**Travail Pratique 3**

**Le Metro de Paris**

L’objectif de ce travail est de vous familiariser avec les graphes, et les algorithmes de manipulation des graphes.

Contrairement aux autres TPs que vous avez eu à faire dans le cadre de ce cours. Aucun code ne vous sera fourni, à part le main et quelques classes objets. Vous être donc libre de tous les choix d’implantation. Vous pouvez, si vous le désirez, utiliser une librairie standard pour les graphes. Vous êtes toutefois tenu de coder les algorithmes demandés. Vous devrez aussi produire un bref rapport (1-2 paragraphes est suffisante) qui explique vos choix d’implantations (liste de successeur ou matrice d’adjacence).

**Le métro de Paris**

Le fichier metro.txt contient un descriptif du métro de Paris (France). Le fichier est divisé en 3 sections, séparé par des ‘$’ et structuré comme suis :

-La première ligne contient le nombre total de stations (376) suivi du nombre total de liens (arcs) directs entre les stations du métro (933). Vous pouvez supprimer cette ligne si vous la jugez superflue.

Les 376 lignes suivantes listent les stations. Chaque ligne contient un numéro séquentiel unique, suivi du nom d’une station. Si une station est un arrêt sur plus d’une ligne du métro, le fichier contiendra une ligne distincte, avec un numéro séquentiel distinct, pour chaque ligne de métro qui passe par cette station. Ainsi, le fichier contient 3 lignes :

0016 Bastille  
0017 Bastille  
0018 Bastille

par ce que 3 lignes de métro différentes passent par la station « Bastille ».

La ligne suivante contient simplement un symbole $, qui sépare les sections. Les 376 lignes suivantes donnent la position en x et en y de chaque station. Notez bien dans le cas où une station se trouve sur plus d’une ligne, les coordonnées pourraient ne pas être exactement les mêmes.

Un second ‘$’ sépare les positions des stations de la section finale du document. Celle-ci contient 933 lignes, et donne les connexions entre les stations. Chaque ligne correspond à une connexion, et contient 3 nombres soit le numéro de la station de départ, le numéro de la station d’arrivée et le temps requis pour passer d’une station à l’autre. Il s’agit d’arcs dirigés, car le métro de Paris comporte un petit nombre de « sens uniques ». Dans le cas où on change de ligne à une station, le temps correspond au temps d’attente entre les trains (120s).

**Les fichiers fournis**

**Main.java** : Pour vous aider, nous vous fournissons un fichier main, qui contient une méthode pour ouvrir et lire le contenu du fichier. Vous pouvez modifier\remplacer cette méthode. Elle ne sera pas évaluée.

Le main contient aussi les signatures des méthodes que vous devez créer. Vous êtes libre de créer d’autres méthodes utilitaires pour organiser votre code d’une manière logique et lisible (en fait ceci est fortement recommandé).

**Station.java** : La classe de base pour une station du métro. Chaque nœud du graphe sera de ce type.

**Trajet.java** : L’objet retourné par les méthodes trajetLePlusRapideDijkstra et trajetLePlusRapideBellManFord.

**Le code à produire**

Vous devez créé une classe station, qui contiendra les informations relatives à une station (nom, numéro, etc.).

Vous devez ensuite intégrer ces nœuds dans un graphe **dirigé**, dont les détails d’implantation sont à votre discrétion.

Vous devez ensuite compléter les méthodes dont la signature est donnée dans le fichier, soit :

**public** Trajet trajetLePlusRapideDijkstra (String debut, String fin)

**public** Trajet trajetLePlusRapideBellManFord (String debut, String fin)

**public** ArrayList<Station> stationsCritiques()

Les deux premières retournent le trajet le plus rapide entre deux stations (conseil : commencez par vous créer une méthode pour passer du nom d’une station à son numéro et vice versa). Ces méthodes retournent un objet de type trajet, qui contient le trajet et le temps requis, en secondes (la somme des poids des arcs).

Il arrive parfois que le service de transport doive fermer une station pour son entretien. Deux stations ne sont jamais fermées simultanément. Une station est dite « critique » si la fermeture de cette station rend au moins une autre station inaccessible. Conseil : rajouter un booléen « en service » dans la classe Station, et créer une méthode utilitaire « accessible » qui détermine si le métro forme un graphe connexe malgré la fermeture d’une station. Puis, fermez tour à tour chaque station du métro.

**Travail à remettre :**

Remettez la totalité du code, incluant les classes fournis sur moodle. Une seule soumission par équipe est suffisante. Si vous travaillez en équipe, incluez un fichier noms.txt qui contient les noms des deux (maximum ferme) co-équipiers. Finalement, incluez un bref rapport qui justifie vos choix d’implantation (matrice d’adjacence ou liste de successeurs, etc.) pour le graphe.