"""

@author: JMG

"""

import numpy as np

#import sys

def ma3():

ma=[]

ligne = [0,10,20,0,30,0]

ma.append(ligne)

ligne = [10,0,30,50,10,0]

ma.append(ligne)

ligne = [20,30,0,10,0,30]

ma.append(ligne)

ligne = [0,50,10,0,0,10]

ma.append(ligne)

ligne = [30,10,0,0,0,0]

ma.append(ligne)

ligne = [0,0,30,10,0,0]

ma.append(ligne)

return ma

def Dijkstra(G,sdebut):

x,y = np.shape(G)

assert x==y

n=x

assert sdebut < n-1

dinfini = 1000

#Création de la liste des couples [d,sp] ( [distance,sommet précédent]) des sommets provisoirement pondérés

SProv = []

#Création de la liste des sommets marqués

S = []

#Création du tableau d'analyse

tableauAnalyse = []

#Initialisation de la liste des sommets provisoirement pondérés

T = list(range(0,n))

#Création de la première liste des couples [d,sp]

for i in T:

SProv.extend([[dinfini,0]])

#La distance du sommet de début est fixée à 0 et son prédecesseur est lui-même

SProv[sdebut][0] = 0

SProv[sdebut][1] = sdebut

#Le sommet de début est marqué définitivement et donc enlevé de la liste des sommets provisoires

T.remove(sdebut)

#Il est ajouté à l'ensemble des sommets marqués

S.append(sdebut)

#Chaque sommet adjacent est marqué du poids de l'arête la reliant à sdebut

for i in T:

print(i,G[sdebut][i])

if G[sdebut][i] != 0:

print ("Detection pour " ,i)

SProv[i][0] = G[sdebut][i]

SProv[i][1] = sdebut

print ("Initialisation", T,SProv)

#On termine en créant la première ligne du tableau d'analyse

tableauAnalyse.append(SProv) #([SProv.copy()])

ligne = 0 #ligne courante du tableau d'analyse

while len(T)>0:

#On recherche la distance la plus petite dans les distances marquées provisoirement

dmini = dinfini

for i in T:

if tableauAnalyse[-1][i][0] <= dmini:

dmini = tableauAnalyse[-1][i][0]

SProvTemp = i

sommetMarque = SProvTemp

print("Sommet marque ",sommetMarque,T)

T.remove(SProvTemp)

S.append(SProvTemp)

#Il faut ajouter une nouvelle ligne dans le tableau d'analyse.

tableauAnalyse.append(tableauAnalyse[-1])

ligne = ligne +1

#Chaque successeur Y du sommet marqué X est marqué par le poids de X + le poids

# de l'arête qui relie X à Y

for i in T:

if G[sommetMarque][i] != 0:

print ("Ajout pour ",sommetMarque,i)

#On récupère la distance provisoirement marqué

distancePM = tableauAnalyse[ligne][i][0]

#On calcule la distance en passant par le sommet marqué

distanceEM = tableauAnalyse[ligne][sommetMarque][0] + G[sommetMarque][i]

#On ajuste la distance avec si la nouvelle distance est plus petite

if distanceEM < distancePM :

print("ajustement pour le sommet ",i," avec (distance,sommet) ( ",distanceEM,",",sommetMarque,")")

tableauAnalyse[ligne][i][0] = distanceEM

tableauAnalyse[ligne][i][1] = sommetMarque

print ("ta[",ligne,"]=",tableauAnalyse[ligne])

print("--------------------")

return tableauAnalyse,S

ma=ma3()

ta,chemin = Dijkstra(ma,4)

print(np.shape(ta))

for i in range(0,np.shape(ta)[0]):

for j in range(0,np.shape(ta)[1]):

print(ta[i][j]) #,end='')

#print()

print(chemin)