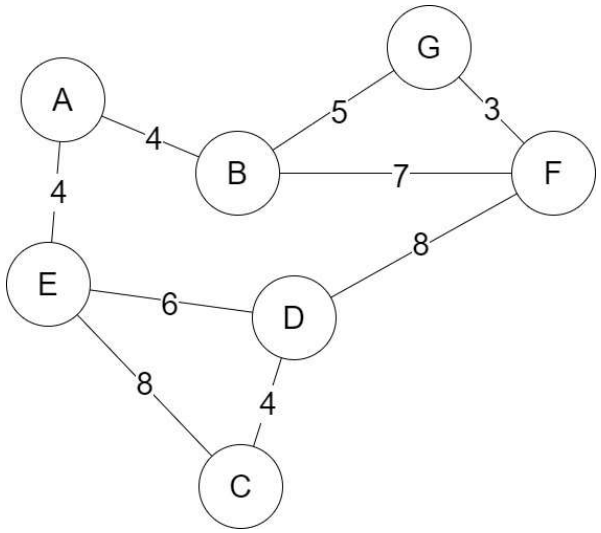
Exercice 3 : Société CarteMap

1. Graphe pondéré G1 :



2. Le chemin le plus court entre A et D est A-E-D pour une distance de 10.

3. Matrice d'adjacence de G1 :

	A	B	C	D	E	F	G
A	0	4	-	-	4	-	-
B	4	0	-	-	-	7	3
C	-	-	0	4	8	-	-
D	-	-	4	0	6	8	-
E	4	-	9	6	0	-	-
F	-	7	-	8	-	0	3
G	-	3	-	-	-	3	0

4. Exemple de représentation du graphe G2 en python :

```
g2 = {'A':['B', 'C', 'H'],
      'B':['A', 'I'],
      'C':['A', 'D', 'E'],
      'D':['C', 'E'],
      'E':['C', 'D', 'G'],
      'F':['G', 'I'],
      'G':['E', 'F', 'H'],
      'H':['A', 'G', 'I'],
      'I':['B', 'H', 'F']}
```

5. Exemple d'un parcours en largeur du graphe G2 issu de A :
A-B-C-H-D-E-I-G-F

6. La fonction `cherche_itinéraires` est récursive car elle s'appelle elle-même (ligne 8).

7. La fonction `cherche_itinéraires` permet de déterminer tous les chemins possibles pour se rendre d'un point à un autre par une recherche en profondeur. Les chemins sont stockées dans la variable `tab_itineraires`.

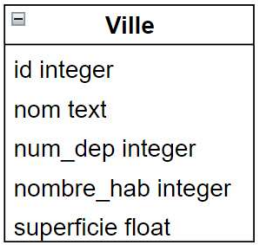
8. La fonction `itineraires_court` permet de sélectionner les plus courts des chemins obtenus par la fonction `cherche_itinéraires`.

```
def itineraires_court(G, dep, arr):
    cherche_itineraires(G, dep, arr)
    tab_court = []
    mini = float('inf')
    for v in tab_itineraires:
        if len(v) <= mini:
            mini = len(v)
    for v in tab_itineraires:
        if len(v) == mini:
            tab_court.append(v)
    return tab_court
```

9. Lors du deuxième appel de `itineraires_court`, la variable `tab_itineraires` n'a pas été réinitialisée. Ainsi elle contient toujours le chemin ['A', 'C', 'E'] du précédent appel. Lors du deuxième appel, tous les autres chemins ajoutés à `tab_itineraires` sont plus longs. Ainsi, c'est le chemin ['A', 'C', 'E'] qui demeure le plus court et qui est le seul à être retourner par le deuxième appel de `itineraires_court`.

10. Le choix d'utiliser un SGBD permet d'éviter la redondance des données et d'assurer leur cohérence.

11. Le schéma relationnel de la table **ville** est :
ville(**id** integer, **nom** text, **num_dep** integer, **nombre_hab** integer, **superficie** float)
ou sous la forme d'une table :



12. L'attribut **id_ville** sert de clé étrangère dans la table **sport** en pointant vers l'attribut **id** (clé primaire) de la table **ville**.

13. Le résultat de la requête SQL est : `[{'nom':'Chamonix'}]`

14. `SELECT nom FROM sport WHERE type = 'piscine';`

15. `UPDATE sport SET note = 7 WHERE id = 3;`
ou :
`UPDATE sport SET note = 7 WHERE nom = 'Ballons perdus';`

16. `INSERT INTO ville VALUES(8, 'Toulouse', 31, 471941, 118);`
ou :
`INSERT INTO ville (id, nom, num_dep, nombre_hab, superficie)`
`VALUES (8, 'Toulouse', 31, 471941, 118);`

17.
`SELECT sport.nom FROM sport JOIN ville ON sport.id_ville=ville.id`
`WHERE ville.nom='Annecy' AND sport.type='mur d'escalade';`