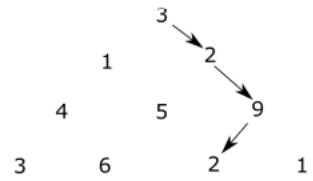
Bac Blanc Mai (Corrigé)

Exercice 1

1.

2.



score = 3+ 2 + 9 + 2 = 16

3.

2 - 5 - 2 ; 2 - 5 - 3 ; 2 - 1 - 3 ; 2 - 1 - 9

4.

Pour représenter les conduits dans une pyramide de 3 niveaux on peut utiliser un nombre écrit en binaire : 11 (à gauche puis à gauche) ; 10 (à droite puis à gauche) ; 01 (à gauche puis à droite) ; 00 (à droite puis à droite).
Pour représenter les conduit pour une pyramide de niveaux, on utilisera un nombre de n - 1 bits. Avec n - 1 bits, on peut écrire 2^{n-1} nombres, on peut donc modéliser 2^{n-1} conduits.
Il existe donc 2^{n-1} conduits dans une pyramide de hauteur n.

5.

Pour une pyramide de hauteur 5 on a 16 possibilités, pour une pyramide de hauteur 6 on a 32 possibilités. À chaque fois que l'on augmente la hauteur d'une unité, on double le nombre de conduits possibles (croissance exponentielle), il va donc être très rapidement impossible de dénombrer toutes les possibilités de conduits.

6.

```
def score_max(i, j, p):
    if i == len(p) - 1:
        return p[len(p) - 1][j]
    else :
        return p[i][j] + max(score_max(i+1,j,p), score_max(i+1,j+1,p))
```

7.

```
def pyramide_nulle(n):
    return [[0]*i for i in range(1,n+1)]
```

Autre méthode

```
def pyramide_nulle(n):
    tab = []
    for i in range(1,n+1):
        t = []
        for j in range(i):
            t.append(0)
        tab.append(t)
    return tab
```

8.

```
def prog_dyn(p):
    n = len(p)
    s = pyramide_nulle(n)
    # remplissage du dernier niveau
    for j in range(n) :
        s[n-1][j] = p[n-1][j]
    for i in range(n-2, -1,-1) :
        for j in range(i+1) :
            s[i][j] = p[i][j]+max(s[i+1][j], s[i+1][j+1])
    # renvoie du score maximal
    return s[0][0]
```

9. Le coût est quadratique car on peut observer la présence de 2 boucles imbriquées dans la fonction prog_dyn, ce qui est souvent le marqueur d'un coût quadratique. Si on veut être plus rigoureux, on peut aussi remarquer que pour une pyramide de hauteur n, on a un nombre d'opérations élémentaires égal à 1+2+3+...+n-1+n = n(n+1)/2 soit, en utilisant la notation O, une complexité en $O(n^2)$.

10. On pourrait de nouveau utiliser la programmation dynamique, mais cette fois en utilisant l'approche descendante (top-down). Cela nous permettrait d'éviter des appels récursifs quand les calculs ont déjà été faits.

    Pour se faire, on peut utiliser un dictionnaire permettant de stocker les valeurs pour chaque couple (i,j).

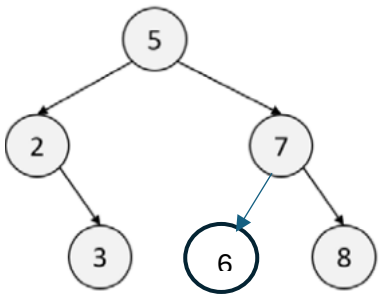
Exercice 2

Partie A

1/ Nœud racine : 5 // Fils de la racine : 2 et 7.

2/ Branche : 5,2 et 3.

3/



Partie B

1/ La fonction __init__ est le constructeur de la classe et permet de créer la racine de l'arbre.

2/ Cet élément ne sera pas rajouté.

3/
arbre = ABR(5)
arbre.insererElement(3)
arbre.insererElement(8)

Partie C

1/ Le parcours infixe.

2/ La complexité est de l'ordre de n*log(n) (pseudo-linéaire) pour ce type de tri est quadratique pour les tris par insertion et sélection.

Exercice 3 :

Partie A :

1. a vaut [10, 8, 9, 9, 8, 10, 6, 7, 8, 8] ; b vaut 'Fondation'
2.

```
def titre_livre(dico, id_livre):
    for i in range(len(dico['id'])):
        if dico['id'][i] == id_livre :
            return dico['titre'][i]
    return None
```

3.

```
def note_maxi(dico):
    n_max = 0
    for n in dico['note']:
        if n > n_max:
            n_max = n
    return n_max
```

4.

```
def livres_note(dico, n):
    tab_titres = []
    for i in range(len(dico['note'])):
        if n == dico['note'][i]:
            tab_titres.append(dico['titre'][i])
    return tab_titres
```

5.

```
def livre_note_maxi(dico):
    n_max = note_maxi(dico)
    return livres_note(dico, n_max)
```

PARTIE B

6. attribut de la classe Livre : self.id ; méthode de la classe Livre : get_id

7.

```
def get_note(self):
        return self.note
```

8.

```
b_r = Livre(8, 'Blade Runner', 'K.Dick', 1968,8)
b = Bibliotheque()
b.ajout_livre(b_r)
```

9.

```
def titre_livre(self, id_livre):
        for livre in self.liste_livre :
            if livre.get_id() == id_livre :
                return livre.get_titre()
        return None
```

PARTIE C

10. La clé primaire doit être unique. Un auteur peut écrire plusieurs livres, l'attribut auteur ne peut donc pas être une clé primaire.
11.

Ubik
Blade Runner

12.

```
SELECT titre
FROM livres
WHERE auteur = 'Asimov' AND ann_pub > 1950;
```

13.

```
UPDATE livres
SET note = 10
WHERE titre = 'Ubik';
```

14. Permet d'éviter la duplication des données : évite de réécrire l'ensemble des informations personnelles d'un auteur pour chacun de ses livres.
15. `id_auteur` est une clé étrangère, elle permet d'établir une relation entre la table `auteurs` et la table `livres` (jointure). À chaque valeur de l'attribut `id_auteur` correspond un auteur dans la table auteurs.
16.

```
SELECT nom, prenom
FROM auteurs
JOIN livre ON id_auteur = auteurs.id
WHERE ann_pub > 1960;
```

17. Cette requête permet d'obtenir le titre des livres écrits par des auteurs ayant moins de 30 ans au moment de leur publication
18. Ce projet ne respecte pas la protection de la vie privée : divulgation des données personnelles (date de naissance, numéro de téléphone…)