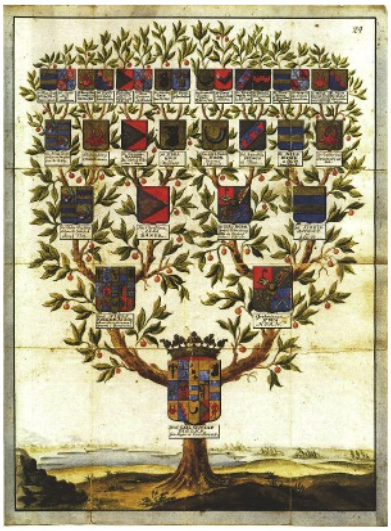
CHAPITRE 6.1 : LES ARBRES

1. VOCABULAIRE



Définitions :

Un **arbre** est une structure de données constituée de **nœuds**, qui peuvent avoir des **enfants** (qui sont eux-mêmes des nœuds). Cette structure est hiérarchisée et le sommet de l'arbre est appelé **racine**.

Un **nœud** est une position dans un arbre. À chaque nœud correspond un sous-arbre. Chaque nœud est défini par son **étiquette**.

Une **feuille** est un nœud qui n'a pas d'enfant.

Le sommet d'un arbre est appelé la **racine**. C'est le seul nœud qui n'a pas de parents.

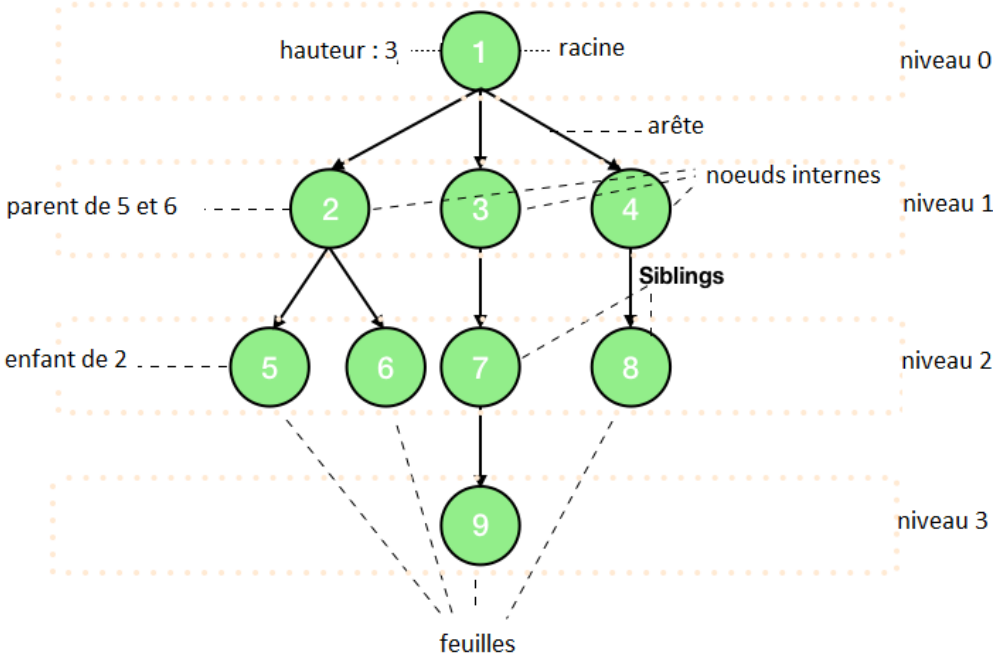
Un **nœud interne** est un nœud autre que la racine et qui n'est pas une feuille.

Une **branche** est une suite finie de nœuds consécutifs de la racine vers une feuille.

La **taille** d'un arbre est le nombre de ses nœuds.

La **profondeur** d'un nœud est la longueur du chemin qui le relie à la racine.

La **hauteur** d'un arbre est la profond de son nœud le plus profond (c'est donc nécessairement une feuille).



Exercices d'application 1 et 2

2. PARCOURIR UN ARBRE

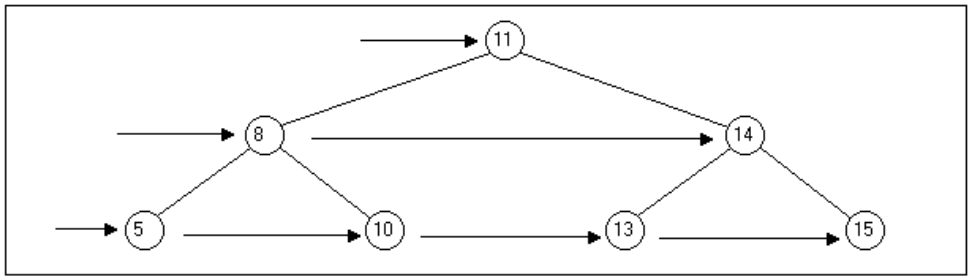
Parcourir un arbre c'est partir d'un nœud et visiter tous les nœuds de l'arbre une seule fois. Ce concept de parcours est très important en algorithmique.
Les parcours permettent notamment de générer une autre représentation de l'arbre (par exemple une liste) ou d'effectuer une recherche dans une structure arborescente.
Il existe différentes façons de parcourir un arbre.

Parcours en largeur (BFS : Breadth First Search)

Dans un parcours en largeur, les nœuds sont parcourus par profondeur croissante. Avec l'exemple donné ci-dessous, les étiquettes seront traitées dans l'ordre suivant :

11 > 8 > 14 > 5 > 10 > 13 > 15

Plus précisément, on commence par explorer un nœud source, puis ses enfants, puis les enfants non explorés des enfants, etc.

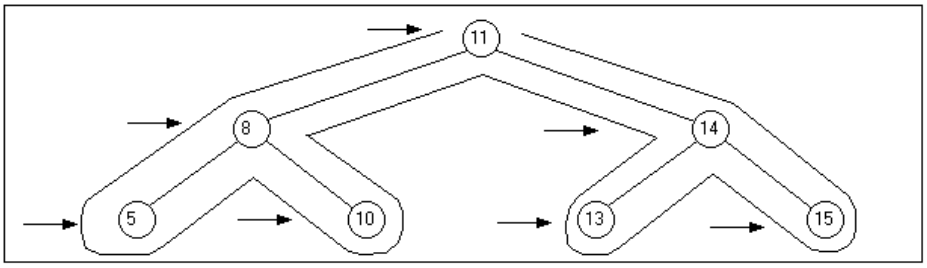


Parcours en profondeur (DFS : Depth First Search)

Préfixe

Un parcours en profondeur **préfixe** est un parcours où l'on effectue le traitement de chaque nœud avant d'explorer le sous-arbre correspondant. Avec l'exemple donné ci-contre, les étiquettes seront visitées dans l'ordre suivant :

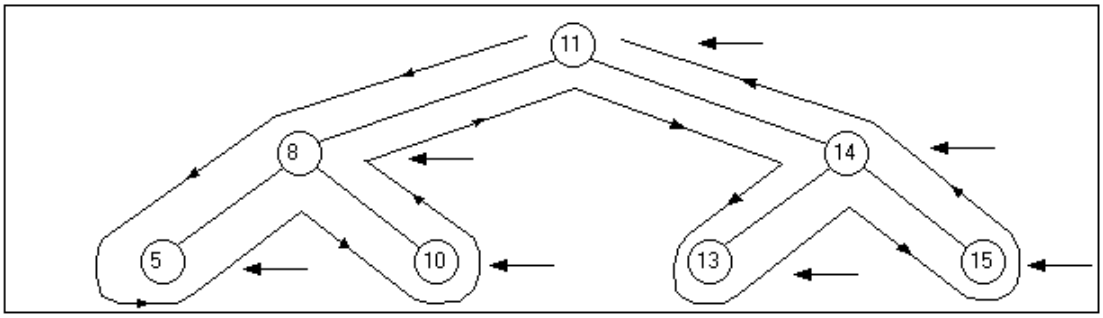
11 > 8 > 5 > 10 > 14 > 13 > 15



Postfixe

- Un parcours en profondeur **postfixe** est un parcours où l'on effectue le traitement de chaque nœud après avoir exploré le sous-arbre correspondant. Avec le même exemple donné ci-dessous, les étiquettes seront visitées dans l'ordre suivant :

5 > 10 > 8 > 13 > 15 > 14 > 11



On peut implémenter facilement les parcours en profondeur de manière récursive :

```
ParcoursPréfixe(arbre A):
    Afficher la racine de A
    Pour tous les enfants e de la racine :
        ParcoursPréfixe( Sous-Arbre issu de e )
```

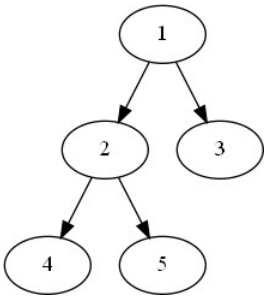
```
ParcoursPostfixe(arbre A):
    Pour tous les enfants e de la racine :
        ParcoursPostfixe( Sous-Arbre issu e )
    Afficher la racine de A
```

Exercices d'applications 3 et 4

3. UN PYTHON DANS UN ARBRE

Voici un exemple de code permettant de construire l'arbre a1 ci-contre :

```
a1 = Noeud(1)
b = Noeud (2)
c = Noeud (3)
d = Noeud (4)
a1.ajoute(b, c)
b.ajoute(d, Noeud (5))
```



On utilise la classe Nœud. La création d'un nœud lui donne pour attribut son étiquette. On définit les enfants d'un nœud grâce à la méthode .ajoute().

Exercices d'applications 5 à 8