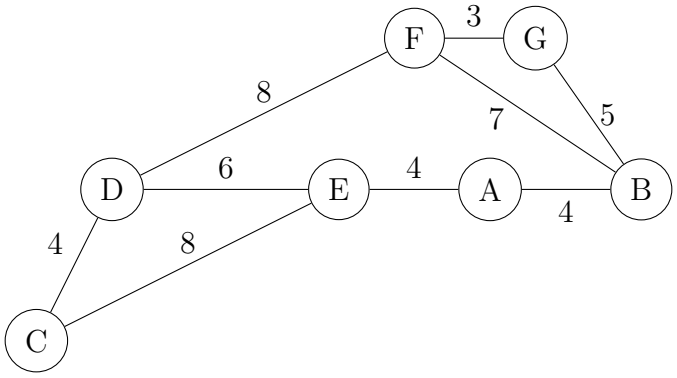
Devoir surveillé n°7 – NSI
Correction

Exercice 1

1. Voici une représentation du graphe G1 :



2. Le chemin le plus court entre les villes A et D est `A-E-D`, qui mesure 10 km.
L'usage de l'algorithme de Dijkstra n'est pas exigé ici, mais :

A	B	C	D	E	F	G	Choix
0	∞	∞	∞	∞	∞	∞	A(0)
	4(A)	∞	∞	4(A)	∞	∞	B(4)
		∞	∞	4(A)	11(B)	9(B)	E(4)
		12(E)	10(E)		11(B)	9(B)	G(9)
		12(E)	10(E)		~~12(G)~~ 11(B)		D(10)

3. La matrice d'adjacence du graphe G1 (avec les sommets triés par ordre alphabétique) est :

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

4. Voici une implémentation en Python du graphe G2 à l'aide d'un dictionnaire :

```
G2 = {'A': ['B', 'C', 'H'], 'B': ['A', 'I'], 'C': ['A', 'D', 'E'],
      'D': ['C', 'E'], 'E': ['C', 'D', 'G'], 'F': ['I', 'G'],
      'G': ['E', 'F', 'H'], 'H': ['A', 'G', 'I'],
      'I': ['B', 'F', 'H']}
```

5. Le parcours en largeur du graphe G2 en partant de A donne la liste suivante :
`[A, B, C, H, I, D, E, G, F]`

6. La fonction `cherche_itineraires` est récursive car elle fait appel à elle-même :
`cherche_itineraires(G, u, end, chaine)`

7. La fonction `cherche_itineraires` a pour rôle de remplir la liste `tab_itineraires` des itinéraires possibles entre `dep` et `arr`.

8. voici la fonction `itineraires_court` complétée :

```
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. La liste `tab_itineraires` est globale, donc elle conserve ses éléments entre deux exécutions de `itineraires_court` si l'on ne relance pas le programme complet.

   Quand on exécute la seconde fois, l'itinéraire le plus court précédent a été conservé, et il reste le plus court une fois qu'on ajoute les nouveaux. C'est donc ce même itinéraire qui est retourné.

   Il faudrait par exemple la réinitialiser à la liste vide au tout début du code de la fonction `itineraires_court`.

   Le mieux serait en fait de ne pas utiliser de variable globale, et que `cherche_itineraires` retourne une liste `tab_itineraires` plutôt que la liste vide.