Épreuve d'Informatique – Session 2019 – Filière PSI

Les candidats sont informés que la précision des raisonnements algorithmiques ainsi que le soin apporté à la rédaction et à la présentation des copies seront des éléments pris en compte dans la notation. Il convient en particulier de rappeler avec précision les références des questions abordées. Si, au cours de l'épreuve, un candidat repère ce qui peut lui sembler être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il est amené à prendre.

Remarques générales :

- Cette épreuve est composée d'un exercice et d'un problème indépendants ;
- Toutes les instructions et les fonctions demandées seront écrites en Python ;
- Les questions non traitées peuvent être admises pour aborder les questions ultérieures ;
- Toute fonction peut être décomposée, si nécessaire, en plusieurs fonctions.

≈ ≈

Exercice : (4 points)

Somme et factoriel

1 pt **Q1-** Écrire la fonction `factoriel(k)` qui reçoit en paramètre un entier positif **k** et qui renvoie la valeur du factoriel de **k** : **k! = 1 * 2 * 3 * … * k**.

Exemples :

- La fonction **factoriel (5)** renvoie le nombre : **120 = 1 * 2 * 3 * 4 * 5**
- La fonction **factoriel (0)** renvoie le nombre : **1**

0.5 pt **Q2-** Déterminer la complexité de la fonction **factoriel (k)**, et justifier votre réponse.

1.5 pt **Q3-** Écrire la fonction `som_fact(L)` qui reçoit en paramètre une liste **L** de nombres entiers positifs. La fonction renvoie la somme des factoriels des éléments de **L**.

Exemple :

L = [5, 3, 0, 6, 1]

La fonction **som_fact (L)** renvoie la valeur de la somme : **5! + 3! + 0! + 6! + 1!**

1 pt **Q4-** Déterminer la complexité de la fonction **som_fact (L)**, et justifier votre réponse.

≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈ ≈

Problème :

Réseau routier

Figure 1 : Extrait du réseau routier du Maroc

Un réseau routier peut être représenté par un dessin qui se compose de points et de traits continus reliant deux à deux certains de ces points : les **points** sont les **villes**, et les **lignes** sont les **routes**. On considère que toutes les routes sont à double sens. Chaque route peut être affectée par une valeur qui peut représenter le temps ou la distance entre deux villes, …

Étant donné un réseau routier, on pourra s'intéresser à la résolution des problèmes suivants :

- Quel est le plus court chemin entre deux villes ?
- Entre deux villes, quel est le nombre de chemins passant par un nombre de routes donné ?
- Est-il possible de passer par toutes les villes du réseau, sans passer deux fois par une même ville ?, si oui, quel est le plus court chemin ?
- Entre deux villes, quel est le chemin ayant le moindre coût ?
- …

Partie I :

Modélisation d'un réseau routier

On considère un réseau routier composé de **n** villes (avec **n ≥ 2**). Les villes du réseau routier sont numérotées par des entiers allant de **0** à **n-1**.

Exemple :

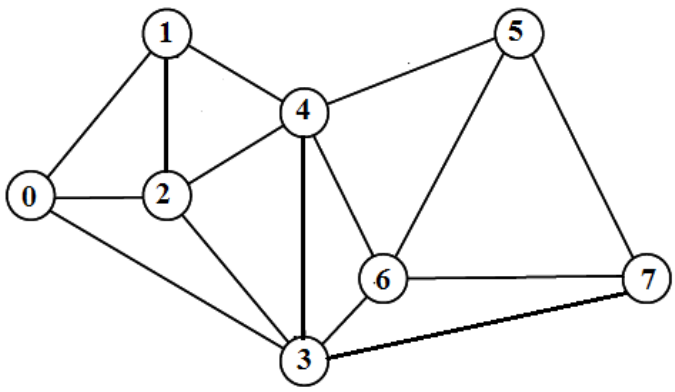

Figure 2 : Réseau routier composé de **15** routes, et de **8** villes numérotées de **0** à **7**.

Pour plus de clarté, tous les exemples de ce problème seront appliqués sur le réseau routier de la **figure 2**.

Représentation du réseau routier

Pour représenter un réseau routier de **n** villes, on utilise une matrice symétrique **R** d'ordre **n** (**n** lignes et **n** colonnes), telle que :

Pour toutes les villes **i** et **j**, telles que **0 ≤ i < n** et **0 ≤ j < n**, on a :

- `R[i,j]= R[j,i]=1`, s'il existe une route qui relie entre la ville **i** et la ville **j** ;
- `R[i,j]= R[j,i]=0`, sinon.

Exemple :

Le réseau routier de la **figure 2** est représenté par la matrice symétrique **R**, d'ordre **8**, suivante :

i/j	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
0	0	1	1	1	0	0	0	0
1	1	0	1	0	1	0	0	0
2	1	1	0	1	1	0	0	0
3	1	0	1	0	1	0	1	1
4	0	1	1	1	0	1	1	0
5	0	0	0	0	1	0	1	1
6	0	0	0	1	1	1	0	1
7	0	0	0	1	0	1	1	0

***NB :** Dans la matrice symétrique **R**, les lignes **i** et les colonnes **j** représentent les villes du réseau routier.*

I. 1- Représentation de la matrice du réseau routier en Python

Pour représenter la matrice symétrique **R** d'ordre **n**, on utilise une liste composée de **n** listes qui sont toutes de même longueur **n**.

Exemple :

La matrice symétrique **R,** du réseau routier de la **figure 2**, est représentée par la liste **R**, composée de **8** listes, de taille **8** chacune :

```
R = [ [0,1,1,1,0,0,0,0],[1,0,1,0,1,0,0,0],[1,1,0,1,1,0,0,0],
      [1,0,1,0,1,0,1,1],[0,1,1,1,0,1,1,0],[0,0,0,0,1,0,1,1],
      [0,0,0,1,1,1,0,1],[0,0,0,1,0,1,1,0]
    ]
```

`R[i][j]` est l'élément de **R**, à la **i^{ème}** ligne et la **j^{ème}** colonne.
Exemples : **R[0][0]** est l'élément : **0**, et **R[0][2]** est l'élément : **1**

`R[i]` est la ligne d'indice **i** dans **R**.
Exemple : **R[0]** est la liste **[0 , 1 , 1 , 1 , 0 , 0 , 0 , 0]**, qui représente la ligne d'indice **0** dans la matrice **R**.

Q 1- À partir de la matrice symétrique **R** de la **figure 2**, donner les résultats des expressions suivantes :

R[4][2] , R[1] , len(R[2]) , len(R)

I. 2- Villes voisines

i *et* ***j*** *sont deux villes dans un réseau routier représenté par une matrice symétrique* ***R****.*

Les villes ***i*** *et* ***j*** *sont* ***voisines****, s'il existe une route entre la vaille* ***i*** *et la ville* ***j****.*

Q 2. a- Écrire la fonction **`voisines(i,j,R)`**, qui reçoit en paramètres deux villes **i** et **j** d'un réseau routier représenté par la matrice symétrique **R**. La fonction renvoie **True** si les villes **i** et **j** sont voisines, sinon, la fonction renvoie **False**.

Exemples :

- La fonction **voisines (3, 0, R)** renvoie **True** ; *(les villes 3 et 0 sont voisines)*
- La fonction **voisines (3, 5, R)** renvoie **False**. *(les villes 3 et 5 ne sont pas voisines)*

Q 2. b- Écrire la fonction **`list_voisines(i,R)`**, qui reçoit en paramètres une ville **i** d'un réseau routier représenté par la matrice symétrique **R**. La fonction renvoie la liste de toutes les villes voisines à la ville **i**.

Exemple :

La fonction **list_voisines (3, R)** renvoie la liste des villes voisines à la ville **3** : **[0 , 2 , 4 , 6 , 7]**

I. 3- Degré d'une ville

Dans un réseau routier, le ***degré*** *d'une ville* ***i*** *est le nombre de villes voisines à la ville* ***i****.*

Q 3- Écrire la fonction **`degre(i,R)`**, qui reçoit en paramètres une ville **i** d'un réseau routier représenté par la matrice symétrique **R**. La fonction renvoie le degré de la ville **i**.

Exemples :

- La fonction **degre (3, R)** renvoie le nombre : **5** *(La ville* ***3*** *possède* ***5*** *villes voisines)*
- La fonction **degre (0, R)** renvoie le nombre : **3** *(La ville* ***0*** *possède* ***3*** *villes voisines)*
- La fonction **degre (2, R)** renvoie le nombre : **4** *(La ville* ***2*** *possède* ***4*** *villes voisines)*

I. 4- Liste des degrés des villes

Q 4. Écrire la fonction **`liste_degres(R)`**, qui reçoit en paramètre la matrice symétrique **R** représentant un réseau routier. La fonction renvoie une liste **D** contenant des tuples. Chaque tuple de **D** est composé de deux éléments : une **ville** du réseau routier, et le **degré** de cette ville.

Exemple :

La fonction **liste_degres (R)** renvoie la liste **D** suivante :

D = [(0 , 3**) , (1 ,** 3**) , (2 ,** 4**) , (3 ,** 5**) , (4 ,** 5**) , (5,** 3**) , (6 ,** 4**) , (7 ,** 3**)]**

I. 5- Tri des villes

Q 5. a- Écrire la fonction **`tri_degres(D)`**, qui reçoit en paramètre la liste **D** des degrés des villes. La fonction trie les tuples de la liste **D** dans l'ordre décroissant des degrés des villes.

Exemple :

D = [(0 , 3**) , (1 ,** 3**) , (2 ,** 4**) , (3 ,** 5**) , (4 ,** 5**) , (5,** 3**) , (6 ,** 4**) , (7 ,** 3**)]**

Après l'appel de la fonction **tri_degres (D)**, on obtient la liste suivante :

[(3 , 5**) , (4 ,** 5**) , (2 ,** 4**) , (6 ,** 4**) , (0 ,** 3**) , (5 ,** 3**) , (1 ,** 3**) , (7 ,** 3**)]**

Q 5. b- Écrire la fonction **`tri_villes(R)`**, qui reçoit en paramètre la matrice symétrique **R** représentant un réseau routier. La fonction renvoie la liste des villes triées dans l'ordre décroissant des degrés des villes.

Exemple :

La fonction **tri_villes (R)** renvoie la liste des villes triées dans l'ordre décroissant des degrés :

[3 , 4 , 2 , 6 , 0 , 5 , 1 , 7]

Partie II :

Coloration optimale des villes d'un réseau routier

Une **coloration** des villes du réseau routier est une affectation de couleurs à chaque ville, de façon à ce que deux villes voisines soient affectées par deux couleurs différentes. Le nombre minimum de couleurs nécessaires pour colorier un réseau routier est appelé : ***nombre chromatique***.

La recherche du nombre chromatique est une question qui intervient dans beaucoup de problèmes concrets :

- ✓ L'établissement d'emplois du temps ;
- ✓ la gestion de ressources partagées ;
- ✓ la compilation dans l'utilisation efficace des registres ;
- ✓ ...

On cherche à construire une liste **C** qui contiendra les couleurs des villes. Ces couleurs seront représentés par des entiers strictement positifs : *chaque élément **C[k]** contiendra la couleur de la ville **k** du réseau routier.*

Pour construire la liste **C** des couleurs, on propose d'utiliser un algorithme, appelé : ***algorithme de glouton***. C'est un algorithme couramment utilisé dans la résolution de ce genre de problèmes, afin d'obtenir des solutions optimales.

Principe de l'algorithme de glouton :

V est la liste des villes triées dans l'ordre décroissant des degrés des villes
C est une liste de même taille que **V**, initialisée par des **0**
Pour toute ville **k** de **V :**
    C[k] ← première couleur non utilisée par les villes voisines à la ville **k**
Retourner la liste **C**

II. 6- Premier entier qui n'existe pas dans une liste d'entiers

Q 6- Écrire la fonction **`premier_entier(L)`**, qui reçoit en paramètre une liste **L** de nombres entiers positifs. La fonction renvoie le premier entier positif qui n'appartient pas à la liste **L**.
Exemple :

La fonction **premier_entier ([0 , 0 , 3 , 1 , 0 , 1 , 0 , 0])** renvoie le nombre : **2**

II. 7- Liste des couleurs des villes voisines à une ville

Q 7- Écrire la fonction **`couleurs_voisines(k,C,R)`**, qui reçoit en paramètres une ville **k** d'un réseau routier représenté par la matrice symétrique **R**, et la liste **C** des couleurs des villes du réseau**.** La fonction renvoie la liste des **C[i]** telle que **i** est une ville voisine à la ville **k**.
Exemples :

- La fonction **couleurs_voisines (4, [0, 0, 0, 1, 0, 0, 0, 0], R)** renvoie la liste : **[0, 0, 1, 0, 0]**
- La fonction **couleurs_voisines (2, [0, 0, 0, 1, 2, 0, 0, 0], R)** renvoie la liste: **[0, 0, 1, 2]**
- La fonction **couleurs_voisines (6, [0, 0, 3, 1, 2, 0, 0, 0], R)** renvoie la liste: **[1, 2, 0, 0]**
- La fonction **couleurs_voisines (0, [0, 0, 3, 1, 2, 0, 3, 0], R)** renvoie la liste: **[0, 3, 1]**

II. 8- Coloration des villes

Q 8- Écrire la fonction **`couleurs_villes(R)`**, qui reçoit en paramètre la matrice symétrique **R** représentant un réseau routier. La fonction renvoie la liste **C** des couleurs des villes, en utilisant le principe de glouton cité ci-dessus.

Exemple :

La fonction **couleurs_villes (R)** renvoie la liste des couleurs : **C = [2 , 1 , 3 , 1 , 2 , 1 , 3 , 2]**

Dans cette liste **C** :

- Les villes **0**, **4** et **7** ont la même couleur : **2** ;
- Les villes **1**, **3** et **5** ont la même couleur : **1** ;
- Les villes **2** et **6** ont la même couleur : **3**.

*NB : Dans cet exemple, le nombre chromatique est **3**.*

Partie III :

Chemins dans un réseau routier

III. 9- Chemin entre deux villes

Dans un réseau routier, un **chemin** est un tuple **T** qui contient des villes du réseau, et qui satisfait les deux conditions suivantes :

- **c1** : Le tuple **T** contient au moins deux villes du réseau routier ;
- **c2** : Deux villes consécutives dans **T** sont voisines.

Exemples :

- Le tuple **(2**, 0, 1, 4**, 3)** est un chemin entre la ville **2** et la ville **3**, composé de **4** routes ;
- Le tuple **(7,** 6, 4, 5, 6**,** 3**, 4)** est un chemin entre la ville **7** et la ville **4**, composé de **6** routes ;
- Le tuple **(** 3, 1, 4, 7 **)** n'est pas un chemin.

Q 9- Écrire la fonction **`chemin_valide(T,R)`**, qui reçoit en paramètres un tuple **T** contenant des villes du réseau routier représenté par la matrice symétrique **R.** La fonction renvoie **True** si le tuple **T** satisfait les deux conditions **c1** et **c2** citées ci-dessus, sinon, la fonction renvoie **False**.

III. 10- Chemin simple entre deux villes

*Dans un réseau routier, un **chemin simple** est un chemin qui passe une seule fois par la même ville.*

Exemples :

- Le chemin **(2**, 0, 1, 4**, 3)** est un chemin simple entre la ville **2** et la ville **3** ;
- Le chemin **(6**, 7, **3**, 4, 2, **3**, **6**, 5 **)** n'est pas simple.

Q 10- Écrire la fonction **`chemin_simple(T,R)`**, qui reçoit en paramètres un chemin **T** dans un réseau routier représenté par la matrice symétrique **R.** La fonction renvoie **True** si le chemin **T** est un simple, sinon, la fonction renvoie **False**.

III. 11- Plus court chemin entre deux villes

On suppose que la fonction **`liste_chemins(i,j,R)`**, reçoit en paramètres deux villes **i** et **j** d'un réseau routier représenté par la matrice symétrique **R.** Cette fonction renvoie une liste qui contient tous les chemins simples entre la ville **i** et la ville **j.**

Exemple :

La fonction **liste_chemins (1, 5, R)** renvoie la liste de tous les chemins simples entre les villes **1** et **5** :

[(**1**,0,2,3,4,**5**),(**1**,0,2,3,4,6,**5**),(**1**,0,2,3,4,6,7,**5**),(**1**,0,3,4,6,7,**5**),
(**1**,0,3,6,4,**5**),(**1**,0,3,6,**5**),(**1**,0,3,6,7,**5**),(**1**,2,0,3,6,**5**),(**1**,2,0,3,6,7,**5**),
(**1**,2,0,3,7,**5**),(**1**,2,0,3,7,6,4,**5**),(**1**,2,3,7,6,**5**),(**1**,4,2,0,3,7,**5**),(**1**,4,2,0,
3,7,6,**5**),(**1**,4,2,3,6,**5**),(**1**,4,2,3,6,7,**5**),(**1**,4,2,3,7,**5**),(**1**,4,2,3,7,6,**5**),(
1,4,3,6,**5**),(**1**,4,3,6,7,**5**),(**1**,4,3,7,**5**),(**1**,4,3,7,6,**5**),(**1**,4,**5**),(**1**,4,6,3,7,
5),(**1**,4,6,**5**), … ,(**1**,4,6,7,**5**)**]**

*Le **plus court chemin** entre une ville **i** et une ville **j** est un chemin simple entre la ville **i** et la ville **j**, et qui traverse le moins de villes.*

***NB** : On peut trouver plusieurs plus courts chemins entre deux villes.*

Q 11- Écrire la fonction **`pc_chemins(i,j,R)`**, qui reçoit en paramètres deux villes **i** et **j** d'un réseau routier représenté par la matrice symétrique **R.** Cette fonction renvoie la liste de tous les plus courts chemins entre la ville **i** et la ville **j**.

Exemples :

- La fonction **pc_chemins (0, 7, R)** renvoie la liste [(**0** , 3 , **7**)] ;
- La fonction **pc_chemins (7, 1, R)** renvoie la liste :
  [(**7** , 3 , 0 , **1**) , (**7** , 3 , 2 , **1**) , (**7** , 3 , 4 , **1**) , (**7** , 5 , 4 , **1**) , (**7** , 6 , 4 , **1**)].

Partie IV :

Calcul du nombre de chemins entre deux villes

Dans cette partie, on considère que le module **numpy** est importé :

```
from numpy import *
```

On suppose que le réseau routier est représenté par la matrice symétrique **R**, et que la matrice **R** est créée par la méthode **array()** du module **numpy**.

Exemple :

```
R = array([ [0,1,1,1,0,0,0,0],[1,0,1,0,1,0,0,0],[1,1,0,1,1,0,0,0],
            [1,0,1,0,1,0,1,1],[0,1,1,1,0,1,1,0],[0,0,0,0,1,0,1,1],
            [0,0,0,1,1,1,0,1],[0,0,0,1,0,1,1,0] ])
```

Dans cette partie, on propose de résoudre le problème suivant :

Soit **p** un entier strictement positif.

Quel est le nombre de chemins (simples ou non) entre une ville **i** et une ville **j**, passant exactement par **p** routes ?

Par exemple, dans le réseau routier de la **figure 2**, pour aller de la ville **1** à la ville **3**, en passant exactement par **5** routes, on peut trouver plusieurs chemins, dont voici quelques uns :

(1,4,5,7,6,3) , (1,4,6,5,4,3) , (1,2,3,4,2,3) , (1,0,2,0,2,3), …

Pour résoudre ce problème, on doit procéder ainsi :

- Calculer la matrice $\mathbf{M = R^p}$ *(matrice symétrique **R** élevée à la puissance **p**)*
- Chaque élément $\mathbf{M[i][j]}$ représente le nombre de chemins, entre la ville **i** et la ville **j**, passant par **p** routes ;
- La matrice $\mathbf{M}$ est symétrique aussi. Le nombre de chemins, entre la ville **i** et la ville **j**, est le même que celui entre la ville **j** et la ville **i**.

Exemples :

$$\begin{bmatrix} 3 & 1 & 2 & 1 & 3 & 0 & 1 & 1 \\ 1 & 3 & 2 & 3 & 1 & 1 & 1 & \boxed{0} \\ 2 & 2 & 4 & 2 & 2 & 1 & 2 & 1 \\ 1 & 3 & 2 & 5 & 2 & 3 & 2 & 1 \\ 3 & 1 & 2 & 2 & 5 & 1 & 2 & 3 \\ 0 & 1 & 1 & 3 & 1 & 3 & 2 & 1 \\ 1 & 1 & 2 & 2 & 2 & 2 & 4 & 2 \\ 1 & \boxed{0} & 1 & 1 & 3 & 1 & 2 & 3 \end{bmatrix}$$

$\mathbf{M = R^2}$

$$\begin{bmatrix} 4 & 8 & 8 & 10 & 5 & 5 & 5 & 2 \\ 8 & 4 & 8 & 5 & 10 & 2 & 5 & \boxed{5} \\ 8 & 8 & 8 & 11 & 11 & 5 & 6 & 5 \\ 10 & 5 & 11 & 8 & 15 & 5 & 11 & 10 \\ 5 & 10 & 11 & 15 & 8 & 10 & 11 & 5 \\ 5 & 2 & 5 & 5 & 10 & 4 & 8 & 8 \\ 5 & 5 & 6 & 11 & 11 & 8 & 8 & 8 \\ 2 & \boxed{5} & 5 & 10 & 5 & 8 & 8 & 4 \end{bmatrix}$$

$\mathbf{M = R^3}$

Par exemple, entre la ville **1** et la ville **7** :

- Il y a **0** chemins qui passent exactement par **deux** routes (*dans la matrice* $M = R^2$) ;
- Il y a **5** chemins qui passent exactement par **trois** routes (*dans la matrice* $M = R^3$).

IV. 12- Produit matriciel

Rappel :

On considère la matrice **A**, de **m** lignes et **n** colonnes, et la matrice **B** de **n** lignes et **p** colonnes. En effectuant le produit matriciel de **A** et **B**, on obtient la matrice **C** de **m** lignes et **p** colonnes, sachant que les coefficients de la matrice **C** sont calculés par la formule suivante :

$$C_{ij} = \sum_{k=0}^{n-1} A_{ik} * B_{kj}$$

Q 12. Écrire la fonction `produit_matriciel(A,B)`, qui reçoit en paramètres deux matrices symétriques **A** et **B**, de même dimension. La fonction calcule et renvoie une nouvelle matrice symétrique **C**, qui contient le produit matriciel de **A** et **B**. *La fonction doit économiser le temps de calcul, en calculant une seule fois les coefficients symétriques dans la matrice* ***C***.

IV. 13- Calcule de la puissance par 'exponentiation rapide'

Lorsqu'il s'agit d'une matrice carrée, le calcul de la puissance devient crucial. **L'exponentiation rapide** est un algorithme qui permet de minimiser le nombre de multiplications effectuées dans le calcul de la puissance.

Pour calculer x^n, le principe de l'exponentiation rapide est le suivant :

- Si **n = 1** alors $x^n = x$
- Si **n est pair** alors $x^n = (x^2)^{n/2}$
- Si **n est impair** alors $x^n = x \,.\, (x^2)^{(n-1)/2}$

Q 13- Écrire la fonction `puissance(R,n)`, reçoit en paramètres la matrice symétrique **R** représentant un réseau routier, et un entier strictement positif **n.** En utilisant le principe de l'exponentiation rapide, la fonction calcule et renvoie la matrice $\mathbf{R^n}$.

Partie V :

Base de données et langage SQL

Réseau routier pondéré :

On associe à chaque route du réseau routier une valeur appelée : **poids**. Le poids de chaque route est la longueur de cette route, exprimée en km (kilomètre). Ainsi, le réseau routier est appelé : *réseau routier pondéré.*

Exemple :

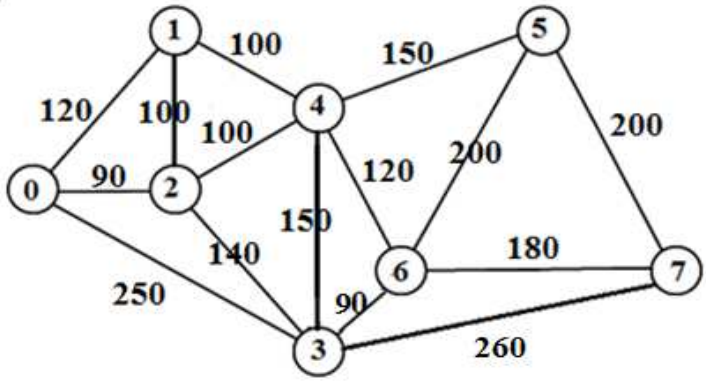

Figure 3 : Réseau routier pondéré par les longueurs des routes en km

Le coût d'un chemin :

*Dans un réseau routier pondéré, le **coût d'un chemin** est égale à la somme des poids des routes qui composent ce chemin.*

Exemples :

Dans le réseau routier pondéré de la figure 3 :

- Le coût du chemin `(1,0,3,6)` est : **460** = 120+250+90
- Le coût du chemin `(4,6,5,7,3)` est : **780** = 120+200+200+260
- Le coût du chemin `(0,1,2,3,4,5,6,7)` est : **1040** = 120+100+140+150+150+200+180

Dans cette partie, on propose de manipuler le réseau routier pondéré par une base de données composée de deux tables : la table : **Villes** et la table : **Chemins**.

Villes	
code	*(entier)*
nom	*(texte)*
couleur	*(entier)*

Chemins	
chemin	*(texte)*
nb_routes	*(entier)*
cout	*(réel)*
ville_depart	*(entier)*
ville_arrivee	*(entier)*

Structure de la table '**Villes**' :

La table **Villes** contient les villes du réseau routier.

- Le champ **code** contient les entiers : 0, 1, 2, 3, …, **n**-1, sachant que **n** est le nombre de villes dans le réseau routier. Chaque entier représente une seule ville du réseau routier.
- Le champ **nom** contient les noms des villes ;
- Le champ **couleur** contient des entiers strictement positifs, qui représentent les couleurs des villes.

Exemples d'enregistrements dans les tables '**Villes**' :

code	nom	couleur
0	Agadir	2
1	Essaouira	1
2	Marrakech	3
3	Ouarzazate	1
4	Safi	2
20	Tanger	7
…	…	…

Structure de la table '**Chemins**' :

La table **Chemins** est composée de **5** champs :

- Le champ **chemin** contient tous les chemins simples qui existent entres les villes du réseau routier. Chaque chemin est représenté par une chaîne de caractères, qui contient les villes du chemin, séparées par le caractère **'–'** (tiré de 6).

  Exemple : La chaîne `'1-0-3-6'` représente le chemin `(1,0,3,6)`
- Le champ **nb_routes** contient le nombre de routes dans chaque chemin ;
- Le champ **cout** contient les coûts des chemins ;
- Le champ **ville_depart** contient le code de la ville de départ de chaque chemin ;
- Le champ **ville_arrivee** contient le code de la ville d'arrivée de chaque chemin.

Exemples d'enregistrements dans la table '**Chemins**' :

chemin	nb_routes	cout	ville_depart	ville_arrivee
0-1	1	120	0	1
1-0	1	120	1	0
2-4-1	2	200	2	1
1-4-2	2	200	1	2
1-0-3-6	3	460	1	6
4-6-5-7-3	4	780	4	3
0-1-2-3-4-5-6-7	7	1040	0	7
1-4-5-7-6-3	5	720	1	3
5-7-6-4-1-0-2-3	7	950	5	3
…	…	…	…	

V. 14- Déterminer les clés primaires et les clés étrangères dans les tables **Villes** et **Chemins**.

Justifier votre réponse.

V. 15- Écrire, en algèbre relationnelle, la requête qui donne le résultat suivant :

Les codes et noms des villes ayant la couleur **1** ou la couleur **2** dans le réseau routier.

Écrire, en langage SQL, les requêtes qui donne les résultats suivants :

V. 16- Le nombre de routes dans le réseau routier.

V. 17- Le nombre chromatique du réseau routier.

V. 18- Les noms des villes et le degré de chaque ville, ayant le degré compris strictement entre **4** et **9**, triés dans l'ordre décroissant des degrés.

V. 19- Modifier les coûts de tous les chemins qui passent par la route **"0-1"**, en ajoutant la valeur **50** à chacun de ces chemins.

V. 20- Le plus petit coût des chemins qui passent par toutes les villes du réseau routier.

≈ ≈ ≈ ≈ ≈ ≈ FIN DE L'ÉPREUVE ≈ ≈ ≈ ≈ ≈ ≈