Exercices sur les arbres

Numérique et Sciences Informatiques - Lycée P. Méchain

Exercice 1 : arbre lexicographique

Soit l'arbre ci-dessous :

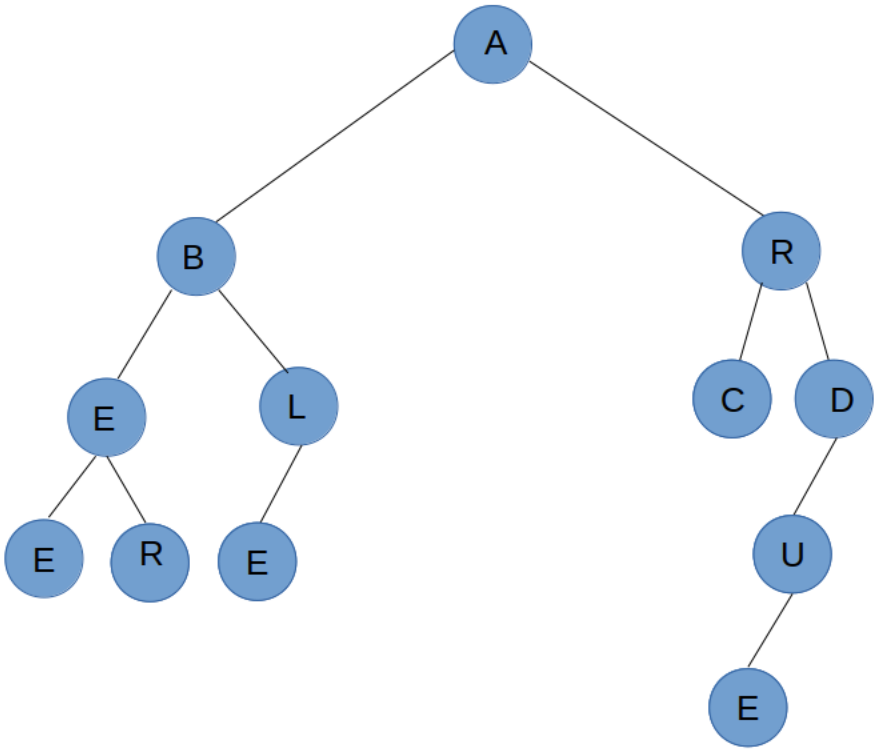

FIGURE 1 – Arbre binaire

1. **Donner** la profondeur de chaque nœud.
2. **Donner** la taille et la hauteur de l'arbre.
3. **Donner** l'ordre des nœuds visités lors d'un parcours en largeur de l'arbre.
4. **Donner** l'ordre des nœuds visités pour les parcours préfixe, postfixe puis infixe de l'arbre.

Exercice 2 : arbre de Huffman

Le codage de Huffman est un algorithme de compression de données sans perte. Le principe du codage de Huffman repose sur la création d'un arbre qui tient compte du nombre d'occurrences de chaque caractère.
Ci-dessous l'arbre obtenu avec la phrase '*je suis en nsi*'. Chaque caractère constitue une feuille de l'arbre. Une feuille contient le caractère et le parcours de l'arbre permet d'obtenir son codage binaire.
La racine et les noeuds interne contiennent le nombre d'occurences des caractères inclus dans les sous-arbres.

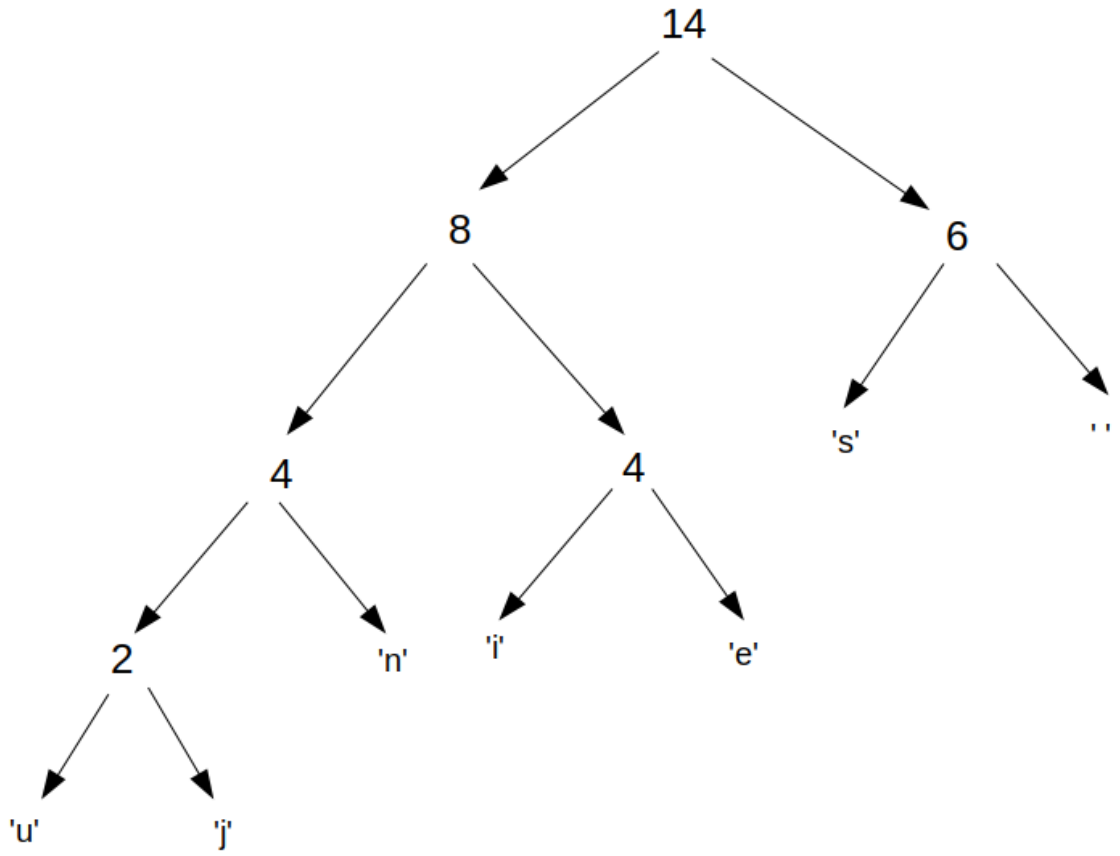

FIGURE 2 – Arbre de huffman

1. **Donner** la profondeur de chaque feuille.
2. **Donner** la taille et la hauteur de l'arbre.
3. Pour déterminer le code binaire de chaque caractère on réalise un parcours de la racine jusqu'aux feuilles en rajoutant à chaque fois au code un **0** ou un **1** selon la branche suivie (**0** : branche de gauche, **1** : branche de droite).
   Par exemple pour le caractère espace le code binaire est '*11*', pour le caractère 's' le code binaire est '*10*'.
   Indiquer le type de parcours à utiliser (largeur ou profondeur en précisant l'ordre préfixe, infixe ou postfixe) pour obtenir le code de chaque caractère.
   Donner le code binaire de chaque caractère.

Exercice 3 : implémentation d'un arbre binaire en Python

On dispose de la classe Nœud ci-dessous :

```
class Noeud:
    def __init__(self, donnee, gauche, droit):
        self.donnee = donnee
        self.gauche = gauche
        self.droit = droit
```

FIGURE 3 – Code de la classe Nœud en Python

On utilise le code ci-dessous afin de créer un arbre binaire.

```
mon_arbre = Noeud(6, Noeud(3, Noeud(2, None, None),
                              Noeud(5,
                                    Noeud(4, None, None),
                                    Noeud(6, None, None))),
                     Noeud(9,
                           Noeud(8, None, None),
                           Noeud(15, None, None)))
```

FIGURE 4 – Utilisation de la classe Nœud pour créer un arbre

1. **Représenter** graphiquement l'arbre obtenu.
2. **Donner** la profondeur de chaque nœud.
3. **Donner** la taille et la hauteur de l'arbre.
4. **Donner** l'ordre des nœuds visités lors d'un parcours en largeur de l'arbre.
5. **Donner** l'ordre des nœuds visités pour les parcours préfixe, postfixe puis infixe de l'arbre.
6. **Écrire** la définition récursive de la fonction *taille(arbre)* qui renvoie la taille de l'arbre.
7. **Écrire** le code Python de la fonction récursive *taille(arbre)* qui renvoie la taille de l'arbre.
8. **Écrire** la définition récursive de la fonction *hauteur(arbre)* qui renvoie la hauteur de l'arbre.
9. **Écrire** le code Python de la fonction récursive *hauteur(arbre)* qui renvoie la hauteur de l'arbre.

Exercice 4 : implémentation du parcours en largeur en Python

Dans cet exercice on utilise la classe Nœud et l'arbre de l'exercice 3.
L'objectif est de réaliser un parcours en largeur en utilisant une File.

1. **Écrire** une fonction *parcours_ largeur(arbre)* qui affiche la valeur de chaque nœud de l'arbre passé en paramètre en utilisant le procédé ci-dessous : :
   - On enfile la racine de l'arbre
   - Tant que la file n'est pas vide :
     - Si le nœud n'est pas vide :
       - On affiche la valeur du nœud
       - On enfile le nœud gauche
       - On enfile le nœud droit
     - On défile
2. **Tester** votre fonction.

Exercice 5 : Stocker un arbre dans une liste

Une autre méthode qui permet de représenter un arbre dans un tableau consiste à ranger le fils gauche d'un nœud i à l'emplacement d'indice $2i + 1$ et son fils droit à l'emplacement d'indice $2i + 2$. Ainsi le tableau est de taille n et la racine se situe à l'indice 0. *nil* représente l'arbre vide.

1. **Représenter** graphiquement l'arbre définit par le tableau [5, 2, 6, 1, 4, *nil*, 7].
2. **Donner** la liste qui représente l'arbre ci-dessous.

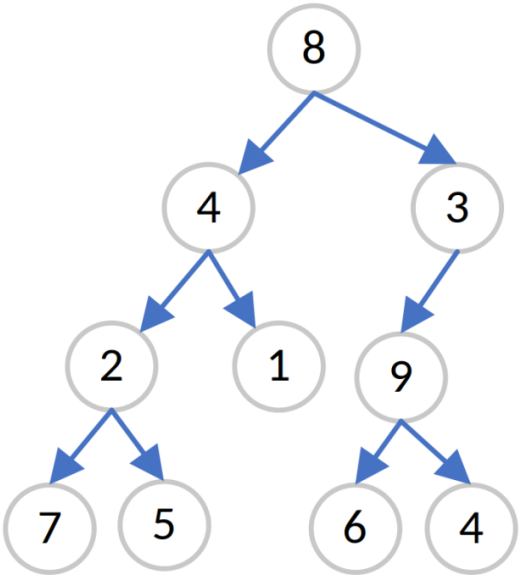

FIGURE 5 – Arbre binaire

3. On considère le père du nœud d'indice i avec $i \geq 1$. Quel est son indice dans le tableau ?

Exercice 6 : hauteur d'un arbre complet

On considère ici qu'un arbre ne contenant que la racine est de hauteur 0.

— **Réaliser** un tableau avec dans la colonne de gauche la taille N de l'arbre complet, dans la colonne du milieu le nombre de chiffres binaires n nécessaires pour représenter la taille de l'arbre et dans la colonne de droite la hauteur h de l'arbre.

— **Exprimer** h en fonction de n.

— **Exprimer** la taille N d'un arbre complet en fonction de n.

Exprimer la hauteur d'un arbre complet en fonction de la taille N de l'arbre.