

Les arbres binaires de recherches

Les arbres binaires de recherche

Un arbre binaire de recherche (ABR) est une structure de donnée composée de noeuds. Chaque noeud a au plus 2 enfants ordonnés d'une manière particulière :

- les enfants à gauche d'un noeud ont des valeurs inférieures à lui.
- les enfants à droite d'un noeud ont des valeurs supérieures à lui.

Et cela doit être vrai pour chaque nœud de l'arbre.

❗ REPÈRES :

Nous travaillerons avec le fichier abr.py fourni, qui contient une classe Noeud et de quoi faire afficher graphiquement les arbres que nous construirons.

Voici la classe Noeud

```
class Noeud:
    # Le constructeur
    def __init__(self, value, left=None, right=None):
        self.value = value
        self.left = left
        self.right = right
        self.parent = None

    def __str__(self):
        return str(self.value)

    def estFeuille(self):
        if not self.left and not self.right:
            return True
        else:
            return False
```

On a rajouté la variable d'instance parent.

Comment construire un ABR?

Par exemple, On souhaite représenter la liste de nombres [6, 8, 3, 1, 4, 9, 2, 7, 5]

• Dans un premier temps on crée la racine de l'arbre.

• Puis on insère les noeuds en respectant les règles d'un ABR.

Comme le montre le schéma ci-contre:

Insertion du 6
6

Insertion du 8
6
8

Insertion du 3
3 6 8

8 > 6 -> droite

3 < 6 -> gauche

Insertion du 1
1 3 6 8
1 < 6 -> gauche
1 < 3 -> gauche

Insertion du 4
1 3 4 6 8

4 < 6 et 4 > 3

Insertion du 9
1 3 4 6 8 9

9 > 6 et 9 > 8

Insertion du 2
1 2 3 4 6 8 9
2 < 6, 2 < 3 et 2 > 1

Insertion du 7
1 2 3 4 6 7 8 9
7 > 6 et 7 < 8

Insertion du 5
1 2 3 4 5 6 7 8 9
5 < 6, 5 > 3 et 5 > 4

L'algorithme d'insertion dans un ABR :

Voici la méthode permettant d'insérer une valeur dans un ABR :

```
def insert(self, valeur):
    if valeur < self.value:
        if self.left is None:
            self.left = Noeud(valeur)
            self.left.parent = self
        else:
            self.left.insert(valeur)
    elif valeur > self.value:
        if self.right is None:
            self.right = Noeud(valeur)
            self.right.parent = self
        else:
            self.right.insert(valeur)
```

À FAIRE 1:

Écrire un algorithme qui correspond à ce code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

À FAIRE 2:

Rajouter cette méthode à la classe Noeud, puis tester le programme:

```
bst = Noeud(6)
bst.insert(8)
bst.insert(3)
bst.insert(1)
bst.insert(4)
bst.insert(9)
bst.insert(2)
bst.insert(7)
bst.insert(5)
graphicarbre(bst)
```

À FAIRE 3:

Écrire une fonction `insertion(arbre, val)` qui réalise le même travail mais sans que ce soit une méthode de classe.

? QUESTION 1:

Récupérer les méthodes des différents parcours (préfixe, infixe et suffixe).

Faites les fonctionner sur notre arbre.

Lequel d'entre eux affiche la liste triée?

.....
.....
.....

? EXERCICE 1 :

⌞ Écrire un programme qui utilise une structure d'ABR pour trier une liste d'entier.

? EXERCICE 2 :

⌞ Écrire un programme qui utilise une structure d'ABR pour trouver le maximum d'une liste d'entier.

? EXERCICE 3 :

⌞ Écrire un programme qui utilise une structure d'ABR pour trouver le minimum d'une liste d'entier.

? EXERCICE 4 :

⌞ Écrire un programme qui utilise une structure d'ABR pour rechercher une valeur dans la liste:

⌞ L=['chat', 'chien', 'souris', 'araignée', 'crapaud', 'grenouille', 'lézard', 'zèbre']

⌞ Un tel algorithme est de complexité logarithmique.

? EXERCICE 5 :

⌞ Écrire un programme qui utilise une structure d'ABR pour trouver le chemin depuis la racine jusqu'à un noeud donné.