Spécialité NSI Terminale
Correction du
contrôle n° 6
Mardi 24 janvier 2023

Exercice 1

1. La plus grande somme racine-feuille est 16 :

$$2 + 7 + 4 + 3 = 16$$

2. a. On peut écrire par exemple les instructions suivantes (l'arbre binaire est associé à la variable `A`) :

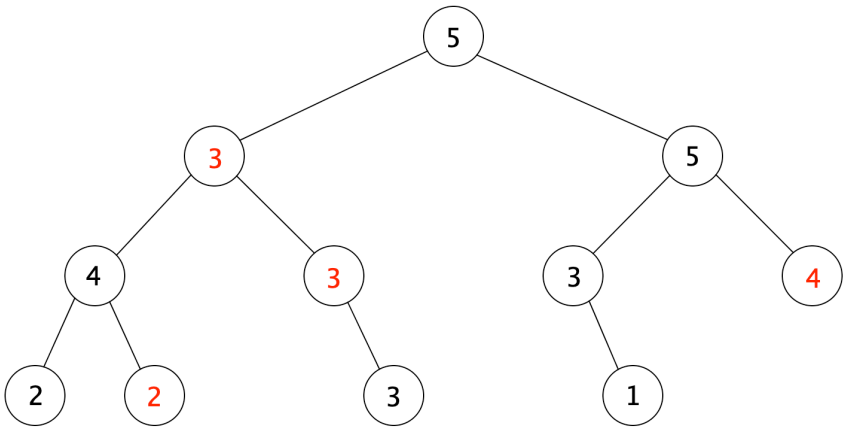
```
A = Noeud(2)
AG = Noeud(7)
AD = Noeud(5)
AG.modifier_sag(Noeud(4))
AG.modifier_sad(Noeud(1))
AD.modifier_sad(Noeud(8))
A.modifier_sag(AG)
A.modifier_sad(AD)
```

b. L'appel de la méthode `niveau` sur l'arbre donne `3`. Il s'agit de la taille de l'arbre.

3. On peut écrire le code suivant :

```
    def pgde_somme(self):
        if self.sag!=None and self.sad!=None:
            sg = self.sag.pgde_somme()
            sd = self.sad.pgde_somme()
            return 1+max(sg, sd)
        if self.sag!=None:
            return self.sag.pgde_somme() + self.etiquette
        if self.sad!=None:
            return self.sad.pgde_somme() + self.etiquette
        return self.etiquette
```

4. a. On doit compléter l'arbre de la façon suivante :



b. On peut écrire la méthode suivante :

```
    def est_magique(self):
        if self.sag==None and self.sad==None:
            return True
        elif self.sag!=None :
            return self.sag.est_magique()
        elif self.sad!=None :
            return self.sad.est_magique()
        else :
            return self.sag.est_magique()\
             and self.sad.est_magique() and\
             self.sag.pgde_somme() == self.sad.pgde_somme()
```

Exercice 2

1. Selon la définition donnée dans l'énoncé, seul l'arbre 1 est un arbre binaire de recherche.

2. a. Le plus petit élément se trouve dans la feuille la plus à gauche de l'arbre. En effet par définition d'un ABR, cet élément sera inférieur ou égal à la clé de son père, qui sera lui-même inférieur ou égal à la clé de tout autre nœud de l'arbre.

b. On peut écrire le code suivant :

```
    def RechercheValeur(cle, arb):
        if est_vide(arb) :
            return False
        if cle < racine(arb) :
            return RechercheValeur(sous-arbre_gauche(arb))
        elif cle > racine(arb) :
            return RechercheValeur(sous-arbre_droit(arb))
        else :
            return True
```

3. a. Il s'agit d'un parcours en profondeur infixe.

b. Le parcours préfixe consiste traiter la racine de la racine *avant* de traiter les deux sous-arbres. On obtient donc le parcours suivant :

7 - 2 - 1 - 5 - 3 - 6 - 10 - 8 - 9

c. Le parcours postfixe consiste traiter la racine de la racine *après* avoir traité les deux sous-arbres. On obtient donc le parcours suivant :

1 - 3 - 6 -5 - 2 - 9 - 8 - 10 - 7

d. Le parcours en largeur donne le résultat suivant :

7 - 2 - 10 - 1 - 5 - 8 - 3 - 6 - 9