Éléments de correction sujet 12 (2022)

Exercice 1

1.

```
def plus_proche_voisin(t, cible):
    dmin = distance(t[0], cible)
    idx_ppv = 0
    n = len(t)
    for idx in range(1,n):
        if distance(t[idx], cible) < dmin:
            idx_ppv = idx
            dmin = distance(t[idx], cible)
    return idx_ppv
```

2. La complexité de la fonction *plus_proche_voisin* est linéaire (O(n)).
3.
   a. Il suffit de rajouter une ligne permettant de calculer la distance entre obj et cible au début de la boucle for :
   ```
   for idx in range(n) :
         d = distance(obj, cible)
         …
   ```
   et de remplacer `distance(obj, cible)` par `d` dans le reste du programme.

   b. Il est nécessaire d'obtenir une liste triée à la fin du processus, il est moins couteux en temps de maintenir la liste triée que de tout trier à la fin du processus.

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

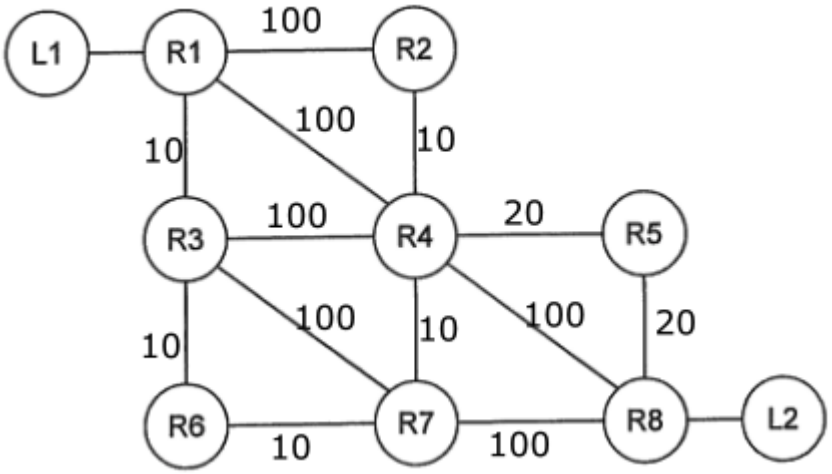
Exercice 2

PARTIE A

1. ifconfig
2. DHCP (totalement hors programme !)
3. 192.168.1.1 (totalement hors programme !)
4. C'est possible et cette adresse serait celle de la box vers Internet (totalement hors programme !)
5. Oui, car les adresses 192.168.X.X ne sont pas routées sur Internet. (totalement hors programme !)

PARTIE B

1. $C = 10^9/50.10^6 = 20$
2.
   a.



   b. L1 -> R1 -> R3 -> R6 -> R7 -> R4 -> R5 -> R8 -> L2 pour un coût total de 80
   c. Il faut qu'au maximum le coût total du chemin L1 -> R1 -> R4 -> R5 -> R8 -> L2 soit de 80. Donc au maximum, le coût de la liaison R1 -> R4 doit être de 40 (40+20+20 = 80). La bande passante de la liaison R1 -> R4 doit être au minimum de 25 Mb/s ($10^9/40 = 2{,}5.10^7$)

Exercice 3

1.
```
UPDATE ModeleVelo
SET Stock = 0
WHERE nomModele = ‘Bovelo’
```
2.
Requête 4 puis Requête 2
3.
   a.
```
SELECT nomModele, idFrabricant
FROM ModeleVelo
WHERE Stock = 0
```
   b.
```
SELECT COUNT(numeroCommande)
FROM Commande
WHERE date >= ‘2022-01-01’
```
   c.
```
SELECT nom
FROM Fabricant
JOIN ModeleVelo ON ModeleVelo.idFabricant =
Fabricant.idFabricant
WHERE Stock > 0
```
4.
Cette requête permet d’obtenir le nom des clients ayant acheté le modèle “Bovelo” (si un même client a acheté plusieurs fois le type de vélo “Bovelo”, son nom n’apparaîtra qu’une seule fois).

Exercice 4

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

2.
```
def distance(p, a, b):
    if distance_points(a,b) != 0:
        return distance_point_droite(p, a, b)
    else :
        return distance_points(a,p)
```

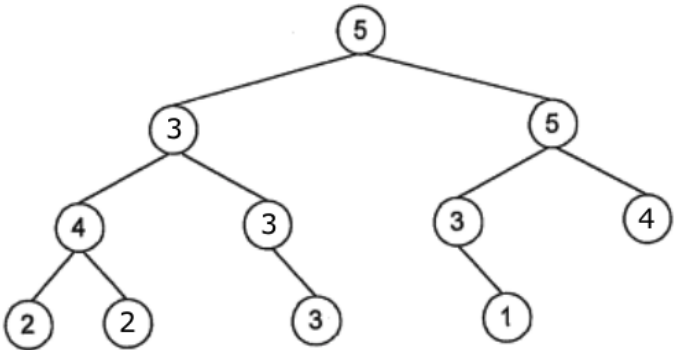
3.
```
def le_plus_loin(ligne):
    n = len(ligne)
    deb = ligne[0]
    fin = ligne[n-1]
    dmax = 0
    indice_max = 0
    for idx in range(1, n-1):
        p = ligne[idx]
        d = distance(p, deb, fin)
        if d > dmax:
            indice_max = idx
            dmax = d
    return (indice_max, dmax)
```

4.
```
def extrait(tab,i,j):
    t=[]
    for idx in range(i, j+1):
        t.append(tab[idx])
    return t
```

5.
```
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 :
            return simplifie(extrait(ligne,0,indice_max)) +
simplifie(extrait(ligne,indice_max,n))
```

Exercice 5

1. 16 (2+7+4+3)
2.
   a.
```
racine = Noeud(2)
sept = Noeud(7)
quatre = Noeud(4)
un = Noeud(1)
cinq = Noeud(5)
huit = Noeud(8)
racine.modifier_sag(sept)
racine.modifier_sad(cinq)
sept.modifier_sag(quatre)
sept.modifier_sad(un)
cinq.modifier_sad(huit)
```
   b.
   2
3.
```
def pgde_somme(self):
        if self.sag != None and self.sad != None:
            ag = self.sag.pgde_somme()
            ad = self.sad.pgde_somme()
            return self.etiquette+max(ag, ad)
        if self.sag != None:
            return self.sag.pgde_somme()+self.etiquette
        if self.sad != None:
            return self.sad.pgde_somme()+self.etiquette
        return self.etiquette
```
4.
   a.



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