Université Polytechnique Hauts-de-France
Institut National des Sciences Appliquées Hauts-de-France **2023-2024**
FISA Informatique 4 Optimisation et Graphes
Semestre 8

Fiche d'exercices pratiques - 12h

Consignes & modalités

Les TPs sont à réaliser en présentiel.
Le travail individuel est encouragé mais les binômes sont autorisés. Dans tous les cas vous devez indiquer en haut de votre (vos) fichier(s) source(s) vos nom(s) et prénom(s).
Ne négligez pas les commentaires pour expliquer vos choix et les fonctions mises en œuvre.

Vous devez remettre une version de votre travail avant la fin de chaque séance sur `Moodle`. Si vous ne déposez pas de travail la note associée sera 0 (**aucun retard accepté**). Une version finale pourra être déposée à partir du lendemain de la dernière séance, et dans la semaine qui suit celle-ci. Si elle est déposée, cette version finale sera prise en compte en complément dans la notation (**attention :** il s'agit bien d'un complément, et pas d'un remplacement).

Le langage de programmation est libre. **Attention** cependant, il n'est bien entendu pas autorisé d'utiliser des bibliothèques ou d'autres outils de manipulation de graphes (i.e., c'est à vous d'écrire ces fonctions, même si elles existent).

Le respect des consignes sera pris en compte dans la notation.

Il est conseillé de lire l'intégralité de l'énoncé avant de commencer le développement.

1 Plus court chemin

Dans cette partie on s'intéresse au problème de plus court chemin dans un graphe, pondéré ou non (i.e., un graphe ou un réseau), entre un sommet source et un sommet destination. Dans la suite de l'énoncé on utilisera pour simplifier le terme graphe pour faire référence à l'outil mathématique et informatique pour la résolution du problème, et le terme réseau pour faire référence à l'application.

Pour modéliser et résoudre le problème le graphe peut ainsi être défini à partir d'un ensemble de sommets associés à des villes, des intersections, ou tout autre élément du réseau (routier, ferroviaire, de télécommunications, etc.) utile pour indiquer un possible changement de direction. Deux sommets x et y sont classiquement reliés entre eux par un arc s'il est possible d'aller de x à y, selon la topologie du réseau (s'il s'agit d'un réseau routier certaines rues peuvent être à sens unique par exemple, et si un coût est associé aux connexions il est possible qu'il soit différent pour aller d'un point à un autre, selon le sens du trajet). Dans ce TP on considérera pour simplifier que le coût associé au déplacement d'un point vers un autre est le même dans les deux sens, ce qui nous permet d'utiliser un graphe non orienté.

Gestion des données

Dans ce TP on représentera schématiquement le réseau via un tableau à deux dimensions dans lequel certaines cases ne sont pas accessibles, comme illustré dans la Figure 1.

Le tableau contient uniquement des valeurs entières dans $\{0, 3\}$ selon les règles suivantes : un 1 représente une case qui peut être parcourue, un 0 une case associée à un obstacle. Les valeurs 2 et 3 sont particulières et représentent respectivement le départ et l'arrivée, qui peuvent se trouver n'importe où dans le réseau.
On associe un *point* à chaque case du tableau qui correspond schématiquement au centre de la case

1	2	1	1	1	1	1	1	0
1	0	1	1	1	1	0	1	1
1	1	1	0	0	1	0	1	1
1	0	1	1	1	1	3	1	1

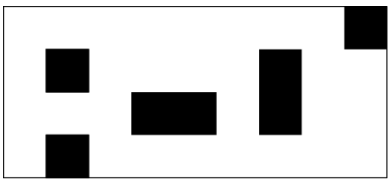

FIGURE 1 – Illustration d'un réseau.

dans la figure. On suppose que chaque case est un carré dont la longueur d'un côté est d'une unité (peu importe l'unité). Ainsi à partir d'une configuration de réseau donnée on peut construire le graphe associé, comme dans la Figure 2 pour notre exemple.
Attention : pour ne pas surcharger la figure les coûts associés aux arêtes ne sont pas mentionnés dans cette figure.
Les coûts des arêtes peuvent cependant être facilement déterminés : deux cases voisines sur une même ligne ou une même colonne sont distantes d'une unité, alors que deux cases voisines sur une diagonale sont distantes de $\sqrt{2}$ unités.
Notez également qu'il sera nécessaire dans le TP de calculer la distance en ligne droite entre deux points du réseau (donc entre deux sommets du graphe). Pour cela on pourra associer à chaque sommet deux coordonnées dans le plan, en choisissant par exemple arbitrairement d'utiliser le point en bas à gauche comme sommet origine de coordonnées $(0, 0)$. Cela permet de facilement calculer la distance entre deux points quelconque du réseau.

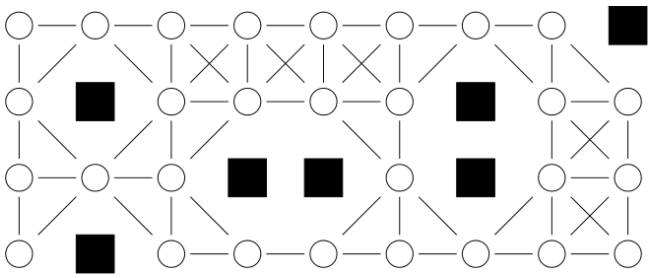

FIGURE 2 – Graphe associé au réseau.

Les informations relatives au réseau sont récupérées via un fichier texte dont le format est le suivant : la première ligne du fichier contient les dimensions du réseau : la hauteur (nombre de lignes du tableau, notée n) et la largeur (nombre de colonnes du tableau, notée m), séparées par un espace. Les n lignes suivantes contiennent m valeurs dans $\{0, 1, 2, 3\}$, séparées par un espace.
On garantit que les n lignes ne contiennent qu'une seule valeur 2 et une seule valeur 3.
La taille limite du réseau est fixée à $n = 500$ et $m = 500$.
Lors du lancement du programme le nom du fichier texte doit pouvoir être passé comme argument.

1.1 Construction du graphe

La première partie du travail consiste à récupérer les informations du réseau et construire le graphe associé, à partir d'un fichier texte donné.
Pour cela vous proposerez :

1. Une structure de données permettant de regrouper toutes les informations que vous jugez utiles pour manipuler un graphe (non orienté ici).
2. Un ensemble de fonctions permettant d'implémenter les fonctionnalités de base d'un graphe (ajout / suppression d'un sommet, d'une arête, etc.)
3. Une fonction de création du graphe à partir des informations contenues dans un fichier texte dont le nom physique est passé en paramètre.

Vous pourrez vérifier le bon fonctionnement de ces fonctions sur différents réseaux dont les fichiers texte sont disponibles sur `Moodle`.

Attention : cette première partie est fondamentale pour la suite du TP, il faut donc veiller à son bon fonctionnement avant de passer à la suite (au moins au niveau implémentation).

1.2 Modélisation mathématique

Dans cette deuxième partie vous allez résoudre le problème du plus court chemin entre deux sommets distincts `s` (la source ou l'origine) et `t` (la destination ou le puits) via une méthode d'optimisation. Pour cela vous proposerez :

1. Une modélisation de ce problème sous la forme d'un programme linéaire. Vous décrirez précisément ce modèle dans un (court) compte-rendu à joindre à vos fichiers sources dans le rendu sur `Moodle`.
2. Une ou plusieurs fonctions permettant d'implémenter et résoudre ce modèle mathématique en utilisant le logiciel `CPLEX`. Il faudra résoudre le modèle et récupérer une solution afin de pouvoir construire le plus court chemin entre les deux sommets particuliers du réseau.
3. Une fonction de sauvegarde d'une solution du problème dans un fichier texte dont le nom sera obligatoirement de la forme `sol_fichier.txt` où `fichier` correspond au nom du fichier texte contenant le réseau. Ainsi si le fichier d'entrée s'intitule `reseau1.txt` alors votre fichier solution s'appelle forcément `sol_reseau1.txt`.

Vous pourrez vérifier le bon fonctionnement de votre méthode sur de petits réseaux dans un premier temps, afin de pouvoir éventuellement contrôler le résultat "à la main". Vous pouvez ensuite tester la méthode pour d'autres réseaux plus grands.

1.3 Algorithme de cheminement

Dans cette troisième partie vous allez considérer une alternative pour résoudre ce problème du plus court chemin d'un sommet vers un autre. Plusieurs solutions sont envisageables, notamment l'utilisation des algorithmes de Dijkstra (on sait ici que les coûts sur les arêtes sont forcément positifs) ou de Bellman-Ford par exemple. Ces deux algorithmes permettent en fait de déterminer un plus court chemin entre un sommet source et tous les autres sommets du réseau.
Vous allez considérer une autre approche qui est souvent utilisée et plutôt bien adaptée pour le cas où le chemin cherché est entre une source et une destination uniquement : l'algorithme A*. Cet algorithme peut être vue comme une extension de l'algorithme de Dijkstra en intégrant un critère heuristique pour guider la recherche. En fait l'idée générale de la méthode peut être résumé succinctement comme suit : plutôt que de considérer tous les sommets adjacents au sommet courant, pourquoi ne pas se focaliser en priorité sur le ou les sommets les plus proches dans la direction de la destination. Schématiquement cela revient en quelque sorte à dire que le chemin en ligne droite est le plus rapide pour relier les deux points.
Vous trouverez une description de cet algorithme ainsi qu'un pseudo-code ici : https://en.wikipedia.org/wiki/A*_search_algorithm
Le principe de base de l'algorithme consiste à choisir le prochain sommet vers lequel se déplacer, disons x, ayant la plus petite valeur définie par $f(x) = g(x) + h(x)$, avec $g(x)$ le coût du chemin depuis le nœud de départ jusque x, et $h(x)$ une valeur heuristique qui estime le coût du chemin le plus court pour aller de x à la destination. Plusieurs distances peuvent être utilisées pour obtenir cette valeur $h(x)$, en particulier la distance de Manhattan ou la distance en ligne droite. Vous utiliserez cette dernière estimation (qui ne prend donc pas en compte la présence des obstacles sur les chemins).
Plusieurs optimisations peuvent être réalisées, notamment au niveau des structures de données utilisées. Il n'est pas forcément demandé ici d'aborder ces optimisations, même s'il est intéressant de prendre conscience de leur impact sur la complexité de l'approche.
On peut noter que l'algorithme de Dijkstra est un cas particulier de l'algorithme A* lorsque l'on fixe $h(x) = 0$ pour tous les sommets du graphe.

1. Proposer une implémentation de l'algorithme A* pour résoudre le problème. Votre algorithme doit permettre de récupérer une solution et de sauvegarder celle-ci dans un fichier texte, sur le

même modèle que dans la question 3 de la partie 1.2.

2. Comparer et commenter le fonctionnement de votre algorithme dans le cas $h(x) = 0$ ou en utilisant la distance en ligne droite sur les différents réseaux fournis (notez qu'il est possible d'implémenter l'algorithme plus efficacement qu'en passant par cette méthode).
3. Comparer les temps de calcul et les solutions obtenues en utilisant le modèle mathématique et la méthode de la section 1.2 avec ces deux méthodes.

2 Problème du voyageur de commerce

Le problème du voyageur de commerce est un problème très connu et très étudié en optimisation. Il fait parti des problèmes NP-Difficile pour lesquels il n'existe pas d'algorithme en temps polynomial permettant d'obtenir une solution optimale en temps et ressources raisonnables. L'énoncé du problème du voyageur de commerce est le suivant : étant donné un ensemble S de points (généralement des "villes"), et connaissant les coûts $d(i, j)$ associés au déplacement entre tout couple de sommets $(i, j) \in A \times A$, trouver un chemin de longueur totale minimale passant exactement une fois par chaque point. Formellement, une instance du problème est un graphe non orienté complet $G = (S, A, D)$ avec S l'ensemble des sommets, A l'ensemble des arêtes et D la matrice des coûts. On cherche alors un cycle hamiltonien de coût minimum dans le graphe G [1].

Dans ce TP une instance du problème sera associée à un fichier texte ayant le format suivant. La première ligne contient le nombre de sommets du graphe et le nombre d'arêtes, séparés par un espace. On a ensuite autant de lignes qu'il y a d'arêtes, avec sur chaque ligne les numéros des deux sommets incidents à l'arête et le coût de cette arête, avec un espace entre chaque valeur. Les sommets sont numérotés à partir de 0.

1. Dans un premier temps vous proposez une fonction pour récupérer les informations contenues dans un fichier dont le nom physique et donné et construire le graphe associé. Il s'agit normalement d'une adaptation de votre fonction de la question 3 de la partie 1.1 pour prendre en compte le format du fichier.

Vous allez ensuite considérer deux façons de résoudre ce problème : via sa modélisation et la résolution de ce modèle, ou via une méthode d'énumération.

2. Donner une formulation du problème du voyageur de commerce sous la forme d'un programme linéaire en nombres entiers [2]. Vous décrirez précisément ce modèle dans le compte-rendu déjà évoqué dans la partie 1.2 de ce TP.
3. Proposer une ou plusieurs fonctions permettant d'implémenter et résoudre ce modèle mathématique par `CPLEX` sur de petits graphes.

Pour tester le modèle et vérifier son bon fonctionnement vous générez quelques graphes aléatoires en utilisant par exemple la méthode proposée par Erdös-Rényi, et qui peut être résumée comme suit :

— Choisir une probabilité p ;
— Créer n sommets ;
— Chaque paire (x, y) de sommets est connectée de façon indépendante selon la probabilité p.

Ici il faut également tirer au sort un coût associé à chaque arête. Celui-ci peut par exemple être tiré au sort dans $[10, 50]$ (on notera qu'avec cette méthode le coût représente autre chose que la distance entre les sommets du graphe car il n'y a pas de garantie de maintenir l'inégalité triangulaire).

4. Tester votre implémentation du modèle sur de petits graphes générés avec la méthode ci-dessus. Analyser et commenter les résultats observés.

1. https://fr.wikipedia.org/wiki/Problème_du_voyageur_de_commerce
2. Voir la formulation de Miller-Tucker-Zemlin : https://en.wikipedia.org/wiki/Travelling_salesman_problem