

Chapitre 4 : Les arbres

Université Mohamed 1er
Faculté des sciences
Oujda

Sommaire

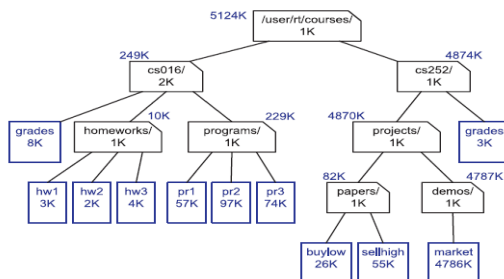
- Notion d'arbre
- Vocabulaire de base
- Arbres binaires
- Implémentation des arbres binaires
- Parcours des arbres binaires

Notion d'arbre

Notion d'arbre

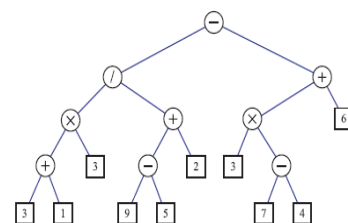
- Un **arbre** est un ensemble d'éléments, appelés **nœuds** (*nodes*) ou **sommets** (*vertex*), organisés d'une manière hiérarchique à partir d'un nœud très particulier, appelé **racine** (*root*) de l'arbre.
- La **structure d'arbre** est **fondamentale en informatique** :
 - Les **fichiers** dans **OS** sont organisés sous forme d'arbre.
 - Un **compilateur** représente un **programme source** sous forme d'un arbre : **l'arbre de syntaxe** (*syntax tree*).

Notion d'arbre



Un **arbre** représentant un **système de fichier**.

Notion d'arbre



Une **expression mathématique** représentée sous d'un **arbre** par un **compilateur**.

Notion d'arbre

- La **structure d'arbre** est utilisée dans de très **nombreuses applications informatiques** :
 - Arbres généalogiques**;
 - Arbres pour représenter des compétitions sportives (**tournois**);
 - Arbres pour les **organigrammes d'entreprises**;
 - Informations **hiérarchiques** (Table des matières d'un livre);
 - Fichiers **HTML, XML**.
 - DOM** (JavaScript), ...etc.

Vocabulaire de base

Vocabulaire de base

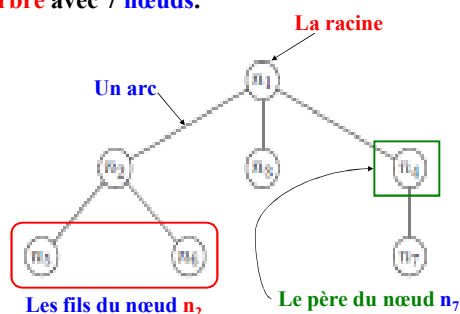
- Définition itérative d'un arbre** :
 - Un **arbre** (*tree*) est un ensemble de **nœuds** reliés entre eux par des **arcs** (*edges*) et possédant les **trois propriétés** suivantes :
 - ✓ Il existe un **nœud particulier unique** appelé **racine**.
 - ✓ Tout nœud n autre que la racine est relié par un **arc** à un autre nœud m (unique), appelé le **père** (*parent/father*) de n .
 - ✓ Tout nœud autre que la racine est relié à la racine par une **série d'arcs** (en traversant les pères successifs) : un arbre est qualifié de **connexe**.

Vocabulaire de base

- Si p est le **père** du nœud q , alors on dit aussi que q est un **fil** (*child*) de p .
- Un **nœud père** peut avoir **0, 1 ou plusieurs fils**.
- Cependant, tout nœud (sauf la racine) a **exactement un père**.
- La **série des pères** partant d'un nœud vers la racine est toujours **unique**.
- En informatique, les arbres sont souvent dessinés avec la **racine en haut** et avec les **pères au-dessus de leurs fils**.

Vocabulaire de base

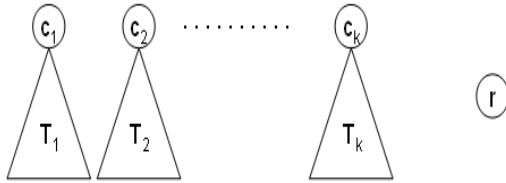
Un **arbre** avec 7 nœuds.



Vocabulaire de base

- Définition récursive d'un arbre** :
 - Un **arbre** est un ensemble de **nœuds** construits récursivement de la manière suivante :
 - ✓ **LA BASE** : un **nœud seul** n est un **arbre** dont la racine est ce nœud n lui-même.
 - ✓ **LA RECURRENCE** : soit r un **nouveau nœud** et soient T_1, T_2, \dots, T_k des **arbres** ($k \geq 1$) ayant respectivement pour racines c_1, c_2, \dots, c_k . On suppose qu'aucun nœud n n'apparaisse plus d'une fois dans les T_i . Comme r est un nouveau nœud, il ne peut donc pas apparaître dans un de ces arbres.

Vocabulaire de base

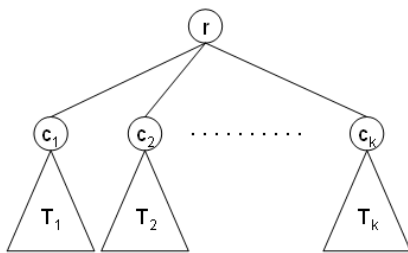


Vocabulaire de base

▪ **Définition récursive d'un arbre :**

- On forme un **nouvel arbre** T à partir de r et de T_1, T_2, \dots, T_k comme suit :
 - ✓ Faire de r la **racine** de l'arbre T ;
 - ✓ **Ajouter un arc** entre r et chacun des racines c_1, c_2, \dots, c_k , de manière que chacun de ces nœuds soit un fils de la racine r .

Vocabulaire de base

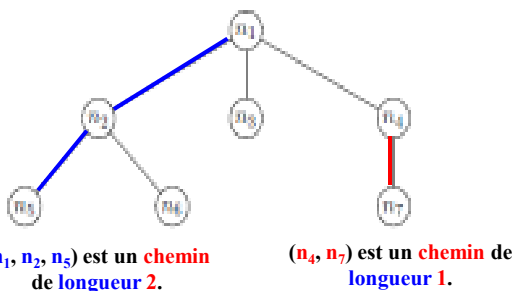


Vocabulaire de base

▪ **Chemins :**

- Soient Ω un **arbre** et (n_1, n_2, \dots, n_k) une **séquence de nœuds** de Ω .
- On dit que (n_1, n_2, \dots, n_k) est un **chemin (path)** depuis n_1 jusqu'à n_k dans l'arbre Ω , si n_i est le **père** de n_{i+1} , pour $i=1, 2, \dots, k-1$.
- La **longueur d'un chemin** (n_1, n_2, \dots, n_k) est le **nombre d'arcs** dans ce chemin (c'est aussi le nombre de nœuds dans le chemin auquel on retranche 1).
- Par convention, un **nœud singulier** (n) est un **chemin de longueur 0**.

Vocabulaire de base



Vocabulaire de base

▪ **Unicité des chemins dans un arbre :**

- **Proposition :**
 - ✓ Dans un **arbre** Ω , il **ne peut y avoir qu'un seul chemin** entre un **nœud** n et un autre **nœud** m .

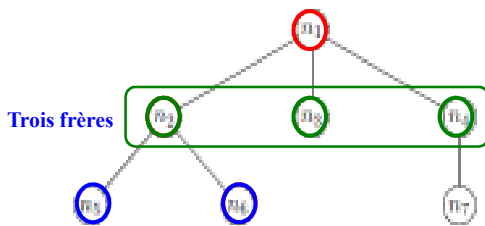
Vocabulaire de base

- **Ancêtres et descendants :**
 - Soit (n_1, n_2, \dots, n_k) un chemin dans un arbre Ω .
 - n_1 est appelé un **ancêtre** (*ancestore*) de n_k .
 - n_k est appelé un **descendant** de n_1 .
 - Si $n_1 \neq n_k$, on dit que
 - ✓ n_1 est un **ancêtre propre** de n_k ,
 - ✓ n_k est un **descendant propre** de n_1 .

Vocabulaire de base

- **Ancêtres, Descendants et Frères :**
 - Les **ancêtres d'un nœud** sont formés par **le nœud lui-même**, de **son père**, puis **le père de son père** et ainsi de suite **jusqu'à la racine**.
 - Les **descendants d'un nœud** sont formés par ce **nœud lui-même**, de **ses fils**, puis **les fils de ses fils** et **ainsi de suite**.
 - Les **nœuds ayant le même père** sont appelés des **frères** (*siblings*).

Vocabulaire de base



Ancêtres de n_6 : n_6 , n_2 et n_1

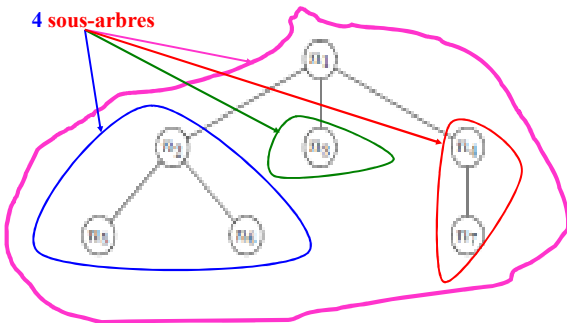
Descendants propres de n_2 : n_5 et n_6

Vocabulaire de base

- **Sous-arbres :**
 - Dans un arbre Ω , un **nœud n accompagné de tous ses descendants propres** (s'il en possède) est appelé un **sous-arbre** (*subtree*) de Ω de racine n .

Vocabulaire de base

4 sous-arbres

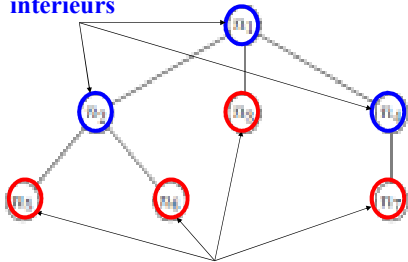


Vocabulaire de base

- **Feuilles, Nœuds intérieurs :**
 - Une **feuille** (*leaf*) est un **nœud** d'un arbre qui **n'a pas de fils**.
 - Un **nœud intérieur** (*interior node*) est un **nœud** qui **au moins un fils**.
 - **Remarque :**
 - ✓ Dans un arbre Ω , tout nœud est :
 - soit une feuille;
 - soit un nœud intérieur.
 - ✓ mais pas les deux.

Vocabulaire de base

Les nœuds intérieurs



Les feuilles (*Leaves*)

Vocabulaire de base

Hauteurs d'un nœud, Hauteur d'un arbre :

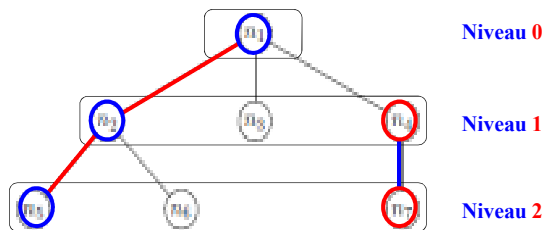
- Dans un **arbre** Ω , la **hauteur d'un nœud** (*height*) n est la **longueur du plus long chemin depuis n jusqu'à une feuille**.
- La **hauteur d'un arbre** est celle de sa **racine**.

Profondeur d'un nœud :

- Dans un **arbre** Ω , la **profondeur** (*depth*) ou le **niveau** (*level*) **d'un nœud n** est la **longueur du chemin depuis la racine jusqu'à n** .
- La **profondeur d'un arbre** est la **valeur maximale des profondeurs** de ses nœuds.

Vocabulaire de base

Hauteurs ou Niveaux :



La profondeur de $n_5 = 2$

La hauteur de $n_4 = 1$

Vocabulaire de base

Profondeur d'un nœud ou d'un arbre :

- Soit x un nœud d'un **arbre** Ω (on écrit $x \in \Omega$).
- Alors la **profondeur** (*level/depth*) de x se calcule **récurivement par la formule suivante** :
 - ✓ $depth(x) = 0$, si $x = root(\Omega)$.
 - ✓ $depth(x) = 1 + depth(father(x))$, sinon.
- La **profondeur** de l'arbre Ω est :
 - ✓ $depth(\Omega) = \text{Max} \{depth(x) \mid x \in \Omega\}$

Vocabulaire de base

Hauteurs d'un nœud, Hauteur d'un arbre :

- La **hauteur** (*height*) de x se calcule **récurivement par la formule suivante** :
 - ✓ $height(x) = 0$, si $x \in \text{Leaves}(\Omega)$ (c'est-à-dire, si x est une feuille de Ω)
 - ✓ $height(x) = 1 + \text{Max} \{height(y) \mid y \in \text{Childs}(x)\}$, sinon.
- La **hauteur** de l'arbre Ω est :
 - ✓ $height(\Omega) = height(root(\Omega)) = depth(\Omega)$

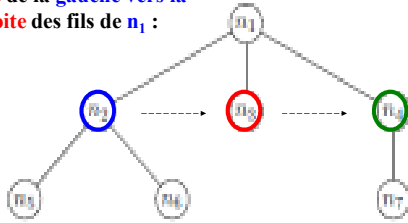
Vocabulaire de base

Arbres ordonnés et ordre gauche->droite :

- Un **arbre** Ω est dit **ordonné** (*ordered tree*) s'il existe une **relation d'ordre** sur les **nœuds de cet arbre** (une relation binaire réflexive, transitive et antisymétrique).
- On peut affecter un **ordre de la gauche vers la droite** aux fils d'un nœud.

Vocabulaire de base

Ordre de la gauche vers la droite des fils de n_1 :



Vocabulaire de base

Arbres ordonnés et ordre gauche->droite :

- Cet **ordre de la gauche vers la droite** peut être étendue de manière à ordonner tous les nœuds d'un arbre :
 - Si m et n sont des frères et si m est à la gauche de n , alors tous les descendants de m sont à la gauche de tous les descendants de n .

Vocabulaire de base

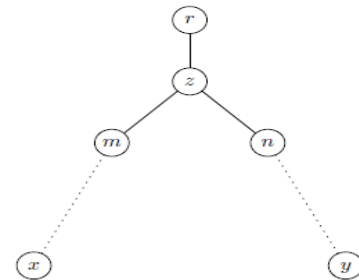
Arbres ordonnés et ordre gauche->droite :

Propriété :

- Soient x et y deux nœuds d'un même arbre Ω tels qu'aucun ne soit l'ancêtre de l'autre. Alors, l'un des deux nœuds sera à la gauche de l'autre.

Vocabulaire de base

Arbres ordonnés et ordre gauche->droite



Vocabulaire de base

Arbres étiquetés :

- un **arbre étiqueté** est un arbre dans lequel chaque nœud possède une **étiquette (label)** ou une **valeur**.
- L'étiquette d'un nœud représente généralement une **information associée au nœud** considéré.
- Elle peut être :
 - soit **simple** (un entier);
 - soit **complexe** (le texte d'un document).

Vocabulaire de base

Arbres étiquetés :

- L'étiquette d'un nœud est **modifiable**, alors que son **nom** ne l'est pas.
- Il est possible que plusieurs nœuds aient la **même étiquette**, mais le **nom** de chaque nœud doit être unique.

Vocabulaire de base

Un arbre étiqueté

FSO, Filière SMI-S4, Printemps 2017 Prof. Abdelmajid DARGHAM 37

Vocabulaire de base

- **Arbre vide, Arbre-feuille, Arbre dégénéré :**
 - Un **arbre** Ω est **vide**, s'il **n'admet aucun nœud**. On le note par : \emptyset .
 - Un **arbre-feuille** est un **arbre réduit à sa racine** r . On le note par $\Omega = (r)$.
 - Un **arbre** est **dégénéré**, (ou **filiforme**) si **tous ses nœuds ont au plus un descendant**.
 - ✓ Un tel arbre possède un **seul chemin simple**.
 - ✓ On le note donc par $\Omega = (n_1, n_2, \dots, n_k)$.

FSO, Filière SMI-S4, Printemps 2017 Prof. Abdelmajid DARGHAM 38

Vocabulaire de base

Un arbre-feuille

Deux arbres dégénérés (filiformes)

FSO, Filière SMI-S4, Printemps 2017 Prof. Abdelmajid DARGHAM 39

Vocabulaire de base

- **Taille d'un arbre, Degré d'un nœud, Branche d'un arbre :**
 - La **taille d'un arbre** Ω est égale au **nombre de ses nœuds**.
 - Le **degré d'un nœud** n dans un **arbre** Ω est égal au **nombre de ses fils (successeurs)**.
 - Une **branche d'un arbre** Ω est un **chemin** qui relie la **racine** à une **feuille** de Ω .
 - ✓ C'est donc un chemin $C = (n_1, n_2, \dots, n_k)$ tel que $n_1 = \text{root}(\Omega)$ et $n_k \in \text{Leaves}(\Omega)$.

FSO, Filière SMI-S4, Printemps 2017 Prof. Abdelmajid DARGHAM 40

Vocabulaire de base

Un arbre Ω

Taille(Ω) = 11

Une **branche** de Ω :
(A, B, E, I, K)

Nœud	A	B	C	D	E	F	G	H	I	J	K
Degré	3	2	0	2	1	0	0	0	2	0	0

Le **degré** de Ω : 3

FSO, Filière SMI-S4, Printemps 2017 Prof. Abdelmajid DARGHAM 41

Vocabulaire de base

- **Propriété :**
 - Un **arbre** a autant de **branches** que de **feuilles**.

FSO, Filière SMI-S4, Printemps 2017 Prof. Abdelmajid DARGHAM 42

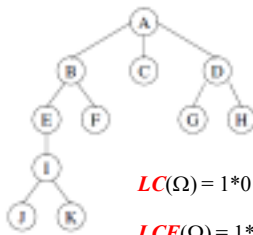
Vocabulaire de base

- **Longueur de cheminement d'un arbre :**
 - La **longueur de cheminement** d'un **arbre** Ω est égale à la **somme des profondeurs** de tous les **nœuds** :
 - ✓ $LC(\Omega) = \sum depth(x), x \in \Omega$.
 - La **longueur de cheminement externe** d'un **arbre** Ω est égale à la **somme des profondeurs** de ses **feuilles** :
 - ✓ $LCE(\Omega) = \sum depth(x), x \in Leaves(\Omega)$.

Vocabulaire de base

- **Longueur de cheminement d'un arbre :**
 - La **longueur de cheminement interne** d'un **arbre** Ω est égale à la **somme des profondeurs** de ses **nœuds intérieurs** :
 - ✓ $LCI(\Omega) = \sum depth(x), x \in Internals(\Omega)$.
 - On a évidemment la relation :
 - ✓ $LC(\Omega) = LCE(\Omega) + LCI(\Omega)$

Vocabulaire de base



$$LC(\Omega) = 1*0 + 3*1 + 4*2 + 1*3 + 2*4 = 22$$

$$LCE(\Omega) = 1*1 + 3*2 + 2*4 = 15$$

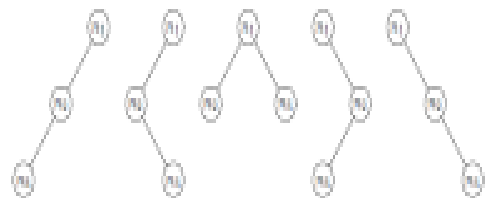
$$LCI(\Omega) = 1*0 + 2*1 + 1*2 + 1*3 = 7$$

Arbres binaires

Arbres binaires

- **Définition :**
 - Un **arbre binaire** (*binary tree*) est un arbre dans lequel **chaque nœud** peut avoir **au plus deux fils** :
 - ✓ un **fils gauche** (*left child*),
 - ✓ et/ou un **fils droit** (*right child*).
- **Remarque :**
 - Le **vocabulaire** pour les **arbres généraux** s'applique aussi aux **arbres binaires**.

Arbres binaires

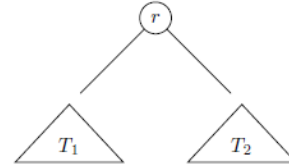


Les 5 arbres binaires avec 3 nœuds.

Arbres binaires

- **Définition (récursive) :**
 - un **arbre binaire** est un arbre défini récursivement comme suit :
 - ✓ **La base :** l'**arbre vide** est un **arbre binaire**.
 - ✓ **La récurrence :** si r un **nouveau nœud** et si T_1 et T_2 sont **deux arbres binaires**, alors on peut créer un **nouvel arbre binaire** tel que :
 - r est la **racine**;
 - T_1 est le **sous-arbre gauche**;
 - T_2 et le **sous-arbre droit**.

Arbres binaires

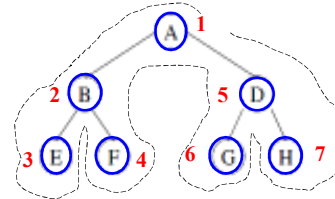


Définition récursive d'un arbre binaire

Arbres binaires

- **Ordres classiques dans un arbre binaire :**
 - Soit $\Omega = (r, \Omega_g, \Omega_d)$ un **arbre binaire** de racine r , de sous-arbre gauche Ω_g et de sous-arbre droit Ω_d .
 - On peut définir **trois ordres classiques** sur Ω :
 - ✓ L'ordre **préfixe (ou pré-ordre)** : r , puis Ω_g , puis Ω_d .
 - ✓ L'ordre **infixe (ou symétrique)** : Ω_g , puis r , puis Ω_d .
 - ✓ L'ordre **postfixe (ou suffixe)** : Ω_g , puis Ω_d , puis r .

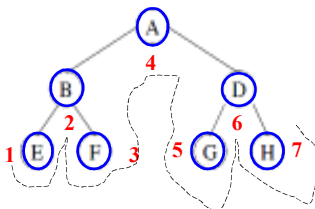
Arbres binaires



L'ordre préfixe

L'ordre préfixe donne : A, B, E, F, D, G, H

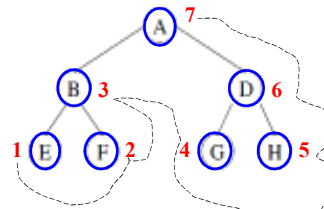
Arbres binaires



L'ordre infixe

L'ordre infixe donne : E, B, F, A, G, D, H

Arbres binaires



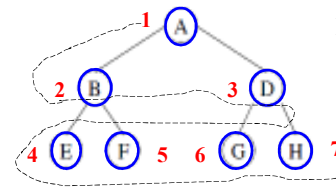
L'ordre postfixe

L'ordre postfixe donne : E, F, B, G, H, D, A

Arbres binaires

- **Parcours hiérarchique (ou en largeur) :**
 - Les **trois ordres classiques** (**préfixe**, **infixe**, **postfixe**) définissent ce qu'on appelle le **parcours en profondeur**.
 - Un autre parcours important d'arbres est appelé **parcours en largeur** (**parcours hiérarchique** ou **parcours par niveaux**).
 - Il consiste à parcourir un arbre **niveau par niveau**, en **partant de la racine**, et **en suivant un ordre de gauche à la droite** pour **chaque niveau**.

Arbres binaires

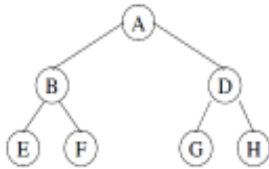


L'ordre hiérarchique

L'ordre hiérarchique donne : A, B, D, E, F, G, H

Arbres binaires

- **Arbre binaire complet :**
 - Un **arbre binaire complet** (*full binary tree*) est un **arbre binaire** dont tous les **nœuds intérieurs sont de degré 2**.



Un arbre binaire complet de profondeur 2

Arbres binaires

- **Propriété :**
 - Un **arbre binaire complet** (**non vide**) ayant n **nœuds intérieurs possède** $(n + 1)$ **feuilles**.
- **Preuve :**
 - Par **récurrence sur le nombre des nœuds intérieurs** n de l'arbre Ω .
 - On pose :
 - ✓ $n_f(\Omega)$ = le nombre de feuilles de Ω .
 - ✓ $n_i(\Omega)$ = le nombre de nœuds intérieurs de Ω .

Arbres binaires

- **Preuve (suite) :**
 - **La base :**
 - ✓ Si $n_i(\Omega) = 0$, alors Ω est réduit à sa racine qui est sa seule feuille ($n_f(\Omega) = 1$).
 - **La récurrence :**
 - ✓ (HR) : soit $n > 0$ et supposant que pour tout arbre binaire complet Ω tel que : $0 \leq n_i(\Omega) < n$, alors $n_f(\Omega) = 1 + n_i(\Omega)$.
 - ✓ Montrons la relation pour n : soit alors un arbre binaire complet Ω tel que $n_i(\Omega) = n$.

Arbres binaires

- **Preuve (suite) :**
 - ✓ Comme $n > 0$, l'arbre binaire complet Ω est composé d'une racine r et de deux sous-arbres (complets) Ω_1 et Ω_2 .
 - ✓ Posons : $n_1 = n_i(\Omega_1)$ et $n_2 = n_i(\Omega_2)$.
 - ✓ Il est clair que : $n = n_1 + n_2 + 1$.
 - ✓ De plus, on a : $0 \leq n_1 < n$ et $0 \leq n_2 < n$.
 - ✓ Par (HR) : $n_f(\Omega_1) = 1 + n_1$ et $n_f(\Omega_2) = 1 + n_2$. Or, $n_f(\Omega) = n_f(\Omega_1) + n_f(\Omega_2) = (1 + n_1) + (1 + n_2) = (n_1 + n_2 + 1) + 1 = n + 1$.

Implémentation des arbres binaires

FSO, Filière SMI-S4, Printemps 2017

Prof. Abdelmajid DARGHAM

61

Implémentation des arbres binaires

Implémentation contiguë (première solution) :

- On utilise un **tableau d'enregistrements** composés de trois champs:
 - ✓ « *info* » : stocke la **valeur** du nœud.
 - ✓ « *left* » : stocke l'**indice du fils gauche** du nœud.
 - ✓ « *right* » : stocke l'**indice du fils droit** du nœud.
- Une **variable** est utilisée pour **mémoriser l'indice de la racine de l'arbre**.

FSO, Filière SMI-S4, Printemps 2017

Prof. Abdelmajid DARGHAM

62

Implémentation des arbres binaires

Codage en C :

```
#define MAX_SIZE 100
typedef struct Node {
    int info, left, right;
} Node;

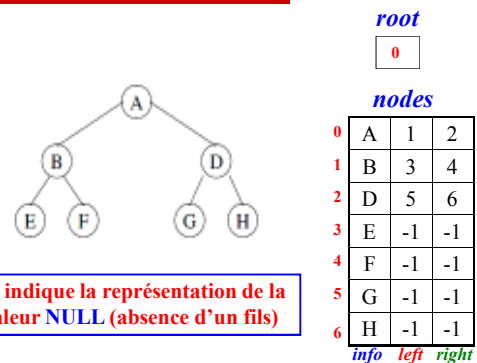
typedef struct tree {
    Node nodes[MAX_SIZE];
    int root;
} tree;
```

FSO, Filière SMI-S4, Printemps 2017

Prof. Abdelmajid DARGHAM

63

Implémentation des arbres binaires



FSO, Filière SMI-S4, Printemps 2017

Prof. Abdelmajid DARGHAM

64

Implémentation des arbres binaires

Implémentation contiguë (deuxième solution) :

- Il suffit de **ranger les nœuds** dans l'**ordre préfixé** et d'**associer à chaque nœud**, les **indicateurs** suivants :
 - ✓ « *info* » : stocke la **valeur** du nœud.
 - ✓ « *left* » : un **booléen** qui indique la présence d'un **sous-arbre gauche** (non vide).
 - ✓ « *right* » : un **booléen** qui indique la présence d'un **sous-arbre droit**.

FSO, Filière SMI-S4, Printemps 2017

Prof. Abdelmajid DARGHAM

65

Implémentation des arbres binaires

Codage en C :

```
#define MAX_SIZE 100
typedef struct Node {
    int info;
    enum {False, True} left, right;
} Node;
typedef struct tree {
    Node nodes[MAX_SIZE];
    int root;
} tree;
```

FSO, Filière SMI-S4, Printemps 2017

Prof. Abdelmajid DARGHAM

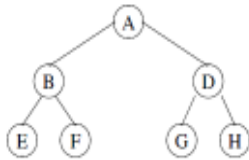
66

Implémentation des arbres binaires

L'ordre **préfixe** donne :
A, B, E, F, D, G, H

root

0



nodes

0	A	1	1
1	B	1	1
2	E	0	0
3	F	0	0
4	D	1	1
5	G	0	0
6	H	0	0

info left right

-1 indique la représentation de la valeur NULL (absence d'un fils)

FSO, Filière SMI-S4, Printemps 2017

Prof. Abdelmajid DARGHAM

67

Implémentation des arbres binaires

Implémentation contiguë pour un arbre binaire complet :

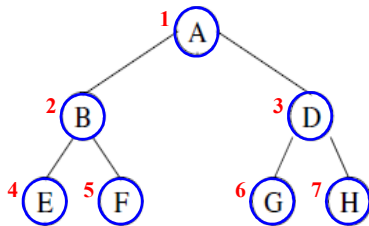
- On **numérote les nœuds** de l'arbre en effectuant un « **parcours en largeur** », **en commençant par 1**.
- Puis, il suffit de considérer que ce **numéro** est l'indice du **i^{ème} nœud** selon cet ordre de parcours.

FSO, Filière SMI-S4, Printemps 2017

Prof. Abdelmajid DARGHAM

68

Implémentation des arbres binaires



FSO, Filière SMI-S4, Printemps 2017

Prof. Abdelmajid DARGHAM

69

Implémentation des arbres binaires

Implémentation contiguë pour un arbre binaire complet :

- On constate que les numéros obéissent aux règles suivantes :
 - ✓ Le **numéro de la racine** est **1**.
 - ✓ Le **numéro du fils gauche** = le **double de celui de son père**.
 - ✓ Le **numéro du fils droit** = le **double de celui de son père, plus 1**.

FSO, Filière SMI-S4, Printemps 2017

Prof. Abdelmajid DARGHAM

70

Implémentation des arbres binaires

Implémentation contiguë pour un arbre binaire complet :

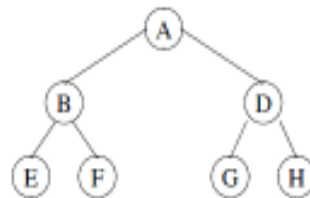
- On utilisera tout simplement un tableau V tel que :
 - ✓ V[1] : stocke l'**information associée** à la **racine**.
 - ✓ V[2*i] : stocke l'**information associée** au **fils gauche** du **nœud** i.
 - ✓ V[2*i+1] : stocke l'**information associée** au **fils droit** du **nœud** i.

FSO, Filière SMI-S4, Printemps 2017

Prof. Abdelmajid DARGHAM

71

Implémentation des arbres binaires



	A	B	D	E	F	G	H
0	1	2	3	4	5	6	7

FSO, Filière SMI-S4, Printemps 2017

Prof. Abdelmajid DARGHAM

72

Implémentation des arbres binaires

- **Implémentation contiguë pour un arbre binaire complet :**
 - Si $i \neq 1$, $V[i / 2]$: stocke l'**information associée** au père du **nœud** i .

Implémentation des arbres binaires

- **Implémentation chaînée d'un arbre binaire :**
 - Les **nœuds** sont représentés par des **enregistrements avec trois champs** qui sont :
 - ✓ « **info** » : d'un **certain type** (stocke la valeur du nœud).
 - ✓ « **left** » : un **pointeur vers le fils gauche** du nœud.
 - ✓ « **right** » : un **pointeur vers le fils droit** du nœud.
 - La valeur **NULL** reflète l'**absence du fils**.

Implémentation des arbres binaires

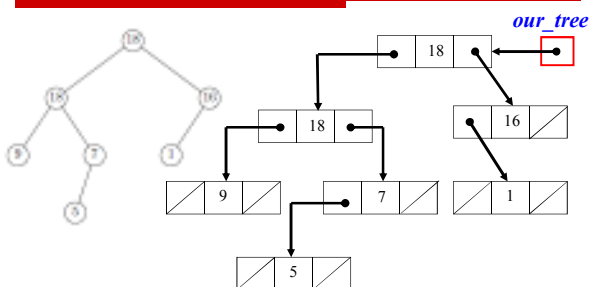
- **Implémentation chaînée d'un arbre binaire :**
 - **L'arbre binaire** lui-même est déterminé par **l'adresse de sa racine**, c'est-à-dire qu'il est représenté par un pointeur vers sa racine.
 - **L'arbre binaire vide** est naturellement représenté par **NULL**.

Implémentation des arbres binaires

- **Code C de la représentation chaînée d'un arbre binaire :**

```
typedef struct Node* tree;
typedef struct Node
{
    tree left; /* le fils gauche */
    int info;
    tree right; /* le fils droit */
} Node;
```

Implémentation des arbres binaires



Parcours des arbres binaires

Parcours des arbres binaires

- **Parcours d'un arbre binaire :**
 - Plusieurs opérations sur les arbres binaires nécessitent le **parcours de l'arbre** tout entier.
 - Par exemple, **l'affichage d'un arbre binaire** nécessite de parcourir tous les nœuds de l'arbre.
 - En partant d'un **nœud N**, nous pouvons effectuer l'une des actions suivantes :

Parcours des arbres binaires

- **Parcours d'un arbre binaire :**
 - ✓ **Visiter** le **nœud (N)**;
 - ✓ **Traverser récursivement** le **sous-arbre gauche (G)**;
 - ✓ **Traverser récursivement** le **sous-arbre droit (D)**;

Parcours des arbres binaires

- **Parcours d'un arbre binaire :**
 - ✓ Si nous adoptons cette stratégie, nous aurons trois types de parcours :
 - ✓ Le **parcours NGD** ou **préfixe (preorder)**
 - ✓ Le **parcours GND** ou **infixe (inorder)**
 - ✓ Le **parcours GDN** ou **postfixe (postorder)**

Parcours des arbres binaires

- **Parcours préfixe :**
- ```
void doPrefixe(tree currentNode)
{
 if(currentNode != NULL)
 {
 /* do some things with currentNode */
 doPrefixe(currentNode->left);
 doPrefixe(currentNode->right);
 }
}
```

## Parcours des arbres binaires

- **Parcours infixe :**
- ```
void doInfixe(tree currentNode)
{
    if(currentNode != NULL)
    {
        doInfixe(currentNode->left);
        /* do some things with currentNode */
        doInfixe(currentNode->right);
    }
}
```

Parcours des arbres binaires

- **Parcours postfixe :**
- ```
void doPostfixe(tree currentNode)
{
 if(currentNode != NULL)
 {
 doPostfixe(currentNode->right);
 doPostfixe(currentNode->left);
 /* do some things with currentNode */
 }
}
```

## Parcours des arbres binaires

- **Parcours d'un arbre binaire :**
  - ✓ Pour effectuer un **parcours total de l'arbre**, ces trois fonctions **doivent être appelées** en leur **passant la racine de l'arbre** comme **argument effectif** :
    - *tree root*;
    - *doPrefixe(root)*;
    - *doInfixe(root)*;
    - *doPostfixe(root)*;

FSO, Filière SMI-S4, Printemps 2017

Prof. Abdelmajid DARGHAM

85

## Parcours des arbres binaires

- **Complexité du parcours d'un arbre binaire :**
  - ✓ Soit un **arbre binaire** ayant ***n* nœuds**.
  - ✓ La **complexité** sera comptée **en nombre de traitements de nœuds**.
  - ✓ **Propriété :**
    - La complexité de chacun des trois **parcours en profondeur d'arbres binaires** est **linéaire** (en  $\Theta(n)$ ).

FSO, Filière SMI-S4, Printemps 2017

Prof. Abdelmajid DARGHAM

86

## Parcours des arbres binaires

- **Exemple 1 : calculer la taille d'un arbre binaire**
- ```
int getSize( tree root)
{
    if(root == NULL)
        return 0;
    else
        return
            1 + getSize(root->left) + getSize(root->right);
}
```

FSO, Filière SMI-S4, Printemps 2017

Prof. Abdelmajid DARGHAM

87

Parcours des arbres binaires

- **Exemple 2 : calculer la profondeur d'un arbre binaire**
- ```
int getHeight(tree root) {
 if(root == NULL)
 return -1;
 else {
 int lh = getHeight(root->left);
 int rh = getHeight(root->right);
 return 1 + (lh > rh ? lh : rh); }
}
```

FSO, Filière SMI-S4, Printemps 2017

Prof. Abdelmajid DARGHAM

88