2023 Amérique du sud jour 1

Exercice 1

Partie A

1.

Destination	Passerelle	Métrique
R1	R1	0
R2	R3	2
R3	R3	1
R4	R3	3
R5	R3	2
R6	R8	2
R7	R3	2
R8	R8	1

2.a.

```
r1 = Routeur("R1")
r4 = Routeur("R4")
r5 = Routeur("R5")
r1.ajout_destination(r4, r3, 3)
r1.ajout_destination(r5, r3, 2)
```

2.b.

```
def ajout_destination(self, dest, pasr, m):
    self.table_routage[dest] = (pasr, m)
```

3. Les voisins sont les routeurs à une métrique de 1.

```
def voisins(self):
    tab = []
    for cle, valeur in self.table_routage:
        if valeur[1] = 1: # métrique à 1
            tab.append(cle) # la destination
    return tab
```

4.

```
def calcul_route(self, dest) :
    route = [self]
    routeur_courant = route[-1]
    while routeur_courant != dest:
        routeur_suivant = routeur_courant.table_routage[dest][0]
        route.append(routeur_suivant)
        routeur_courant = routeur_suivant
    return route
```

Partie B

5. Le champ `id` de la table `Routeur` est une clé primaire. Il ne peut donc pas posséder deux valeurs identiques. Or la valeur `3` existe déjà, il y aura donc une erreur.

6. `SELECT nom FROM Routeur WHERE prix >= 2500 AND prix <= 7000;`

7.

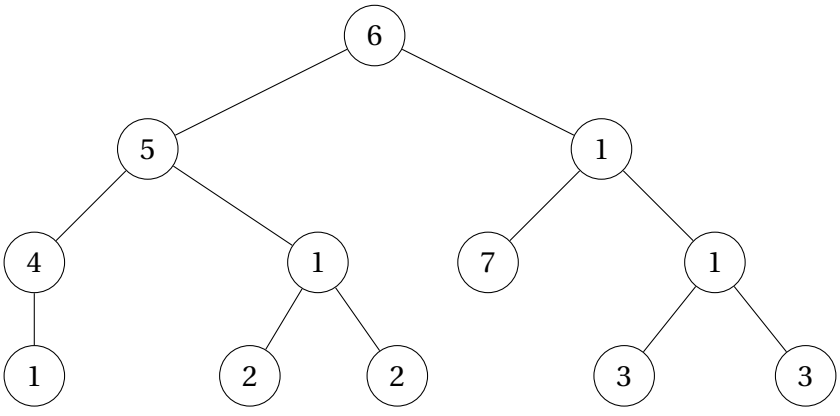
nom
Knuth
Hooper
Pouzin

8.

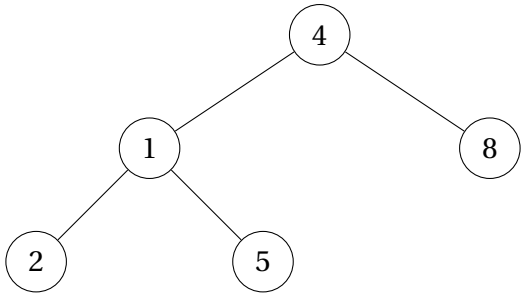
```
SELECT nom FROM Routeur
JOIN Commande ON Routeur.id = Commande.idRouteur
WHERE idClient = 2
ORDER BY nom;
```

Exercice 2

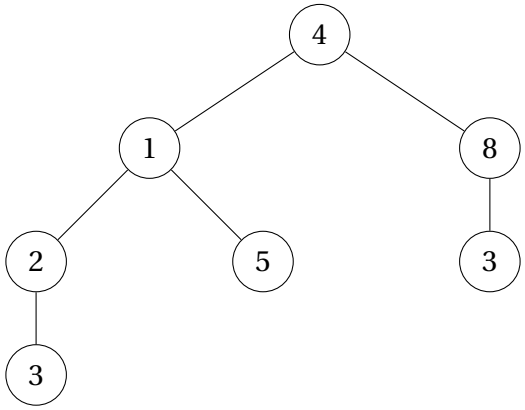
1.



2.a.



2.b.




```
n31 = [3, []]
n32 = [3, []]
n2 = [2, [n31]]
n5 = [5, []]
n8 = [5, [n32]]
n1 = [1, [n2, n5]]
a = [4, [n1, n8]]
```

3.

```
def poids(arbre):
    if arbre == [] :
        return 0
    p = 0
    file = creer_file()
    enfiler(file, arbre)
    while not est_vide(f) :
        arbre = defiler(f)
        for sous_arbre in arbre[1] :
            enfiler(file, sous_arbre)
        p = p + arbre[0]
    return p
```

4.

```
def est_mobile(arbre):
    if len(arbre[1]) < 2:
        return True
    else:
        a = arbre[1][0]
        b = arbre[1][1]
        return poids(a) == poids(b) and est_mobile(a) and est_mobile(b)
```

Exercice 3

1. Il faut calculer le nombre total de pixels et multiplier par trois couleurs :

$$1000 \times 1500 \times 3 = 4500000 \text{ octets} = 4.5 \text{ Mo}$$

2.

$$0 \leq i < n$$

$$0 \leq j < m$$

3.a. De type `list`. C'est un tableau à deux dimensions.

3.b. Il y a $n \times m$ appels à la fonction `energie`.

4.

```
def calcule_energie(couture, tab_energie):
    e = 0
    for i,j in couture:
        e = e + tab_energie[i][j]
    return e
```

5.

```
def indices_proches(m, i, j):
    indices = [(i,j)]
    if j > 0 :
        indices.append((i,j-1))
    if j < m-1 :
        indices.append((i,j+1))
    return indices
```

6.

```
def calcule_couture(j, tab_energie) :
    n = len(tab_energie)
    m = len(tab_energie[0])
    couture = []
    couture.append((0,j))
    for i in range(1, n):
        indices_pixels = indices_proches(m, i, j)
        pixel_min = min_energie(tab_energie, indices_pixels)
        couture.append(pixel_min)
    return couture
```

7.

```
def meilleure_couture(tab_energie) :
    m = len(tab_energie[0])
    # On initialise le minimum avec la première couture
    couture_mini = calcule_couture(0, tab_energie)
    energie_mini = calcule_energie(couture_mini, tab_energie)
    for j in range(1, m):
        couture = calcule_couture(j, tab_energie)
        energie = calcule_energie(couture, tab_energie)
        if energie < energie_mini:
            energie_mini = energie
            coutuer_mini = couture
    return couture
```