# Rapport du projet Metro

TAILY Pierrick 21600098 -

06 mai 2020

## 1 Information sur le projet

Le programme demander devait permettre de choisir des stations de metro parmis celle existante et de calculer le chemin le plus court entre les deux chemins donné en amont.
Des données ont été ajouté dans le fichier metro.txt, tel que l'autorisait le projet, nous avons ajouté les numéros de ligne ainsi que les terminus des lignes pour connaître les directions et les arrêts finaux.

## 2 Structures de données

Les déclarations des structures de données sont dans *graph.h*.

**SUBWAY** contient le graphe qui représente le metro parisien.

**GRAPH** contient un tableau de 2 stations remplis par les stations choisis par l'utilisateur, un entier représentant le nombre de station dans le graphe, un tableau des stations avec leur nom et leur numéro de ligne, et un tableau à 2 dimensions où chaque chemin est déterminés avec leurs temps de trajet.

**POINT** contient les informations d'un sommet. Il y a le nom de la station, l'id de la station, le numero de la ligne dans le réseau du metro parisien.

**PATH** contient les informations d'un chemin entre deux stations, avec le numéro de la station du terminus et le temps en seconde du chemins.

**DIJKSTRA**, stock le plus court chemin calculé avec l'algorithme de Dijkstra, contient l'id de la station de départ, l'id de la station d'arrivée.

# 3 Fonctionnement général

Tout d'abord, on initialise les différentes variables que l'on va utiliser pour lancer nos différents fonctions ensuite on initialise le graphe du metro avec "init_subway" grâce au nombre de station qu'on a récupérer précédemment avec "count_nb_points". Les chemins sont initialisé à nul au début.

Le tableau des stations est alors initialisé pour contenir les informations de chaque station (le nom de la station, l'id de la station, le numéro de sa ligne). Ce processus est réalisé par la fonction "init_station".

Le tableau des chemins entre les stations est aussi initialisé avec le numéro du terminus et le temps en seconde de chaque chemin. Sachant que le metro est à double sens, le chemins opposé est également initialisé avec le même temps en seconde mais avec le terminus opposé. Ce processus est réalisé par la fonction "init_system".

Suite à cela on demande à l'utilisateur d'entré la gare à partir de laquelle il souhaite partir.

Une fois que l'utilisateur valide la gare de départ, on vérifie que son entré, qui peut soit être le nom de la station ou son id, correspond aux gares récupérer dans le fichier metro.txt puis l'id de la gare est stocké dans une variable.

Ensuite on demande à l'utilisateur d'entré la gare d'arrivé où il souhaite se rendre et on répète le processus de vérification et de stockage de la donnée.

Puis on lance la fonction "operation_short_way" qui va calculer le plus court chemin entre la station de départ et la station d'arrivé avec la fonction "operation_dijkstra" puis écrit dans le terminal le plus court chemin avec la fonction "write_way".

Finalement, la mémoire est libérée grâce au fonction "free" pour les variables initialisé au début, "free_graph" pour libérer la mémoire de la structure graph ainsi que "free_way_dijkstra" qui libère la mémoire stocker par l'initialisation de la structure dijkstra dansla fonction "operation_dijkstra".

# 4 Fonctionnement de l'Algorithme Dijkstra

— On stocke le fait que aucune station n'ait été traité, aucune station n'ait encore de père, les plus petites distances sont encore indéfinis.

— On traite la station de départ et on met à jour les plus petites distances avec ses successeurs en utilisant les temps en secondes.

— Tant qu'on a des stations à traiter, on prend la plus petite des plus petites distances et on regarde si on peut amélioré les plus petites distances de ces successeurs en passant par ce sommet. Si c'est le cas alors ce dernier devient son père.

— Le tableau des pères est alors initialisé complétement.