T NSI Les Arbres

Table des matières

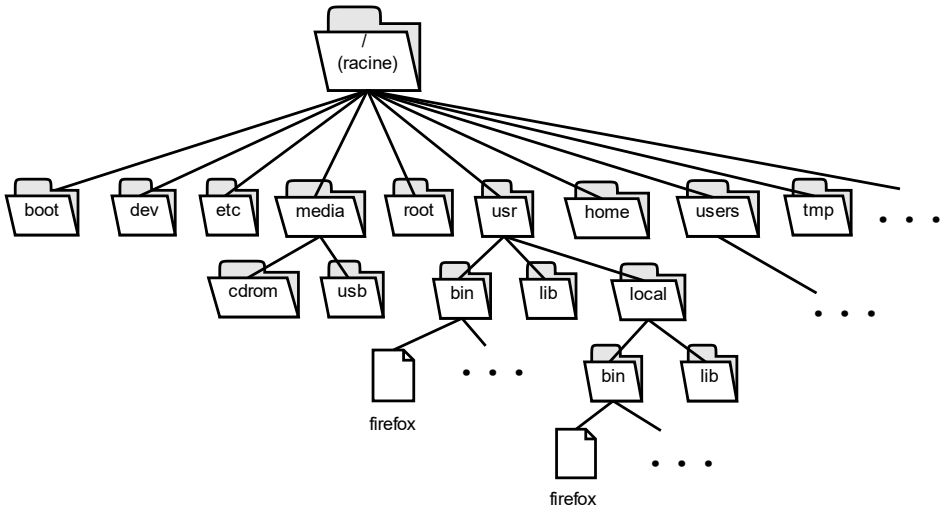
1. Définition

Un arbre est une structure de données

- hiérarchique
- composée de nœuds et de relations de précédence entre ces nœuds (sous forme récursive).

1.1.Exemples :

1.1.1. Arborescence des fichiers et dossiers dans les systèmes de fichiers des OS



T NSI Les Arbres

1.1.2. Standards de données structurées : XML, HTML etc …

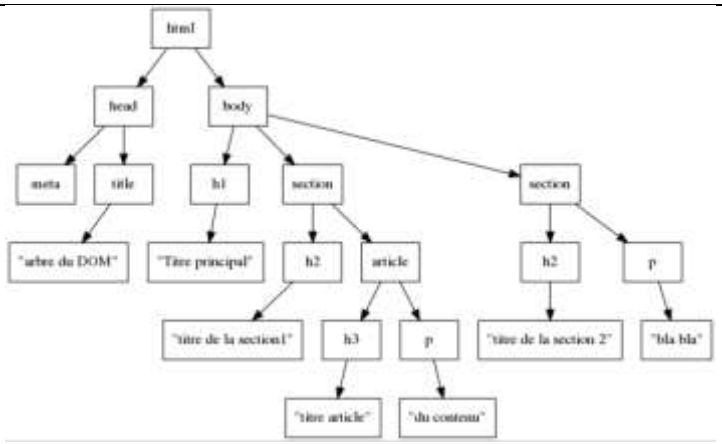
Code HTML (ou plus généralement code XML)	Arbre associé du DOM : **Document Object Model interface de programmation** normalisée par le W3C, qui permet à des scripts d'examiner et de modifier le contenu du navigateur web

```
<!DOCTYPE html>
<html lang="fr">
    <head>
        <meta charset="utf-8">
        <title> arbre du DOM  </title>
    </head>

    <body>
        <h1> Titre principal </h1>

        <section>
            <h2> titre de la section 1 </h2>
            <article>
                <h3> titre article </h3>
                <p> du contenu </p>
            </article>
        </section>

        <section>
            <h2> titre de la section 2 </h2>
            <p> bla bla </p>
        </section>
    </body>
</html>
```

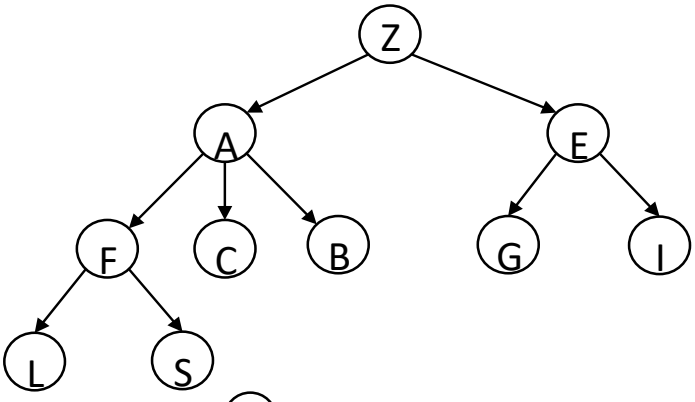


On associe à cette arborescence le vocabulaire usuel sur les arborescences :

- le nœud <html> est le **nœud racine**
- le nœud <html> a deux **fils** : les nœuds <head> et <body>
- les nœuds sans descendants sont les **feuilles**

1.2.Vocabulaire :

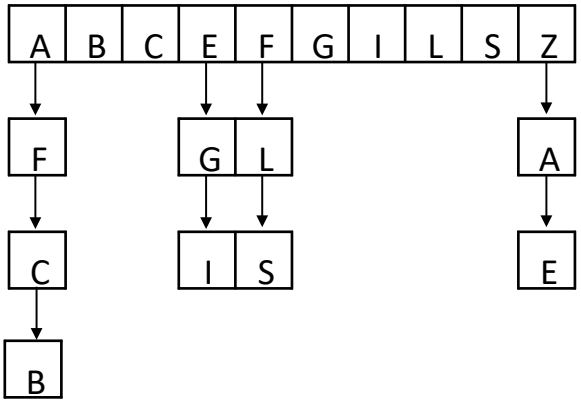
- La racine est le nœud sans prédécesseur (point d'accès au contenu de l'arbre entier)
- Une feuille est un nœud sans successeur
- Une branche est la suite des nœuds liant la racine à une feuille
- Un fils est un successeur d'un nœud
- Le père est le prédécesseur d'un nœud
- Le degré d'un nœud est le nombre de fils de ce nœud
- La profondeur d'un nœud est le nombre de prédécesseur entre ce nœud et la racine
- La hauteur d'un arbre est la profondeur maximale de tous les nœuds + 1
- La taille d'un arbre est le nombre de nœuds

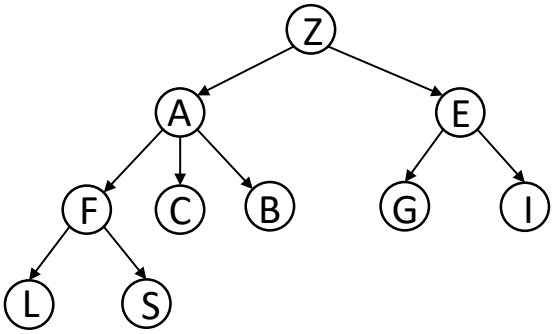


- Racine (arbre) = Z
- Feuilles (arbre) = { L S C B G I }
- Branche (S) = { S F A Z }
- Fils (A) = {F C B }
- Père (F) = A
- Degré (A) = 3
- Profondeur(C) = 2
- Hauteur (arbre) = 3+1 =4

1.3.Représentation d'un arbre

- Par liste d'adjacence





Liste de listes mettant en relation chaque père et ses fils

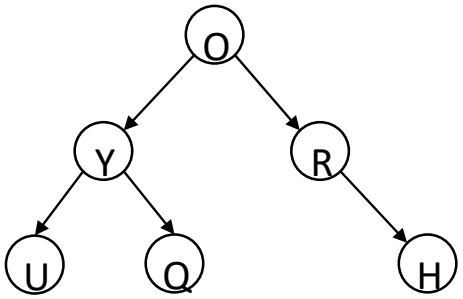
- Par tableau 2D

	A	B	C	E	F	G	I	L	S	Z
A	0	1	1	0	1	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	1	1	0	0	0
F	0	0	0	0	0	0	0	1	1	0
G	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0
Z	1	0	0	1	0	0	0	0	0	0

Le nœud de la ligne a comme fils le nœud de la colonne

- Par structure récursive voir la suite Arbre Binaire de Recherche

2. Les arbres binaires



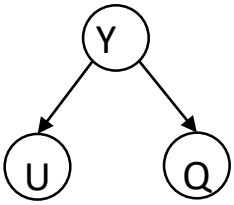
2.1. Définition

Un arbre binaire est un arbre qui a au plus 2 fils (i.e. 0, 1 ou 2)

Le degré maximal d'un nœud est 2

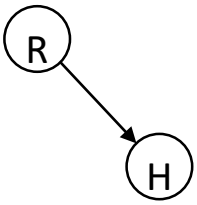
On appelle fils gauche (ou sous arbre gauche ou sag) le premier successeur

Fils gauche (O) =

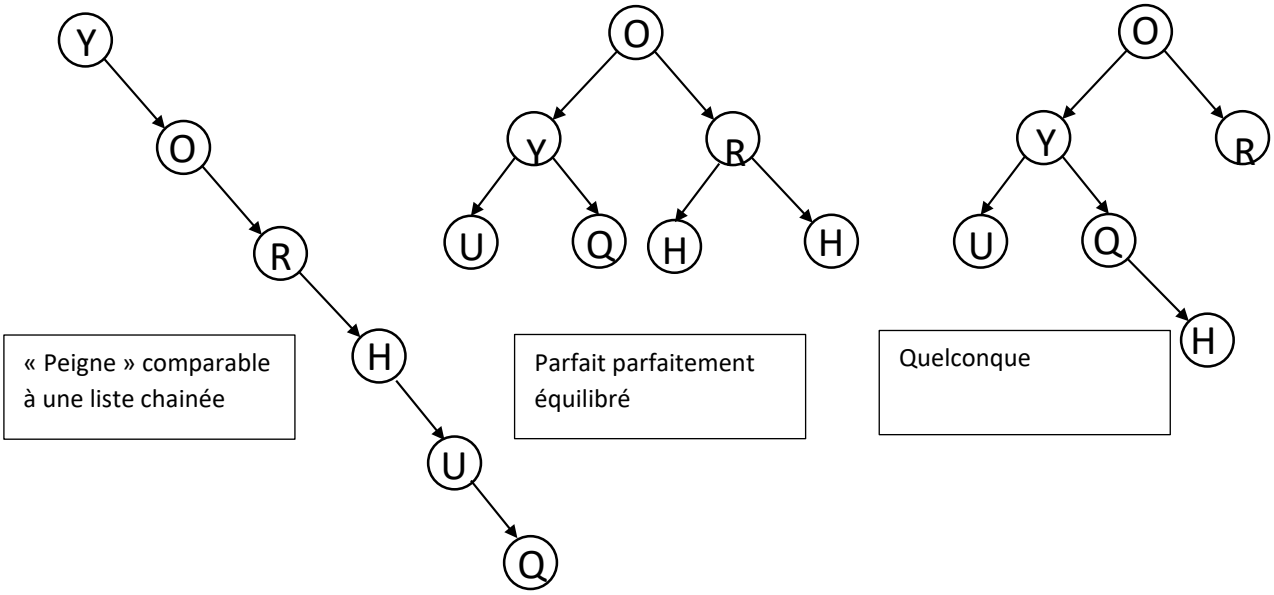


On appelle fils droit (ou sous arbre droit ou sad) le deuxième successeur

Fils droit (O) =



Un arbre binaire peut être dégénéré ou équilibré ou aucun des deux



Pour un arbre binaire le nombre de nœuds en fonction de la hauteur est $h \leq N \leq 2^h - 1$

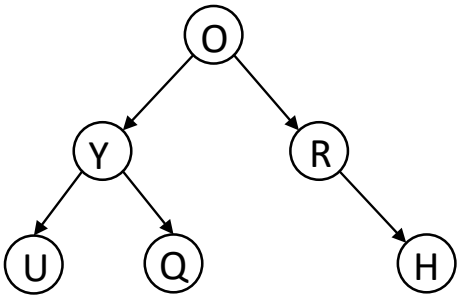
Ex arbre peigne N = 5+1=6 arbre parfait N = 7 = 2^3 -1

2.2.Les différents parcours

On a plusieurs façons de visiter tous les nœuds d'un arbre, donnant des ordres de visite différents

- Parcours en profondeur
  - Parcours en ordre (infixe) : fils gauche, nœud, fils droit
  - Parcours en pré-ordre (préfixe) : nœud, fils gauche, fils droit
  - Parcours en post-ordre (postfixe) : fils gauche, fils droit, nœud
- Parcours en largeur
  - Parcours niveau après niveau (i.e. profondeur par profondeur)

Exemple :



- Parcours en ordre (infixe) : U Y Q O R H
- Parcours en pré-ordre (préfixe) : O Y U Q R H
- Parcours en post-ordre (postfixe) : U Q Y H R O
- Parcours en largeur : O Y R U Q H

3. Implémentation d'un arbre (binaire) en Python

3.1.La classe Arbre

Un arbre vide est représenté par l'attribut info à None

Les attributs fg et fd sont des arbres donc tous les nœuds sont des arbres, une feuille a deux attributs fg et fd à None

```
class Arbre :
    def __init__(self,info=None,fg=None,fd=None) :
        self.info = info
        self.fg = fg
        self.fd = fd

a1 = Arbre()      # arbre vide
a2 = Arbre(2)     # arbre avec un nœud (la racine)
```

3.2.Différentes méthodes et fonctions

3.2.1. Méthode pour ajouter un fils gauche

```
def insert_gauche(self, valeur):
      if self.fg == None:
         self.fg = Arbre(valeur)
      else:
         new_node = Arbre(valeur)
         new_node.fg = self.fg
         self.fg = new_node
```

3.2.2. Méthode pour ajouter un fils gauche

```
def insert_droite(self, valeur):
      if self.fd == None:
         self.fd = Arbre(valeur)
      else:
         new_node = Arbre(valeur)
         new_node.fd = self.fd
         self.fd = new_node
```

3.2.3. Fonction pour obtenir la hauteur de l'arbre

```
def hauteur(A):
       if A == None : return 0
       else:
              return 1+ max(hauteur(A.fg),hauteur(A.fd))
```

3.2.4. Fonction pour effectuer un parcours infixe

```
def parcours_infixe(A):
       if A is None : return
       parcours_infixe(A.fg)
       print(A.info)
       parcours_infixe(A.fd)
```

3.2.5. Fonction pour effectuer un parcours infixe

```
def parcours_preordre(A):
      if A is None : return
      print(A.info)
      parcours_preordre(A.fg)
      parcours_preordre(A.fd
```