Lola Pires Pinto 21706287

Rapport du projet Métro

Présentation du projet

Ce projet consiste à écrire un programme informatique permettant de déterminer le plus court itinéraire pour aller d'une station à une autre dans le métro Parisien. Ce projet est écrit en langage C.

Description des structures de données

La structure **SOMMET** contient les informations d'un sommet. On y trouve ainsi le numéro du sommet, le nom de la station et le numéro de la ligne correspondant.

La structure **GRAPHE** contient les informations du graphe. On y trouve ainsi, le nombre de sommets qu'il contient, un tableau de sommets de la taille du nombre de sommet du graphe et un tableau monodimensionnel qui pour chaque sommet indique s'il a été visité ou non.

Le tableau **d[TAILLE]** qui indique la distance du sommet de départ à chaque sommet.

Le tableau **predecesseur[TAILLE]** qui indique pour chaque sommet son prédécesseur.

Le tableau **poids[TAILLE][TAILLE]** qui indique le poids des arrêtes entre chaque sommet.

Fonctionnement général

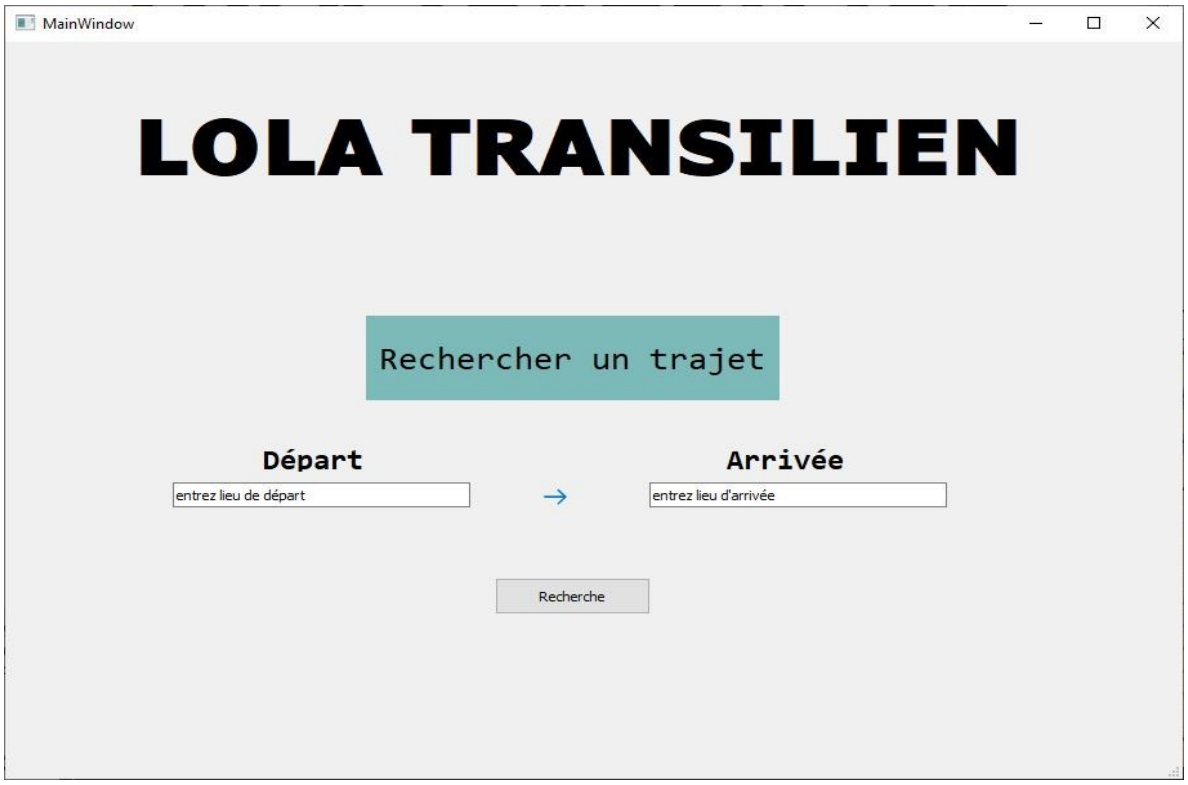
Tout d'abord, on initialise le graphe. On initialise le nombre de sommets du graphe et initialise les deux tableaux de la structure.
On initialise ensuite les sommets du graphe. Pour cela, on lit dans le fichier «metro.txt» et au fur et à mesure de la lecture on stock pour chaque sommet leurs numéros, les noms des stations ainsi que leur ligne.
On initialise ensuite le poids des arrêtes correspondant au temps en seconde entre les stations en lisant dans le fichier «metro.txt» également.
On applique alors la fonction djikstra sur l'ensemble de ces données.

Principale fonction: l'algorithme djikstra

Cet algorithme nous permet de calculer le plus court chemin entre deux stations du métro parisien. Son implémentation est la suivante;
On commence par initialiser les données à partir du premier sommet choisit. On initialise la distance du sommet initial à zéro et à l'infini pour tous les autres sommets. On marque également le sommet de départ comme exploré. Et on met ensuite à jour les distances avec ses voisins.
On choisit un sommet non exploré de distance minimum avec le sommet initial et on le marque comme exploré. On explore ensuite tous les autres sommets afin de pouvoir trouver ses voisins.
On met à jour ainsi la distance des voisins et les prédécesseurs. Si la nouvelle distance trouvée et inférieur à celle initial alors on la remplace. On parcourt de cette façon tous les autres sommets.
On appelle par la suite la fonction *le_plus_court_chemin* qui renvoie le plus court chemin en remontant les prédécesseurs du sommet final au sommet initial. On stock ce chemin dans un tableau afin de faciliter la fonction d'affichage.
Pour finir, une fonction d'affichage affiche l'itinéraire trouvé.

Améliorations possible

1) Possibilité de donner le nom de la station de départ et d'arrivée plutôt que leurs numéros.
2) Voici un exemple d'interface graphique que l'on pourrait développer. Elle pourrait récupérer un sommet de départ et un sommet d'arrivée et d'en afficher l'itinéraire et éventuellement le plan du métro.
Ce n'est ici que de l'affichage. Voici le résultat et une portion du code correspondant.




```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>914</width>
    <height>619</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <widget class="QPushButton" name="pushButton">
    <property name="geometry">
     <rect>
      <x>380</x>
      <y>450</y>
      <width>121</width>
      <height>31</height>
     </rect>
    </property>
    <property name="text">
     <string>Recherche</string>
    </property>
   </widget>
   <widget class="QFrame" name="frame">
    <property name="geometry">
     <rect>
      <x>280</x>
      <y>230</y>
      <width>321</width>
      <height>71</height>
     </rect>
    </property>
    <property name="palette">
     <palette>
      <active>
       <colorrole role="Base">
        <brush brushstyle="SolidPattern">
         <color alpha="255">
          <red>255</red>
          <green>255</green>
          <blue>255</blue>
         </color>
        </brush>
       </colorrole>
```