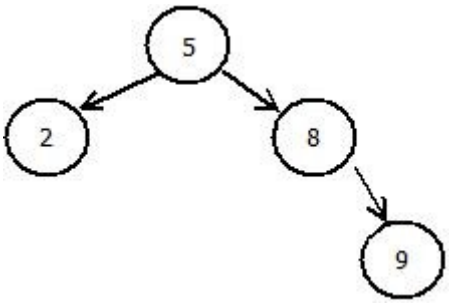
Michaël Guedj

TP d'Algorithmique avancée

Arbres binaires (2)

Soit l'arbre binaire schématisé ainsi :



Que nous implémentons en Python comme suit :

```
class Noeud(object):
    # proprietes
    def __init__(self):
        self.val = None
        self.fg = None
        self.fd = None

    def __str__(self):
        return self.val


# -- creation des sommets
a = Noeud() # a : un objet, instance de la classe Noeud
a.val = 5

b = Noeud()
b.val = 2

c = Noeud()
c.val = 8


f = Noeud()
f.val = 9


# -- creation des arcs
a.fg = b
a.fd = c
```

```
    c.fd = f
```

Nous donnons, en outre, l'implémentation du prédicat `est_feuille` (resp. `est_vide`) prenant en argument un sommet, et testant si ce sommet est une feuille (resp. si ce sommet est vide).

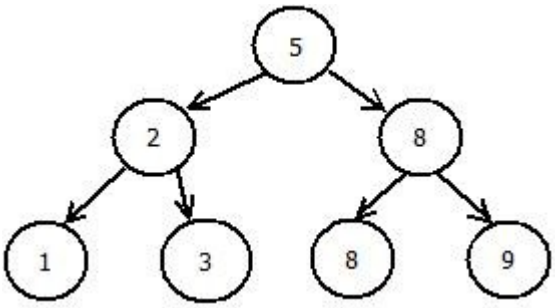
```
def est_feuille(s):
    if est_vide(s):
        return False
    return s.fg == None and s.fd == None

def est_vide(s):
    return s == None
```

Si *s* est un nœud, on note *Arbre(s)* l'arbre ayant pour racine le nœud *s*.

Exercice 1

Implémentez l'arbre binaire schématisé ci-dessous.



Exercice 2

Implémentez la fonction `somme`, prenant en argument un nœud *s* ; et retournant la **somme** de *Arbre(s)* ; i.e. la somme des valeurs des nœuds de *Arbres(s)* :

$$somme\left(Arbre(s)\right) := \sum_{s' \in Arbre(s)} s'.val$$

Exercice 3

Implémentez le prédicat `est_abr`, prenant en argument un nœud *s* ; et retournant *Vrai* si *Arbre(s)* est un **arbre binaire de recherche** ; et *Faux* sinon.

Exercice 4

Implémentez la fonction `rcc_abr`, prenant en argument un nœud *s*, ainsi qu'une valeur *x* ; et qui retourne *Vrai* si la valeur *x* est présente dans *Arbre(s)* ; et *Faux* sinon.

Note : *Arbre(s)* est supposé est être un arbre binaire de recherche.

Exercice 5

Effectuez le **parcours infixe**, de l’arbre implémenté en Exercice 1.