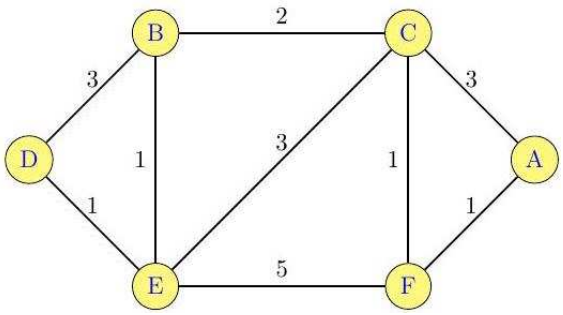
L'algorithme de Dijkstra est un algorithme permettant de déterminer la plus courte chaîne (ou chemin si le graphe est orienté) entre certains points d'un graphe pondéré. On va travailler ici avec des graphes $\mathcal{G} = (S, \mathcal{A}, w)$ où :

S est l'ensemble des sommets et $\mathcal{A}$ est une partie de $S \times S$, appelée ensemble des arêtes et enfin $w : \mathcal{A} \mapsto \mathbb{R}_+$ qui à chaque arête associe un réel appelé poids de l'arête.

Exercice 1. On considère le graphe 1 suivant, noté $(S, \mathcal{A}, w)$ donné par sa représentation graphique.



⋆ Préciser S, $\mathcal{A}$.

⋆ Donner les caractéristiques de ce graphe.

Définition. Soit un graphe pondéré $\mathcal{G} = (S, \mathcal{A}, w)$ où les sommets sont numérotés de 1 à n c'est à dire $S = \{s_i, i \in [[1; n]]\}$. La matrice des poids d'un graphe $M = (g_{i,j})_{(i,j)\in[[1;n]]^2}$ telle que pour tout couple de sommets (s_i, s_j) :

⋆ Si $(s_i, s_j) \in \mathcal{A}$ alors $g_{i,j} = w(s_i, s_j)$.

⋆ Si $(s_i, s_j) \notin \mathcal{A}$ alors $g_{i,j} = +\infty$.

Exercice 2. Le graphe de l'exercice précédent possède 6 sommets. Après renommage des sommets, la matrice des poids de ce graphe est :

$$\begin{array}{c} \\ \mathrm{D} \\ \mathrm{B} \\ \mathrm{E} \\ \mathrm{C} \\ \mathrm{F} \\ \mathrm{A} \end{array} \begin{array}{c} \begin{array}{cccccc} \mathrm{D} & \mathrm{B} & \mathrm{E} & \mathrm{C} & \mathrm{F} & \mathrm{A} \end{array} \\ \begin{pmatrix} 0 & 3 & 1 & +\infty & +\infty & +\infty \\ 3 & 0 & 1 & 2 & +\infty & +\infty \\ 1 & 1 & 0 & 3 & 5 & +\infty \\ +\infty & 2 & 3 & 0 & 1 & 3 \\ +\infty & +\infty & 5 & 1 & 0 & 1 \\ +\infty & +\infty & +\infty & 3 & 1 & 0 \end{pmatrix} \end{array}$$

⋆ Expliquer pourquoi la matrice est symétrique

On peut représenter cette matrice en Python sous la forme d'un tableau de type array avec le script Python suivant :

```
Import numpy as np
Inf = np.inf
G= np.array([[0,3,1,Inf,Inf,Inf],[3,0,1,2,Inf,Inf],[1,1,0,3,5,Inf],
[Inf,2,3,0,1,3],[Inf,Inf,5,1,0,1],[Inf,Inf,Inf,3,1,0]]
```

L'algorithme de Dijkstra consiste en la recherche des plus courtes chaînes menant d'un sommet unique $s \in S$ à chaque autre sommet d'un graphe pondéré $\mathcal{G} = (S, \mathcal{A}, w)$.

Étude sur un exemple

Étape 0 :

Sur l'exemple précédent, et si on choisit $s = D$, l'étape d'initialisation s'écrit :

D	B	E	C	F	A
0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$

On affecte le poids 0 au sommet origine D et on attribue provisoirement un poids $+\infty$ aux autres sommets. Le sommet D de poids 0 est sélectionné.

Étape 1 :

L'opération de relâchement d'une arête $(u, v) \in S \times S$ consiste en un test permettant de savoir s'il est possible, en passant par u, d'améliorer la plus courte chaîne jusqu'à v.

- ⋆ Si le test est négatif, on n'effectue pas de mise à jour.
- ⋆ Si le test est positif, cela signifie qu'une chaîne de plus petite poids est exhibée pour passer du sommet s au sommet v :

On prend $u = D$ et on obtient alors une mise la mise à jour suivante

D	B	E	C	F	A
0	3	1	$+\infty$	$+\infty$	$+\infty$

Étape 2 :

On sélectionne, comme aux prochaines étapes, le sommet u qui vérifie les propriétés suivantes :

- ⋆ u n'a pas encore été sélectionné,
- ⋆ le poids associé est le plus faible (parmi tous les sommets non encore sélectionnés).

Le sommet D a déjà été sélectionné, on choisit de prendre alors $u = E$.

On cherche alors à déterminer, pour tout sommet $v \in S$ si on peut mettre à jour les poids des sommets à l'aide d'un chemin dont le dernier arête est (u, v).

Étant donnée l'étape précédente, cela consiste à considérer les chaînes de s à v comportant deux arêtes 2 (autrement dit les chaînes $s - u - v$) :

D	B	E	C	F	A
0	2	1	4	6	$+\infty$

Étape 3 :

On prend alors $u = B$. On cherche alors à déterminer, pour tout sommet $v \in S$ si on peut mettre à jours les poids des sommets à l'aide d'une chaîne dont la dernière arête est $\{u, v\}$.

Étant donnée l'étape précédente, cela consiste à considérer les chaînes de s à v de trois arêtes (autrement dit les chaînes $s - \ldots - u - v$) :

D	B	E	C	F	A
0	2	1	4	6	$+\infty$

Étape 4 :

On prend alors $u = B$.. On reproduit le schéma et étant donnée l'étape précédente, cela consiste à considérer les chaînes de s à v de 4 arêtes (autrement dit les chaînes $s - \ldots - \ldots - u - v$) :

D	B	E	C	F	A
0	2	1	4	5	7

Étape 5 :

On prend alors $u = F$

Étant donnée l'étape précédente, cela consiste à considérer les chaînes de s à v de 5 arêtes (autrement dit les chaînes $s - \ldots - \ldots - \ldots - u - v$) :

D	B	E	C	F	A
0	2	1	4	5	6

Étape 6 :

On prend alors $u = A$

Étant donnée l'étape précédente, cela consiste à considérer les chaînes de s à v contenant 6 arêtes (autrement dit les chaînes $s - \ldots - \ldots - \ldots - \ldots u - v$)

D	B	E	C	F	A
0	2	1	4	5	6

Fin de l'algorithme

Une fois tous les sommets visités, on a trouvé tous les plus courtes chaînes (de s à tout v) contenant un nombre d'arêtes inférieur ou égale au nombre de sommets en tout. On a donc trouvé tous les plus courtes de chaînes (de s à tout v).

On obtient alors dans un même tableau l'ensemble des étapes :

D	B	E	C	F	A
0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
0	3	1	$+\infty$	$+\infty$	$+\infty$
0	2	1	4	6	$+\infty$
0	2	1	4	6	$+\infty$
0	2	1	4	5	7
0	2	1	4	5	6
0	2	1	4	5	6

Exercice 3. $\star$ Appliquer l'algorithme de Dijkstra avec origine E.

D	B	E	C	F	A

Exercice 4. $\star$ Appliquer l'algorithme de Dijkstra avec origine C.

D	B	E	C	F	A

Script Python

On a crée un script **DijkstraDist (G, depart)** qui prend en paramètre la matrice des poids G et le sommet départ. Si la matrice est de taille $n \in \mathbb{N}$, les sommets sont nommés par un entier dans $[[0; n-1]]$. Cet algorithme renvoie la distance des plus courtes chaînes de départ à tout sommet v.

```
def dijkstra(G,depart):
    # On récupère le nombre de sommets du graphe
    N = np.size(G,0)
    # Initialisation du tableau des plus courts chemins
    pcc = list()
    #le tableau des plus courtes chaînes est complété avec Inf
    for i in range(N):
        pcc.append([Inf, False])
    #le poids associé au départ est 0
    sommet_u = depart
    dist_u = 0
    pcc[depart][0] = 0
    # Le premier sommet sélectionné est le sommet depart
    pcc[depart][1] = True
    # On compte le nombre de sommets sélectionnés
    cpt = 0
    while cpt != N-1:
        # À chaque étape, la solution optimale doit être conservée
        minimum = Inf
        # Étape de relâchement
        for k in range(N):
            # Si le sommet k n'a pas encore été sélectionné
            if pcc[k][1] == False:
                dist_uv = G[sommet_u][k]
                # Distance totale du chemin s -> ... -> u -> v
                dist_totale = dist_u + dist_uv
                # Mise à jour du tableau des plus courts chemins
                if dist_totale < pcc[k][0]:
                    pcc[k][0] = dist_totale

                # Mise à jour de la solution minimale à cette étape si ça l'améliore
                if pcc[k][0] < minimum:
                    minimum = pcc[k][0]
                    prochain_sommet_select = k
                # On a traité complétement un sommet

        cpt = cpt + 1
        # Le sommet à traiter est sélectionné

        sommet_u = prochain_sommet_select
        pcc[sommet_u][1] = True
        # Mise à jour du poids du sommet selectionné
        dist_u = pcc[sommet_u][0]
    return(pcc)
```

Exercice 5. Tester le avec le graphe 1.

Exercice 6. On va modifier le tableau précédent avec pour départ D pour permettre de retrouver la plus courte chaîne. Pour cela, si à une étape, le coefficient associé à un sommet a changé on signifie alors au dessus de la nouvelle valeur de quel sommet provient cette modification. Par exemple en ligne 3 du tableau, la valeur associé à B est passée de $+\infty$ à 3 on rajoute en exposant (D) et pour la ligne 4 la valeur associée à C est passée de $+\infty$ à 4, on écrit (E) en exposant.

D	B	E	C	F	A
0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
0	$3^{(D)}$	1	$+\infty$	$+\infty$	$+\infty$
0	2	1	$4^{(B)}$	$6^{(E)}$	$+\infty$
0	2	1	4	6	$+\infty$
0	2	1	4	5	7
0	2	1	4	5	6
0	2	1	4	5	6

⋆ Compléter le tableau précédent.

⋆ Reconstituer alors le plus court chaîne de D vers A, celui de D vers E et celui de D vers B.

Exercice 7. Implémenter la fonction DijsktraDistchaîne(G, depart) qui prend en paramètre la matrice des poids G et le sommet origine départ et renvoie le poids des plus courts chaînes de départ à tout sommet v ainsi que le dernier sommet utilisé pour calculer ce poids.
On utilisera les deux lignes **pcc. append ([Inf, False, None])** et **pcc[k][2]= sommet_u** qu'on placera judicieusement.

Exercice 8. ⋆ Implémenter la fonction dijkstraPCC (G, depart, arrivee) qui permet d'afficher la plus courte chaîne de départ à arrivée.

⋆ Tester votre fonction en prenant pour origine D puis E.