Université Polytechnique Hauts-de-France
Institut National des Sciences Appliquées Hauts-de-France **2022-2023**
FISA 4A Graphes et Algorithmes
Semestre 8

Fiche d'exercices pratiques

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`. Celle-ci n'est composée que des fichiers sources réalisés pour le TP. 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.

Partie 1 : représentation et manipulation d'un graphe

Dans ce TP on manipule des graphes orientés et pondérés (i.e., des réseaux) : à chaque arc est associé une origine, une destination, et un coût (réel, ≥ 0). Ces graphes permettront de modéliser les problèmes de plus court chemin et de flot dans un réseau. 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. Un sommet sera caractérisé par un identifiant, un entier, unique.

Cette première partie est consacrée au développement des fonctionnalités de base pour construire le graphe associé au problème traité. Vous pouvez vous baser sur ce que vous aviez réalisé dans le TP de `Graphes et Algorithmes` du semestre 6, afin de :

1. Définir une structure de données permettant de regrouper toutes les informations que vous jugez utiles pour manipuler un graphe orienté pondéré.
2. Les fonctions de manipulation qui vous semblent nécessaire pour un tel graphe, notamment :
   (a) Création d'un graphe (vide ou non)
   (b) Ajout d'un sommet - suppression d'un sommet
   (c) Ajout d'un arc - suppression d'un arc
   (d) Savoir si deux sommets sont adjacents
   (e) Charger un graphe depuis un fichier texte - sauvegarder un graphe dans un fichier texte

Pour les fonctions du dernier point le fichier texte a nécessairement le format suivant :
Une première ligne avec le nombre de sommets puis le nombre d'arcs (séparés par un espace). Puis on a une ligne pour chaque arc contenant les deux identifiants des extrémités (*origine* espace *destination*) suivis d'un espace et du coût de l'arc. Ainsi par exemple on pourrait avoir un fichier intitulé

`mon_graphe.txt` contenant :
7 6 4.5
1 2 3.0
1 3 7.2
3 4 5.1
5 1 4.1
5 6 7.8
4 7 2.5

Partie 2 : plus court chemin

Modélisation mathématique

On considère dans un premier temps 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 proposez :

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
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 données du graphe. Par exemple si le fichier d'entrée s'intitule `reseau1.txt` alors votre fichier solution s'appelle forcément `sol_reseau1.txt`. Le fichier contiendra la liste des identifiants des sommets sur le plus court chemin (séparés par un espace), dans l'ordre de parcours, sur une première ligne. La valeur du plus court chemin sera sur une seconde ligne.

Vous pouvez vérifier le bon fonctionnement de votre méthode sur les petits réseaux afin de pouvoir éventuellement contrôler le résultat "à la main".

Algorithme de cheminement

Dans cette 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 arcs 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.

1. Proposer les fonctions nécessaires à l'implémentation de la méthode de Bellamn-Ford pour résoudre ce problème.
2. Ajouter une fonction de sauvegarde d'une solution dans un fichier texte, sur le même modèle que dans la partie précédente.

Comparaison - grands graphes

Pour faire une comparaison de performance entre le modèle résolu par `CPLEX` et votre implémentation de l'algorithme de Bellman-Ford nous avons besoin de graphes possiblement de grande taille, et ayant des caractéristiques différentes.

1. Reprendre le générateur de graphes du module du semestre 6 (méthode de Erdös-Rényi), et l'adapter pour y ajouter la génération aléatoire de coûts sur les arcs (chaque arc aura un coût tiré au hasard entre 1.0 et 10.0)
2. Effectuer des tests avec les deux approches et donner quelques éléments de comparaison sur les performances observées, en faisant varier la taille du graphe et sa densité.

Partie 3 : Flot maximum dans un réseau

Dans cette partie on s'intéresse au problème de circulation d'objets (voitures, informations, etc.) dans un réseau (routier, informatique, etc.). Ce problème correspond au problème de l'écoulement d'un flot de valeur maximale dans un réseau et il est définit à partir d'une source `s` (sommet n'ayant pas de prédécesseur) et d'un puits `t` (sommet n'ayant pas de successeur). Dans ce problème le coût des arcs correspond à la capacité d'objet pouvant passer sur l'arc.

On considère ici des réseaux dans lesquels il n'y a qu'une seule source et qu'un seul puits, et on cherche à un trouver un flot réalisable de valeur maximale.

1. Proposer et implémenter une modélisation mathématique de ce problème dans `CPLEX`
2. Vérifier le bon fonctionnement de cette approche sur des réseaux de petite taille que vous construisez "à la main".
3. Implémenter une version de l'algorithme de Ford-Fulkerson permettant de résoudre ce problème.

Vous pouvez ensuite réfléchir à comment adapter le générateur de graphes afin de s'assurer que les graphes obtenus ne contiennent qu'une seule source et un seul puits (exactement).

Vous pouvez alors tester les deux approches sur des graphes de taille et densité variable, comme dans la Partie 2.

Partie 4 : sac-à-dos avec contraintes disjonctives

Le problème du sac-à-dos avec contraintes disjonctives est une extension du problème classique du sac-à-dos dans laquelle il y a une incompatibilité entre certains couples d'objets. Ce problème peut être formuler à partir d'un graphe de conflit $G = (S, A)$, avec $S = \{1, \ldots, n\}$ l'ensemble des n objets, et $(i, j) \in A$ si et seulement si les objets i et j sont incompatibles. On note que le graphe associé est non orienté.

A chaque objet $i \in S$ est associé un profit $c_i \geq 0$ et un poids $a_i \geq 0$. Connaissant la capacité maximale du sac b, le problème revient à chercher un sous-ensemble d'objets deux à deux compatibles, qui maximise le profit total obtenu en respectant la capacité du sac.

1. Proposer et implémenter une formulation mathématique du problème dans `CPLEX`.
2. Vérifier le bon fonctionnement du modèle sur de petites instances.
3. Proposer une approche *simple* pour construire une solution faisable pour ce problème.
4. Tester l'approche proposée sur des instances de plus grande taille (fournies) et comparer les résultats avec ceux de `CPLEX`

Les instances fournies ont le format suivant :
La première ligne du fichier contient le nombre d'objets (n), puis le nombre d'incompatibilité et enfin la capacité du sac (b).
Ensuite on a les n profits des objets.
Viennent ensuite les n poids des objets.
Enfin on a une ligne par incompatibilité, avec à chaque fois les deux sommets incompatibles (attention les numéro des sommets commencent à 1).

Rappels

Le plagiat est passible de commission de discipline et de sanctions.
Il n'y a aucun intérêt (pour vous comme pour l'équipe enseignante) que vous fournissiez une version d'un ou plusieurs algorithmes disponibles sur Internet et / ou proposée par un autre membre de la promotion.