# TP : Parcourir un arbre binaire

## Introduction

Un arbre binaire peut être parcouru afin de connaître les nœuds qu'il contient. Il est possible de parcourir cet arbre verticalement, appelé **parcours en profondeur** ou horizontalement, appelé **parcours en largeur**.

Il y a trois façons distinctes de parcourir un arbre en profondeur.

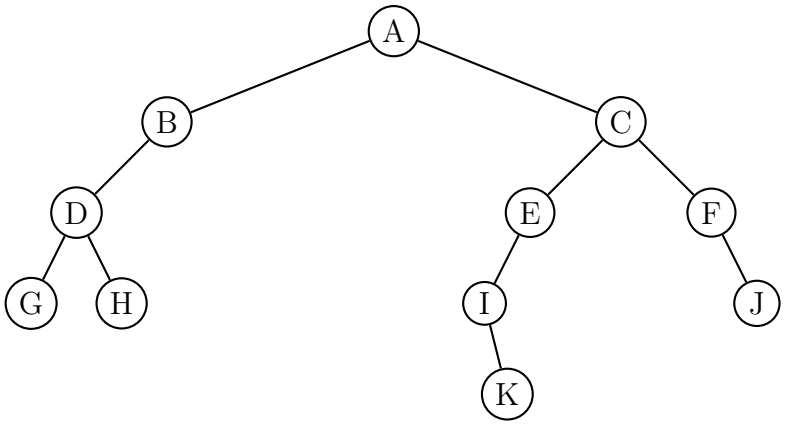
— parcours **infixe** : on parcourt l'arbre gauche, on note la racine puis on parcourt l'arbre droit.

— parcours **préfixe** : on note la racine de l'arbre, ensuite on parcourt l'arbre gauche puis l'arbre droit.

— parcours **suffixe** : on parcourt l'arbre gauche, puis l'arbre droit et on note la racine.

Le parcours en largeur consiste à relever les nœuds par niveau dans l'ordre croissant de profondeur.

**1)** Donner les 4 parcours de l'arbre binaire ci-dessous.




**2)** Un arbre binaire de taille 6 a pour parcours infixe : 1-2-3-4-5-6.

**a)** Représenter cet arbre.

**b)** La racine de l'arbre a pour valeur 6. Donner une représentation de cet arbre.

**c)** L'arbre binaire est bien tassé. Représenter cet arbre.

## Parcours en profondeur

Les algorithmes des parcours en profondeur sont proches et s'appuie sur l'algorithme récursif suivant. Par exemple, pour le parcours infixe de l'arbre binaire :

> Si l'arbre n'est pas vide :
> — on parcourt l'arbre gauche ;
> — on note la racine ;
> — on parcourt l'arbre droit.

**1)** Écrire, en python, la fonction **parcours_infixe** qui prend en paramètre un arbre binaire et affiche le parcours infixe de l'arbre binaire.

**2)** Modifier votre fonction en ajoutant le paramètre **parcours** de type string pour que la fonction renvoie la chaine parcours contenant le parcours infixe.

**3)** Créer les fonctions **parcours_prefixe** et **parcours_sufffixe** pour les deux autres parcours puis vérifier avec des arbres binaires.

**4)** Recopier la fonction **parcours_infixe** puis modifier cette copie pour qu'elle renvoie une file contenant le parcours infixe.

**5)** Vérifier vos différentes fonctions avec les arbres binaires construits dans l'introduction de ce TP.

# Parcours en largeur

Le parcours d'un arbre binaire peut se faire horizontalement, par niveau de nœuds . On dit que c'est un **parcours en largeur**. Pour ce parcours, il est nécessaire d'utiliser une file dans laquelle nous aurons les arbres non encore visités.

L'algorithme de ce parcours en largeur est le suivant :

```
def parcours_largeur(arbre):

    on crée une chaine vide parcours qui contiendra le parcours en largeur

    on crée une file vide F et on enfile arbre

    Tant que la file non vide :

        on défile la tête dans la variable a (a est un arbre binaire)

        si arbre a non vide :

            on ajoute la valeur du noeud dans la chaine parcours

            si arbre gauche non vide, on enfile arbre gauche

            si arbre droit non vide, on enfile arbre droit

    on renvoie la chaine parcours
```

**1)** Écrire en python l'algorithme de parcours en largeur d'un arbre.

**2)** Contrôler les parcours en largeur avec vos différents arbres saisis.

**3)** Peut-on remplacer la file par une pile. Si oui, le réaliser, sinon expliquer.