Institut Universitaire
de Technologie
Aix*Marseille Université

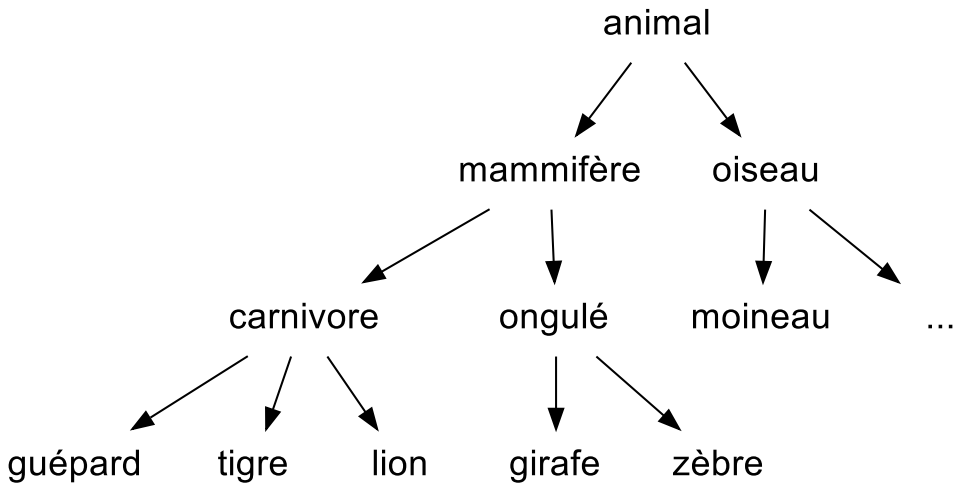
M3103
Algorithmique avancée

Chapitre 02 - Les arbres

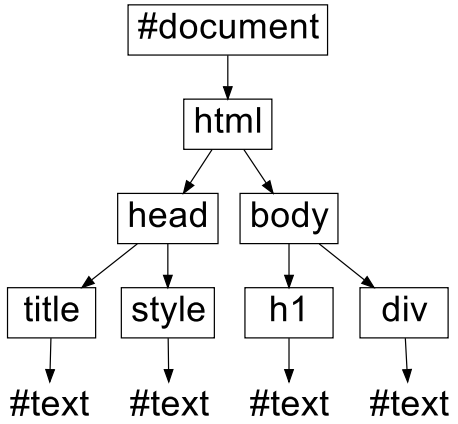
Les arbres

Les arbres font partis des structures les plus utilisées en informatique.

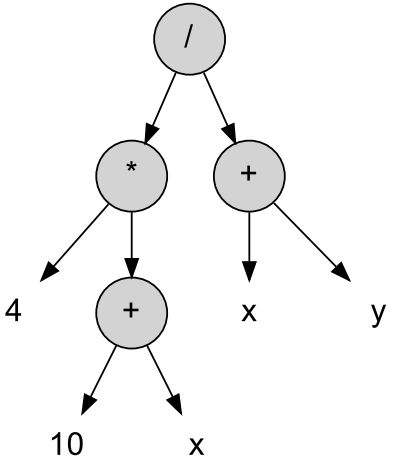
- Classification :
  ontologie



- Représentation de structures :
  DOM (Document Object Model)



- Représentation d'expression :
  (4(10+x))/(x+y)*



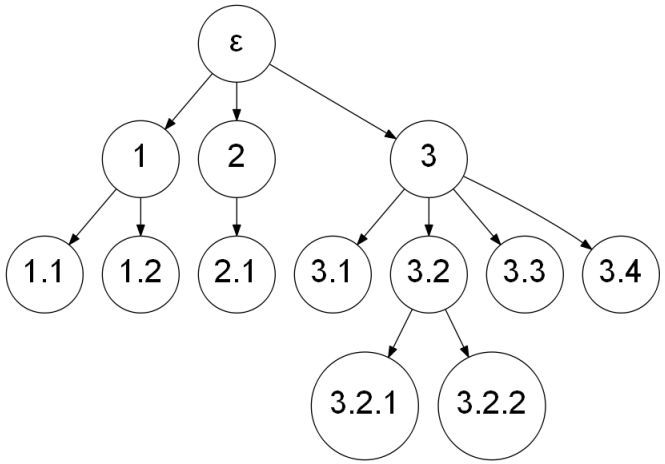
Définition

Les Arbres

Définition

Arbre

Graphe connexe sans cycle.



Un arbre est constitué par un ensemble de nœuds reliés par des arcs.

Arbre enraciné

On parle d'arbre enraciné lorsque l'arbre possède un nœud racine et que l'on a les propriétés suivantes :

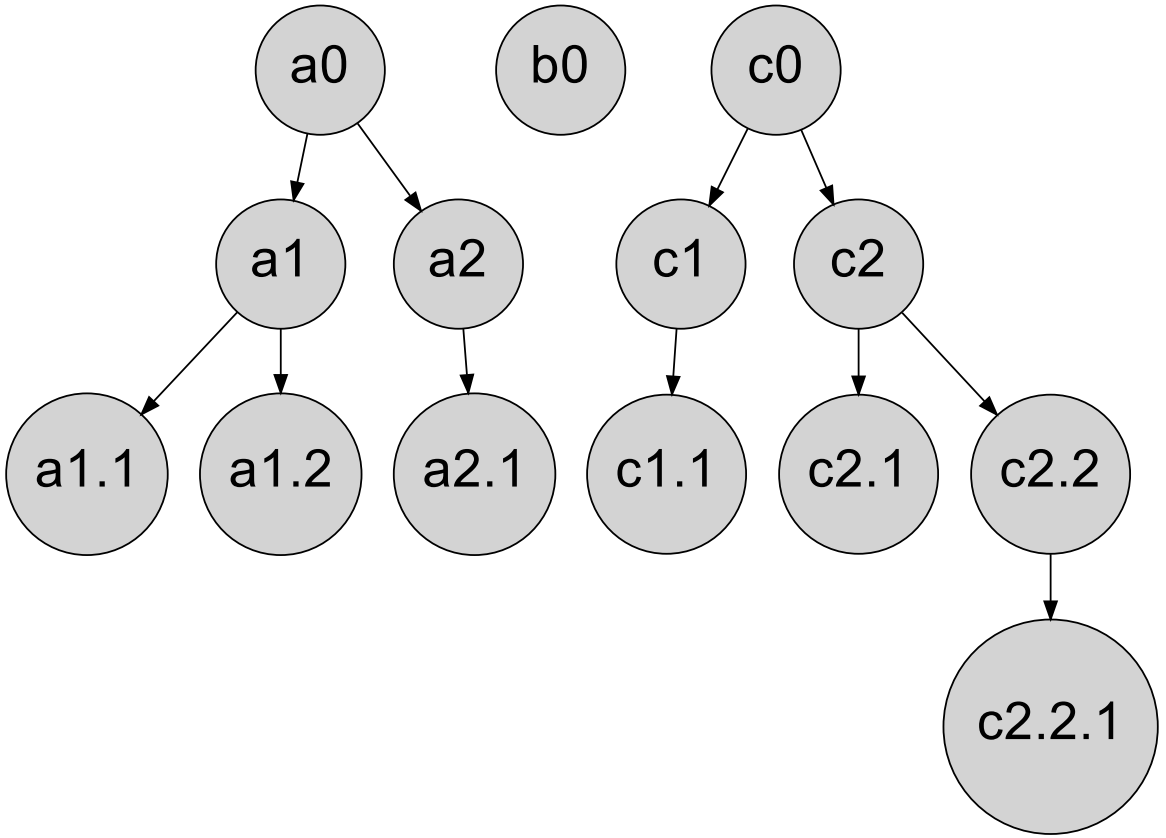
- La racine n'a pas de père.
- Tout nœud autre que la racine ne possède qu'un et un seul père.
- Un arbre est connexe.

Un nœud peut avoir 0 ou plusieurs fils
Un nœud a exactement 1 père

Définition

Forêt

Graphe dont chaque composante connexe est un arbre.



Vocabulaire

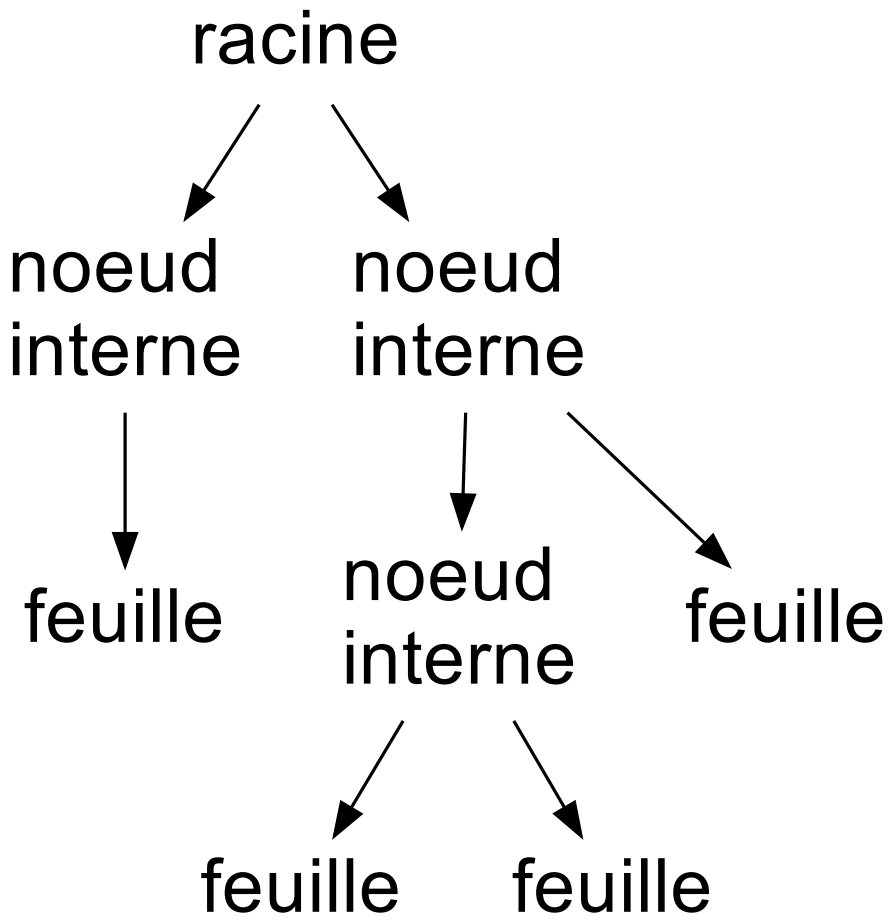
Nœud
Sommet d'un arbre.

Nœud interne
Sommet qui n'est ni une feuille, ni une racine.

Racine
Nœud sans prédécesseur.

Feuille
Nœud qui n'a pas de successeur (pas de fils).

Degrés d'un nœud
Nombre de ses successeurs.



Relations entre les nœuds

Ancêtres d'un nœud n
Ensemble des nœuds trouvés sur le chemin unique entre n et la racine.

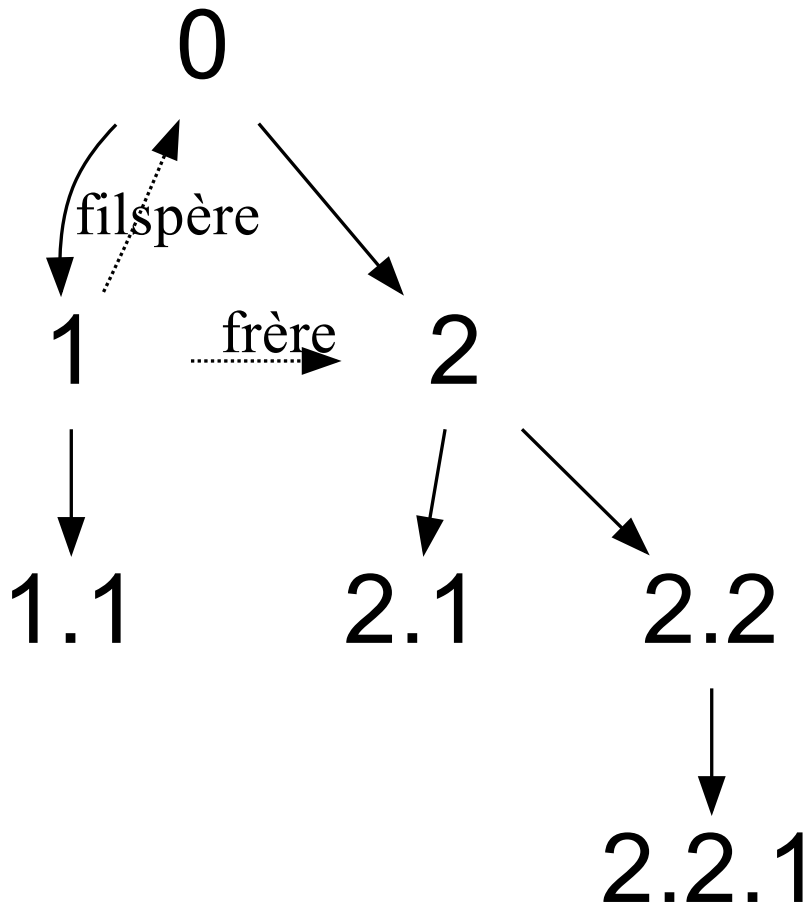
Descendance d'un nœud
On dit que le nœud b est un descendant du nœud a si et seulement si a est un ancêtre de b.

Racine
Ancêtre de tous les nœuds.

Père
Ascendant immédiat.

Frère
Nœud ayant tous le même père.

Fils
Descendant immédiat.
On parle de 1er fils, 2ème fils, 3ème fils, etc…



Mesure sur les arbres

Taille d'un arbre
Nombre de nœud de l'arbre.

Hauteur d'un nœud n (profondeur)
Longueur du chemin qui va de la racine à n.
Par définition, la hauteur de la racine est de 0.
Les fils d'un nœud de profondeur i sont de profondeur $i + 1$.

Hauteur d'un arbre
Plus grande profondeur des feuilles de l'arbre.
Par définition, la hauteur de l'arbre vide est de -1.

Longueur de cheminement d'un arbre
Somme des profondeurs de chacun des nœuds.

Longueur de cheminement externe
Somme des profondeurs de chacune des feuilles.

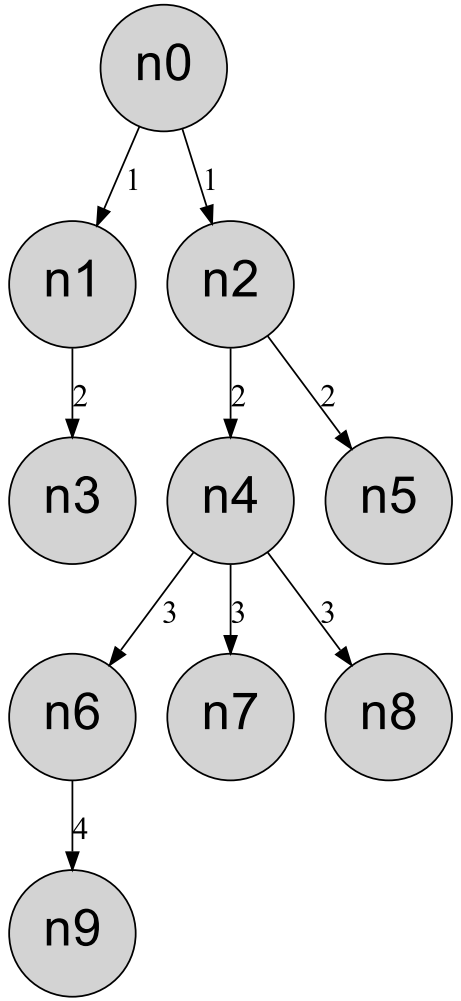
Longueur de cheminement interne
Somme des profondeurs des nœuds internes.

Mesures sur les arbres

Exemples :

Hauteur de l'arbre : 4
Taille de l'arbre : 10
Nombre de feuilles : 5
Hauteur du nœud n6 : 3

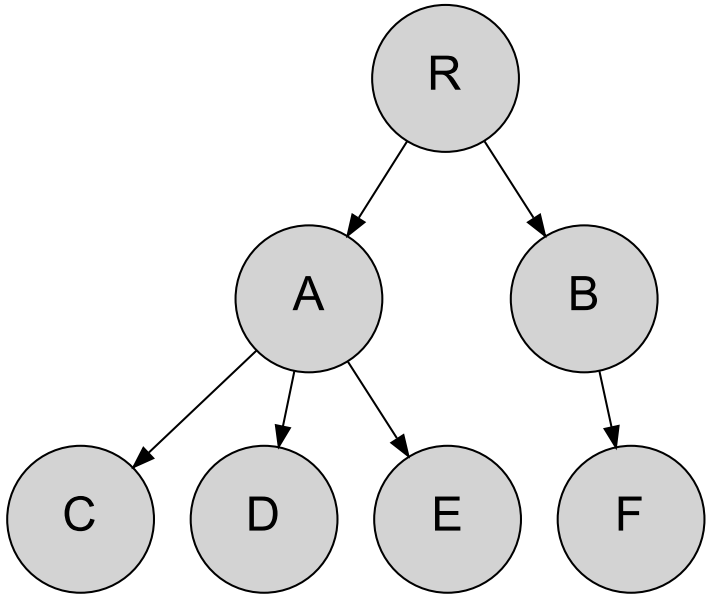
Longueur de cheminement : 21
Longueur de cheminement externe : 14
Longueur de cheminement interne : 7

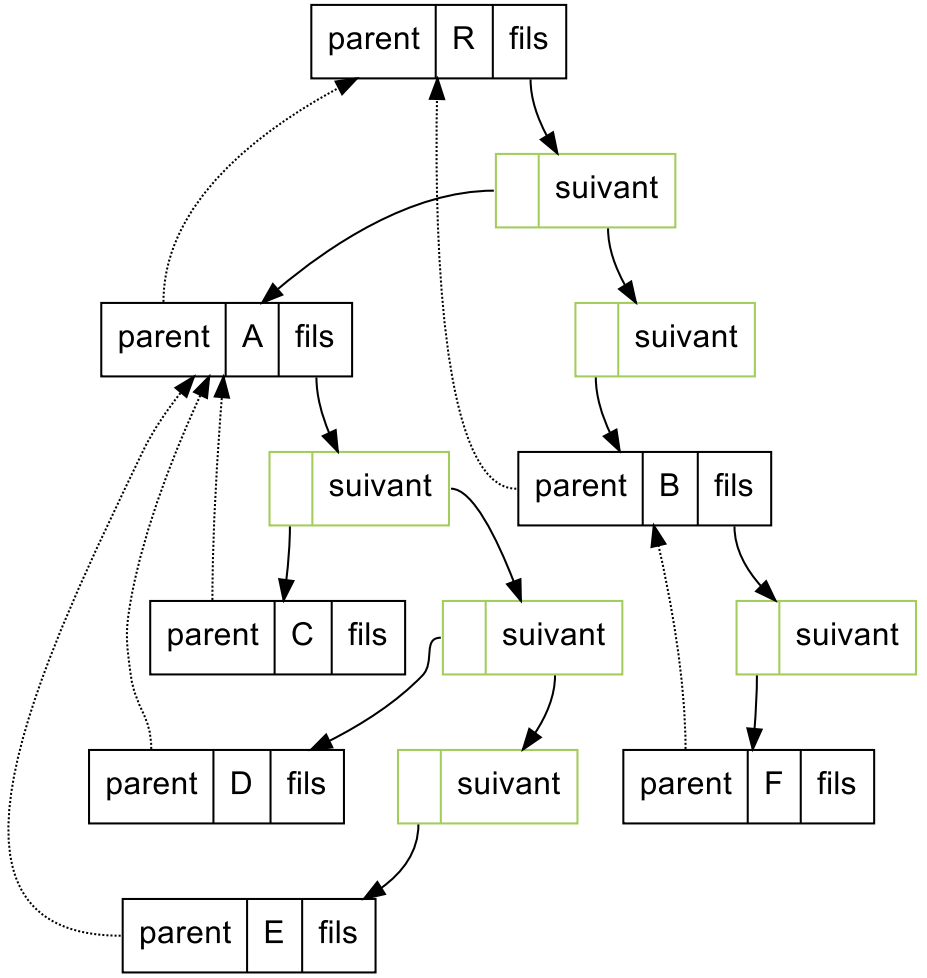


Implantations

Implantation

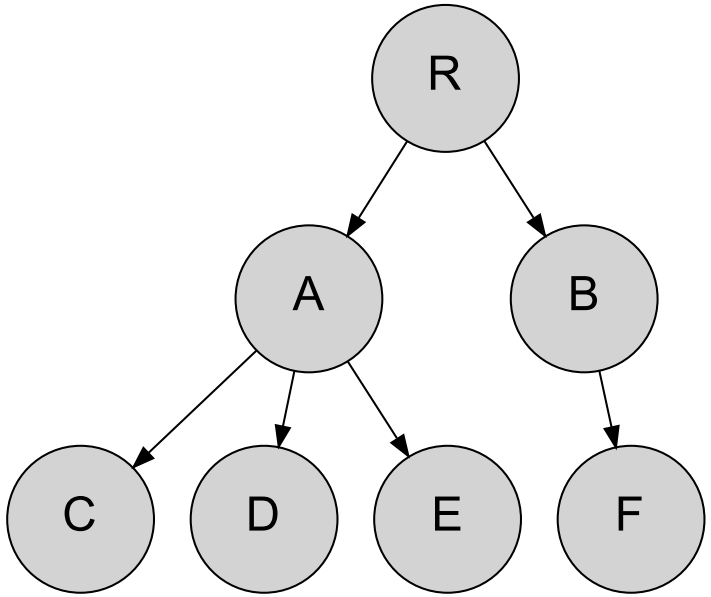
- Implantation par liste de fils

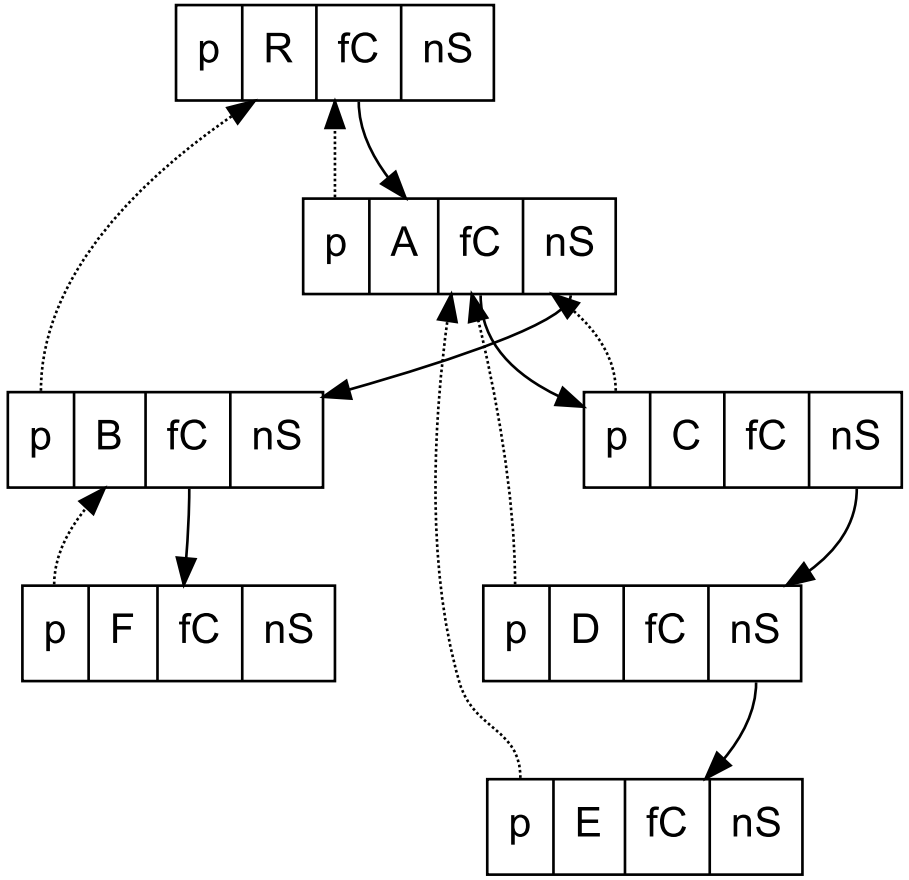




Implantation

- Implantation par liste d'adjacences
  first-child-next-sibling
  Left-Child/Right-Sibling



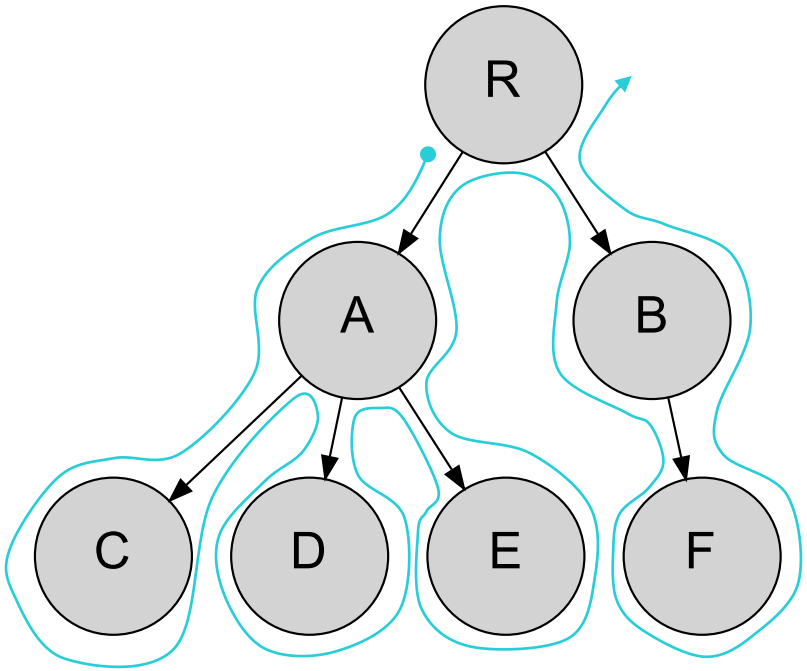


Parcours

Parcours

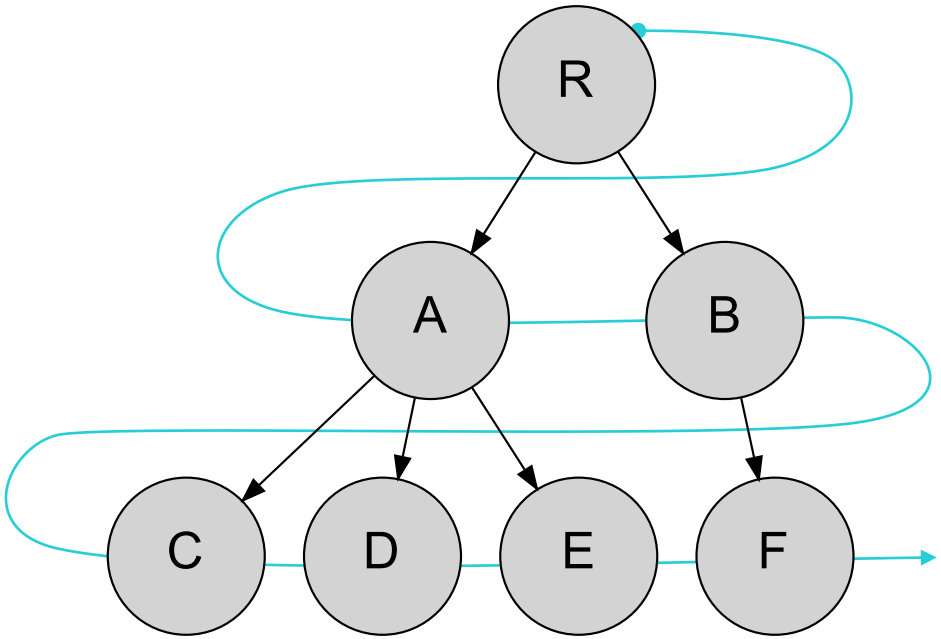
Parcours en profondeur

parcours branche par branche.



Parcours en largeur

parcours niveau par niveau.



Parcours en profondeur

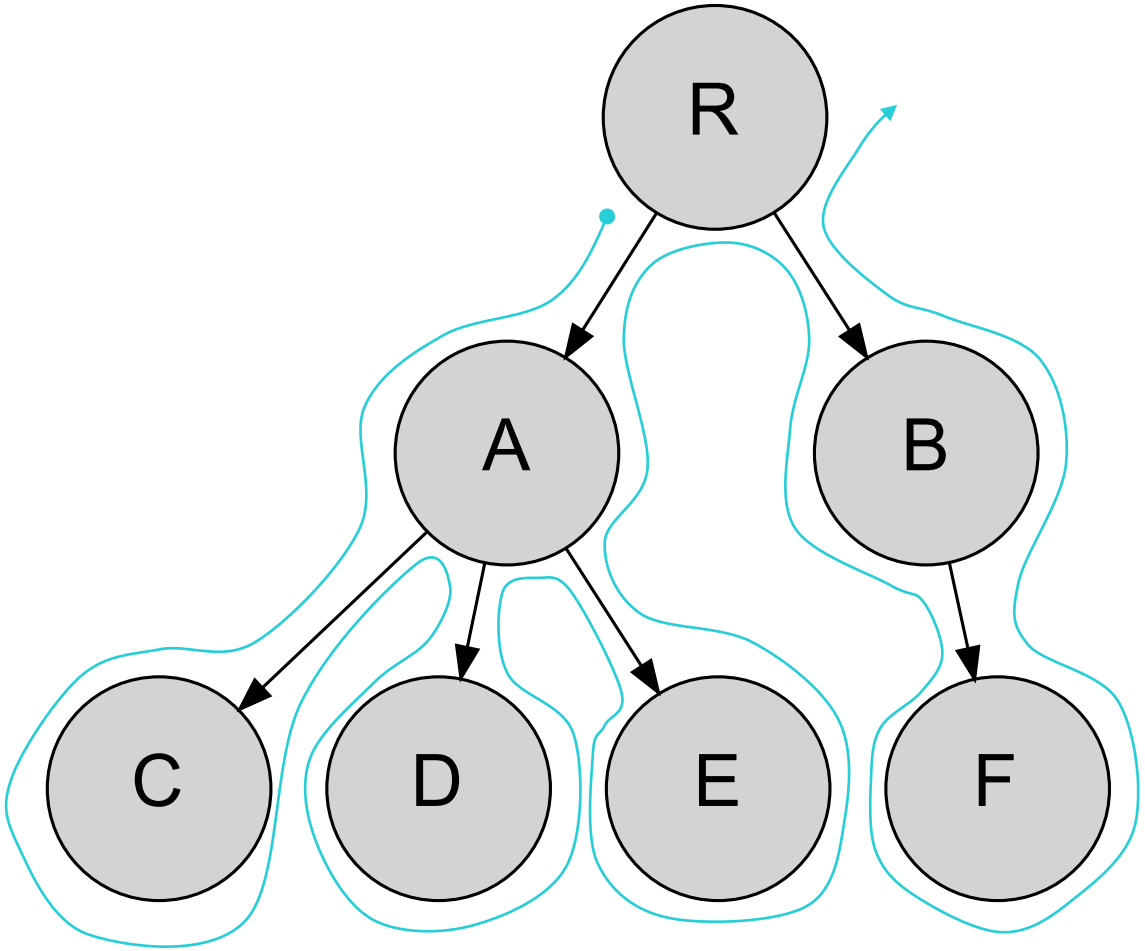
Parcours préfixe

Soit un arbre non vide $A = (r, a_1, a_2, \ldots, a_k)$,

$P_{préfixe}(A) = (r).P_{préfixe}(a_1).\ldots.P_{préfixe}(a_k)$

Résultat

(R,A,C,D,E,B,F)



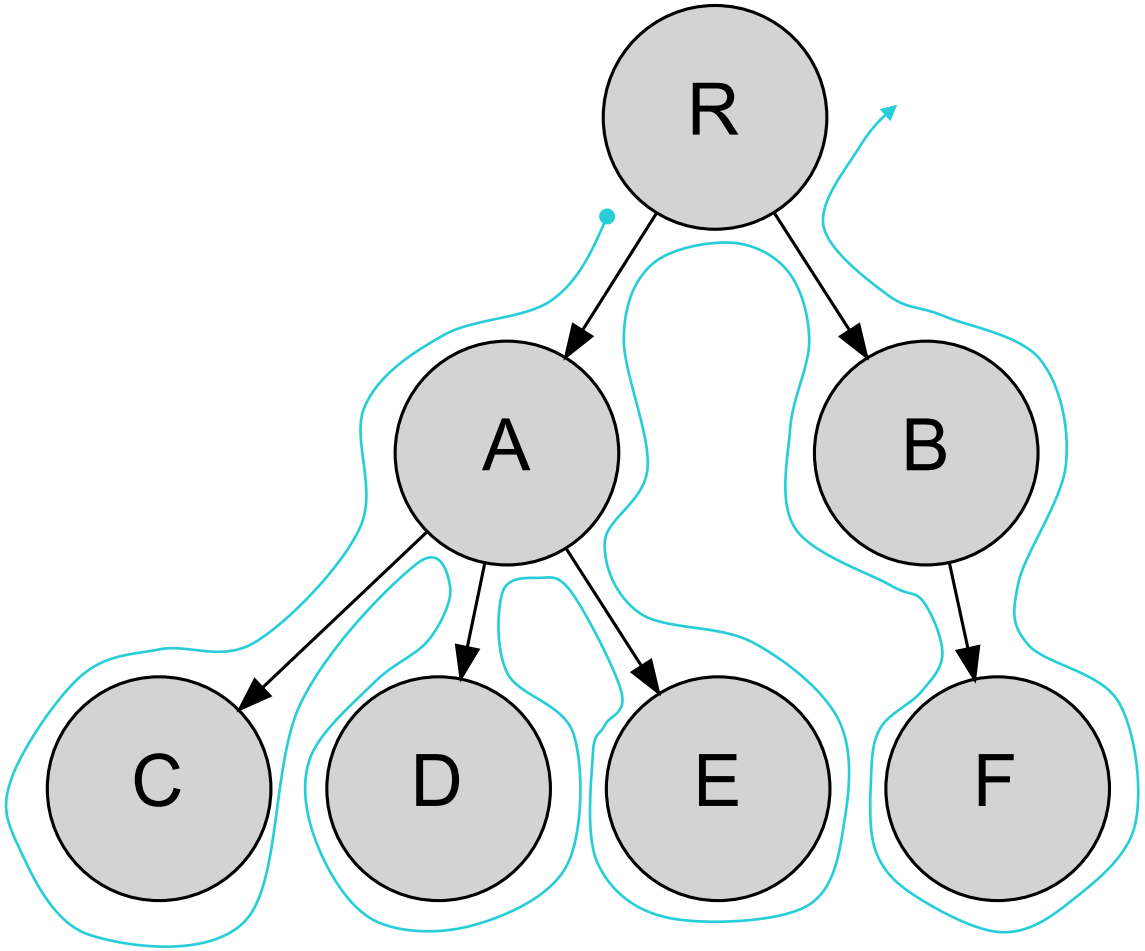
Parcours en profondeur

Parcours suffixe

Arbre non vide $A = (r, a_1, a_2, \ldots, a_k)$

$P_{suffixe}(A) = P_{suffixe}(a_1). \ldots . P_{suffixe}(a_k). (r)$

Résultat
(C,D,E,A,F,B,R)

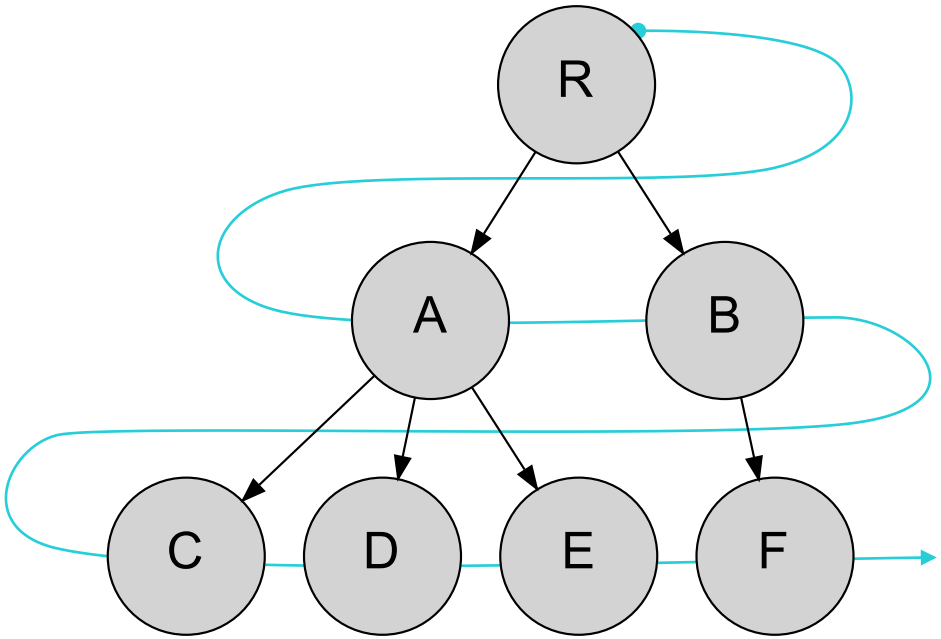


Parcours en largeur

Parcours hiérarchique

On commence par la racine et on visite les nœuds niveau par niveau, de gauche à droite On se déplace de frère en frère pour chaque niveau.

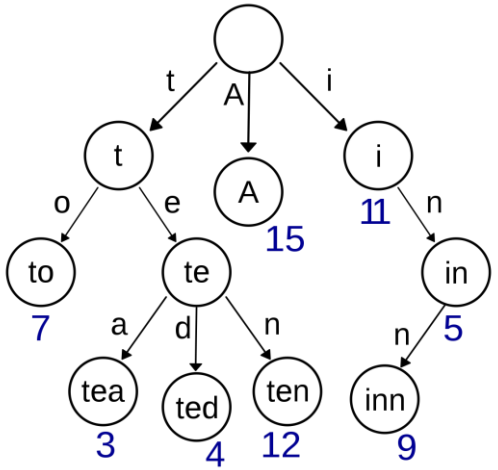
Résultat
(R,A,B,C,D,E,F)



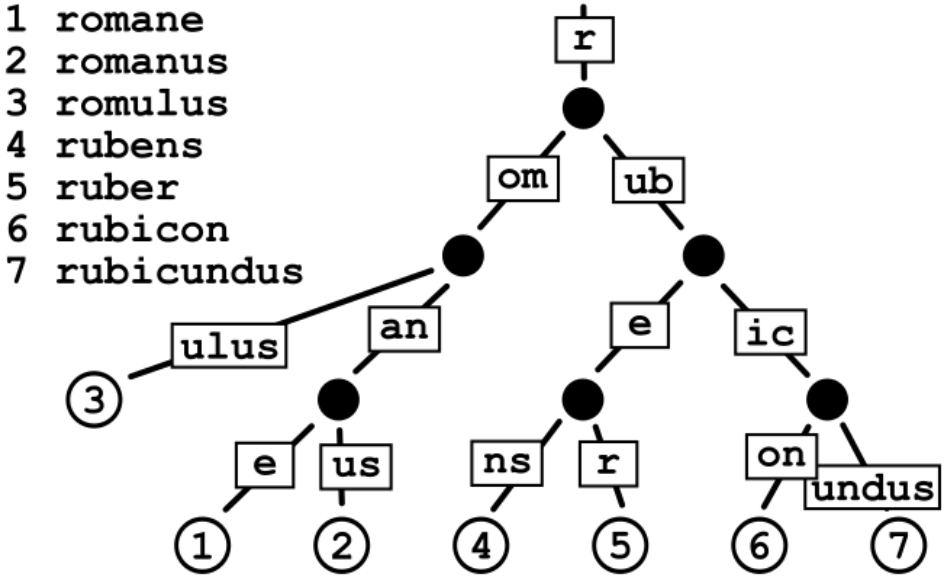
Utilisation

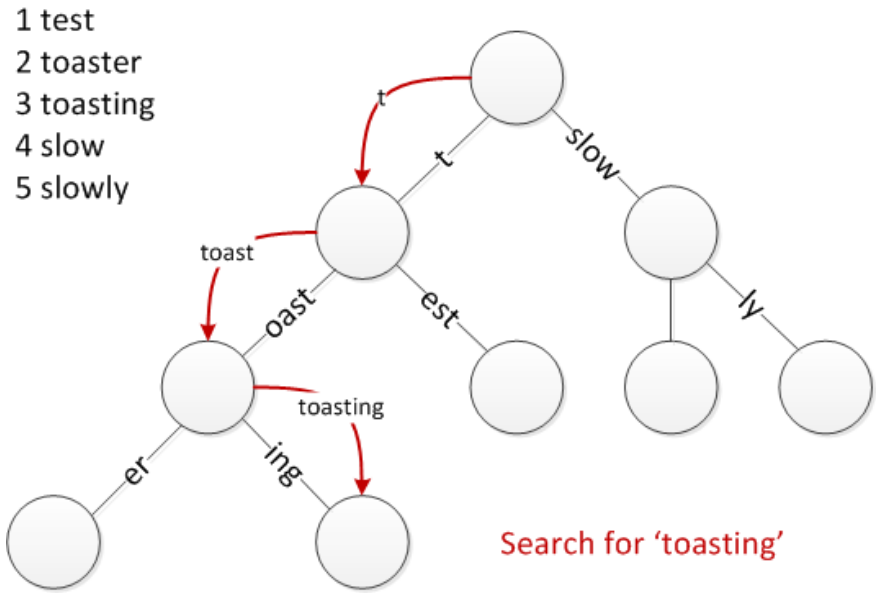
Exemples d'utilisation

- Arbre trie (arbre préfixe) :
  utilisé pour représenter un tableau associatif,
  algorithme de complétion,
  correction automatique,
  compression par dictionnaire (LZ77)/



- Arbre RADIX / PATRICIA / cir-tri : plus compact que arbre trie





illustrations wikipedia

Les arbres binaires

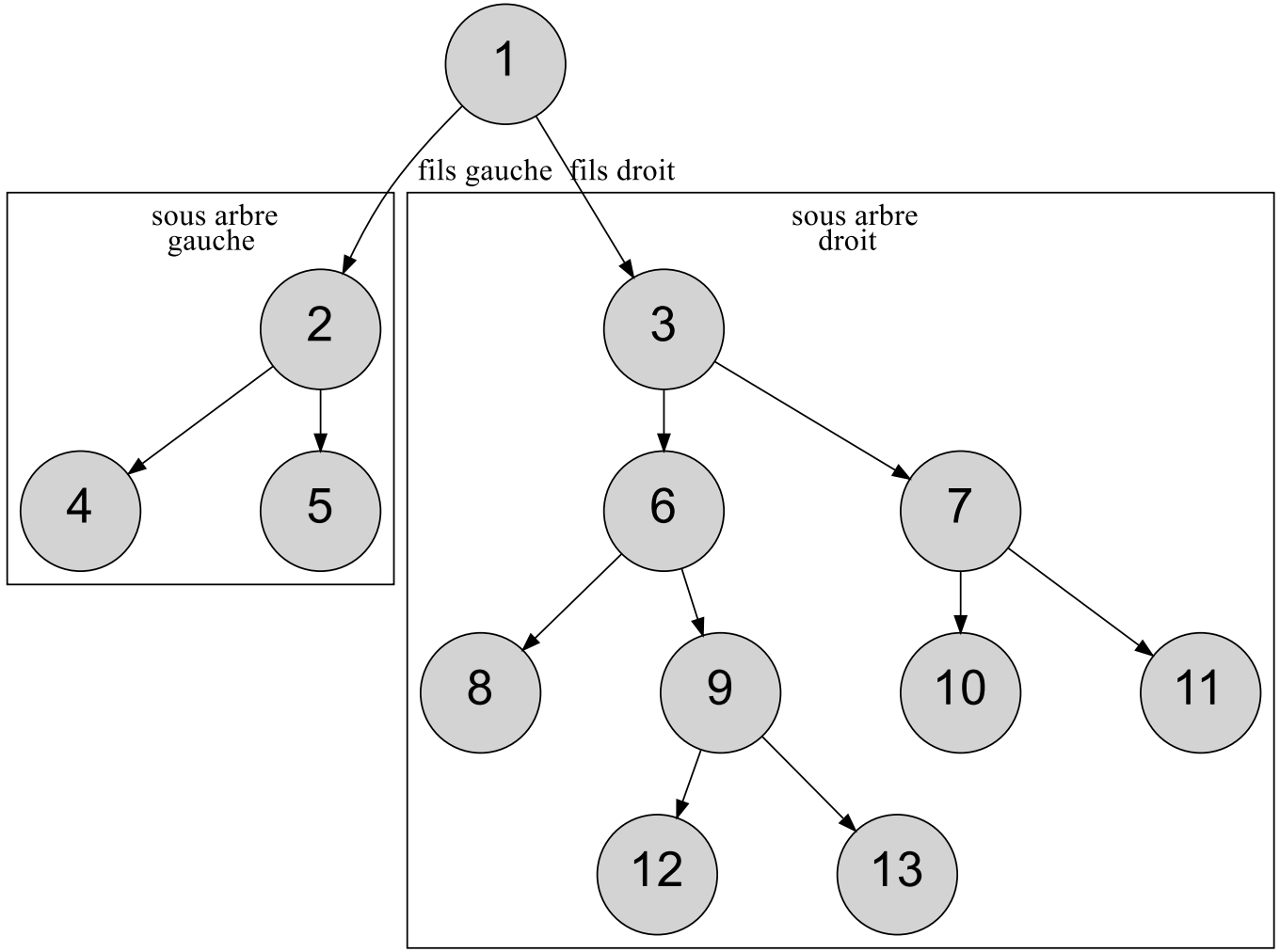
Les arbres

Définition

Un arbre binaire est un arbre :

- Soit vide ;
- Soit composé d'une racine et de deux sous arbres, le sous arbre gauche et le sous arbre droit.

La racine du sous arbre gauche est appelé **fils gauche** et la racine du sous arbre droit est appelé **fils droit**.



Arbres binaires particuliers

Arbre dégénéré (ou filiforme)
Tout nœud a au plus un fils.

Arbre complet
Tous les niveaux sont remplis.
Un arbre complet de profondeur h a $1 + 2 + 2^2 + \cdots + 2^h = 2^{h+1} - 1$ nœuds.

Arbre binaire parfait
Tous les niveaux sont remplis sauf éventuellement le dernier mais alors tous les nœuds sont à gauches.

Arbre binaire localement complet
Chaque nœud à 0 ou 2 fils.

Arbre binaire équilibré
La profondeur entre chaque branche reste équilibrée à l'aide d'un critère d'équilibre.

Propriété

Soit A un arbre binaire : $hauteur(A) \leq taille(A) - 1$

Dans le cas d'un arbre dégénéré, on a $hauteur(A) = taille(A) - 1$

Soit un arbre binaire de taille n et de hauteur h, alors : $\lfloor \log_2 n \rfloor \leq h \leq n - 1$

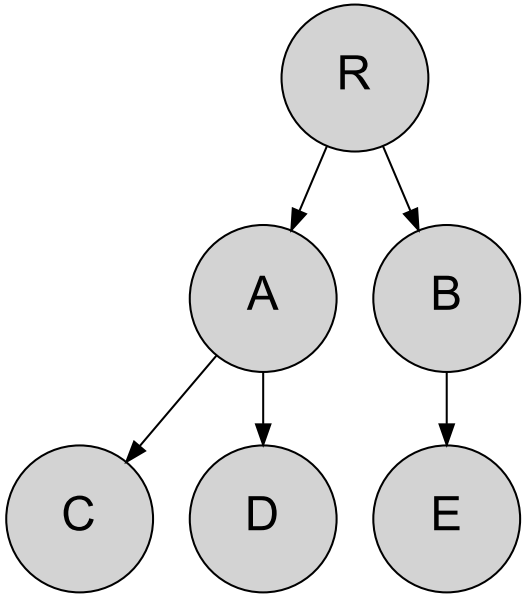
Instinctivement, l'arbre ayant le plus petit nombre de feuille est un arbre filiforme pour lequel on sait que $n = h + 1$

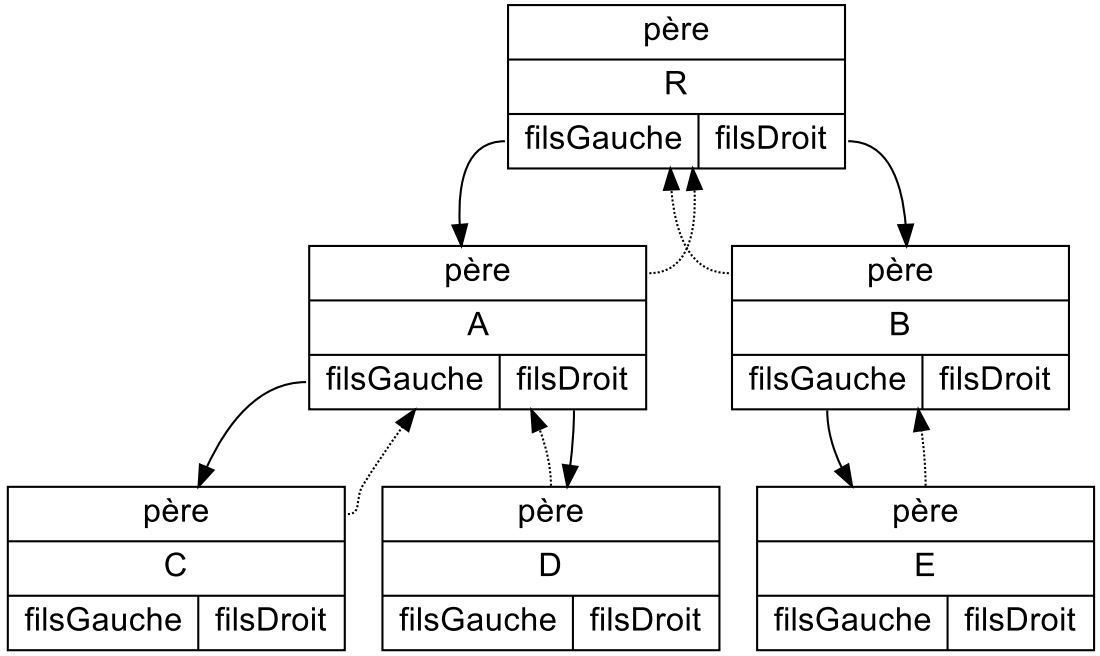
Pour la borne supérieur, le plus grand nombre de nœuds est donnée par un arbre complet pour lequel on sait que $n = 2^{h+1} - 1$

On a alors la propriété suivante :
Tout arbre binaire non vide B ayant f feuilles vérifie : $hauteur(B) \geq \lceil \log_2 f \rceil$

Implantation

- **Représentation chaînée**
  Chaque nœud contient un lien vers la racine du sous arbre gauche et un lien vers la racine du sous arbre droit



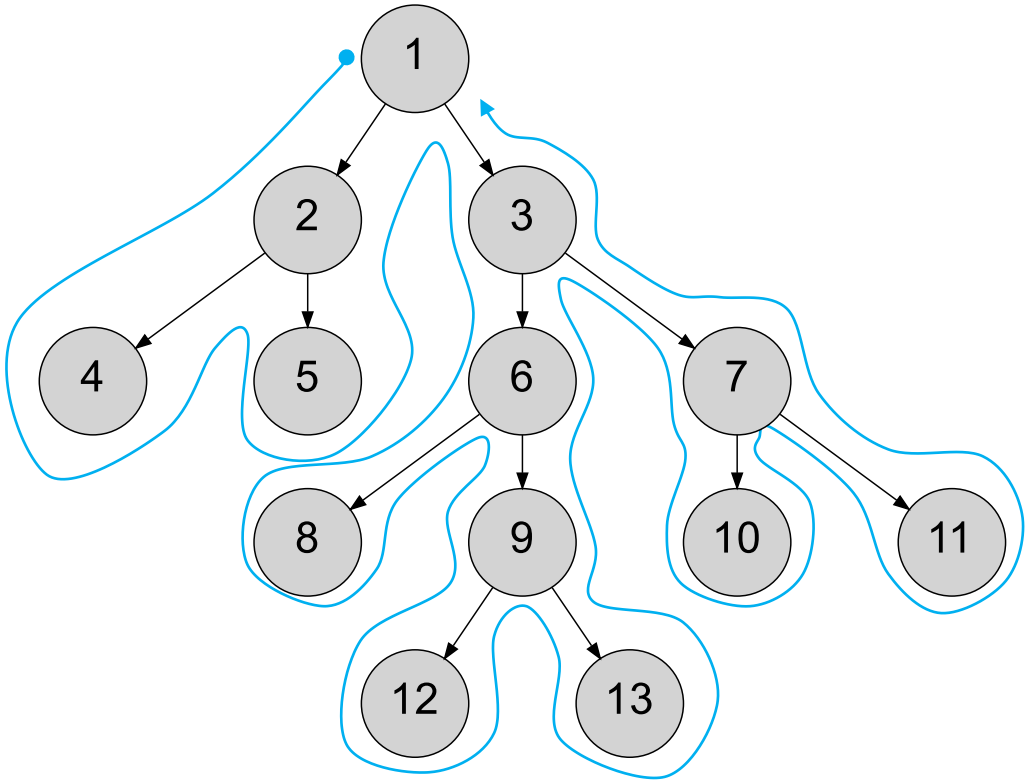


- **Représentation par liste d'adjacence**
  Chaque nœud contient un lien vers le 1er fils et une liste des frères droit.

Parcours

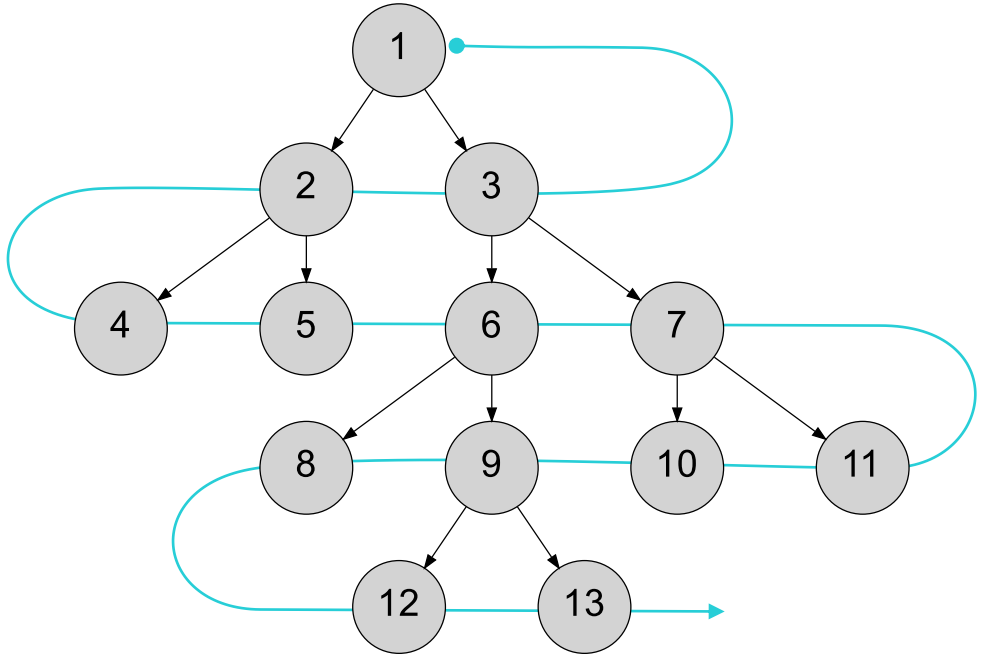
Parcours en profondeur

parcours branche par branche.



Parcours en largeur

parcours niveau par niveau.



Parcours en profondeur

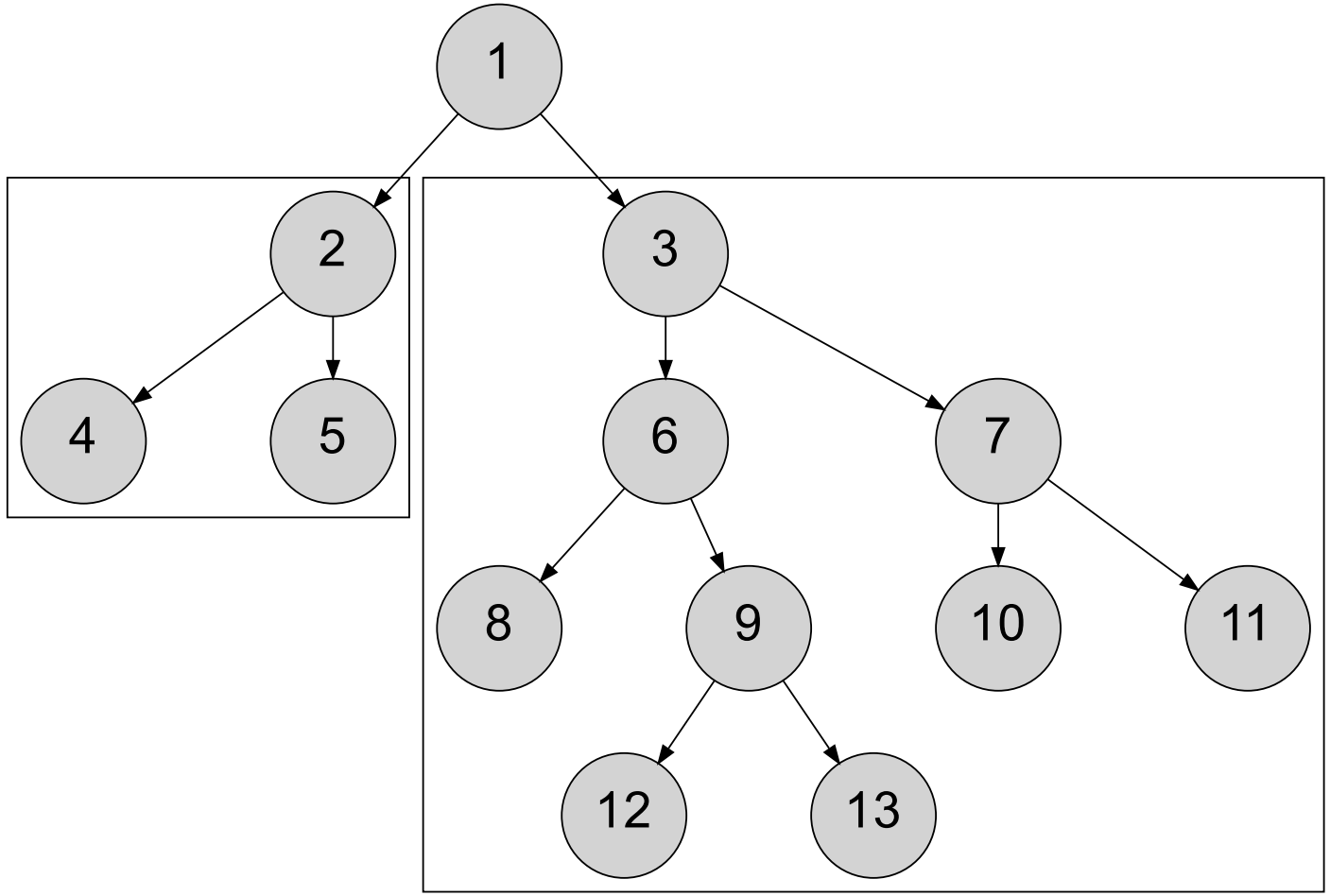
Parcours préfixe

Soit un arbre non vide $A = (r, a_1, a_2, \ldots, a_k)$,

$P_{préfixe}(A) = (r).P_{préfixe}(a_1).\ldots.P_{préfixe}(a_k)$

Résultat

(1,2,4,5,3,6,8,9,12,13,7,10,11)



Parcours en profondeur

Parcours infixe (symétrique) – uniquement pour les arbres binaires

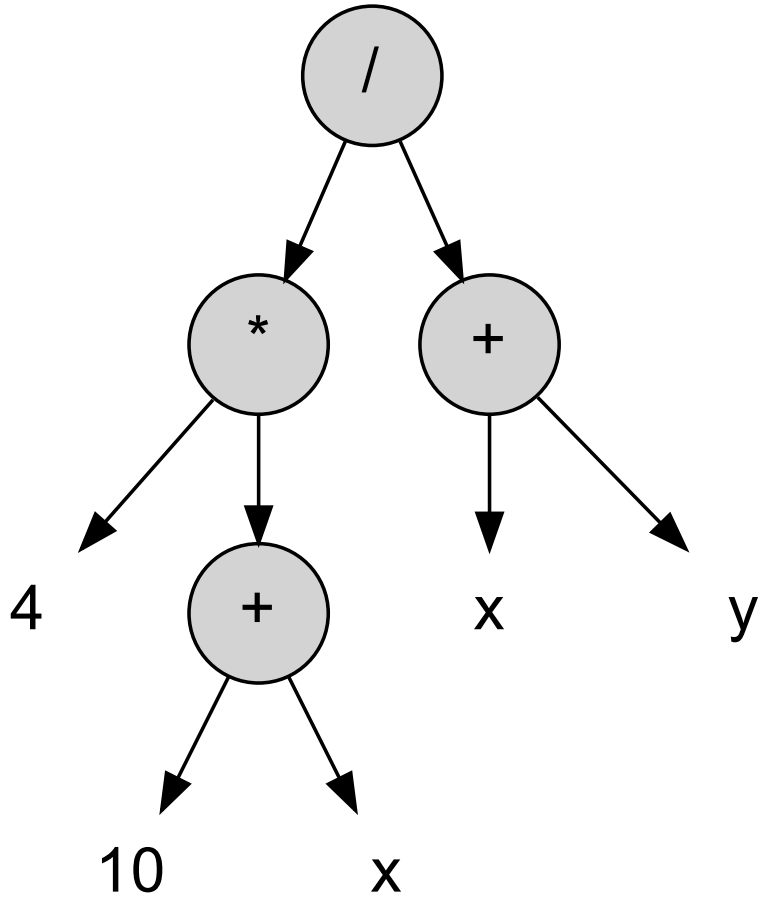
Soit un arbre non vide $A = (r, A_{gauche}, A_{droit})$,

$$P_{infixe}(A) = P_{infixe}(A_{gauche}).(r).P_{infixe}(A_{droit})$$

ou A_{gauche} désigne le sous arbre gauche et
A_{droit} le sous arbre droit du nœud r.

Résultat

$[[4 * [10 + x]]/[x + y]]$



Parcours en profondeur

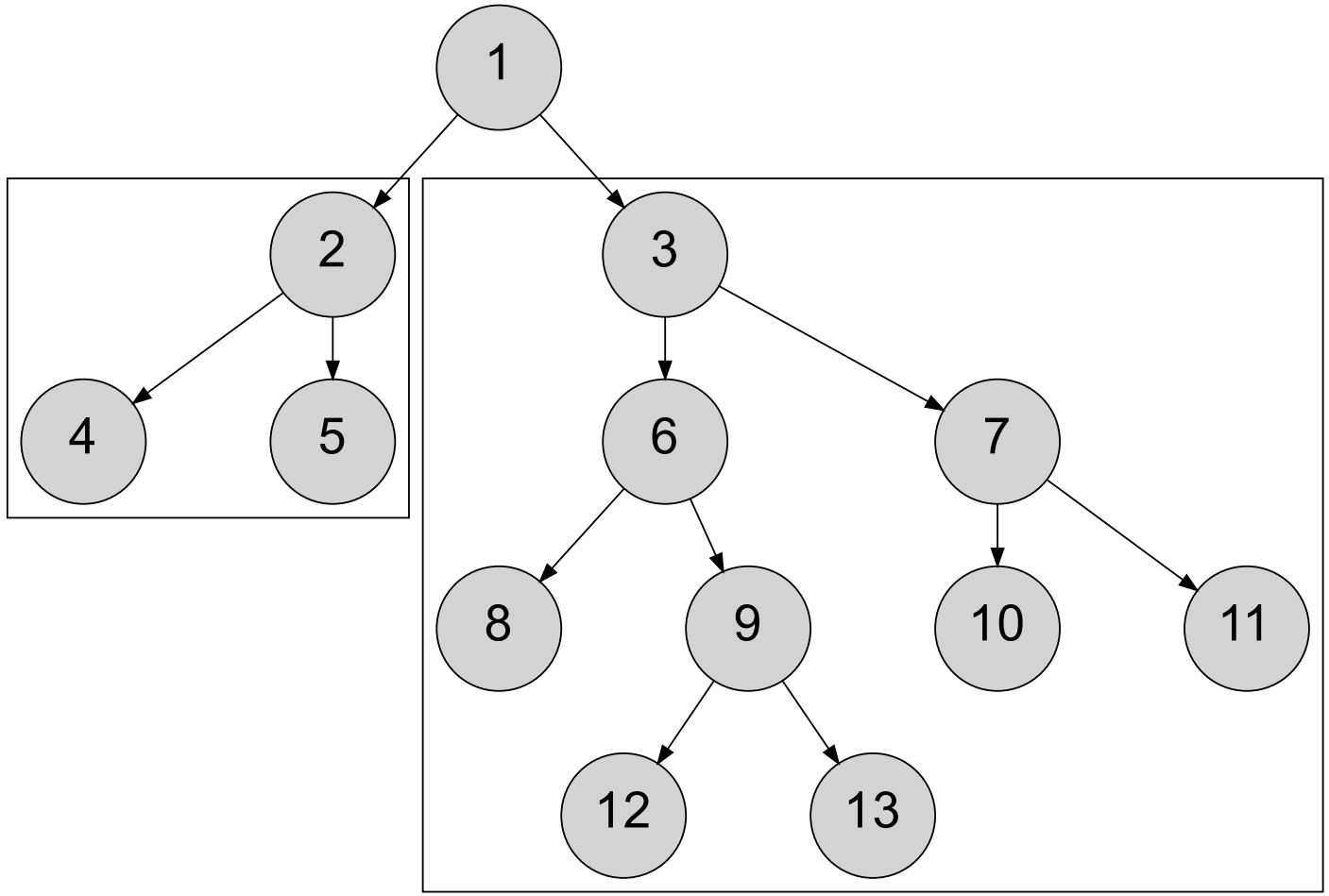
Parcours suffixe

Arbre non vide $A = (r, a_1, a_2, \ldots, a_k)$

$P_{suffixe}(A) = P_{suffixe}(a_1). \ldots . P_{suffixe}(a_k). (r)$

Résultat

(4,5,2,8,12,13,9,6,10,11,7,3,1)

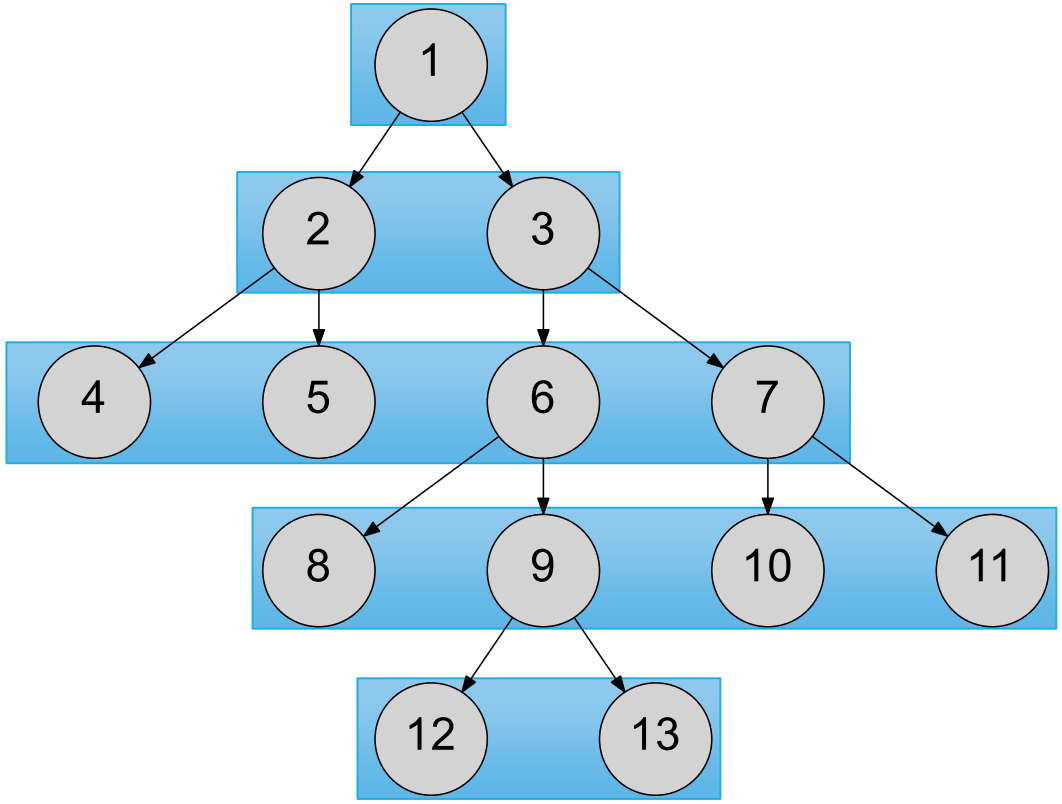


Parcours en largeur

Parcours hiérarchique

On commence par la racine et on visite les nœuds niveau par niveau, de gauche à droite On se déplace de frère en frère pour chaque niveau.

Résultat
(1,2,3,4,5,6,7,8,9,10,11,12,13)



Les arbres binaires de recherche

Définition

Un arbre binaire de recherche (ABR) est un arbre binaire dans lequel chaque nœud possède une clé et vérifiant les propriétés suivante :

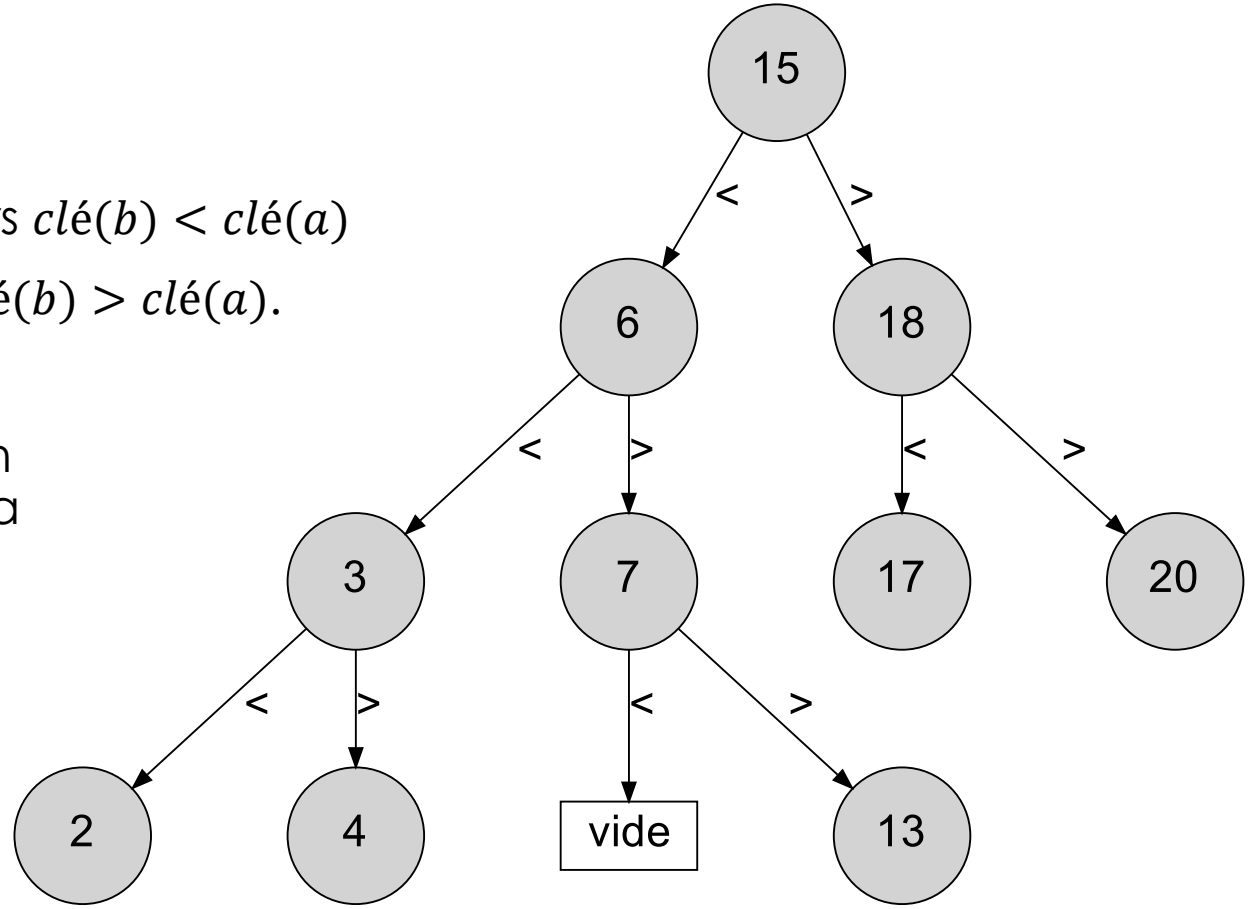
Soit a un nœud de l'arbre :

- a a au plus 2 fils (un fils gauche et un fils droit)
- Si b est un nœud du sous arbre gauche de a alors $clé(b) < clé(a)$
- Si b est un nœud du sous arbre droit de a alors $clé(b) > clé(a)$.

Chaque nœud contient un élément et la répartition des éléments dans l'arbre va permettre de guider la recherche en faisant des comparaisons.

Remarque :
Le parcours infixe d'un ABR produit la suite des éléments triés par ordre croissant.

Ici : (2,3,4,6,7,13,15,17,18,20)



Recherche dans un ABR

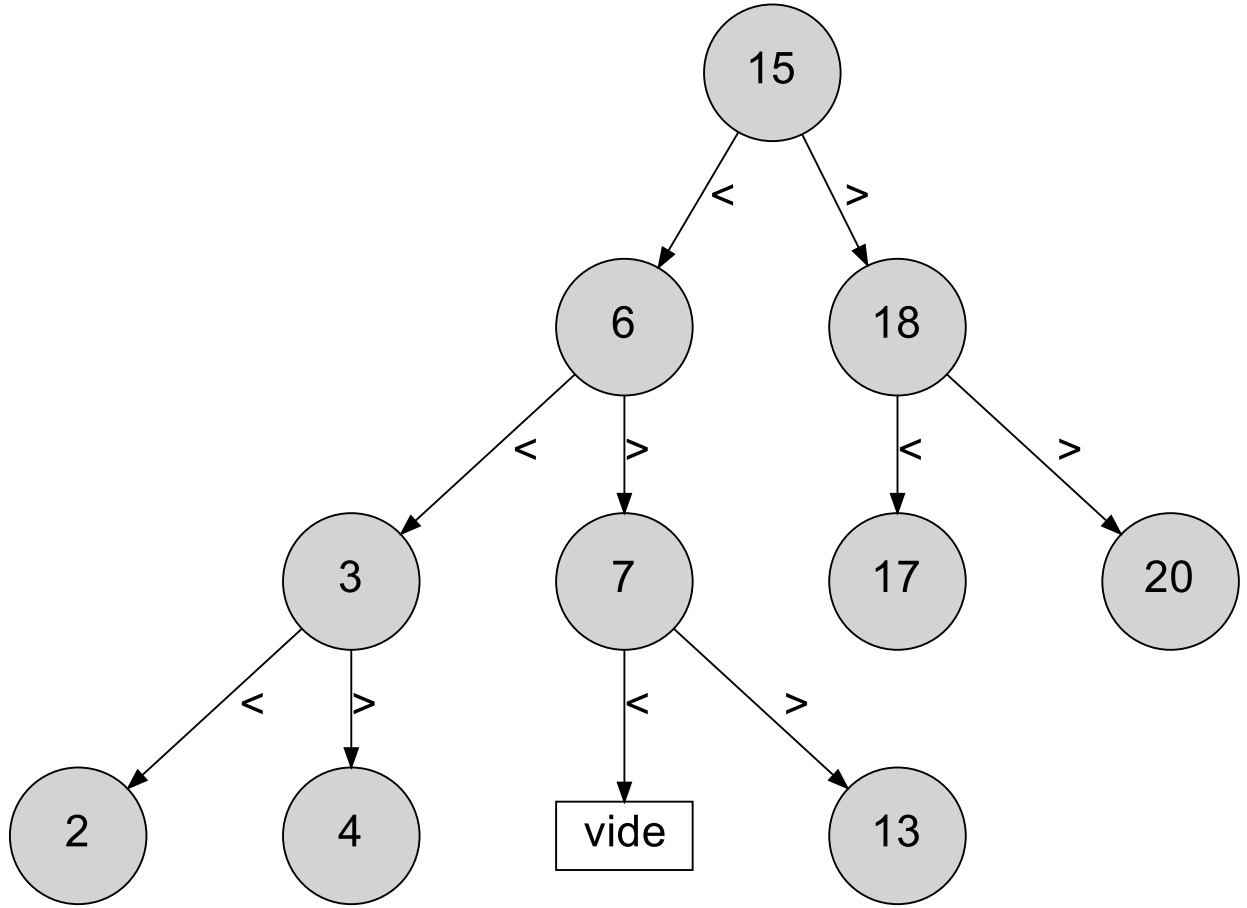
Pour rechercher un élément dans un ABR :

1. On compare cet élément au contenu de la racine.
2. Si c'est l'élément recherché, on a trouvé et on s'arrête là ;
3. Sinon,
    1. Si l'élément recherché est inférieur à la racine, on recommence dans le sous arbre gauche
    2. Si l'élément recherché est supérieur à la racine, on recommence dans le sous arbre droit.
4. Si le sous arbre est vide alors, l'élément n'a pas été trouvé.

Recherche dans un ABR

Exemple de recherche dans un ABR

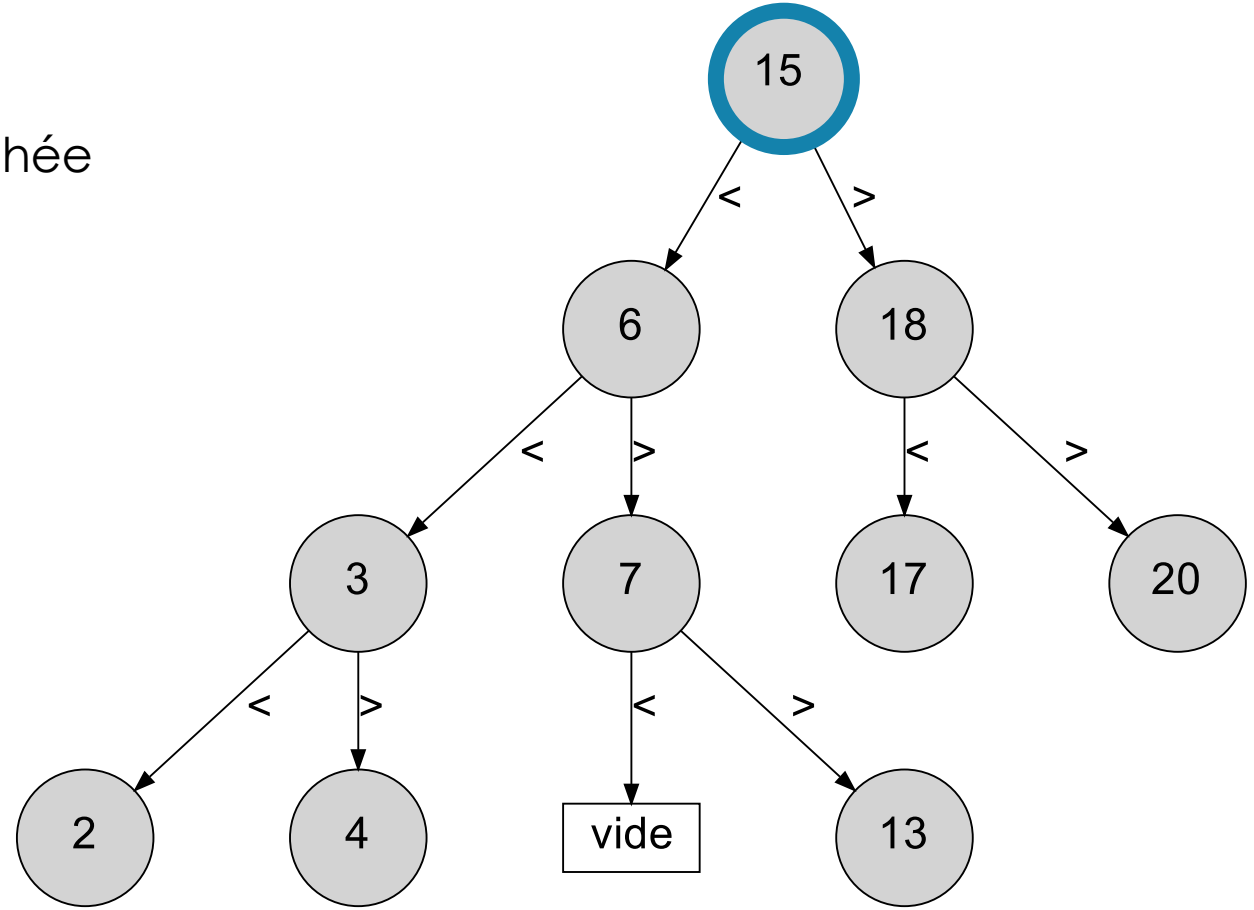
Recherche de 7.



Recherche dans un ABR

Exemple de recherche dans un ABR
Recherche de 7.

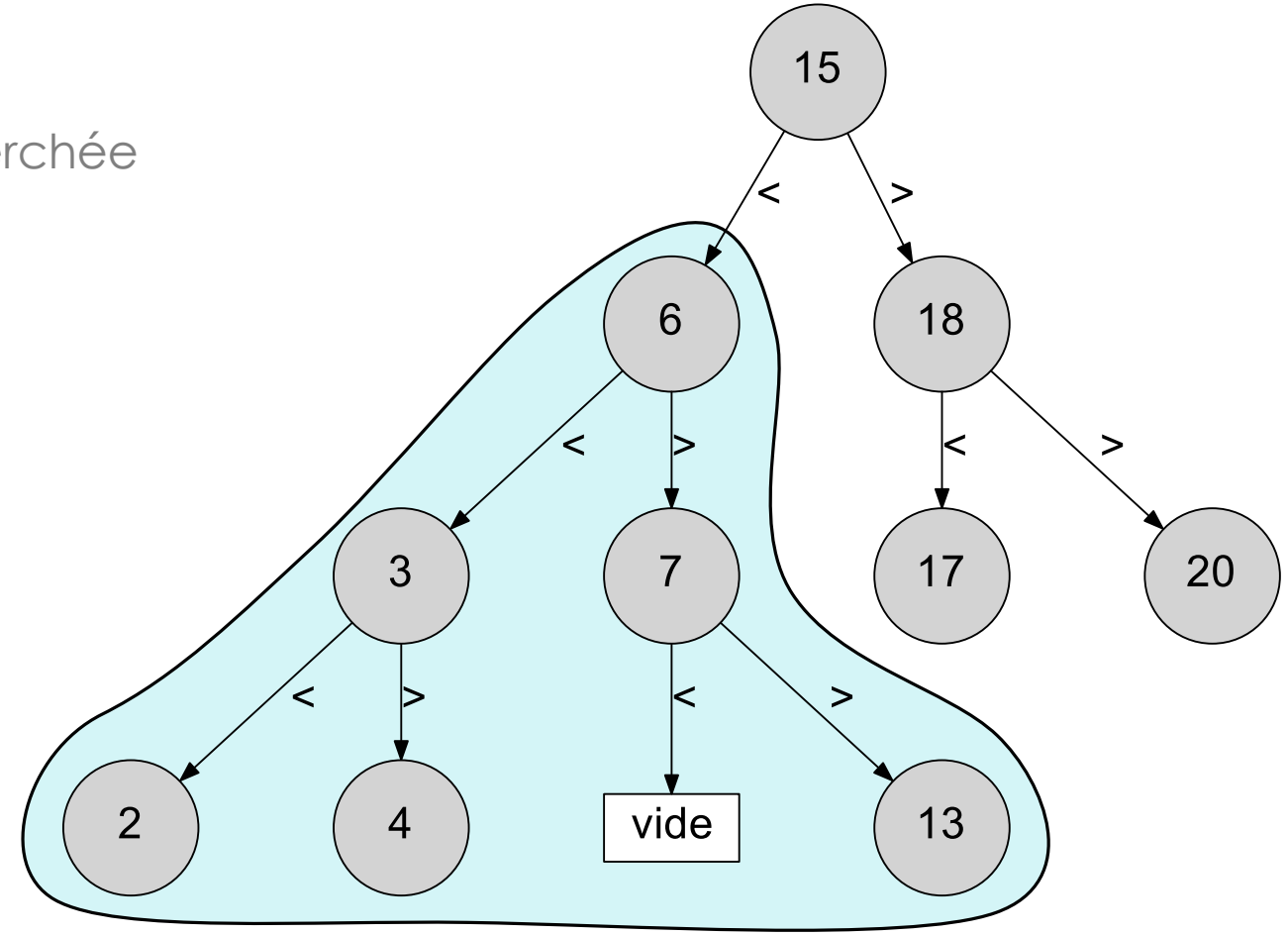
1. 15≠7 : la racine ne contient pas la valeur recherchée



Recherche dans un ABR

Exemple de recherche dans un ABR
Recherche de 7.

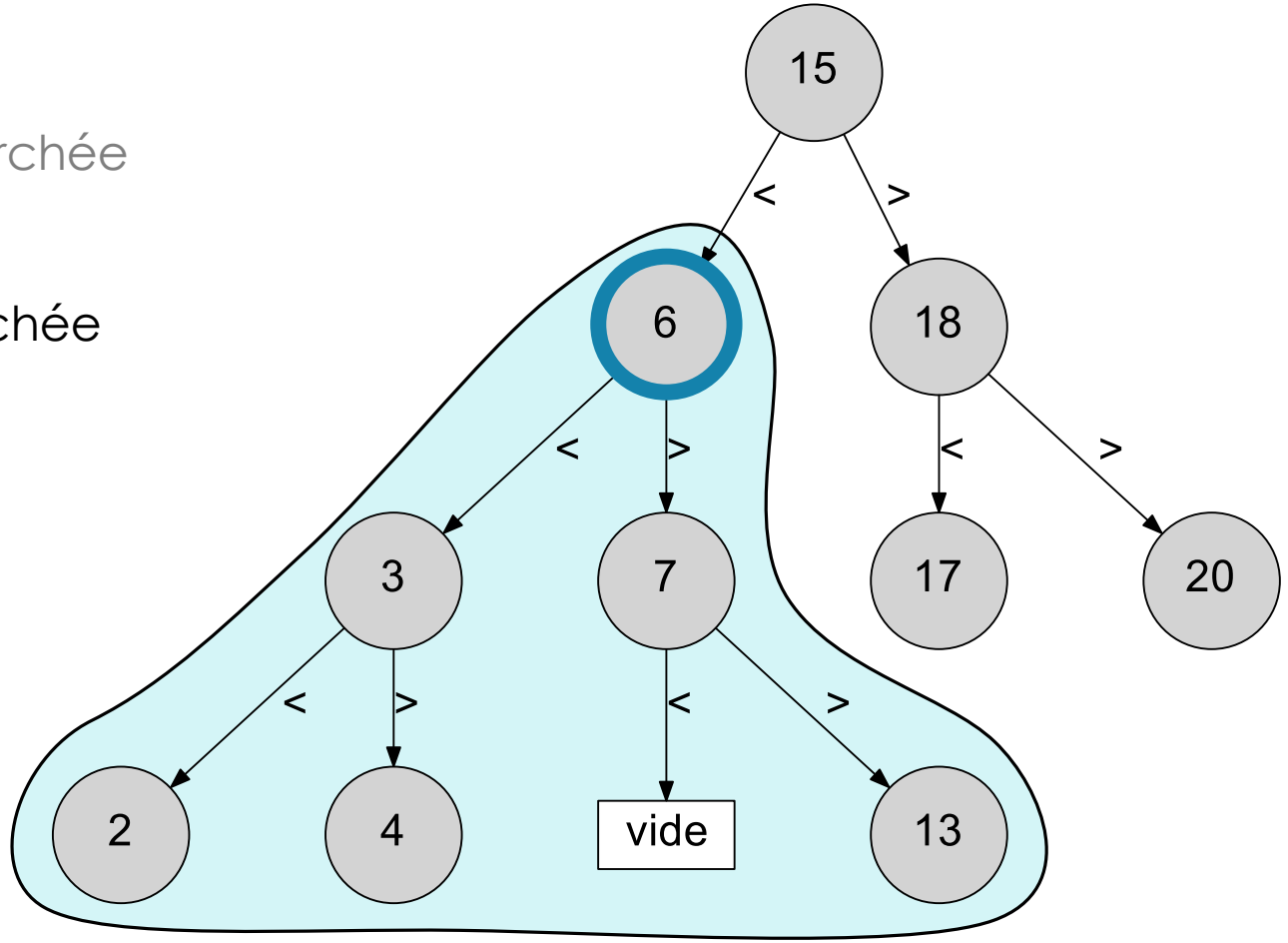
1. 15≠7 : la racine ne contient pas la valeur recherchée
2. 7<15 : on part dans le sous arbre gauche



Recherche dans un ABR

Exemple de recherche dans un ABR
Recherche de 7.

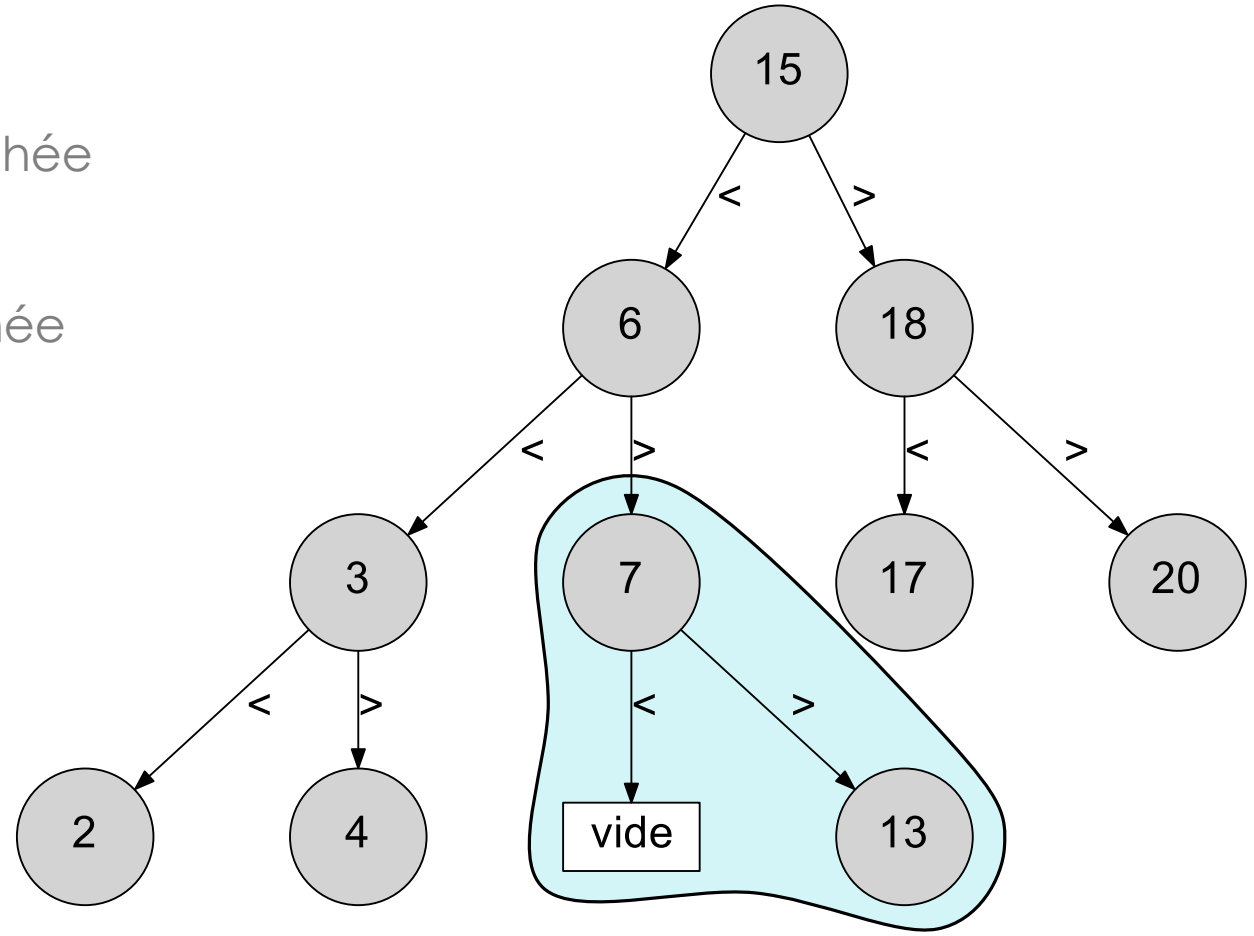
1. 15≠7 : la racine ne contient pas la valeur recherchée
2. 7<15 : on part dans le sous arbre gauche
3. 6≠7 : la racine ne contient pas la valeur recherchée



Recherche dans un ABR

Exemple de recherche dans un ABR
Recherche de 7.

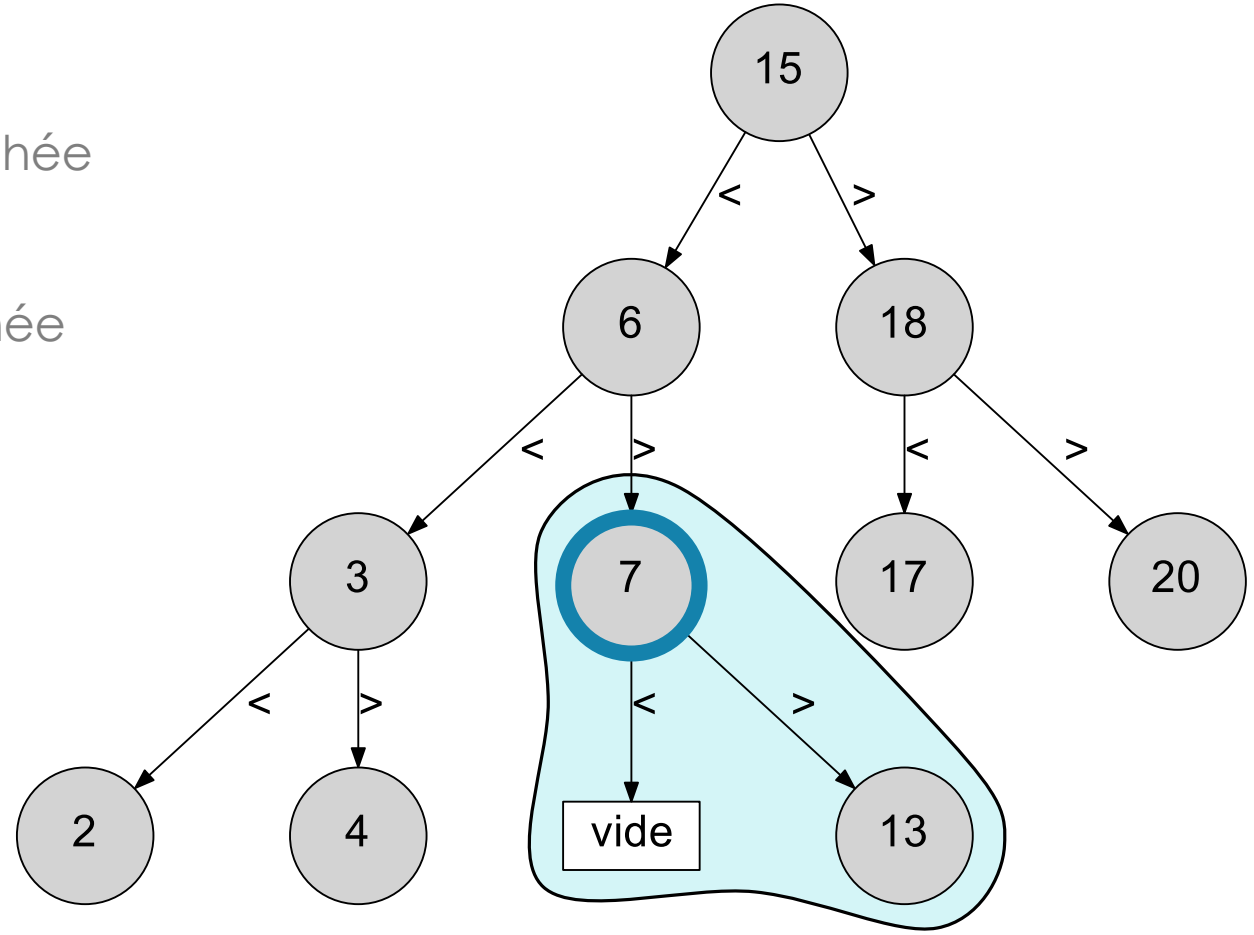
1. 15≠7 : la racine ne contient pas la valeur recherchée
2. 7<15 : on part dans le sous arbre gauche
3. 6≠7 : la racine ne contient pas la valeur recherchée
4. 7>6 : on part dans le sous arbre droit



Recherche dans un ABR

Exemple de recherche dans un ABR
Recherche de 7.

1. 15≠7 : la racine ne contient pas la valeur recherchée
2. 7<15 : on part dans le sous arbre gauche
3. 6≠7 : la racine ne contient pas la valeur recherchée
4. 7>6 : on part dans le sous arbre droit
5. **7=7 : la racine contient la valeur recherchée**



Insertion dans un ABR

L'insertion peut se faire de deux manières

Insertion aux feuille de l'ABR ;

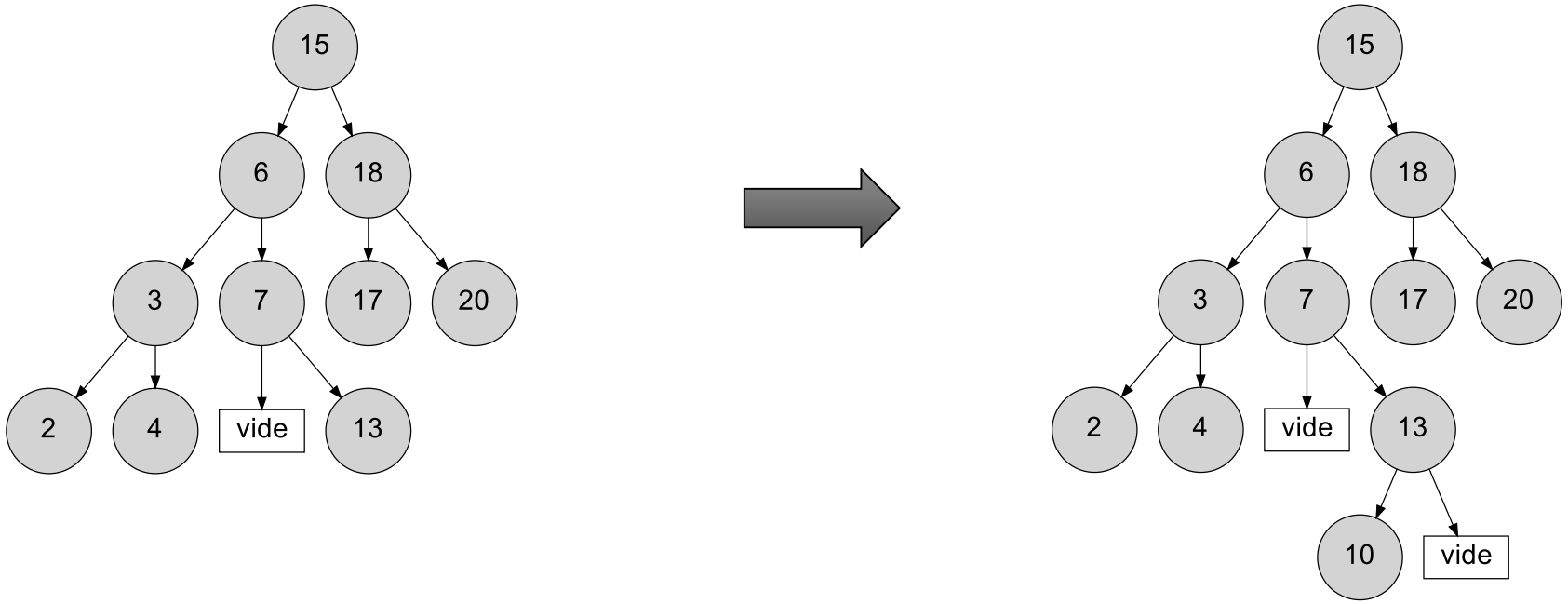
Insertion à la racine de l'ABR.

Insertion dans un ABR

Insertion d'une valeur x aux feuilles

On recherche la valeur x à insérer dans l'arbre. Lorsque l'on arrive au niveau d'une feuille f, si x est inférieur au contenu de f, on ajoute un nouveau nœud de valeur x comme fils gauche, sinon, on ajoute un nouveau nœud de valeur x comme fils droit.

Exemple :
Insertion de 10.



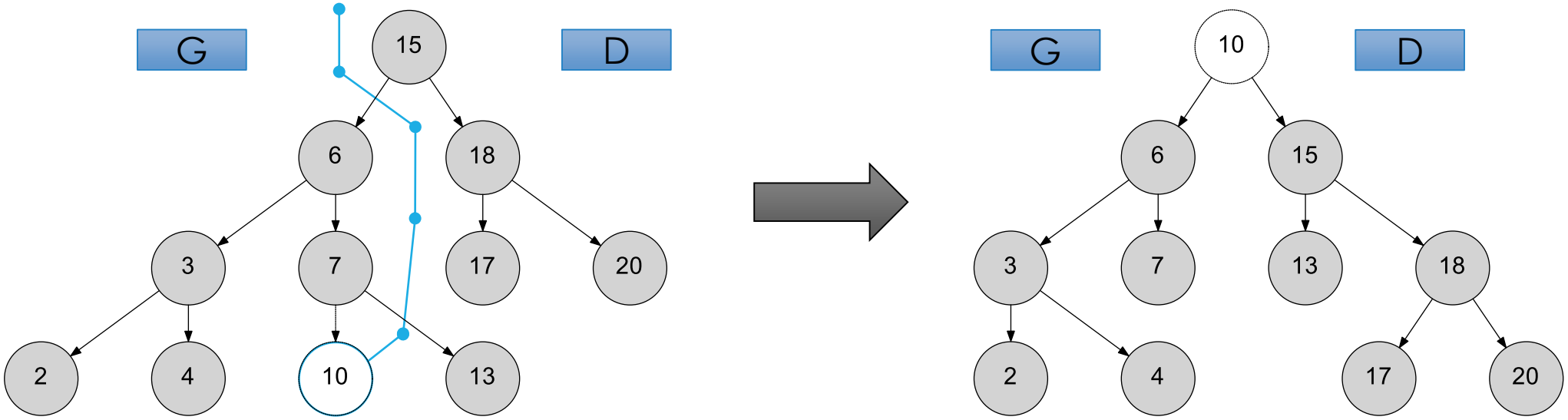
L'avantage de cette méthode est que pour un arbre de n nœuds, elle se fait en $O(\log n)$ dans le cas moyen et en $O(n)$ dans le pire des cas. En revanche, cette approche peut conduire à des arbres déséquilibrés (comme des peignes) car elle modifie la structure

Insertion dans un ABR

Insertion d'une valeur x à la racine

Pour ajouter un élément x à la racine, il faut définir un chemin de coupe afin d'obtenir deux sous arbres G(éléments $\leq x$) et D(éléments $> x$). x devient la nouvelle racine de l'ABR constitué par $< G, x, D >$.

Exemple :
Insertion de 10.



Suppression dans un ABR

Suppression d'un nœud a dans un ABR.

On commence par chercher le nœud a dans l'arbre. On a alors les cas suivants :

- Si le nœud a est une feuille : on supprime directement ;
- Si le nœud a a un fils b : on remplace a par b ;
- Si le nœud a a deux fils : il y a deux solutions
  - Remplacer le nœud $\boldsymbol{a}$ par celui qui lui est immédiatement inférieur i.e. le nœud qui contient le plus grand élément de son sous arbre gauche.
  - Remplacer le nœud $\boldsymbol{a}$ par celui qui lui est immédiatement supérieur i.e. par celui qui contient le plus petit élément de son sous arbre droit.

Si l 'on ne fait pas attention, l'opération de suppression peut rapidement conduire à des arbres déséquilibrés.

ABR avec critère d'équilibre

Les arbres binaires de recherche

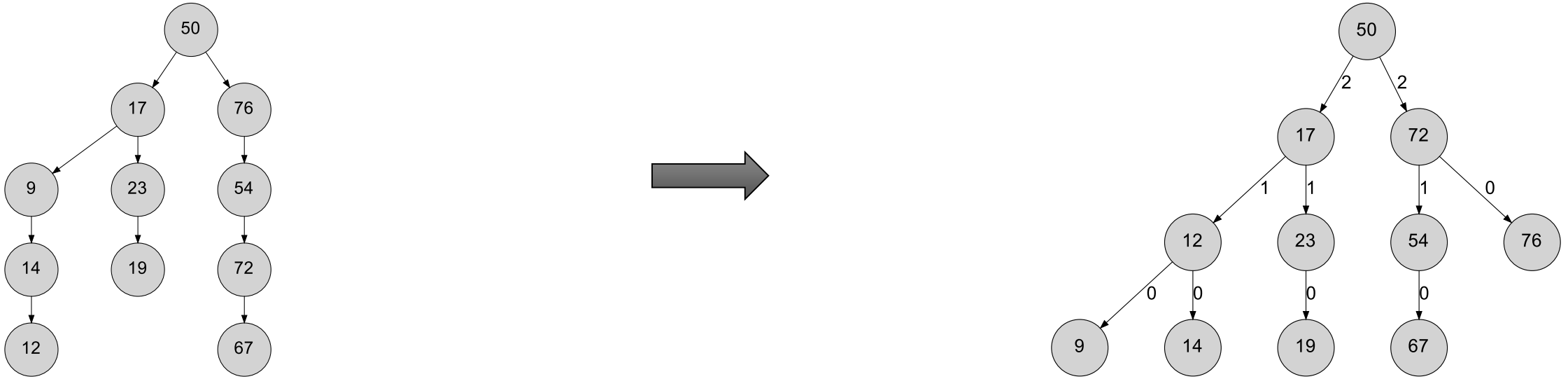
Arbres binaires équilibrés

Définition

Un arbre binaire à critère d'équilibre est un arbre binaire qui possède un, ou plusieurs, critères d'équilibres. Ces critères peuvent porter sur différents éléments comme : nombre de nœuds des fils, différence de hauteurs entre les deux sous arbres droit et gauche, etc.

Rôle

Le rôle de ces critères est d'éviter la création d'arbres dégénérés comme des arbres filiformes par exemple. Dans le cas d'un ABR, cette dégénérescence peut se traduire par une perte de performance lors des opérations de recherche.

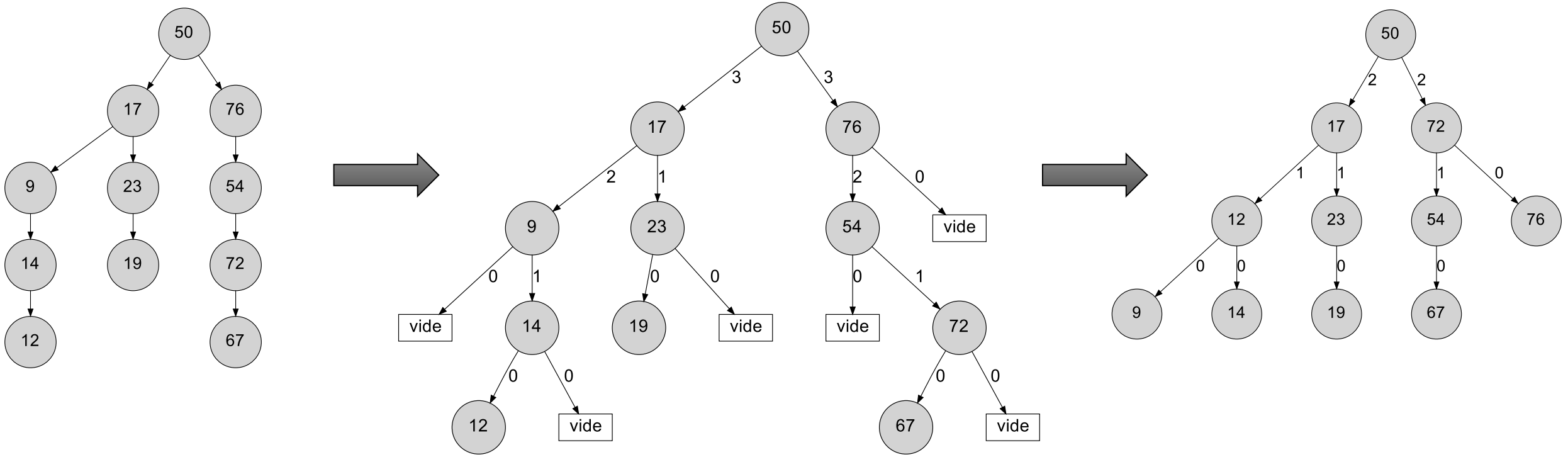


Arbres AVL

Définition

Premier ABR automatiquement équilibrés. Le critère porte sur la hauteur des deux sous arbres d'un même nœud qui doit être telle qu'elle ne diffère pas plus d'un.

Exemple

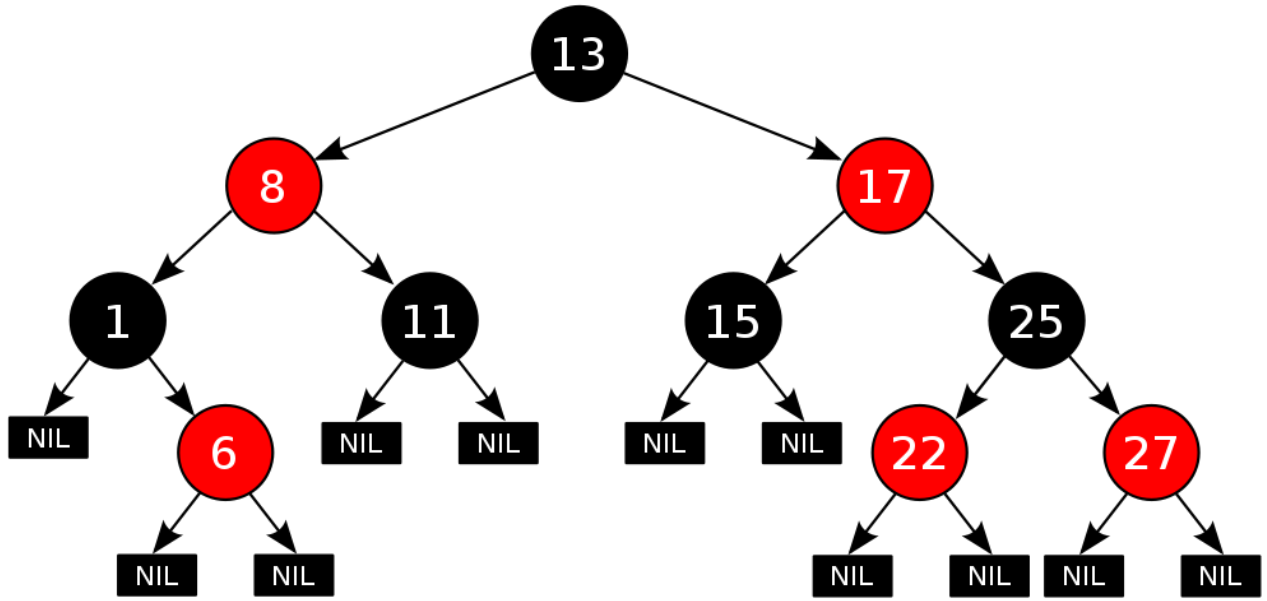


Arbre rouge et noir

Définition
Un arbre bicolore est un arbre binaire de recherche dans lequel on associe à chaque nœud une couleur. Cette couleur est soit rouge soit noire.

Les nœuds des arbres rouge et noir doivent vérifier les critères suivants :

- La racine est noire.
- Le parent d'un nœud rouge est noir.
- Le chemin de chaque feuille à la racine contient le même nombre de nœuds noirs.



Opérations sur les arbres équilibrés

Les opérations **d'insertion** et de **suppression** sur les arbres à critère d'équilibre peuvent être **complexes** car il faut **maintenir les critères**. Très souvent les insertions et suppressions font appelles à de **nombreuses études de cas**.

La **base** des opérations **d'insertion** et de **suppression** sur les arbres à critères d'équilibres repose sur **les opérations de rotations droite et gauche**.

Opération de rotation (droite/gauche)

La rotation d'un ABR permet de changer la structure de l'arbre sans invalider l'ordre des éléments. La rotation revient à faire remonter un nœud dans l'arbre et à en faire redescendre un autre.

Rotation gauche:

$pivot \leftarrow racine.filsDroit$
$racine.filsDroit \leftarrow pivot.filsGauche$
$pivot.filsGauche \leftarrow racine$
$racine \leftarrow pivot$

Rotation droite :

$pivot \leftarrow racine.filsGauche$
$racine.filsGauche \leftarrow pivot.filsDroit$
$pivot.filsDroit \leftarrow racine$
$racine \leftarrow pivot$

Exemple: