PROJECT REPORT

METRO MAP

1. Problem Statement:

In this project we will be showing the routes and the shortest distance routes and their minimum fare. As there is a need for a system which shows all the paths as well as most importantly the shortest path between source and destination. There are many people who travel through the metro every day therefore they require a system which shows every service given by the metro.

2. INTRODUCTION :

2.1 Motivation:

This topic for the project was chosen by us because it is much more interesting than the other topics which are mostly related to games and management. This project is slightly different. Secondly the aim of this project is to create something beneficial for the people, it will show you all the routes, the fare related to the routes and the shortest route and in today's life knowing all this information is more important than other things.

2.2 Objective:

1. Users can view the metro station list .
2. User can add as many metro stations as he/she wants .
3. Users can add routes between different stations with adding particular distance between them .
4. User can also view all existing routes between the stations .

2.3 Scope :

1. Provide a route filtering method based on the travel cost difference between an alternative route and the shortest route.
2. Propose a route filtering method based on train operational plans.
3. Put forward a two-step framework to generate route choice sets on metro networks, based on both of the mentioned above.

3. METHODOLOGY

Dijkstra Algorithm

Dijkstra's algorithm allows us to find the shortest path between any two vertices of a graph. When it comes to weighted graphs, it’s not necessary that neighbouring nodes always have the shortest path. However, the neighbour with the shortest edge can’t be reached by any shorter path. The reason is that all other edges have larger weights, so going through them alone would increase the distance. Dijkstra’s algorithm uses this idea to come up with a greedy approach. In each step, we choose the node with the shortest path. We fix this cost and add this node’s neighbours to the queue. Therefore, the queue must be able to order the nodes inside it based on the smallest cost. We can consider using a priority queue to achieve this. We still have one problem. In unweighted graphs, when we reached a node from a different path, we were sure that the first time should have the shortest path. In weighted graphs, that's not always true. If we reach the node with a shorter path, we must update its distance and add it to the queue. This means that the same node could be added multiple times. Therefore, we must always compare the cost of the extracted node with its actual stored cost. If the extracted distance is larger than the stored one, it means this node was added in an early stage. Later, we must have found a shorter path and updated it. We have also added the node again to the queue, so this extraction can be ignored safely.

This project is implemented in C++ language.

1. **Software Required :**
● C++/C ● MS Word ● Web Browser: Microsoft Internet Explorer, Mozilla, Google Chrome or later ● Visual studio code editor and compiler.
2. **Hardware Requirements**
● Processor: Minimum 1 Gz; Recommended 2 Gz or more ● Ethernet connection (LAN) or wireless adapter (Wi-Fi) ● Hard Drive : Minimum 32 GB; Recommended 64 or more ● Memory (RAM): Minimum 1 GB; Recommended 4GB or above

CONCLUSION

This project gives an integrated service which provides all information about the metro rail in Delhi and its routes for the public. Main aim of this project is to give the shortest distance route and hence be very useful to save time. The contribution that this project will be able to make is toward those people who are daily metro travellers.