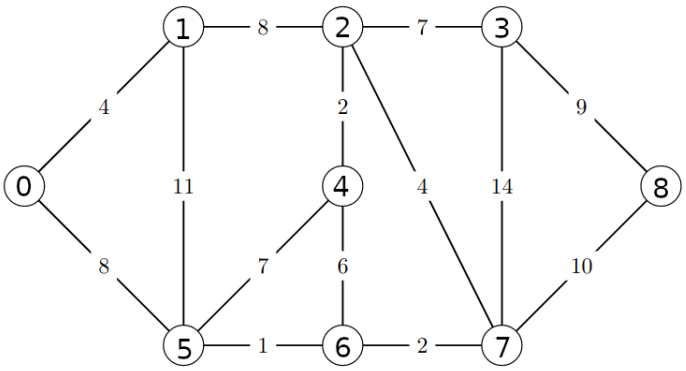
Informatique – TP23

Vésale Nicolas - Henrik Thys

Exercice 1: Création d'une matrice d'adjacences.

On considère le graphe suivant:



1. Proposer une valeur qui conviendrait pour la variable `inf`, qui représente la quantité « infinie ».
2. Créer une matrice de 10 lignes et colonnes `A`, dont toutes les cases valent `inf`. La modifier pour qu'elle représente la matrice d'adjacences de ce graphe.
3. En appliquant à la main l'algorithme de Djikstra, trouver la distance minimale entre `0` et `8`.

Exercice 2: Une amélioration de l'implémentation de l'algorithme de Djikstra.

Pour améliorer l'implémentation de l'algorithme de Djikstra, on va utiliser une **file de priorité** pour stocker chaque sommet avec une priorité, en l'occurrence la distance du plus court chemin trouvé pour l'instant entre `Si` et ce sommet. À chaque itération, on sélectionne le sommet ainsi stocké de priorité minimale. Pour ceci, on utilise le module suivant:

```
import heapq as hq
fp=[(1,'A')]#crée une file avec 'A' de priorité 1
#la complexité de toutes les opérations suivantes est O(ln(len(fp)))
hq.heappush(fp,(2,'B'))#ajoute 'B' avec priorité 2 à la file
hq.heappop(fp)#renvoie l'élément de plus petite priorité de la file
```

Complétez la fonction suivante pour implémenter l'algorithme de Djikstra à l'aide d'une file de priorité:

```
def Djikstra(A,Si,Sf):
    d=[inf for i in range(len(A[0]))]#liste des distances
    d[Si]=0
    fp=? #initialise la file
    while len(fp)!=0:
        #renvoie le sommet Sm de distance minimale dmin dans fp
        dmin,Sm=?
        for i in range(len(A[Sm])):
            if dmin+A[Sm][i]<d[i]:
                d[i]=dmin+A[Sm][i]
                ? #ajoute à la file (i,d[i])
    return d[Sf]#renvoie inf si on ne peut pas atteindre Sf
```

Pourriez-vous améliorer votre algorithme pour qu'il rende un chemin de distance minimale si il existe ?