Proposition de correction

Exercice 1

Partie A

Q1

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

Q2.a

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

Q2.b

```
def ajout_destination(self, dest : 'Routeur', pasr : 'Routeur', m : int):
    """
    @brief      ajout à la table de routage
    @param      dest -- un routeur de destination
    @param      pasr -- un routeur passerelle
    @param      m -- la métrique (un entier m)
    """
    self.table_routage[dest] = (pasr, m)
```

Q3

```
def voisins(self) -> list:
    """
    @return     liste des routeurs directement connectés au routeur
    """
    liste = []
    for routeur in self.table_routage:
        if routeur != self:
```

```
            pasr, m = self.table_routage[routeur]
            if m == 1:
                liste.append(routeur)
        return liste
```

Q4

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

Partie B

Q5

La clef primaire de valeur égale à 3 existe déjà dans la table Routeur.

Q6

SELECT nom

FROM Routeur

WHERE prix >= 2500 AND prix <= 7000

ORDER BY nom ASC

Q7

Knuth

Hooper

Pouzin

Q8

SELECT DISTINCT Routeur.nom

FROM Routeur, Commande

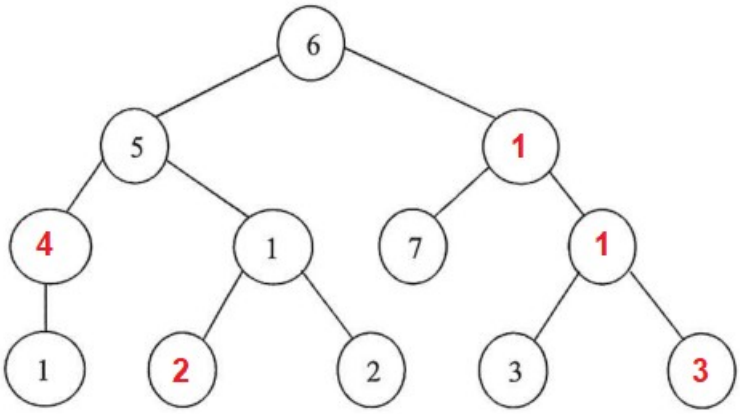
WHERE Routeur.id = Commande.idRouteur

AND Commande.idClient = 2

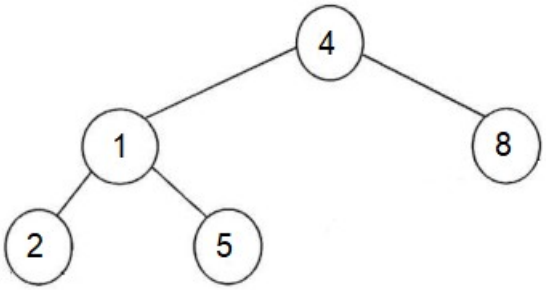
ORDER BY Routeur.nom ASC

EXERCICE 2

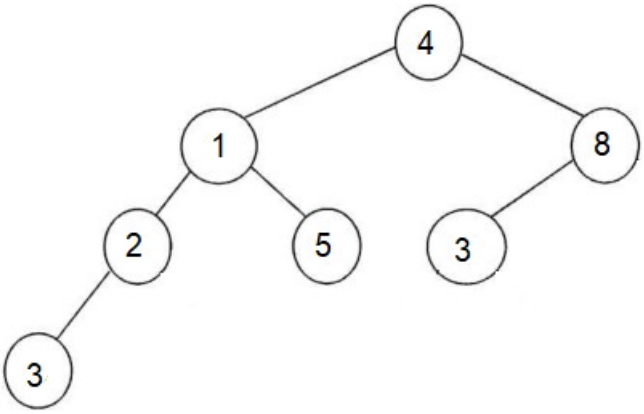
Q1



Q2.a



Q2.b




```
n3 = [3, []]
n2 = [2, [n3]]
n5 = [5, []]
n8 = [8, [n3]]
n1 = [1, [n2, n5]]
a = [4, [n1, n8]]
```

Q3

```
def poids(arbre):
    if arbre == [] :
            return 0
    p = 0
    file = creer_file()
    enfiler(file, arbre)
    while est_vide(file) is False:
            arbre = defiler(file)
            for sous_arbre in arbre[1] :
                    enfiler(file, sous_arbre)
            p = p + arbre[0]
    return p
```

Q4

```
def est_mobile(arbre : list) -> bool:
    """
    @brief  détermine si un arbre est mobile
    @param  arbre -- un arbre
    @return True si cet arbre est un arbre mobile, et False sinon
    """
    sag, sad = arbre
    if len(sad) == 2:
        meme_poids = poids(sad[0]) == poids(sad[1])
        return meme_poids and est_mobile(sad[0]) and est_mobile(sad[1])
    else:
        return True
```

Exercice 3

Q1

taille = largeur x hauteur x 3 = 1500 x 1000 x 3 o = 4,5 Mo

Q2

$0 \leq i < n$

$0 \leq j < m$

Q3.a

Type list

Q3.b

n x m

Q4

```
def calcule_energie(couture : list, tab_energie : list) -> int:
    """
    @param  couture -- tableau représentant une couture
    @param tab_energie -- tableau des énergies
    @return l'énergie de la couture
    """
    energie = 0
    for pixel in couture:
        i, j = pixel
        energie += tab_energie[i][j]
    return energie
```

Q5

```
def indices_proches(m : list, i : int, j : int) -> list:
    """
    @param  m -- nombre de colonnes de l'image
    @param i -- indice de ligne
    @param j -- indice de colonne
    @return positions des pixels compatibles
    """
    liste = []
    for x in range(j-1, j+2):
        if (j > -1) and (j < m):
            liste += [(i,x)]
    return liste
```

Q6

```
def calcule_couture(j, tab_energie) :
    n = len(tab_energie)       # hauteur de l'image
    m = len(tab_energie[0])  # largeur de l'image
    couture = []
    couture.append((0, j))     # premier pixel de la couture
    for i in range(1, n):
        mini = j
        for pixel in indices_proches(m, i, j):
            if tab_energie[i][pixel[1]] < tab_energie[i][mini]:
                mini = pixel[1]
        j = mini
    return couture
```

Q7

```
def meilleure_couture(tab_energie : list) -> list:
    """
    @param tab_energie -- tableau des énergies
    @return couture d'energie minimale
    """
    m = len(tab_energie[0])  # largeur de l'image

    meilleure = calcule_couture(0, tab_energie)
    energie_mini = calcule_energie(meilleure, tab_energie)
    for j in range(1, m):
        couture = calcule_couture(j, tab_energie)
        energie = calcule_energie(couture, tab_energie)
        if energie < energie_mini:
            meilleure = couture
            energie_mini = energie
    return meilleure
```