JavaExpert's

Eurídice Rodrigues - R2016741
Christos Adamakis - E20191118
Waldmilson Vilhete - R20181147

Contents

Introduction

The purpose of this project is develop Graphs Algorithm, seen in the classes.
We have to explore the Object-Oriented concepts and Graphs Theory, principally the algorithms to calculate the shortest path (Bellman-Ford, Prim, and Dijkstra) between two points.

The problem prepared is composed of **163** cities (nodes/vertices) and **337** roads (edges) organized in an interconnected network of roads. The main objective is to implement algorithms to calculate the shortest distance between cities.

Throughout the endless routes from a city to another we are called to find the optimum one. For that purpose we implement the city and road class as well as Dijkstra and Bellman Ford algorithms. The user can chose the origin city, the destination city, the criteria and the algorithm. The results are displayed in a window showing the corresponding results.

The user can specify the criterion he finds more important, total cost, distance or time. Since we implement two algorithms we are able to make comparison between the results of each for the same configurations.

A uml diagram was created so we can display in a more interpretable way the code.

Methods

During the computation classes we were able to get some techniques and knowledge with respect to object oriented program, class diagrams, algorithms of graphs, and so on.In addition, the project that we are now presenting is based on all those learning and concepts that we have been gathering.

We could successfully complete the project through an ensemble of tools that contributed to the performance of the works. Tools like this integrated development environment called “Intellij”, a software written in java, provides us a huge advantage when writing our code because it can analyze our code, looking for connections between symbols across all project files and languages.
Using this information it provides in depth coding assistance, quick navigation, clever error analysis, and, of course, refactorings.
Other tools used were the concepts and conventions of object oriented program.

Analysis

There are three classes which holds data for cities,routes and roads, so let us start first by describing each of them and later move on to the objects which actually utilizes these classes.

The cities class is created to help us store the information needed from the cities csv file. For our purposes we extract the city id, name, district, warehouse. Also the class need to have an arraylist of roads so we can later store all the roads that have as origin city the equivalent city.

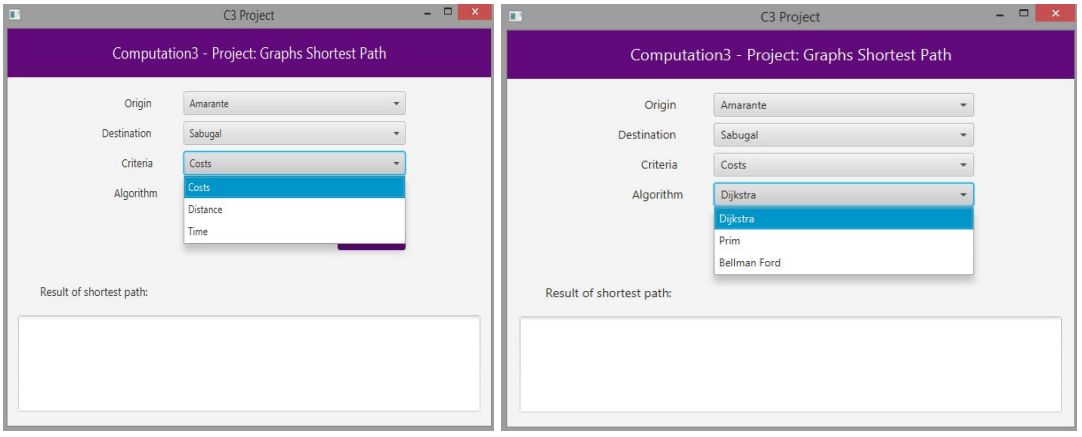
The road class takes data from a road csv containing information for the road id, name, origin and destination city name as well as the distance covered, the time needed to reach the destination and some variables related to the cost.

We implement the Dijkstra Algorithm as well as Bellman Ford.
For Dijkstra we consult the theoretical code displayed in the equivalent presentation of Computation III course.
The cost from origin to origin is implemented to zero. In order to be able to run this code we insert a double table with the size of the count of cities so we can store there the cost from the origin to the destination. Moreover we need a queue for the cities so we know which ones we have visited.
Cities have ids running from zero to 163 so we took advantage of this fact and help us access the tables. The city with ith id has an index for the table like i - 1.
We also added two functions to help us. The first one is to help us determine when to terminate the procedure of the algorithm and the second one’s purpose is to find the minimum cost of road so we can assign a new value to continue the algorithm.
In the end we extract the results containing toll costs or distance or time.

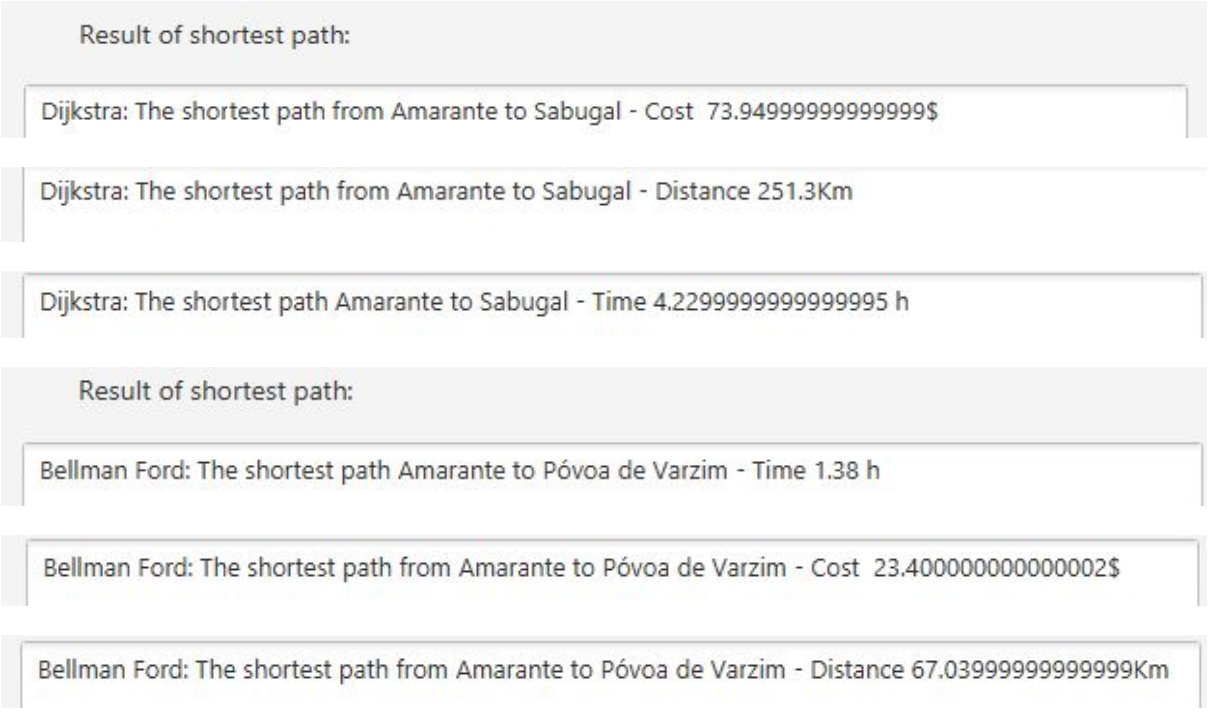
For Bellman Ford we took our inspiration from an online code listed in the end.
Similarly with Dijkstra we need a table for to store the cost.
However, there is not a need for a queue since we use for loops, but we created a graph for all the roads of the datasets. We repeat the procedure for V-1 times and V is the size of cities class.
Helpful functions were created in order to initiate and fill the graph.
The results are returned in a similar manner to Dijkstra.

Results

The results of each code are logical and the algorithm was able to give us optimal results for each pair of origin and destination cities that have roads.
As for the comparison almost always we get the same result from Dijkstra and Bellman Ford. That is a good sign for us since it means both of them are able to find the same results. We can assume that there are indeed some routes that are better and we can enforce it with the results of both algorithms.
For illustration the user can choose in the manner displayed:



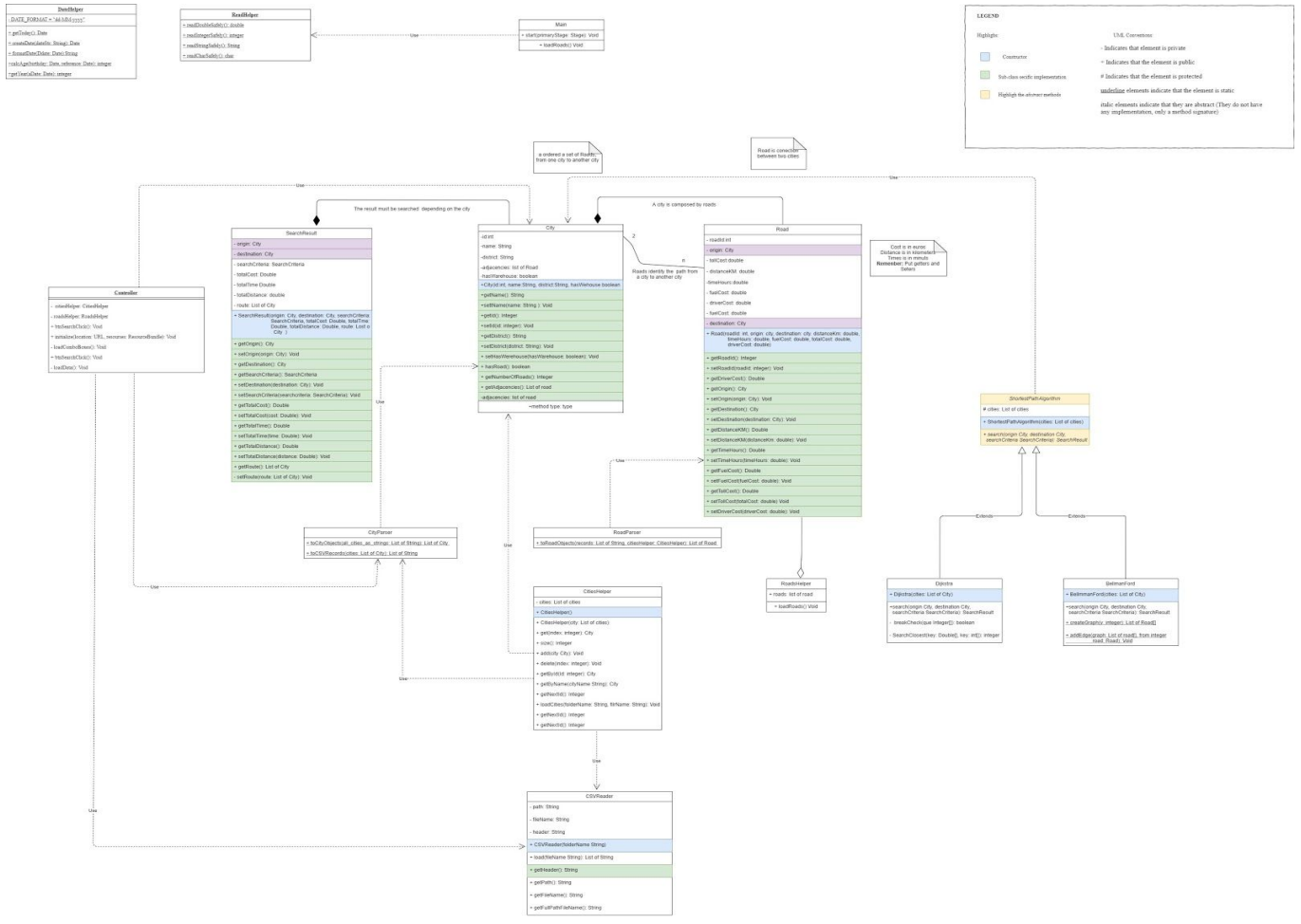
The results are shown like below:



Conclusion

Regarding the Intellij we can say that it was a really useful program for analysis.
As for our results we were able to create a program that gave us the results we needed. There are not many differences on the results of each algorithm so we can assume that both are fitting to our purposes.
Cities that are close geographically gave usually lower numbers on results on contrast to cities far away which is reasonable.
Algorithms like Dijkstra and Bellman Ford are really useful to solving problems like finding the cheapest road and they can be applied in order to reduce the cost of traveling.

Annex
UML Diagram



References

- Contents from the theoretical Classes
- Some Input data made available by professor
- Link for the implementation code of BellmanFord Algorithm:

https://github.com/williamfiset/Algorithms/blob/master/com/williamfiset/algorithms/graphtheory/BellmanFordAdjacencyList.java