

Intelligence Artificielle 2

Cours 2 et 3 - Graphes et Chemins

Steve Lévesque, Tous droits réservés © où applicables

Table des matières

1 Graphes

- Définition
- Liste des algorithmes en lien avec les graphes

2 Démonstration avec module Scipy - Dijkstra

- Démonstration Dijkstra - Théorie (explication algorithmique)
- Démonstration Dijkstra - Pratique (Python et module SciPy)

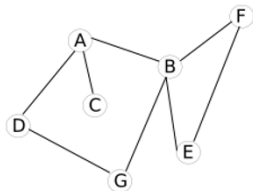
Définition

Définitions de mots et phrases courantes :

- Graphe : composé de nœud (sommet) et d'arêtes (arcs)
- Orientation :
 - Graphe orienté : graphe ayant une direction préétablie sur les arêtes
 - Graphe non-orienté : la direction est arbitraire (non définie)
- Chaîne : chemin ou suite d'arêtes
- Cycle : chaîne qui commence et se termine au même noeud, peut avoir 2 ou plusieurs noeuds

Matrice d'adjacence - Non orienté

Matrice d'adjacence d'un graphe non orienté :



0	1	1	1	0	0	0
1	0	0	0	1	1	1
1	0	0	0	0	0	0
1	0	0	0	0	0	1
0	1	0	0	0	1	0
0	1	0	0	1	0	0
0	1	0	1	0	0	0

	A	B	C	D	E	F	G
A	0	1	1	1	0	0	0
B	1	0	0	0	1	1	1
C	1	0	0	0	0	0	0
D	1	0	0	0	0	0	1
E	0	1	0	0	0	1	0
F	0	1	0	0	1	0	0
G	0	1	0	1	0	0	0

Figure: https://dav74.github.io/site_nsi_term/c9c/

Matrice d'adjacence - Orienté

Matrice d'adjacence d'un graphe orienté :

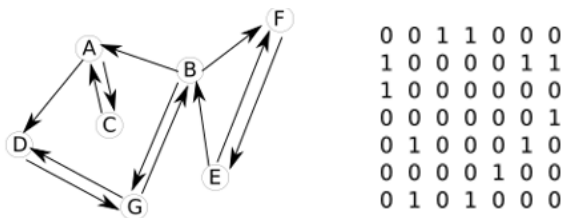


Figure: https://dav74.github.io/site_nsi_term/c9c/

Matrice d'adjacence - Pondéré

Matrice d'adjacence d'un graphe Pondéré :

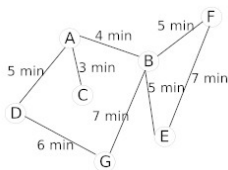


Figure:
https://dav74.github.io/site_nsi_term/c9c/

0	4	3	5	0	0	0
4	0	0	0	5	5	7
3	0	0	0	0	0	0
5	0	0	0	0	0	6
0	5	0	0	0	7	0
0	5	0	0	7	0	0
0	7	0	6	0	0	0

Figure:
https://dav74.github.io/site_nsi_term/c9c/

Code de matrice en Python

Listing: https://dav74.github.io/site_nsi_term/c9c/

```
1  # https://dav74.github.io/site\_nsi\_term/c9c/
2
3  m = [
4      [0, 1, 1, 1, 0, 0, 0],
5      [1, 0, 0, 0, 1, 1, 1],
6      [1, 0, 0, 0, 0, 0, 0],
7      [1, 0, 0, 0, 0, 0, 1],
8      [0, 1, 0, 0, 0, 1, 0],
9      [0, 1, 0, 0, 1, 0, 0],
10     [0, 1, 0, 1, 0, 0, 0],
11 ]
```

Liste des algorithmes en lien avec les graphes

Voici une liste d'algorithmes utilisables sur les graphes pour calculer l'ordre de traverse, la plus petite distance, etc.

- Algorithme du plus petit chemin (Dijkstra, algorithme glouton)
- Traverse de Graphes :
 - DFS, Depth First Search (Profondeur)
 - BFS, Breadth First Search (Largeur)
- A* (A star)

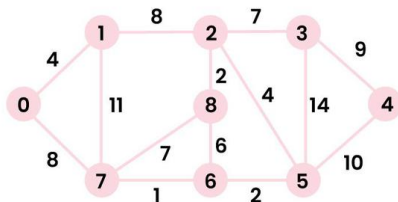
Pour en savoir plus :

<https://www.geeksforgeeks.org/graph-theory-tutorial/>

Démonstration avec module Scipy - Dijkstra

Une démonstration sera faite du fonctionnement de Dijkstra ainsi que son utilisation (facile et directe) avec le module SciPy.

Voici le graphe qu'on va utiliser :



Working of Dijkstra's Algorithm



Figure: <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>

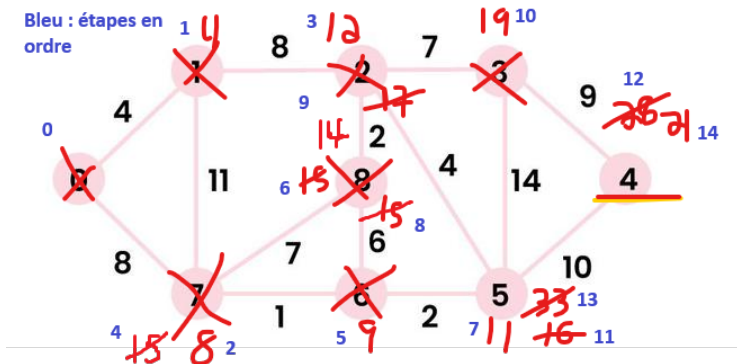
Démonstration Dijkstra - Théorie (explication algorithmique)

Étapes globales :

- 1 Commencer du point de départ (i.e. noeud 0 dans notre exemple)
- 2 Calculer la distance de tous les voisins du noeud actuel
 - Annoter la distance sur le noeud, et calculer le total
 - Passer au prochain voisin du noeud actuel jusqu'à tant que tous les voisins du noeud respectif soient explorés
- 3 Passer au prochain noeud (i.e. noeud 1)
- 4 Répéter l'étape 2 pour le nouveau noeud actuel (i.e. noeud 1)
 - Si une réponse est inférieure pour un nouveau chemin en contrepartie d'un ancien chemin évalué, faire le remplacement avec la nouvelle distance plus optimale
- 5 Continuer avec les étapes 3 et 4 jusqu'à la fin du graphe.

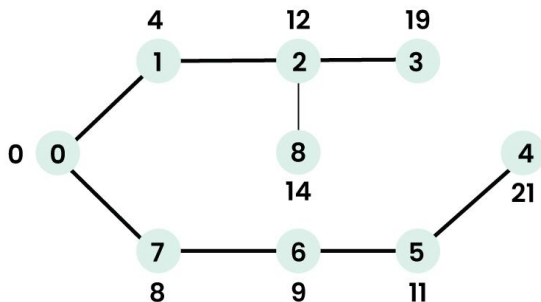
Démonstration Dijkstra - Théorie (explication algorithme)

Exécution à la main :



Démonstration Dijkstra - Théorie (explication algorithme)

Résultat final sans annotations :



Working of Dijkstra's Algorithm



Figure: <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>

Démonstration Dijkstra - Théorie (explication algorithme)

Un exemple d'exécution de Dijkstra étape par étape :

Source :

https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

Démonstration Dijkstra - Pratique (Python et module SciPy)

Listing: <https://www.geeksforgeeks.org/scipy-csgraph-compressed-sparse-graph/>

```
1 from scipy.sparse import csr_matrix
2 from scipy.sparse.csgraph import dijkstra, shortest_path
3
4 graph = csr_matrix([ # We must create the graph in a matrix structure.
5     [0, 4, 0, 0, 0, 0, 0, 8, 0],
6     [4, 0, 8, 0, 0, 0, 0, 11, 0],
7     [0, 8, 0, 7, 0, 4, 0, 0, 2],
8     [0, 0, 7, 0, 9, 14, 0, 0, 0],
9     [0, 0, 0, 9, 0, 10, 0, 0, 0],
10    [0, 0, 4, 14, 10, 0, 2, 0, 0],
11    [0, 0, 0, 0, 0, 2, 0, 1, 6],
12    [8, 11, 0, 0, 0, 0, 1, 0, 7],
13    [0, 0, 2, 0, 0, 0, 6, 7, 0]
14 ])
15 res = dijkstra(csgraph=graph, directed=False, indices=0)
16 res = shortest_path(csgraph=graph, method='D', directed=False, indices=0)
17 print(res) # [ 0.  4. 12. 19. 21. 11.  9.  8. 14.] Sorted by node's index
```

Bibliographie

- <https://coin-or.github.io/pulp/main/index.html>
- https://dav74.github.io/site_nsi_term/c9c/
- <https://www.geeksforgeeks.org/graph-theory-tutorial/>
- <https://www.geeksforgeeks.org/scipy-csgraph-compressed-sparse-graph/>