Le conteneur Arbre

Juan Angel Lorenzo del Castillo

ING1-GI Algorithmique procédurale II

EISTI Pau

Mars 2016

Contenu

Contenu

Définitions

Définition de théorie de graphes

Un arbre est un graphe (orienté ou non), connexe et sans cycle.

Définition récursive

Un arbre est un arbre qui possède des liens ou des pointeurs vers d'autres arbres.

- **Arbre non enraciné :** Il n'y a pas d'ordre ou de hiérarchie entre ses éléments.
- **Arbre enraciné :** Arbre hiérarchique dans lequel on peut établir des niveaux. Il existe un noeud (sommet) spécial, la racine, dans lequel il y a un chemin simple de la racine à chaque noeud.
  - On va considérer toujours l'arbre enraciné comme un graphe orienté.

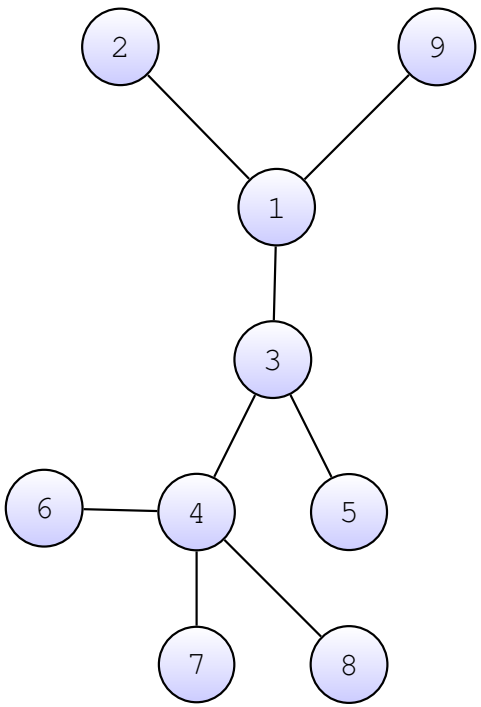
Définitions

Définition de théorie de graphes

Un arbre est un graphe (orienté ou non), connexe et sans cycle.

Définition récursive

Un arbre est un arbre qui possède des liens ou des pointeurs vers d'autres arbres.



- **Arbre non enraciné :** Il n'y a pas d'ordre ou de hiérarchie entre ses éléments.
- **Arbre enraciné :** Arbre hiérarchique dans lequel on peut établir des niveaux. Il existe un noeud (sommet) spécial, la racine, dans lequel il y a un chemin simple de la racine à chaque noeud.
  - On va considérer toujours l'arbre enraciné comme un graphe orienté.

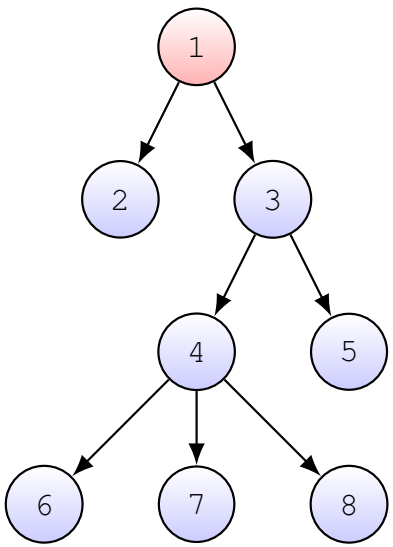
Définitions

Définition de théorie de graphes

Un arbre est un graphe (orienté ou non), connexe et sans cycle.

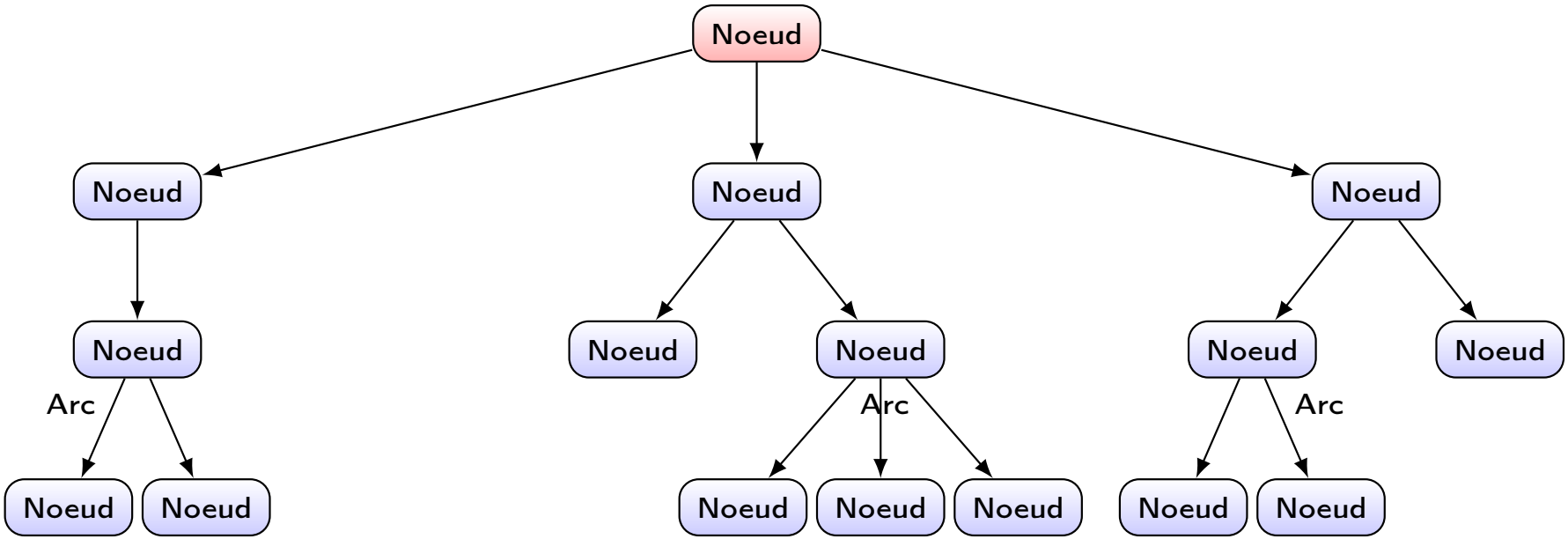
Définition récursive

Un arbre est un arbre qui possède des liens ou des pointeurs vers d'autres arbres.



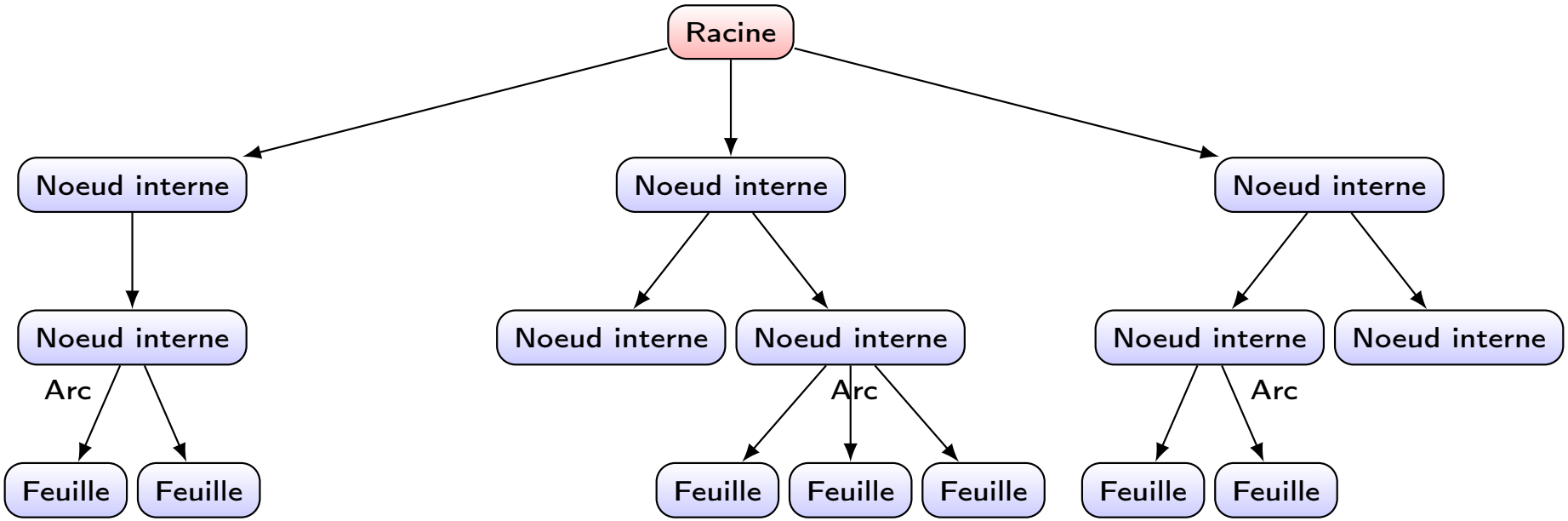
- **Arbre non enraciné :** Il n'y a pas d'ordre ou de hiérarchie entre ses éléments.
- **Arbre enraciné :** Arbre hiérarchique dans lequel on peut établir des niveaux. Il existe un noeud (sommet) spécial, la racine, dans lequel il y a un chemin simple de la racine à chaque noeud.
  - On va considérer toujours l'arbre enraciné comme un graphe orienté.

Terminologie



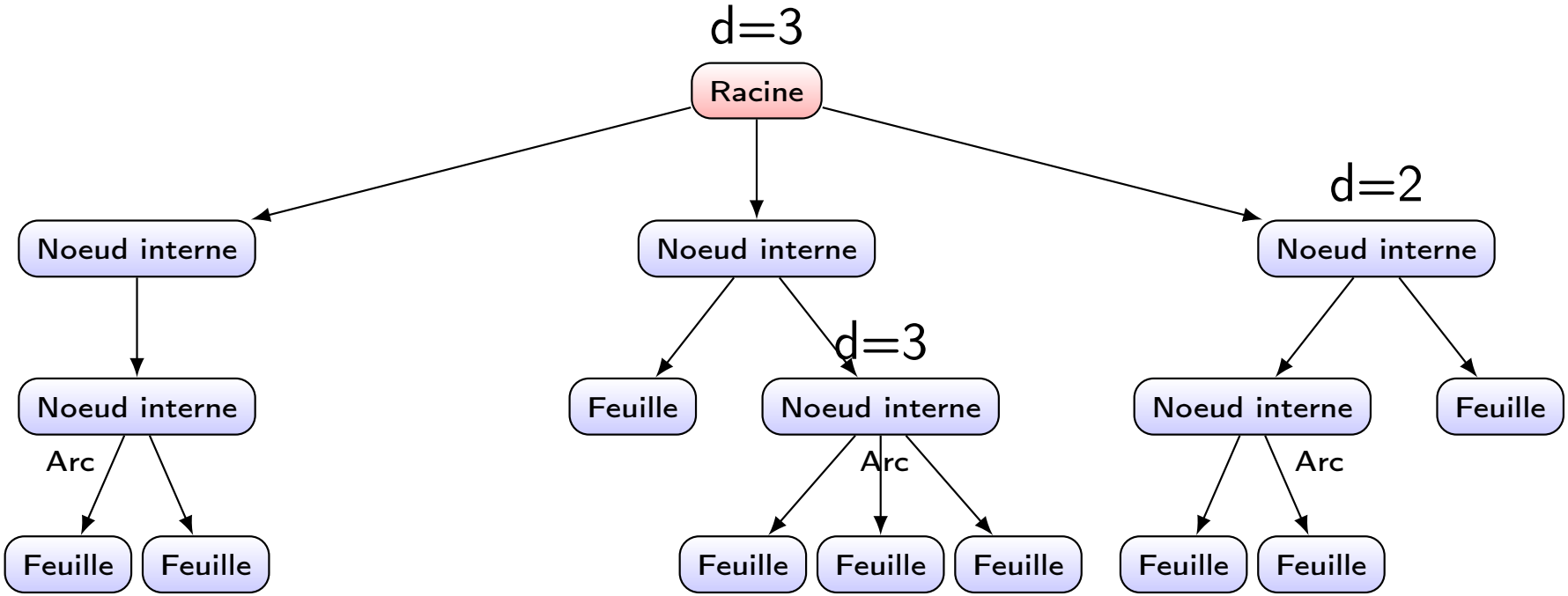
- **noeud externe ou feuille :** c'est un noeud qui n'a pas d'arc sortant.
- **noeud interne :** c'est un nœud qui n'est pas externe.
- **degré** d'un nœud : c'est le nombre d'arc(s) sortants du nœud.
- nœuds **frères** ou **sœurs** : des nœuds qui ont le même parent.
- Un noeud **u** est un **ancêtre** de **w** si w est dans l'arbre enraciné à u.
- Un noeud **w** est un **descendant** de **u** si w est dans l'arbre enraciné à u.

Terminologie



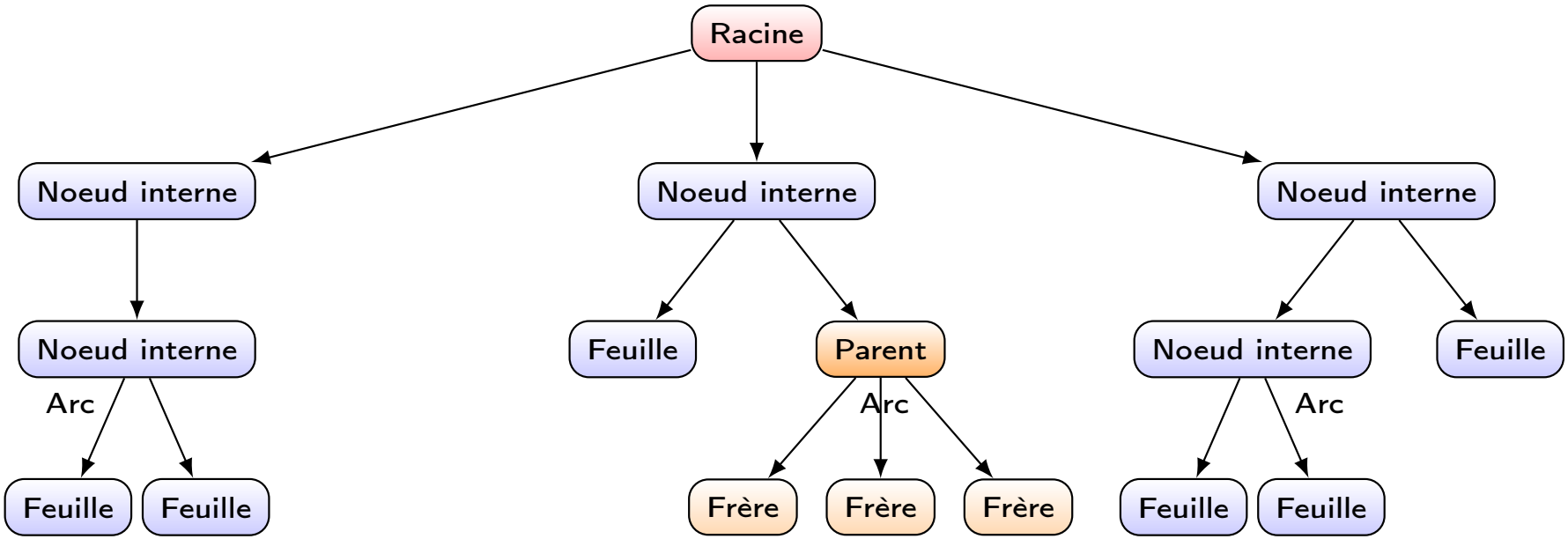
- **noeud externe ou feuille :** c'est un noeud qui n'a pas d'arc sortant.
- **noeud interne :** c'est un nœud qui n'est pas externe.
- **degré** d'un nœud : c'est le nombre d'arc(s) sortants du nœud.
- nœuds **frères** ou **sœurs** : des nœuds qui ont le même parent.
- Un noeud **u** est un **ancêtre** de **w** si w est dans l'arbre enraciné à u.
- Un noeud **w** est un **descendant** de **u** si w est dans l'arbre enraciné à u.

Terminologie



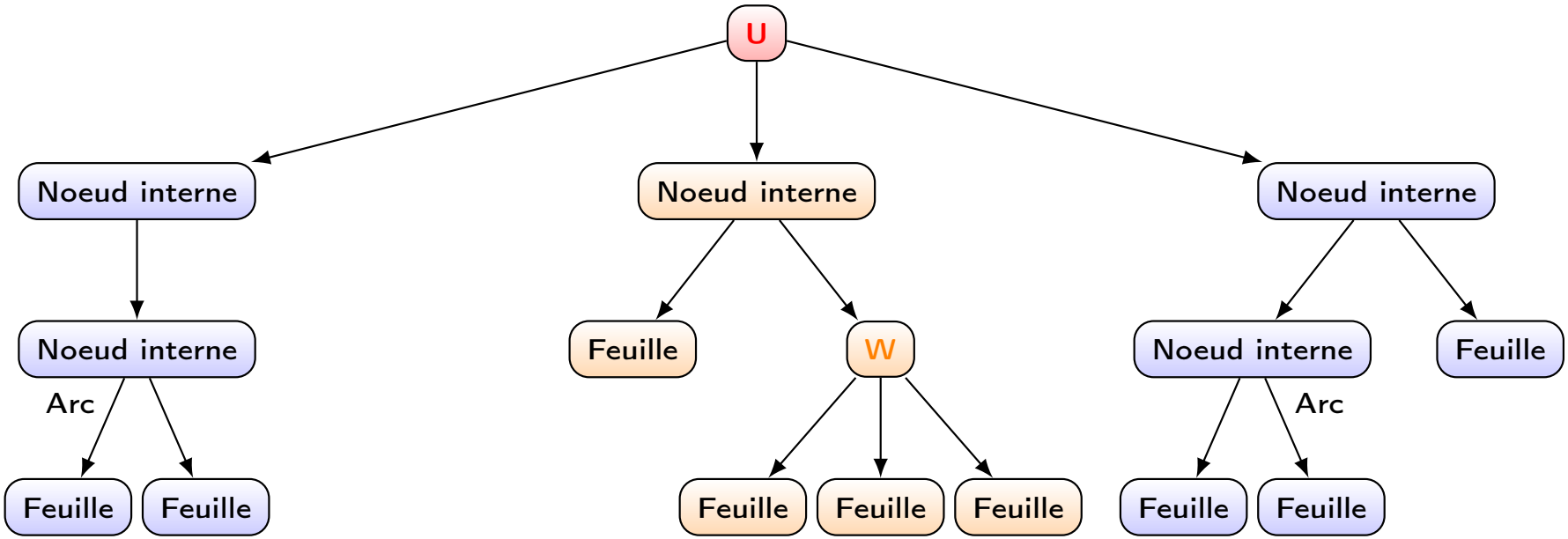
- **noeud externe ou feuille :** c'est un noeud qui n'a pas d'arc sortant.
- **noeud interne :** c'est un nœud qui n'est pas externe.
- **degré** d'un nœud : c'est le nombre d'arc(s) sortants du nœud.
- nœuds **frères** ou **sœurs** : des nœuds qui ont le même parent.
- Un noeud **u** est un **ancêtre** de **w** si w est dans l'arbre enraciné à u.
- Un noeud **w** est un **descendant** de **u** si w est dans l'arbre enraciné à u.

Terminologie



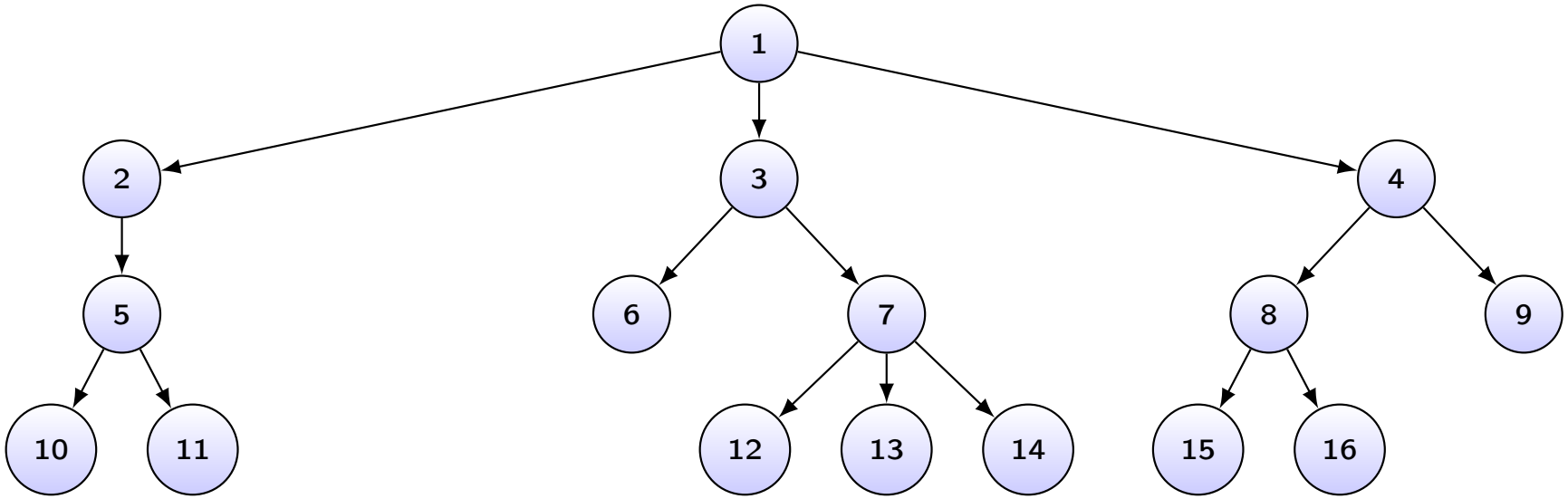
- **noeud externe ou feuille :** c'est un noeud qui n'a pas d'arc sortant.
- **noeud interne :** c'est un nœud qui n'est pas externe.
- **degré** d'un nœud : c'est le nombre d'arc(s) sortants du nœud.
- nœuds **frères** ou **sœurs** : des nœuds qui ont le même parent.
- Un noeud **u** est un **ancêtre** de **w** si w est dans l'arbre enraciné à u.
- Un noeud **w** est un **descendant** de **u** si w est dans l'arbre enraciné à u.

Terminologie



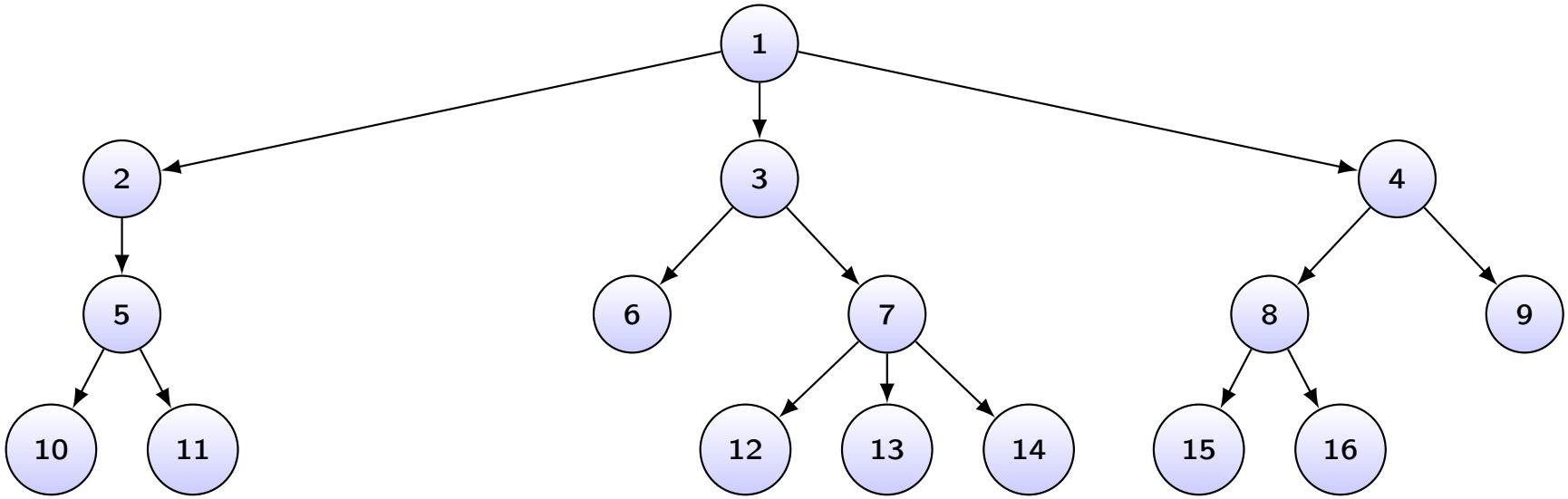
- **noeud externe ou feuille :** c'est un noeud qui n'a pas d'arc sortant.
- **noeud interne :** c'est un nœud qui n'est pas externe.
- **degré** d'un nœud : c'est le nombre d'arc(s) sortants du nœud.
- nœuds **frères** ou **sœurs** : des nœuds qui ont le même parent.
- Un noeud **u** est un **ancêtre** de **w** si w est dans l'arbre enraciné à u.
- Un noeud **w** est un **descendant** de **u** si w est dans l'arbre enraciné à u.

Terminologie



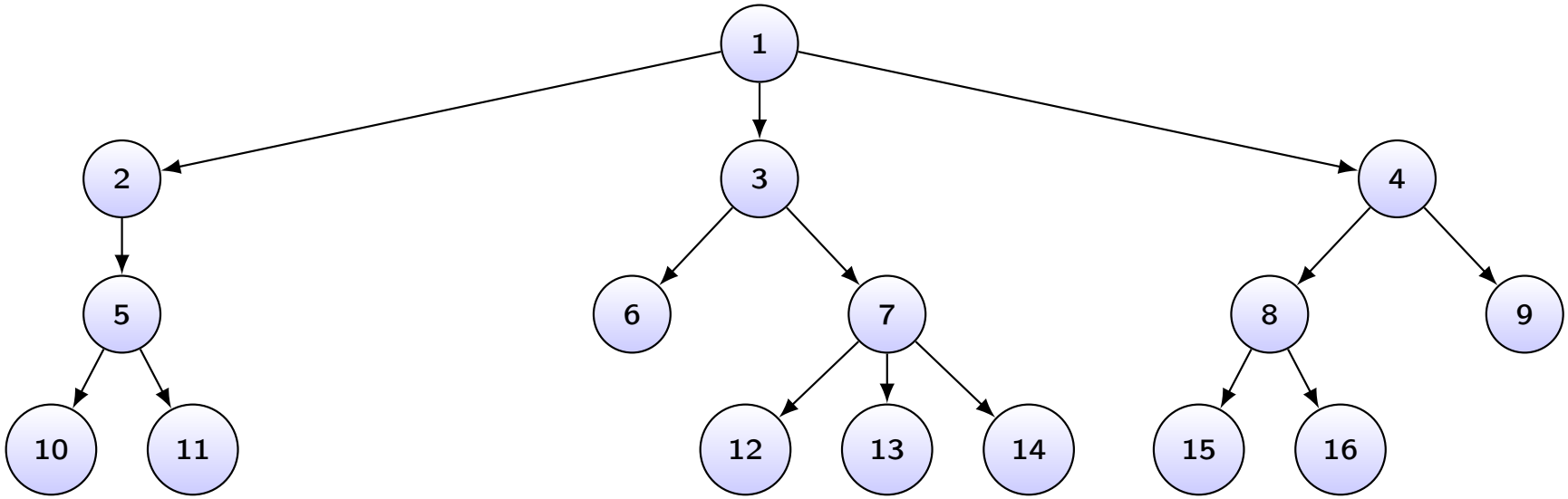
- **Arbre ordonné :** il existe un ordre entre les enfants des nœuds internes.
- **Arbre numéroté :** les enfants de chaque nœud sont étiquetés par des entiers positifs distincts (numéroté $\rightarrow$ ordonné).
  - **i-ème enfant absent :** si aucun enfant n'est étiqueté par i
  - **Arbre k-aire** (ou **d'arité k**) **:** c'est un arbre dont les noeuds ne comporteront qu'au maximum k fils.
  - Un **arbre binaire** est un arbre d'arité 2.

Terminologie



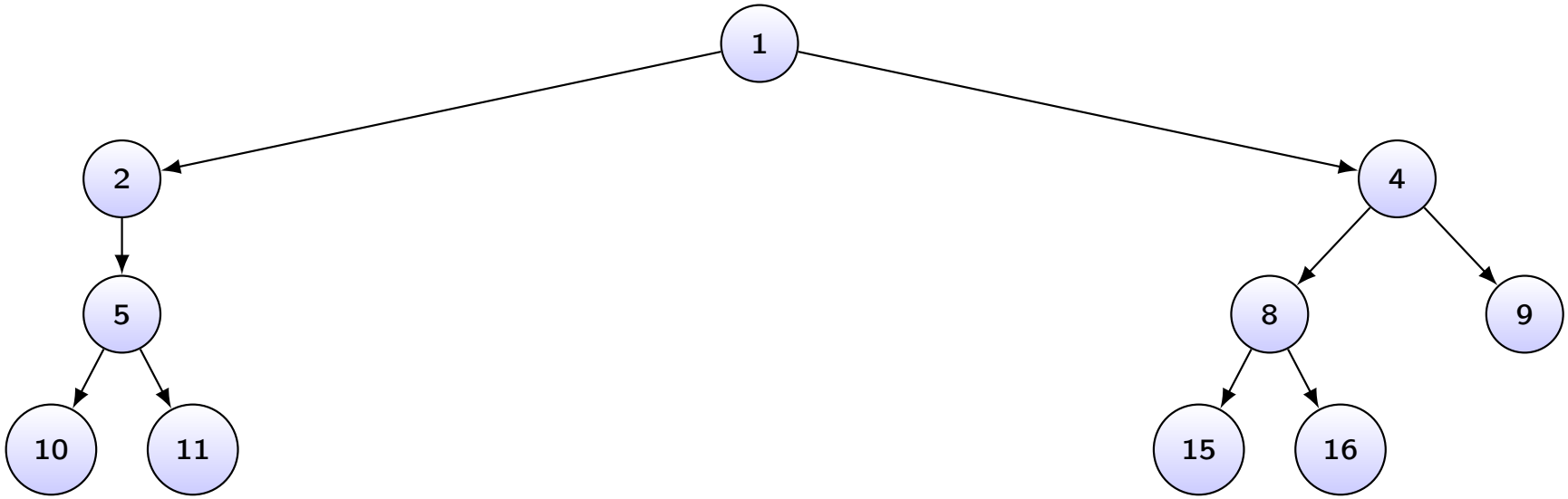
- **Arbre ordonné :** il existe un ordre entre les enfants des nœuds internes.
- **Arbre numéroté :** les enfants de chaque nœud sont étiquetés par des entiers positifs distincts (numéroté $\rightarrow$ ordonné).
  - **i-ème enfant absent :** si aucun enfant n'est étiqueté par i
  - **Arbre k-aire** (ou **d'arité k**) **:** c'est un arbre dont les noeuds ne comporteront qu'au maximum k fils.
  - Un **arbre binaire** est un arbre d'arité 2.

Terminologie



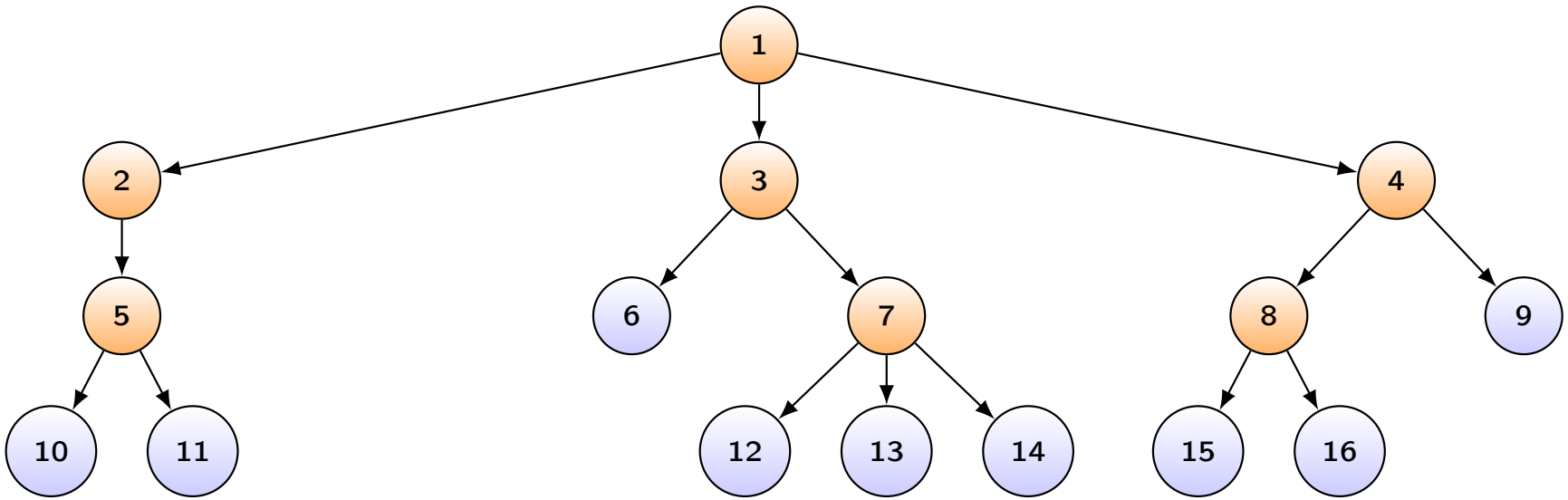
- **Arbre ordonné :** il existe un ordre entre les enfants des nœuds internes.
- **Arbre numéroté :** les enfants de chaque nœud sont étiquetés par des entiers positifs distincts (numéroté $\rightarrow$ ordonné).
  - **i-ème enfant absent :** si aucun enfant n'est étiqueté par i
  - **Arbre k-aire** (ou **d'arité k**) **:** c'est un arbre dont les noeuds ne comporteront qu'au maximum k fils.
  - Un **arbre binaire** est un arbre d'arité 2.

Terminologie



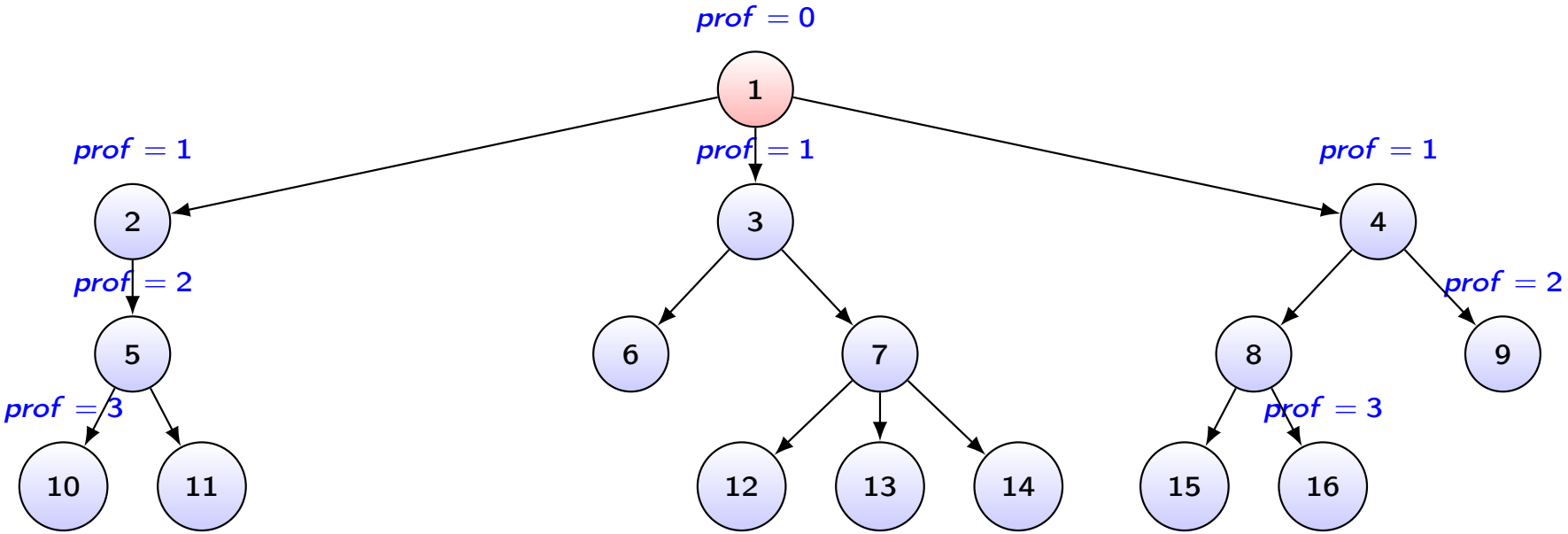
- **Arbre ordonné :** il existe un ordre entre les enfants des nœuds internes.
- **Arbre numéroté :** les enfants de chaque nœud sont étiquetés par des entiers positifs distincts (numéroté $\rightarrow$ ordonné).
  - **i-ème enfant absent :** si aucun enfant n'est étiqueté par i
  - **Arbre k-aire** (ou **d'arité k**) **:** c'est un arbre dont les noeuds ne comporteront qu'au maximum k fils.
  - Un **arbre binaire** est un arbre d'arité 2.

Terminologie



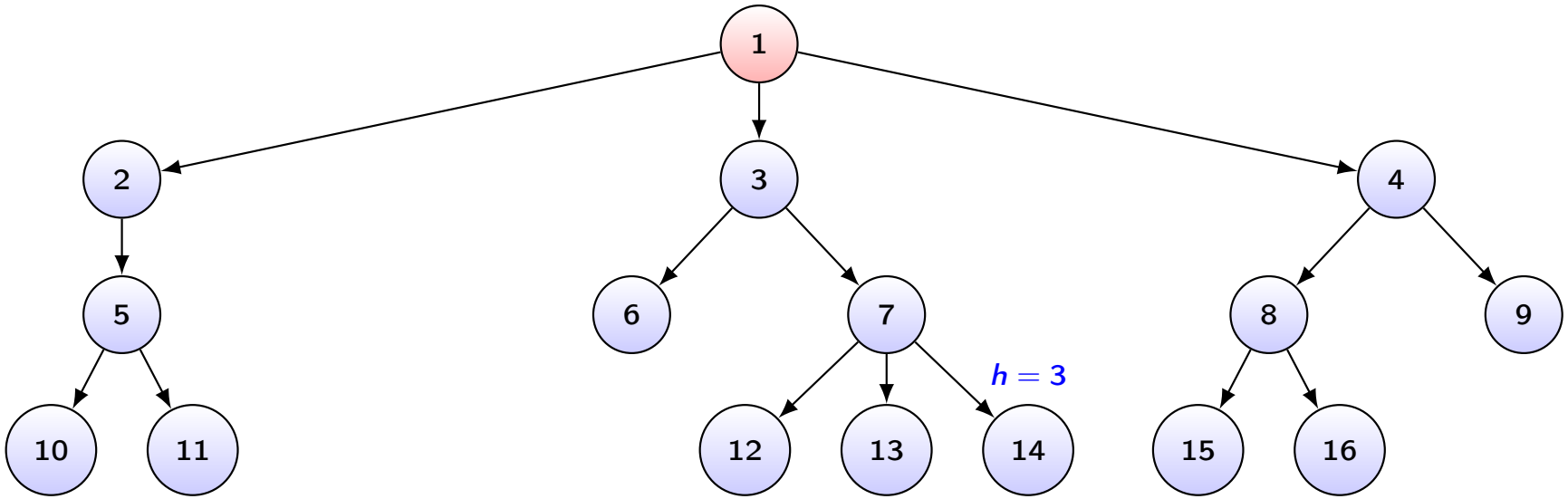
- **Taille d'un arbre :** c'est le nombre de nœuds internes de l'arbre.
- **Profondeur d'un nœud :** c'est la distance du chemin qui relie la racine au nœud.
- **Hauteur d'un arbre :** c'est la profondeur maximale de ses nœuds.
- Théorème : Si on note h la hauteur d'un arbre k-aire et n la taille d'un arbre alors
  - $h = n$ dans le pire des cas (h la plus grande)
  - $h = \lceil log_k(n) \rceil$ dans le meilleur des cas

Terminologie



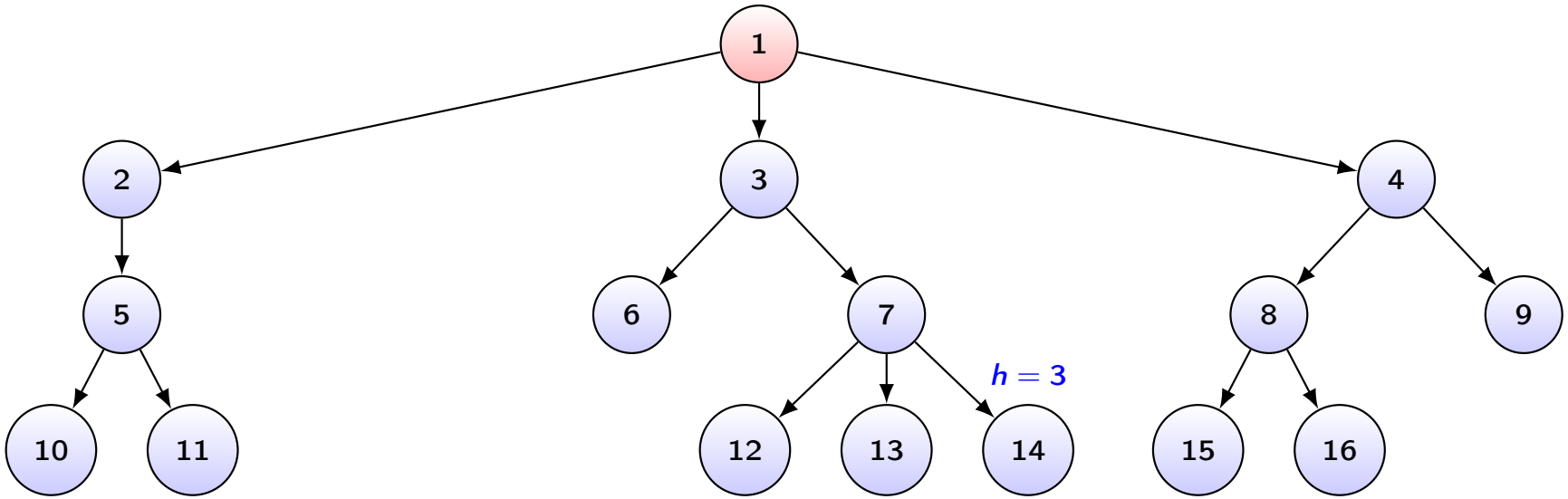
- **Taille d'un arbre :** c'est le nombre de nœuds internes de l'arbre.
- **Profondeur d'un nœud :** c'est la distance du chemin qui relie la racine au nœud.
- **Hauteur d'un arbre :** c'est la profondeur maximale de ses nœuds.
- Théorème : Si on note h la hauteur d'un arbre k-aire et n la taille d'un arbre alors
  - $h = n$ dans le pire des cas (h la plus grande)
  - $h = \lceil log_k(n) \rceil$ dans le meilleur des cas

Terminologie



- **Taille d'un arbre :** c'est le nombre de nœuds internes de l'arbre.
- **Profondeur d'un nœud :** c'est la distance du chemin qui relie la racine au nœud.
- **Hauteur d'un arbre :** c'est la profondeur maximale de ses nœuds.
- Théorème : Si on note h la hauteur d'un arbre k-aire et n la taille d'un arbre alors
  - $h = n$ dans le pire des cas (h la plus grande)
  - $h = \lceil log_k(n) \rceil$ dans le meilleur des cas

Terminologie



- **Taille d'un arbre :** c'est le nombre de nœuds internes de l'arbre.
- **Profondeur d'un nœud :** c'est la distance du chemin qui relie la racine au nœud.
- **Hauteur d'un arbre :** c'est la profondeur maximale de ses nœuds.
- Théorème : Si on note h la hauteur d'un arbre k-aire et n la taille d'un arbre alors
  - $h = n$ dans le pire des cas (h la plus grande)
  - $h = \lceil log_k(n) \rceil$ dans le meilleur des cas

Contenu

Parcours d'un arbre

Parcours

Algorithme qui permet de visiter tous les noeuds d'un arbre. Il appelle une fonction, méthode où procédure sur tous les noeuds (ou les sous arbres).

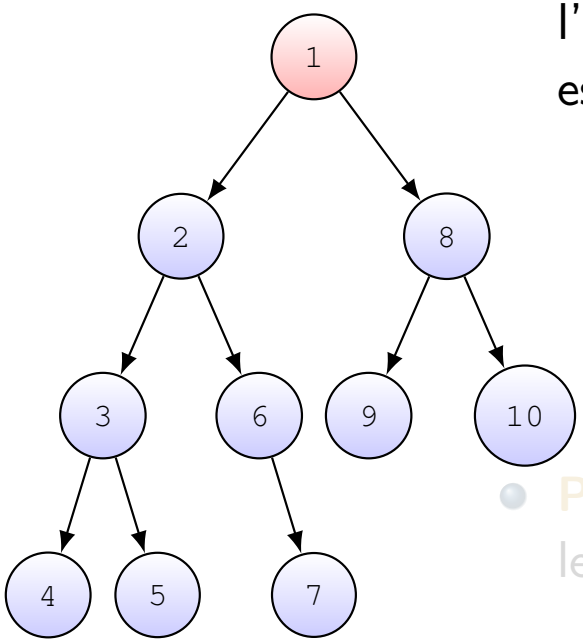
- **Parcours en profondeur :** lorsque, systématiquement, si l'arbre n'est pas vide, le parcours de l'un des deux sous-arbres est terminé avant que ne commence celui de l'autre.
  - On explore jusqu'au bout une branche pour passer à la suivante (on va vers les fils avant d'aller vers les frères).

- **Parcours en largueur :** lorsqu'il procède en croissant selon les niveaux.
  - On explore les noeuds un par un sur un même niveau. On passe ensuite sur le niveau suivant, et ainsi de suite (on va vers les frères avant de parcourir les fils).

Parcours d'un arbre

Parcours

Algorithme qui permet de visiter tous les noeuds d'un arbre. Il appelle une fonction, méthode où procédure sur tous les noeuds (ou les sous arbres).

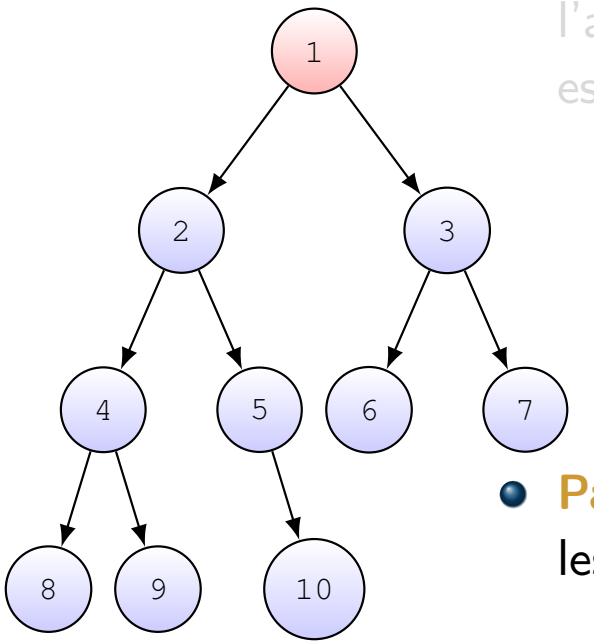


- **Parcours en profondeur :** lorsque, systématiquement, si l'arbre n'est pas vide, le parcours de l'un des deux sous-arbres est terminé avant que ne commence celui de l'autre.
  - On explore jusqu'au bout une branche pour passer à la suivante (on va vers les fils avant d'aller vers les frères).
- **Parcours en largueur :** lorsqu'il procède en croissant selon les niveaux.
  - On explore les noeuds un par un sur un même niveau. On passe ensuite sur le niveau suivant, et ainsi de suite (on va vers les frères avant de parcourir les fils).

Parcours d'un arbre

Parcours

Algorithme qui permet de visiter tous les noeuds d'un arbre. Il appelle une fonction, méthode où procédure sur tous les noeuds (ou les sous arbres).



- **Parcours en profondeur :** lorsque, systématiquement, si l'arbre n'est pas vide, le parcours de l'un des deux sous-arbres est terminé avant que ne commence celui de l'autre.
  - On explore jusqu'au bout une branche pour passer à la suivante (on va vers les fils avant d'aller vers les frères).

- **Parcours en largeur :** lorsqu'il procède en croissant selon les niveaux.
  - On explore les noeuds un par un sur un même niveau. On passe ensuite sur le niveau suivant, et ainsi de suite (on va vers les frères avant de parcourir les fils).

Parcours en profondeur

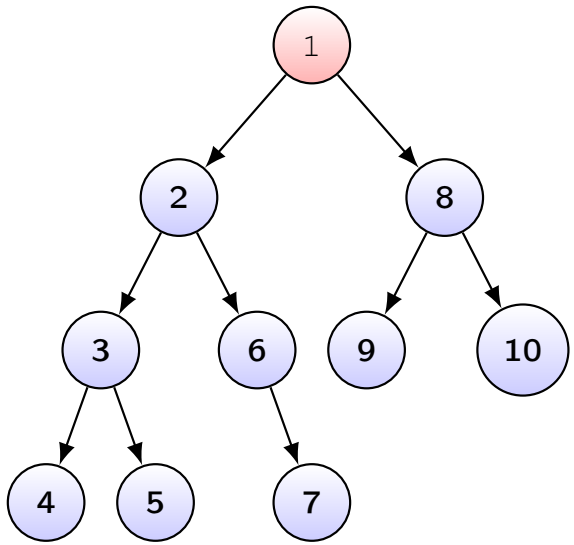
Types

Préfixe : chaque nœud est visité avant leurs fils.

1. On visite la racine de l'arbre
2. Parcours préfixe des sous-arbres de gauche à droite

Dans un arbre binaire, on l'appelle parcours R G D (Racine Gauche Droit).

Parcours : `1,2,3,4,5,6,7,8,9,10`



(On considère toujours un parcours gauche à droite)

Parcours en profondeur

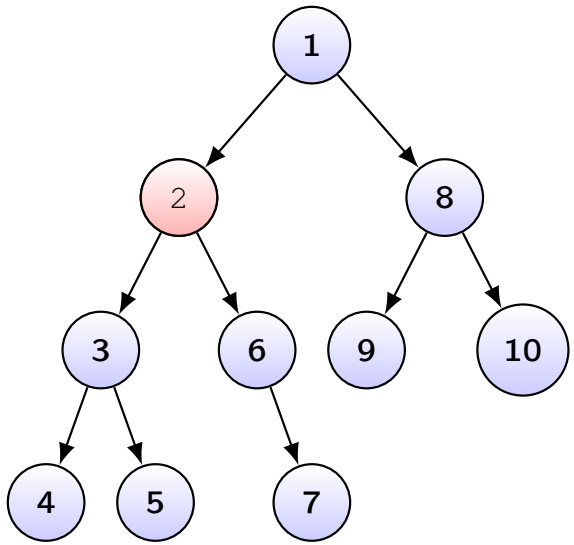
Types

Préfixe : chaque nœud est visité avant leurs fils.

1. On visite la racine de l'arbre
2. Parcours préfixe des sous-arbres de gauche à droite

Dans un arbre binaire, on l'appelle parcours R G D (Racine Gauche Droit).

Parcours : `1,2,3,4,5,6,7,8,9,10`



(On considère toujours un parcours gauche à droite)

Parcours en profondeur

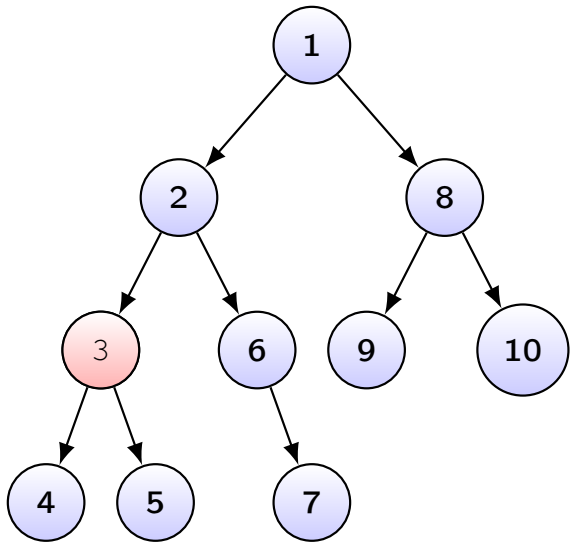
Types

Préfixe : chaque nœud est visité avant leurs fils.

1. On visite la racine de l'arbre
2. Parcours préfixe des sous-arbres de gauche à droite

Dans un arbre binaire, on l'appelle parcours R G D (Racine Gauche Droit).

Parcours : `1,2,3,4,5,6,7,8,9,10`



(On considère toujours un parcours gauche à droite)

Parcours en profondeur

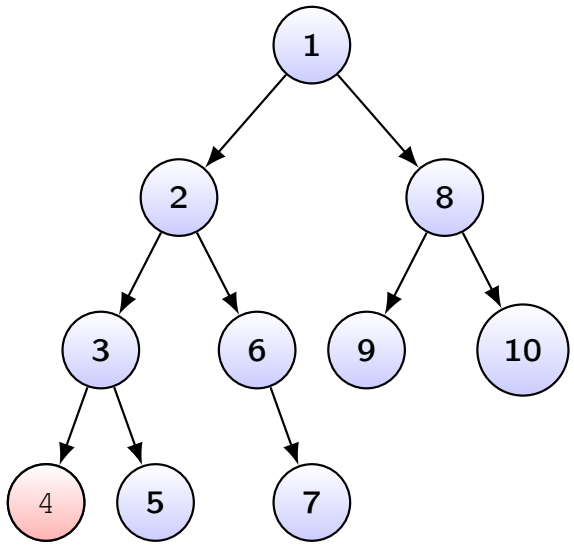
Types

Préfixe : chaque nœud est visité avant leurs fils.

1. On visite la racine de l'arbre
2. Parcours préfixe des sous-arbres de gauche à droite

Dans un arbre binaire, on l'appelle parcours R G D (Racine Gauche Droit).

Parcours : `1,2,3,4,5,6,7,8,9,10`



(On considère toujours un parcours gauche à droite)

Parcours en profondeur

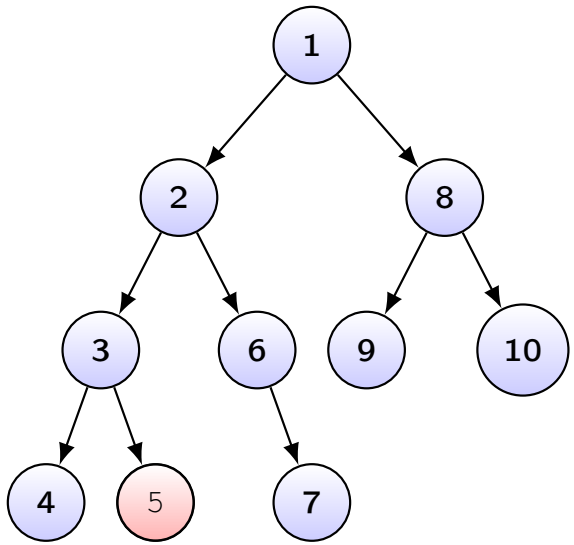
Types

Préfixe : chaque nœud est visité avant leurs fils.

1. On visite la racine de l'arbre
2. Parcours préfixe des sous-arbres de gauche à droite

Dans un arbre binaire, on l'appelle parcours R G D (Racine Gauche Droit).

Parcours : `1,2,3,4,5,6,7,8,9,10`



(On considère toujours un parcours gauche à droite)

Parcours en profondeur

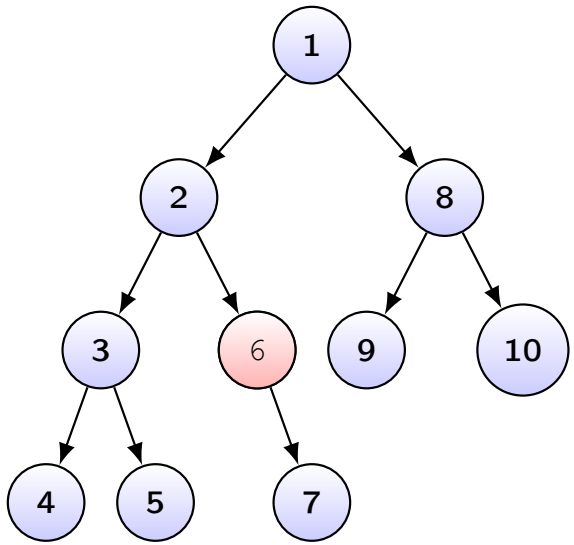
Types

Préfixe : chaque nœud est visité avant leurs fils.

1. On visite la racine de l'arbre
2. Parcours préfixe des sous-arbres de gauche à droite

Dans un arbre binaire, on l'appelle parcours R G D (Racine Gauche Droit).

Parcours : `1,2,3,4,5,6,7,8,9,10`



(On considère toujours un parcours gauche à droite)

Parcours en profondeur

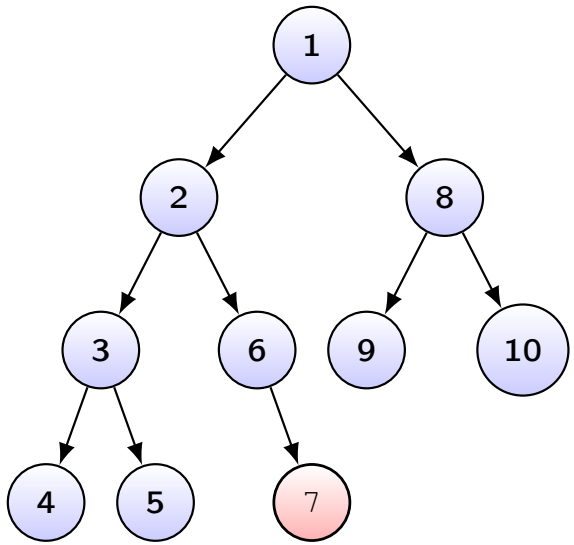
Types

Préfixe : chaque nœud est visité avant leurs fils.

1. On visite la racine de l'arbre
2. Parcours préfixe des sous-arbres de gauche à droite

Dans un arbre binaire, on l'appelle parcours R G D (Racine Gauche Droit).

Parcours : `1,2,3,4,5,6,7,8,9,10`



(On considère toujours un parcours gauche à droite)

Parcours en profondeur

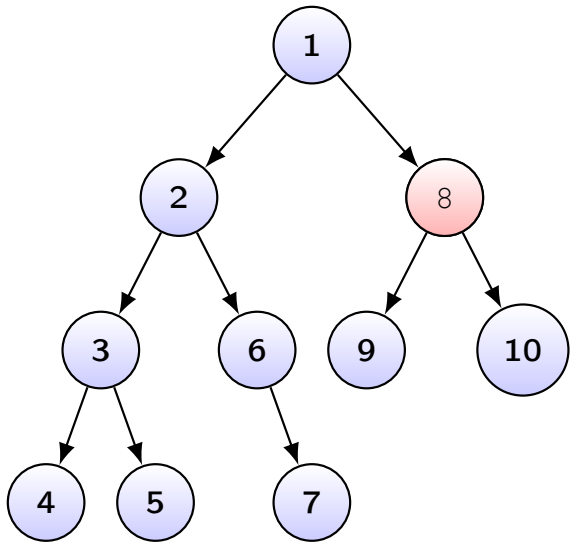
Types

Préfixe : chaque nœud est visité avant leurs fils.

1. On visite la racine de l'arbre
2. Parcours préfixe des sous-arbres de gauche à droite

Dans un arbre binaire, on l'appelle parcours R G D (Racine Gauche Droit).

Parcours : `1,2,3,4,5,6,7,8,9,10`



(On considère toujours un parcours gauche à droite)

Parcours en profondeur

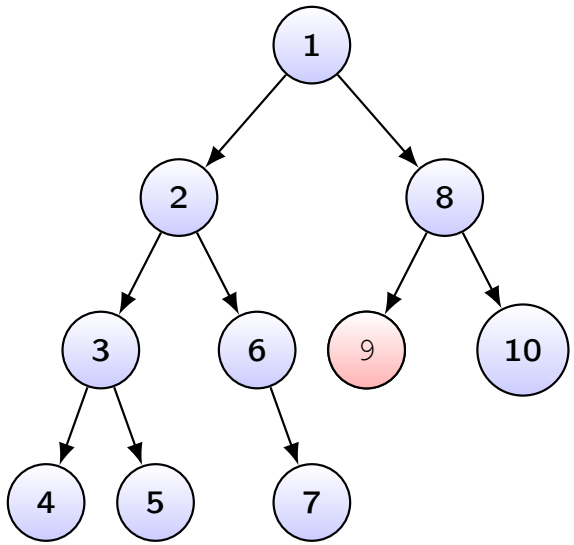
Types

Préfixe : chaque nœud est visité avant leurs fils.

1. On visite la racine de l'arbre
2. Parcours préfixe des sous-arbres de gauche à droite

Dans un arbre binaire, on l'appelle parcours R G D (Racine Gauche Droit).

Parcours : `1,2,3,4,5,6,7,8,9,10`



(On considère toujours un parcours gauche à droite)

Parcours en profondeur

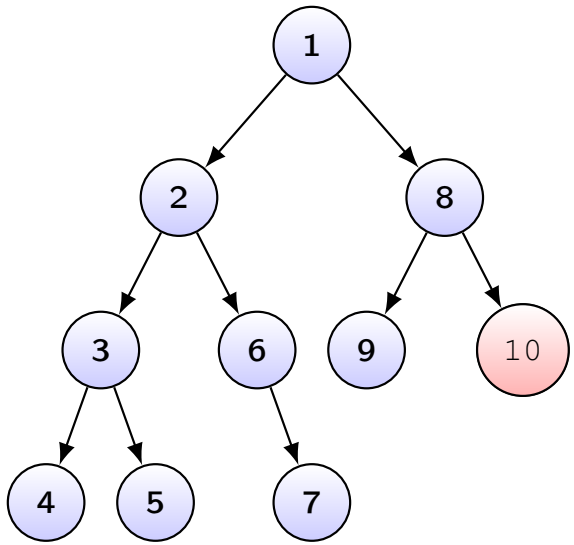
Types

Préfixe : chaque nœud est visité avant leurs fils.

1. On visite la racine de l'arbre
2. Parcours préfixe des sous-arbres de gauche à droite

Dans un arbre binaire, on l'appelle parcours R G D (Racine Gauche Droit).

Parcours : `1,2,3,4,5,6,7,8,9,10`



(On considère toujours un parcours gauche à droite)

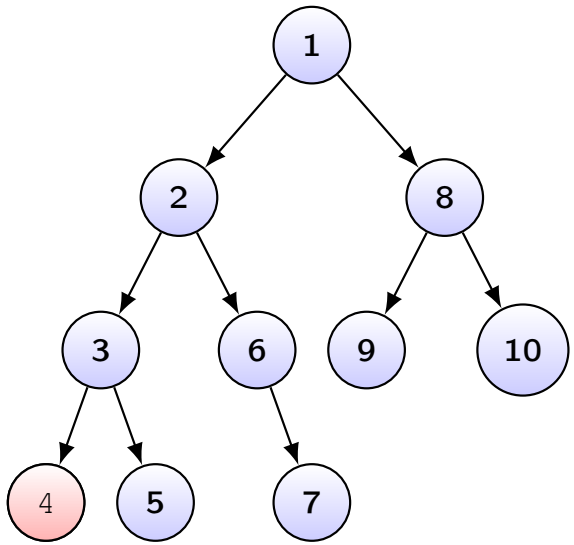
Parcours en profondeur

Types

Postfixe (ou suffixe) : on visite chaque nœud après avoir visité chacun de ses fils.

1. Parcours postfixe des sous-arbres de gauche à droite
2. On visite la racine de l'arbre

Dans un arbre binaire, on l'appelle parcours G D R (Gauche Droit Racine).



Parcours : `4,5,3,7,6,2,9,10,8,1`

(On considère toujours un parcours gauche à droite)

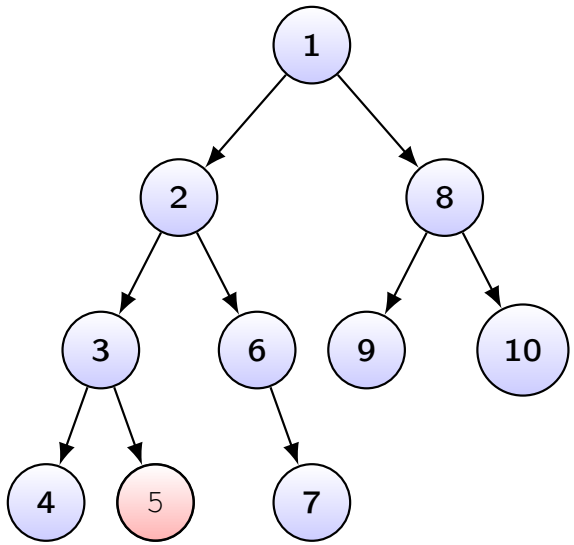
Parcours en profondeur

Types

Postfixe (ou suffixe) : on visite chaque nœud après avoir visité chacun de ses fils.

1. Parcours postfixe des sous-arbres de gauche à droite
2. On visite la racine de l'arbre

Dans un arbre binaire, on l'appelle parcours G D R (Gauche Droit Racine).



Parcours : `4,5,3,7,6,2,9,10,8,1`

(On considère toujours un parcours gauche à droite)

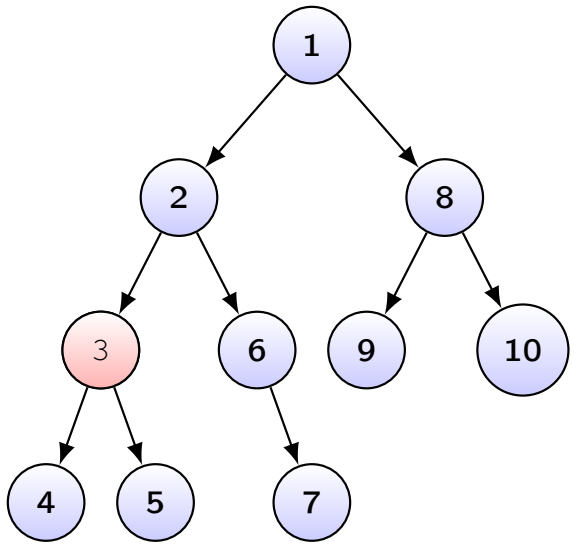
Parcours en profondeur

Types

Postfixe (ou suffixe) : on visite chaque nœud après avoir visité chacun de ses fils.

1. Parcours postfixe des sous-arbres de gauche à droite
2. On visite la racine de l'arbre

Dans un arbre binaire, on l'appelle parcours G D R (Gauche Droit Racine).



Parcours : `4,5,3,7,6,2,9,10,8,1`

(On considère toujours un parcours gauche à droite)

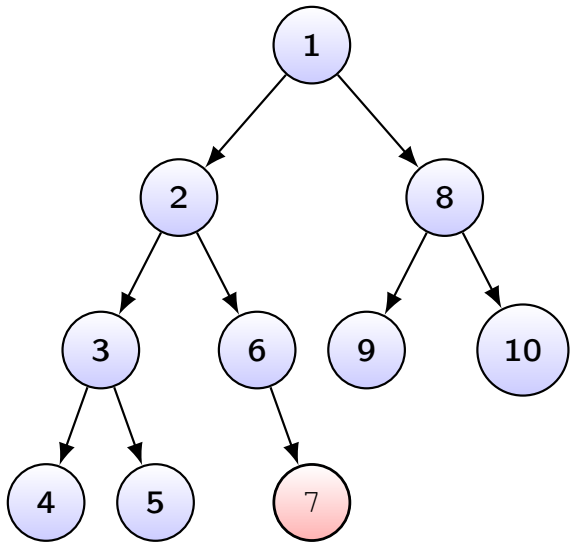
Parcours en profondeur

Types

Postfixe (ou suffixe) : on visite chaque nœud après avoir visité chacun de ses fils.

1. Parcours postfixe des sous-arbres de gauche à droite
2. On visite la racine de l'arbre

Dans un arbre binaire, on l'appelle parcours G D R (Gauche Droit Racine).



Parcours : `4,5,3,7,6,2,9,10,8,1`

(On considère toujours un parcours gauche à droite)

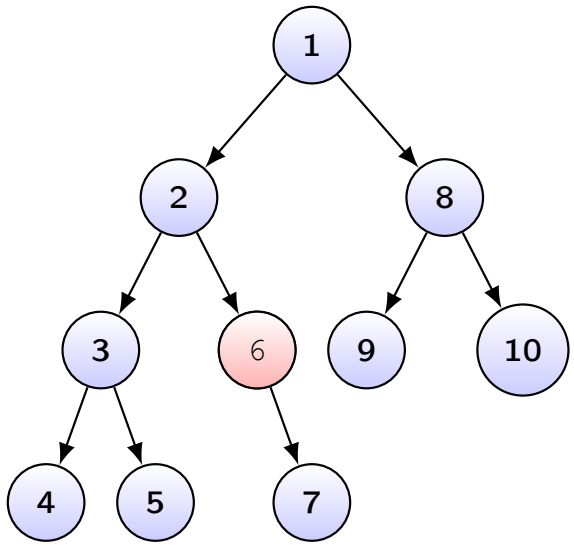
Parcours en profondeur

Types

Postfixe (ou suffixe) : on visite chaque nœud après avoir visité chacun de ses fils.

1. Parcours postfixe des sous-arbres de gauche à droite
2. On visite la racine de l'arbre

Dans un arbre binaire, on l'appelle parcours G D R (Gauche Droit Racine).



Parcours : `4,5,3,7,6,2,9,10,8,1`

(On considère toujours un parcours gauche à droite)

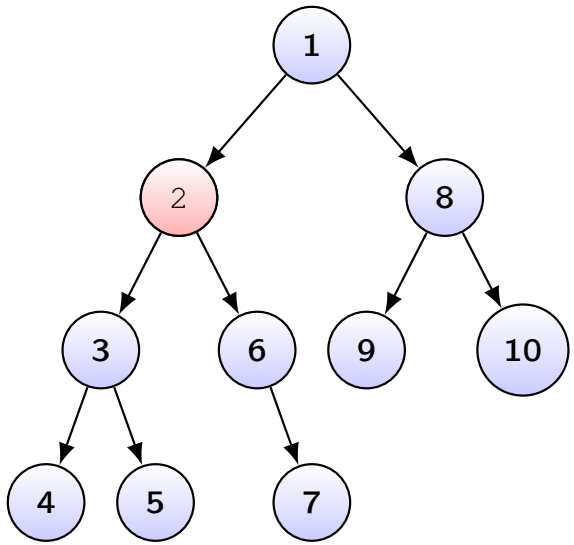
Parcours en profondeur

Types

Postfixe (ou suffixe) : on visite chaque nœud après avoir visité chacun de ses fils.

1. Parcours postfixe des sous-arbres de gauche à droite
2. On visite la racine de l'arbre

Dans un arbre binaire, on l'appelle parcours G D R (Gauche Droit Racine).



Parcours : `4,5,3,7,6,2,9,10,8,1`

(On considère toujours un parcours gauche à droite)

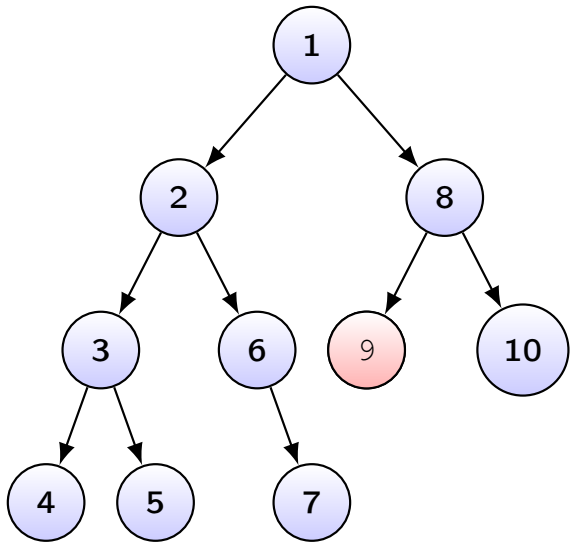
Parcours en profondeur

Types

Postfixe (ou suffixe) : on visite chaque nœud après avoir visité chacun de ses fils.

1. Parcours postfixe des sous-arbres de gauche à droite
2. On visite la racine de l'arbre

Dans un arbre binaire, on l'appelle parcours G D R (Gauche Droit Racine).



Parcours : `4,5,3,7,6,2,9,10,8,1`

(On considère toujours un parcours gauche à droite)

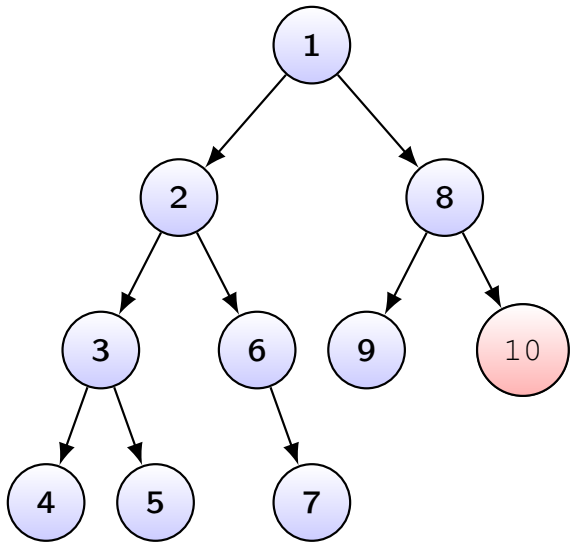
Parcours en profondeur

Types

Postfixe (ou suffixe) : on visite chaque nœud après avoir visité chacun de ses fils.

1. Parcours postfixe des sous-arbres de gauche à droite
2. On visite la racine de l'arbre

Dans un arbre binaire, on l'appelle parcours G D R (Gauche Droit Racine).



Parcours : `4,5,3,7,6,2,9,10,8,1`

(On considère toujours un parcours gauche à droite)

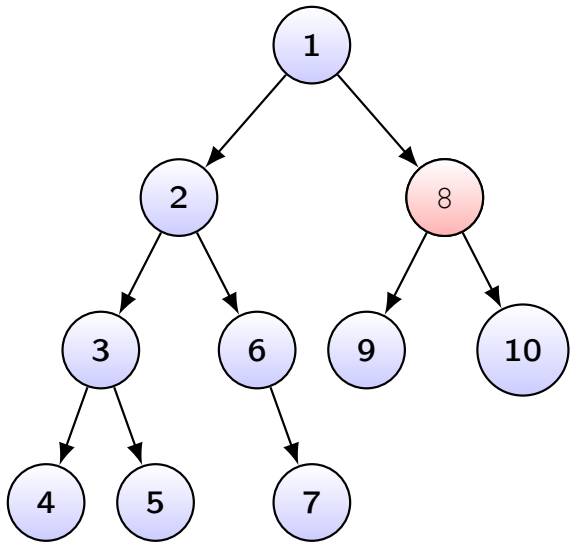
Parcours en profondeur

Types

Postfixe (ou suffixe) : on visite chaque nœud après avoir visité chacun de ses fils.

1. Parcours postfixe des sous-arbres de gauche à droite
2. On visite la racine de l'arbre

Dans un arbre binaire, on l'appelle parcours G D R (Gauche Droit Racine).



Parcours : `4,5,3,7,6,2,9,10,8,1`

(On considère toujours un parcours gauche à droite)

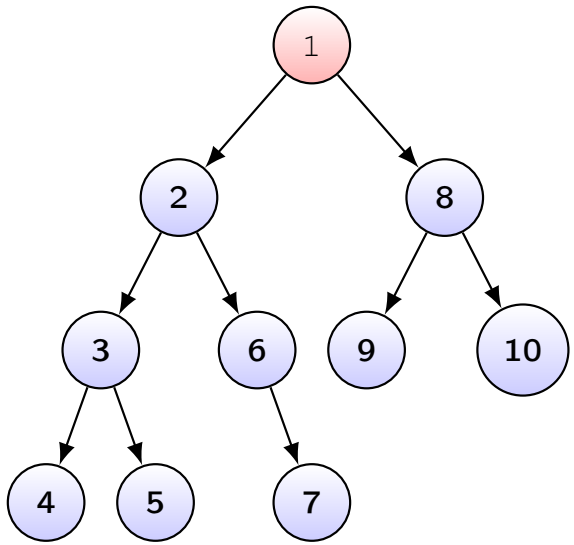
Parcours en profondeur

Types

Postfixe (ou suffixe) : on visite chaque nœud après avoir visité chacun de ses fils.

1. Parcours postfixe des sous-arbres de gauche à droite
2. On visite la racine de l'arbre

Dans un arbre binaire, on l'appelle parcours G D R (Gauche Droit Racine).



Parcours : `4,5,3,7,6,2,9,10,8,1`

(On considère toujours un parcours gauche à droite)

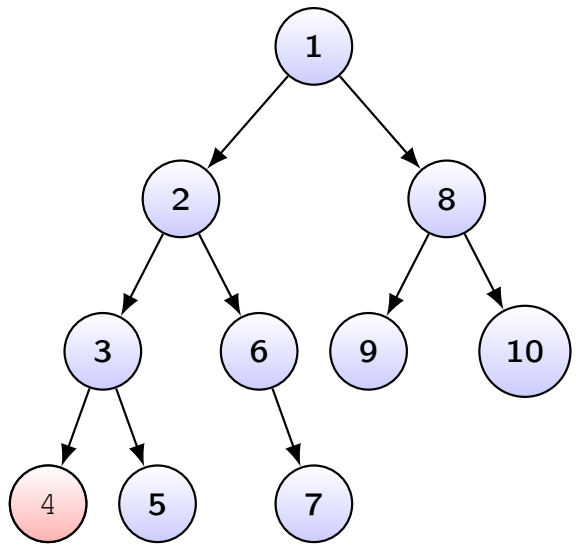
Parcours en profondeur

Types

Infixe (ou symétrique) : (dans un arbre binaire seulement) on visite chaque nœud après son fils à gauche et avant son fils à droite.

1. Parcours infixe du sous-arbre gauche
2. On visite la racine de l'arbre
3. Parcourt infixe du sous-arbre droit

On l'appelle parcours G R D (Gauche Racine Droit).



Parcours : `4,3,5,2,6,7,1,9,8,10`

(On considère toujours un parcours gauche à droite)

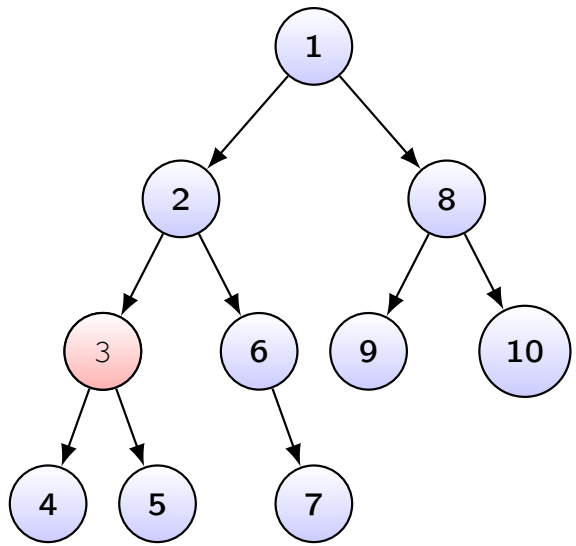
Parcours en profondeur

Types

Infixe (ou symétrique) : (dans un arbre binaire seulement) on visite chaque nœud après son fils à gauche et avant son fils à droite.

1. Parcours infixe du sous-arbre gauche
2. On visite la racine de l'arbre
3. Parcourt infixe du sous-arbre droit

On l'appelle parcours G R D (Gauche Racine Droit).



Parcours : `4,3,5,2,6,7,1,9,8,10`

(On considère toujours un parcours gauche à droite)

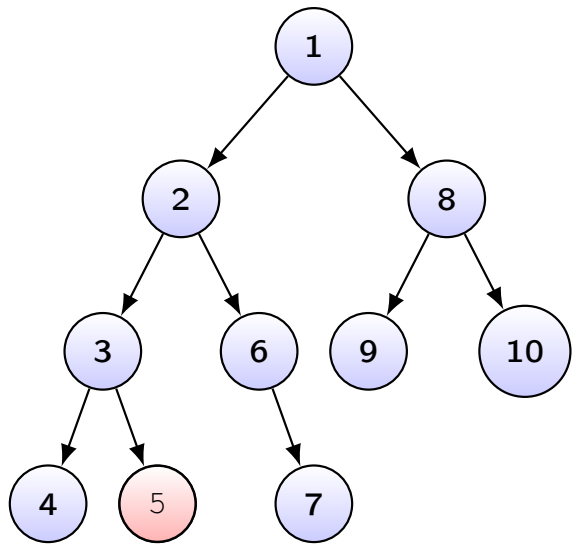
Parcours en profondeur

Types

Infixe (ou symétrique) : (dans un arbre binaire seulement) on visite chaque nœud après son fils à gauche et avant son fils à droite.

1. Parcours infixe du sous-arbre gauche
2. On visite la racine de l'arbre
3. Parcourt infixe du sous-arbre droit

On l'appelle parcours G R D (Gauche Racine Droit).



Parcours : `4,3,5,2,6,7,1,9,8,10`

(On considère toujours un parcours gauche à droite)

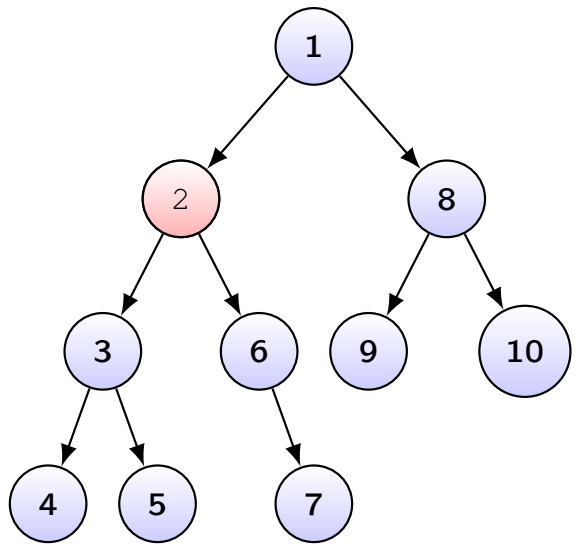
Parcours en profondeur

Types

Infixe (ou symétrique) : (dans un arbre binaire seulement) on visite chaque nœud après son fils à gauche et avant son fils à droite.

1. Parcours infixe du sous-arbre gauche
2. On visite la racine de l'arbre
3. Parcourt infixe du sous-arbre droit

On l'appelle parcours G R D (Gauche Racine Droit).



Parcours : `4,3,5,2,6,7,1,9,8,10`

(On considère toujours un parcours gauche à droite)

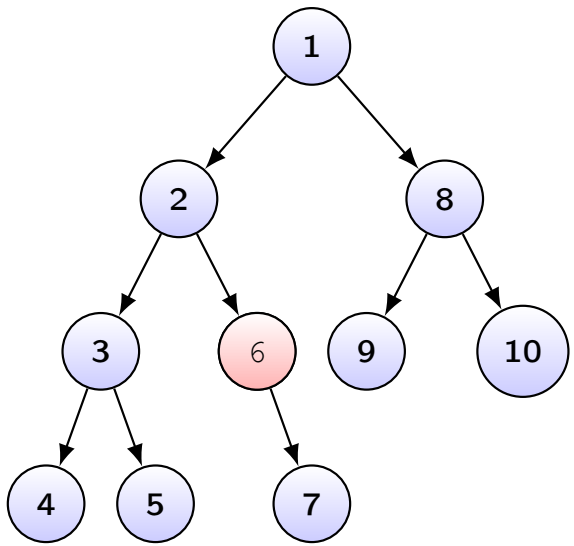
Parcours en profondeur

Types

Infixe (ou symétrique) : (dans un arbre binaire seulement) on visite chaque nœud après son fils à gauche et avant son fils à droite.

1. Parcours infixe du sous-arbre gauche
2. On visite la racine de l'arbre
3. Parcourt infixe du sous-arbre droit

On l'appelle parcours G R D (Gauche Racine Droit).



Parcours : `4,3,5,2,6,7,1,9,8,10`

(On considère toujours un parcours gauche à droite)

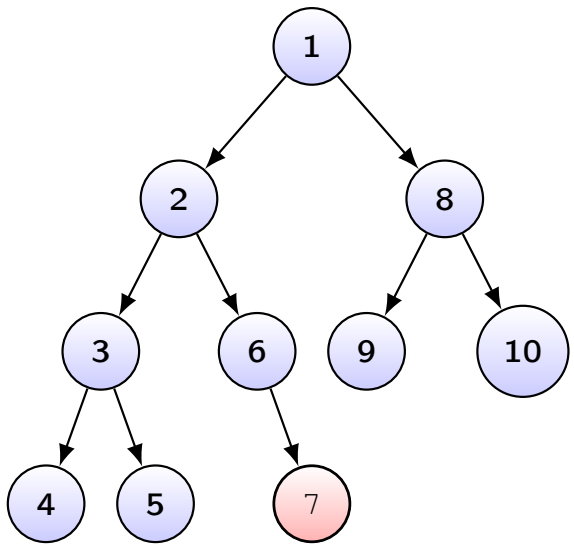
Parcours en profondeur

Types

Infixe (ou symétrique) : (dans un arbre binaire seulement) on visite chaque nœud après son fils à gauche et avant son fils à droite.

1. Parcours infixe du sous-arbre gauche
2. On visite la racine de l'arbre
3. Parcourt infixe du sous-arbre droit

On l'appelle parcours G R D (Gauche Racine Droit).



Parcours : `4,3,5,2,6,7,1,9,8,10`

(On considère toujours un parcours gauche à droite)

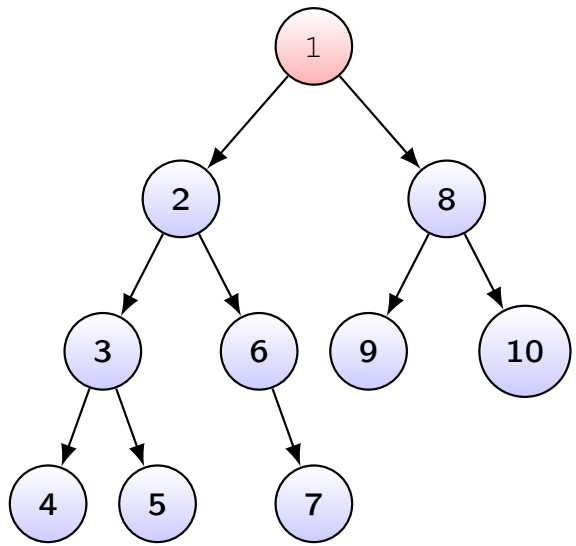
Parcours en profondeur

Types

Infixe (ou symétrique) : (dans un arbre binaire seulement) on visite chaque nœud après son fils à gauche et avant son fils à droite.

1. Parcours infixe du sous-arbre gauche
2. On visite la racine de l'arbre
3. Parcourt infixe du sous-arbre droit

On l'appelle parcours G R D (Gauche Racine Droit).



Parcours : `4,3,5,2,6,7,1,9,8,10`

(On considère toujours un parcours gauche à droite)

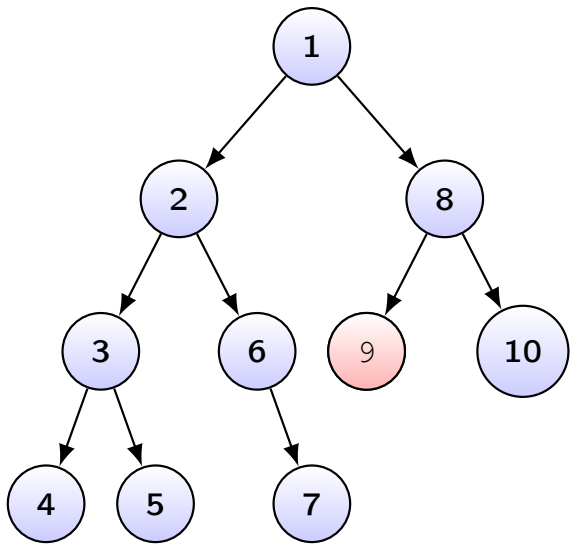
Parcours en profondeur

Types

Infixe (ou symétrique) : (dans un arbre binaire seulement) on visite chaque nœud après son fils à gauche et avant son fils à droite.

1. Parcours infixe du sous-arbre gauche
2. On visite la racine de l'arbre
3. Parcourt infixe du sous-arbre droit

On l'appelle parcours G R D (Gauche Racine Droit).



Parcours : `4,3,5,2,6,7,1,9,8,10`

(On considère toujours un parcours gauche à droite)

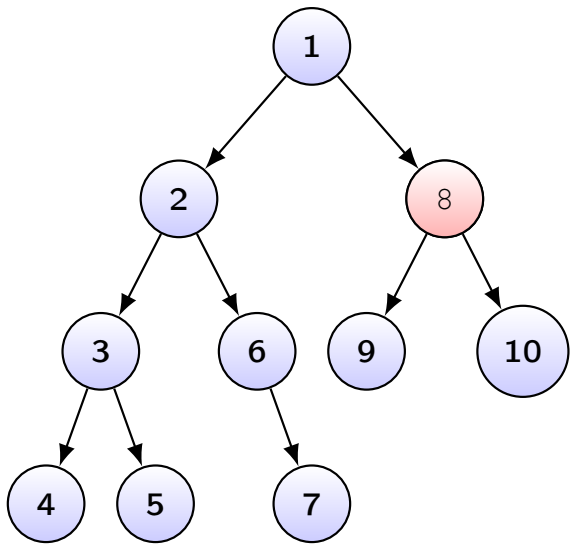
Parcours en profondeur

Types

Infixe (ou symétrique) : (dans un arbre binaire seulement) on visite chaque nœud après son fils à gauche et avant son fils à droite.

1. Parcours infixe du sous-arbre gauche
2. On visite la racine de l'arbre
3. Parcourt infixe du sous-arbre droit

On l'appelle parcours G R D (Gauche Racine Droit).



Parcours : `4,3,5,2,6,7,1,9,8,10`

(On considère toujours un parcours gauche à droite)

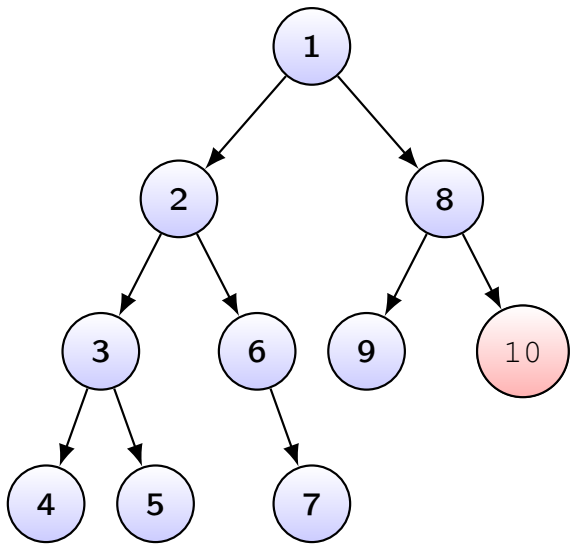
Parcours en profondeur

Types

Infixe (ou symétrique) : (dans un arbre binaire seulement) on visite chaque nœud après son fils à gauche et avant son fils à droite.

1. Parcours infixe du sous-arbre gauche
2. On visite la racine de l'arbre
3. Parcourt infixe du sous-arbre droit

On l'appelle parcours G R D (Gauche Racine Droit).



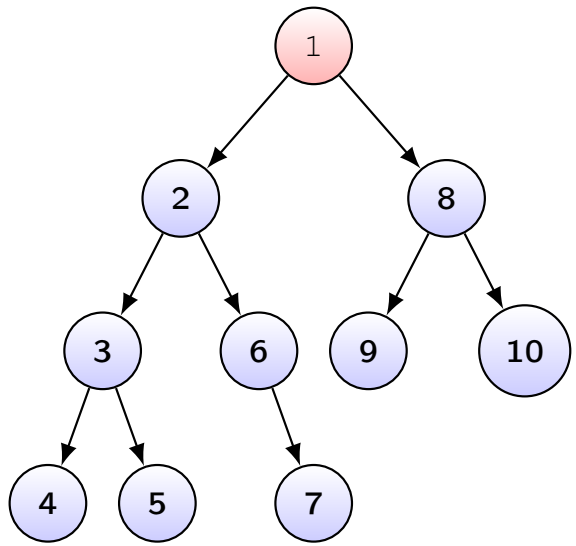
Parcours : `4,3,5,2,6,7,1,9,8,10`

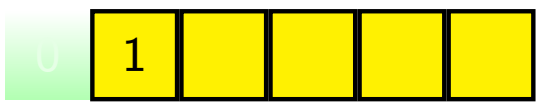
(On considère toujours un parcours gauche à droite)

Parcours en largueur

Algorithme : (Source : Wikipedia.)

```
ParcoursLargeur(a: Arbre d'Elements)
{
    noeud : Arbre d'Elements
    f : File d'Arbre d'Elements
    creerFile(f)
    enfiler(f, a)
    Tant que (longueur(f) != 0)
    {
      noeud = defiler(f)
      Visiter(noeud)
      Pour i ← 1 à degre(noeud)
        enfiler(f, recEnfant(noeud,i))
      finPour
    }
}
```

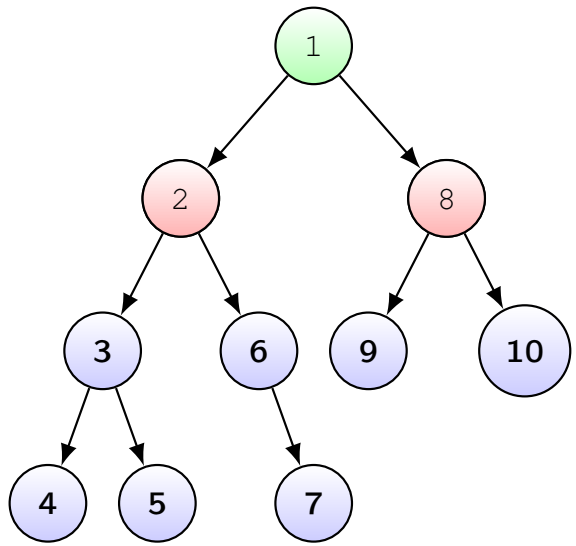


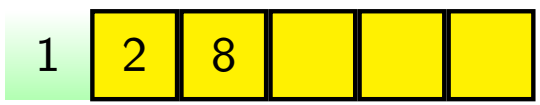


Parcours en largueur

Algorithme : (Source : Wikipedia.)

```
ParcoursLargeur(a: Arbre d'Elements)
{
    noeud : Arbre d'Elements
    f : File d'Arbre d'Elements
    creerFile(f)
    enfiler(f, a)
    Tant que (longueur(f) != 0)
    {
      noeud = defiler(f)
      Visiter(noeud)
      Pour i ← 1 à degre(noeud)
        enfiler(f, recEnfant(noeud,i))
      finPour
    }
}
```

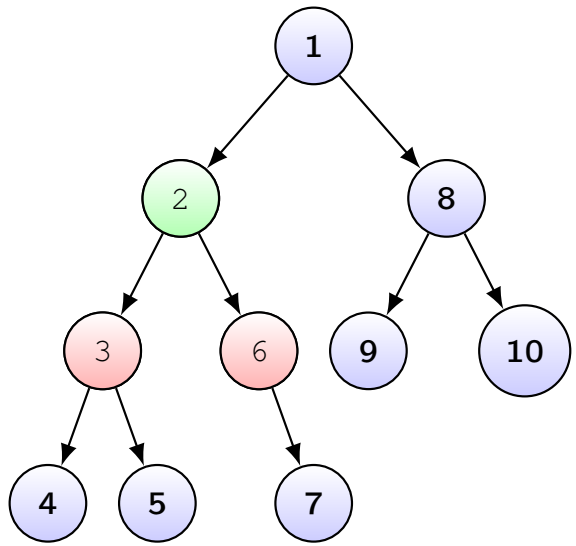


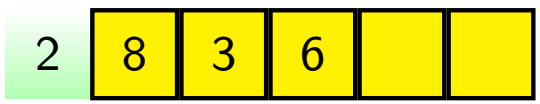


Parcours en largueur

Algorithme : (Source : Wikipedia.)

```
ParcoursLargeur(a: Arbre d'Elements)
{
    noeud : Arbre d'Elements
    f : File d'Arbre d'Elements
    creerFile(f)
    enfiler(f, a)
    Tant que (longueur(f) != 0)
    {
      noeud = defiler(f)
      Visiter(noeud)
      Pour i ← 1 à degre(noeud)
        enfiler(f, recEnfant(noeud,i))
      finPour
    }
}
```

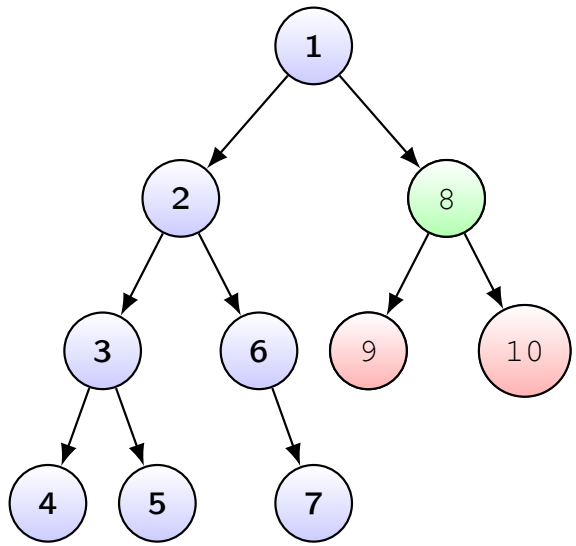


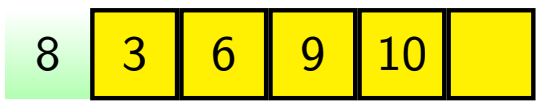


Parcours en largueur

Algorithme : (Source : Wikipedia.)

```
ParcoursLargeur(a: Arbre d'Elements)
{
    noeud : Arbre d'Elements
    f : File d'Arbre d'Elements
    creerFile(f)
    enfiler(f, a)
    Tant que (longueur(f) != 0)
    {
      noeud = defiler(f)
      Visiter(noeud)
      Pour i ← 1 à degre(noeud)
        enfiler(f, recEnfant(noeud,i))
      finPour
    }
}
```

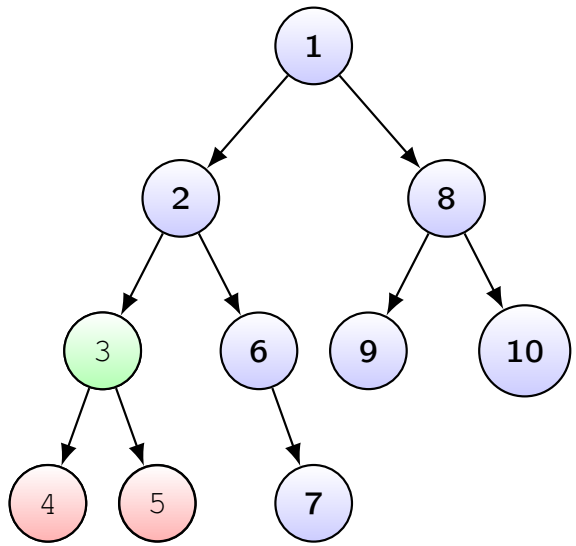


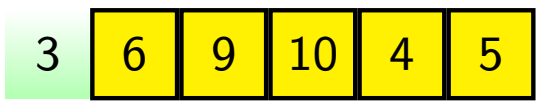


Parcours en largueur

Algorithme : (Source : Wikipedia.)

```
ParcoursLargeur(a: Arbre d'Elements)
{
    noeud : Arbre d'Elements
    f : File d'Arbre d'Elements
    creerFile(f)
    enfiler(f, a)
    Tant que (longueur(f) != 0)
    {
      noeud = defiler(f)
      Visiter(noeud)
      Pour i ← 1 à degre(noeud)
        enfiler(f, recEnfant(noeud,i))
      finPour
    }
}
```

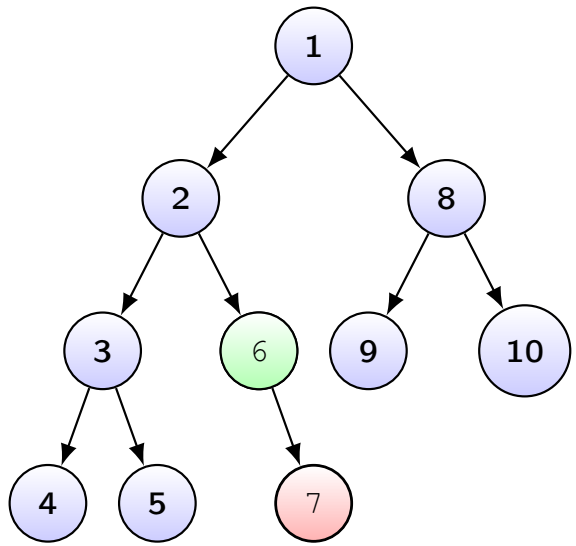




Parcours en largueur

Algorithme : (Source : Wikipedia.)

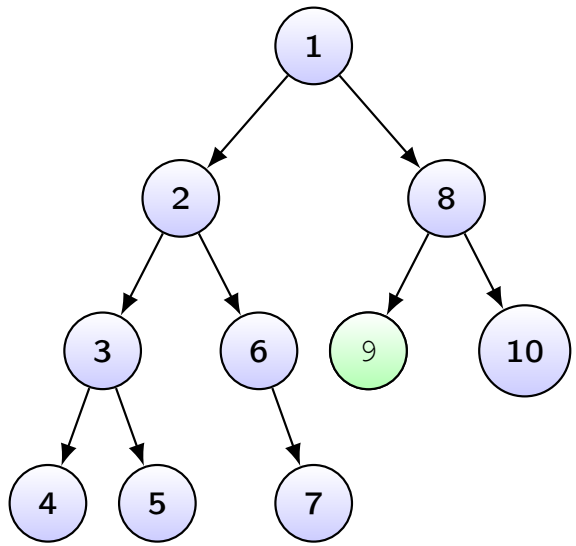
```
ParcoursLargeur(a: Arbre d'Elements)
{
    noeud : Arbre d'Elements
    f : File d'Arbre d'Elements
    creerFile(f)
    enfiler(f, a)
    Tant que (longueur(f) != 0)
    {
      noeud = defiler(f)
      Visiter(noeud)
      Pour i ← 1 à degre(noeud)
        enfiler(f, recEnfant(noeud,i))
      finPour
    }
}
```

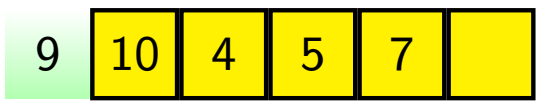


Parcours en largueur

Algorithme : (Source : Wikipedia.)

```
ParcoursLargeur(a: Arbre d'Elements)
{
    noeud : Arbre d'Elements
    f : File d'Arbre d'Elements
    creerFile(f)
    enfiler(f, a)
    Tant que (longueur(f) != 0)
    {
      noeud = defiler(f)
      Visiter(noeud)
      Pour i ← 1 à degre(noeud)
        enfiler(f, recEnfant(noeud,i))
      finPour
    }
}
```

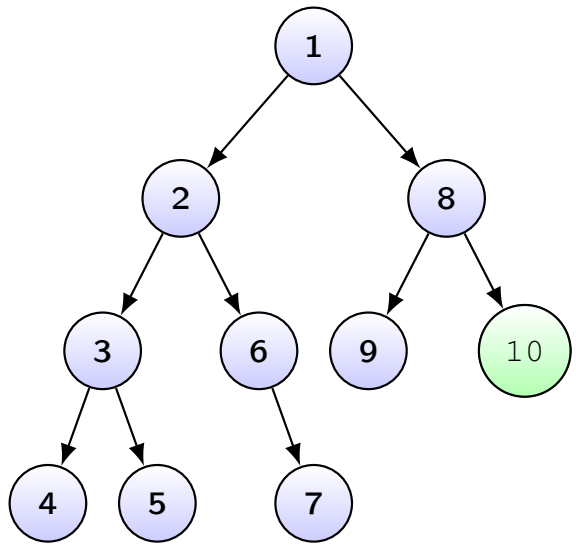


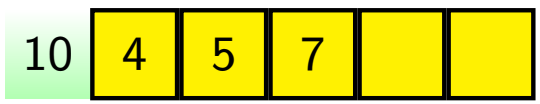


Parcours en largueur

Algorithme : (Source : Wikipedia.)

```
ParcoursLargeur(a: Arbre d'Elements)
{
    noeud : Arbre d'Elements
    f : File d'Arbre d'Elements
    creerFile(f)
    enfiler(f, a)
    Tant que (longueur(f) != 0)
    {
      noeud = defiler(f)
      Visiter(noeud)
      Pour i ← 1 à degre(noeud)
        enfiler(f, recEnfant(noeud,i))
      finPour
    }
}
```

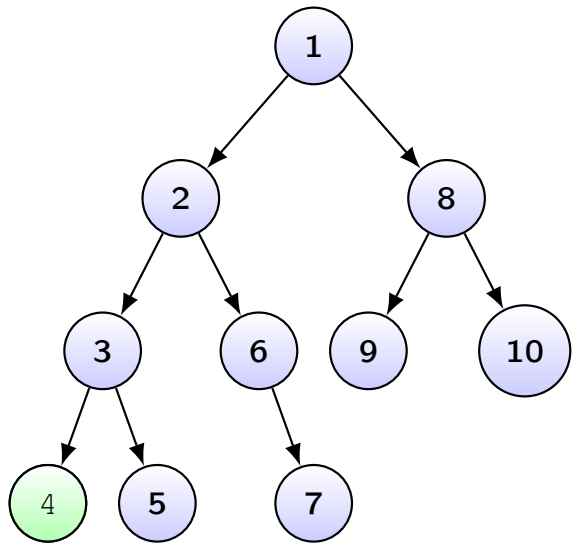


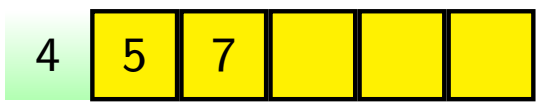


Parcours en largueur

Algorithme : (Source : Wikipedia.)

```
ParcoursLargeur(a: Arbre d'Elements)
{
    noeud : Arbre d'Elements
    f : File d'Arbre d'Elements
    creerFile(f)
    enfiler(f, a)
    Tant que (longueur(f) != 0)
    {
      noeud = defiler(f)
      Visiter(noeud)
      Pour i ← 1 à degre(noeud)
        enfiler(f, recEnfant(noeud,i))
      finPour
    }
}
```

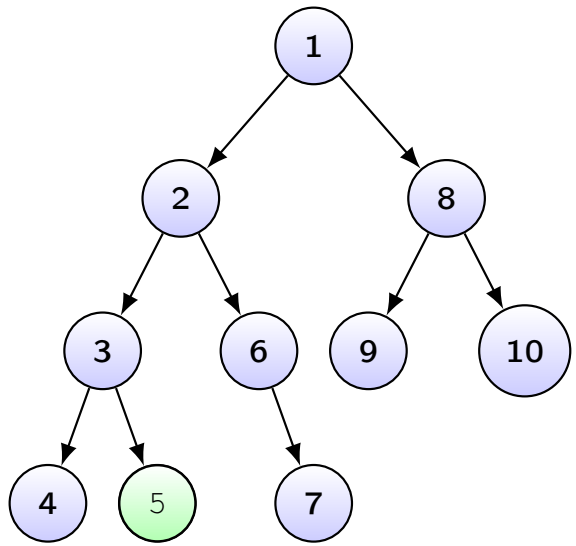


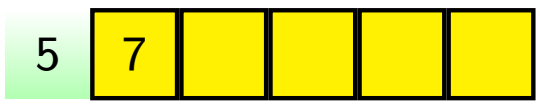


Parcours en largueur

Algorithme : (Source : Wikipedia.)

```
ParcoursLargeur(a: Arbre d'Elements)
{
    noeud : Arbre d'Elements
    f : File d'Arbre d'Elements
    creerFile(f)
    enfiler(f, a)
    Tant que (longueur(f) != 0)
    {
      noeud = defiler(f)
      Visiter(noeud)
      Pour i ← 1 à degre(noeud)
        enfiler(f, recEnfant(noeud,i))
      finPour
    }
}
```

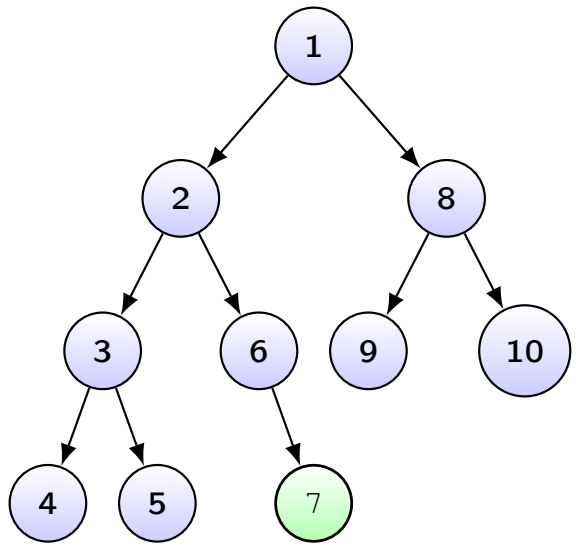


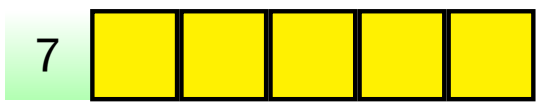


Parcours en largueur

Algorithme : (Source : Wikipedia.)

```
ParcoursLargeur(a: Arbre d'Elements)
{
    noeud : Arbre d'Elements
    f : File d'Arbre d'Elements
    creerFile(f)
    enfiler(f, a)
    Tant que (longueur(f) != 0)
    {
      noeud = defiler(f)
      Visiter(noeud)
      Pour i ← 1 à degre(noeud)
        enfiler(f, recEnfant(noeud,i))
      finPour
    }
}
```





Contenu

Opérations sur un arbre

Méthodes de construction et d'initialisation :

```
fonction creerArbre(e: Element) : Arbre d'Elements
```

Méthodes de mise à jour :

```
procedure modifierRacine (a : Arbre d'Elements (E/S),
                          e : Element)
procedure ajouterFeuille (a : Arbre d'Elements (E/S),
                          e : Element)
procedure supprimerFeuille (a : Arbre d'Elements (E/S))
```

Méthodes d'accès :

```
fonction estFeuille (a : Arbre d'Elements) : Booleen
fonction racine (a : Arbre d'Elements) : Element
fonction recEnfant (a : Arbre d'Elements, pos : Entier) :
                                        Arbre d'Elements
fonction degre (a : Arbre d'Elements) : Entier
```

Pré-conditions

Méthodes de construction et d'initialisation :

fonction **creerArbre**(e) : Aucun

Méthodes de mise à jour :

procedure **modifierRacine** (a, e) : a initialisé
procedure **ajouterFeuille** (a, e) : a initialisé
procedure **supprimerFeuille** (a) : a initialisé

Méthodes d'accès :

fonction **estFeuille** (a) : a initialisé
fonction **racine** (a) : a initialisé
fonction **recEnfant** (a,pos) : a initialisé, $1 \leq pos \leq degre(a)$
fonction **degre** (a) : a initialisé

Post-conditions

Méthodes de construction et d'initialisation :

fonction **creerArbre** (e) : arbre initialisé

Méthodes de mise à jour :

procedure **modifierRacine** (a, e) : racine de **a** egal à **e**
procedure **ajouterFeuille** (a, e) : **a** un fils de valeur **e** en plus
procedure **supprimerFeuille** (a) : **a** n'existe plus

Méthodes d'accès :

fonction **estFeuille** (a) : aucun
fonction **racine** (a) : aucun
fonction **recEnfant** (a,pos) : aucun
fonction **degre** (a) : aucun

Contenu

Sauvegarde d'un arbre comme une liste

```
typeNoeud Structure
    elt : Element
    noFils : Entier
Fin Structure

Pré-condition : ar est un arbre non vide

Procedure creerListeDeArbre(res : Liste d'arbre d'Elements (S),
                                  ar : Arbre d'Elements (E))
    n : TypeNoeud
    creerListe(res)
    Pour i ← 1 à degre(ar)
        creerListeDeArbreRec(res, recEnfant(ar,i), i)
    Fin Pour
    n.elt = racine(ar)
    n.noFils = 0
    ajouter(res, n)
Fin Procedure
```

Sauvegarde d'un arbre comme une liste (cont.)

```
Procedure creerListeDeArbreRec(res:Liste d'arbre d'Elements (E/S),
                               ar : Arbre d'Elements (E),
                               no : Entier)
    n : TypeNoeud
    i : Entier
    Pour i ← 1 à degre(ar)
        creerListeDeArbreRec(res, recEnfant(ar,i), i)
    Fin Pour
    n.elt = racine(ar)
    n.noFils = no
    ajouter(res, n)
Fin Procedure
```