**L’algorithme de Dijkstra ou comment trouver le plus court chemin ?**

**INTRODUCTION**

Le but de l’algorithme de Dijkstra est de trouver un chemin le plus court entre deux sommets dans un graphe pondéré.

**DEFINITION D’UN GRAPHE**

Le [graphe](https://www.techno-science.net/definition/6449.html) est noté *G* = (*S*,*A*) où :

* l'[ensemble](https://www.techno-science.net/glossaire-definition/Ensemble.html) *S* est l'ensemble des sommets du graphe *G* ;
* l'ensemble *A* est l'ensemble des arêtes de *G* tel que : si (*s*1,*s*2) est dans *A*, alors il existe une arête depuis le nœud *s*1 vers le nœud *s*2 ;
* on définit la procédure *Poids*(*s*1,*s*2) définie sur *A* qui renvoie le poids positif de l'arête reliant *s*1 à *s*2 (et un poids [infini](https://www.techno-science.net/glossaire-definition/Infini.html) pour les paires de sommets qui ne sont pas connectées par une arête).

Les données du problème sont : le graphe pondéré, un sommet d’entrée I, et un sommet de sortie O.

Le graphe présenté ci-dessous possède les caractéristiques suivantes.

• Il n’est pas orienté : si (u, v) 2 A alors on a (v, u) 2 A.

Ainsi, si on peut aller de u à v, alors on peut aussi aller de v à u.

• Il est simple : il y a au plus une arête entre deux sommets.

• Il est pondéré : à chaque arête on associe une valeur positive appelée poids.

Ce poids représente la distance entre les deux sommets.

Ici :

 *S* = { A,B,C,D,E,F },

 *A* = {(A, B),...,(E,F),...,(F, A)}

Les associations de deux arêtes que contient *A* possèdent chacune un poids.

D’après le graphe par exemple : (D,B) = 3 , (A,B) = $\infty $

**PRINCIPE DE L’ALGORITHME DE DIJKSTRA**

L’algorithme de Dijkstra consiste à construire progressivement, à partir des [données](https://www.techno-science.net/definition/222.html) initiales, un sous-graphe dans lequel sont classés les différents sommets par ordre croissant de leur distance minimale au sommet de départ. La distance correspond à la somme des poids des arêtes empruntées.

La première étape consiste à mettre de côté le sommet de départ et à repérer la distance du sommet de départ aux autres sommets du graphe. Cette distance est infinie si aucun arc ne relie le sommet au sommet de départ, elle est de *n* s'il existe un arc reliant ce sommet au sommet de départ et que le poids le plus faible (s'il existe plusieurs arcs) est de *n*.

La [seconde](https://www.techno-science.net/definition/1516.html) étape consiste à repérer le sommet qui possède alors la plus courte distance au sommet de départ et à le mettre de côté. Pour tous les sommets restants, on compare alors la distance trouvée précédemment à celle que l'on obtiendrait via le sommet que l'on vient de mettre de côté et on ne conserve que la plus petite des valeurs. et on continue ainsi jusqu'à épuisement des sommets ou jusqu'à sélection du sommet d'arrivée.

**EXEMPLE D’APPLICATION**

L'exemple suivant montre les étapes successives dans la résolution du chemin le plus court dans un graphe. Les nœuds symbolisent des villes identifiées par une lettre et les arêtes indiquent la distance entre ces villes. On cherche à déterminer le plus court trajet pour aller de la ville A à la ville J.

|  |  |
| --- | --- |
| https://www.techno-science.net/illustration/Definition/350px/DijkstraBis01.png**Étape 1** : à partir de la ville A, 3 villes sont accessibles, B, C, et E qui se voient donc affectées des poids respectifs de 85, 217, 173, tandis que les autres villes sont affectées d'une distance infinie. | https://www.techno-science.net/illustration/Definition/350px/DijkstraBis02.png**Étape 2** : la distance la plus courte est celle menant à la ville B. Le passage par la ville B ouvre la voie à la ville F (85+80 = 165). |
| https://www.techno-science.net/illustration/Definition/350px/DijkstraBis03.png**Étape 3** : La distance la plus courte suivante est celle menant à la ville F. Le passage par la ville F ouvre une voie vers la ville I (415). | https://www.techno-science.net/illustration/Definition/350px/DijkstraBis04.png**Étape 4** : La distance la plus courte suivante est alors celle menant à la ville E. Le passage par la ville E ouvre une voie vers la ville J (675). |
| https://www.techno-science.net/illustration/Definition/350px/DijkstraBis05.png**Étape 5** : la distance la plus courte suivante mène alors à la ville C. Le passage par la ville C ouvre une voie vers la ville G (403) et la ville H (320). | https://www.techno-science.net/illustration/Definition/350px/DijkstraBis06.png**Étape 6** : la distance la plus courte suivante mène à ville H(320). Le passage par la ville H ouvre une voie vers la ville D et un raccourci vers la ville J (487< 675). |
| https://www.techno-science.net/illustration/Definition/350px/DijkstraBis07.png**Étape 7** : la distance la plus courte suivante mène à la ville G et ne change aucune autre distance. | https://www.techno-science.net/illustration/Definition/350px/DijkstraBis08.png**Étape 8** : la distance la plus courte suivante mène à la ville I. Le passage par la ville I ouvre un chemin vers la ville J qui n'est pas intéressant (415+ 84 > 487). |
| https://www.techno-science.net/illustration/Definition/350px/DijkstraBis09.png**Étape 9** : la distance la plus courte suivante mène à la ville J (487).On connait ainsi le chemin le plus court menant de A à J, il passe par C et H et mesure 487 km. |  |

**PRÉSENTATION SOUS FORME DE TABLEAU**

L'illustration par une série de graphes peut se révéler un peu longue. Il est d'autre part un peu plus difficile de repérer le chemin le plus court à l'issue du dessin.

Ainsi l'algorithme de Dijsktra est souvent réalisé à l'aide d'un tableau dans lequel chaque étape correspond à une ligne.

À partir de la matrice des arcs reliant les diverses villes :



On construit un tableau dans lequel les distances d'un sommet au sommet de départ sont regroupées dans une même colonne. Les sommets sélectionnés sont soulignés.

Les distances des voies ouvertes par la sélection d'un nouveau sommet sont barrées si elles sont supérieures à des distances déjà calculées.

Quand un sommet est sélectionné, c'est que l'on a découvert sa distance minimale au sommet, il est alors inutile de chercher d'autres distances de ce sommet au [point](https://www.techno-science.net/definition/2520.html) de départ.

**ALGORITHME DE DIJKSTRA**



La construction de ce tableau donne non seulement la distance minimale de la ville A à la ville J mais aussi le chemin à suivre (J - H - C - A) ainsi que toutes les distances minimales de la ville A aux autres villes rangées par ordre croissant.

On remarque que le tableau représente une matrice symétrique

**PROGRAMMATION EN PYTHON**

Les différents liens internet se rapportant à la programmation en Python de l’algorithme de Dijkstra se situent dans la sitographie.

Ne connaissant pas en détail le langage Python, je ne préfère pas essayer d’expliquer les différents codes que j’ai pu trouver sur internet.

J’ai donc sélectionné différents codes m’ayant l’air correct qui se trouvent dans les différents URL .

**SITOGRAPHIE**

**Site détaillé et synthétique :**

[**https://www.techno-science.net/glossaire-definition/Algorithme-de-Dijkstra.html**](https://www.techno-science.net/glossaire-definition/Algorithme-de-Dijkstra.html)

**Théorie :** [**https://www.imo.universite-paris-saclay.fr/~damien.thomine/fr/archives/Compendium/CAPES/Graphes/AlgorithmeDijkstra.pdf**](https://www.imo.universite-paris-saclay.fr/~damien.thomine/fr/archives/Compendium/CAPES/Graphes/AlgorithmeDijkstra.pdf)

**Exemple exercice :** [**https://www.normalesup.org/~dconduche/informatique/PT/Cours/Dijkstra.pdf**](https://www.normalesup.org/~dconduche/informatique/PT/Cours/Dijkstra.pdf)

**TP :** [**https://arnaud.jobin.pro/BCPST2/Informatique/TP\_10\_11\_prof.pdf**](https://arnaud.jobin.pro/BCPST2/Informatique/TP_10_11_prof.pdf)

**Programmation en Python :**

[**https://mpsib-camille-guerin.pagesperso-orange.fr/Python/Dijkstra/dijkstra.pdf**](https://mpsib-camille-guerin.pagesperso-orange.fr/Python/Dijkstra/dijkstra.pdf)

[**https://www.udacity.com/blog/2021/10/implementing-dijkstras-algorithm-in-python.html**](https://www.udacity.com/blog/2021/10/implementing-dijkstras-algorithm-in-python.html)

[**https://mpegon.perso.math.cnrs.fr/documents/BCPST/TP06.pdf**](https://mpegon.perso.math.cnrs.fr/documents/BCPST/TP06.pdf)

[**https://www.delftstack.com/fr/howto/python/dijkstra-algorithm-python/**](https://www.delftstack.com/fr/howto/python/dijkstra-algorithm-python/)



**from** collections **import** deque

 **def** dijkstraAlgo(graph, vertex):

 queue = deque([vertex])

 distance = {vertex: 0}

 **while** queue:

 t = queue.popleft()

 print("Visite du sommet " + str(t))

 **for** voisin **in** graph[t]:

 queue.append(voisin)

 nouvelle\_distance = distance[t] + graph[t][voisin]

 **if**(voisin not **in** distance or nouvelle\_distance < distance[voisin]):

 distance[voisin] = nouvelle\_distance

 print("Met à jour le sommet " + str(voisin) + " avec la distance : " + str(nouvelle\_distance))

 **return** distance

#Liste d'adjacence du graphe

graph = {'A':{'B':135,'C':4},'B':{'E':5},'C':{'E':161,'D':2},'D':{'E':3},'E':{}}

distance = dijkstraAlgo(graph,'A')

print("Distances" + str(distance))

La sortie devrait être :

Visite du sommet A

Met à jour le sommet B avec la distance : 135

Met à jour le sommet C avec la distance : 4

Visite du sommet B

Met à jour le sommet E avec la distance : 140

Visite du sommet C

Met à jour le sommet D avec la distance : 6

Visite du sommet E

Visite du sommet E

Visite du sommet D

Met à jour le sommet E avec la distance : 9

Visite du sommet E

Distances{'A': 0, 'B': 135, 'C': 4, 'E': 9, 'D': 6}