Cours Structures de données

Arbres (Trees)

Pr F.Omary

2019-2020

Objectifs

- **Etudier des structures non linéaires**
  - **Arbres binaires**
  - **Arbres binaires de recherche**
  - **Arbres maximiers ou Tas**
  - **Arbres équilibrés**

Contenu

- Introduction
- Terminologie
- Arbres binaires
- Arbres binaires de recherche
- Arbres maximiers ou Tas
- Arbres équilibrés

Arbres (Trees)
Introduction

Notion d'Arbre (Tree)

- **Les arbres sont les structures de données les plus importantes en informatique**
- **Ce sont des *structures non linéaires* qui permettent d'obtenir des algorithmes plus performants que lorsqu'on utilise des structures de données linéaires telles que les listes et les tableaux**
- **Ils permettent une organisation naturelle des données**

Notion d'Arbre (Tree) Exemples

- ***Organisation des fichiers dans les systèmes d'exploitation ;***
- ***Organisation des informations dans un système de bases de données ;***
- ***Représentation de la structure syntaxique des programmes sources dans les compilateurs ;***
- ***Représentation d'une table de matières ;***
- ***Représentation d'un arbre généalogique ;***
- ***…***

Arbres (Trees)
Terminologie

Terminologie (1)

- Un arbre est un ensemble d'éléments appelés ***nœuds*** (***ou sommets***), liés par une relation (*dite de "parenté")* induisant une structure hiérarchique parmi ces nœuds.

- Un nœud, comme tout élément d'une liste, peut être de n'importe quel type.

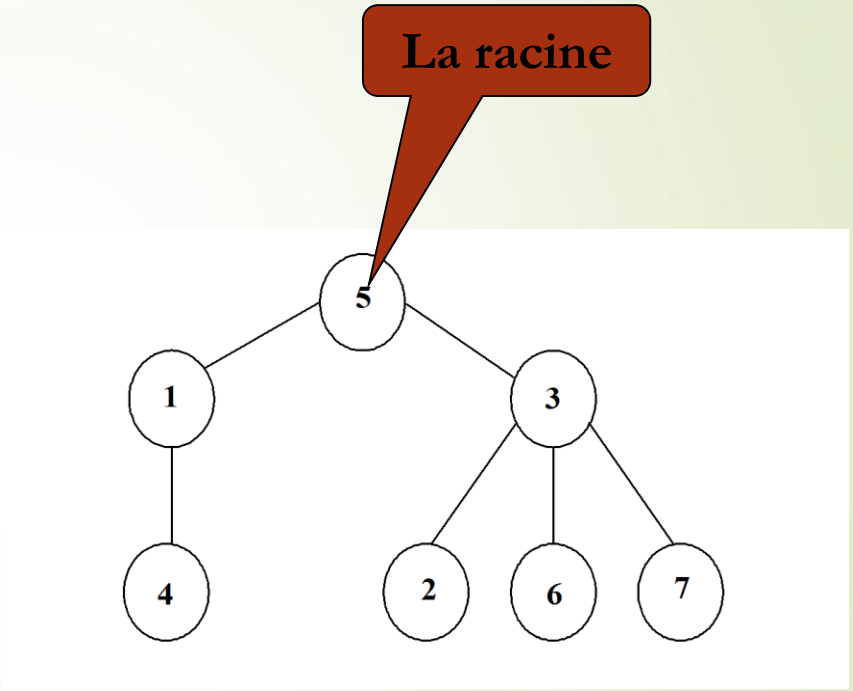
Terminologie (1) (suite)

- D'une manière plus formelle, une structure d'arbre de type de base T est :
  - soit la ***structure vide*** ;
  - soit un noeud de type T, appelé ***racine***, associé à un nombre fini de structures d'arbre disjointes du type de base T appelées ***sous arbres***
- C'est une définition récursive ; la récursivité est une propriété des arbres et des algorithmes qui les manipulent
- **Une liste** est un cas particulier des arbres (***arbre dégénéré***), où tout noeud a au plus un sous arbre

Illustration & Exemple

- Pour illustrer une structure d'arbre, on modélise le plus souvent un nœud par une information inscrite dans un cercle et les liens par des traits.

- ***Par convention***, on dessine les arbres avec la racine en haut et les branches dirigées vers le bas.



Exemple d'arbre formé de 7 nœuds (des entiers)

Terminologie (2)

- **La terminologie utilisée dans les structures d'arbres est empruntée :**
  - **aux arbres généalogiques :**
    - ***Père ;***
    - ***Fils ;***
    - ***Frère ;***
    - ***Descendant ;***
    - **…**
  - **et à la botanique :**
    - ***Feuille ;***
    - ***Branche ;***
    - **…**

Terminologie (3)

- ***Fils (ou enfants) :***
  - **Chaque nœud d'un arbre pointe vers un ensemble éventuellement vide d'autres nœuds ; ce sont ses fils (*ses enfants*).**
  - **Sur l'exemple précédent, le nœud 5 a deux fils : 1 et 3, le nœud 1 a un fils : 4, et le nœud 3 a trois fils : 2, 6 et 7.**
- ***Père :***
  - **Tous les nœuds d'un arbre, sauf un, ont un père et un seul. Un nœud p est père du nœud n si et seulement si n est fils de p.**
  - **Par exemple, le père de 2 est 3, celui de 3 et 5.**
- ***Frères :***
  - **Deux nœuds ayant le même père.**
  - **Les nœuds 2, 6 et 7 sont des frères.**
- ***Racine :***
  - **Le seul nœud sans père.**
  - **5 est la racine de l'arbre précédent.**

Terminologie (4)

- ***Feuilles (ou nœuds terminaux, ou nœuds externes) :***
  - **Ce sont des noeuds sans fils.**
  - **Par exemple, 4, 2, 6 et 7.**
- ***Nœud interne :***
  - **Un noeud qui n'est pas terminal.**
  - **Par exemple, 1, 3 et 5.**
- ***Degré d'un noeud :***
  - **Le nombre de fils de ce noeud.**
  - **Sur l'exemple, 5 est de degré deux, 1 est de degré un, 3 est de degré trois et les feuilles (4, 2, 6, 7) sont de degré nul.**
- ***Degré d'un arbre (ou arité) :***
  - **Plus grand degré des nœuds de l'arbre. Un arbre de degré n est dit *n-aire***
  - **Sur l'exemple, l'arbre est un arbre 3-aire.**

Terminologie (5)

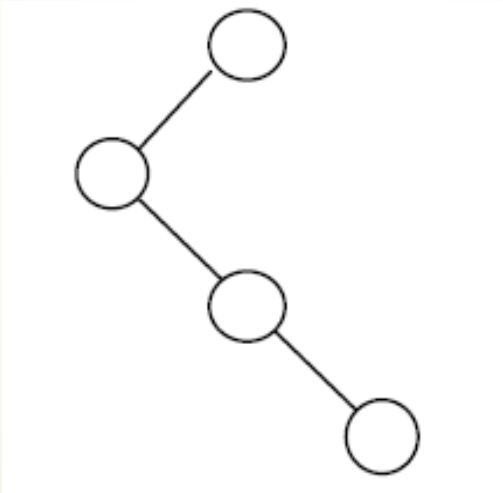
- *Taille d'un arbre :*
  - Le nombre total des nœuds de l'arbre.
  - Sur l'exemple, l'arbre est de taille 7.
- *Chemin :*
  - Une suite de noeuds d'un arbre $(n_1, n_2, \ldots, n_k)$ tel que n_i = père(n_i+1) pour $1 \leq i \leq k-1$ est appelée chemin entre le nœud n_1 et le nœud n_k.
  - La longueur d'un chemin est égale au nombre de nœuds qu'il contient moins 1.
  - Sur l'exemple, le chemin qui mène du nœud 5 au nœud 6 est de longueur 2.
- *Branche :*
  - Un chemin qui commence à la racine et se termine à une feuille.
  - Par exemple, les chemins (5, 1, 4), (5, 3, 2), (5, 3, 6) et (5, 3, 7).
- *Ancêtre :*
  - Un nœud A est un ancêtre d'un nœud B s'il existe un chemin de A vers B.
  - Par exemple, les ancêtres de 2 sont 2, 3 et 5
- *Descendant :*
  - Un nœud A est un descendant d'un nœud B s'il existe un chemin de B vers A.
  - Sur l'exemple, 5 admet les 7 nœuds de l'arbre comme descendants.

Terminologie (6)

- ***Sous arbre :***
  - **Un sous arbre d'un arbre A est constitué de tous les descendants d'un nœud quelconque de A.**
  - **Les ensembles de noeuds {3, 2, 6, 7} et {2} forment deux sous arbres de l'exemple précédent.**
- ***Hauteur (ou profondeur, ou niveau) d'un noeud :***
  - **Longueur du chemin qui relie la racine à ce nœud.**
  - **La racine est elle même de hauteur 0, ses fils sont de hauteur 1, et les autres noeuds de hauteur supérieure à 1.**
- ***Hauteur d'un arbre :***
  - **Plus grande profondeur des nœuds de l'arbre supposé non vide, c'est-à-dire *h(A) = Max{h(x) ; x noeud de A}***
  - **L'arbre de l'exemple est de profondeur 2.**
  - ***Par convention*, un arbre vide a une hauteur de -1.**

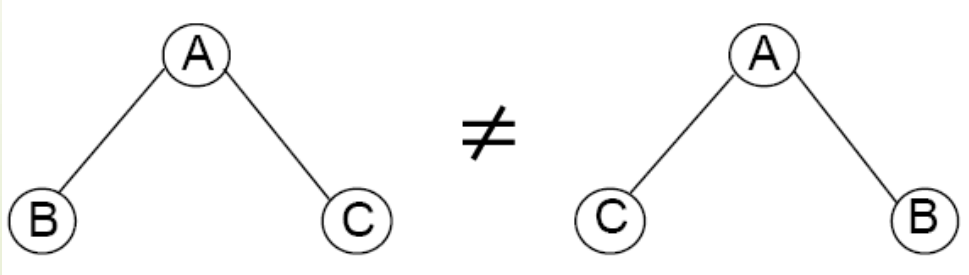
Terminologie (7)

- ***Arbre dégénéré ou filiforme :***
  - **Un arbre dont chaque nœud a au plus au fils**



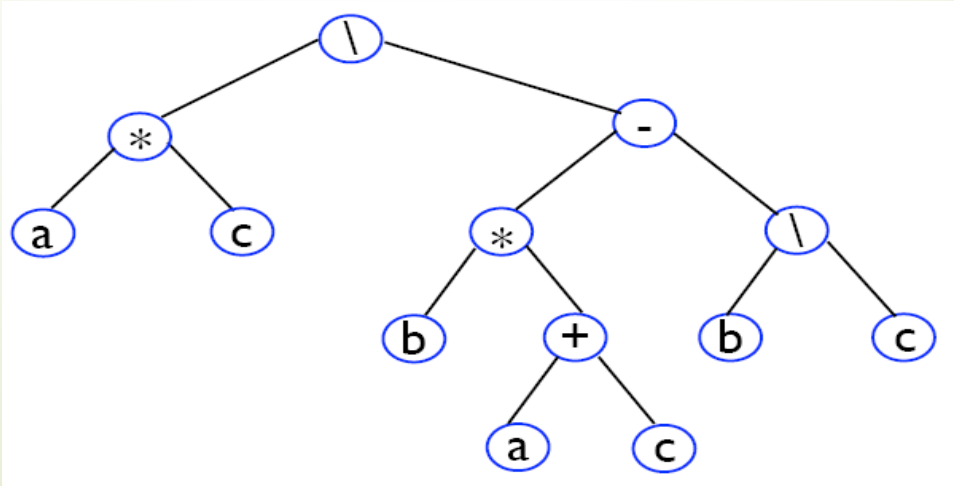
Terminologie (7)

- ***Arbre ordonné :***
  - **Un arbre où la position respective des sous arbres reflète une relation d'ordre. En d'autres termes, si un nœud a k fils, il existe un 1er fis, un 2ème fils, …, et un kème fils.**
  - **Les deux arbres de la figure qui suit sont différents si on les regarde comme des arbres ordonnés, mais identiques si on les regarde comme de simples arbres.**



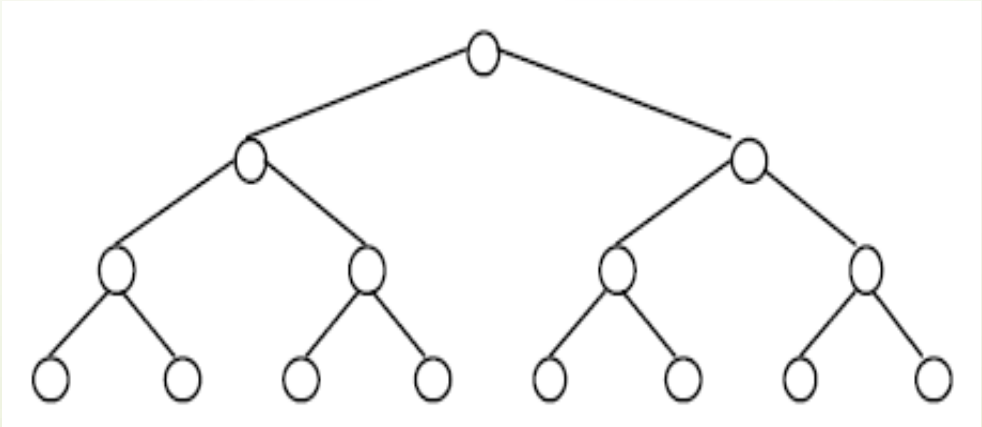
Terminologie (8)

- ***Arbre binaire :***
  - **Un arbre où chaque noeud a au plus deux fils.**
  - **Quand un nœud de cet arbre a un seul fils, on précise s'il s'agit du *fils gauche* ou du *fils droit*.**
  - **La figure qui suit montre un exemple d'arbre binaire dans lequel les nœuds contiennent des caractères.**



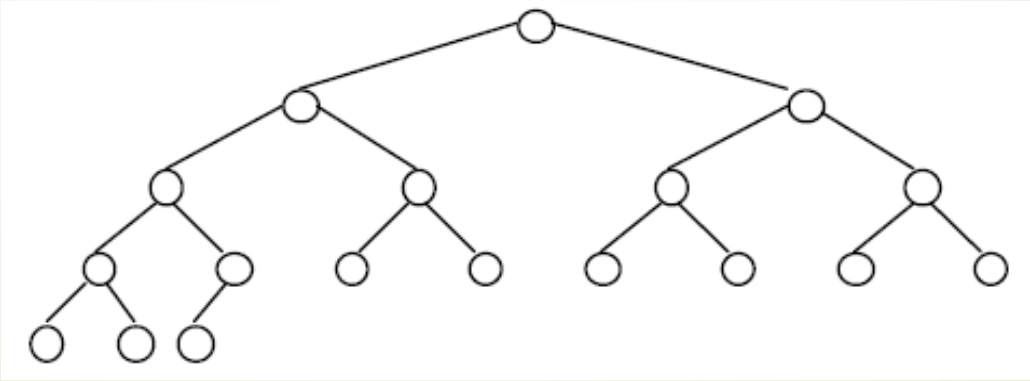
Terminologie (9)

- ***Arbre binaire complet :***
  - **Arbre binaire dont chaque niveau est rempli.**



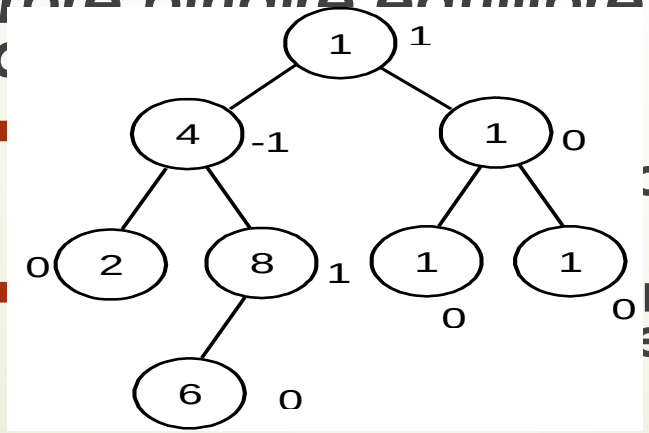
Terminologie (10)

- ***Arbre binaire parfait (ou presque complet) :***
  - **Arbre binaire dont chaque niveau est rempli sauf éventuellement le dernier**
  - **Dans ce cas les nœuds terminaux (*feuilles*) sont groupés le plus à gauche possible.**



Terminologie (11)

- ***Facteur d'équilibre d'un nœud d'un arbre binaire :***
  - **Hauteur du sous arbre partant du fils gauche du nœud moins la hauteur du sous arbre partant de son fils droit.**
- ***Arbre binaire équilibré (au sens des ha***
  - **pour chaque nœud, cteur d'équilibre est**
  - **n place à côté de eur d'équilibre.**

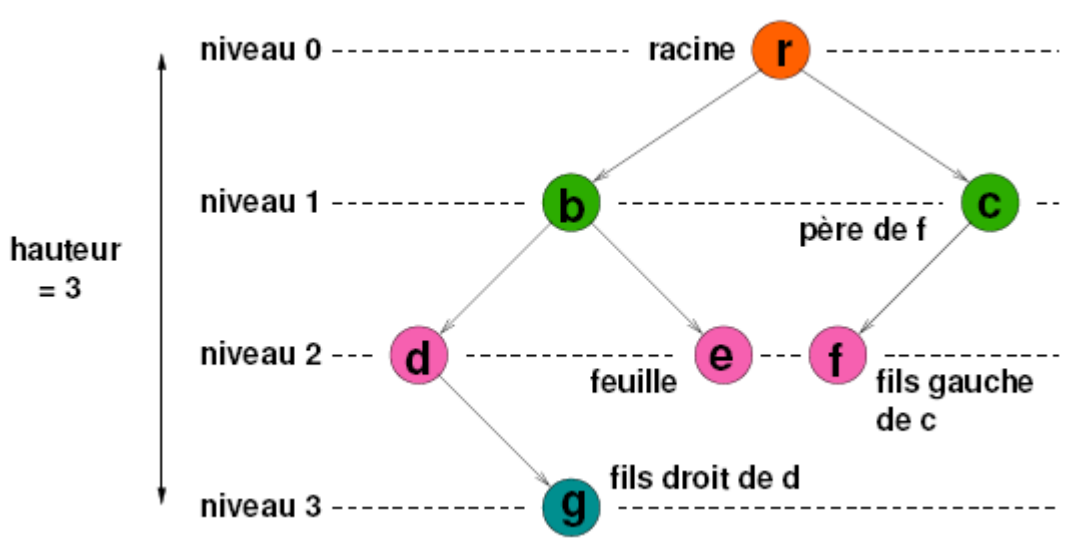


Arbres Binaires (Binary Trees)

Définition

- **Un arbre binaire A** est :
  - soit **vide** (**A = ( )** ou **A = ø**),
  - soit **de la forme A = <r, A1, A2>**, c-à-d composé :
    - d'un nœud **r** appelé ***racine*** contenant un élément
    - et de deux arbres binaires disjoints **A1** et **A2**, appelés respectivement ***sous arbre gauche*** (*ou* ***fils gauche***) et ***sous arbre droit*** (*ou* ***fils droit***).

Exemple d'arbre binaire



Type Abstrait Arbre_Binaire

```
Type Arbre_Binaire
Utilise Noeud, Elément, Booléen
Opérations
  arbre_vide : → Arbre_Binaire
  est_vide   : Arbre_Binaire → Booléen
  cons       : Noeud x Arbre_Binaire x Arbre_Binaire → Arbre_Binaire
  racine     : Arbre_Binaire → Noeud
  gauche     : Arbre_Binaire → Arbre_Binaire
  droite     : Arbre_Binaire → Arbre_Binaire
  contenu    : Noeud → Elément
Préconditions
  racine(A) est-défini-ssi est_vide(A) = faux
  gauche(A) est-défini-ssi est_vide(A) = faux
  droite(A) est-défini-ssi est_vide(A) = faux
Axiomes
  Soit, r : Nœud, A1, A2 : Arbre_Binaire
  racine(<r, A1, A2>) = r
  gauche(<r, A1, A2>) = A1
  droite(<r, A1, A2>) = A2
```

Opérations sur un Arbre Binaire (1)

- **`arbre_vide : → Arbre_Binaire`**
  - opération d'initialisation; crée un arbre binaire vide.
- **`est_vide : Arbre_Binaire → Booléen`**
  - teste si un arbre binaire est vide ou non.
- **`cons : Noeud x Arbre_Binaire x Arbre_Binaire → Arbre_Binaire`**
  - *cons(r,G,D)* construit un arbre binaire dont le sous arbre gauche est *G* et le sous arbre droit est *D*, et *r* est le nœud racine qui contient une donnée de type Elément.
- **`racine : Arbre_Binaire → Noeud`**
  - si `A` est un arbre binaire non vide alors `racine(A)` retourne le nœud racine de `A`, sinon un message d'erreur.

Opérations sur un Arbre Binaire (2)

- **`gauche : Arbre_Binaire → Arbre_Binaire`**
  - si `A` est un arbre binaire non vide alors `gauche(A)` retourne le sous arbre gauche de `A`, sinon un message d'erreur.
- **`droite : Arbre_Binaire → Arbre_Binaire`**
  - si `A` est un arbre binaire non vide alors `droite(A)` retourne le sous arbre droit de `A`, sinon un message d'erreur.
- **`contenu : Noeud → Elément`**
  - permet d'associer à chaque noeud d'un arbre binaire une information de type Elément.

Opérations Auxiliaires

Extension Type Arbre_Binaire

Utilise Entier, Booléen

Opérations

taille : Arbre_Binaire → Entier

hauteur : Arbre_Binaire → Entier

feuille : Arbre_Binaire → Booléen

Préconditions

Axiomes

Soit, r : Noeud, A1, A2 : Arbre_Binaire

taille(arbre_vide) = 0

taille(<r, A1, A2>) = 1 + taille(A1) + taille(A2)

hauteur(arbre_vide) = -1

si hauteur(A1) > hauteur(A2) alors hauteur(<r, A1, A2>) = 1+hauteur(A1)

sinon hauteur(<r, A1, A2>) = 1 + hauteur(A2)

si est_vide(A) = faux et est_vide(gauche(A)) = vrai

et est_vide(droit(A)) = vrai

alors feuille(A) = vrai

sinon feuille(A) = faux

Parcours d'arbre binaire

- **Un parcours d'arbre permet d'accéder à chaque nœud de l'arbre :**
  - **Un traitement (*test, affichage, comptage, etc.*), dépendant de l'application considérée, est effectué sur l'information portée par chaque nœud**
  - **Chaque parcours de l'arbre définit un ordre sur les nœuds**
- **On distingue :**
  - ***Les parcours de gauche à droite (le fils gauche d'un nœud précède le fils droit) ;***
  - ***Les parcours de droite à gauche (le fils droit d'un nœud précède le fils gauche).***
- **On ne considèrera que les parcours de gauche à droite**
- **On distingue aussi deux catégories de parcours d'arbres :**
  - ***Les parcours en profondeur ;***
  - ***Les parcours en largeur.***

Parcours en profondeur

- **Soit un arbre binaire A = <r, A1, A2>**

- **On définit trois parcours en profondeur de cet arbre :**
  - ***Le parcours préfixe ;***
  - ***Le parcours infixe ou symétrique ;***
  - ***Le parcours postfixe ou suffixe.***

Parcours en profondeur
Parcours préfixe

- **En abrégé *RGD* (*Racine, Gauche, Droit*)**
- **Consiste à effectuer dans l'ordre :**
  - ***Le traitement de la racine r ;***
  - ***Le parcours préfixe du sous arbre gauche A1 ;***
  - ***Le parcours préfixe du sous arbre droit A2.***
- ***L'ordre correspondant s'appelle l'ordre préfixe***

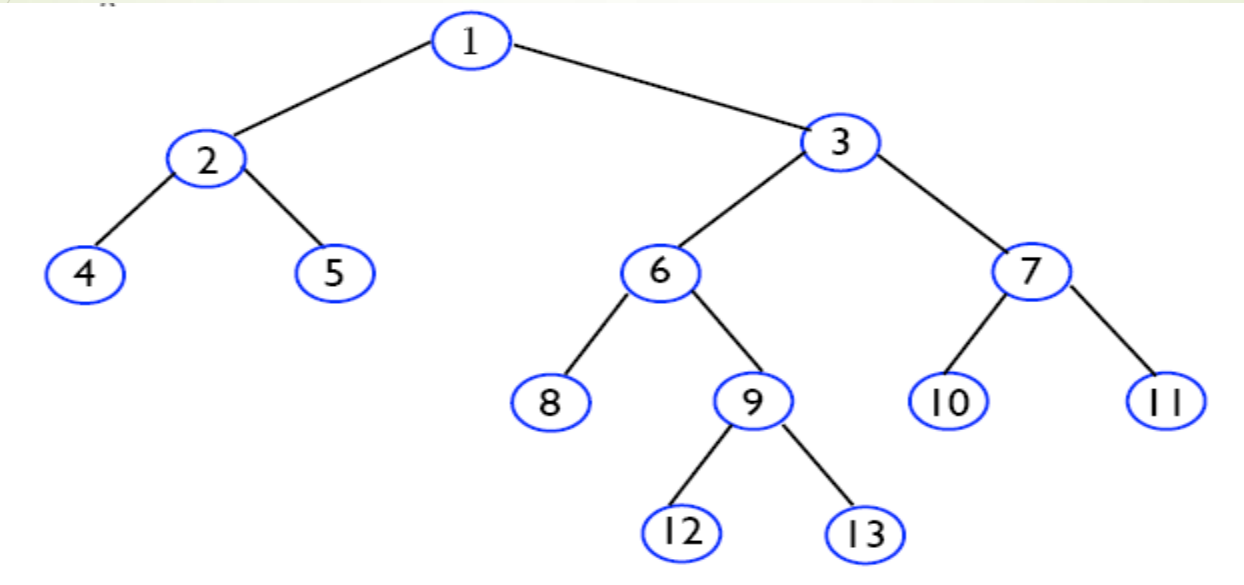
Parcours en profondeur
Parcours infixe ou symétrique

- **En abrégé *GRD* (*Gauche*, *Racine*, *Droit*)**
- **Consiste à effectuer dans l'ordre :**
  - ***Le parcours infixe du sous arbre gauche A1 ;***
  - ***Le traitement de la racine r ;***
  - ***Le parcours infixe du sous arbre droit A2.***
- ***L'ordre correspondant s'appelle l'ordre infixe***

Parcours en profondeur parcours postfixe ou suffixe

- **En abrégé *GDR* (*Gauche, Droit, Racine*)**
- **Consiste à effectuer dans l'ordre :**
  - ***Le parcours postfixe du sous arbre gauche A1 ;***
  - ***Le parcours postfixe du sous arbre droit A2 ;***
  - ***Le traitement de la racine r.***
- ***L'ordre correspondant s'appelle l'ordre suffixe***

Exemple de Parcours en profondeur (affichage du contenu des nœuds)



Le parcours préfixe affiche les nœuds dans l'ordre : ***1, 2, 4, 5, 3, 6, 8, 9, 12, 13, 7, 10, 11***
Le parcours infixe affiche les nœuds dans l'ordre : ***4, 2, 5, 1, 8, 6, 12, 9, 13, 3, 10, 7, 11***
Le parcours postfixe affiche les nœuds dans l'ordre : ***4, 5, 2, 8, 12, 13, 9, 6, 10, 11, 7, 3, 1***

Parcours en largeur

- **On explore les noeuds :**
  - ***niveau par niveau,***
  - ***de gauche à droite,***
  - ***en commençant par la racine.***

- **Exemple :**
  - **Le parcours en largeur de l'arbre de la figure précédente affiche la séquence d'entiers suivante : *1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13***

Représentations d'un arbre binaire

- **Représentation par tableau (*par contiguïté*)**
- **Représentation par pointeurs (*par chaînage*)**

Représentation contiguë d'un arbre binaire

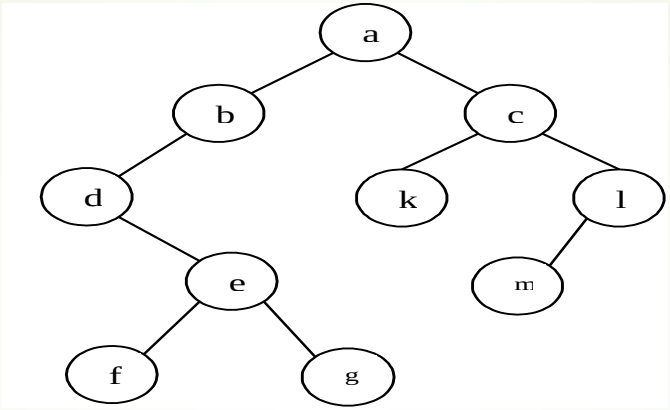
- **On caractérise un arbre binaire par :**
  - **sa taille (*nombre de nœuds*) ;**
  - **sa racine (*indice de son emplacement dans le tableau de nœuds*)**
  - **un tableau de nœuds.**

- **Chaque nœud contient trois données :**
  - **une information de type Elément ;**
  - **deux entiers (*indices dans le tableau désignant respectivement l'emplacement des fils gauche et droit du nœud*).**

Représentation contiguë d'un arbre binaire

```
#define NB_MAX_NOEUDS 15
typedef int Element;
typedef struct noeud {
 Element val;
 int fg;
 int fd;
} Noeud;
typedef Noeud TabN[Nb_MAX_NOEUDS];
typedef struct arbre {
 int nb_noeuds;
 int racine;
 TabN les_noeuds;
} Arbre_Binaire
```

Exemple de Représentation contiguë



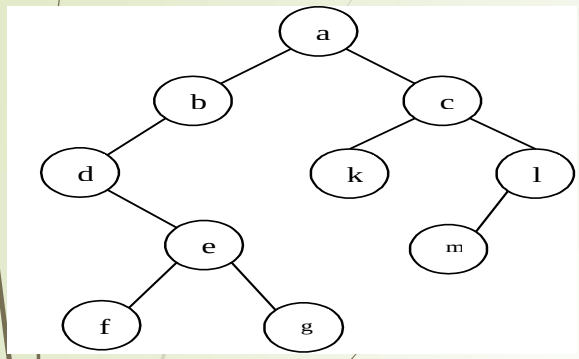
nb_noeuds	racine
10	2

les_noeuds	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
val		d	a	g	b	c		f	m	e	l		k		
fg		-1	4	-1	1	12		-1	-1	7	8		-1		
fd		9	5	-1	-1	10		-1	-1	3	-1		-1		

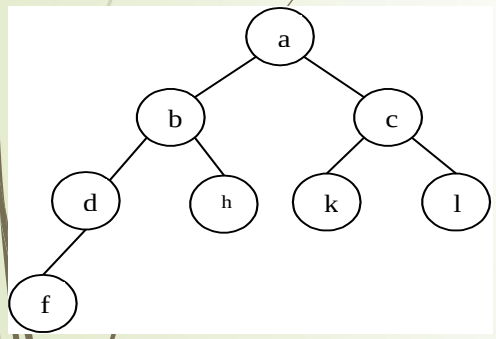
Autre représentation contiguë d'un arbre binaire

- Repose sur l'ordre hiérarchique (*numérotation des nœuds niveau par niveau et de gauche à droite*)
- On rappelle que pour stocker un arbre binaire de hauteur h, il faut un tableau de 2^{h+1}-1 éléments
- On organise le tableau de la façon suivante :
  - Le noeud racine a pour indice 0 (*en langage C*) ;
  - Soit le noeud d'indice i dans le tableau, son fils gauche a pour indice 2i +1, et son fils droit a pour indice 2(i+1).
- Représentation idéale pour les arbres binaires parfaits. En effet, elle ne gaspille pas d'espace.

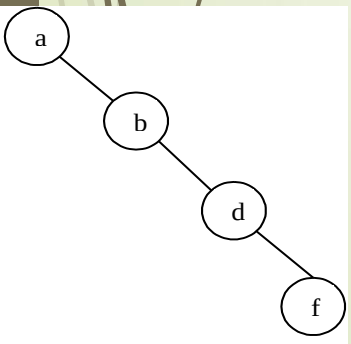
Autre représentation contiguë d'un arbre binaire (Exemples)



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
a	b	c	d		k	l		e					m				f	g



0	1	2	3	4	5	6	7
a	b	c	d	h	k	l	f



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a		b				d								f

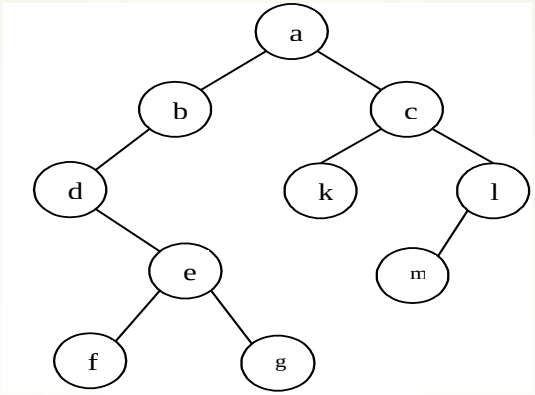
Représentation chaînée d'un arbre binaire

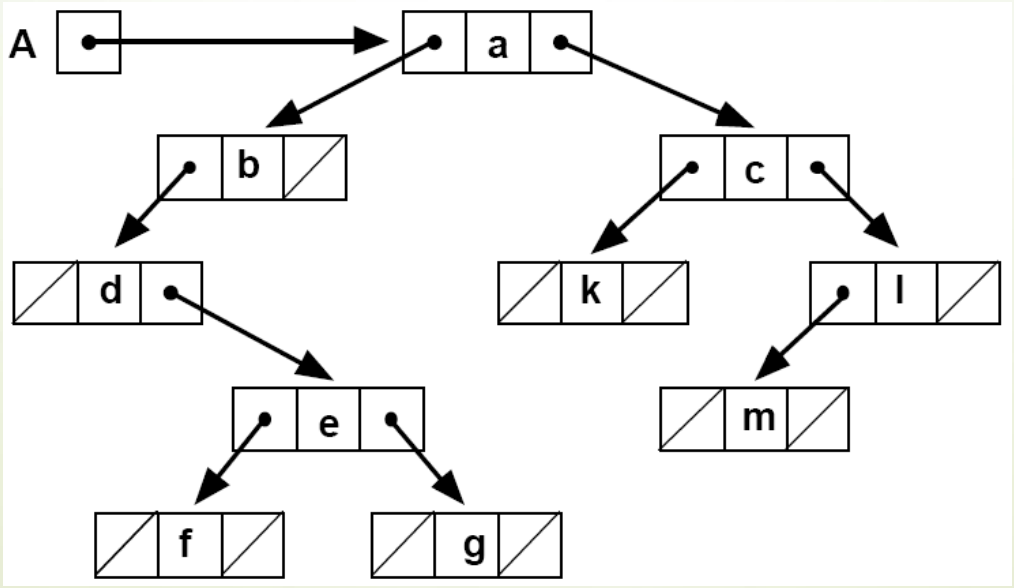
- **Chaque nœud a trois champs :**
  - **val (*l'élément stocké dans le noeud*) ;**
  - ***fg* (*pointeur sur fils gauche*) ;**
  - ***fd* (*pointeur sur fils droit*).**
- **Un arbre est désigné par un pointeur sur sa racine**
- **Un arbre vide est représenté par le pointeur NULL**

Représentation chaînée en C d'un arbre binaire


```
typedef int Element;
typedef struct noeud *Pnoeud;
typedef struct noeud {
 Element val;
 Pnoeud fg;
 Pnoeud fd;
} Noeud;
typedef Noeud *Arbre_Binaire;
```

Exemple de Représentation chaînée d'un arbre binaire





Réalisation chaînée d'un arbre binaire


```
Arbre_Binaire arbre_vide() {
 return NULL;
}

Booleen est_vide(Arbre_Binaire A) {
 return A == NULL ;
}

Pnoeud nouveau_noeud(Element e) {
// faire une allocation mémoire et placer l'élément e
// en cas d'erreur d'allocation, le pointeur renvoyé est
   NULL
 Pnoeud p = (Pnoeud) malloc(sizeof(Noeud));
 if (p != NULL) {
  p->val = e;
  p->fg = NULL;
  p->fd = NULL;
 }
 return (p);
}
```

Réalisation chaînée d'un arbre binaire


```
Arbre_Binaire cons(Noeud *r,
                   Arbre_Binaire G,
                       Arbre_Binaire D) {
 r->fg = G ;
 r->fd = D ;
 return r ;
}


Noeud racine(Arbre_Binaire A) {
// précondition : A est non vide !
 if (estvide(A)) {
  printf("Erreur : Arbre vide !\n");
  exit(-1);
 }
 return (*A) ;
}
```

```
Arbre_Binaire gauche(Arbre_Binaire A) {
// précondition : A est non vide !
 if (estvide(A)) {
  printf("Erreur : Arbre vide !\n");
  exit(-1);
 }
 return A->fg ; /* ou bien (*A).fg; */
}

Arbre_Binaire droite(Arbre_Binaire A) {
// précondition : A est non vide !
 if (estvide(A)) {
  printf("Erreur : Arbre vide !\n");
  exit(-1);
 }
 return A->fd ; /* ou bien (*A).fd; */
}

Element contenu(Noeud n) {
 return n.val;
}
```

Exemples d'Applications d'Arbre Binaire

- ***Recherche dans un ensemble de valeurs*** :
  - Les arbres binaires de recherche ;
- ***Tri d'un ensemble de valeurs :***
  - Le parcours GRD d'un arbre binaire de recherche ;
  - Un algorithme de tri efficace utilisant une structure de *tas ;*
- ***Représentation d'une expression arithmétique*** :
  - Un parcours GDR pour avoir une notation postfixée ;
- ***Méthodes de compression :***
  - Le codage de *Huffman u*tilisant des arbres binaires ;
  - La compression d'images utilisant des *quadtrees (arbres quaternaires, où chaque nœud non feuille a exactement quatre fils) ;*
- …

Arbres de Recherche Equilibrés Exemples (3)

- ***Les B arbres :***
  - ***Arbres de recherche équilibrés qui sont conçus pour être efficaces sur d'énormes masses de données stockées sur mémoires secondaires ;***
  - ***Chaque nœud permet de stocker plusieurs clés ;***
  - ***Généralement, la taille d'un nœud est optimisée pour coïncider avec la taille d'un bloc (ou page) du périphérique, en vue d'économiser les coûteux accès d'entées sorties.***
- **…**