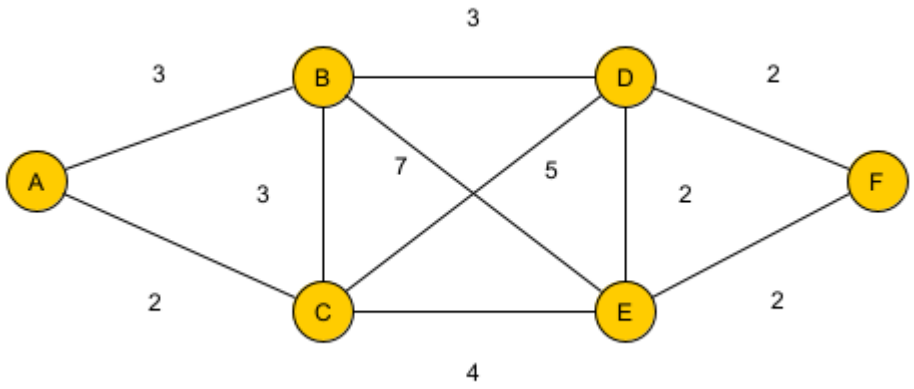
T.D. En plus : Graphes : Méthode de Dijkstra

On considère un graphe pondéré à n sommets, qu'on représentera en Python par une matrice de $\mathcal{M}_n(\mathbb{R})$ contenant les distances entre les sommets (et par convention -1 s'il n'y a pas d'arêtes entre les sommets).

On choisit un sommet de départ, et on souhaite appliquer la méthode de Dijkstra pour trouver la distance minimale à chacun des autres sommets.

On pourra tester le programme avec le graphe suivant, en partant du point A :



Compléter le programme suivant :

```
Graphe=np.array(  …   )
depart=…
n=len(Graphe)
distances=1000*np.ones(n)
distances[depart]=0
sommets_restants=list(range(n))
predecesseurs=np.zeros(n)
predecesseurs[depart]=None

while sommets_restants!=[]:

   #Recherche du sommet suivant
   mini=1000
   sommet=0
   for i in sommets_restants:
     if distances[i]<mini:
        _____________
        sommet=_____________
   sommets_restants.remove(sommet)
```

```
 #Calcul des nouvelles distances
   for successeur in range(n):
      if (successeur in sommets_restants) and Graphe[sommet,successeur]>=0 :
         dist_successeur=_________________
         nouvelle_distance=distances[sommet] + dist_successeur
         if  _________________ :
            distances[successeur]=nouvelle_distance
            predecesseurs[successeur]=sommet

print(distances, predecesseurs)
```

En plus : Compléter le programme suivant pour qu'il stocke dans la liste solution le chemin de A à F : (remarque : la méthode liste.reverse permet d'inverse l'ordre des éléments de liste)

```
depart=0
arrivee=5
solution=[]
while _________:
   solution.append(arrivee)
   arrivee=___________
solution.reverse()
print(solution)
```