Chap 07. Arbres binaires

Livre p 147 – Chap 8 Arbres binaires

1. Une structure abstraite : les arbres binaires

a. Définitions et vocabulaire

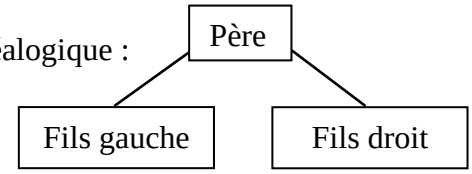
- Un **nœud** est une structure abstraite comportant une valeur (ou étiquette) et deux pointeurs (droite et gauche) qui dirigent soit vers un autre nœud, soit vers le vide.
- Un **arbre binaire** peut être défini de manière récursive :
  - Soit il est vide
  - Soit il est composé de 3 éléments :
    - Une **racine** comportant une valeur
    - Un nœud correspondant au sous-arbre droit
    - Un nœud correspondant sous-arbre gauche

Remarques : Un nœud dont les deux pointeurs dirigent vers le vide est appelé une **feuille**.
Les liaisons entre deux nœuds sont appelées : **arêtes** ou **branches**.
Les arbres sont généralement représentés « à l'envers », c'est-à-dire du haut (la racine) vers le bas (les feuilles)…

Vocabulaire généalogique

En plus du vocabulaire botanique on utilise aussi au vocabulaire généalogique :

- **Père**
- **Enfants** : Fils droit et fils gauche (nœuds non vides).



b. Mesures d'un arbre

Définitions

- La **taille** d'un arbre est le nombre total de nœuds de l'arbre.
- La **profondeur** d'un nœud est le nombre de nœuds en partant de la racine jusqu'à lui.
- La **hauteur** d'un arbre est la plus grande profondeur de ses feuilles.
  Remarque : La hauteur de l'arbre vide est 0.
  Attention : il existe une autre définition de la hauteur d'un arbre dans laquelle on ne compte pas la racine… l'arbre vide a alors une hauteur égale à -1 : Bien lire les consignes des exercices !

Cas particuliers

Un arbre est dit **dégénéré** (ou filiforme) si tous les nœuds qui ne sont pas de profondeur maximale ont 1 unique fils.
Un arbre binaire est dit **complet** (ou parfait) si tous les nœuds qui ne sont pas de profondeurs maximales possèdent exactement 2 fils.
Un arbre est dit **équilibré** si toutes les feuilles ont la même profondeur.

Propriété

Si on note t la taille d'un arbre binaire et h sa hauteur, alors on a : $h \leq t \leq 2^h - 1$.
(h correspond à un arbre dégénéré et $1 + 2 + 2^2 + \ldots + 2^{h-1} = 2^h - 1$ correspond à un arbre complet)

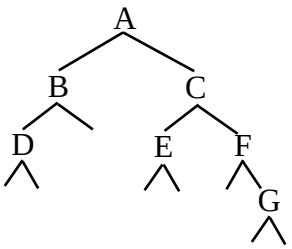
c. Parcours d'un arbre

Il existe différentes façons de parcourir un arbre binaire :

- Le parcours **préfixe** : En partant de la racine on visite récursivement le père, puis le fils gauche, puis le fils droit.
- Le parcours **infixe** : En partant de la racine on visite récursivement le fils gauche, puis le père, puis le fils droit.
- Le parcours **postfixe** (ou **suffixe**) : En partant de la racine on visite récursivement le fils gauche, puis le fils droit, puis le père.
- Le parcours **en largeur** : On part de la racine et on descend progressivement en visitant tous les nœuds situés à la même profondeur de gauche à droite.

Exemples de parcours :

- Parcours **préfixe** : A – B – D – C – E – F – G
  plus précisément : (A, (B, (D, ,),), (C, (E, ,), (F, , (G, ,))))
- Parcours **infixe** : D – B – A – E – C – F – G
  plus précisément : (((, D,), B,), A, ((, E,), C, (, F, (, G,))))
- Parcours **postfixe** : D – B – E – G – F – C – A
  plus précisément : ((((, , D), , B), ((, , E), (, (, , G), F), C), A)
- Parcours en largeur : A – B – C – D – E – F – G



Remarque : La taille de cet arbre est 7, sa hauteur est 4.

2. Implémentation en Python

a. Deux nouveaux objets

En python, on peut créer les objets **Noeud** et **Arbre** :

Exemple précédent :

```
g = Noeud(None, "G", None)
f = Noeud(None, "F", g)
e = Noeud(None, "E", None)
d = Noeud(None, "D", None)
c = Noeud(e, "C", f)
b = Noeud(d, "B", None)
a = Noeud(b, "A", c)
tree = Arbre()
tree.racine = a
```

```
class Noeud:
    """un noeud d'un arbre binaire"""
    def __init__(self, g, v, d):
        self.gauche = g
        self.valeur = v
        self.droit  = d
```

```
class Arbre:
    """un arbre binaire"""
    def __init__(self):
        self.racine = None
```

b. Fonctions et méthodes sur ces nouveaux objets

Les mesures et parcours peuvent être définis ainsi :

- **Taille** d'un arbre :
  Exemple : taille(a) ou tree.taille()
- **Hauteur** d'un arbre :
  Exemple : hauteur(a) ou tree.hauteur()
- **Parcours préfixe** d'un arbre :
  Exemple : prefixe(a) ou tree.prefixe()
- **Parcours infixe** d'un arbre :
  Exemple : infixe(a) ou tree.infixe()
- **Parcours postfixe** d'un arbre :
  Exemple : postfixe(a) ou tree.postfixe()

```
def taille(a):
    """le nombre de noeuds de l'arbre a"""
    if a is None:
        return 0
    else:
        return 1 + taille(a.gauche) + taille(a.droit)
```

```
def hauteur(a):
    """la hauteur de l'arbre a"""
    if a is None:
        return 0
    else:
        return 1 + max(hauteur(a.gauche), hauteur(a.droit))
```

```
class Arbre:
    """un arbre binaire"""
    def __init__(self):
        self.racine = None

    def taille(self):
        return taille(self.racine)

    def hauteur(self):
        return hauteur(self.racine)

    def prefixe(self):
        return prefixe(self.racine)

    def infixe(self):
        return infixe(self.racine)

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

```
def prefixe(a):
    """affiche les éléments de a dans un parcours préfixe"""
    if a is None:
        return ""
    else :
        return "{} {} {}".format(a.valeur, prefixe(a.gauche), prefixe(a.droit))

def infixe(a):
    """affiche les éléments de a dans un parcours infixe"""
    if a is None:
        return ""
    else :
        return "{} {} {}".format(infixe(a.gauche), a.valeur, infixe(a.droit))

def postfixe(a):
    """affiche les éléments de a dans un parcours postfixe"""
    if a is None:
        return ""
    else :
        return "{} {} {}".format(postfixe(a.gauche), postfixe(a.droit), a.valeur)
```