CHAPITRE 3 : LES ARBRES BINAIRES

Les arbres sont très utilisés en informatique, d'une part parce que les informations sont souvent hiérarchisées, et peuvent être représentées naturellement sous une forme arborescente, et d'autre part, parce que les structures de données arborescentes permettent de stocker des données volumineuses de façon que leur accès soit efficace.

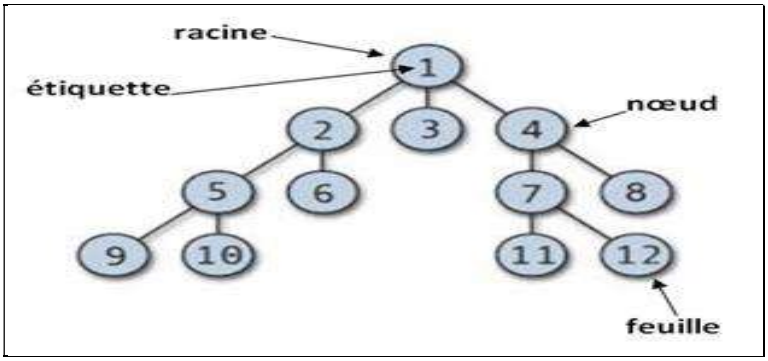
I. Notions générales sur les arbres :

I-1. Définition :

Un arbre est un ensemble organisé de nœuds dans lequel chaque **nœud** a un père et un seul, sauf un nœud que l'on appelle la **racine**.

Si le nœud **p** est le père du nœud **f**, nous dirons que f est un fils de p, et si le nœud p n'a pas de fils nous dirons que c'est une **feuille**.

Chaque nœud porte une valeur(ou étiquette ou clé). On a l'habitude, lorsqu'on dessine un arbre, de le représenter comme le schéma ci-dessous :



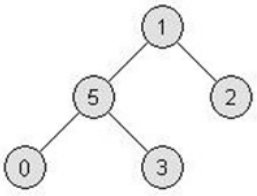
I-2. Arbre binaire :

Dans un arbre binaire, chaque nœud possède **au plus** deux nœuds fils au niveau inférieur, habituellement appelés **fils gauche** et **fils droit**.

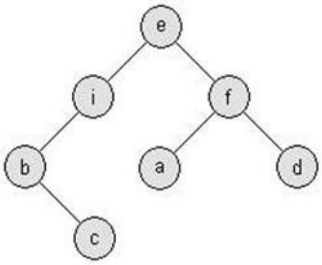
Exemple :

Arbre binaire d'un seul nœud

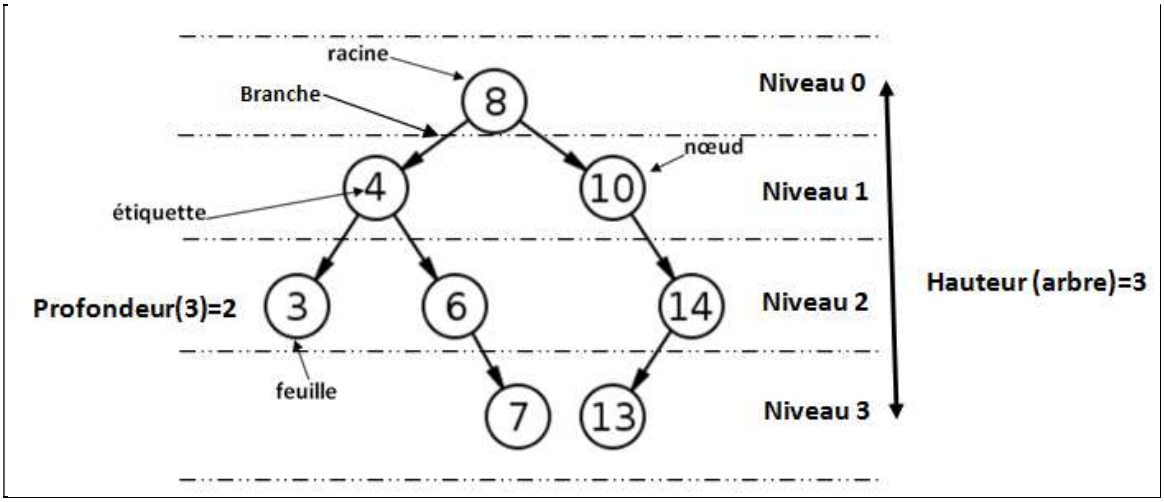
Arbre binaire de 5 nœuds



Arbre binaire de 7 nœuds



I-3. La terminologie de base sur les arbres :



- **les descendants** : d'un nœud p sont les nœuds qui apparaissent dans ses sous-arbres
- **un ancêtre** : d'un noeud p est soit son père, soit un ancêtre de son père,
- **un frère** : d'un noeud p est un fils du père de p, et qui n'est pas p.
- **Taille d'un arbre** : nombre de noeuds dans l'arbre
- **le chemin** : On appelle chemin du noeud p , la suite des nœuds par lesquels il faut passer pour aller de la racine vers le noeud p :

  Exemple :
  Chemin du nœud(6) = (8,4,6)
  Chemin du noeud (13) = (8,10,14,13)
- **Niveau d'un nœud :** Les nœuds d'un arbre se répartissent par niveaux : le premier niveau (par convention ce sera le niveau 0) contient la racine seulement, le deuxième niveau contient les deux fils de la racine,. . . , les noeuds du niveau k sont les fils des noeuds du niveau k – 1,. . . .
- **Profondeur d'un nœud :** la profondeur d'un nœud p est égale au nombre de branches à partir de la racine pour aller jusqu'au nœud p.

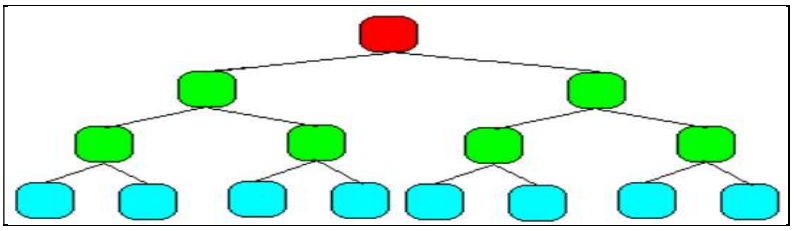
  Exemple : Profondeur du nœud (13) = 3
- **Hauteur d'un arbre :** Distance maximum (nombre de branches) pour aller de la racine à une des feuilles de l'arbre.
  Exemple : Hauteur de l'arbre d l'exemple = 3

  Attention, la définition de la hauteur varie en fonction des auteurs. Pour certains la hauteur d'un arbre contenant un seul nœud est 1.

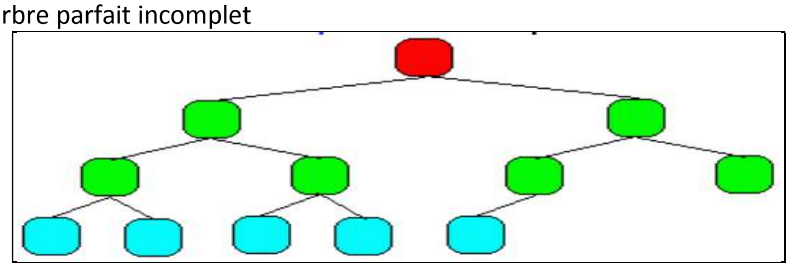
- **Degré d'un nœud :** Par définition le degré d'un nœud est égal au nombre de ses descendants (enfants).
  Dans l'arbre binaire : $0 \leq$ le degré d'un nœud ≤ 2

- **Arbre parfait** : c'est un arbre binaire dont tous les nœuds de chaque niveau sont présents sauf éventuellement au dernier niveau où il peut manquer des nœuds (nœuds terminaux = feuilles), dans ce cas l'arbre parfait est un arbre binaire incomplet et les feuilles du dernier niveau **doivent être regroupées à partir de la gauche** de l'arbre.

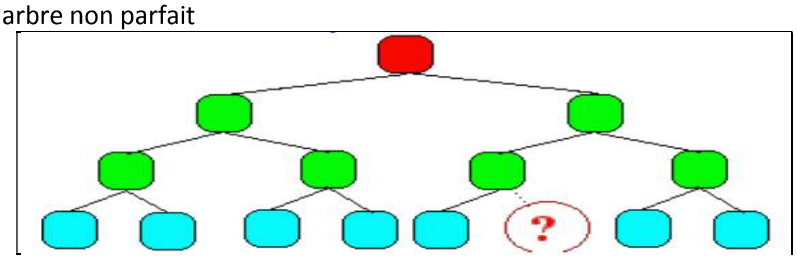
  Exemple : arbre parfait complet



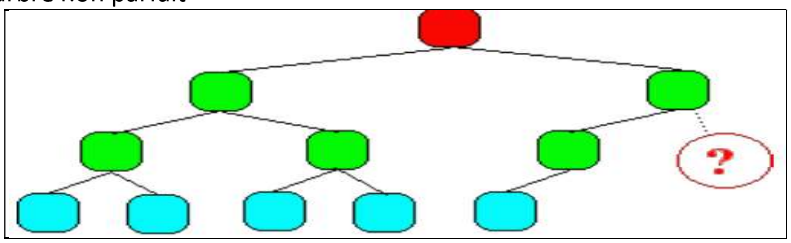
Exemple : arbre parfait incomplet



Exemple 1:arbre non parfait



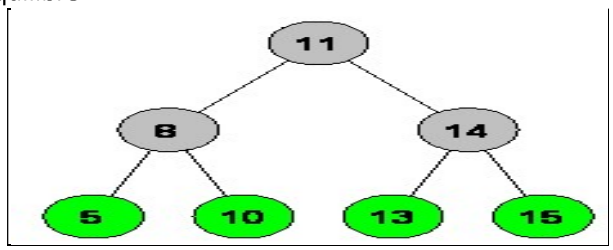
Exemple 2:arbre non parfait



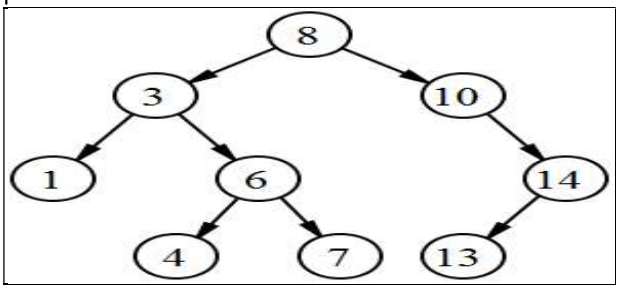
Rq : Un arbre binaire parfait se représente classiquement dans un tableau (une seule liste)

- **Arbre équilibré** : On dit qu'un arbre binaire A est équilibré si tous les chemins menant de la racine à une feuille ont pour longueur **Hauteur(A)** ou **Hauteur(A)-1**

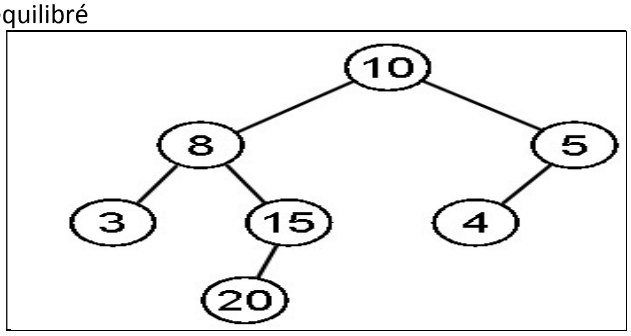
Exemple 1: arbre équilibré



Exemple 2: arbre équilibré



Exemple 3: arbre équilibré



II. Représentation d'un arbre en Python:

Il existe différentes manières de représenter un arbre binaire en Python. A l'aide :

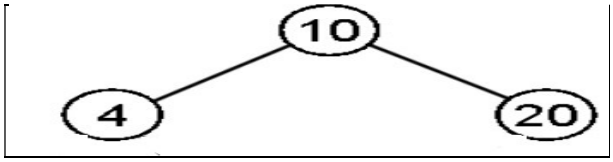
- d'une liste de listes
- d'un tableau (une lise)

II-1. arbre sous forme d'une liste de listes :

Arbre=[étiquette,fils_Gauche, fils_Droit]

- fils_Gauche est un sous arbre : **fils_Gauche =[étiquette, fils_Gauche, fils_Droit]**
- fils_Droit est un sous arbre : **fils_Gauche =[étiquette, fils_Gauche, fils_Droit]**
- Arbre vide : **arbre=[]**
- **Feuille =[étiquette, [], []]**

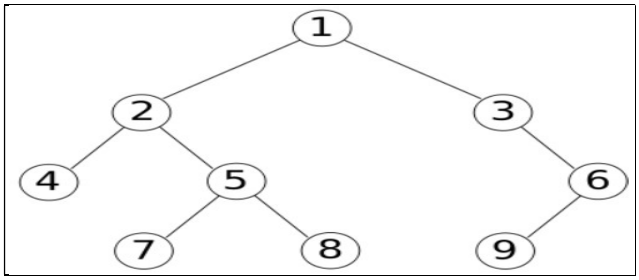
Exemple 1:



L'arbre suivant sera représenté en Python sous forme :

A=[10,[4, [ ], [ ]],[20,[ ],[ ]]]

Exemple 2:

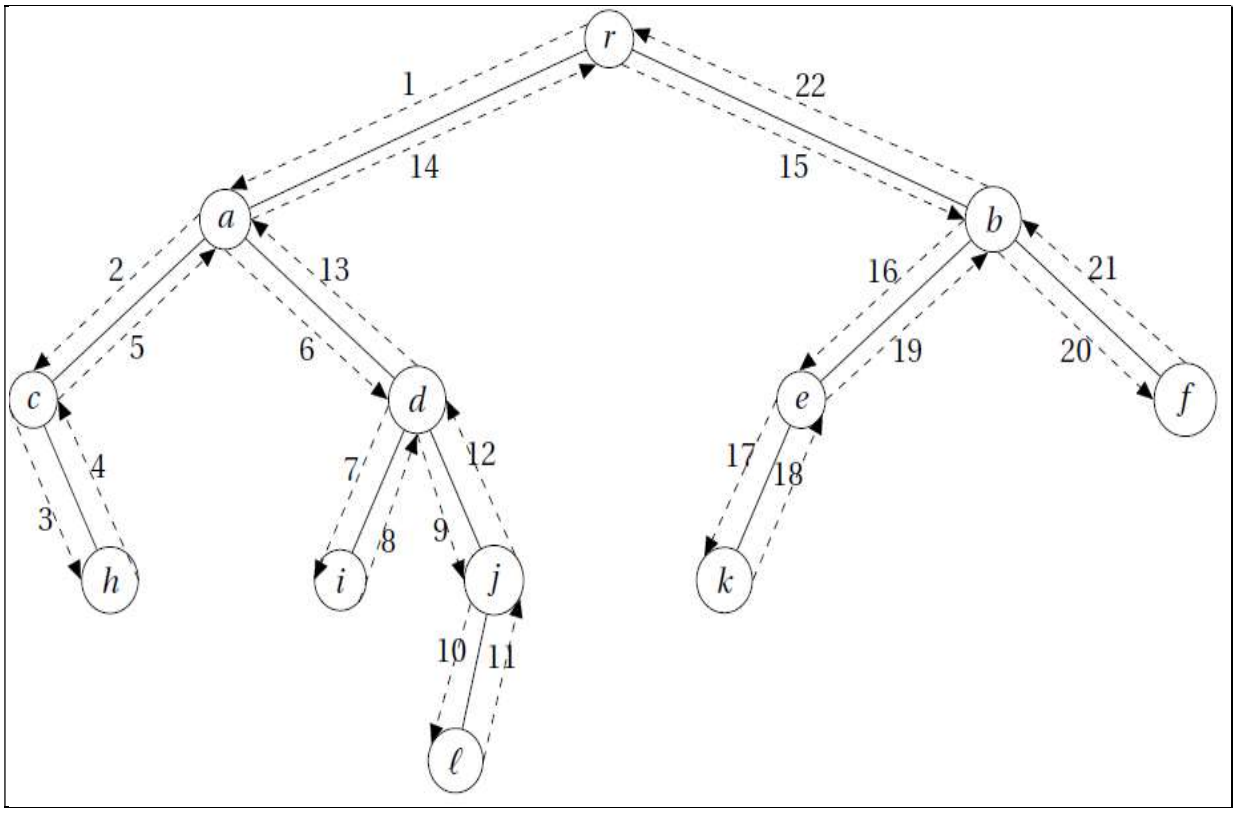


L'arbre suivant sera représenté en Python sous forme :

A=[1,[2,[4, [ ], [ ]],[5,[7, [ ], [ ]] ,[8, [ ], [ ]]]],[3, [ ],[6,[9, [ ], [ ]], [ ]]]]

Parcours d'un arbre binaire :

On se balade autour de l'arbre en suivant les pointillés dans l'ordre des numéros indiqués :



A partir de ce contour, on définit trois parcours des sommets de l'arbre :

- **l'ordre préfixe** : on liste chaque sommet la première fois qu'on le rencontre dans la balade. Ce qui donne ici : **r,a,c,h,d,i,j,l,b,e,k, f**
- **l'ordre postfixe** : on liste chaque sommet la dernière fois qu'on le rencontre. Ce qui donne ici : **h,c, i ,l, j ,d,a,k,e, f ,b, r**
- **l'ordre infixe** : on liste chaque sommet ayant un fils gauche la seconde fois qu'on le voit et chaque sommet sans fils gauche la première fois qu'on le voit. Ce qui donne ici : **c,h,a,i,d,l,j,r,k,e,b,f**

II-2. arbre sous forme d'un tableau :

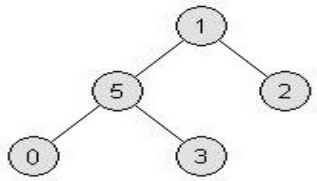
On peut convenir de stocker les étiquettes dans un tableau L, en numérotant les nœuds de haut en bas et de gauche à droite pour chaque niveau, L[k] contenant l'étiquette du nœud n° k s'il existe, sinon (valeur absente de l'ensemble des étiquettes possibles, par exemple :[])
Donc :

- La première case (indice 0) de la liste constitue la racine de l'arbre,
- Pour toute case portant l'indice i, son fils gauche sera situé à l'indice 2i+1et son fils droit à l'indice 2i+2
- Une case avec la valeur **[]** signifie un arbre vide (l'absence de nœud).

Cette représentation est avantageuse dans le cas des arbres **parfaits** (on stocke les n nœuds dans L[0], L[1],…,L[n-1]). Réciproquement on stocke n valeurs dans un arbre de hauteur $\lfloor \log_2 n \rfloor$

Voici quelques exemples :

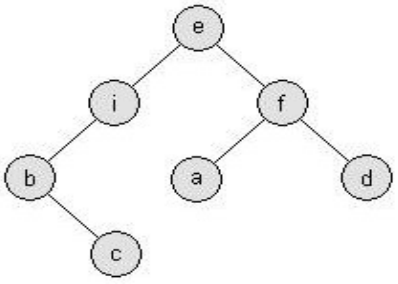
Exemple 1:



En Python :

0	1	2	3	4
1	**5**	**2**	**0**	**3**

Exemple 2:



En Python :

0	1	2	3	4	5	6	7	8
'e'	**'i'**	**'f'**	**'b'**	**[]**	**'a'**	**'d'**	**[]**	**'c'**

Toutefois elle a de gros défauts dans le cas général : dans le pire des cas, on peut avoir besoin d'un tableau de taille $\mathbf{2^n}$ pour stocker n nœuds ! De plus, la structure récursive de l'arbre binaire n'apparaît pas clairement