Exercice 1. Dessine-moi un arbre

Le tableau ci-dessous décrit un arbre :

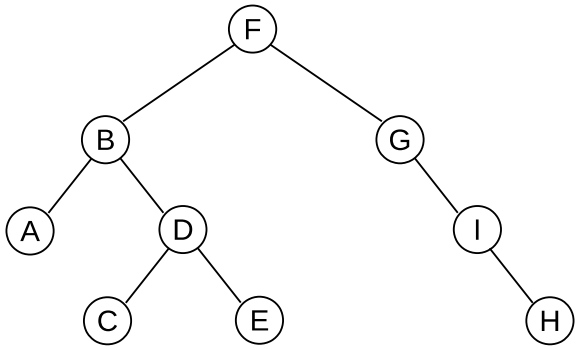
Père	Fils gauche	Fils droit
A	B	C
B	D	E
C	F	G
D	H	I
G	J	K
F	L	

1. Représenter l'arbre correspondant.
2. Quelle est la hauteur de cet arbre ?
3. Quelle est la taille de cet arbre (nombre de nœuds) ? Combien a-t-il de feuilles ?
4. Cet arbre est-il binaire ? complet ?

Exercice 2. Avec un dictionnaire (sujet de l'épreuve pratique)

Dans cet exercice, un arbre binaire de caractères est stocké sous la forme d'un dictionnaire où les clefs sont les caractères des nœuds de l'arbre et les valeurs, pour chaque clef, la liste des caractères des fils gauche et droit du nœud.

Par exemple, l'arbre :



est stocké dans :

```
a = {'F':['B','G'], 'B':['A','D'], 'A':['',''], 'D':['C','E'], 'C':['',''], \
'E':['',''], 'G':['','I'], 'I':['','H'], 'H':['','']}
```

Écrire une fonction récursive taille prenant en paramètres un arbre binaire arbre sous la forme d'un dictionnaire et un caractère lettre qui est la valeur du sommet de l'arbre, et qui renvoie la taille de l'arbre à savoir le nombre total de nœuds.

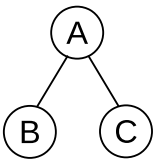
On pourra distinguer les 4 cas où les deux « fils » du nœud sont `''`, le fils gauche seulement est `''`, le fils droit seulement est `''`, aucun des deux fils n'est `''`.

Exemple :

```
>>> taille(a,'F')
9
>>> taille(a,'B')
5
>>> taille(a,'H')
1
>>> taille(a,'I')
2
```

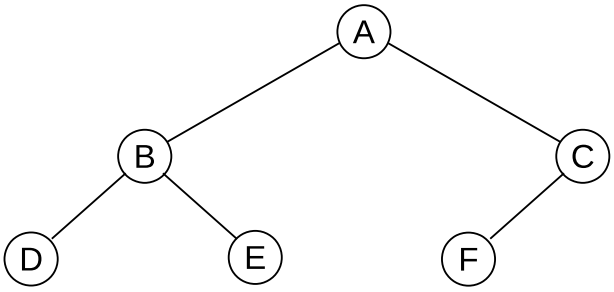
Exercice 3. Un arbre dans un tableau

Comme vous l'avez sans doute constaté, il est difficile de représenter un arbre sur une seule ligne. Une solution est de représenter l'arbre sous forme d'un tableau de tableaux.



Cet arbre se représente par le tableau : ['A',['B',[],[]],['C',[],[]]]

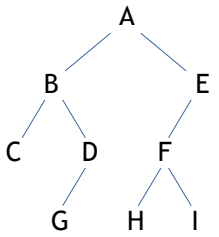
Écrire le tableau représentant l'arbre ci-dessous :



Exercice 4. Extrait du sujet 0 (écrit)

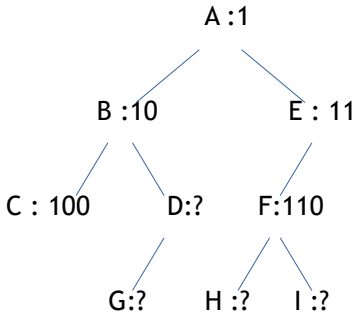
Dans cet exercice, on utilisera la convention suivante : la hauteur d'un arbre binaire ne comportant qu'un nœud est 1.

1. Déterminer la taille et la hauteur de l'arbre binaire suivant :



2. On décide de numéroter en binaire les nœuds d'un arbre binaire de la façon suivante :
    - la racine correspond à 1 ;
    - la numérotation pour un fils gauche s'obtient en ajoutant le chiffre 0 à droite au numéro de son père ;
    - la numérotation pour un fils droit s'obtient en ajoutant le chiffre 1 à droite au numéro de son père ;

Par exemple, dans l'arbre ci-dessous, on a utilisé ce procédé pour numéroter les nœuds A, B, C, E et F.



2.1. Dans l'exemple précédent, quel est le numéro en binaire associé au nœud G ?

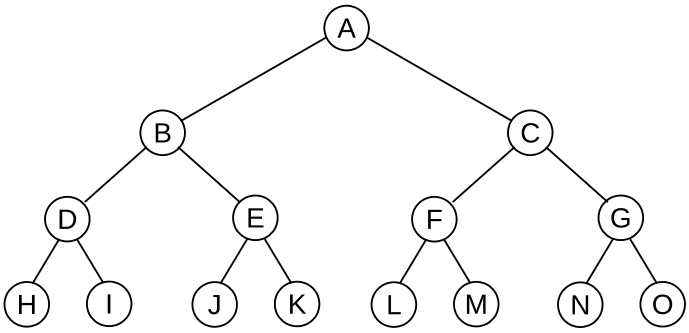
2.2. Quel est le nœud dont le numéro en binaire vaut 13 en décimal ?

2.3. En notant h la hauteur de l'arbre, sur combien de bits seront numérotés les nœuds les plus en bas ?

3. Justifier que pour tout arbre de hauteur h et de taille $n \geqslant 2$, on a :

$$h \leqslant n \leqslant 2^h - 1$$

4. Un arbre binaire est dit complet si tous les niveaux de l'arbre sont remplis.



On décide de représenter un arbre binaire complet par un tableau de taille $n + 1$, où n est la taille de l'arbre, de la façon suivante :

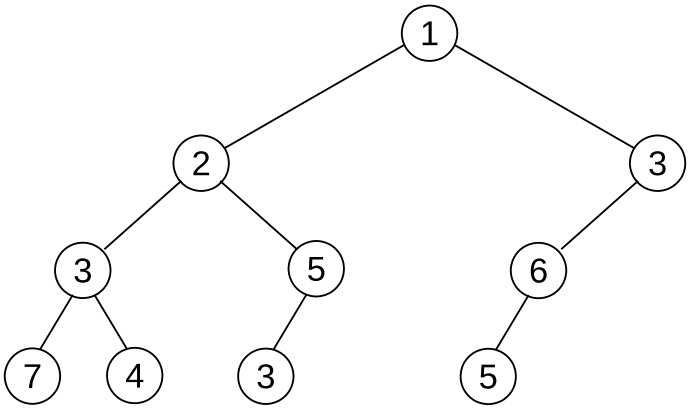
- La racine a pour indice 1 ;
- Le fils gauche du nœud d'indice i a pour indice $2 \times i$;
- Le fils droit du nœud d'indice i a pour indice $2 \times i + 1$;
- On place la taille n de l'arbre dans la case d'indice 0.

4.1. Déterminer le tableau qui représente l'arbre binaire complet de l'exemple précédent.

4.2. On considère le père du nœud d'indice i avec $i \geqslant 2$. Quel est son indice dans le tableau ?

Exercice 5. La classe Noeud

On veut implémenter l'arbre ci-dessous :



à l'aide de la classe Noeud ainsi définie :

```
class Noeud:
    def __init__(self ,valeur):
        self.valeur = valeur
        self.gauche = None
        self.droit = None
    def __str__(self):
        return str(self.valeur)
```

Attention : la classe Noeud est différente de celle du cours.

Dans cette arbre, les valeurs ne peuvent être que des entiers.

A partir de la questions 2, toutes les fonctions sont **récursives**.

1. Implémenter l'arbre représenté ci-dessus en utilisant la classe Noeud.
   Le nœud 1 sera appelé `racine`.
2. Créer une fonction affiche(a) qui reçoit le nœud racine de l'arbre en paramètre et **affiche** l'arbre avec des parenthèses autour du nœud et de ses sous arbres.
   On pourra agréablement s'inspirer très fortement du code du cours.

```
>>> affiche(racine.gauche.gauche)
( ( 7 ) 3 ( 4 ) )
>>> affiche(racine)
( ( ( ( 7 ) 3 ( 4 ) ) 2 ( ( 3 ) 5 ) ) 1 ( ( ( 5 ) 6 ) 3 ) )
```

3. Écrire une fonction `tailleArbre(a)` qui reçoit le nœud racine de l'arbre en paramètre et renvoie la taille de l'arbre (nombre de nœuds).

```
>>> tailleArbre(racine)
10
```

4. Ajouter une fonction `hauteur(a)` qui reçoit le nœud racine de l'arbre en paramètre et renvoie la hauteur de l'arbre.

```
>>> hauteur(racine)
4
```

5. Créer une fonction `feuilles(a)` qui reçoit le nœud racine de l'arbre en paramètre et renvoie le nombre de feuilles dans l'arbre.

```
>>> feuilles(racine)
4
```

6. Compléter ce magnifique code d'une fonction `cherche(a, x)` qui reçoit le nœud racine de l'arbre et une entier *x* en paramètres et renvoie le nombre d'occurrences de *x* dans l'arbre.

```
>>> cherche(racine,3)
3
>>> cherche(racine,7)
1
>>> cherche(racine,9)
0
```