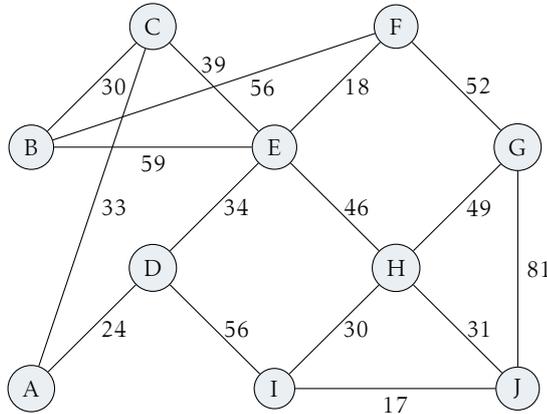


Objectifs de ce TP :

- utiliser l'algorithme de Dijkstra pour déterminer les plus courts chemins entre un sommet et tous les autres sommets du graphe,
- puis utiliser ce qui précède pour déterminer le plus court chemin entre un sommet de départ et un sommet d'arrivée bien définis.

On considère le graphe suivant :



1. Récupérer la matrice des poids associée à ce graphe.
2. Écrire la liste des sommets dans une chaîne de caractères nommée S.
3. Définir une fonction `initialisation(s)` qui prend en argument d'entrée un sommet $s \in S$ et renvoie trois listes à n éléments (où n est l'ordre du graphe) :
 - la liste LDM (liste des distances minimales) contenant les distances de s aux autres sommets, initialisée à 0 pour s et ∞ pour les autres sommets,
 - la liste LSC (liste des sommets choisis) contenant `True` si le sommet est choisi, `False` sinon ; elle est initialisée à `False` partout, et `True` pour le sommet s ,
 - la liste LP (liste des prédécesseurs) ne contenant initialement que `None`.
4. Compléter le programme suivant de sorte que l'appel de `dijkstra(M, s)` renvoie les listes LDM, LSC, LP à la fin de l'exécution de l'algorithme de Dijkstra sur le graphe dont M est la matrice des poids en partant du sommet s.

PETITE REMARQUE

On note au passage la structure d'une matrice en Python : une liste de listes dans un environnement `np.array()` pour que chaque liste soit écrite sur une nouvelle ligne...

```

1 import numpy as np
2 inf=np.inf
3 S='ABCDEFGHJIJ' #liste des sommets
4
5 def initialisation(s):
6     depart=S.index(s)
7     .....
8     .....
9     .....
10    .....
11    .....
12    return LDM,LSC,LP
13
14 def dijkstra(M,s):
15     n=len(S) #n=ordre du graphe
16     LDM,LSC,LP=initialisation(s)
17     rang_choisi=.....
18     dist_choisi=0 #dist entre s et le sommet choisi
19     for nb in range(1,n):
20         minimum=...
21         for k in ..... #k=rang du sommet exploré
22             if .....
23                 dist_prov=.....
24                 if .....
25                     LDM[k]=.....
26                     LP[k]=.....
27                 if .....
28                     minimum=.....
29                     rang_prochain_choisi=.....
30     rang_choisi=.....
31     LSC[rang_choisi]=.....
32     dist_choisi=.....
33     return LDM,LSC,LP

```

5. Écrire une fonction `dijkstra_pcc(M,s_depart,s_arrivee)` qui renvoie le plus court chemin pour aller de `s_depart` à `s_arrivee` ainsi que sa longueur.

En guise de vérification des programmes :

- `initialisation('A')` renvoie : `([0, inf, inf, inf, inf, inf, inf, inf, inf, inf], [True, False, False, False, False, False, False, False, False, False], [None, None, None, None, None, None, None, None, None, None])`
- `dijkstra(Mpoids,'A')` renvoie `([0, 63.0, 33.0, 24.0, 58.0, 76.0, 128.0, 104.0, 80.0, 97.0], [True, True, True, True, True, True, True, True, True, True], [None, 'C', 'A', 'A', 'D', 'E', 'F', 'E', 'D', 'I'])`
- `dijkstra_pcc(Mpoids,'A','G')` renvoie `(['A', 'D', 'E', 'F', 'G'], 128.0)`