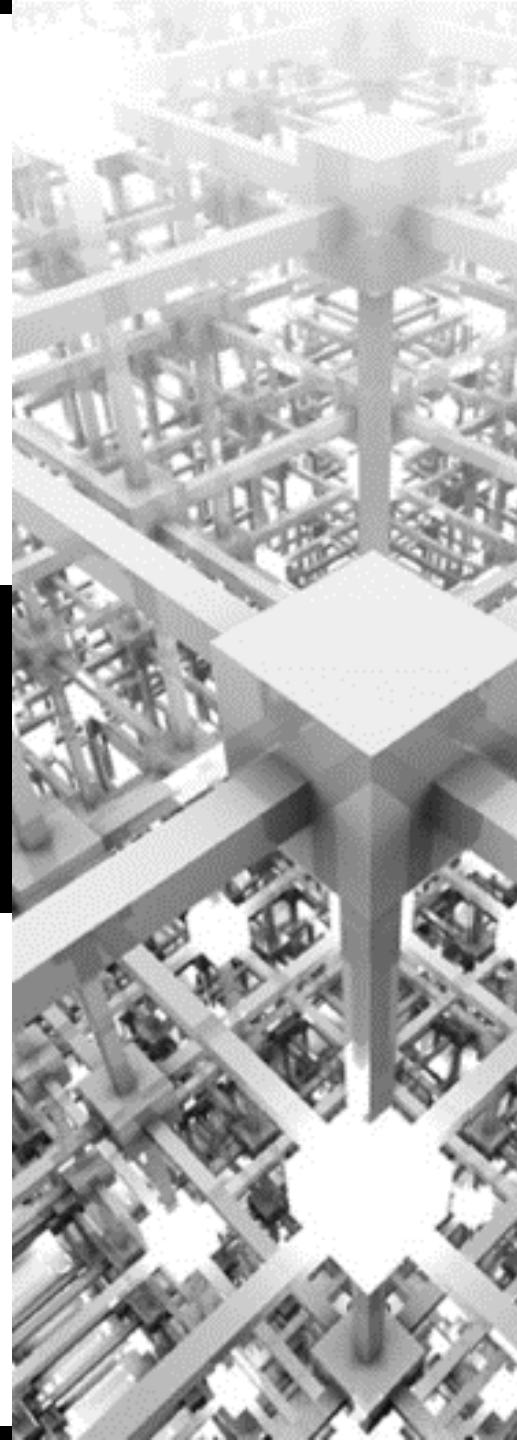
ENSAM

STRUCTURES DE DONNÉES EN C

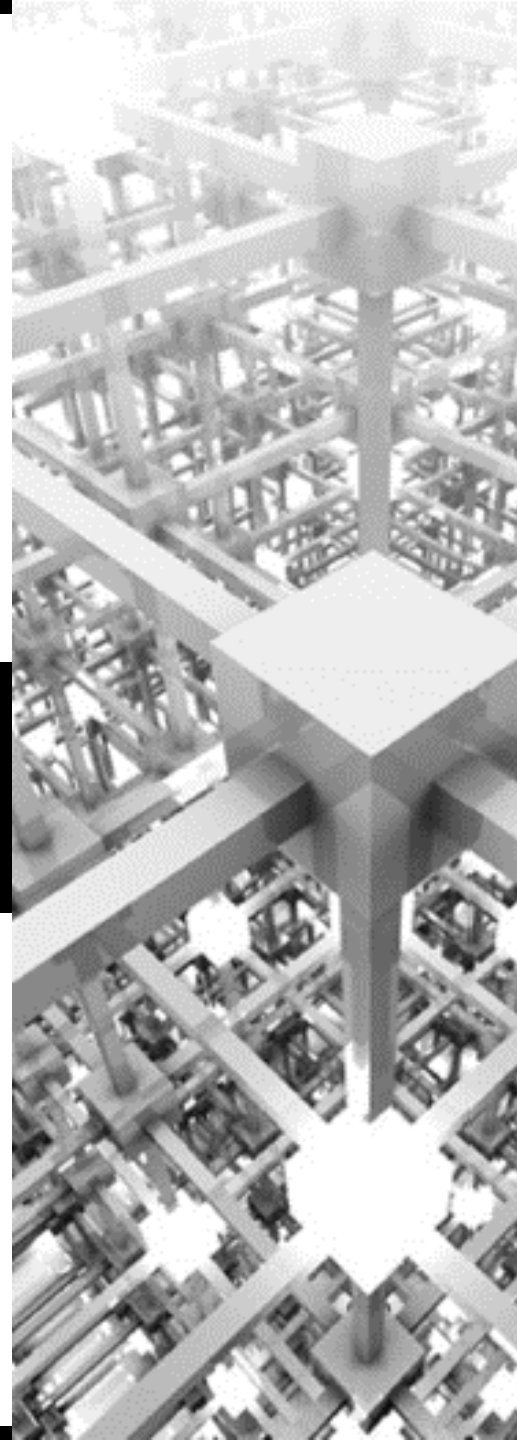
1ère Année «Cycle ingénieur : Intelligence Artificielle et Génie Informatique»

2023/2024

Dep. Informatique
Pf. CHERGUI Adil



UH2C | ENSAM



Séance 3

LES ARBRES

Objectifs de la séance :

- Le rôle et l'utilité des arbres
- Les différents types d'arbres
- L'implémentation des arbres binaires

Pf. Adil CHERGUI

INTRODUCTION

Définition

L'**arbre** est une structure de donnée qui généralise la liste chainée : alors qu'une cellule de liste a un seul successeur (la cellule suivante), dans un arbre il peut y en avoir plusieurs. On parle alors de **nœud** (au lieu de cellule).

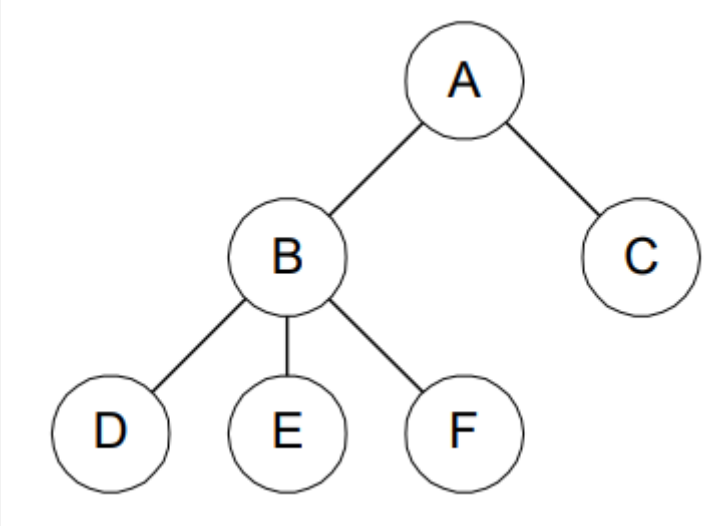
Un arbre est une **structure de données** composée d'un ensemble de **nœuds**.

Tous les arbres sont **hiérarchiques** de nature. Intuitivement, la **hiérarchie** signifie que dans un arbre existe une relation « **père-fils** » entre les nœuds.

Un **nœud père** peut avoir plusieurs **nœud fils**. Un fils n'a qu'un seul père, et tous les nœuds ont un ancêtre commun appelé la **racine de l'arbre** (le seul nœud qui n'a pas de père)

Chaque nœud contient l'information spécifique de l'application et des pointeurs vers d'autres nœuds (d'autres sous-arbres).

Exemple 1 :



Par exemple : B et C sont des fils de A et B est le père de D, E et F

INTRODUCTION

Définition

L'**arbre** est une structure de donnée qui généralise la **liste chainées** : alors qu'une cellule de liste a un seul successeur (la cellule suivante), dans un arbre il peut y en avoir plusieurs. On parle alors de **nœud** (au lieu de cellule).

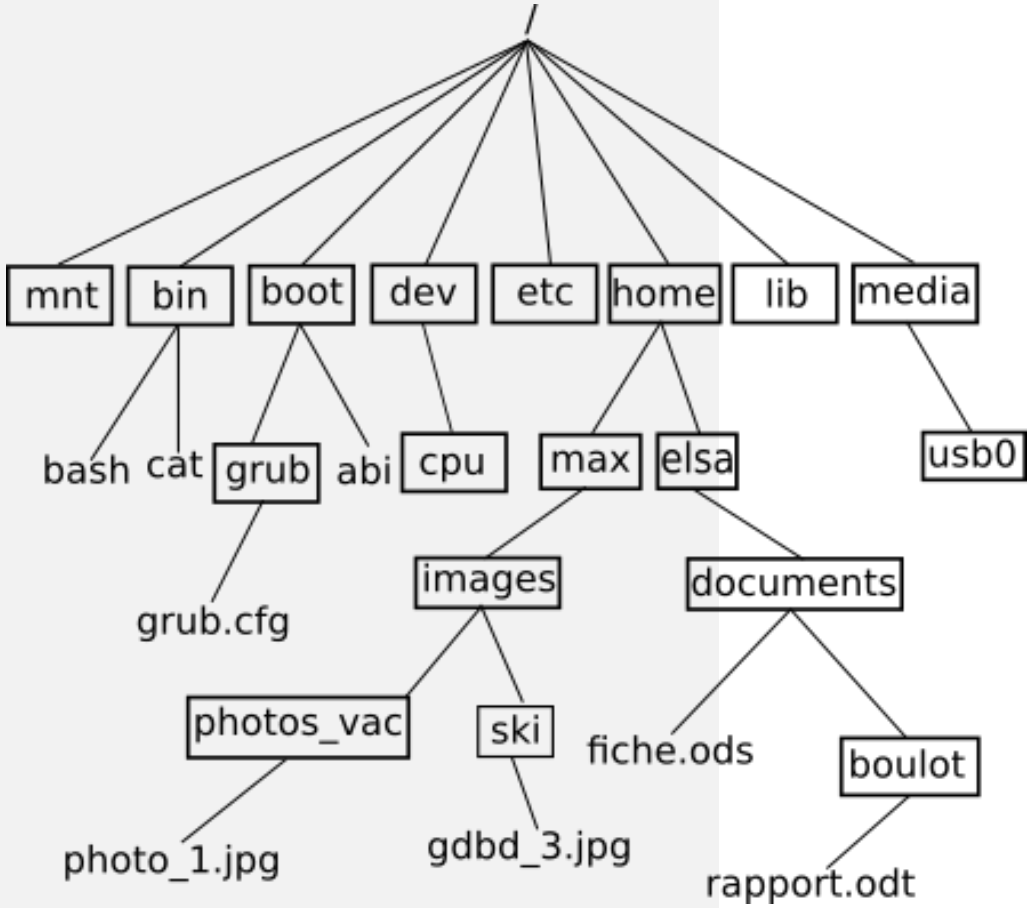
Un arbre est une **structure de données** composée d'un ensemble de **nœuds**.

Tous les arbres sont **hiérarchiques** de nature. Intuitivement, la **hiérarchie** signifie que dans un arbre existe une relation « **père-fils** » entre les nœuds.

Un **nœud père** peut avoir plusieurs **nœud fils**. Un fils n'a qu'un seul père, et tous les nœuds ont un ancêtre commun appelé la **racine de l'arbre** (le seul nœud qui n'a pas de père)

Chaque nœud contient l'information spécifique de l'application et des pointeurs vers d'autres nœuds (d'autres sous-arbres).

Exemple 2 :



L'organisation des fichiers dans les systèmes d 'exploitation

TERMINOLOGIES

Définition

Feuilles (ou nœuds externes) : les nœuds ne pointant vers aucun autre nœud sont appelés feuilles.

Nœuds internes : Ensemble des nœuds qui sont suivis par au moins un nœud.

Racine : Il existe un nœud au niveau 1 qui n'est pointé par aucun autre nœud : c'est la racine de l'arbre.

Niveau Chaque étage de l'arbre est appelé niveau et les niveaux sont numérotés de **1** pour la racine à **n** la feuille la plus basse. La hauteur d'un arbre c'est le niveau maximum atteint par une **feuille.**

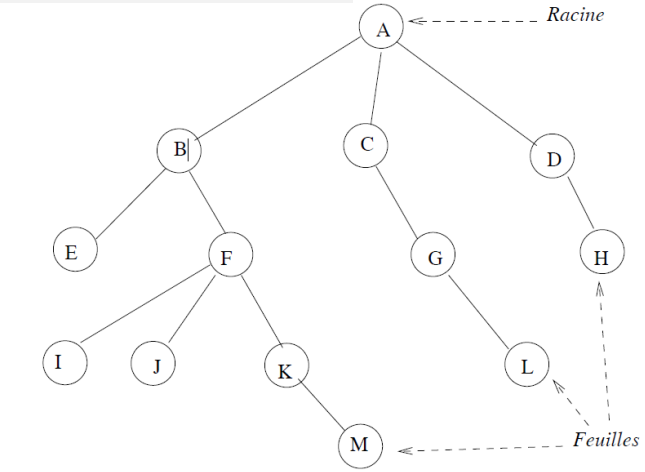
Branche : Un parcours de la **racine** vers une **feuille** est une Branche.

La hauteur (ou profondeur) d'un nœud est la longueur de la **branche qui le lie à la racine,** par exemple la hauteur du nœud K est **4** (exemple 1).

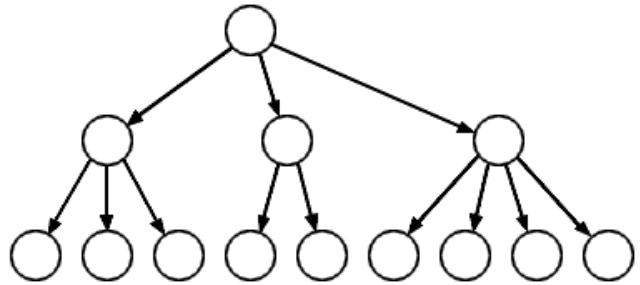
La hauteur d'un arbre est la longueur de la plus grande branche de cet arbre, dans (exemple 1). La hauteur de l'arbre est **5** .

Sous-arbre : un sous arbre d'un arbre est l'ensemble de n'importe quel nœud de l'arbre et tous ses descendants. Le sous-arbre du **nœud n** est le sous-arbre dont la racine est un fils de **n**.

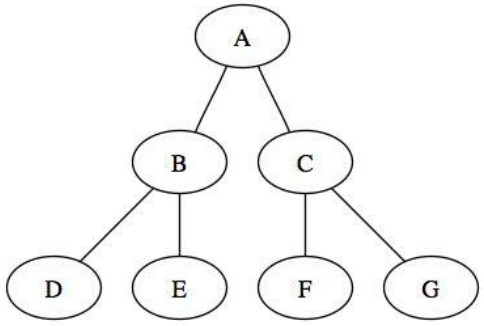
1



2



3



TERMINOLOGIES

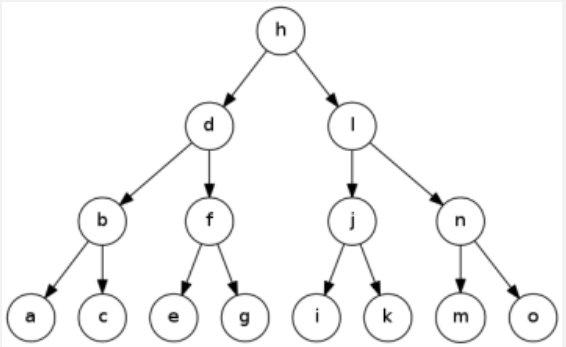
Définition

Degré d'un nœud : on appelle degré d'un nœud, le nombre de successeurs de ce nœud.

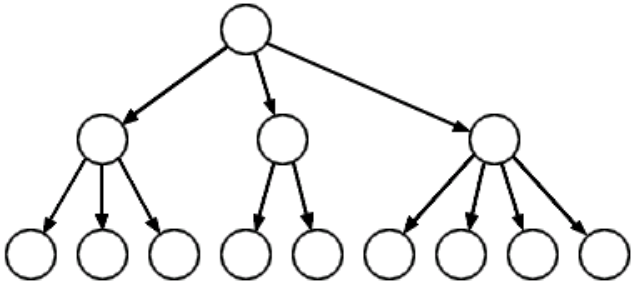
Degré d'un arbre : si N est le degré maximum des nœuds de l'arbre, l'arbre est dit n-aire. Sur l' (exemple 2), l'arbre est de degré 4. L'arbre est un **arbre 4-aire**.

Taille d'un arbre : c'est le nombre total de nœuds de l'arbre.

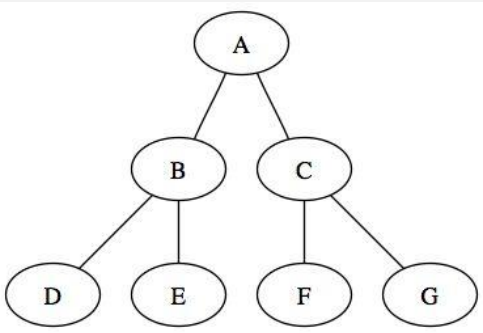
1



2



3



TERMINOLOGIES

Définition

Arbre ordonné : si l'ordre des sous-arbres est significatif, on dit que l'arbre est ordonné (arbre généalogique par exemple).

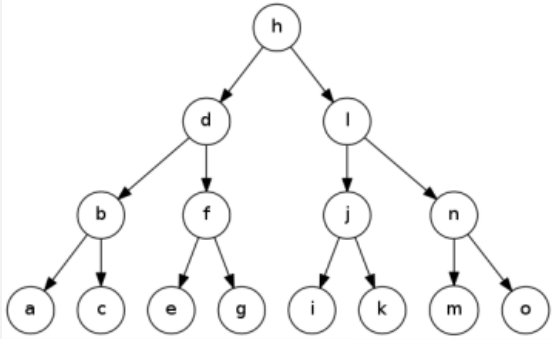
Arbre binaire : un arbre binaire est un type d'arbre ordonné tel que chaque nœud a aux plus deux fils et quand il n'y en a qu'un, on précise s'il s'agit du fils droit ou du fils gauche.

Arbre binaire parfaitement équilibré : un arbre binaire est parfaitement équilibré si pour chaque nœud, les nombres de nœuds des sous-arbres gauche et droit diffèrent au plus d'un.

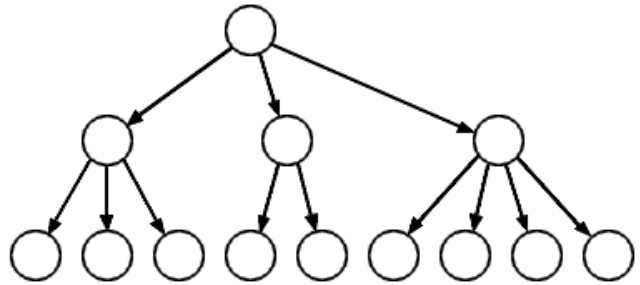
Arbre binaire équilibré : un arbre binaire est équilibré si pour chaque nœud, les hauteurs des sous-arbres gauche et droit diffèrent au plus d'un.

Arbre binaire complet : c'est un arbre strictement binaire ou toutes les feuilles ont le même niveau, c'est-à-dire de taille **2^k-1** (k étant le niveau des feuilles équivalent aussi à la hauteur de l'arbre) (exemple 3).

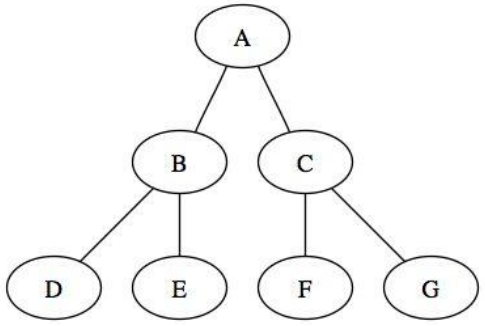
1



2



3



TERMINOLOGIES

Exemple :

– Le nœud A est la **racine** de l'arbre.

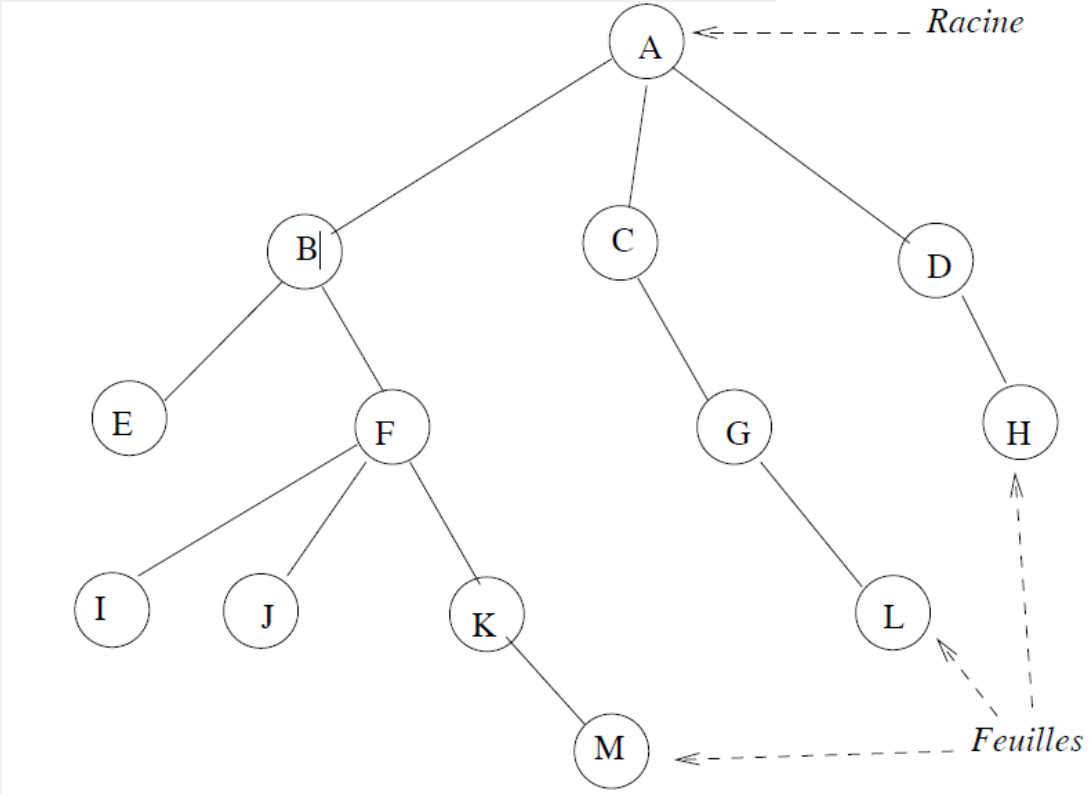
– Les nœuds E, I, J, M, L et H sont des **feuilles**.

– Les nœuds B, C, D, F, G et K sont des **nœuds intermédiaires**.

– Si une branche relie un nœud ***ni*** à un nœud ***nj*** situé plus bas, on dit que ***ni*** est un **ancêtre** de ***nj*** .

– Dans un arbre, un nœud n'a **qu'un seul père** (ancêtre direct).

– Un nœud peut contenir une ou plusieurs valeurs.



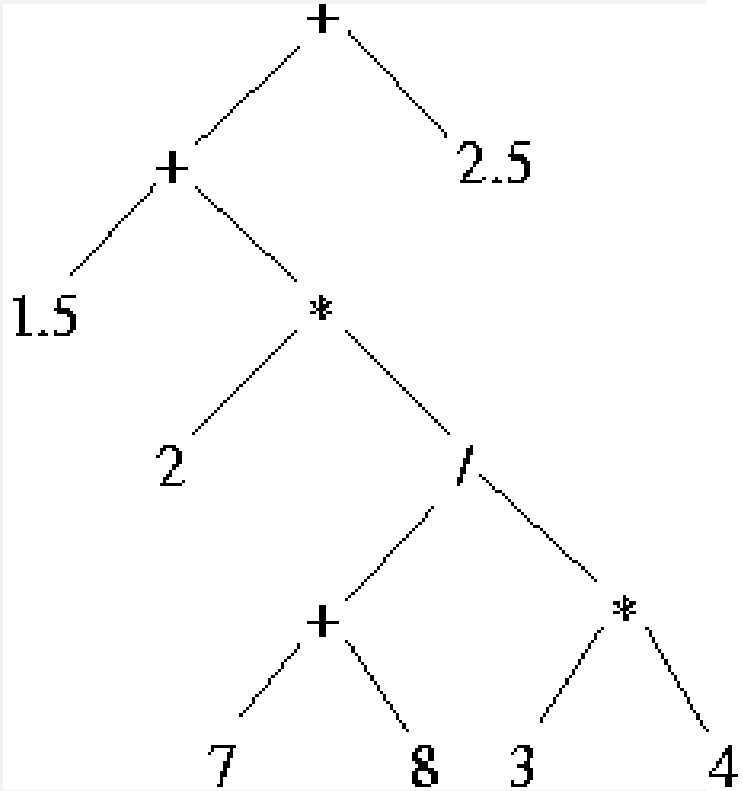
EXEMPLES D'APPLICATIONS UTILISANT LES ARBRES

Expression arithmétique : arbre binaire ordonné

Une expression arithmétique ayant des opérateurs binaires peut être schématisée sous la forme d'un arbre binaire. La Figure ci-contre représente l'expression arithmétique :

```
(1.5 + 2 * ((7+8) / (3*4))) + 2.5
```

L'arbre est ordonné : permuter 2 sous-arbres change l'expressions



EXEMPLES D'APPLICATIONS UTILISANT LES ARBRES

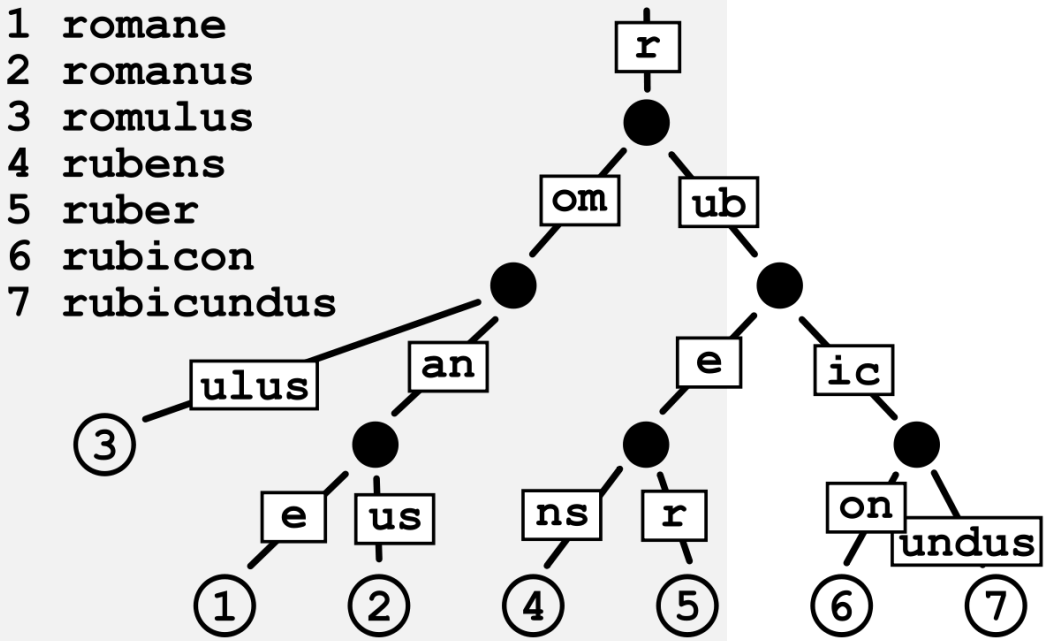
Représentation de caractères

Soient à représenter la liste des mots suivants : romane, romanus, romulus, rubens, ruber, rubicon, rubicundus.

Les débuts **communs** peuvent n'être mémorisés qu'une seule fois sous forme d'un arbre de caractères. L'arbre peut aussi se noter sous la forme équivalente suivante :

(r (om (ulus, an (e , us)), ub (e (ns , r) , ic (on , undus)))).

En fait, il faut ajouter un caractère '*' en fin de chaque mot, pour pouvoir distinguer les mots sous-chaînes d'un mot plus long.



un **arbre radix** ou **arbre PATRICIA**
(pour *Practical Algorithm To Retrieve Information Coded In Alphanumeric* en anglais et signifiant algorithme commode pour extraire de l'information codée en alphanumérique)

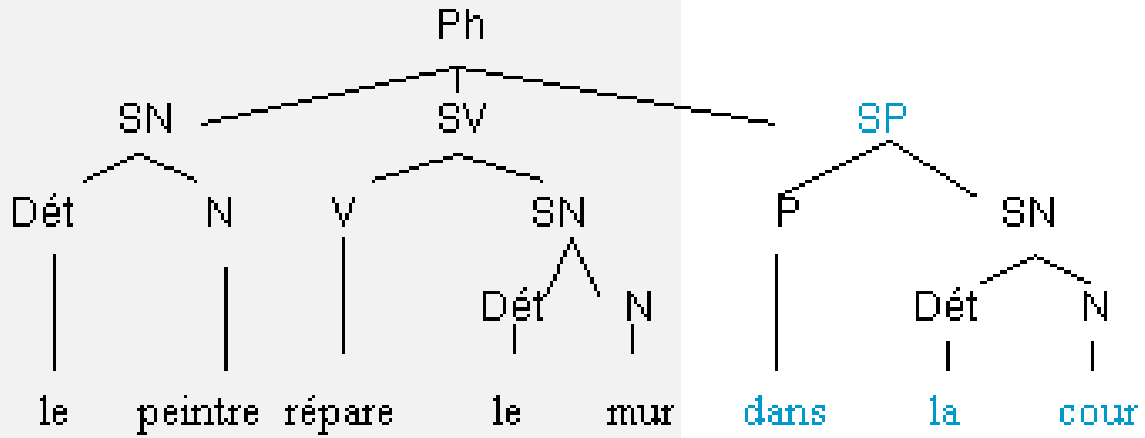
EXEMPLES D'APPLICATIONS UTILISANT LES ARBRES

Structure d'une phrase : arbre n-aire ordonné

Dans les traitements informatiques de la langue naturelle, on a souvent besoin de connaître la structure d'une phrase pour :

- **traduire** cette phrase d'une langue dans une autre langue.
- **prononcer** la phrase sur **synthétiseur** de parole (une bonne **intonation** nécessite une certaine connaissance de la structure de la phrase).
- **comprendre le sens** de la phrase et agir en fonction de la commande donnée en langage naturel.

Dans certaines applications, le langage peut être contraint, c'est-à-dire limité par le vocabulaire de l'application et par les constructions syntaxiques acceptées qui doivent être conformes à la grammaire de l'application. Par contre, en synthèse de parole, le synthétiseur doit être capable de prononcer n'importe quel mot, nom propre ou abréviation.



EXEMPLES D'APPLICATIONS UTILISANT LES ARBRES

D'autres exemple :

ou encore

• Utilisés dans l'implémentation des systèmes de bases de données, et dans l'organisation du système de fichiers d'un système d'exploitation ou dans n'importe quel système hiérarchique.

• l'interface graphique d'un logiciel est constituée de fenêtres et sous-fenêtres qui forment un arbre.

• la nomenclature d'un objet est un arbre : l'arbre des composants d'une voiture (moteur, carrosserie, sièges, etc.), de la structure de la Terre (continents, pays, etc.), du corps humain (tête, tronc, membres, etc.).

• la classification des objets.

LES ARBRES BINAIRES

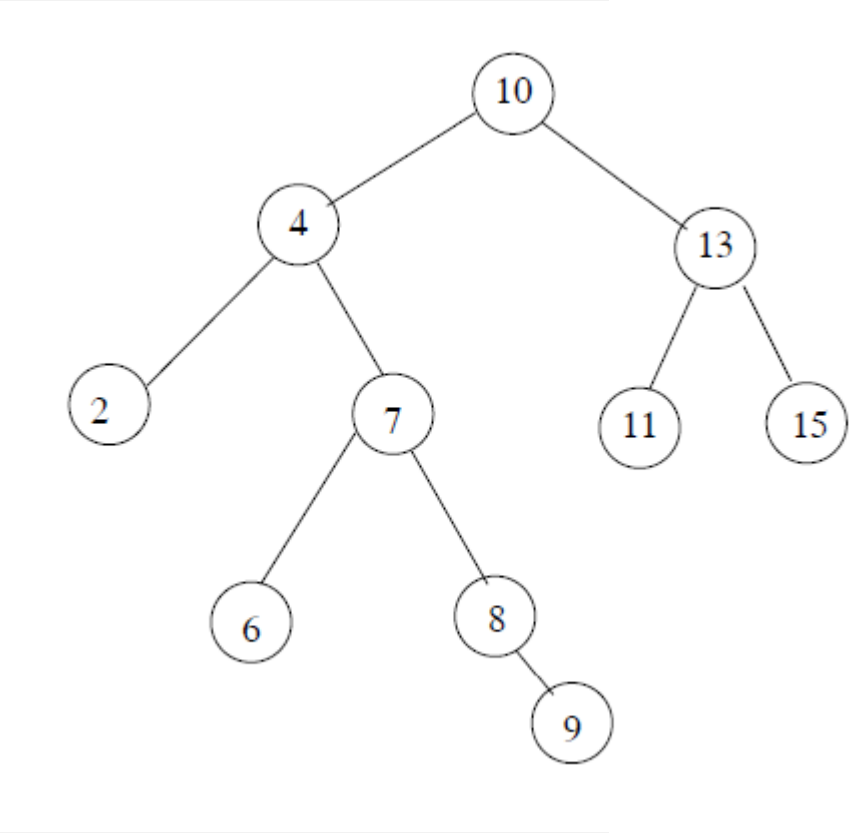
Arbres binaires de recherche

Un arbre binaire est un arbre dans lequel chaque nœud possède au plus deux fils. L'arbre des expression arithmétique présenté plus tôt en est un bon exemple contrairement à l'arbre de décomposition syntaxique (qu'on nomme arbre n-aire).

Un **Arbre Binaire de Recherche (abrégé ABR)** est un arbre binaire qui possède la propriété fondamentale suivante:

- tous les nœuds du sous-arbre de gauche d'un nœud de l'arbre ont une valeur inferieure ou égale à la sienne.
- tous les nœuds du sous-arbre de droite d'un nœud de l'arbre ont une valeur supérieure ou égale à la sienne.

Exemple: **figure ci-contre ➔**



LES ARBRES BINAIRES DE RECHERCHE

Recherche dans l'arbre

Un arbre binaire de recherche est fait pour faciliter la recherche d'informations. La recherche d'un nœud particulier de l'arbre peut être définie simplement de manière récursive:

Soit un sous-arbre de racine ***ni***,

– si la valeur recherchée est celle de la racine ***ni***, alors la recherche est terminée. On a trouvé le nœud recherché.

– sinon, si ***ni*** est une feuille (pas de fils) alors la recherche est infructueuse et l'algorithme se termine.

– si la valeur recherchée est plus grande que celle de la racine alors on explore le sous-arbre de droite c'est à dire qu'on remplace ***ni*** par son nœud fils de droite et que l'on relance la procédure de recherche à partir de cette nouvelle racine.

– de la même manière, si la valeur recherchée est plus petite que la valeur de ***ni***, on remplace ***ni*** par son nœud fils de gauche avant de relancer la procédure..

Si l'arbre **est équilibré,** chaque itération divise par 2 le nombre de nœuds candidats. La complexité est donc en ***O(log2 n***) si **n** est le nombre de nœuds de l'arbre.

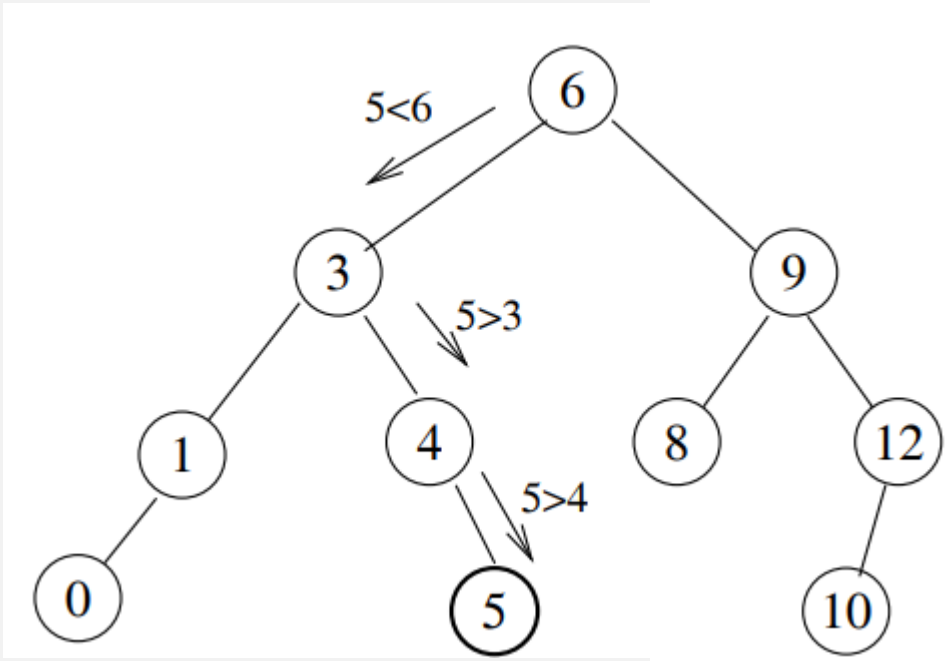
LES ARBRES BINAIRES DE RECHERCHE

Ajout d'un élément

Pour conserver les propriétés d'un arbre binaire de recherche nécessite de, l'ajout d'un nouvel élément ne peut pas se faire n'importe comment.

L'algorithme récursif d'ajout d'un élément peut s'exprimer ainsi:

- **soit x la valeur de l'élément à insérer.**
- **soit v la valeur du nœud racine *ni* d'un sous-arbre.**
  - **si ni n'existe pas, le créer avec la valeur x. fin.**
  - **sinon**
    - **si x est plus grand que v,**
      - **remplacer *ni* par son fils droit.**
      - **recommencer l'algorithme à partir de la nouvelle racine.**
    - **sinon**
      - **remplacer *ni* par son fils gauche.**
      - **recommencer l'algorithme à partir de la nouvelle racine**

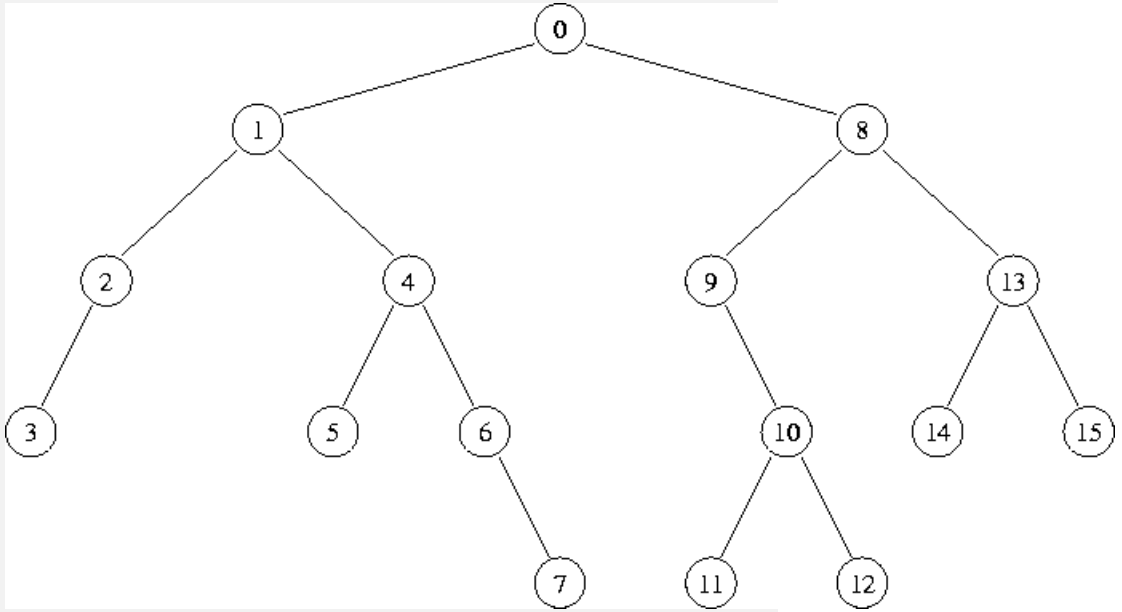


Ajout de la valeur 5 dans l'arbre

PARCOURS D'ARBRE BINAIRE

Les types de parcours

Un parcours d'arbre est une façon d'ordonner les nœuds d'un arbre afin de les parcourir. On peut le voir comme une fonction qui à un arbre associe une liste de ses nœuds même si la liste n'est souvent pas explicitement construite par le parcours.



On distingue essentiellement deux types de parcours : le **parcours en largeur** et les **parcours en profondeur**.

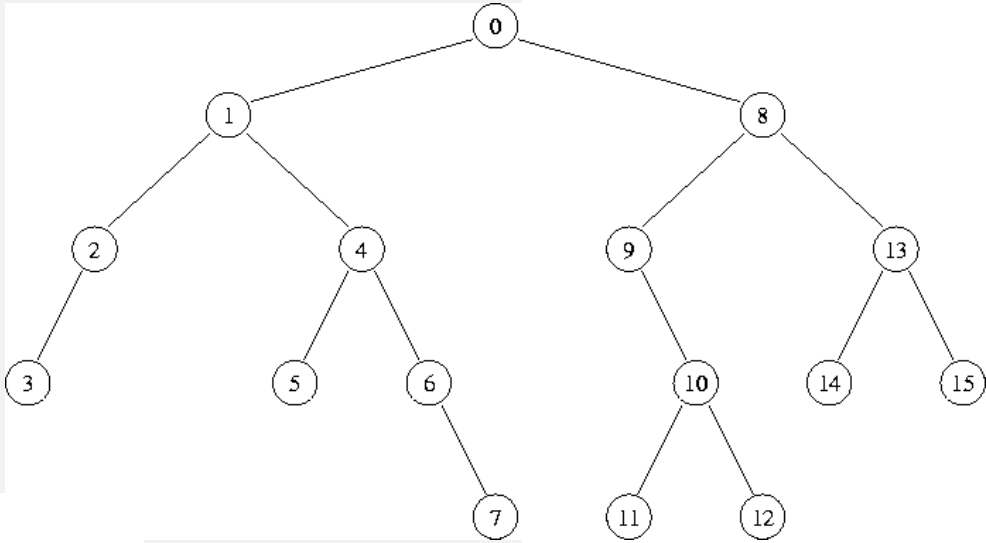
Parmi les parcours en profondeur, on distingue à nouveau le parcours **préfixe**, le parcours **infixe** et le parcours **suffixe.**

Les **parcours en profondeur** se définissent de manière récursive sur les arbres. Le parcours d'un arbre consiste à traiter la racine de l'arbre et à parcourir récursivement les sous-arbres gauche et droit de la racine. Les parcours préfixe, infixe et suffixe se distinguent par l'ordre dans lequel sont faits ces traitements

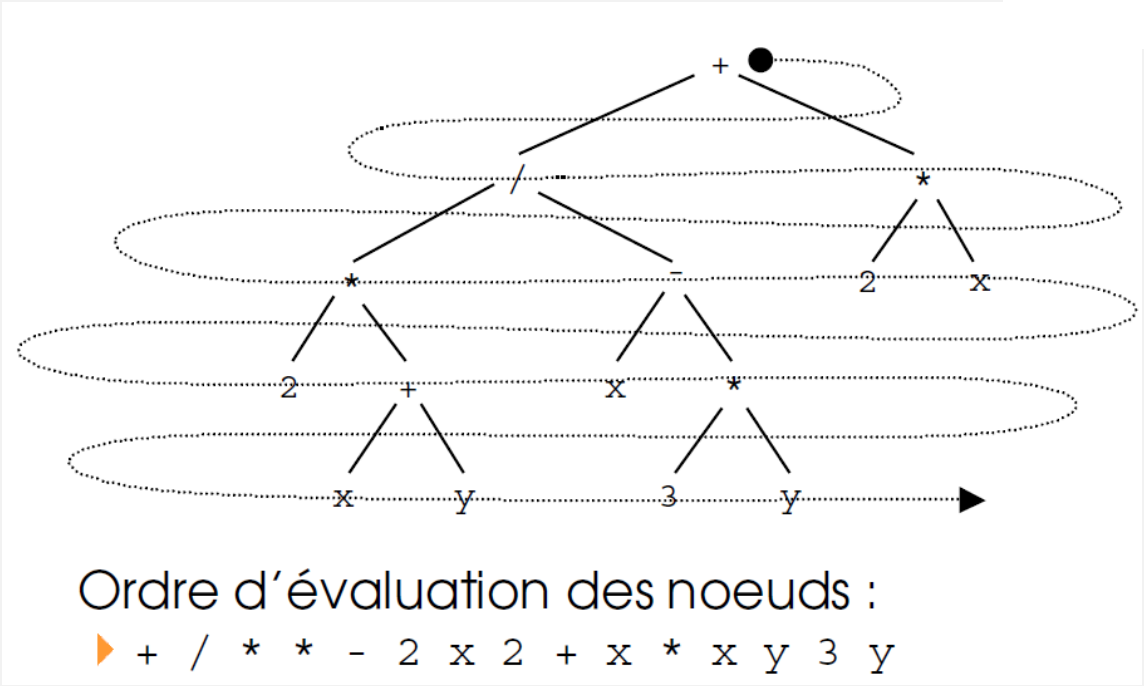
PARCOURS D'ARBRE BINAIRE

Parcours en largeur

Le **parcours en largeur** consiste à parcourir l'arbre niveau par niveau. Les nœuds de niveau 0 sont d'abord parcourus puis les nœuds de niveau 1 et ainsi de suite. Dans chaque niveau, les nœuds sont parcourus de la gauche vers la droite. Le parcours en largeur de l'arbre ci-dessus parcours les nœuds dans l'ordre [0,1,8,2,4,9,13,3,5,6,10,14,15,7,11,12].



Un autre exemple



PARCOURS D'ARBRE BINAIRE

Parcours en largeur algorithme

L'algorithme est identique à la version itérative du parcours préfixé mais avec l'utilisation d'une file. La file d'attente contient au départ la racine. L'élément en tête de file est extrait et ses fils sont ajoutés à la file jusqu'à ce que la file soit vide.

Il n'est pas nécessaire de développer tout un module de gestion de file. Nous connaissons la taille maximum de la file, c'est la taille de l'arbre, il suffit d'allouer un tableau de pointeurs de nœud et de gérer l'indice de tête pour les sorties et l'indice de queue pour les entrées

Le principe ici est de remonter un tableau qui contient tous les nœuds de l'arbre rangés par niveau dans le sens d'un parcours en largeur. Le premier point est d'avoir la taille de l'arbre. Ensuite nous avons besoin d'allouer un tableau t_noeud ou de pointeurs t_noeud*, nous avons opté **pour un tableau de pointeurs**. Le premier nœud à l'indice 0 est le nœud racine de l'arbre. Ensuite sont rentrés s'ils existent les fils gauche et droite de chaque nœud qui entrent dans le tableau.

PARCOURS D'ARBRE BINAIRE

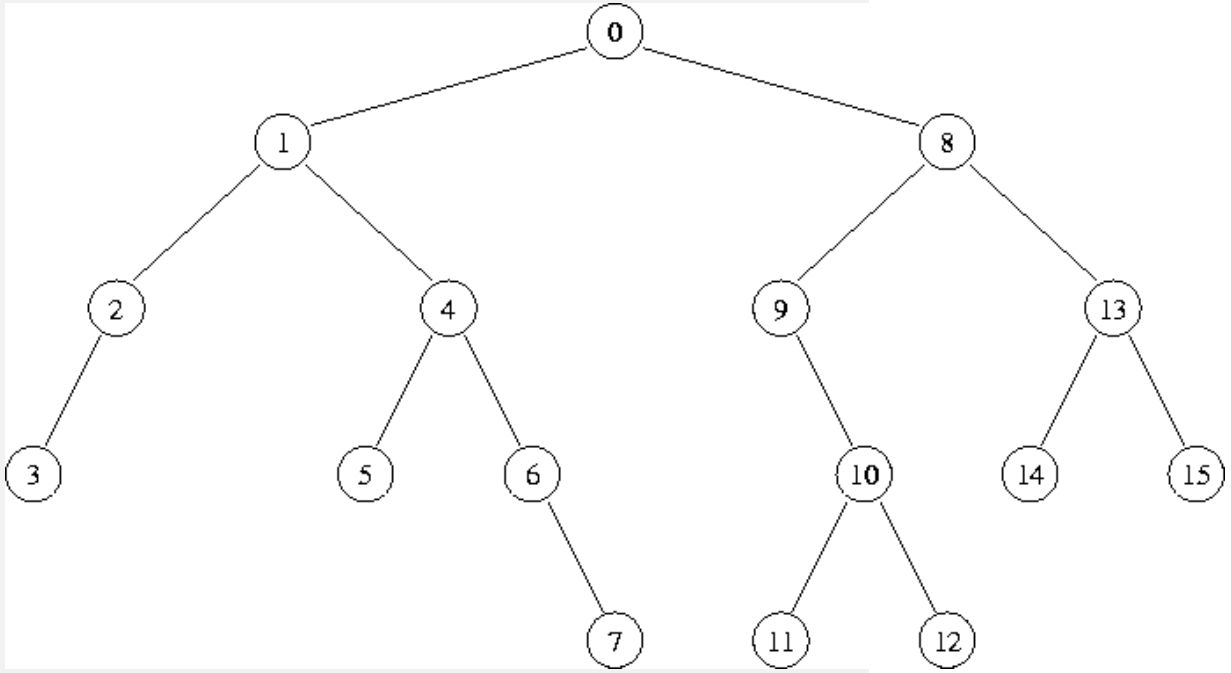
Parcours en profondeur

.

- Dans le parcours **préfixe**, la racine est traitée avant les appels récursifs sur les sous-arbres gauche et droit (faits dans cet ordre). Le parcours préfixe de l'arbre ci-dessus parcourt les nœuds dans l'ordre [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15].

- Dans le parcours **infixe**, le traitement de la racine est fait entre les appels sur les sous-arbres gauche et droit. Le parcours infixe de l'arbre ci-dessus parcourt les nœuds dans l'ordre [3,2,1,5,4,6,7,0,9,11,10,12,8,14,13,15].

- Dans le parcours **suffixe**, la racine est traitée après les appels récursifs sur les sous-arbres gauche et droit (faits dans cet ordre). Le parcours suffixe de l'arbre ci-dessus parcourt les nœuds dans l'ordre [3,2,5,7,6,4,1,11,12,10,9,14,15,13,8,0].

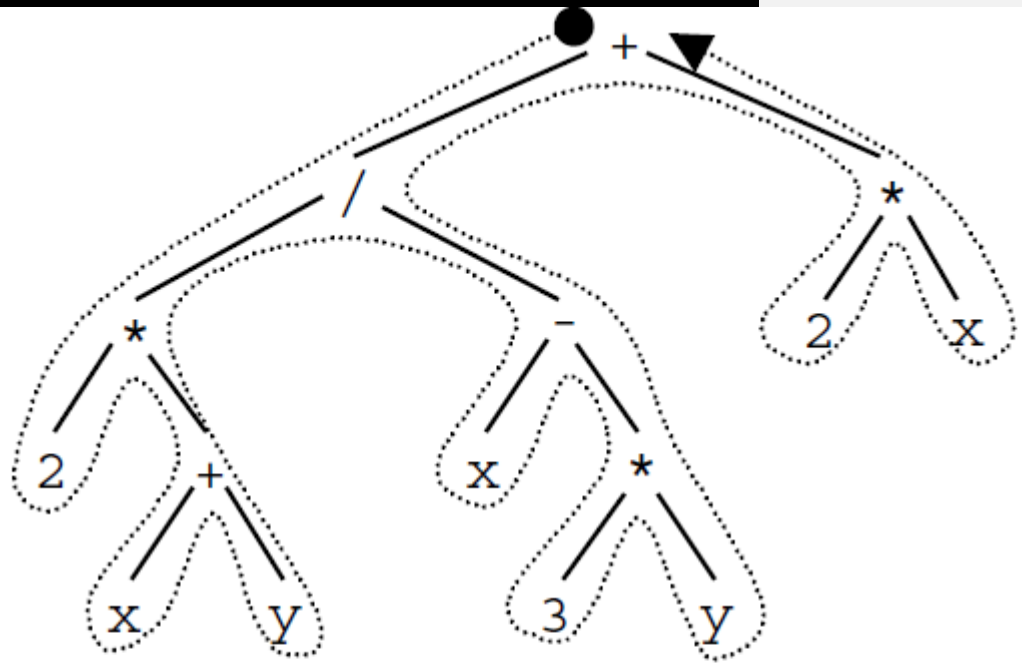


PARCOURS D'ARBRE BINAIRE

Parcours en profondeur

.

- Exemple 2 :



Parcours infixé

- 2 * x + y / x – 3 * y + 2 * x

Parcours préfixé :

- + / * 2 + x y – x * 3 y * 2 x

Parcours postfixé

- 2 x y + * x 3 y * - / 2 x * +

INSERTION DANS UN ARBRE

Respect des principes de ABR

La fonction d'insertion qui permet d'ajouter un élément dans l'arbre et donc de le créer de manière à ce qu'il respecte les propriétés d'un arbre binaire de recherche peut s'écrire ainsi:

On peut noter par ailleurs que l'arbre qui sera construit dépendra de l'ordre dans lequel seront insérées les différentes valeurs. En particulier, si les valeurs sont insérées dans un ordre croissant on obtiendra un arbre **complètement déséquilibré** (**ou dégénérer**), chaque nœud n'ayant qu'un fils droit (gauche si les valeurs sont dans un ordre décroissant).