Classe de Terminale NSI

Devoir surveillé

Durée : 30 min + 30 min

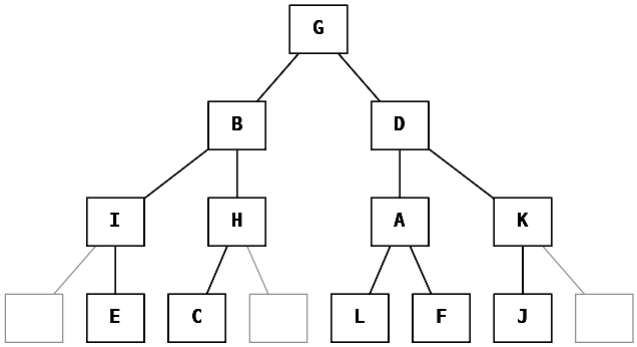
Première partie (10 points)

Dans tout le sujet, on suppose qu'on a à notre disposition deux classes `Pile` et `File` dont les instances sont munies des traditionnelles méthodes `est_vide`, `empiler`/`enfiler` et `depiler`/`defiler`.

Exercice 1 (5 points)

On rappelle qu'une façon de représenter un arbre binaire en mémoire est d'utiliser une liste en respectant quelques règles :

- si `h` désigne la hauteur de l'arbre binaire (en considérant qu'un arbre réduit à sa racine est de hauteur 1), alors la longueur du tableau est `2**h`,
- l'indice `0` du tableau n'est pas utilisé, on y stocke la valeur `None`,
- la racine de l'arbre est placée à l'indice `1`,
- l'enfant gauche (s'il existe) du nœud stocké à l'indice `k` est stocké à l'indice `2*k`,
- l'enfant droit (s'il existe) du nœud stocké à l'indice `k` est stocké à l'indice `2*k + 1`,
- lorsqu'un nœud n'a pas d'enfant gauche et/ou d'enfant droit, on stocke `None` à l'indice correspondant.



1. Représenter par une liste l'arbre binaire ci-dessus. Les indices des éléments de la liste sont indiqués.

Liste															
Indices	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

2.

a. Rappeler le principe d'un parcours d'arbre en largeur.

b. Donner l'ordre dans lequel les nœuds de l'arbre ci-dessus sont parcourus lors d'un parcours en largeur.

3.

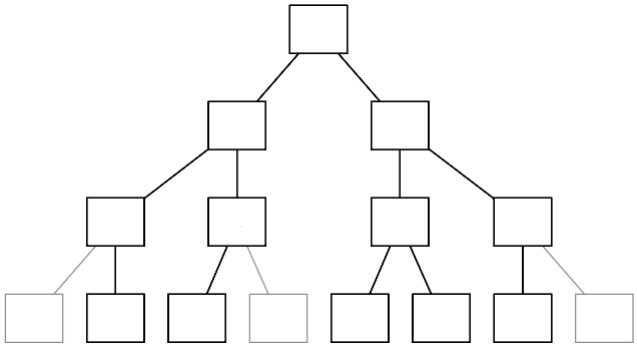
a. Rappeler le principe d'un parcours d'arbre en profondeur infixe.

b. Donner l'ordre dans lequel les nœuds de l'arbre ci-dessus sont parcourus lors d'un parcours en profondeur infixe.

4.

a. Justifier que l'arbre précédent n'est pas un arbre binaire de recherche.

b. Placer dans l'arbre ci-dessous les mêmes étiquettes que dans l'arbre précédent, en faisant en sorte que l'arbre devienne un arbre binaire de recherche.



Exercice 2 (5 points)

On donne la définition de deux classes `Noeud` et `ArbreBinaire`.

```
class Noeud:
    def __init__(self, etiquette, gauche=None, droit=None):
        """
        Crée une instance de la classe Noeud.
        - Entrées : etiquette (type quelconque)
                    gauche, droit (instances de la classe Noeud, ou None si pas d'enfant)
        """
        self.etiquette = etiquette
        self.gauche = gauche
        self.droit = droit

    def est_feuille(self):
        return self.gauche is None and self.droit is None

class ArbreBinaire:
    def __init__(self, racine):
        """
        Crée une instance de la classe ArbreBinaire.
        - Entrée : racine (instance de la classe Noeud, ou None pour un arbre vide)
        """
        self.racine = racine

    def est_vide(self):
        return self.racine is None

    def sa_gauche(self):
        return ArbreBinaire(self.racine.gauche)

    def sa_droit(self):
        return ArbreBinaire(self.racine.droit)

    def est_feuille(self):
        return self.racine.est_feuille()
```

1. Écrire les spécifications manquantes. Pour les méthodes de la classe `ArbreBinaire`, préciser celle(s) qui provoque(nt) une erreur en cas d'appel sur un arbre binaire vide.
2. Dessiner l'arbre suivant :

```
arbre = ArbreBinaire(Noeud(10, Noeud(3, Noeud(0), Noeud(7, Noeud(5, None, Noeud(6)), None)), None))
```

3. Justifier qu'il s'agit d'un arbre binaire de recherche.
4. Sur le dessin de la question précédente, insérer successivement dans l'arbre binaire de recherche les valeurs `1`, `9` et `8`, dans cet ordre.
5. **BONUS :** Redessiner l'arbre binaire de recherche de la question précédente, après avoir retiré les valeurs `9` et `6`, dans cet ordre.

Seconde partie (10 points)

Copier sur le bureau le fichier `devoir5.zip` depuis le dossier `Devoir` du réseau.

Lorsque vous aurez terminé, vous renommerez votre fichier `nom_prenom.ipynb` et vous le déposerez dans le dossier `Rendu` du réseau. Attention, une fois déposé dans `Rendu`, votre travail n'est plus modifiable.