COURS – LES ARBRES BINAIRES

Table des matières

I) connaissance théoriques / sur papier

A)définition

Un arbre binaire est une structure de données récursive formée :
- d'un nœud racine (valeur principale),
- de deux sous-arbres : un gauche et un droit, qui sont eux-mêmes des arbres binaires (ou vides).

Chaque nœud peut donc avoir :
- 0 enfant (feuille),
- 1 enfant (gauche ou droit),
- 2 enfants.

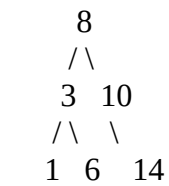
B) Vocabulaire

- **Nœud** : élément de l'arbre contenant une valeur.
- **Racine** : nœud au sommet de l'arbre.
- **Feuille** : nœud sans enfants.
- **Sous-arbre** : arbre contenu dans un autre.
- **Hauteur/profondeur** : nombre maximal de niveaux entre la racine et une feuille.
- **Arbre vide** : arbre sans nœud.

Ce vocabulaire est à connaître par cœur. Sans lui, de nombreux points seront perdus en examen.

C) Représentation

Exemple visuel (à dessiner) :



- Racine : 8
- Feuilles : 1, 6, 14
- Sous-arbre gauche de 8 : arbre avec racine 3

II) parcours d'un arbre

Soit une classe nœud :

```
class noeud:
    def __init__(self, valeur, gauche=None, droite=None):
        self.valeur = valeur
        self.gauche = gauche
        self.droite = droite
```

A)parcours préfixe

Le principe du parcours **préfixe** est de parcourir chaque élément de l'arbre dans l'ordre : **la racine , le sous arbre gauche, le sous-arbre droit** et de retenir chaque nœud la **première fois** qu'on le rencontre. En python on le code comme suit :

```
def parcours_prefixe(noeud):
    if nœud :
        print(noeud.valeur)
        parcours_prefixe(noeud.gauche)
        parcours_prefixe(noeud.droite)
```

Il s'agit de la même idéeque la récursivité. On appelle jusqu'à qu'une condition soit remplie, ici que les sous arbres aient la valeur de None.

B)Parcours postfixe/suffixe

Le principe du parcours **postfixe/suffixe** est de parcourir chaque élément de l'arbre **dans l'ordre** : racine ,sous-arbre gauche, sous-arbre droit et de retenir chaque nœud la **dernière fois** qu'on le rencontre. En python on le code comme suit :

```
def parcours_postfixe(noeud):
    if nœud :
        parcours_postfixe(noeud.gauche)
        parcours_postfixe(noeud.droite)
        print(noeud.valeur)
```

C) Parcours infixe

Le principe du parcours **infixe** est de parcourir le sous-arbre **gauche**, **ensuite** de renvoyer le **Nœud**, et **enfin** examiner le sous-arbre **droit**. En python on le code comme suit :

```
def parcours_infixe(noeud):
    if nœud :
        parcours_infixe(noeud.gauche)
        print(noeud.valeur)
        parcours_infixe(noeud.droite)
```

Attention : il est vital de savoir parcourir à la main des arbres binaires en suivant chacun des trois parcours. Des exercices seront disponibles en fin de leçon mais ceux-ci ont peu de chances de suffire pour des examens.

III) Connaissances pratiques

A) Représentation en Python (structure)

Soit la même classe nœud :

```
class noeud:
    def __init__(self, valeur, gauche=None, droite=None):
        self.valeur = valeur
        self.gauche = gauche
        self.droite = droite
```

Exemple :

```
arbre = noeud(8)
arbre.gauche=6
arbre.droite=9
```

C) Les arbres binaires de recherche (ABR)

Un arbre binaire est un arbre binaire de recherche si :
- Son sous-arbre gauche contient toutes les valeurs inférieures à la racine
- Son sous-arbre droit contient toutes les valeur supérieures à la racine
- la différence des profondeurs des sous-arbres gauches et droit est comprise entre -1 et 1 (seulement 1 de profondeur d'écart). Si cette condition est remplie, on dit qu'un arbre quelconque et équilibré.

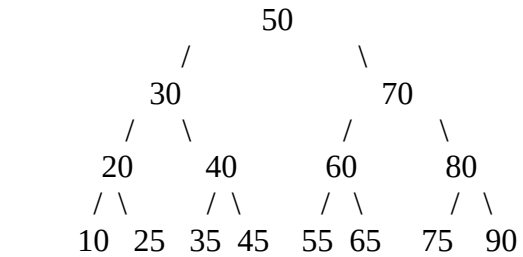
D) Recherche dans un ABR

Exemple de recherche dans un ABR :

```
def rechercher(noeud, valeur):
    if noeud is None:
        return False
    if noeud.valeur == valeur:
        return True
    elif valeur < noeud.valeur:
        return rechercher(noeud.gauche, valeur)
    else:
        return rechercher(noeud.droite, valeur)
```

Bonus:Exercices

Soit un arbre :



Dessiner cet arbre .

1: Effectuer le parcours préfixe sur cet arbre.

2: Effectuer le parcours suffixe sur cet arbre.

3: Effectuer le parcours infixe sur cet arbre.

4: Quelle est la profondeur de cet arbre ?

5: Quelle valeur a la racine de cet arbre ?

6 : Cet arbre est-il équilibré ?

7: S’agit-il d’un arbre binaire de recherche ?

8: Redessiner l’arbre en y insérant la valeur 68.

Bonus:Correction

1 : 10, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 90

2 : 10, 25, 20, 35, 45, 40, 30, 55, 65, 60, 75, 90, 80, 70, 50

3 : 10, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 90

4: 3 ou 4, suivant si l'on compte la racine ; En examen, les deux sont acceptés.

5: Elle est évidemment de 50.

6 : profondeur arbre gauche : 3

profondeur arbre droit : 3

On a : 3-3=0, 0 est compris entre -1 et 1 donc cet arbre est équilibré.

7 : on remarque que l'arbre est équilibré et le sous-arbre gauche de cet arbre contient toutes les valeurs inférieures à la racine, l'inverse pour le droit, donc il s'agit d'un arbre binaire de recherche.

8: 65 obtiendra un sous-arbre droit avec comme valeur 68, qui sera une feuille.

Ceci conclut ce cours.