Rendu du Projet C++ Avancé - 4AI02

**Enoncé du sujet :**[**Sujet du projet**](https://moodle-sciences.upmc.fr/moodle-2018/pluginfile.php/544260/mod_assign/intro/CppAvanceTPprojet.pdf)

**Fichiers du projet à compléter :**[**Archive des fichiers du projet**](https://moodle-sciences.upmc.fr/moodle-2018/pluginfile.php/544260/mod_assign/intro/archive_etudiant.zip?time=1554809664646)

**Roadmap and style guide :**[**English version**](https://moodle-sciences.upmc.fr/moodle-2018/pluginfile.php/544260/mod_assign/intro/CppAvanceTPprojet_en.pdf)

**Récapitulatif :**

Le projet consiste à encapsuler et rédiger un algorithme du plus court chemin, l'algorithme de Dijkstra. Cet algorithme célèbre est à la base de la détermination du trajet optimal d'un point A à un point B, pour le métro parisien par exemple.   
  
Pour ce faire, deux bases de données de deux fichiers chacune sont fournis, réprésentant un réseau simple (stations.csv, connections.csv) et le réseau RATP (s.csv, c.csv). Le projet devra fonctionner avec le réseau RATP (plus dense) mais il est préférable de commencer avec le réseau simple car il est possible de le tracer à la main pour vérifier ses trajets.   
  
Le sujet s'articule autour de 3 classes:   
- Generic\_mapper.hpp, qui calcule le trajet, qui dérive de   
- Generic\_connection\_parser.hpp, s'occupant de répertorier les connexions dérivant lui même de   
- Generic\_station\_parser.hpp, qui lui gère les stations.   
  
L'implémentation doit être ascendante, on commencera à implémenter la classe pour les stations, qui ensuite seront reliés en connections, pour enfin estimer des trajets.   
  
Une quartrième classe Grade.hpp implémente deux objets de test statique, qui pourront être appelés en fournissant une instanciation du projet. Cette classe retournera des codes erreurs en cas de problèmes, et sert donc à évaluer le bon fonctionnement de l'implémentation proposée. L'utilisation de cette classe est facultative et ne rentre pas dans l'évaluation, elle sert uniquement d'aide afin que vous puissez vérifier votre implémentation.  
  
L'objet evaluate\_small évalue le reseau simple, tandis que evaluate évalue le réseau RATP. Les erreurs sont attrapables via une gestion des exceptions. 

**Quelques exemples d'implémentation avec le trajet Saint-Lazare / Bastille, du réseau RATP:**

*Test 1/ On remarque dans le fichier s.csv que Saint-Lazare à pour id le nombre 1722, et 2062 pour Bastille. Si l'on a crée un programme qui prend quatre arguments, le fichier de stations, celui de connections, l'identitifant de station de départ, et celui d'arrivée, on peut avoir un programme se comportant comme ceci :* 

**% ./mon\_programme "FICHIER\_STATIONS" "FICHIER\_CONNECTIONS" 1722 2062**

Arg 1: stations

Arg 2: connections

Arg 3: start

Arg 4: end

Best way from Saint-Lazare (line 3) to Bastille (line 1) is:

Walk to Saint-Lazare, line 14 (180 secs)

Take line 14 (SAINT-LAZARE - OLYMPIADES) - Aller

From Saint-Lazare to Châtelet (368 secs)

Walk to Châtelet, line 1 (180 secs)

Take line 1 (CHATEAU DE VINCENNES - LA DEFENSE) - Aller

From Châtelet to Bastille (367 secs)

After 1095 secs, you have reached your destination!

*Test 2/ evaluate.dijkstra(obj, false) // test avec les id de stations* 

**% ./mon\_programme "FICHIER\_STATIONS" "FICHIER\_CONNECTIONS"**

Arg 1: stations

Arg 2: connections

On lance les tests:

============================ Grade 1 ============================

Stations: seems ok

============================ Grade 2 ============================

Connections: seems ok

============================ Grade 3 ============================

First tests

Tests: seems ok

Would you try complete test (wait ~ 7 mins and 57 secs) ? (N/y): n

*Test 3/ evaluate.dijkstra(obj, true) // test avec les noms de stations* 

**% ./mon\_programme "FICHIER\_STATIONS" "FICHIER\_CONNECTIONS"**

Arg 1: stations

Arg 2: connections

On lance les tests:

============================ Grade 1 ============================

Stations: seems ok

============================ Grade 2 ============================

Connections: seems ok

========================== Grade 3 bis ==========================

First tests

Tests: seems ok.

Would you try complete test (wait ~ 43 mins and 24 secs) ? (N/y): y

0% 2 secs elapsed, wait 32 mins and 49 secs

0% 5 secs elapsed, wait 34 mins and 18 secs

0% 8 secs elapsed, wait 37 mins and 32 secs

0% 11 secs elapsed, wait 36 mins and 39 secs

0% 14 secs elapsed, wait 36 mins and 50 secs

0% 17 secs elapsed, wait 36 mins and 16 secs

0% 19 secs elapsed, wait 35 mins and 25 secs

1% 22 secs elapsed, wait 34 mins and 46 secs

1% 24 secs elapsed, wait 34 mins and 36 secs

Une fois de plus, l'implémentation proposée est un exemple, votre projet pouvant se comporter légèrement différement si nécéssaire.

|  |  |
| --- | --- |
|  | archive_etudiant.zip [archive\_etudiant.zip](https://moodle-sciences.upmc.fr/moodle-2018/pluginfile.php/544260/mod_assign/introattachment/0/archive_etudiant.zip?forcedownload=1) |
|  | CppAvanceTPprojet_en.pdf [CppAvanceTPprojet\_en.pdf](https://moodle-sciences.upmc.fr/moodle-2018/pluginfile.php/544260/mod_assign/introattachment/0/CppAvanceTPprojet_en.pdf?forcedownload=1) |

|  |  |
| --- | --- |
|  | CppAvanceTPprojet.pdf [CppAvanceTPprojet.pdf](https://moodle-sciences.upmc.fr/moodle-2018/pluginfile.php/544260/mod_assign/introattachment/0/CppAvanceTPprojet.pdf?forcedownload=1) |