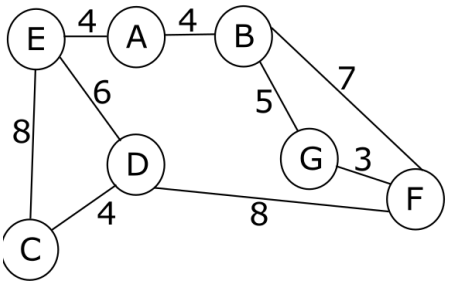
1. 

2. A -> E -> D avec 10 Km

3.
```
0400400
4000075
0004800
0040680
4086000
0708003
0500030
```

4.
```
d = {'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', 'F', 'H']}
```

5. A-B-C-H-D-E-G-I-F

6. à la ligne 8 la fonction `cherche_itineraires` s'appelle elle-même, la fonction `cherche_itineraires` peut donc être qualifiée de fonction récursive.

7. La fonction `cherche_itineraires` permet de recenser tous les itinéraires (tous les chemins) pour aller de la ville `start` à la ville `end`.

8.
```
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. Le problème vient de la liste Python `tab_itineraires`. Cette liste est vide au début de l'exécution du programme p1 (ligne 1). Tous les itinéraires possibles pour aller de la ville `start` à la ville `end` sont ajoutés à cette liste grâce à l'appel à la fonction `cherche_itineraires` (ligne 14).
   À la suite de l'appel de la fonction `itineraires_court(G2, 'A', 'E')` depuis la console, la liste `tab_itineraires` contient les itinéraires suivants :
   `[['A', 'B', 'I', 'F', 'G', 'E'], ['A', 'B', 'I', 'H', 'G', 'E'], ['A', 'C', 'D', 'E'], ['A', 'C', 'E'], ['A', 'H', 'G','E'], ['A', 'H', 'I', 'F', 'G', 'E']]`
   Quand on appelle la fonction `itineraires_court(G2, 'A', 'F')` toujours depuis la console, la liste `tab_itineraires` n'a pas été "vidée" (puisque le programme p1 n'a pas été exécuté de nouveau), elle contient donc toujours, entre

autres, l'itinéraire `['A', 'C', 'E']`. La fonction `itineraires_court` permet de choisir l'itinéraire le plus court parmi ceux qui se trouvent dans la liste `tab_itineraires`. Sachant que les itinéraires pour aller de A à F ont au minimum une taille de 4, c'est donc l'itinéraire `['A', 'C', 'E']` qui est renvoyé par la fonction, d'où le problème constaté. L'exécution du programme p1 entre les deux appels fait disparaitre le problème puisque la liste est vidée lors de la 2e exécution.

10. L'utilisation d'un SGBD apporte beaucoup d'avantage par rapport à l'utilisation d'un simple fichier au format texte :
    - les SGBD permettent de gérer les autorisations d'accès.
    - les accès concurrents sont gérés par les SGBD
    - les SGBD proposent aussi de gérer la redondance des données (pour palier aux problèmes de pannes ou de surcharges)
11. ville(id : INT, nom : TEXT, num_dep : INT, nombre_hab : INT, superficie : FLOAT)
12. `id_ville` est une clé étrangère, elle permet d'effectuer une jointure entre la table `sport` et la table `ville`
13. On obtient : Chamonix
14. ```
    SELECT nom
    FROM sport
    WHERE type = 'piscine';
    ```
15. ```
    UPDATE sport
    SET note = 7
    WHERE nom = 'Ballon perdus';
    ```
16. ```
    INSERT INTO ville
    VALUES
    (8, 'Toulouse', 31, 471941, 118);
    ```
17. ```
    SELECT sport.nom
    FROM sport
    JOIN ville ON ville.id = id_ville
    WHERE ville.nom = 'Annecy';
    ```