

**Project:** Metro Reservation System

**Subject**: Data Structures

**Submitted By:**

Amogh Sanjeev Gupta- 16104018

Mahima Jain- 16104039

Ritik Gupta- 16104025

Rizul Singh- 16104017

**Introduction**

A metro reservation system is made by using the concept of graphs, shortest distance, Dijkstra’s Algorithm and common STL concepts for the implementation of the algorithms. The interface takes the source and distance as the input and on the basis of the input, the shortest distance between the two stations is calculated and the fare for the same is calculated and is printed as output.

**Concepts used**

1. **Graphs**

Graph is a data structure that consists of following two components:

1. A finite set of vertices also called as nodes.

2. A finite set of ordered pair of the form (u, v) called as edge. The pair is ordered because (u, v) is not same as (v, u) in case of directed graph(di-graph). The pair of form (u, v) indicates that there is an edge from vertex u to vertex v. The edges may contain weight/value/cost.

Graphs are used to represent many real life applications: Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are also used in social networks like linkedIn, facebook. For example, in facebook, each person is represented with a vertex(or node). Each node is a structure and contains information like person id, name, gender and locale. See this for more applications of graph.

1. **Adjacency matrix**

Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph. Let the 2D array be adj[][], a slot adj[i][j] = 1 indicates that there is an edge from vertex i to vertex j. Adjacency matrix for undirected graph is always symmetric. Adjacency Matrix is also used to represent weighted graphs. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w.

1. **Dijkstra’s Algorithm**

Given a graph and a source vertex in graph, find shortest paths from source to all vertices in the given graph.

Dijkstra’s algorithm is very similar to Prim’s algorithm for minimum spanning tree. Like Prim’s MST, we generate a SPT (shortest path tree) with given source as root. We maintain two sets, one set contains vertices included in shortest path tree, other set includes vertices not yet included in shortest path tree. At every step of the algorithm, we find a vertex which is in the other set (set of not yet included) and has minimum distance from source.

Below are the detailed steps used in Dijkstra’s algorithm to find the shortest path from a single source vertex to all other vertices in the given graph.

Algorithm

1) Create a set sptSet (shortest path tree set) that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.

2) Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.

3) While sptSet doesn’t include all vertices

a) Pick a vertex u which is not there in sptSetand has minimum distance value.

b) Include u to sptSet.

c) Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

**Functions used**

1. Print()- Traverses and prints the shortest path between two stations.
2. isReachable()- Smallest number of stations between source and destination.
3. Add()- Adds station in the list.
4. Search()- Searches the source and destination and returns it’s index
5. Check()- Checks whether the source and destination spellings are correct or not.

**Conclusion**

The interface prompts the user to choose whether he/she wants to book a station. If the user chooses yes, a list of stations is printed out of which the user chooses his source and destination. After that, the interface calculates the shortest distance and minimum fare between the two stations.