TP 25 - Algorithme de Dijkstra

Considérant un graphe pondéré avec des poids positifs, l'algorithme de Dijkstra permet d'identifier le plus court chemin d'un sommet de départ fixé vers n'importe quel autre sommet d'un graphe (à condition qu'un tel chemin existe).
Étant donné un graphe $G = (S, A)$ et un sommet de départ $s \in S$, on définit :

- un dictionnaire `D` dont :
  - les clés sont les sommets atteignables à partir de s
  - la valeur associée à la clé a est le couple `(d,p)` tel que :
    d est la distance entre s et a
    p est le voisin entrant de a sur le chemin de s à a
- une liste `SaV` de sommets à visiter.

L'algorithme est le suivant : **Prog**

Entrée : $s \in S$ un sommet et
 M la matrice de poids du graphe $G(S, A)$
 On note $d(s_1, s_2)$ le poids de l'arc $(s_1, s_2) \in A$

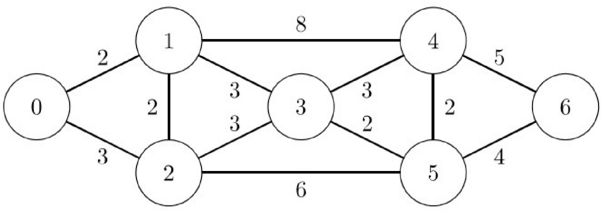
 # Initialiser :
D le dictionnaire avec une relation
 de clé s et de valeur $(0, \mathtt{None})$
`SaV` la liste avec l'élément s
Tant que `SaV` n'est pas vide faire
 Chercher dans `SaV` le sommet p le plus proche de s
 Retirer p de `SaV`
 # Mise à jour des distances des voisins de p
 Pour v parcourant les voisins de p :
  Poser $r \leftarrow d(s,p) + d(p,v)$
  Si v n'est pas une clé de D alors
   ajouter v à `SaV`
   ajouter à D la relation de clé v et de valeur (r, p)
  SinonSi $r < d(s, v)$ alors
   mettre à jour la valeur associée à v par (r, p)
  FinSi
 FinPour
FinTantQue

Sortie : D le dictionnaire de Dijkstra associé à s

Remarque – À chaque étape, on visite un nouveau sommet, qui est définitivement retiré de `SaV` : les seuls sommets ajoutés sont les voisins du sommet en cours de visite qui n'ont pas été visité précédemment. À partir d'un certain rang, tous les sommets atteignables auront été visités et l'algorithme s'arrête. Les clés de `D` donne la composante connexe contenant s.

Exemple

Appliquons l'algorithme de Dijkstra au graphe suivant pour le sommet s_1 :



`SaV`	p	$d_+(p)$
[1]	1	[0,2,3,4]

  On intialise D à {1:(0,None)}

v	D
0	{1:(0,None),0:(2,1)}
2	{1:(0,None),0:(2,1),2:(2,1)}
3	{1:(0,None),0:(2,1),2:(2,1),3:(3,1)}
4	{1:(0,None),0:(2,1),2:(2,1),3:(3,1),4:(8,1)}

`SaV`	p	$d_+(p)$
[0,2,3,4]	0	[1,2]

v	D
1	pas de changement car 4>0
2	$\cdots$

`SaV`	p	$d_+(p)$
[2,3,4]	2	[0,1,3,5]

v	D
0,1,3	$\cdots$
5	{1:(0,None),0:(2,1),2:(2,1),3:(3,1),4:(8,1),5:(8,2)}

`SaV`	p	$d_+(p)$
[3,4,5]	3	[1,2,4,5]

v	D
1,2	$\cdots$
4	{1:(0,None),0:(2,1),2:(2,1),3:(3,1),4:(6,3),5:(8,2)}
5	{1:(0,None),0:(2,1),2:(2,1),3:(3,1),4:(6,3),5:(5,3)}

`SaV`	p	$d_+(p)$
[4,5]	5	[2,3,4,6]

v	D
2,3,4	$\cdots$
6	{1:(0,None),0:(2,1),2:(2,1),3:(3,1),4:(6,3),5:(5,3),6:(9,5)}

`SaV`	p	$d_+(p)$	v	D
[4,6]	4	[1,3,5,6]	1,3,5,6	$\cdots$

`SaV`	p	$d_+(p)$	v	D
[6]	6	[4,5]	4,5	$\cdots$

Le dictionnaire de Dijkstra de sommet inital s_1 est :
$\{1:(0,\mathtt{None}), 0:(2,1), 2:(2,1), 3:(3,1), 4:(6,3), 5:(5,3), 6:(9,5)\}$
Pour déterminer le chemin minimal de s_1 à un sommet s, il suffit de remonter le chemin à partir de s jusqu'à tomber sur `None`.
Par exemple, entre s_1 et s_6 : $s_6 \leftarrow s_5 \leftarrow s_3 \leftarrow s_1$
Le chemin est donc $\big((s_1,s_3),(s_3,s_5),(s_5,s_6)\big)$ de poids 9.

Exercice 1 Définir les fonctions suivantes :

- `voisins(s,M)` retourne la liste des voisins du sommet s du graphe de matrice de poids `M`
- `dmin(L,D)` retourne le sommet v de la liste `L` de plus courte distance au sommet de départ, les distances étant données dans le dictionnaire de type "Dijkstra" `D` et `L` la liste des clés de `D`
- `Dijkstra(s,M)` retourne le dictionnaire de Dijkstra de sommet initial s et de graphe associé à `M`
- `chemin(s,a,M)` retourne le chemin de plus courte distance du sommet s au sommet a du graphe associé à `M` par la méthode de Dijkstra.

Quel est l'intérêt de travailler à partir du sommet p, le plus proche de s dans `SaV` ?