Theta(n)$ | $+\,\Theta(1)$ | $+\,\Theta(1)$ |

# Quelle structure de données pour représenter un ensemble ?

**Solution 1 : une liste de ses éléments, sans doublon**

| | tableau | | liste chaînée | |
|---|---|---|---|---|
| | non trié | trié | non triée | triée |
| recherche | $\Theta(n)$ | $\Theta(\log n)$ | $\Theta(n)$ | $\Theta(n)$ |
| insertion | $+\,\Theta(1)$ | $\Theta(n)$ | $+\,\Theta(1)$ | $+\,\Theta(1)$ |
| suppression | $\Theta(n)$ | $\Theta(n)$ | $+\,\Theta(1)$ | $+\,\Theta(1)$ |
| minimum | $\Theta(n)$ | $\Theta(1)$ | $\Theta(n)$ | $\Theta(1)$ |

# Quelle structure de données pour représenter un ensemble ?

**Solution 1 : une liste de ses éléments, sans doublon**

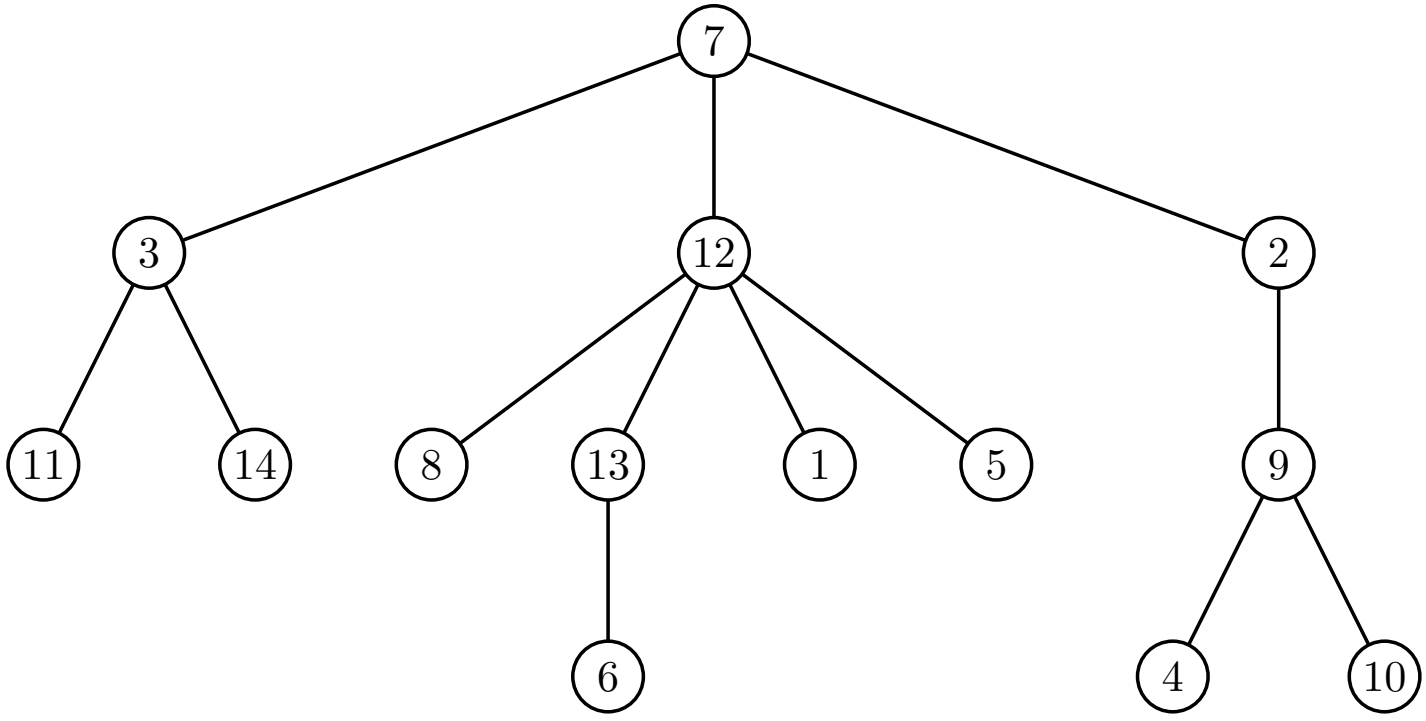
| | tableau | | liste chaînée | |
|---|---|---|---|---|
| | non trié | trié | non triée | triée |
| recherche | $\Theta(n)$ | $\Theta(\log n)$ | $\Theta(n)$ | $\Theta(n)$ |
| insertion | $+\,\Theta(1)$ | $\Theta(n)$ | $+\,\Theta(1)$ | $+\,\Theta(1)$ |
| suppression | $\Theta(n)$ | $\Theta(n)$ | $+\,\Theta(1)$ | $+\,\Theta(1)$ |
| minimum | $\Theta(n)$ | $\Theta(1)$ | $\Theta(n)$ | $\Theta(1)$ |
| sélection du $k^e$ | $\Theta(kn)$ | $\Theta(1)$ | $\Theta(kn)$ | $\Theta(k)$ |

# Quelle structure de données pour représenter un ensemble ?

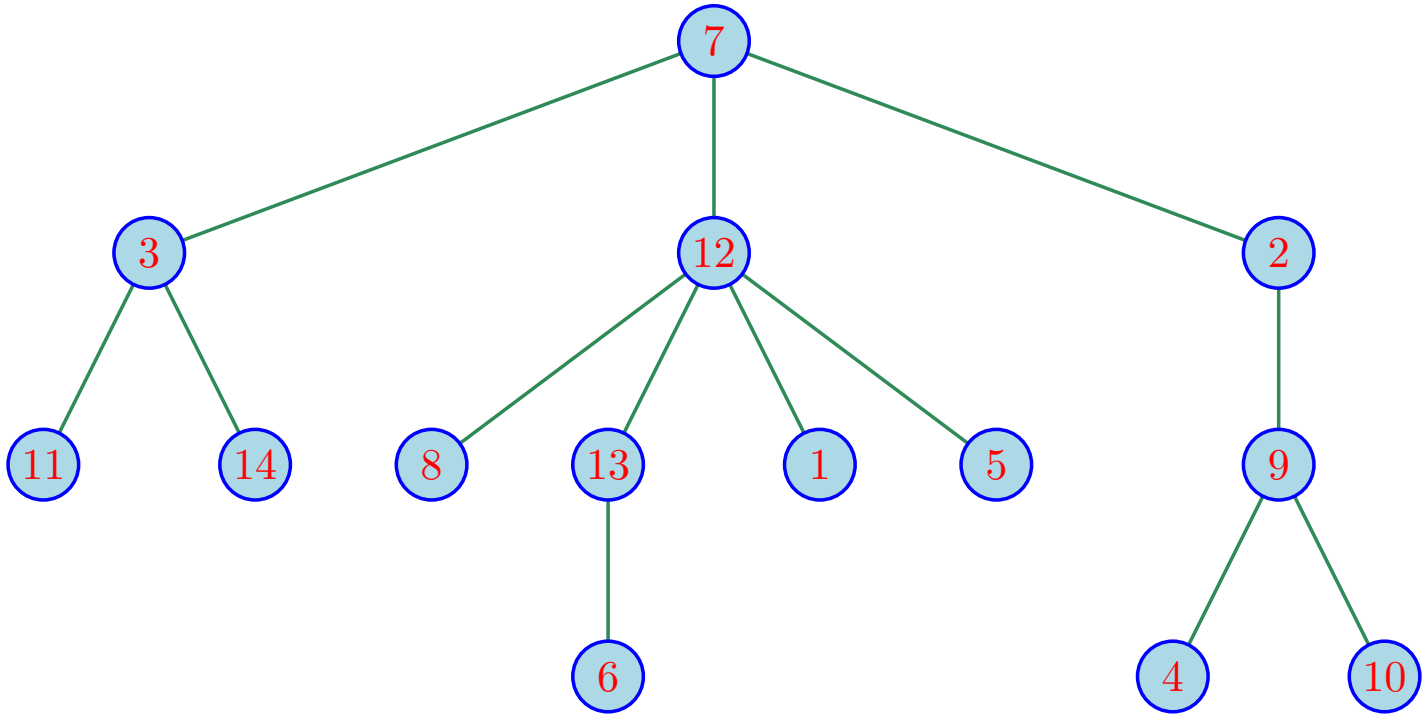
Solution 1 : une liste de ses éléments, sans doublon

| | tableau | | liste chaînée | |
|---|---|---|---|---|
| | non trié | trié | non triée | triée |
| recherche | $\Theta(n)$ | $\Theta(\log n)$ | $\Theta(n)$ | $\Theta(n)$ |
| insertion | $+\Theta(1)$ | $\Theta(n)$ | $+\Theta(1)$ | $+\Theta(1)$ |
| suppression | $\Theta(n)$ | $\Theta(n)$ | $+\Theta(1)$ | $+\Theta(1)$ |
| minimum | $\Theta(n)$ | $\Theta(1)$ | $\Theta(n)$ | $\Theta(1)$ |
| sélection du $k^e$ | $\Theta(kn)$ | $\Theta(1)$ | $\Theta(kn)$ | $\Theta(k)$ |
| union | $\Theta(n^2)$ | $\Theta(n)$ | $\Theta(n^2)$ | $\Theta(n)$ |

# Idée : utiliser des arbres

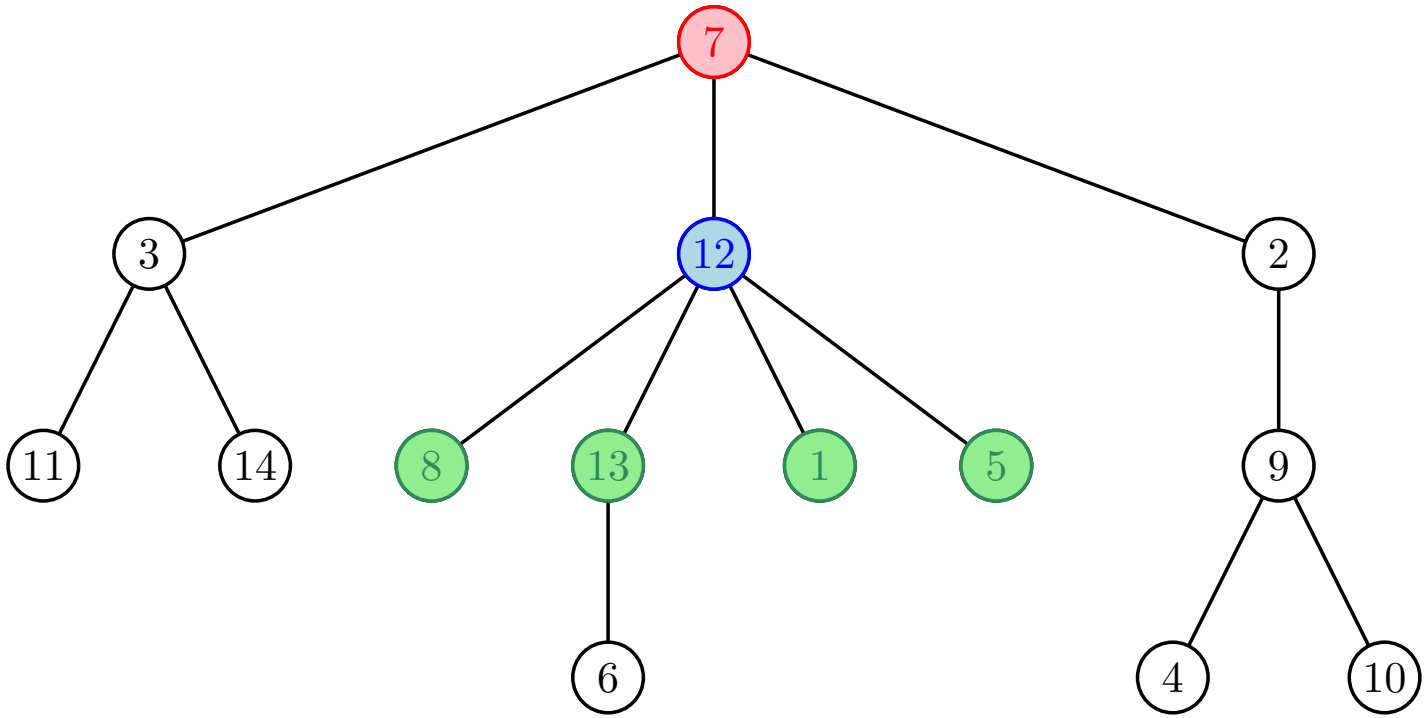


# Idée : utiliser des arbres



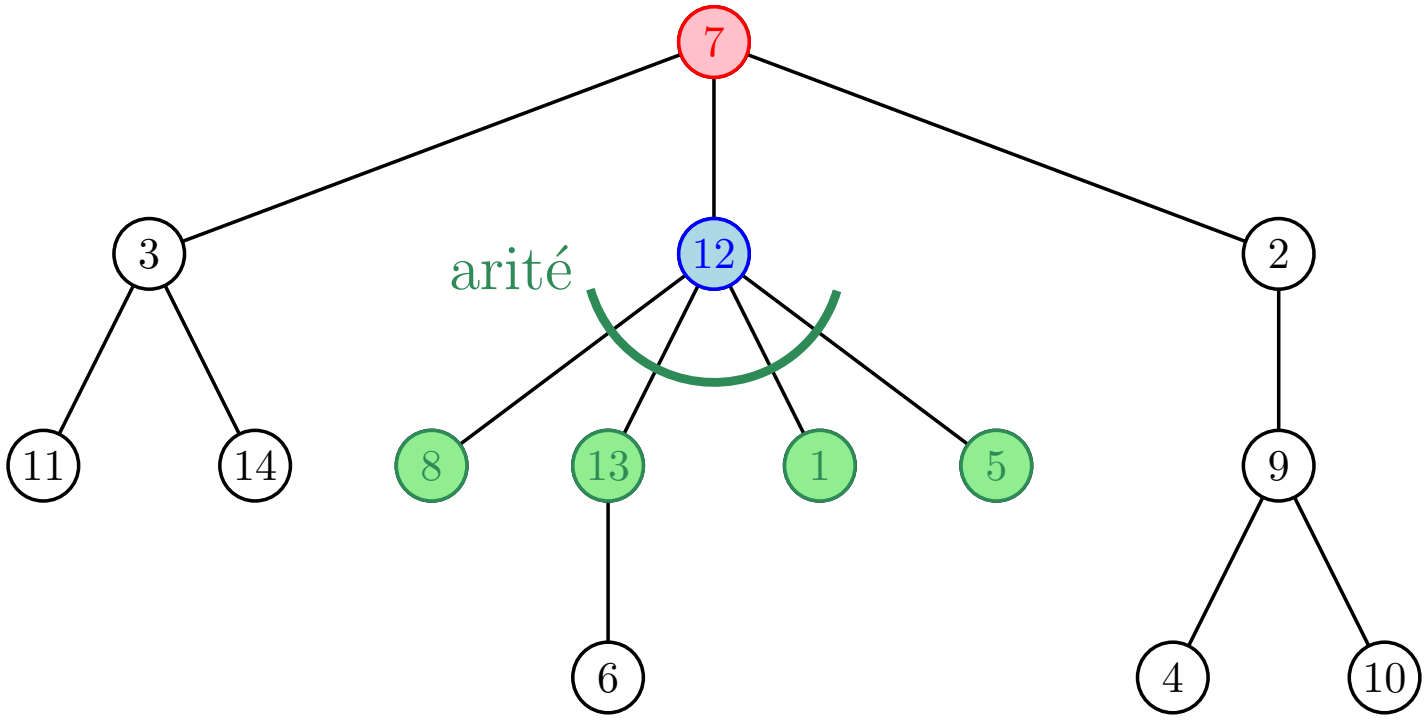

sommets contenant des étiquettes reliés par des arêtes

# Idée : utiliser des arbres



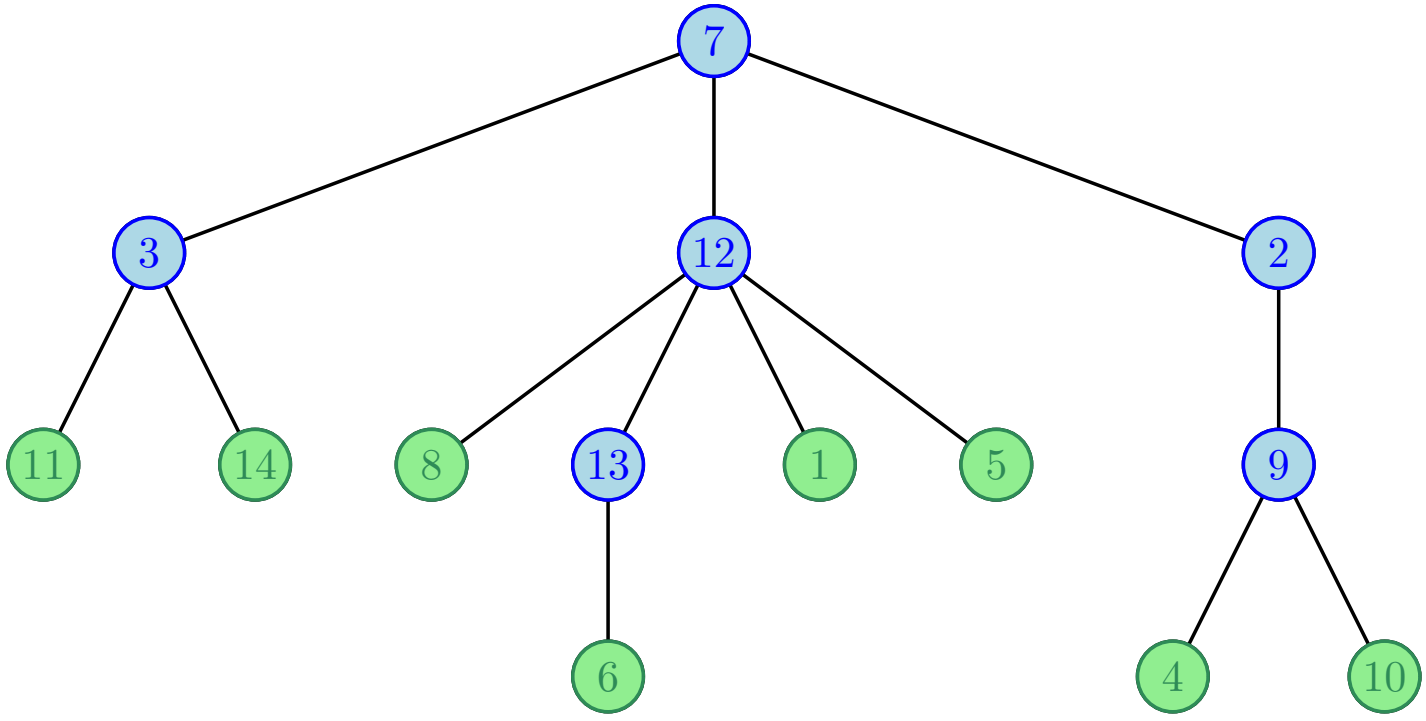

hiérarchie entre les sommets : père , fils

# Idée : utiliser des arbres



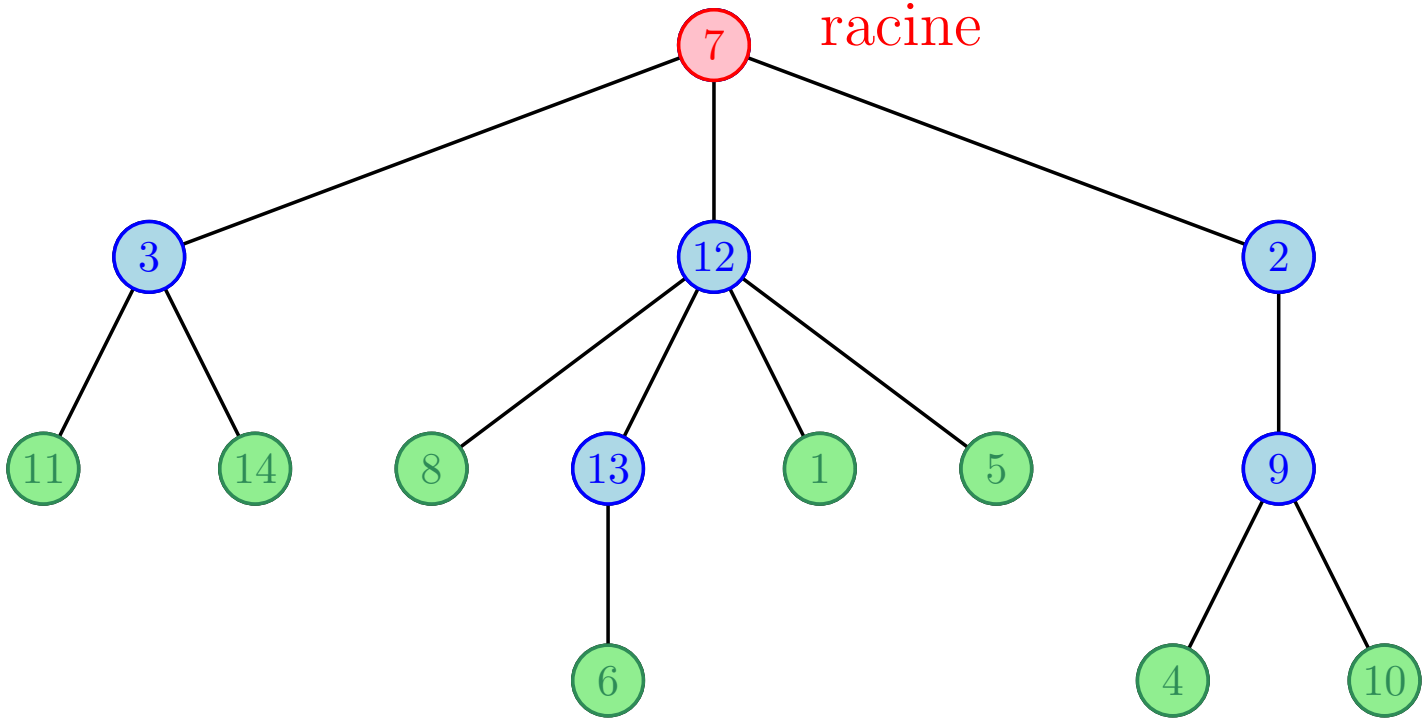

hiérarchie entre les sommets : père , fils

# Idée : utiliser des arbres



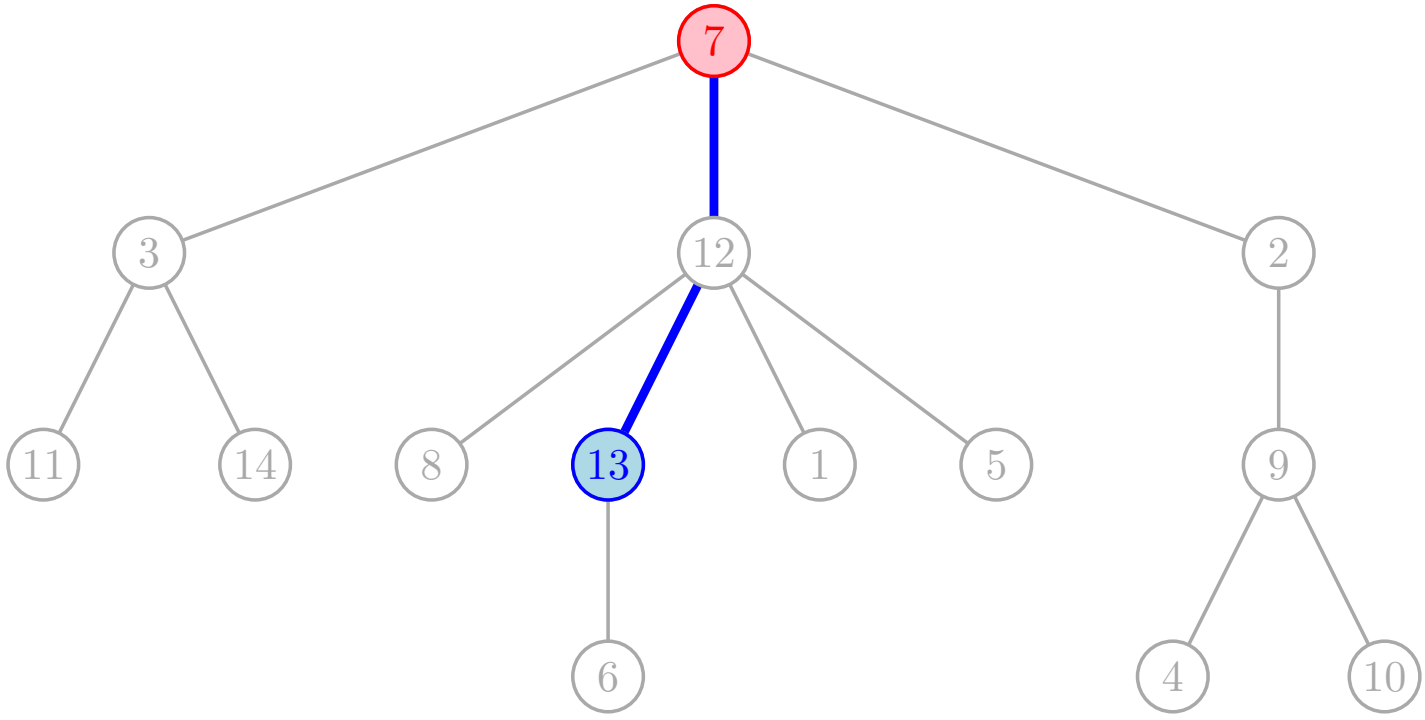

sommet = nœud ou feuille

# Idée : utiliser des arbres



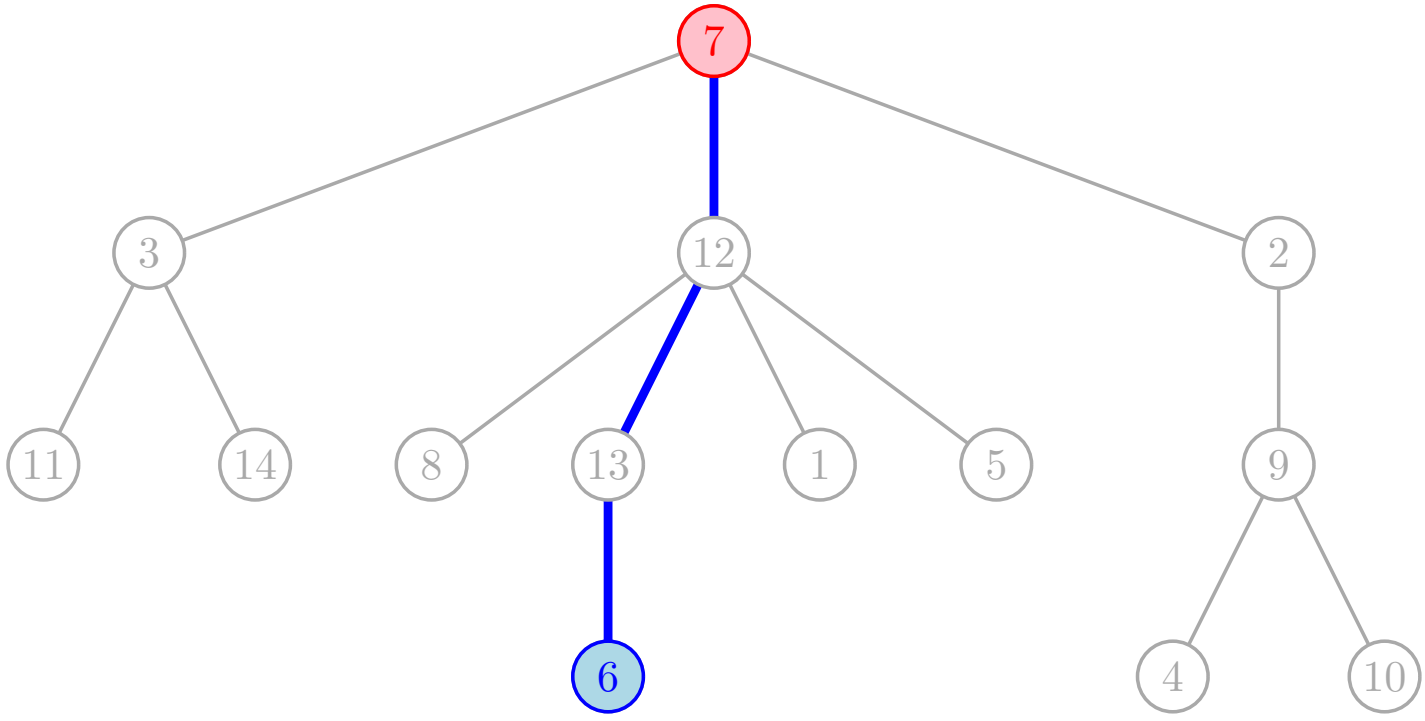

sommet = nœud ou feuille

# Idée : utiliser des arbres



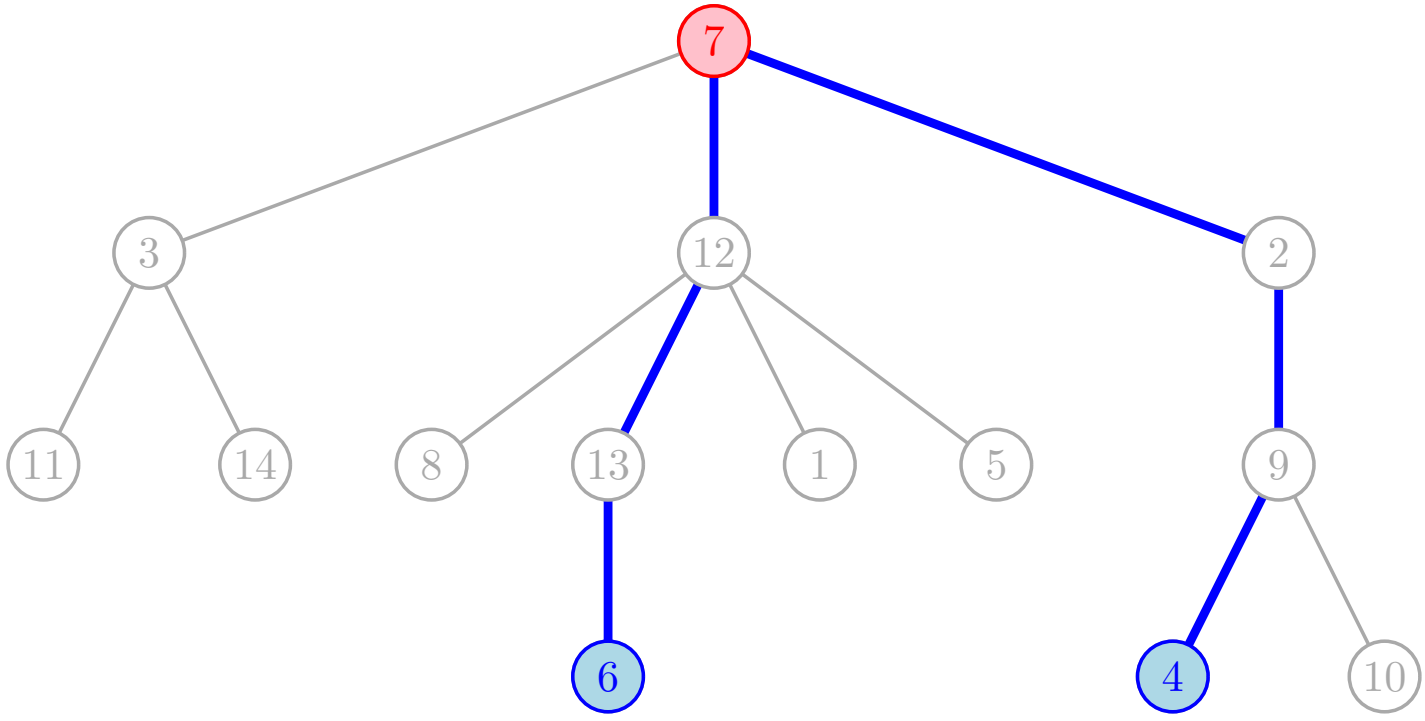

profondeur d'un sommet = distance à la racine

# Idée : utiliser des arbres




hauteur de l'arbre = profondeur maximale

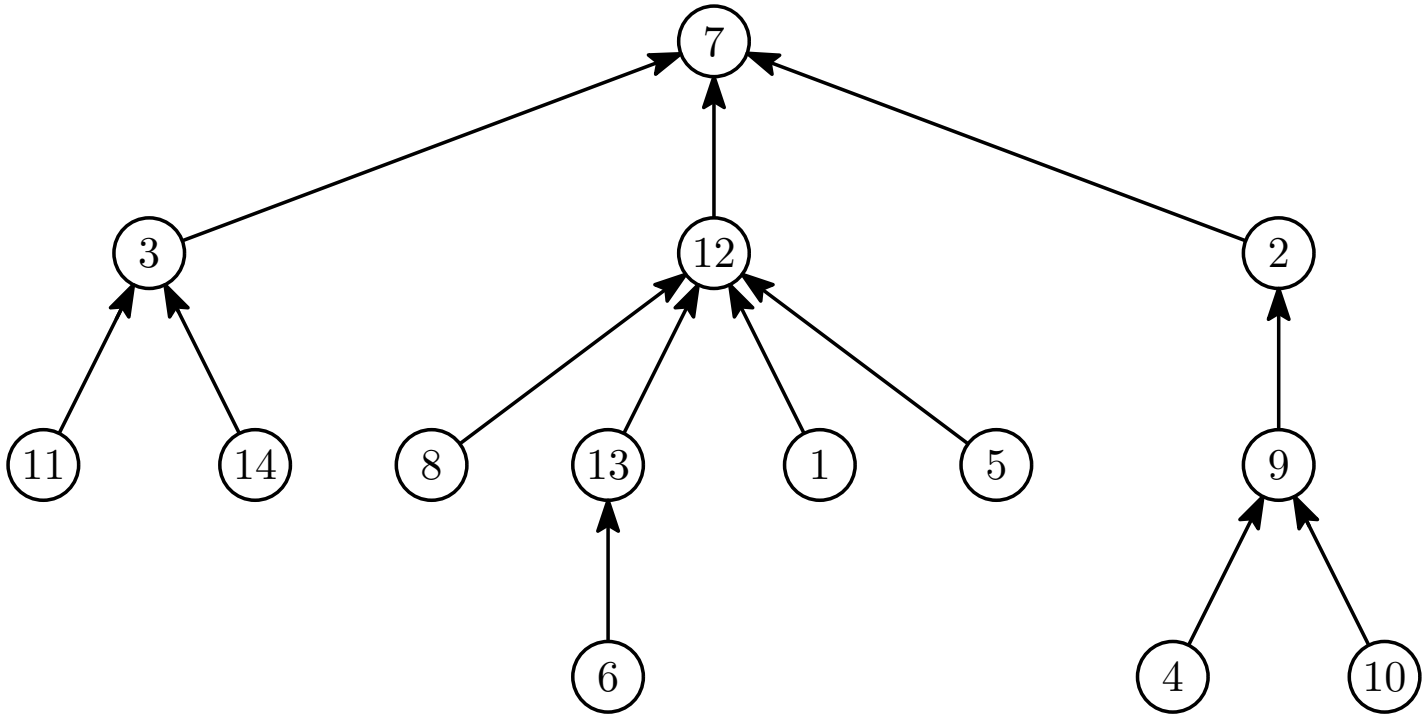
# Idée : utiliser des arbres




hauteur de l'arbre = profondeur maximale

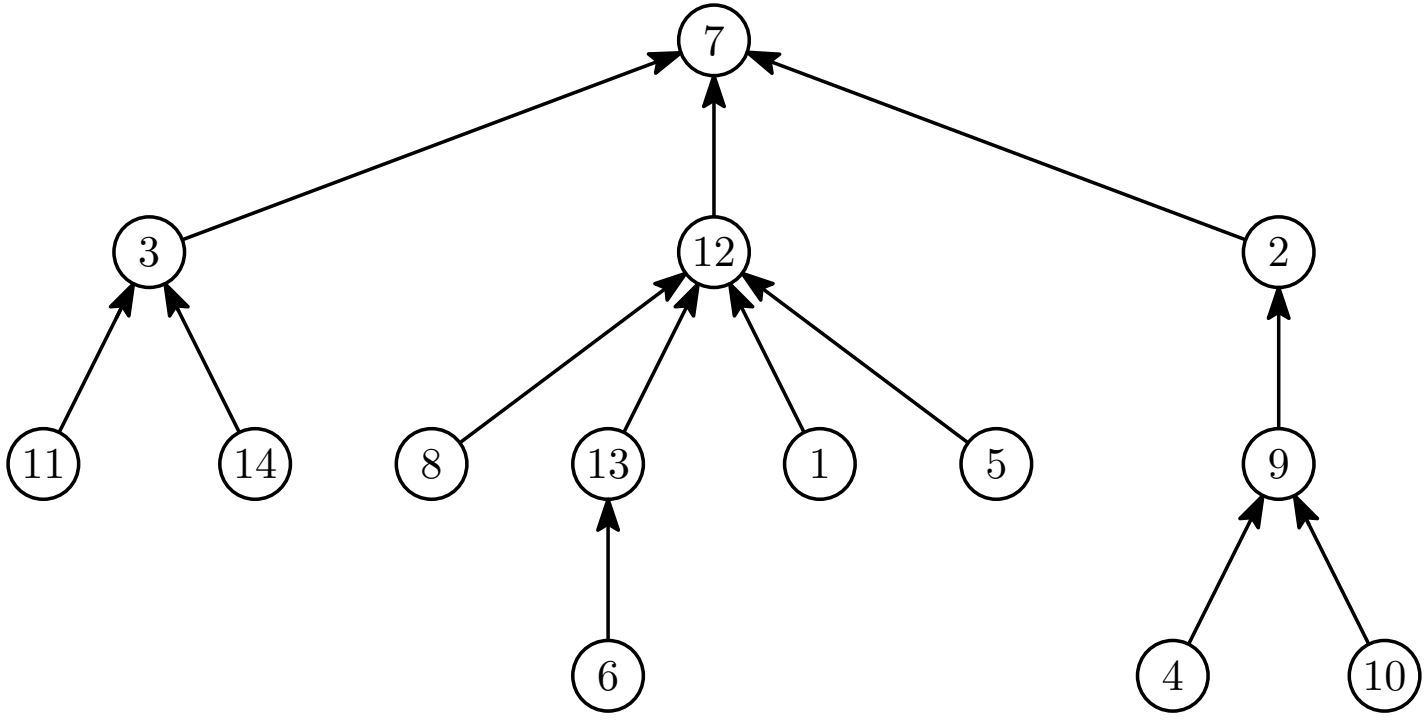
# Représentation des arbres

pointeur vers le père (dans chaque nœud, ou dans un tableau)



# Représentation des arbres

pointeur vers le père (dans chaque nœud, ou dans un tableau)



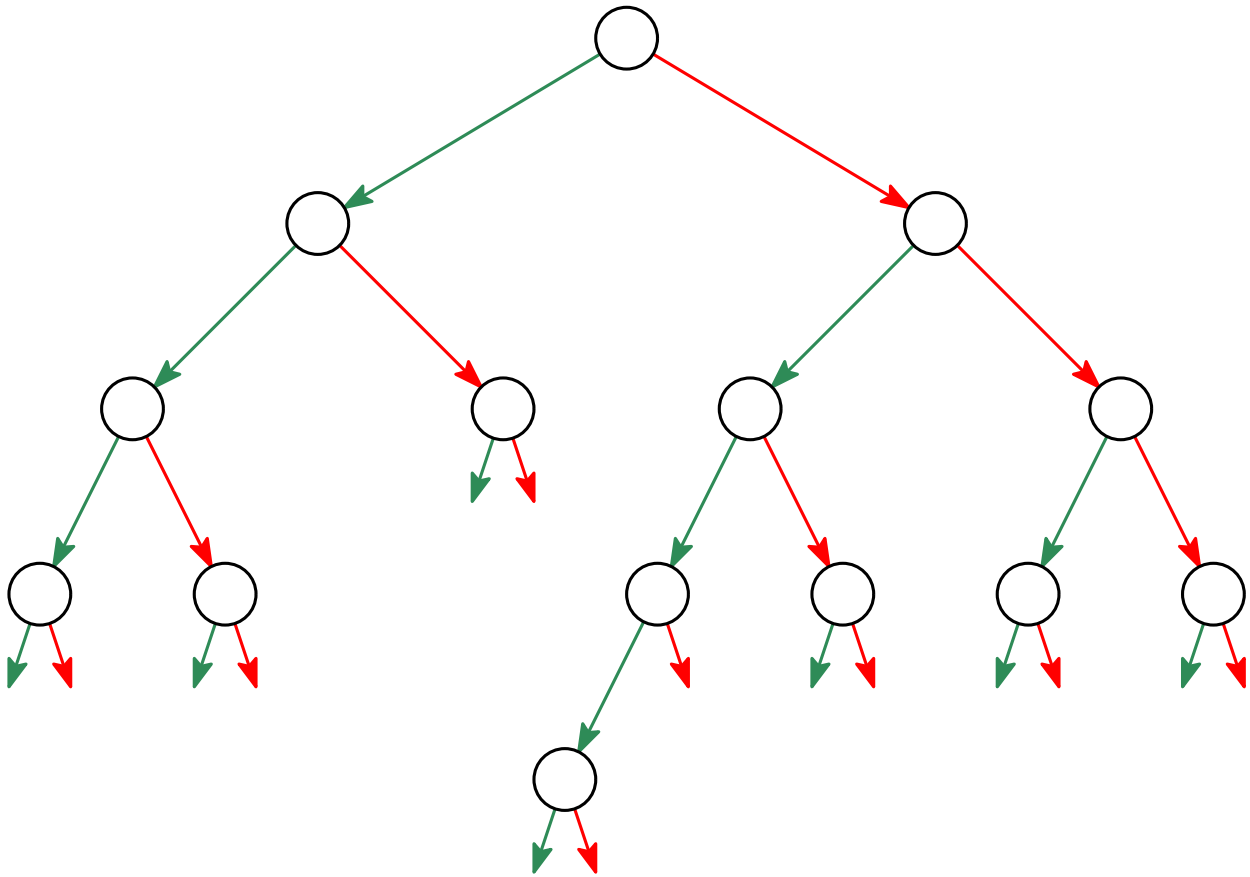

avantage : représentation très compacte

inconvénient : ne peut être parcouru que de bas en haut

$\Longrightarrow$ utilisée pour représenter une partition en sous-ensembles (structure *union-find*)

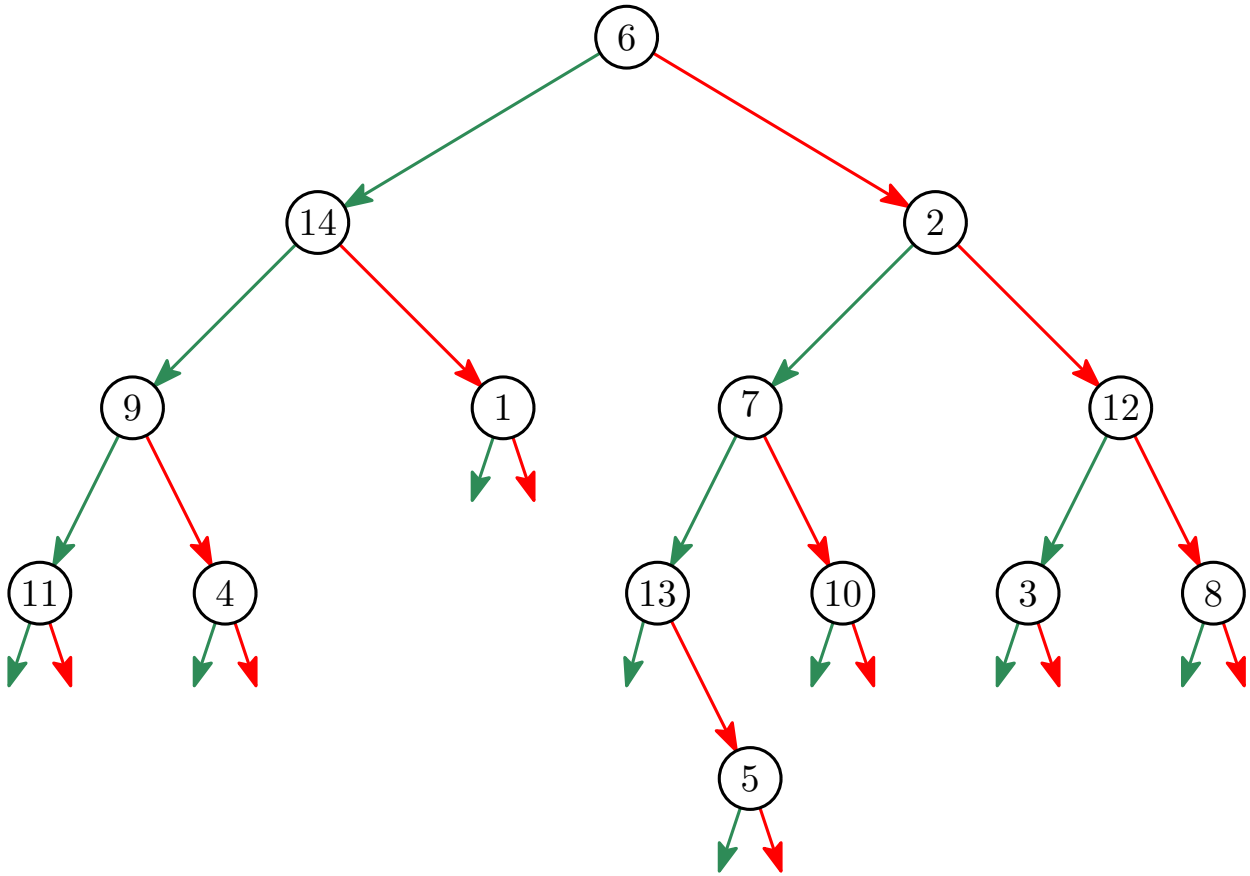
# Représentation des arbres

cas des arbres à arité fixe (en particulier les arbres binaires) :
un pointeur vers chaque fils (plus éventuellement le père)



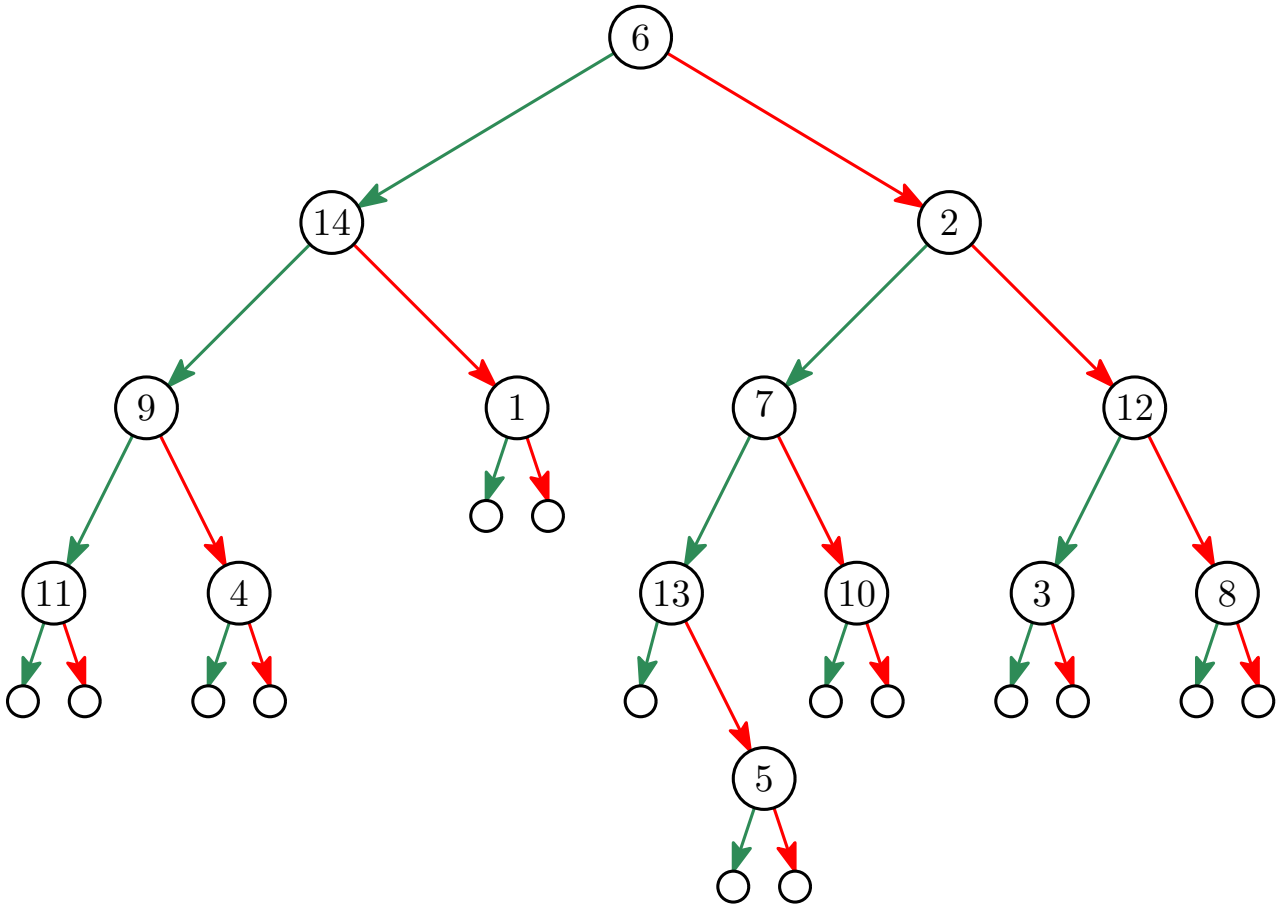
# Représentation des arbres

cas des arbres à arité fixe (en particulier les arbres binaires) :
un pointeur vers chaque fils (plus éventuellement le père)



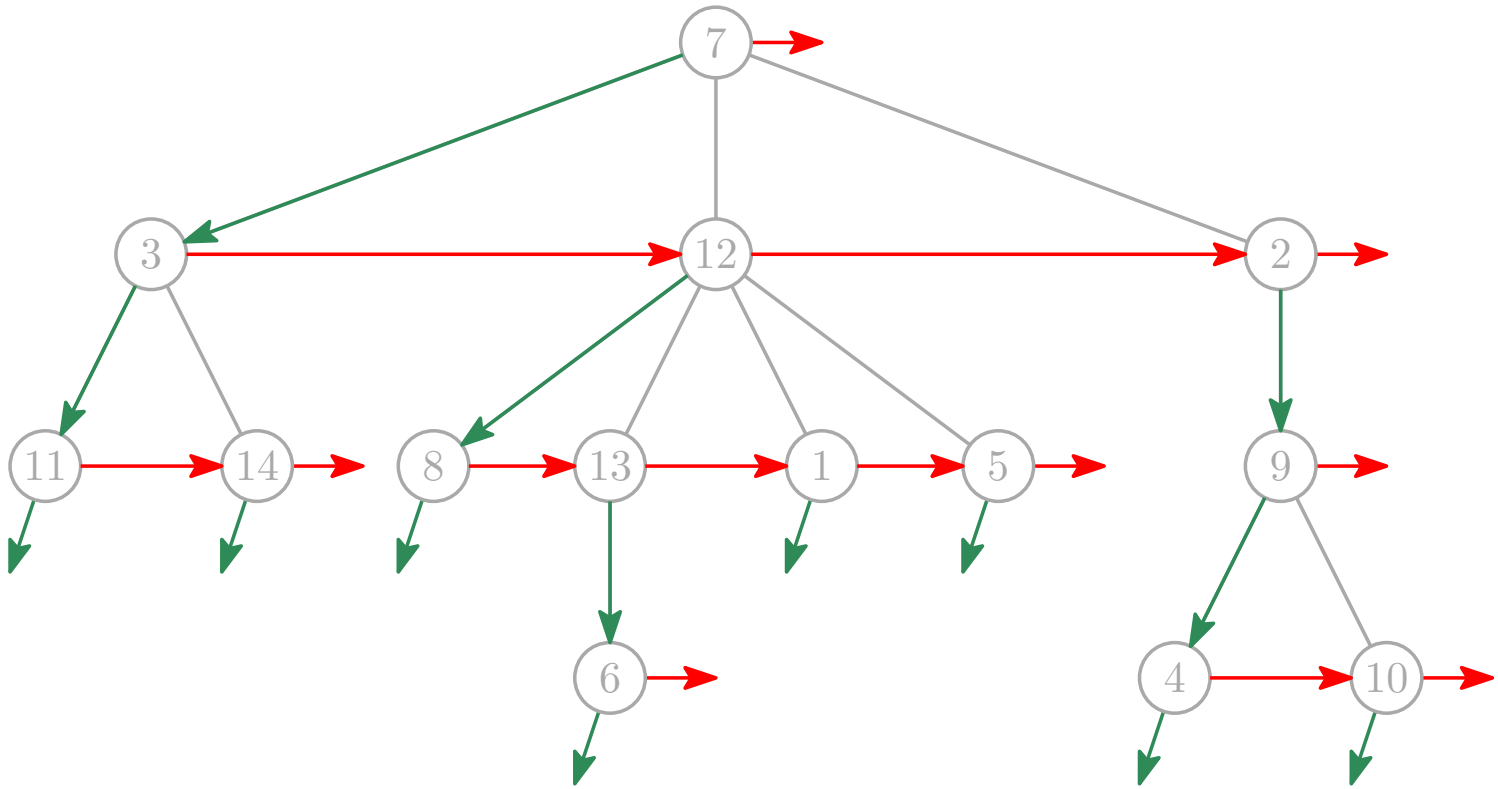
# Représentation des arbres

cas des arbres à arité fixe (en particulier les arbres binaires) :
un pointeur vers chaque fils (plus éventuellement le père)



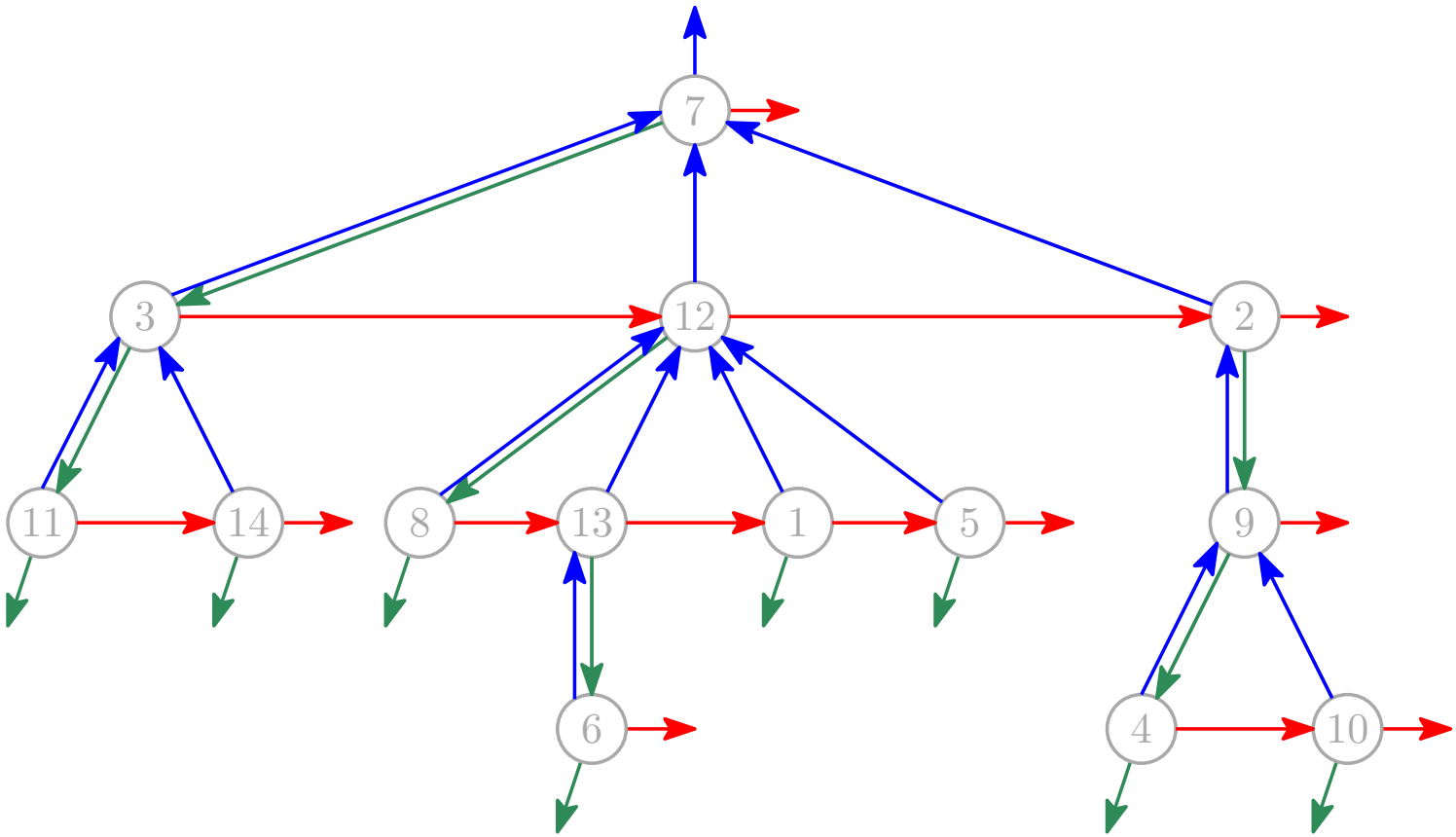
# Représentation des arbres

cas général : pointeurs vers le fils aîné et le frère cadet



# Représentation des arbres

cas général : pointeurs vers le fils aîné et le frère cadet (plus éventuellement le père)



# Parcourir un arbre

parcours en profondeur générique :

```
def parcours(A) :
  pre_traitement(A)
  for i, B in enumerate(sous_arbres(A)) :
    parcours(B)
    post_traitement(A, i)
```

# Parcourir un arbre

parcours en profondeur générique :

```
def parcours(A) :
  pre_traitement(A)
  for i, B in enumerate(sous_arbres(A)) :
    parcours(B)
    post_traitement(A, i)
```

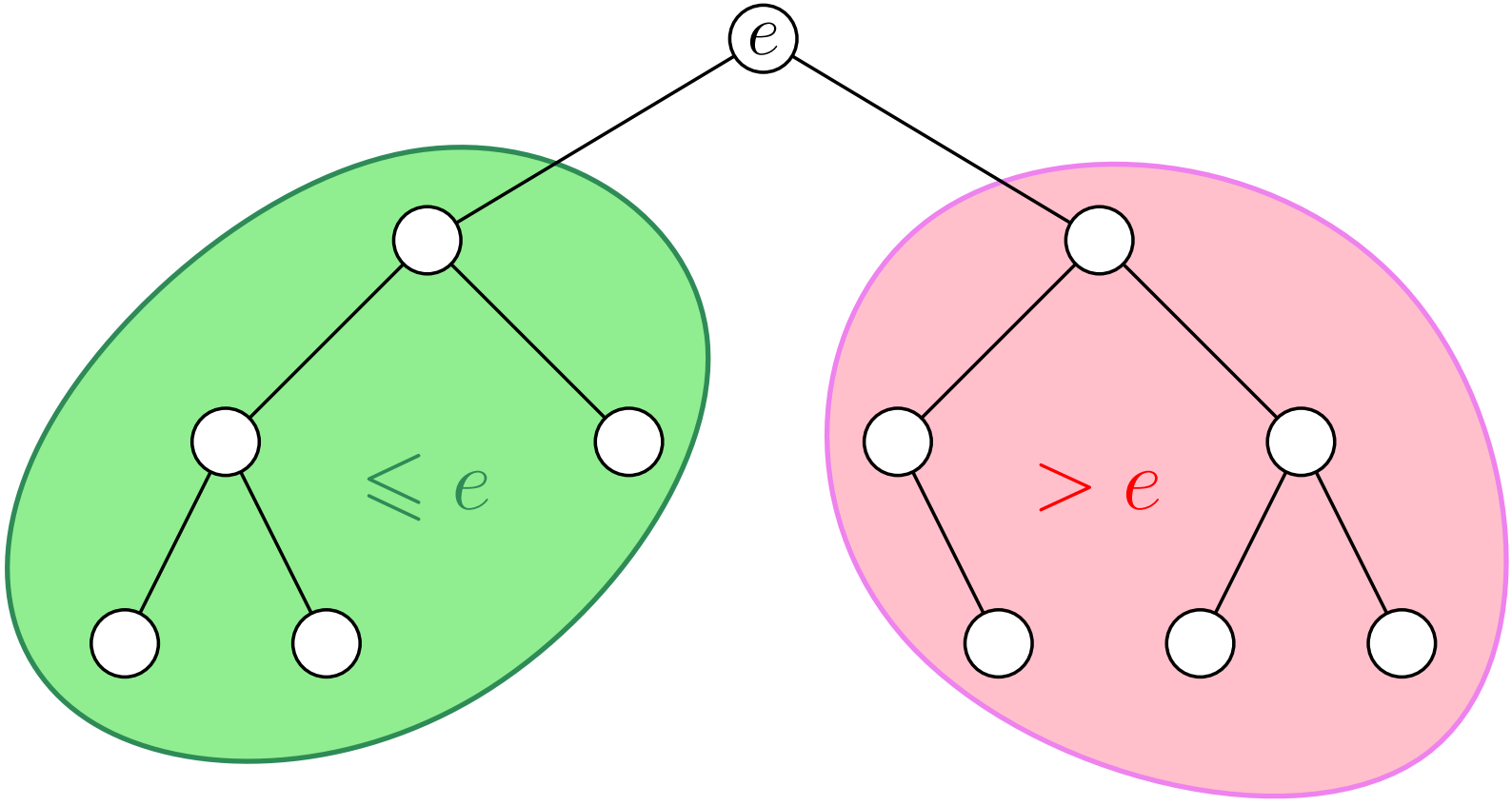
variation selon les traitements intermediaires :

- s'il y a seulement un prétraitement : *parcours préfixe*
- s'il y a seulement un posttraitement : *parcours postfixe*
- dans le cas binaire, s'il y a seulement un traitement intermédiaire : *parcours infixe*

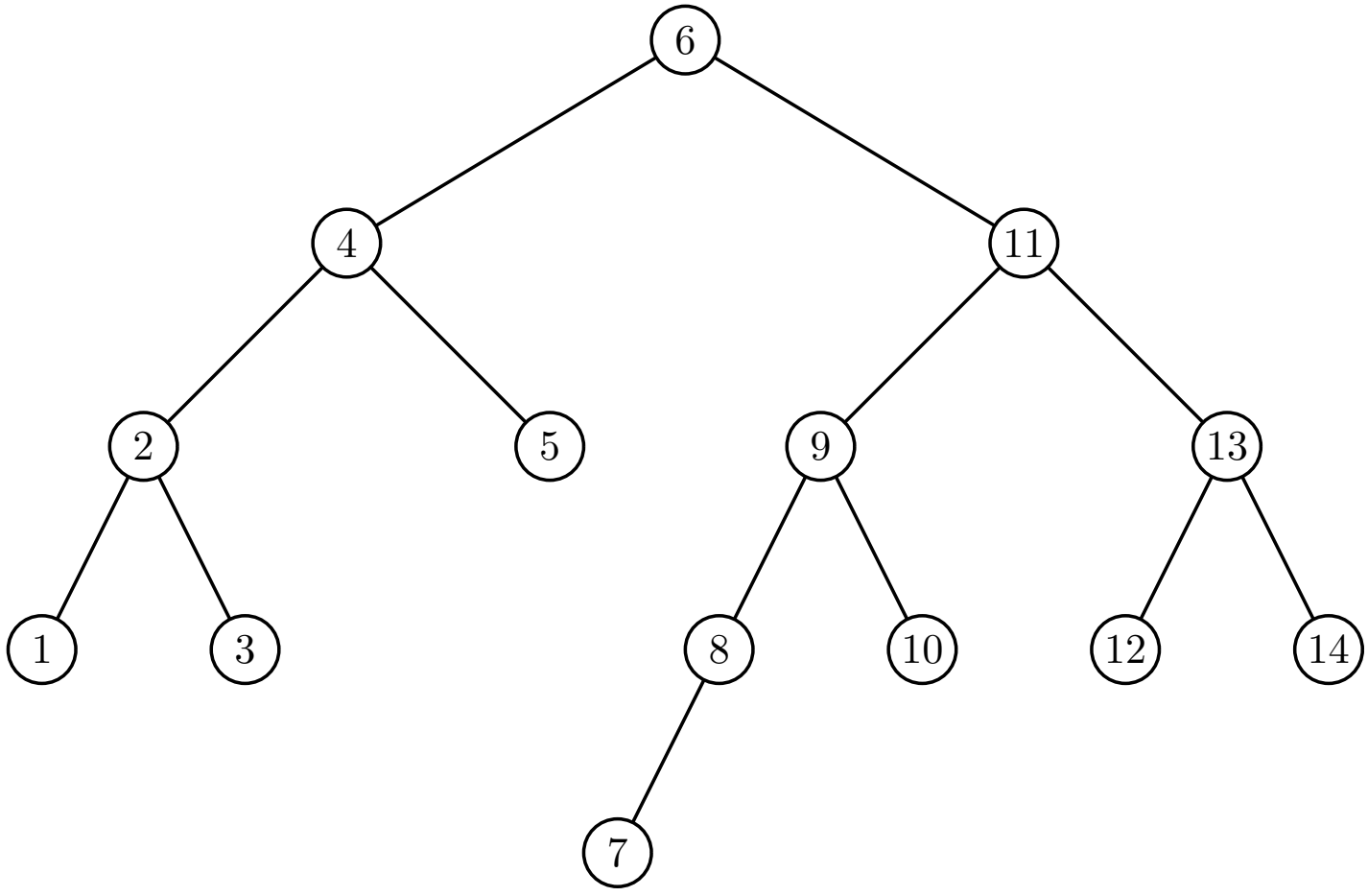
## les arbres binaires de recherche (ABR)

en chaque nœud, l'étiquette est comprise entre

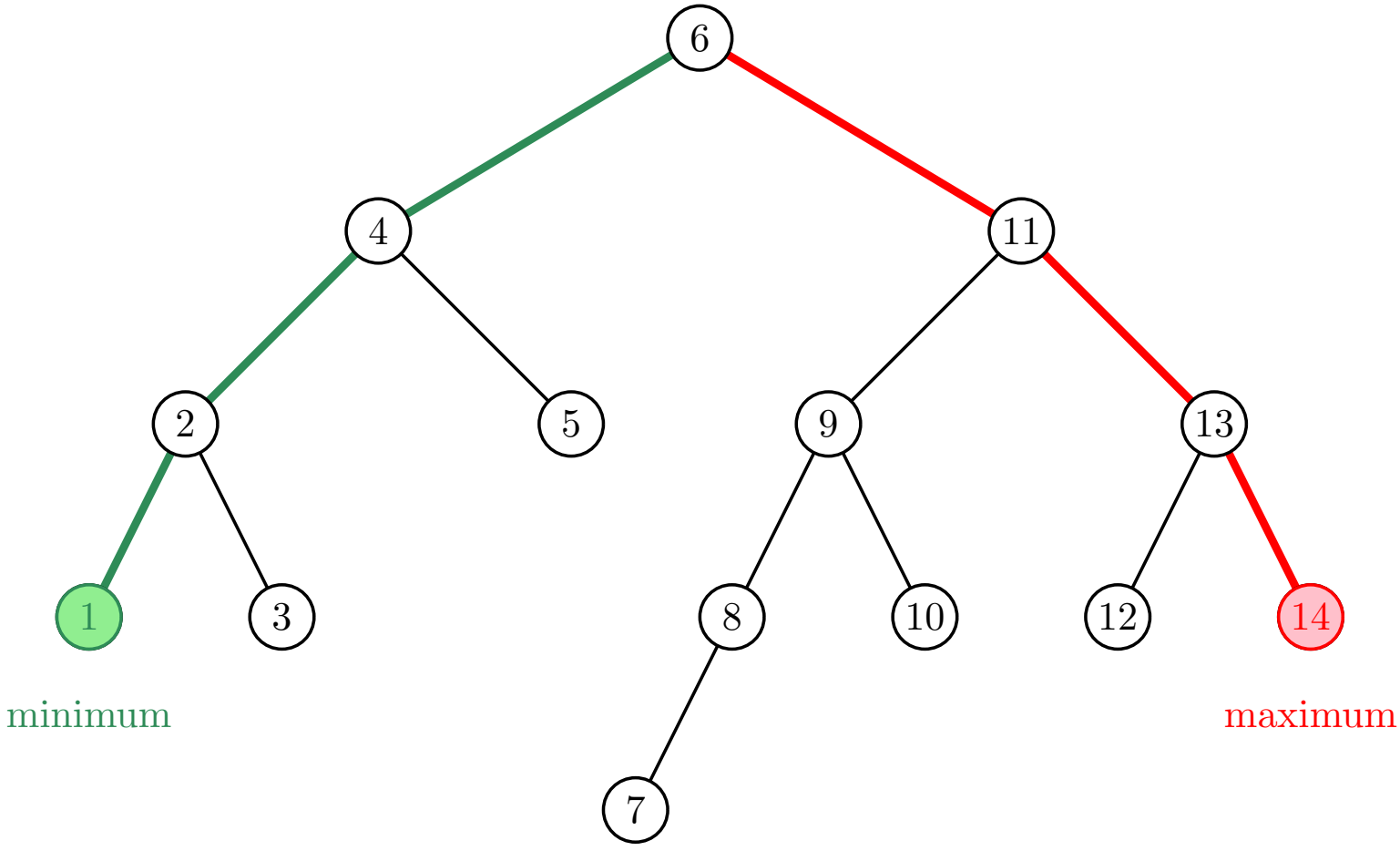
- les étiquettes du sous-arbre gauche (plus petites) et
- celles du sous-arbre droit (plus grandes)



# Arbre binaire de recherche



# Arbre binaire de recherche



# Arbre binaire de recherche