TP 1: Parcours d'Arbre Binaire.

HACINI Malik

September 2023

Table des matières

1 Introduction

Le but de ce TP est de représenter un arbre binaire en python via une classe, puis de le parcourir en profondeur de 3 façons différentes :
-Préfixe
-Postfixe
-Infixe

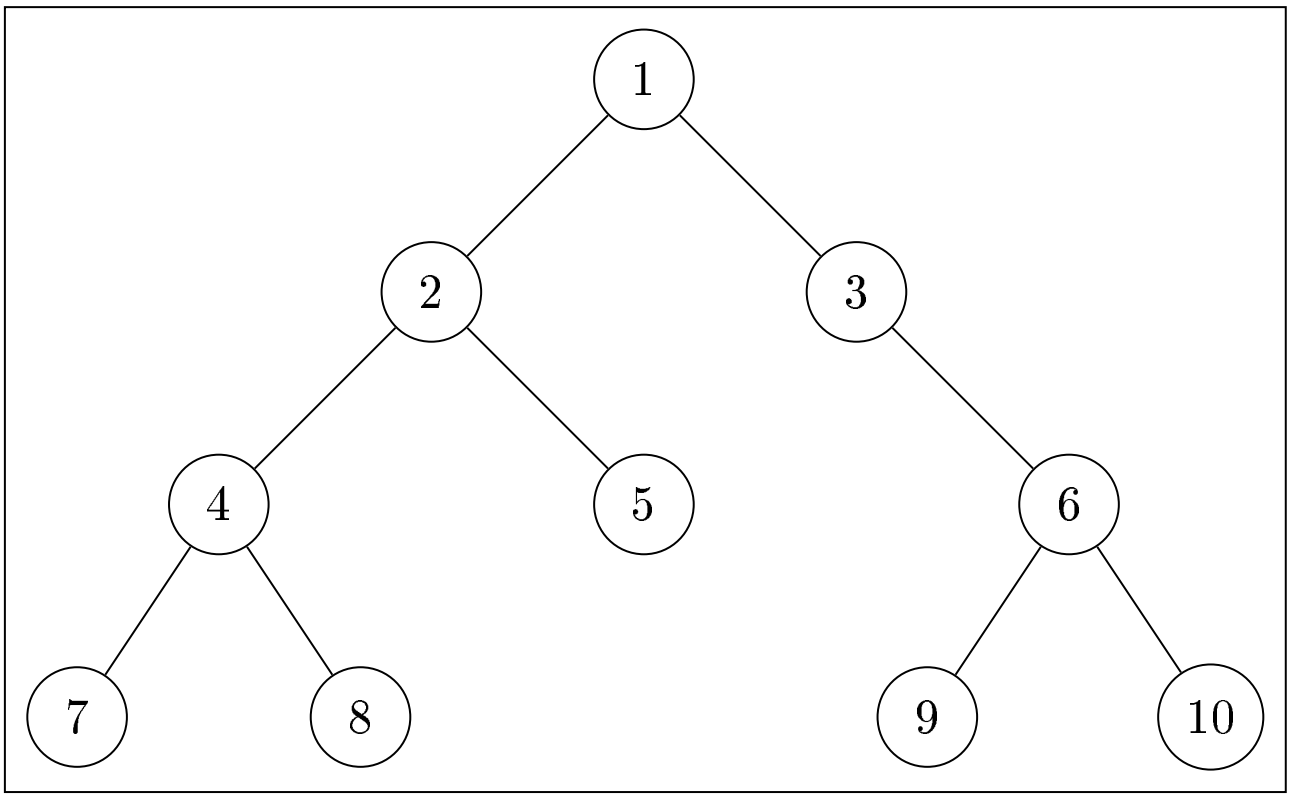
2 Classe "Noeud"

Voici l'implémentation de la classe Noeud. Chaque noeud a pour attribut l'information qu'il porte (un entier) et ses fils gauches et droits, d'autres noeuds. On ajoute aussi les méthode de classe ajouter-d et ajouter-g, qui permettent de créer un noeud, fils gauche ou droit d'un autre.

```
    from __future__ import annotations

class Noeud:
    def __init__(self,info=int,f_g=None,f_d=None):
        self.info=info
        self.f_g=f_g
        self.f_d=f_

    #Ajoute un noeud a l'arbre, en tant que fils de son pere (self)
    def ajouter_d(self,info):
        self.f_d = Noeud(info)
        return self.f_d

    def ajouter_g(self,info):
        self.f_g = Noeud(info)
        return self.f_g
```

3 Construction de l'Arbre.

3.1 Schéma.



3.2 Implémentation.

```
#On construit l'arbre de bas en haut.
n1=Noeud(1)
n2=n1.ajouter_g(2)
n3=n1.ajouter_d(3)
n4=n2.ajouter_g(4)
n5=n2.ajouter_d(5)
n6=n3.ajouter_d(6)
n7=n4.ajouter_g(7)
n8=n4.ajouter_d(8)
n9=n6.ajouter_g(9)
n10=n6.ajouter_d(10)
```

4 Parcours de l'Arbre.

Nous définissons chaque parcours comme une méthode différente de la classe Noeud.

4.1 Préfixe.

ordre : r -> g -> d

```
def prefixe(self)->list:
    info=[self.info]
    if self.f_g!= None:
        info= info + self.f_g.prefixe()
    if self.f_d!= None:
        info= info + self.f_d.prefixe()
    return info
```

4.2 Postfixe.

Ordre : g d r

```
    def postfixe(self)->list:
    info=[]
    if self.f_g!=None:
        info= info + self.f_g.postfixe()

    if self.f_d!=None:
        info= info + self.f_d.postfixe()
    info = info + [self.info]
    return info
```

4.3 Infixe.

ordre : g r d

```
    def infixe(self)->list:
    info=[]
    if self.f_g!=None:
        info= info + self.f_g.infixe()

    info = info + [self.info]

    if self.f_d!=None:
        info= info + self.f_d.infixe()
    return info
```

5 Tests.

On teste avec pytest des noeuds de tout types, sur chaque parcours.

5.1 Préfixe.

```
    def test_prefixe():

    assert n1.prefixe()==[1, 2, 4, 7, 8, 5, 3, 6, 9, 10]
    assert n4.prefixe()==[4, 7, 8]
    assert n5.prefixe()==[5]
    assert n9.prefixe()==[9]
```

5.2 Postfixe.

```
    def test_postfixe():
assert n1.postfixe()==[7, 8, 4, 5, 2, 9, 10, 6, 3, 1]
assert n4.postfixe()==[7, 8, 4]
assert n5.postfixe()==[5]
assert n9.postfixe()==[9]
```

5.3 Infixe.

```
    def test_infixe():
    assert n1.infixe()==[7, 4, 8, 2, 5, 1, 3, 9, 6, 10]
    assert n4.infixe()==[7, 4, 8]
    assert n5.infixe()==[5]
    assert n9.infixe()==[9]
```

5.4 Résultats

On lance pytest : Les tests sont tous réussis YAHOUU c fini.