UCLouvain

LINFO1114 - Mathématiques discrètes

Rapport de projet - 2022/2023

Distance du plus court chemin: Dijkstra, Bellman-Ford et Floyd-Warshall

Groupe 13

Noma - Étudiant

13752101 - HUET Anatole,
04621900 - KHEIRALLAH Cédric,
31662200 - LARAKI Narjis

Professeur : Marco Saerens

December 24, 2022

Contents

1 Introduction

Dans ce projet, notre objectif a été de comprendre et mettre en œuvre trois algorithmes permettant de calculer la matrice de distance des plus courts chemins entre toutes les paires de nœuds d'un graphe non-dirigé, connecté et pondéré G. Les algorithmes que nous avons utilisés sont ceux vus en cours comme celui de Dijkstra, de Bellman-Ford et de Floyd-Warshall.

2 Rappels théoriques

2.1 Dijkstra

L'algorithme de Dijkstra est un algorithme permettant de résoudre le problème du plus court chemin. Étant donné un graphe non-orienté, dont les arêtes sont munies de poids, et deux sommets de ce graphe, l'algorithme de Dijkstra peut trouver un chemin entre les deux sommets dans le graphe de poids minimum.

Pour trouver le plus court chemin du sommet A au sommet Z, on attribue la valeur 0 au sommet de départ A et ∞ aux autres sommets. Ceci est noté $L_0(A) = 0$ et $L_0(V) = \infty$.
L'algorithme forme ensuite des ensembles de sommets notés S_k (ensemble S après k itérations). À chaque itération on ajoute les sommets U, les sommets qui ne sont pas présents dans l'ensemble S_{k-1} et ont le chemin adjacent le plus court.

Soit V un sommet qui n'est pas contenu dans l'ensemble S_k et $L_k(V)$ la longueur du plus court chemin de A à V ne contenant que les sommets présents dans S_k.
Le plus court chemin de A à V contenant uniquement les éléments de S_k (et donc les chemins passant par ces éléments) est soit un plus court chemin passant uniquement par les sommets que l'on connait déjà (S_{k-1}) soit un nouveau plus court chemin de A à U qui doit maintenant être considéré (on ajoute donc l'arête u, v).
Dans ce cas nous avons $L_k(V) = minL_{k-1}(V), L_{k-1}(U) + w(U, V)$

2.2 Bellman-Ford

L'algorithme de Bellman-Ford permet de calculer des plus courts chemins depuis un sommet spécifique dans un graphe orienté pondéré.

Il diffère de l'algorithme de Djikstra du fait qu'il accepte la présence d'arcs de poids négatif ce qui fait qu'il est possible de détecter les circuits où le poids total est strictement négatif.

Soit $d[t, k]$ la distance du sommet source s à t avec un chemin contenant au plus k arcs.
La formule de cet algorithme étant : $d[t, k] = min[d[t, k-1], min_{arc(u,t)}(d[u, k-1] + poids(u, t))]$

2.3 Floyd-Warshall

L'algorithme de Floyd-Warshall est un algorithme utile pour trouver les plus courts chemins dans un graphe (qu'il soit dirigé ou non). Toutefois, il a la particularité par rapport aux autres algorithmes présentés ici de calculer directement toutes les distances entre paires de nœuds. Cependant, cet algorithme ne fonctionne pas avec des cycles négatifs.

L'objectif de cet algorithme est de trouver le chemin le plus court entre 2 sommets en cherchant le chemin le plus court entre le chemin direct du sommet 1 au sommet 2 ou de la somme des poids en passant par un sommet intermédiaire.

La formule générale de cet algorithme est : $A^k[i, j] = min(A^{k-1}[i, j], A^{k-1}[i, k] + A^{k-1}[k, j])$

3 Calcul théorique et numérique via Dijkstra

Afin de calculer la distance entre A et J, nous allons remplir le tableau ci-dessous pour chaque itération de l'algorithme de Dijkstra.
Chaque nœud ayant terminé d'être visité est in

	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10
A	0	**0**	-	-	-	-	-	-	-	-	-
B	inf	5	5	**5**	-	-	-	-	-	-	-
C	inf	4	**4**	-	-	-	-	-	-	-	-
D	inf	5	5	5	**5**	-	-	-	-	-	-
E	inf	inf	6	6	6	6	**6**	-	-	-	-
F	inf	inf	5	5	5	**5**	-	-	-	-	-
G	inf	inf	inf	10	10	10	10	10	10	**10**	-
H	inf	inf	inf	inf	inf	9	7	**7**	-	-	-
I	inf	inf	inf	inf	9	7	7	7	**7**	-	-
J	inf	inf	inf	inf	inf	inf	inf	12	10	10	**10**

Table 1: Table des itérations pour l'algorithme de Dijkstra de A à J

Il suffit à présent de lire le plus court chemin dans le tableau à chaque étape pour trouver la plus courte distance entre A et J

- Itération L0 : /
- Itération L1 : A
- Itération L2 : A -> C
- Itération L3 : A -> B
- Itération L4 : A -> D
- Itération L5 : A -> C -> F
- Itération L6 : A -> C -> E
- Itération L7 : A -> C -> F -> H
- Itération L8 : A -> C -> F -> I
- Itération L9 : A -> B -> G
- Itération L10 : A -> C -> F -> I -> J

Nous pouvons ainsi déduire que le plus court chemin entre A et J est A -> C -> F -> I -> J et que sa distance est égale à 10 ce qui correspond aux résultats obtenus avec les algorithmes de Dijkstra, Bellman-Ford et Floyd-Warshall.

4 Résultats d'implémentation

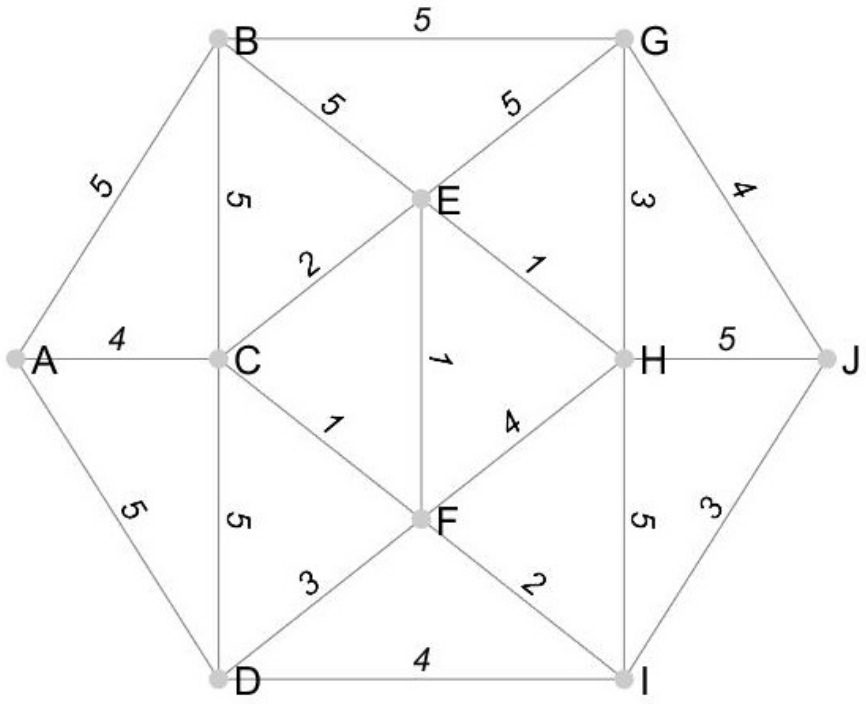

Figure 1: Graphe initial non-dirigé, connecté et pondéré

```
[[ 0.  5.  4.  5. inf inf inf inf inf inf]
 [ 5.  0.  5. inf  5. inf  5. inf inf inf]
 [ 4.  5.  0.  5.  2.  1. inf inf inf inf]
 [ 5. inf  5.  0. inf  3. inf inf  4. inf]
 [inf  5.  2. inf  0.  1.  5.  1. inf inf]
 [inf inf  1.  3.  1.  0. inf  4.  2. inf]
 [inf  5. inf inf  5. inf  0.  3. inf  4.]
 [inf inf inf inf  1.  4.  3.  0.  5.  5.]
 [inf inf inf  4. inf  2. inf  5.  0.  3.]
 [inf inf inf inf inf inf  4.  5.  3.  0.]]
```

Figure 2: Matrice de coûts C déduite du graphe G

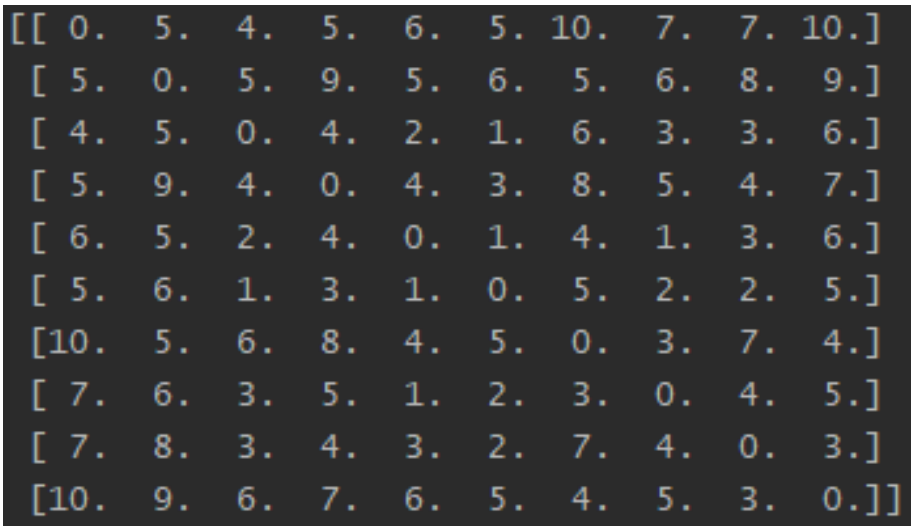

Figure 3: Matrice de distances D obtenue par Dijkstra, Bellman-Ford et Floyd-Warshall

5 References

Prim and Floyd-Warshall Comparative Algorithms in Shortest Path Problem

Floyd-warshall algorithm to determine the shortest path based on android

Improving The Floyd-Warshall All Pairs Shortest Paths Algorithm

Algorithme de Bellman-Ford

Bellman–Ford Algorithm | DP-23

Dijkstra's algorithm