Projet LINFO1114 – Mathématiques discrètes « Distance du plus court chemin: Dijkstra, Bellman-Ford et Floyd-Warshall »

Professeur	Marco Saerens marco.saerens@uclouvain.be
Téléphone	010 47 92 46
Bureau	b.125
Adresse	Université catholique de Louvain Place des Doyens 1 1348 Louvain-La-Neuve Belgique
Assistants	Diego Eloi diego.eloi@uclouvain.be Flore Vancompernolle Vromman flore.vancompernolle@uclouvain.be
Date	Novembre 2024

Objectif : Le but de ce travail est d'implémenter et de tester trois algorithmes permettant de calculer la matrice de distance des plus courts chemins entre toutes les paires de noeuds d'un graphe. Les algorithmes que vous devrez utiliser sont ceux de Dijkstra, Bellman-Ford et Floyd-Warshall. Vous travaillerez par groupes de *trois* étudiant.e.s impérativement (merci de vous inscrire dans un groupe sur Moodle). Chaque groupe trouvera un graphe différent sur Moodle, sur lequel il devra calculer la matrice de distance des plus courts chemins via ces trois algorithmes. Il faudra suivre rigoureusement ces algorithmes, comme détaillé au cours pour l'algorithme de Dijkstra. Vous trouverez toutes les informations utiles dans les slides du cours et de TPs, ainsi que dans le livre de référence du cours (ouvrage de Rosen), mais aussi dans des ouvrages de théorie des graphes ou d'algorithmique (par exemple les ouvrages *Introduction to the design and analysis of algorithms, 3th ed.* (2012) de Anany Levitin, *Algorithms, 4th ed.* (2015) de Robert Sedgewick, ou *Algorithmes - notions de base* (2013) de Thomas Cormen).

Fonctions (Python 3) : Dans le cadre de l'implémentation des trois algorithmes en Python, les signatures des fonctions sont :

```
def Dijkstra(C : np.matrix) -> np.matrix
def Bellman_Ford(C : np.matrix) -> np.matrix
def Floyd_Warshall(C : np.matrix) -> np.matrix
```

- **Input :** Une matrice (numpy) $n \times n$ de coûts **C** d'un graphe non-dirigé, pondéré et connecté G. Cette matrice contient les coûts c_{ij} (non-négatifs) associés aux liens (i, j) du graphe. Par la suite, n sera le nombre de noeuds de G que l'on peut déduire de la matrice.
- **Output :** Une matrice $n \times n$ **D** contenant les distances des plus courts chemins entre toutes les paires de noeuds du graphe G. Donc l'élément d_{ij} de la matrice **D** renvoyée

contient la distance du plus court chemin entre le noeud i et le noeud j.

Uniquement l'algorithme de Floyd-Warshall permet de calculer directement toutes les distances entre paires de noeuds. Ceux de Dijkstra et Bellman-Ford calculent les distances de tout noeud au départ d'un noeud de départ prédéfini et devront donc être exécutés dans une boucle permettant le calcul de toutes les distances entre noeuds.

Bien sûr, vous ne pouvez pas utiliser des fonctions prédéfinies qui calculent la matrice de distance des plus courts chemins directement. Le but du projet est justement de comprendre et d'implémenter ces fonctions par vous-même. Vous pouvez néanmoins vous aider de librairies telles que Numpy pour les opérations matricielles.

De plus, il vous est demandé d'ajouter à votre code une procédure "main" (1) qui lit un fichier .csv (que vous devez créer et definir vous-même) contenant la matrice de coûts **C** du graphe fourni sur Moodle (séparez vos valeurs par une virgule et chaque ligne de la matrice par un passage à la ligne), (2) qui lance le calcul des matrices de distance des plus courts chemins (via vos trois fonctions) et (3) qui imprime les résultats à l'écran (la matrice de coûts et les trois matrices de distances). Le fichier .csv est donc créé à partir du dessin de graphe fourni sur Moodle et doit contenir les pondérations (coûts) associées à ce graphe. Notez que la matrice de coûts est similaire à une matrice d'adjacence, sauf qu'elle contient des coûts, et dès lors des valeurs très élevées (représentant $+\infty$, signifiant que l'on ne peut pas passer par ce lien) lorsqu'il n'y a pas de lien (au lieu de 0 dans le cas de la matrice d'adjacence, lien manquant). Concrètement, dans les matrices de coûts qui serviront à tester votre code et dans le fichier .csv que vous devez écrire, les liens manquants seront encodés par une valeur positive importante par rapport aux coûts observés sur les liens ; en pratique 1000000000000 ($= 10^{12}$). Notez que vous pouvez néanmoins représenter ces valeurs par ∞ dans les impressions et dans le rapport.

Données : Un graphe non-dirigé, connecté et pondéré (coûts) représentant un réseau de 10 noeuds avec des coûts (pondérations) associés aux liens. Une illustration de ce graphe est disponible sur Moodle dans le fichier "Graphes".

Rapport et code : Le rapport est un fichier PDF (**8 pages maximum** ; écrit en LaTeX) qui se compose de :

- Une introduction.
- Un rappel théorique décrivant les trois algorithmes (Dijkstra, Bellman-Ford et Floyd-Warshall) à implémenter, avec les équations et les références aux sources dont vous vous êtes inspirées.
- Le calcul théorique et numérique (fait à la main et détaillé en LaTeX via un tableau comme au TP) du plus court chemin entre le noeud A et le noeud J du graphe qui vous est assigné sur Moodle via l'algorithme de Dijkstra (et uniquement Dijkstra). Vérifiez que vous obtenez les mêmes valeurs que votre implémentation.
- Dans le cadre de l'implémentation en Python de votre procédure *main*, l'impression à l'écran de :

  - — la matrice de coûts **C** que vous avez définie d'après le graphe assigné à votre groupe,
  - — les trois matrices de distances **D** (elles devraient être identiques).
- Le code complet en annexe – n'oubliez pas de bien commenter ce code! Les annexes ne font pas partie des 8 pages.

Attention, respectez bien ces consignes car nous nous baserons sur celles-ci pour calculer la note du projet de manière semi-automatique. Egalement, n'oubliez évidemment pas de mentionner vos noms, NOMA et votre numéro de groupe sur la page de couverture du rapport.

Langage de programmation : L'implémentation devra impérativement être codée en Python 3 en respectant les consignes et les signatures énoncées ci-dessus.

Evaluation et consignes : Le projet est obligatoire et à réaliser par groupes de trois étudiantes et étudiants. L'évaluation portera sur le contenu du rapport (maximum 8 pages) et le code (lisibilité, structure, **commentaires**,...) et comptera pour 3 points sur 20 dans la note finale (le reste des points étant donné par l'examen écrit). Le rapport doit être très professionnel [1], du type article scientifique ou technique, et doit contenir les références sur lesquelles vous vous êtes basés (y compris les librairies Python). Les différents fichiers, c'est-à-dire les fichiers de code source (deux fichiers dans un même directory ShortestPath : celui contenant la fonction main et celui contenant les fonctions Dijkstra, Bellman_Ford et Floyd_Warshall), le fichier .csv et le rapport en pdf, tous compressés ensemble (nom du fichier compressé : mot "groupe" et numéro du groupe concaténés (2 chiffres), suivi par les noms de famille des membres du groupe séparés par des underscore et par ordre alphabétique ; par exemple "groupe05_Eloi_Vancompernolle_Saerens" [2]), sont à remettre sur Moodle au plus tard le dernier jour avant le début du blocus de la session de janvier (dimanche 22 décembre 2024), avant 23h55. Si vous rendez le projet en retard, nous retirons 1 point sur 20 (note du projet) plus 1 point par jour de retard, mais nous n'acceptons pas plus de 3 jours de retard (au-delà du 25/12, ce sera 0 pour le projet). Par exemple, si vous le rendez à 23h58 le jour de la deadline, vous aurez $-1/20$. Si vous le rendez le lendemain, ce sera $-2/20$. La note sera la même pour tous les membres du groupe.

Notez que vous pouvez vous inspirer de Chat GPT, mais vous devez bien comprendre l'algorithme et réécrire le code *à votre façon*. En effet, nous pourrons poser une question d'examen relative à la matière de ce projet, par exemple le pseudo-code d'un des algorithmes implémentés.

Bon travail!

---

1. Par exemple pas de copier/coller d'images de formule mathématique ou d'algorithme.
2. Nous ne corrigeons pas les projets qui ne respectent pas cette consigne : vous devrez re-soumettre le projet avec pénalités.