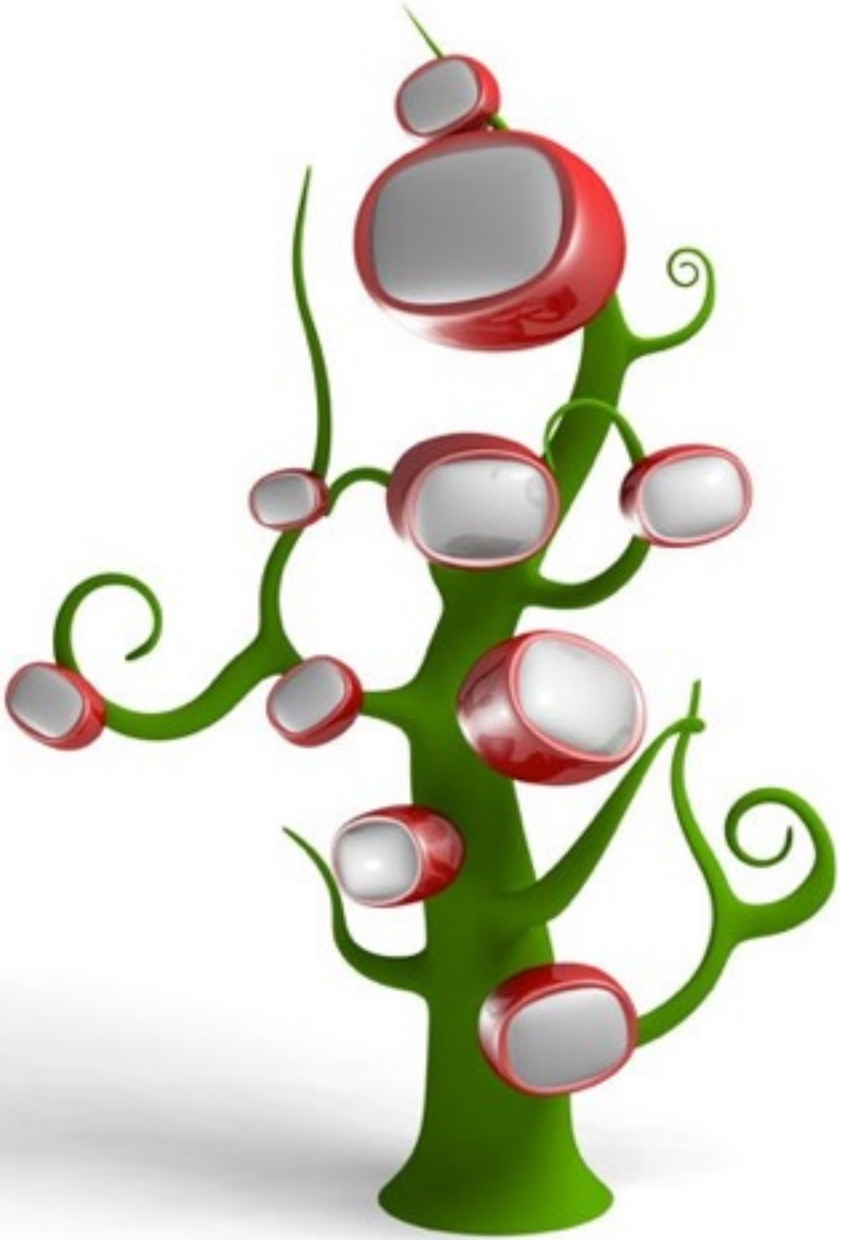


Chapitre 4

Les Arbres

I. Introduction

L’arbre est une structure de données fondamentale en informatique.

On rencontre cette structure dans tous les domaines de l’informatique.

Exple:

I. Algorithmique (les méthodes de Tri, …),

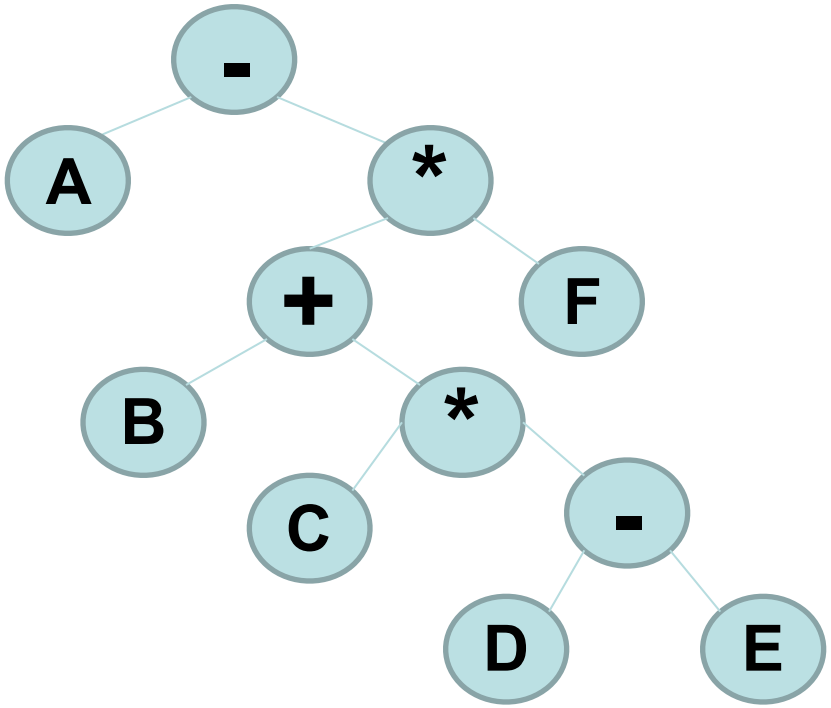
II. Compilation (les arbres syntaxiques, …),

III. Intelligence Artificielle (Arbre de décision,…)

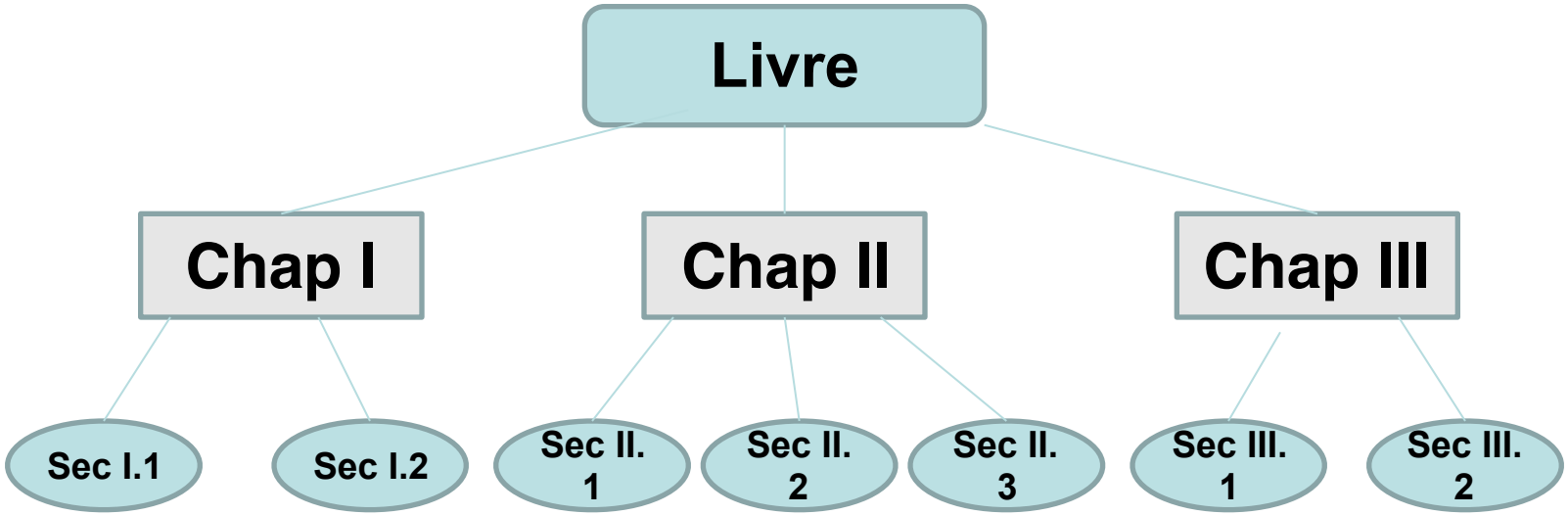
IV. Etc….

Exemple 1: (Représentation d'une expression arithmétique)

A – (B + C * (D – E)) * F



Exemple 2: (Représentation d'une table de matière d'un livre)



II.Terminologie de Base

Arbre, Sous-arbre et Nœud

➢ ***Arbre*** : un *arbre* est une collection d'éléments homogènes organisés en niveaux.

➢ ***Nœud*** : Chaque élément appelé *nœud* d'un niveau donnée peut être relié à plusieurs éléments du niveau inférieur par des branches ne formant pas de boucles (sinon on a affaire à un graphe).

➢ Un arbre peut, aussi être défini, comme étant un graphe orienté acyclique.

➢ on peut définir simplement un arbre de façon récursive tel qu'un arbre d'éléments de type T est soit:

- ✓ Vide
- ✓ Formée d'une donnée de type T appelée « *Racine* »et d'un ensemble de taille variable d'arbre de type T appelés « *sous-arbres.* »

➢ On appelle « *feuille* » la racine d'un arbre n'ayant pas lui même de sous-arbres.

➢ La racine et les feuilles sont des « *nœuds* » particuliers.

➢ Un sous-arbre d'un arbre et appelé son « *fils* », et inversement celui-ci sera appelé « *père* »de celui là.

➢ Deux sous-arbres du même arbre sont des « *frères* ».

Exemple

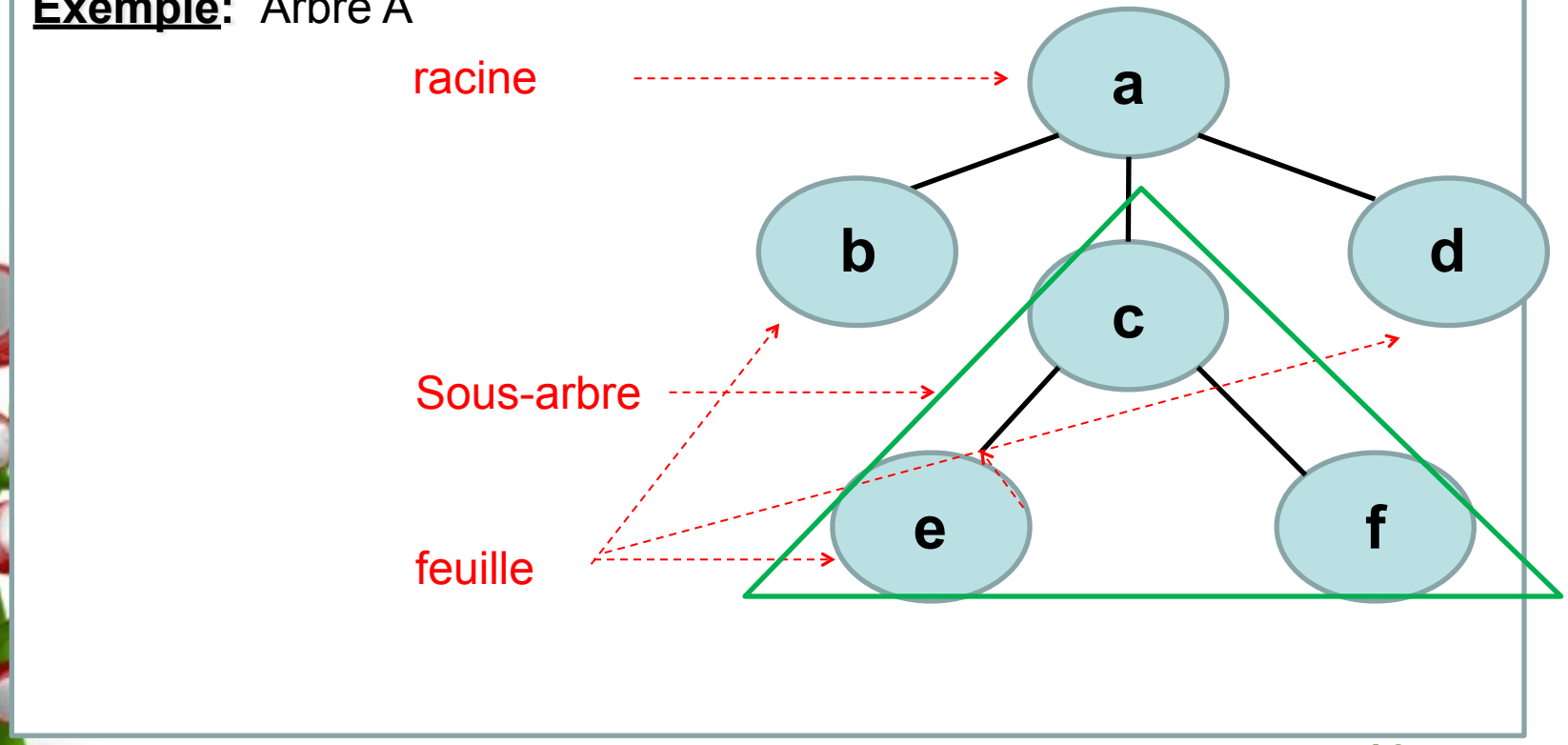
a : racine de A

b; e; f; d :feuilles de A;

c est le père de e et de f

d fils de a.

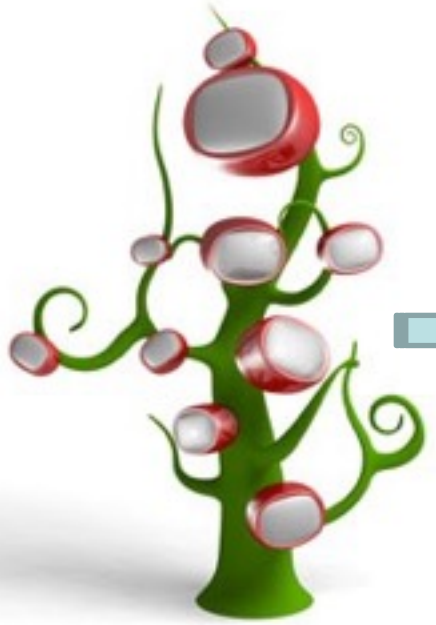
Exemple: Arbre A



Chemin

Soit $n_1,n_2,\ldots,n_k$ une suite de nœuds d'un arbre telle que n_i est le parent de n_{i+1} pour $1 \leq i < k$.

Cette suite est appelée « *chemin* » entre le nœud n_1 et le nœud n_k.



La longueur d'un chemin est égale au nombre d'arcs: *le nombre de nœuds qu'il contient* **-** **1** .

exemple: Dans l'arbre **A** le chemin *a-c-f* est de longueur 2

Descendant et ascendant

S'il existe un chemin entre 2 nœuds x et y, on dit que x est un «*ascendant*» ou un «*ancêtre* »de y et réciproquement que y est un «*descendant* » de x.

Exemple:

Dans l'arbre **A**

- les ascendants de c sont : c et a
- et ses descendants sont: c, e et f.

Remarque: Tout nœud est à la fois l'un de ses propres descendants et l'un de ses propres ascendants.

Profondeur:

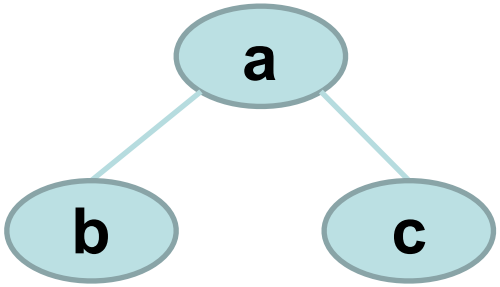
La « *profondeur* » d'un nœud est la longueur du chemin entre la racine et ce nœud.

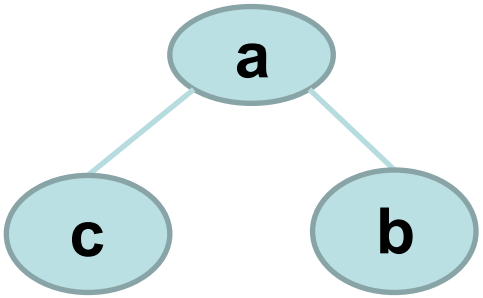
exemple: La profondeur du nœud d de l'arbre **A**, est égale à 1

II. 1 Ordre sur les nœuds d'un arbre

Les fils d'un nœud sont habituellement ordonnés de *gauche* vers *la droite*.

Exemple: Les deux arbres ordonnés suivants sont différents :





Si l'on désire ignorer explicitement l'ordre des fils, on dit que l'on travaille sur un arbre non ordonné.

L'ordre « *gauche-droite* » permettant la comparaison d'enfants d'un même père peut être étendu pour comparer deux nœuds qui ne sont pas liés par la relation « *ascendant-descendant* ».

Si a et b sont 2 frères et si a est à gauche de b, alors tout descendant de a est à gauche de tout descendant de b.

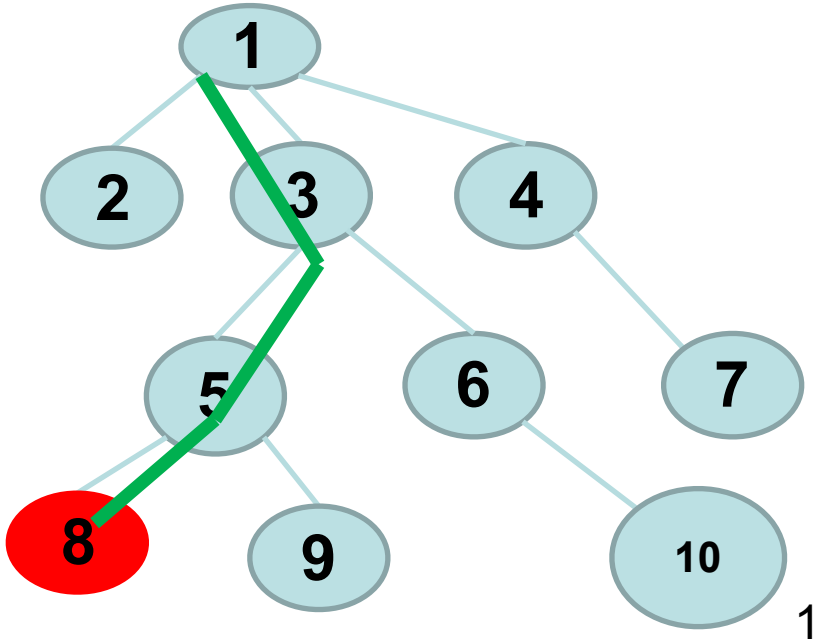
Exemple

D’après la figure suivante le nœud 8 est :

– à gauche des nœuds 9,6,10,4,7

– à droite du nœud 2 ;

– Il n’est ni à droite ni à gauche de ses ascendants 1, 3, 5.



Une règle simple pour trouver tous le nœuds situés à droite ou à gauche d'un nœud **n** est de tracer le chemin reliant la racine à ce dernier. Toute les branches de l'arbre partant sur la gauche de ce chemin aboutissent aux nœuds situés à gauche de n et réciproquement.

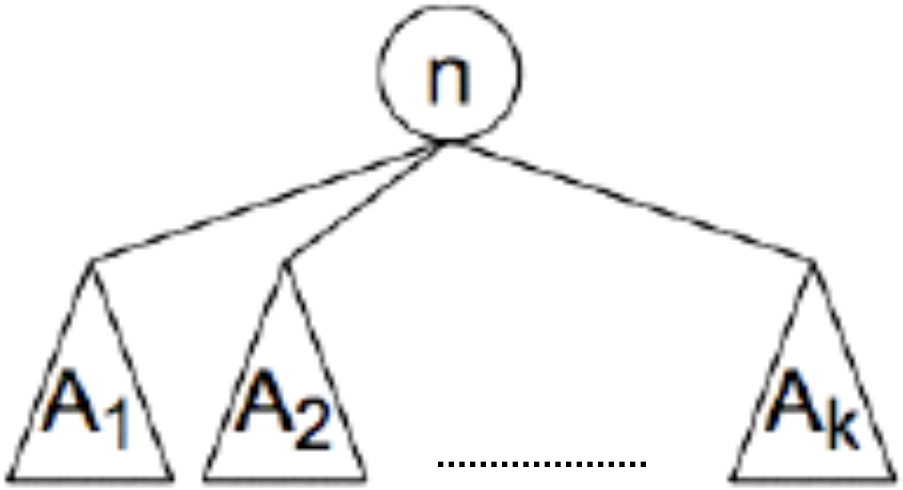
II. 2 Parcours d'arbre

Il existe plusieurs manières de parcourir les nœuds d'un arbre. Les trois parcours les plus importants sont les parcours:

- ✓ « *préfixé* » (ou pré-ordre),
- ✓ « *post-fixé* » (ou post-ordre)
- ✓ « *infixé* » (ou ordre)

Un parcours d'un arbre est tout algorithme permettant d'accéder <u>une et une seule fois</u> à tout les nœuds de l'arbre.

Soit **A** un arbre de racine **n** et $\mathbf{A_1}, \mathbf{A_2}, \ldots, \mathbf{A_k}$ ***k*** sous-arbres de **A**



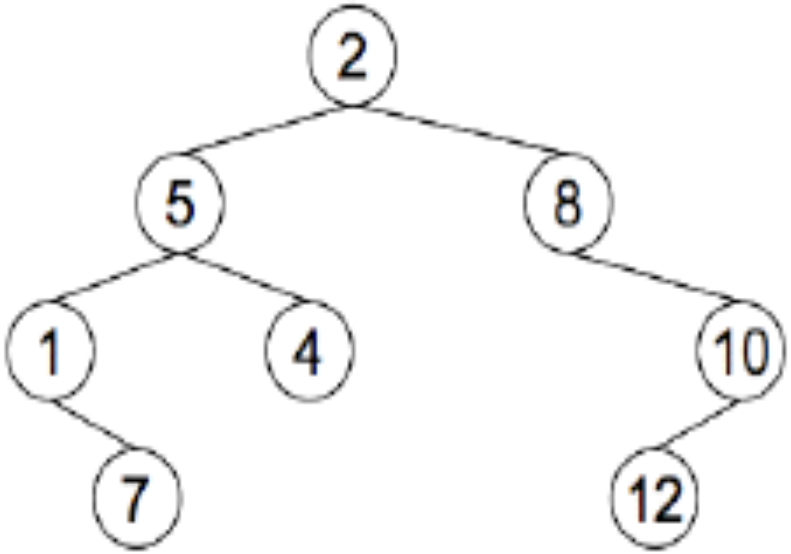
arbre A : *arbre K-aire*

Parcours préfixé

1. Pour effectuer le parcours *préfixé* des nœuds de **A**, on commence par la racine **n** puis on parcourt en préfixé les nœuds de A_1, puis ceux de A_2 . . .ainsi de suite jusqu'aux nœuds de A_k.

exemple (parcours préfixé)

L'arbre T



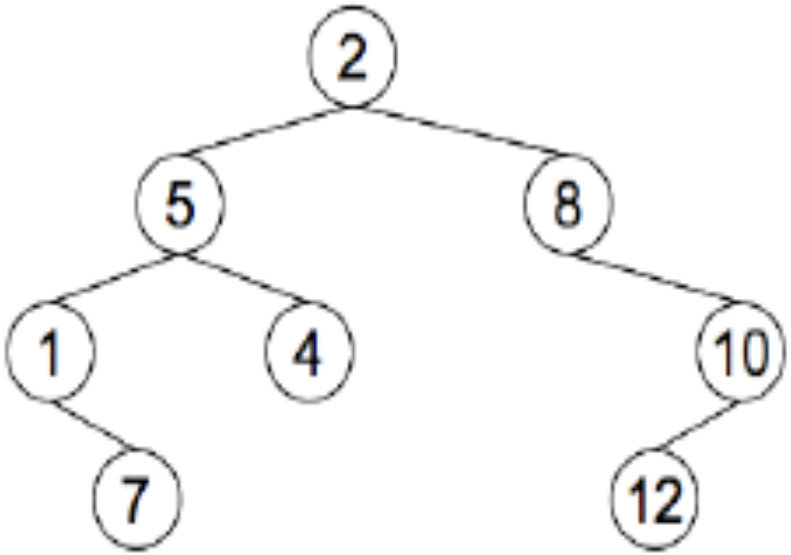
Le parcours préfixé de T : 2, 5, 1, 7, 4, 8, 10, 12.

Parcours postfixé

2. Pour effectuer le parcours *post-fixé* des nœuds de **A**, on parcourt en post-fixé les nœuds de A_1, puis ceux de A_2 jusqu'à A_k, puis on termine avec la racine **n**.

exemple (parcours post-fixé)

L'arbre T



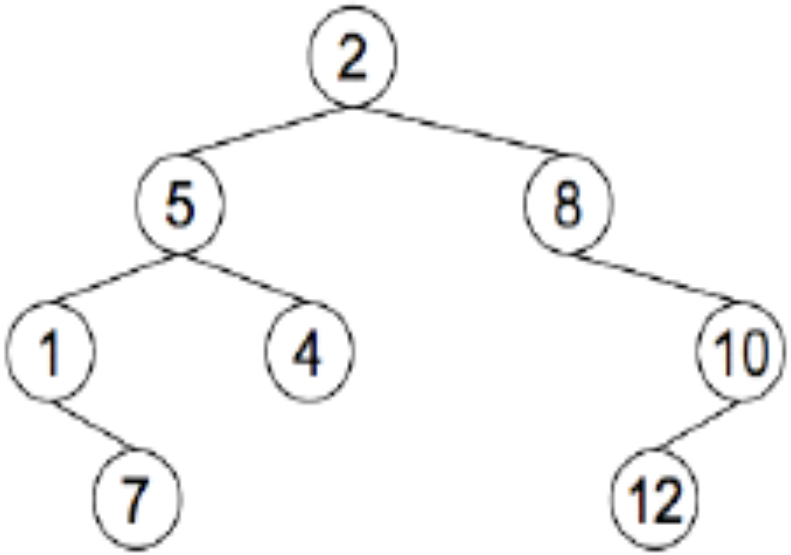
Le parcours post-fixé de T : 7,1,4,5,12,10,8,2.

Parcours postfixé

3. Pour effectuer le parcours *infixé* des nœuds de **A**, on parcourt en infixé les nœuds de A_1 ensuite on passe par la racine n et puis on effectue le parcours infixé de A_2 jusqu'à A_k.

exemple (parcours infixé)

L'arbre T



Le parcours infixé de T : 1, 7, 5, 4, 2, 8, 12, 10.

Les principes des procédures récursives PREFIXE (resp. INFIXE) permettant le parcours préfixé (resp.infixé) des nœuds d'un arbre sont les suivants :

Procédure PREFIXE (n: nœud)

Début
    traiter (n)
    pour chaque fils f de n, s'il y en a, depuis le plus à gauche jusqu'au plus à droite
    faire
        PREFIXE(f)
    finpour
fin

La procédure effectuant le parcours postfixé peut être facilement déduite à partir de la procédure PREFIXE() *(traiter (n) vient après la boucle).*

Procédure INFIXE (n : noeud)

Début

si n est une feuille **alors** traiter(n)

sinon

INFIXE (le fils le plus à gauche de n)

traiter (n)

pour chaque fils f de n, à l'exception du plus à gauche, depuis la gauche jusqu'à la droite **faire**

INFIXE(f)

finpour

finsi

Fin

III. Les Arbres Binaires

III.1 Définitions

❑ Arbre Binaire:

Un arbre binaire est un arbre dans lequel chaque nœud possède 0, 1 ou 2 fils (fils gauche et fils droite).

Un AB peut être vide. Chaque nœud stocke une donnée appelée valeur du nœud.

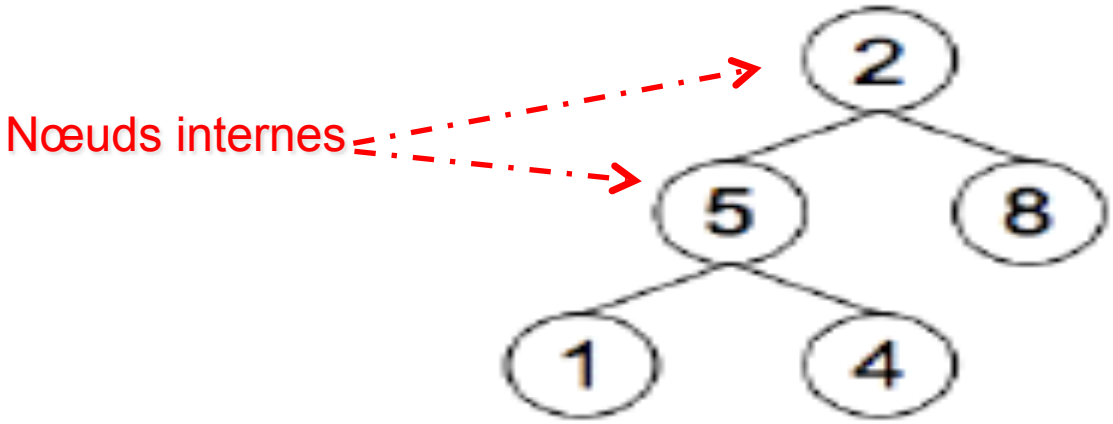
La définition récursive d'un AB se réduit à :

Un AB est :

- ✓ soit vide ;
- ✓ soit formé d'une racine et deux sous-arbres : le sous-arbre gauche et le sous-arbre droit qui sont eux-mêmes des AB.

Arbre Binaire Complet

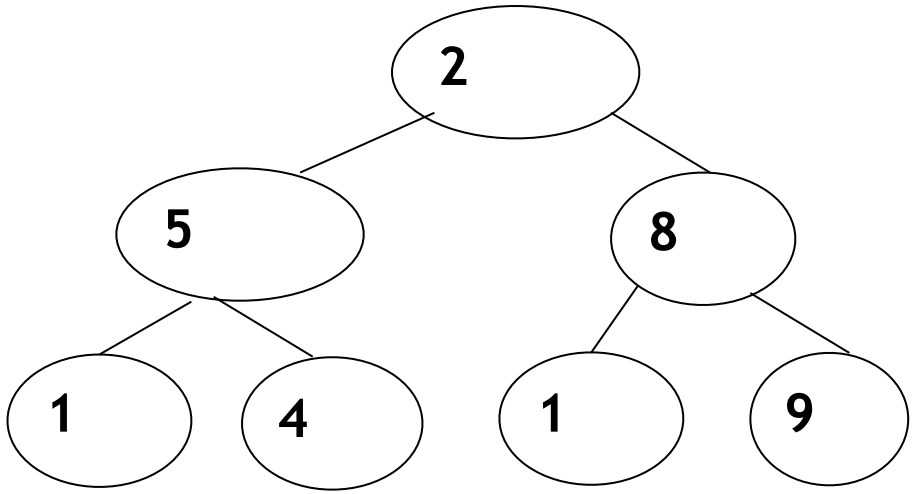
Un arbre binaire est dit *complet* si tout nœud interne possède exactement 2 fils.



Arbre Binaire Complet

Arbre Binaire Parfait

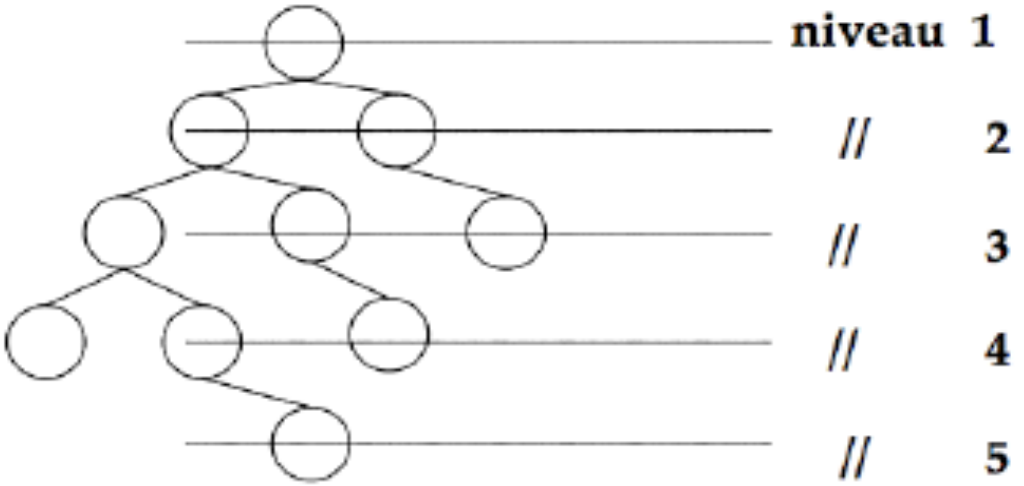
Un arbre binaire est dit *Parfait* si il est complet et que toutes les feuilles sont au même niveau.



Niveau d'un nœud

Le niveau d'un nœud n d'un AB peut être défini comme suit :

- ✓ 1 si n est la racine ;
- ✓ x si le niveau du père de n est x-1.



Niveaux des nœuds dans un arbre

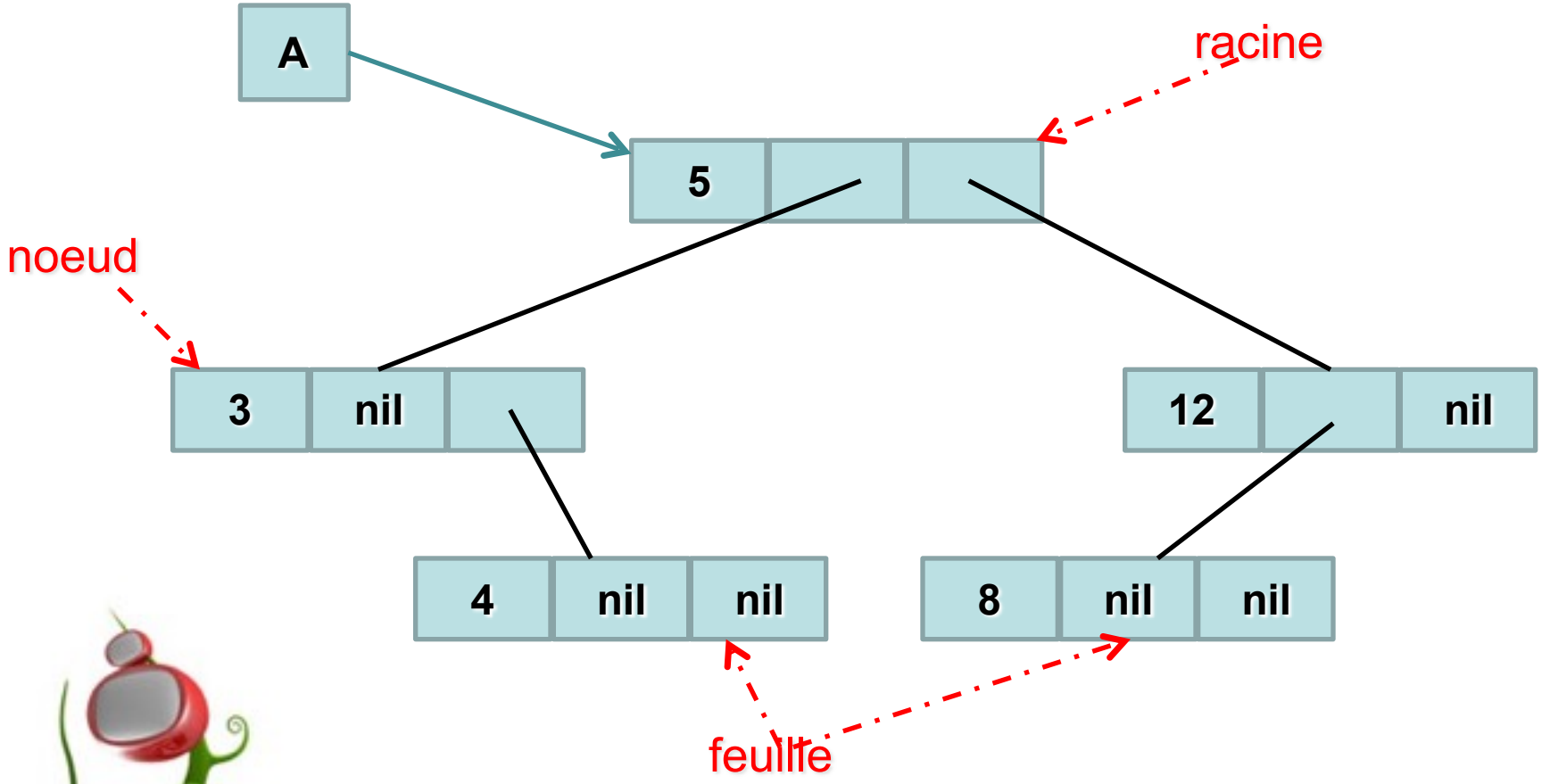
Hauteur d'un arbre : La hauteur d'un AB est le max des niveaux des noeuds.

III.2 Représentation des AB par des pointeurs

On utilise une représentation où chaque nœud de l'arbre contient l'information propre au nœud et deux pointeurs sur les sous-arbres gauche et droite. On définit alors le type NOEUD comme suit :

```
Type  NOEUD = enregistrement
                    val : TypeElement
                    fg, fd : ^NOEUD
                finenregistrement

Type  AB = ^NOEUD
```



Représentation d'un arbre par des pointeurs

Si A est une feuille alors : A^.fg = nil **et** A^.fd = nil

Application:

Soit A un arbre binaire représenté par des pointeurs. Écrire une procédure récursive permettant les parcours préfixé.

```
Procédure PREFIXE (A: AB )

Début
    si A ≠ nil alors
        Traiter (A)
        PREFIXE(A^.fg )
        PREFIXE(A^.fd )
    finsi
Fin
```