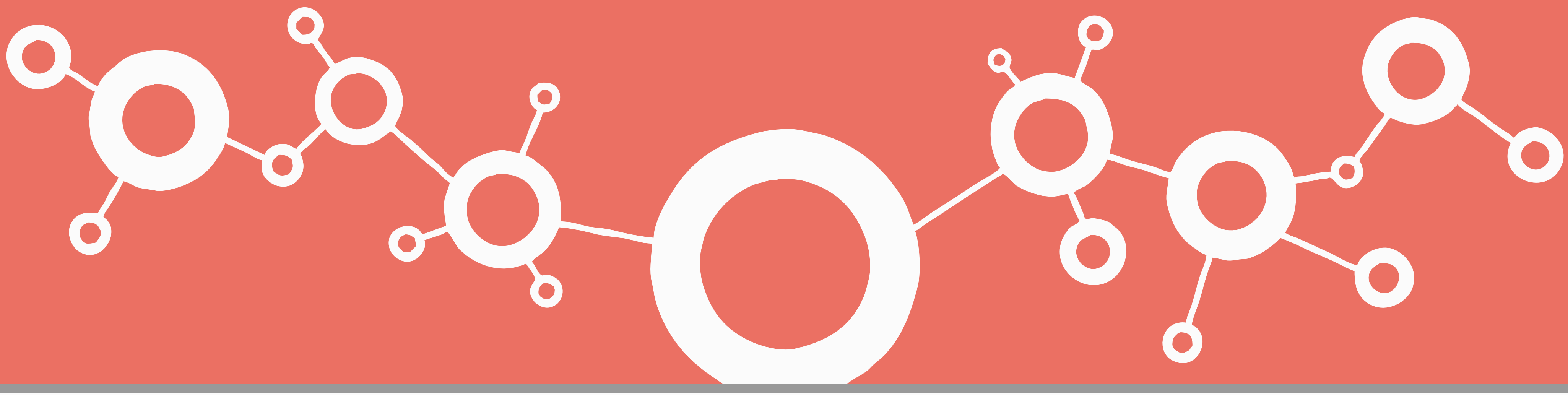
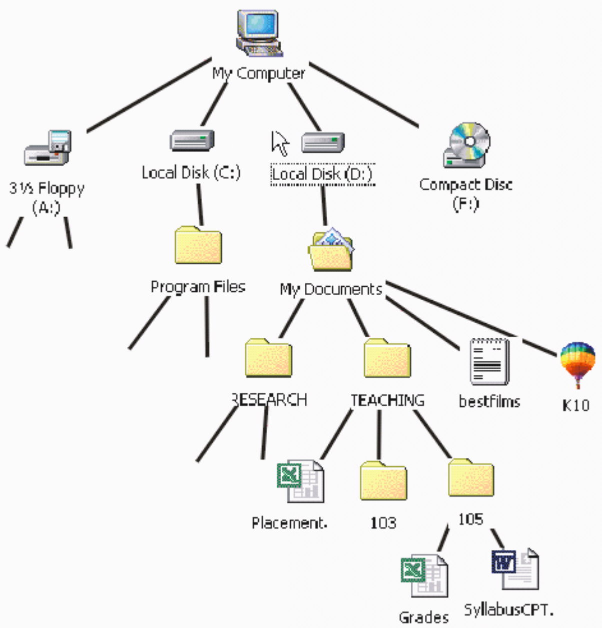
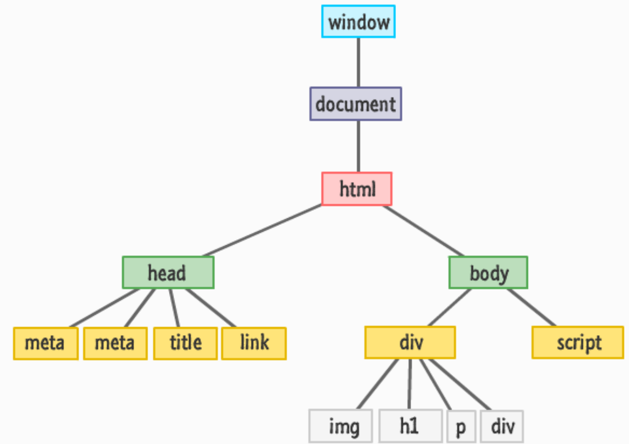


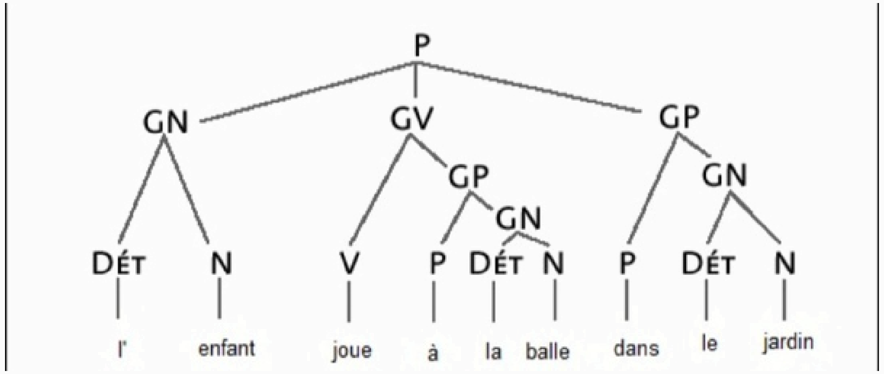
Ch. 5 : Arbres

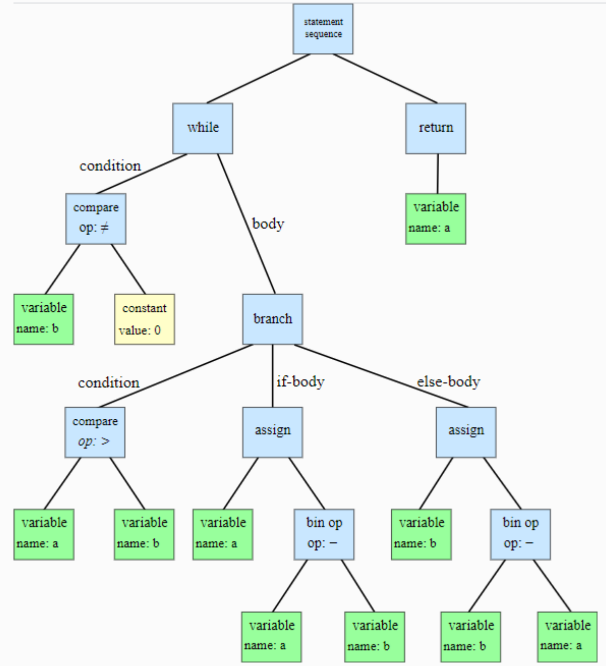
Structures arborescentes

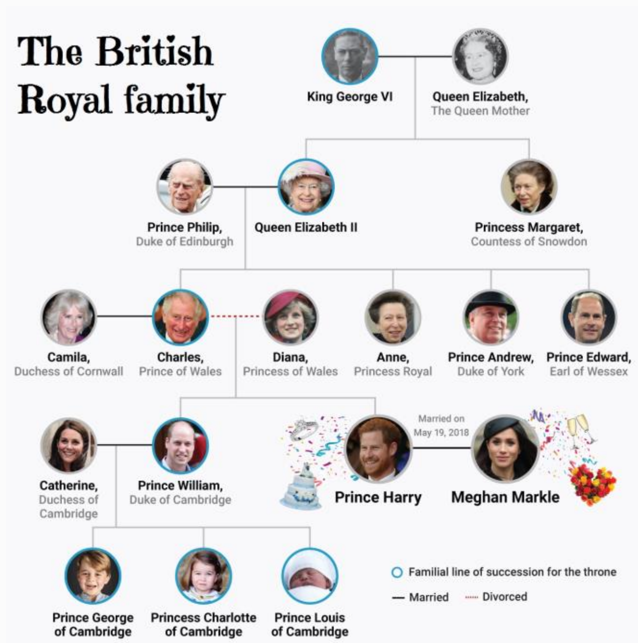
NOMBREUX USAGES EN INFORMATIQUE









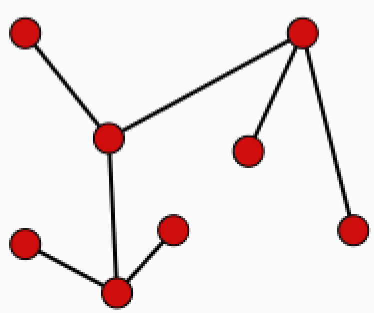


Structures arborescentes

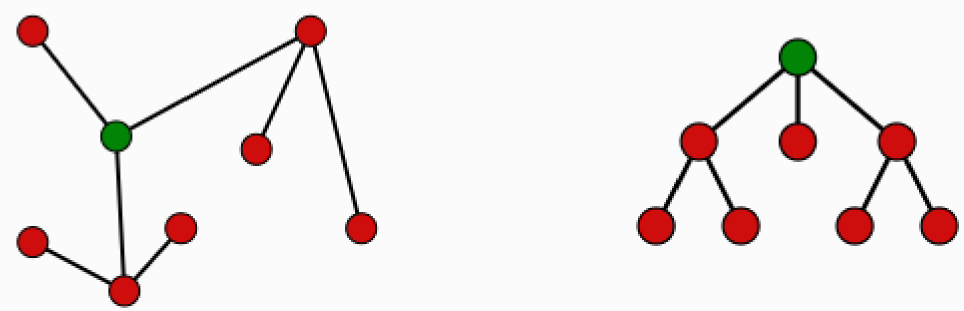
TERMINOLOGIE

Arbre

- mathématiques : *graphe* connexe (= « en un morceau ») et sans cycle
- informatique : arbre *enraciné* ou *arborescence*



arbre

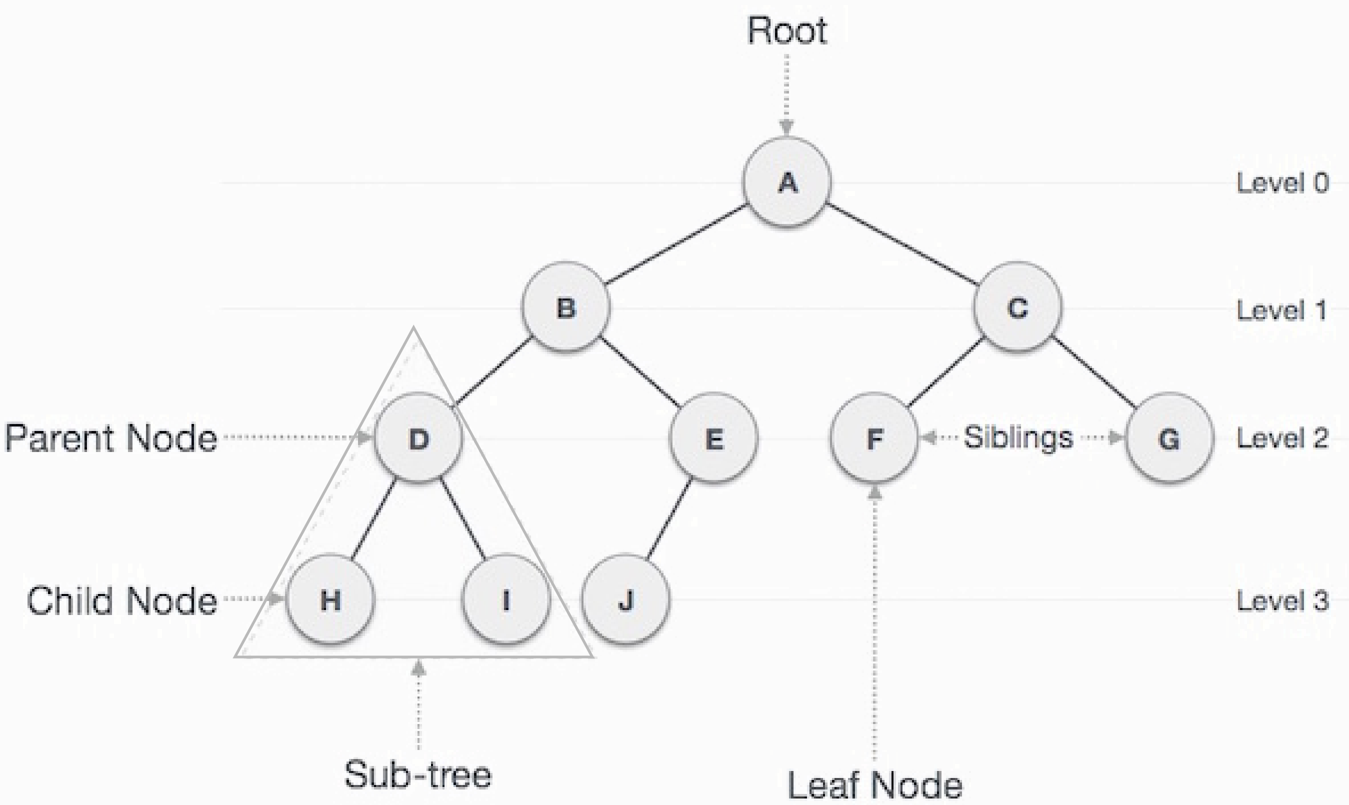


arbre enraciné ou arborescence

Un arbre est une généralisation d'une liste chaînée

Structures arborescentes

TERMINOLOGIE

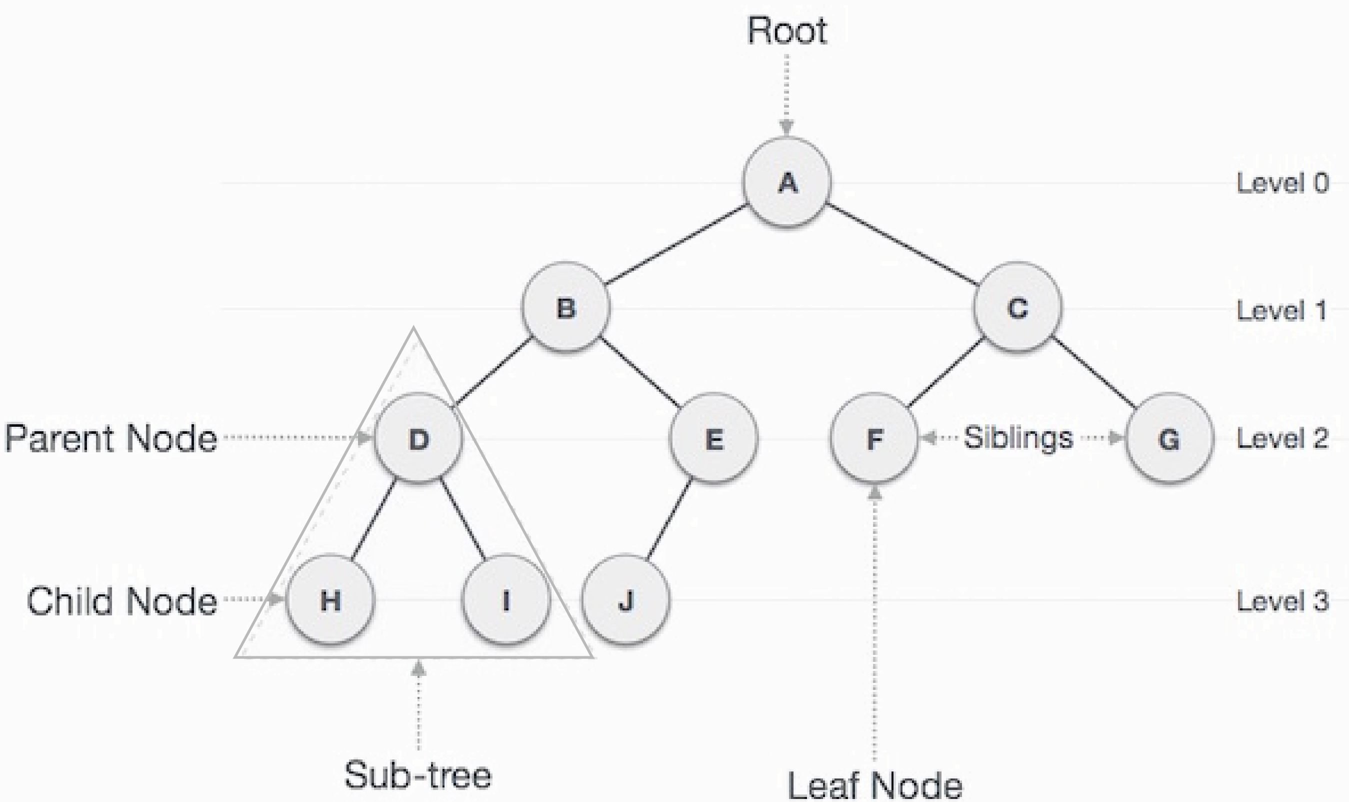


Racine	nœud au sommet de l'arbre
Parent	nœud avec un enfant
Frères	nœuds avant le même parent
Feuilles	nœuds sans enfant
Taille de l'arbre	nombre de nœuds dans l'arbre
Arête	lien entre deux nœuds voisins

Propriété : un arbre est une structure de données récursive

Structures arborescentes

TERMINOLOGIE



Chemin	succession d'arêtes adjacentes
Longueur d'un chemin	nombre d'arêtes du chemin
Niveau k	nœuds à distance k de la racine
Profondeur d'un nœud	longueur du chemin (direct) entre la racine et ce nœud
Hauteur d'un nœud	longueur du plus long chemin entre ce nœud et une feuille
Hauteur de l'arbre	Hauteur de la racine

Propriété : deux nœuds d'un arbre sont reliés par un unique chemin (direct ou *élémentaire*)

Structures arborescentes

OPERATIONS COMMUNES SUR LES ARBRES

Opérations classiques (penser à un système de fichiers)

- Enumérer tous les éléments de l'arbre
- Enumérer les éléments d'un sous-arbre
- Rechercher un élément
- Ajouter un élément
- Ajouter un sous-arbre (greffage ou *grafting*)
- Supprimer un élément
- Supprimer un sous-arbre (élagage ou *pruning*)
- Trouver l'ancêtre commun le plus proche de deux nœuds

Applications : logiciels de généalogie, ou dans les compilateurs de langages orientés objet avec héritage pour déterminer la classe mère commune la plus proche de deux classes

Parcours d'arbres

COMMENT RETROUVER UNE INFORMATION DANS UN ARBRE

Un arbre est une structure de données *non linéaire* :

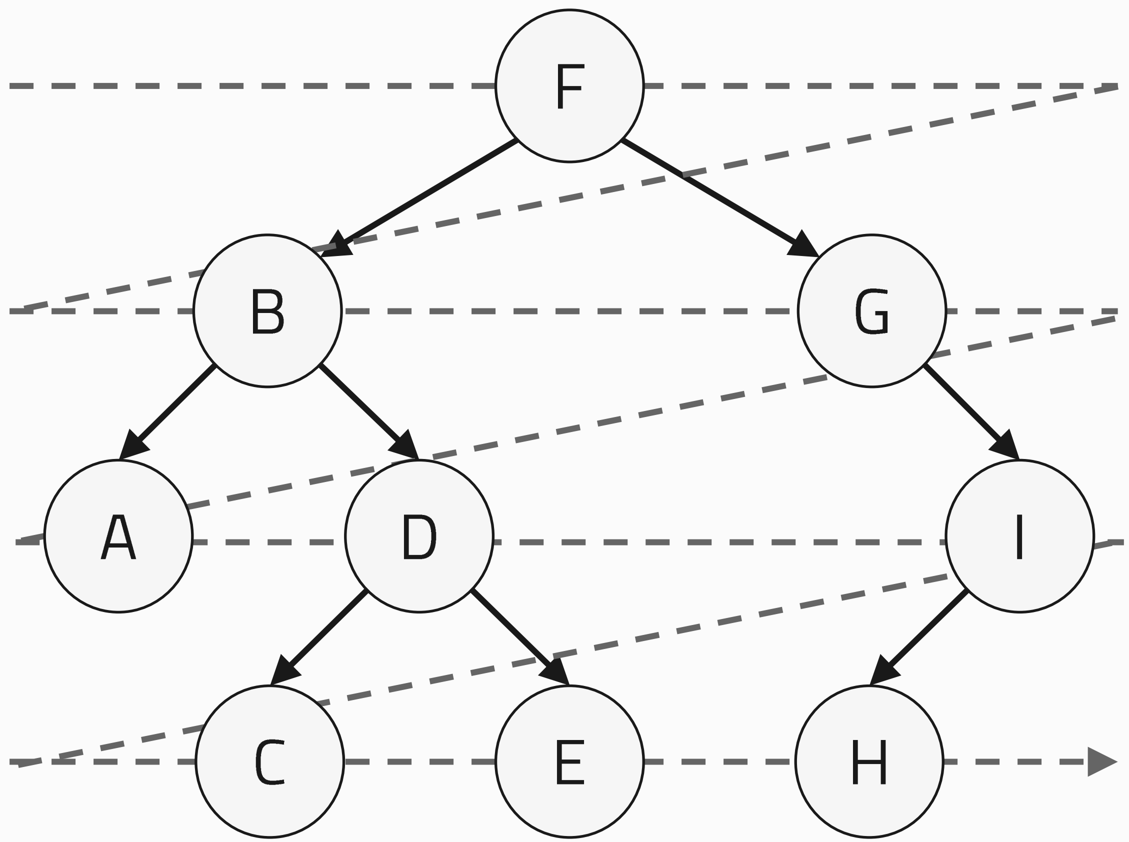
il existe donc plusieurs manières de parcourir ses

éléments

Parcours en largeur

OU BFS (BREADTH-FIRST SEARCH)

Principe : on parcourt les nœuds niveau par niveau, de gauche à droite :

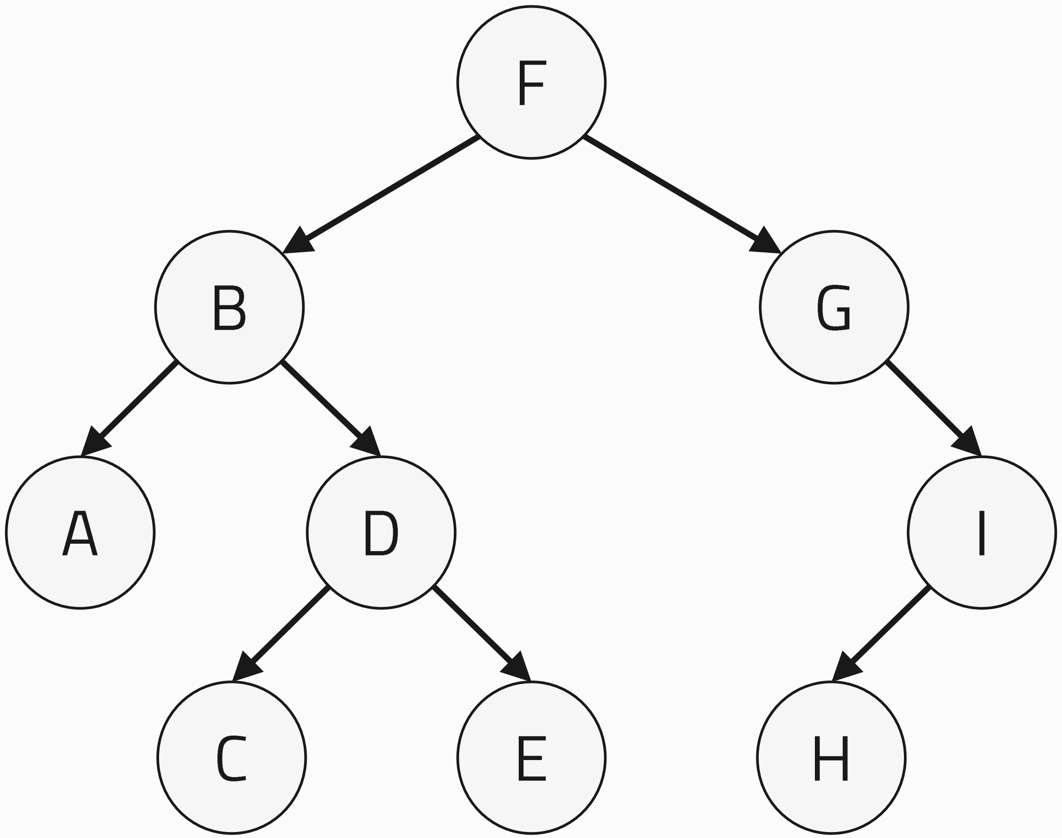


Ordre de parcours : F B G A D I C E H

Parcours en largeur

OU BFS (BREADTH-FIRST SEARCH)

Implémentation itérative : on utilise une *file* pour mémoriser les nœuds à visiter



File d'attente

F	B	G	A	D	I	C	E	H

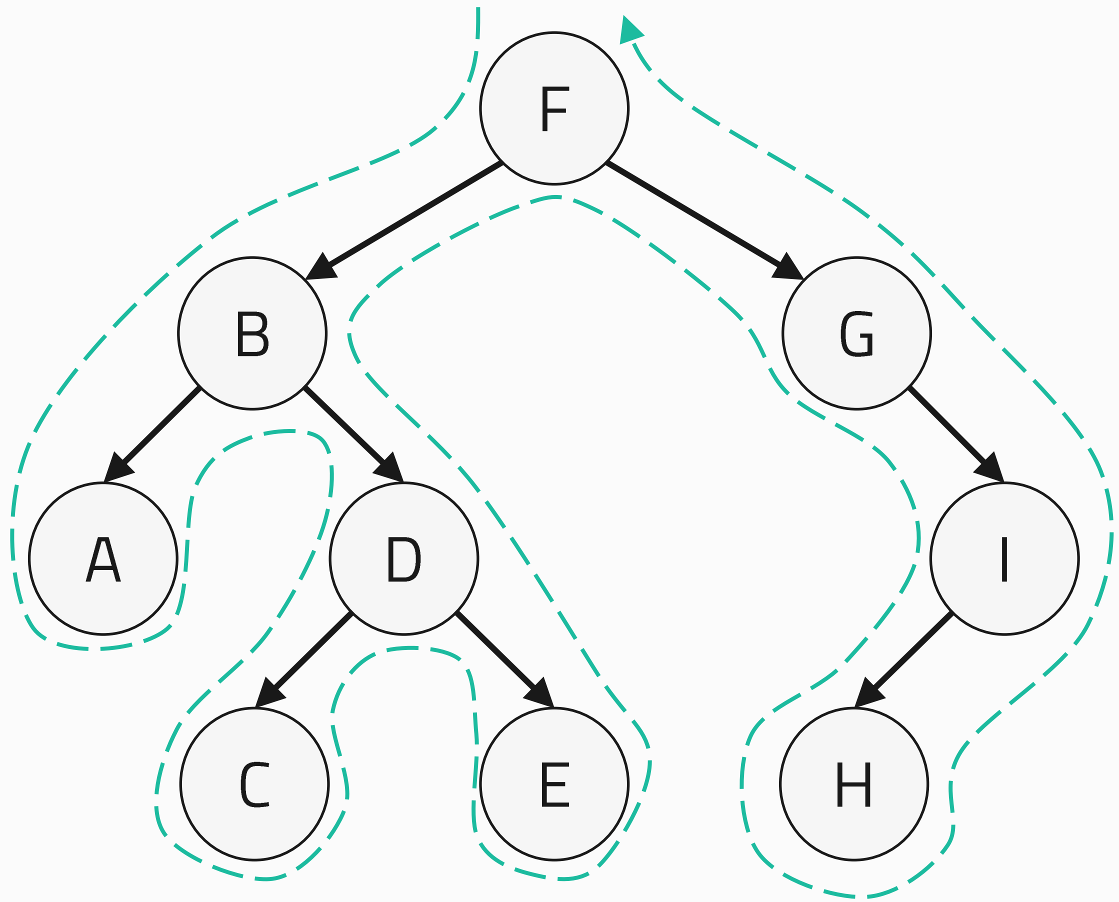
Ordre de parcours

F	B	G	A	D	I	C	E	H

Parcours en profondeur

OU DFS (DEPTH-FIRST SEARCH)

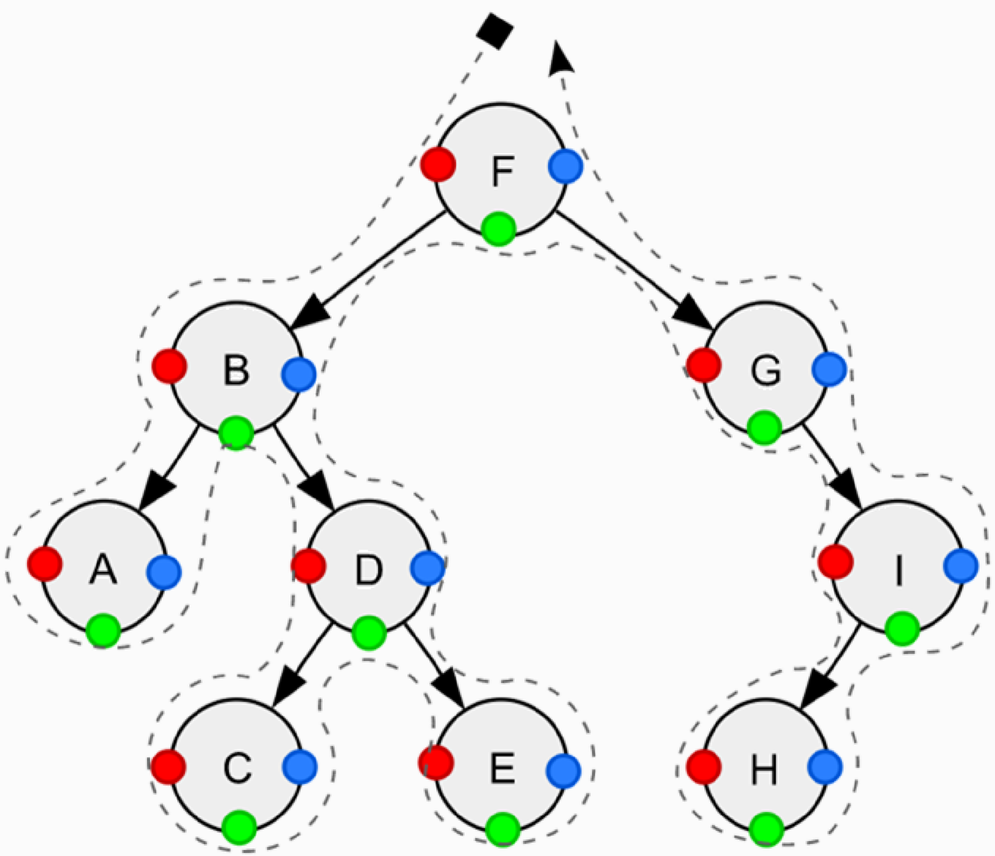
Principe : on descend le plus possible vers les enfants d'un nœud avant de passer au nœud frère



Parcours en profondeur

OU DFS (DEPTH-FIRST SEARCH)

3 ordres possibles pour marquer les sommets rencontrés :



Ordre préfixe : on note un nœud dès qu'on passe à gauche :

F B A D C E G I H

Ordre infixe : on note un nœud dès qu'on passe dessous :

A B C D E F G H I

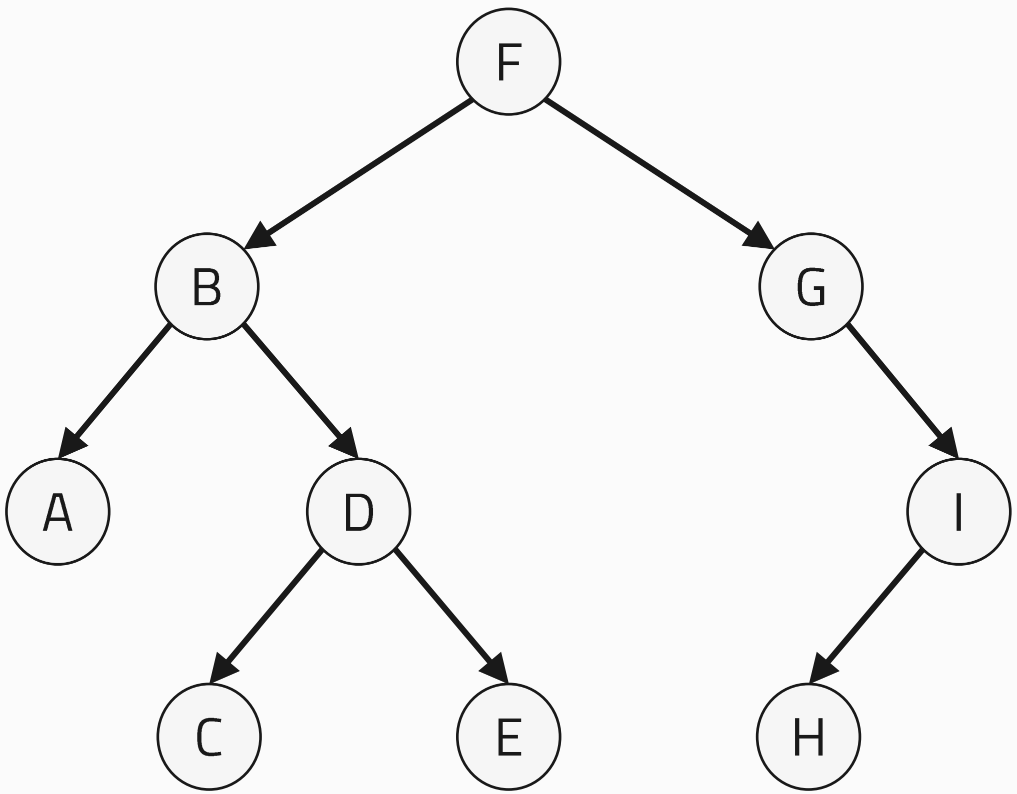
Ordre postfixe : on note un nœud dès qu'on passe à droite :

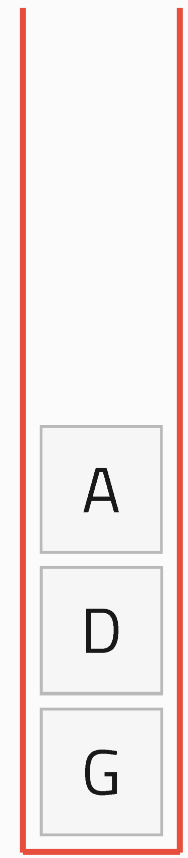
A C E D B H I G F

Parcours en profondeur d’un arbre

OU DFS (DEPTH-FIRST SEARCH)

Implémentation itérative: on utilise une *pile* pour mémoriser les nœuds à visiter…





Pile

Ordre de parcours préfixe :

F B A D C

E G I H

Ou bien on laisse l’ordinateur gérer la pile, en utilisant un algorithme *récursif* !

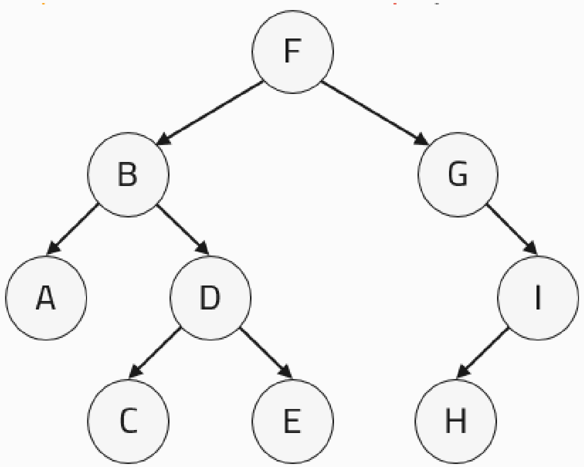
Arbre binaire

DEFINITIONS

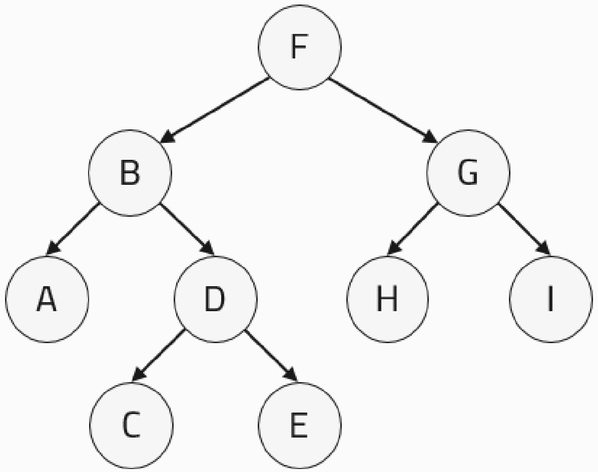
Arbre binaire : chaque nœud a au plus deux enfants

Arbre binaire strict : chaque nœud possède 0 ou 2 enfants

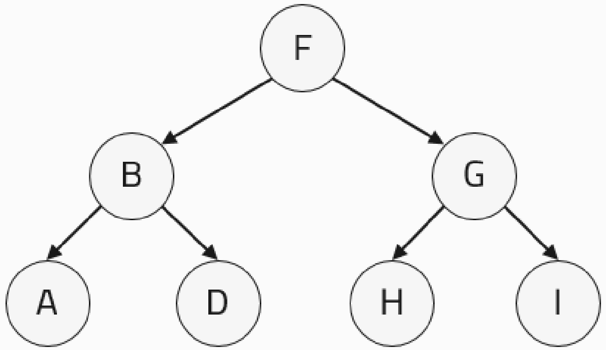
Arbre binaire parfait : arbre binaire strict où toutes les feuilles sont à la même profondeur



Arbre binaire non strict



Arbre binaire strict



Arbre binaire parfait

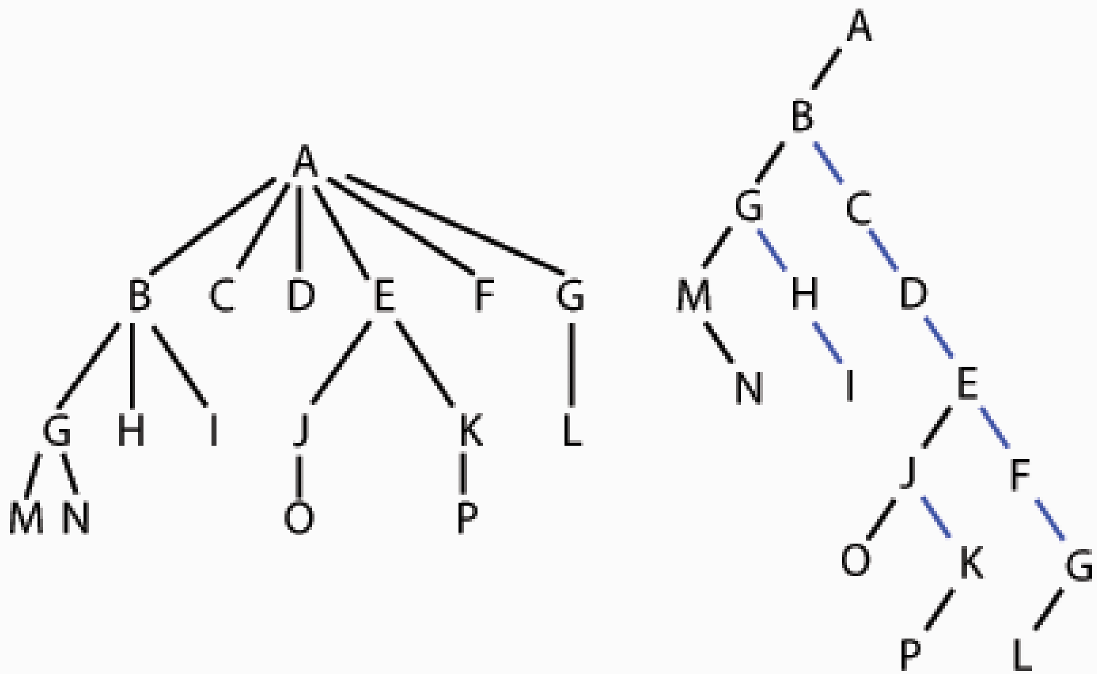
Arbre binaire

INTERÊTS

Propriété : n’importe quel arbre peut être transformé en arbre binaire

Principe :

- chaque nœud N de l’arbre de gauche est associé à un nœud N’ dans l’arbre de droite
- le fils gauche de N’ est le fis gauche de N
- le fils droit de N’ est le frère suivant de N



Principaux intérêts des arbres binaires :

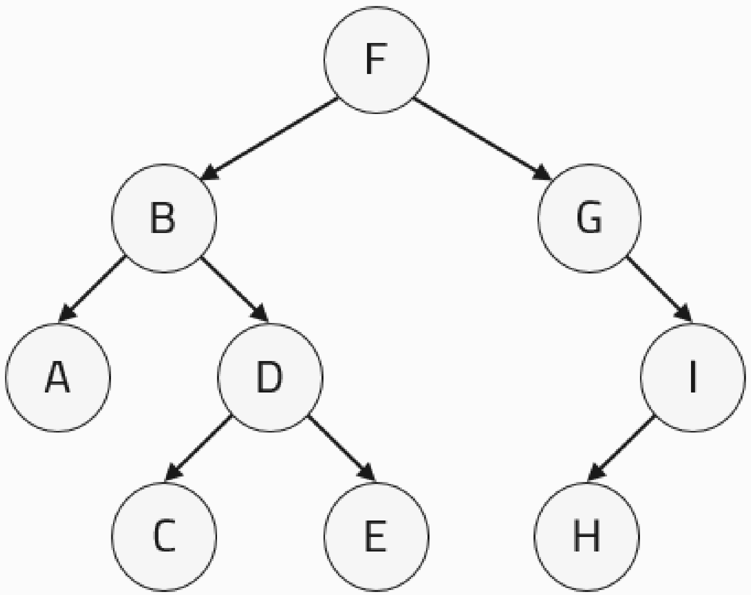
- modélisent les processus de décision « oui / non »
- utilisés en compression de données (Codage de Huffman)
- permettent d’étendre la recherche dichotomique à des structures de données non linéaires

Arbre binaire de recherche

DEFINITION

Arbre binaire dans lequel tout nœud interne possède :

- une valeur supérieure à son fils gauche
- une valeur inférieure à son fils droit



Intérêt : permet de rapidement (*O(log n) en moyenne*)

- insérer une valeur
- rechercher une valeur
- supprimer une valeur

Démo

Arbre binaire de recherche

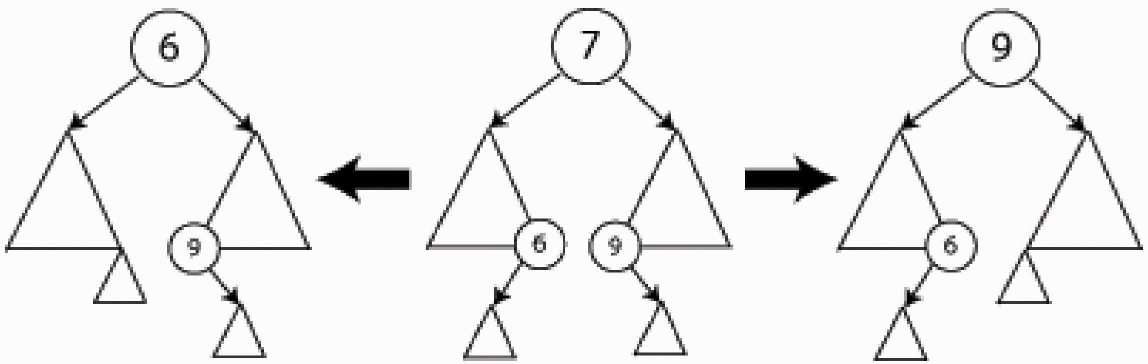
OPERATIONS ELEMENTAIRES

Insertion et recherche : simples à implémenter

Suppression : plusieurs cas à prendre en compte

- Suppression d'une feuille : il suffit de l'enlever de l'arbre
- Suppression d'un nœud avec un fils unique : on le remplace par son fils
- Suppression d'un nœud *N* avec deux enfants : deux possibilités :
  - on échange *N* avec son *successeur* le plus proche
    (le nœud le plus à gauche du sous-arbre droit)
  - ou on échange *N* avec son *prédécesseur* le plus proche
    (le nœud le plus à droite du sous-arbre gauche)
  - puis on recommence la suppression de *N*
    (qui a à présent 0 ou 1 seul nœud fils)

On conserve ainsi un arbre binaire de recherche

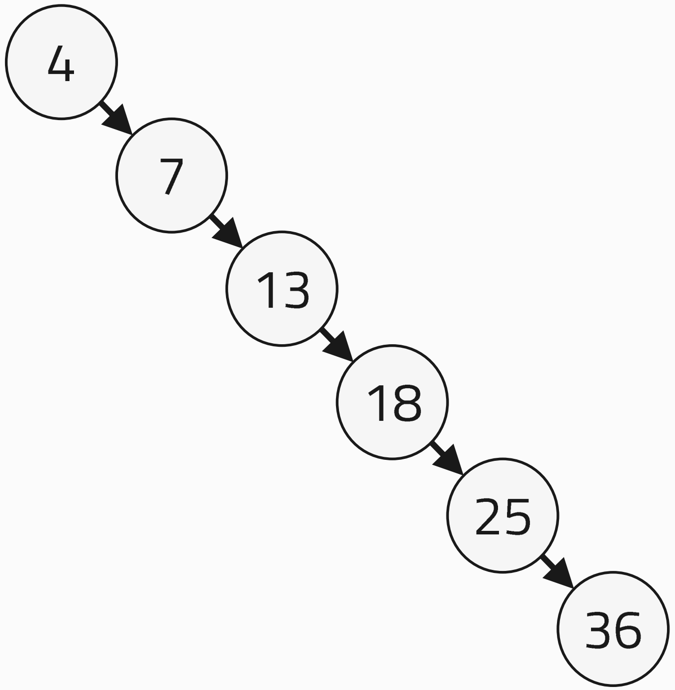


Arbre binaire de recherche

ARBRE DEGENERE

Cas pathologique : on insère des valeurs *triées* dans un arbre binaire de recherche

Exemple : 4, 7, 13, 18, 25, 36



On obtient une liste chaînée !

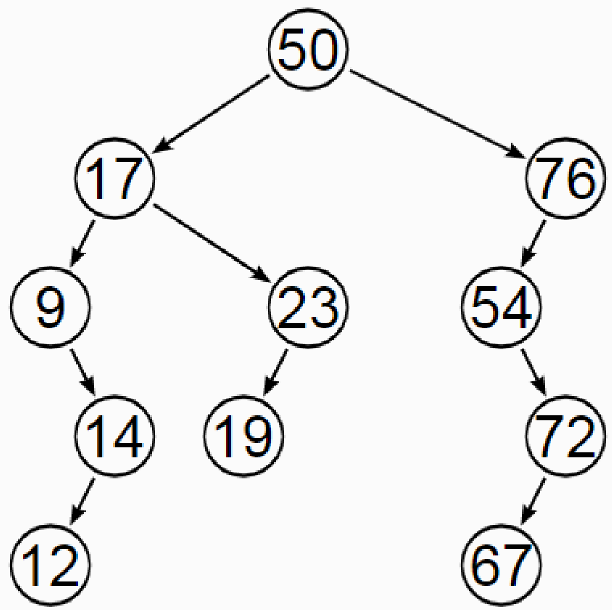
Les performances passent de O(log n) à O(n) !

Arbre binaire de recherche

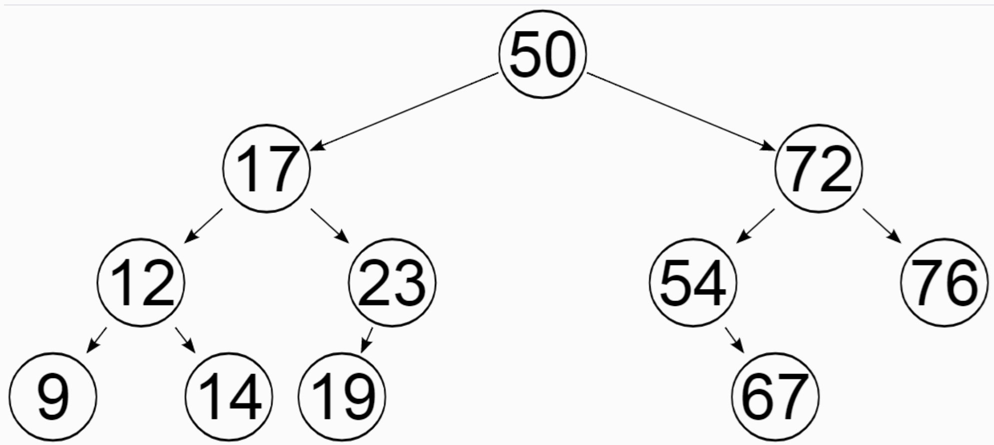
ARBRE EQUILIBRE

Solution : maintenir un arbre constamment équilibré

Arbre équilibré : pour **tout** nœud, la différence de hauteur de l'arbre gauche et de l'arbre droit est au plus 1



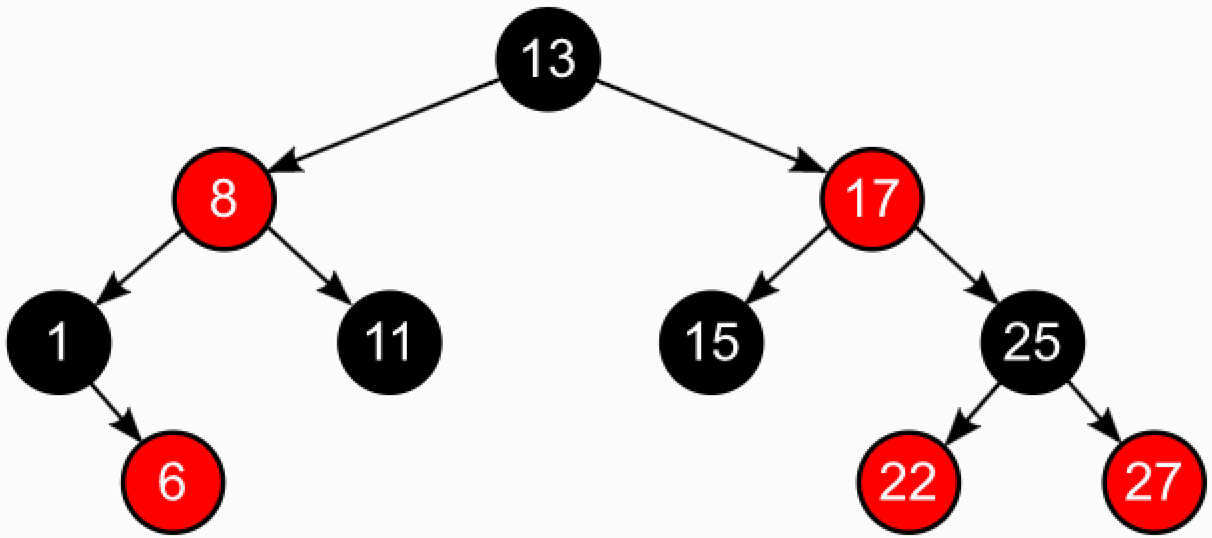
Arbre binaire de recherche déséquilibré
(le sous-arbre 9-14-12 est déséquilibré)



Le même arbre, après équilibrage

Exemples d'arbres binaires de recherche

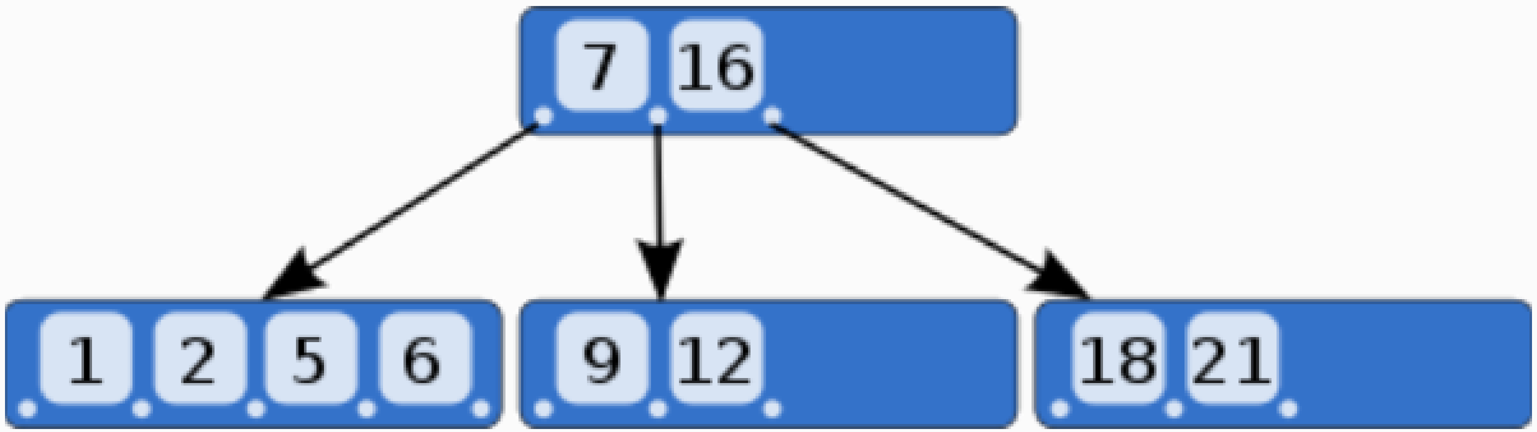
- Arbres AVL (1962) : historiquement les premiers arbres binaires de recherche équilibrés
- Arbres rouges-noir (1978) : chaque nœud possède une couleur ; tous les chemins de la racine à une feuille contiennent le même nombre de nœuds noirs



La complexité de toutes les opérations (recherche, insertion, suppression) reste *logarithmique*

Autres arbres de recherche équilibrés

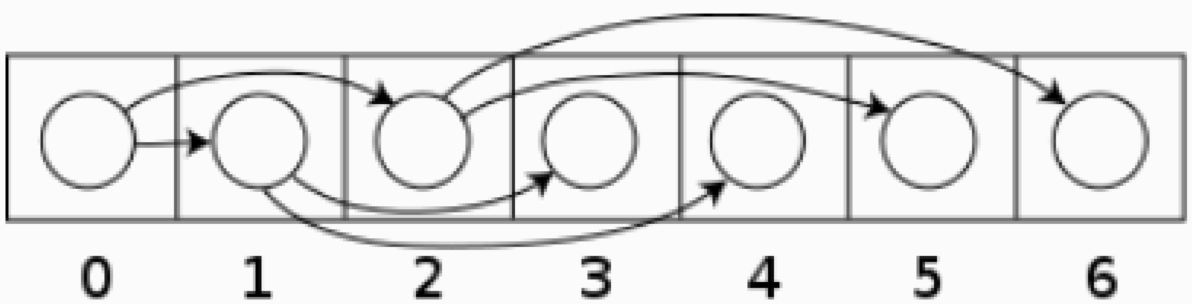
- B-arbres : généralisation des arbres binaires de recherche équilibrés (un B-arbre n’est pas nécessairement binaire), utilisés dans les bases de données (gestion des index) et les systèmes de fichiers (NTFS, btrfs, Ext4…)



Arbre binaire de recherche

STOCKAGE

Remarque : on peut stocker les arbres binaires de manière très compacte, avec un tableau :



Pour tout nœud en position k :

- son fils gauche se trouve en position $2k + 1$

- son fils droit se trouve en position $2k + 2$

- son père se trouve en position $\left\lfloor \frac{k-1}{2} \right\rfloor$