Dijkstra's algorithm for traffic data planning project

Yousri Aboudaoud
28712992
yousri.aboudaoud@etu.sorbonne-universite.fr

Abstract

The hereby document aims to implement in C++ the work done by (Dijkstra, 1959) and apply it in a subway traffic application.

1 Introduction

The aim of this project is to create an application that determines the shortest route between two stations on the Paris metro network.

In graph theory, a directed graph is a set of points (here, entrances to metro stations) connected by directed links (here, connections between stations, in seconds).

In the context of this project, each link represents a possibility to connect two points, in a specific direction, and at a specific cost. It's a possible connection between these points. The shortest path problem, in this system, is to evaluate the minimum cost of going from point A to point B. Dijkstra's algorithm is a computational method for solving this problem.

2 Building a Network: A Detailed Implementation in project.cpp

The main file is the project.cpp that implements a class called "Network" that inherits from another class "Generic_mapper" (Inheriting as well its methods "compute_travel" and "compute_and_display_travel", which details will be seen in a later section) which itself inherits from a class "Generic_connection_parser" (Inheriting the method "read_connections",which details will be seen in a later section), this class inherits as well from the class "Generic_station_parser"(Inheriting the method "read_stations",which details will be seen in a later section).

The file creates an instance of the class "Network", and uses the methods defined in the class, As to which, a detailed explanation will be given in the next section.

3 Detailed description of the methods

The main function where an instance of Network has been implemented uses the following functions:

- read_stations: Function that reads a file that contains multiple informations on stations (name, address, index...) and stores them in a hashmap in the following way stations_hashmap[index]=station (NB: station is a data structure). This functions updates the attribute "stations_hashmap" used in the computation of the shortest path
- read_connections: Function that reads a file that contains stations' names and the transfer time between connected stations and stores them in a hashmap in the following way connections_hashmap[start_station_ID][arrival_station_ID] = transfer_time This functions updates the attribute "stations_hashmap" used in the computation of the shortest path.
- compute_travel: Function that computes the shortest path between two stations, this is the function that implements the said algorithm developed by (Dijkstra, 1959).
- compute_and_display_travel: Function that computes the shortest path between two stations (using the aforementioned compute_travel method and displays the shortest path.

- compute_and_display_travel: Overloaded function that operates the same as the previous function for the only exception that this version uses the names of the station instead of their IDs by converting them using the function "find_id_station"

- find_id_station: Function that finds the ID of a given station using its name.

4 Results

First result is displayed in figure 1. In this example two IDs have been used from the file "connections.csv".

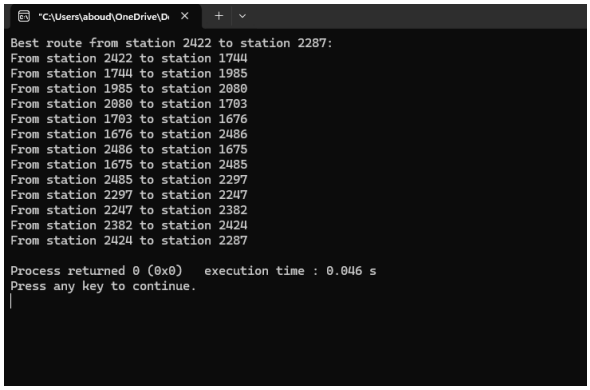

Figure 1: Output example to go from a station 2422 to a station 2287.

Second result is displayed in figure 2. This example uses other IDs picked at random to verify the repeatability of the algorithm

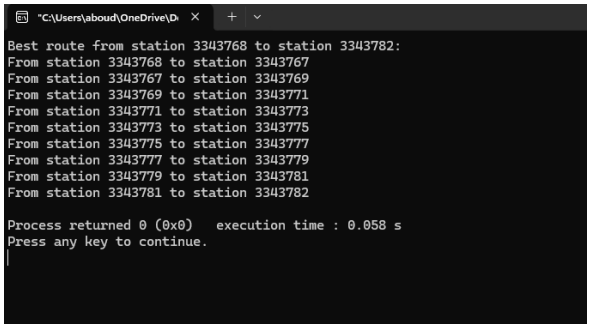

Figure 2: Output example to go from a station 3343768 to a station 3343782.

Finally, a third example is displayed in figure 3. This example shows an application of the algorithm using stations' names instead of their IDs the first part of the image shows the result of the algorithm highlighted in a red box, as for the second part of the image, it shows the implementation that resulted in the first part. Here we can see that the variables "_start_name" and "_end_name" are the names of stations picked at random, which will be converted to stations IDs.

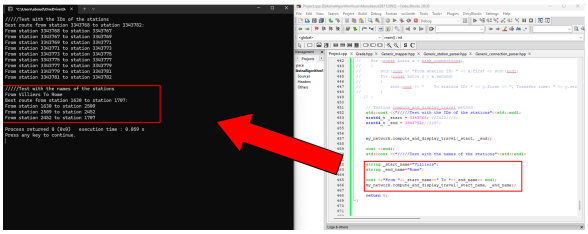

Figure 3: Output example using stations' names instead of their IDs to go from a station Villiers to a station Rome.

References

Edsger W Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.