2022 Amérique du sud jour 2

Exercice 1

1.

```
if distance(t[idx],cible) < dmin:
    dmin = distance(t[idx],cible)
    idx_ppv = idx
```

2. La complexité est en O(n) car il y a une boucle qui parcourt le tableau.

3.a. On créer une variable `doc = distance(obj, cible)` et de remplacer les occurences de `distance(obj, cible)` par `doc`

3.b. Cela permet de regarder seulement le premier élément pour savoir si on a trouvé un nouvel élément plus proche.

3.c.

```
def insertion(kppv, idx, d):
    n = len(kppv)
    i = 0
    while kppv[i][1] > d and i < n:
        i += 1
    kppv.insert(i, (idx, d))
```

Exercice 2

Partie A

[Beaucoup de hors programme ici]

1. `ifconfig`

2. DHCP

3. 192.168.1.1 (192.168.1.255 est ladresse de broadcast)

4. C'est possible et cette adresse serait celle de la box vers Internet.

5. Oui, car les adresses 192.168.x.x ne sont pas routées sur Internet.

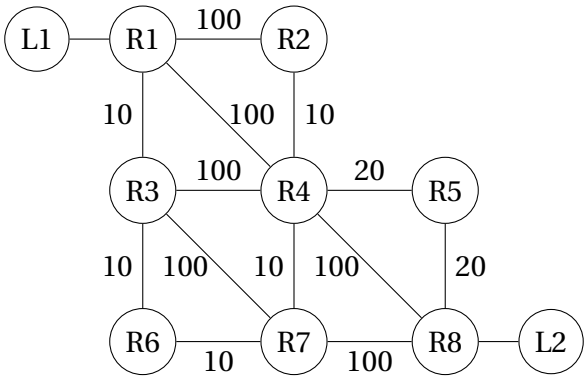
Partie B

1. $C = \frac{10^9}{50\times10^6} = \frac{100\times10^7}{50\times10^6} = 2\times10^1 = 20$

2.a.
Voici les coûts pour chaque liaison :

— Eth : 100

— VDSL : 20
— VDSL2 : 10



2.b. R1 R3 R6 R7 R4 R5 R8 pour un coût de 80.

2.c. Le coût le plus fiable pour atteindre R4 est 40. Le coût maximal de cette liaison devra donc être 40. On en déduit la bande passante minimale :

$$\text{BP} = \frac{10^9}{C} = \frac{10^9}{40} = \frac{100 \times 10^7}{40} = 2.5 \times 10^7 = 25\,\text{Mb/s}$$

Exercice 3

1.
```sql
UPDATE ModeleVelo SET Stock = 0 WHERE nomModele = 'Bovelo';
```

2. Requête 4 puis Requête 2

3.a.

```sql
SELECT nomModele, idFabriquant FROM ModelVelo WHERE Stock = 0;
```

3.b.

```sql
SELECT COUNT(numeroCommande) FROM Commande WHERE date >= '2022-01-01';
```

3.c.

```sql
SELECT nom FROM Fabricant
JOIN ModelVelo
ON Fabricant.idFabricant = ModeleVelo.idFabrbicant
WHERE Stock > 0;
```

4. Les noms des clients ayant acheté un Bovelo.

Exercice 4

1.a.
```
from math import sqrt
```

1.b.

```
def distance_points(a,b):
    return sqrt((a[0] - b[0])**2 + (a[1] - b[1])**2)
```

2.

```
def distance(p, a, b):
    if a == b:
        return distance_points(p, a)
    else:
        return distance_point_droite(p, a, b)
```

3.

```
def le_plus_loin(ligne):
    n = len(ligne)
    deb = ligne[0]
    fin = ligne[n-1]
    dmax = 0
    indice_max = 0
    for idx in range(1, n-1):
        p = ligne[idx]
        d = distance(p, deb, fin)
        if d > dmax:
            dmax = d
            indice_max = idx
    return (indice_max, dmax)
```

4.

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

ou avec un slice (hors programme, mais plus pratique) :

```
def extrait(tab, i, j):
    return tab[i,j+1]
```

5.

```python
def simplifie(ligne, seuil):
    n = len(ligne)
    if n <= 2:
        return ligne
    else:
        indice_max, dmax = le_plus_loin(ligne)
        if dmax <= seuil:
            return (ligne[0], ligne[n-1])
        else:
            simplifie(extrait(ligne, 0, indice_max) , seuil)
                + simplifie(extrait(ligne, indice_max, n-1) seuil)
```

Exercice 5

1. Voici toutes les sommes racine-feuille :

- $2+7+4+2=15$
- $2+7+4+3=16$
- $2+7+1+5=15$
- $2+5+3+1=11$
- $2+5+8=15$

Le maximum est donc 16.

2.a.

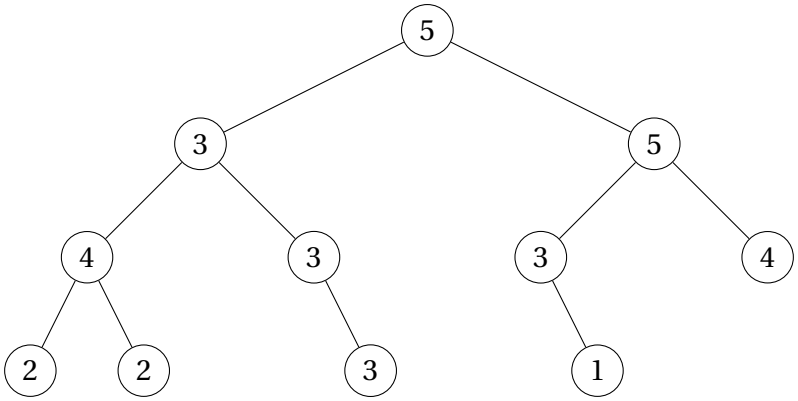
```python
n2 = Noeud(2)
n7 = Noeud(7)
n4 = Noeud(4)
n1 = Noeud(1)
n5 = Noeud(5)
n8 = Noeud(8)

n7.modifier_sag(n4)
n7.modifier_sad(n1)
n5.modifier_sad(n8)
n2.modifier_sag(n7)
n2.modifier_sad(n5)
```

2.b. 2

3.

```python
def pgde_somme(self):
    if self.sag != None and self.sad != None:
        sg = self.sag.pgde_somme()
        sd = self.sad.pgde_somme()
        return self.etiquette + max(sg, sd)
    if  self.sag != None:
        return self.etiquette + self.sag.pgde_somme()
    if  self.sad != None:
        return self.etiquette + self.sad.pgde_somme()
    return self.etiquette
```

4.a.
La somme des chemins doit être 14 d'après le seul chemin complet.



4.b.

```
def est_magique(self):
    if self.sag != None and self.sad != None:
        return self.sag.est_magique() and
                self.sad.est_magique() and
                self.sag.pgde_somme() == self.sad.pgde_somme()
    if  self.sag != None:
        return self.sag.est_magique()
    if  self.sad != None:
        return self.sad.est_magique()
    return True
```