Arbres binaires - Exercices

Dans tout ce document, par arbre on entend arbre binaire.

Exercice 1 : construire des arbres

1. Dessiner tous les arbres binaires de taille 2.
2. Dessiner tous les arbres binaires de taille 3.

Exercice 2 : dénombrer des arbres

Sachant qu'il existe

- 1 arbre vide ;
- 1 arbre de taille 1 ;
- 2 arbres de taille 2 ;
- 5 arbres de taille 3 ;
- 14 arbres de taille 5.

Déterminer sans les construire le nombre d'arbres de taille 5.

Exercice 3 : une relation entre la hauteur et la taille d'un arbre

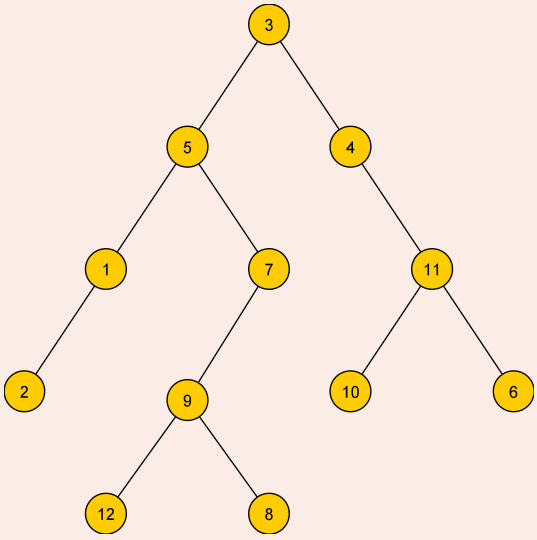
Soit h un entier naturel et A un arbre de hauteur h.

1. Combien, au minimum, A possède-t-il de nœuds ?
2. Combien, au maximum, A possède-t-il de nœuds ?

On en déduit l'encadrement suivant :
soit N la taille d'un arbre de hauteur h, alors on a

$$\leq N \leq$$

Exercice 4 : Parcours « à la main »

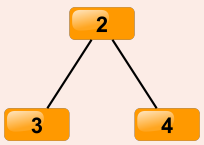


1. Écrire les valeurs de l'arbre dans l'ordre de son parcours préfixe.
2. Faire de même avec un parcours infixe.
3. Faire de même avec un parcours postfixe.

Exercice 5

1. Créer un fichier `Node.py` et implémenter la classe `Node` vue en cours.
2. Implémenter la méthode `__str__` qui utilise une sous-fonction récursive qui :
   - si on lui demande d'afficher `None` renvoie `''` ;
   - sinon (c'est qu'elle doit bien afficher un nœud) ouvre une parenthèse, affiche récursivement le sous-arbre gauche, puis affiche la valeur du nœud, le sous-arbre droit et enfin ferme la parenthèse.

   sur l'arbre suivant



   qui est créé par

```
a = Node(3)
b = Node(4)
c = Node(2, a, b)
```

`print(c)` devra renvoyer `'((3)2(4))'`

3. Implémenter la méthode d'instance **`size`** qui renvoie un `int` qui est la taille de l'arbre (s'aider du cours).
4. Comment trouver récursivement la hauteur d'un arbre ? Proposer une « méthode logique » et implémenter la méthode d'instance **`height`**, qui renvoie un `int` qui est la hauteur de l'arbre.
5. Implémenter la méthode d'instance `__eq__` qui renvoie **`True`** si deux arbres sont égaux et **`False`** sinon (trouver une méthode récursive).

Exercice 6 : Parcours

1. Ajouter à la classe `Node` une méthode d'instance **`prefix`** qui renvoie un `str` qui est la chaîne de caractères obtenue en concaténant toutes les valeurs des nœuds de l'arbre au cours de son parcours préfixe.
2. De même implémenter **`infix`** et **`postfix`**.