Correction

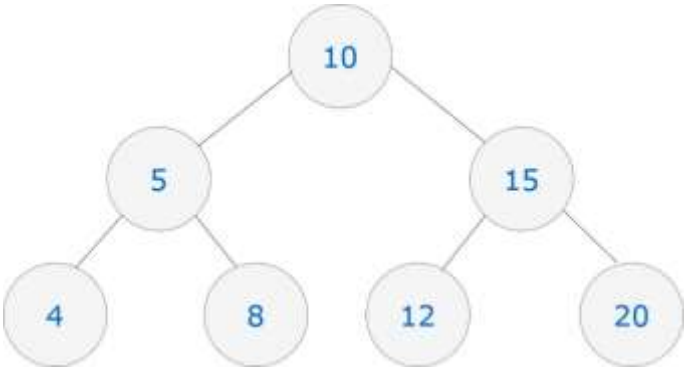
NSI - 2021 Métropole Libre Jour 2 (21-NSIJ2ME2)

Exercice 3 - Arbres binaires, Programmation Orientée Objet

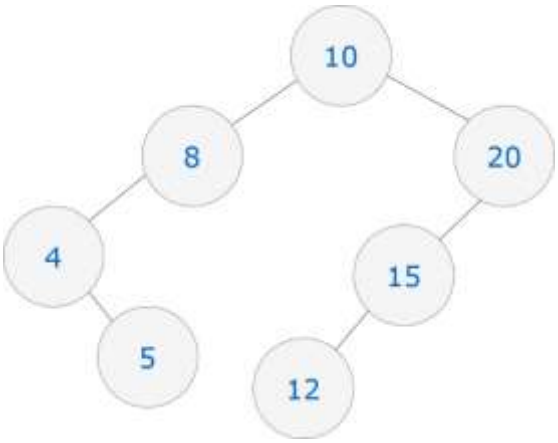
1. a. La taille de l'arbre est de 7.

1. b. La hauteur de cet arbre est de 4.

2.



3.



4.
```
def hauteur(self):
    return self.racine.hauteur()
```

5. Méthode taille de la classe Nœud :

```
def taille(self):
    if self.gauche and self.droit:
        return 1 + self.gauche.taille() + self.droit.taille()
    elif self.gauche:
        return 1 + self.gauche.taille()
    elif self.droit:
        return 1 + self.droit.taille()
    else:
        return 1
```

Correction

NSI - 2021 Métropole Libre Jour 2 (21-NSIJ2ME2)

Autre solution pour la méthode taille de la classe Nœud :

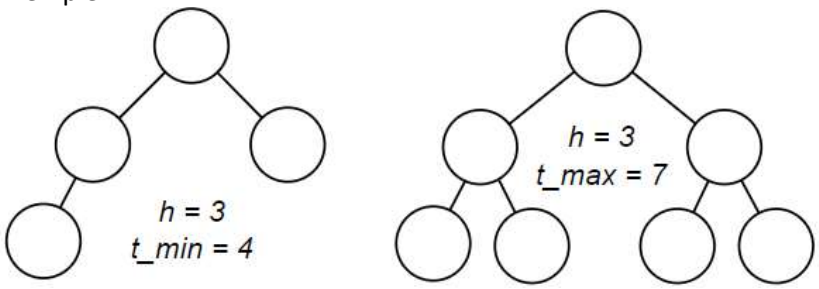
```
def taille(self):
    if self.gauche == None and self.droite == None:
        return 1
    elif self.gauche == None:
        return 1 + self.droite.taille()
    elif self.droite == None:
        return 1 + self.gauche.taille()
    else:
        return 1 + self.gauche.taille() + self.droite.taille()
```

Méthode taille de la Classe Arbre :

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

6. a. Pour trouver la taille minimale d'un arbre binaire de recherche « bien construit » de hauteur h il faut considérer la taille maximale d'un arbre de hauteur $h-1$ auquel on ajoute un nœud.
Soit $t\_\min = 2\hat{}(h-1) - 1 + 1 = 2\hat{}(h-1)$.

Exemple :



6. b.

```
def bien_construit(self):
    h = self.racine.hauteur()
    return 2**(h - 1) <= self.racine.taille() <= 2**h - 1
```