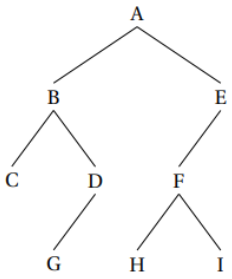
ET_1_Ex3_Corrigé

Exercice 3 : Arbres binaires et les arbres binaires de recherche

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

Question 1

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



Arbre binaire

1. On appelle **taille d'un arbre** le nombre de noeuds présents dans cet arbre.
2. Dans un **arbre binaire**, un noeud possède au plus 2 fils.
3. On appelle **profondeur d'un nœud** ou d'une feuille dans un arbre binaire le nombre de nœuds du chemin qui va de la racine à ce nœud. La racine d'un arbre est à une profondeur 1 (ici, mais cela dépend de la convention).
4. On appelle **hauteur d'un arbre** la profondeur maximale des nœuds de l'arbre.

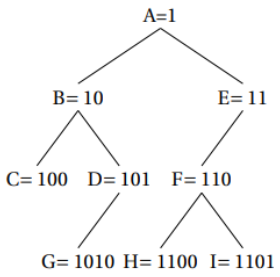
— Par définition, la taille d'un arbre est le nombre de noeud qu'il contient. Ici il y en a 9 donc la taille est 9.

— Par définition la hauteur est le nombre de noeuds du chemin le plus long dans l'arbre, ici les chemins les plus long sont ABDG, AEFH et AEFI. Comme la hauteur d'un arbre ne contenant qu'un noeud est 1, la hauteur de cet arbre est 4.

Question 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.

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



On a donc G :1010.

2. **Quel est le nœud dont le numéro en binaire vaut 13 en décimal?**
$13_{10} = 1101_2$ or I :1101 donc cela correspond au noeud I.
3. **En notant h la hauteur de l'arbre, sur combien de bits seront numérotés les nœuds les plus en bas?**
A chaque niveau de l'arbre on rajoute 1 bit donc les numéros des noeuds les plus bas (les feuilles) contiennent h bits.

4. **Justifier que pour tout arbre de hauteur h et de taille $n \geq 2$, on a : $h \leq n \leq 2^h - 1$.**

   — Soit un arbre de hauteur h. Il contient un maximum de noeuds si il est complet. Or un arbre binaire complet de hauteur h contient $1+2+2^2+...+2^{h-1}$ noeuds. C'est la somme d'une suite géométrique de raison 2 donc :

   $$1+2+2^2+...+2^{h-1} = \frac{2^h-1}{2-1} = 2^h - 1$$

   On a donc $n \leq 2^h - 1$.

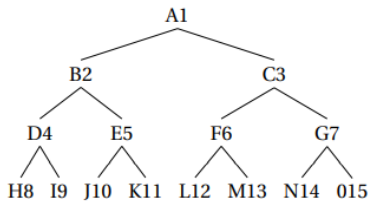
   — Soit un arbre de hauteur h. La hauteur maximale que l'on peut obtenir avec un minimum de noeuds est le cas où chaque père n'a qu'un seul fils . Dans ce cas la hauteur est égale à n. Donc $h \leq n$.

   — Donc $h \leq n \leq 2^h - 1$

Question 3

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.

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



On a donc le tableau

$$\boxed{[15, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O]}$$

2. **On considère le père du nœud d'indice i avec $i \geq 2$. Quel est son indice dans le tableau ?**
   Le père d'un fils d'indice i a pour indice $i/2$ si i est pair et $(i-1)/2$ sinon.

   Sous python on peut dans ce cas utiliser la fonction // .
   $a//b$ donne le quotient de la division euclidienne de a par b .

Question 4

Écrire une fonction recherche ayant pour paramètres un arbre *arbre* et un élément *element*.
Cette fonction renvoie *True* si *element* est dans l'arbre et *False* sinon. L'arbre sera représenté par un tableau comme dans la question précédente.

```
def recherche(arbre,element):
    '''In : arbre et element entier
      Out : true si element est dans la liste'''
    taille = arbre[0]
    i=1
    while i<=taille:
        if element==arbre[i]:
            return True
        elif element>arbre[i]:
            i=2*i+1
        else:
            i=2*i
    return False
```