Efrei Paris, 2023
Théorie des graphes
Projet : « *Métro, Boulot, Dodo* »

youssef.ait-el-mahjoub@efrei.fr

1 Le contexte

Ce projet est à réaliser en groupe de 4 étudiants (exceptionnellement en groupe de 3). Il consiste, dans un premier temps, d'écrire un programme permettant de déterminer le plus court itinéraire pour aller d'une station à une autre dans le métro Parisien. Dans un second temps, vous êtes amenés à y extraire l'arbre couvrant de poids minimum. Le programme est à écrire dans le langage de votre choix, de préférence : Python, C, Java, C++, ...

Date de remise du projet (sur Moodle) : **31/10/2022 - 23h59**.

A la fin du projet, il conviendra de rendre un dossier avec les noms des étudiants « $nom1\_nom2\_nom3\_nom4.zip$ ». Le dossier doit contenir :

- Un rapport au **format pdf** décrivant le travail effectué, les structures de données choisies, les principales procédures, l'explication de la méthode algorithmique utilisée ... le rapport ne doit pas dépasser 10 pages.
- Le code de votre programme ainsi qu'un fichier « Makefile » ou « script »permettant de lancer et exécuter votre programme.

2 Les données

- Les données représentent un graphe $G(V, E)$ non orienté.
- Les données sont à récupérer dans le fichier « metro.txt ».
- Ouvrir le fichier, comprendre comment il est organisé. Pourquoi certains sommets sont dupliqués (exemple « Arts et Métiers »).
- Observez les stations terminus, les branchements selon les directions.
- Ce fichier date de (1998 - 2002), certaines nouvelles stations sur des lignes qui ont été rallongées n'y figurent pas. Il ne parait pas nécessaire de les ajouter.

3 Le travail à faire

3.1 Connexité

Il est possible que le fichier « metro.txt » soit incomplet et que quelques rares liaisons y aient été omises. Votre premier travail de programmation doit

donc consister à développer un module qui vérifie que votre graphe est bien connexe, c'est à dire qu'ils est possible d'aller de n'importe quelle station à n'importe quelle autre. S'il s'avérait que des liaisons manquaient pour assurer cette connexité, à vous bien sûr de les rajouter.

3.2 Le plus court chemin

Il convient de permettre à un utilisateur d'indiquer une station de départ et une station d'arrivée et que votre programme calcule et indique le meilleur itinéraire pour aller de l'une à l'autre.

Résultat minimal : L'utilisateur doit pouvoir saisir une station de départ et une station de destination (exemple ci-dessous « Carrefour Pleyel » et « Villejuif, P. Vaillant Couturier »). L'itinéraire du PCC calculé doit être quelque chose comme :

- Vous êtes à Carrefour Pleyel.
- Prenez la ligne 13 direction Châtillon-Montrouge.
- A Champs Élysées, Clémenceau, changez et prenez la ligne 1 direction Château de Vincennes.
- A Palais Royal, Musée du Louvre, changez et prenez la ligne 7 direction Villejuif, Louis Aragon.
- Vous devriez arriver à Villejuif, P. Vaillant Couturier dans environ 29 minutes.

3.3 L'arbre couvrant

Ici, il vous est demandé d'extraire l'arbre couvrant de poids minimum (ACPM) en utilisant l'algorithme de Kruskal. Vous êtes libres de choisir la façon d'afficher votre ACPM.

4 Les bonus

— **Partie PCC** : Une carte du métro Parisien affichée à l'écran (ou tout du moins une partie de celle-ci). L'utilisateur clique sur les stations de départ et d'arrivée et le plus court chemin s'affiche alors directement sur le plan de métro à l'écran, la durée du trajet est affichée aussi.

— **Partie ACPM** : L'utilisateur clique sur une zone dans la carte (haut à gauche par exemple), ensuite l'ACPM est affiché sur le plan de métro.

Vous disposez de la carte du métro « metrof_r.png » et des coordonnées des stations dans « pospoints.txt » ... vous êtes aussi libres d'imaginer d'autres types d'interfaces graphiques.

Bon courage !