Jaypee Institute Of Information Technology

Project Report

The Urban Subway

Submitted To:

Dr. Vivek Kumar Singh

Dr. Manish Kumar Thakur

Ms. Nishtha Ahuja

Submitted By:

Kashish(20103101)

Sparsh Celly(20103102)

Shivansh(20103105)

Certificate

This is to certify that Kashish, Sparsh Celly and Shivansh Srivastava of B-Tech, second year from CSE Branch of Batch B4 have successfully created the project on the topic "**The Urban Subway**" under the guidance of Dr. Vivek Kumar Singh, Dr. Manish Kumar Thakur and Ms. Nishtha Ahuja.

Dr. Vivek Kumar Singh
(Lab faculty)

Dr. Manish Kumar Thakur
(Lab faculty)

Ms. Nishtha Ahuja
(Lab faculty)

DATE OF SUBMISSION:17-12-2021

ACKNOWLEGMENT

We would like to place on record our deep sense of gratitude to Dr. Vivek Kumar Singh, Dr. Manish Kumar Thakur and Miss Nishtha Ahuja for giving us an opportunity to work together and their constant guidance and support throughout the project.

Finally, an honourable mention goes to our friends and family for their support to do us this project. Without help of the particular's mentioned above, we would have faced many difficulties while pursuing this project.

The Urban Subway

Contents

Title-: The Urban Subway

Introduction

Today in this fast pacing world, time has the most significant role. One may think of wasting little amount of money but cannot waste time and travelling via roads with so much traffic is a most common time-consuming activity. What if there's a system in which there's no scope of wasting time? Traffic and this so-called time-consuming activities don't play any role in it. Then the travelling difficulties would be sorted without any issue.

To provide a sorted travelling, we came up with a project "**The Urban Subway**" that provides the user the best way to travel through metro rails. A Metro is a kind of Railway Network that is designed to ferry large numbers of passengers' to short distances using rail cars.

This is a project that provides the shortest metro route between two places to ease the journey and make it economical. This project will also let people know the cost and time of the journey. This project is built in a way to help mankind save time and energy. No issue of traffic or long routes, just a click and one can know how to travel to the destination easily.

Motivation To Work

In our day-to-day life we come across many issues while travelling-The rising prices of fuel, high fare of public automobiles, time consuming traffic, long routes to reach destination, high risks of road accidents and many more. But people rarely use metros because according to them they have to go through the long routes which is more time consuming as half of them doesn't know the shortest route between lines that makes the journey easy. Almost every person has experienced this problem while travelling. This was the significant reason to inspire us to make the project where we can find the shortest routes between stations to reach our destination peacefully, without any sound of horn, without spending lots of money and most importantly without travelling for hours.

Objective

"The Urban Subway" mainly aims at time management and cost reduction. This project is a simple C++ project that tells a person about the shortest route to reach a destination. With the help of different colours we are showing when and where to change the line. The project also notifies the user about the distance between each station that is to be passed and total cost and time of the journey. It manages all the details about the stations and the metro routes. The key objective is to build a project that can help users get free from travel objections. The project is a result of a lot of hard-work of the team members.

Functionality

- **To display output in a neat and clean manner.**

  The project output that tells us about the shortest route via metro is presented in an understandable form so that none type of confusion is created.

- **To give all the specifications about the stations.**

The output will inform the user about all the stations he have to pass to reach his destination in less amount of time.

- **To show the shortest path to reach destination.**

  There can be many ways to reach the destination. The path that is displayed by the project is the shortest path among all routes possible.

- **To give the interchange station.**

  The output will display the station where the user should interchange to reach the final destination. To make this more understandable the output will be shown using different colours.

- **To display the path change**

  To help the user reach destination, the program will show interchanges between different colour lines.

- **To calculate total fare of the journey.**

  The project's output notifies the user about the total fare the journey will cost him calculated according to the per km formula

- **To display total time of the journey.**

  The project also tell about the total time the user will take to reach the destination.

Implementation

This project uses graph data structure to find the shortest route (edges according to graph) between two stations. The graph works according to a node, so one station is node to measure the distance for the second node and where ever the distance will be shortest will be considered as the best route to travel. Also, the total time and cost will be calculated and displayed (calculated according to per km.). The program uses weighted and directed graph as data-structures to help us work efficiently. A weighted graph is a graph in which each branch is given a numerical weight. A weighted graph is therefore a special type of labelled graph in which the labels are numbers (which are usually taken to be positive).

Here these numbers are taken as distances between two stations/nodes.
A directed graph, also called a digraph, is a graph in which the edges have a direction. This is usually indicated with an arrow on the edge and here it will show the direction of the metro line.

Software

VS Code, Codeblocks

Code

```
#include <bits/stdc++.h>

using namespace std;

vector<pair<int,int>> graph[101];

int dist[101];

int V=101;

vector<int> arr;

int fare;

int timetaken;


void printGraph()

{

   //auto it = graph[0].begin();

   //cout<<endl<<it->first<<it->second;

   int v, w;


   for (int u = 0; u < 101; u++)

   {

      cout << "Node " << u << " makes an edge with \n";
```

```
        for (auto it = graph[u].begin(); it!=graph[u].end(); it++)
        {
            v = it->first;
            w = it->second;
            cout << "\tNode " << v << " with edge weight ="<< w << "\n";
        }
        cout << "\n";
    }
}
void joinstats(vector <pair<int, int> > graph[], int u,int v, int wt)
{
    graph[u].push_back(make_pair(v, wt));
    graph[v].push_back(make_pair(u, wt));
}
void assign(map<int,string> metro)
{
    map <int, string> :: iterator ptr;
    ptr=metro.begin();

    int i=0;
    //red
    for(; i<=21; i++,ptr++)
    {
        joinstats(graph,ptr->first,ptr->first+1,dist[i]);
    }
    //yellow
```

```
    for(i=22; i<=56; i++,ptr++)
    {
        joinstats(graph,ptr->first,ptr->first+1,dist[i]);
    }

    //blue
    for(i = 58; i<=99; i++,ptr++)
    {
        joinstats(graph,ptr->first,ptr->first+1,dist[i]);
    }

     joinstats(graph,8,30,0);
     joinstats(graph,35,85,0);

}
void printPath(int parent[], int j,int source)
{

   // Base Case : If j is source
    if (parent[j] == source)
        return;

    printPath(parent, parent[j],source);

    //cout<<parent[j]<<" ";
    arr.push_back(parent[j]);
```

```
}
void dijkstra(int source,int dest)
{
    priority_queue<pair<int,int>,vector<pair<int,int> >,greater<pair<int,int> > > pq;
   vector<int> distTo(101,INT_MAX);

   int parent[101];
   parent[0]=-1;
   distTo[source] = 0;
   pq.push(make_pair(0,source));           // (dist,from)

   while( !pq.empty() )
   {
      int dist = pq.top().first;
      int prev = pq.top().second;
      pq.pop();

    //parent done on my own to store path

      vector<pair<int,int> >::iterator it;
      for( it = graph[prev].begin() ; it != graph[prev].end() ; it++)
      {
         int next = it->first;
         int nextDist = it->second;
         if( distTo[next] > distTo[prev] + nextDist)
```

```
            {
                parent[next] = prev;
                distTo[next] = distTo[prev] + nextDist;
                pq.push(make_pair(distTo[next], next));
            }
        }

    }
    printPath(parent,dest,source);

    timetaken = distTo[dest]*2;
    fare = distTo[dest]*3;

}
int main()
{
    map<int, string> metro;

    cout<<"**WELCOME TO DMRC WEBSITE**"<<endl<<endl;
  //inputing station and distance list
   fstream newfile;
   newfile.open("stations.txt",ios::in);
   if (newfile.is_open())
    {
      string name;
      int i=1;
```

```
  while(getline(newfile,name ))
  {
     metro.insert(pair<int, string>(i,name));
     i++;
  }
}
  newfile.close();

fstream newfile2;
newfile2.open("distances.txt",ios::in);
if(newfile2.is_open())
{
  string readdis;
  int i=0;
  while(getline(newfile2,readdis ))
  {
     dist[i]=stoi(readdis);
     i++;
  }
}
  newfile2.close();


  //display
map<int, string>::iterator itr;
cout<<"List of functioning metro stations:"<<endl<<endl;
```

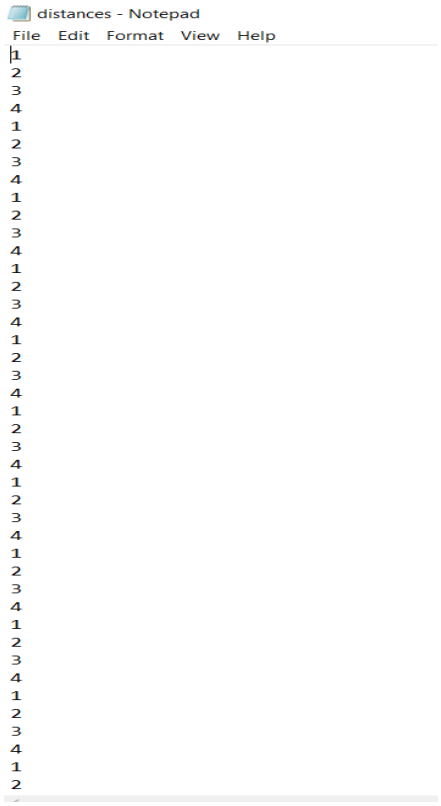
```
cout<<endl<<"RED LINE-"<<endl;
for(itr = metro.begin(); itr != metro.end(); ++itr)
{
   if(itr->first==22)
   {
      cout<<endl<<"YELLOW LINE-"<<endl;
   }
   if(itr->first==57)
   {
      cout<<endl<<"BLUE LINE-"<<endl;
   }
   cout<<" "<<itr->first<<" -> "<<itr->second<<endl;
}

int source=0;
cout<<endl<<endl;
cout<<"Enter your starting metro station number: ";
cin>>source;
cout<<endl<<endl;
if(source==0 || source>101)
   {
      cout<<"ERROR: Wrong entry - Please try again"<<endl;
      return 0;
   }

int dest=0;
```

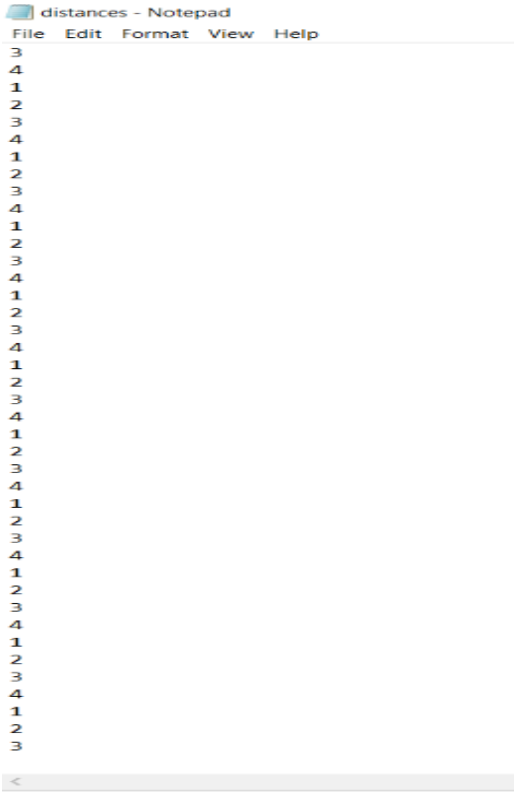
```
    cout<<"Enter your Destination station number: ";
    cin>>dest;
    cout<<endl<<endl;
    if(source==0 || source>101 || dest==source)
       {
          cout<<"ERROR: Wrong entry - Please try again"<<endl;
          return 0;
       }
//making graph
    assign(metro);

    //printGraph();
    dijkstra(source,dest);

    cout<<"This is the most efficient path: "<<endl<<endl;

    itr=metro.find(source);
       cout<<itr->second<<"->";

    for(int j=0; j<arr.size();j++)
    {
       if(arr[j]==30 && arr[j-1]==8)
       {
          cout<<endl<<endl<<"CHANGE FOR YELLOW LINE"<<endl;

       }if(arr[j]==8 && arr[j-1]==30)
```

```
        {
            cout<<endl<<endl<<"CHANGE FOR RED LINE"<<endl;

        }if(arr[j]==85 && arr[j-1]==35)
        {
            cout<<endl<<endl<<"CHANGE FOR BLUE LINE"<<endl;

        }if(arr[j]==35 && arr[j-1]==85)
        {
            cout<<endl<<endl<<"CHANGE FOR YELLOW LINE"<<endl;

        }

        itr=metro.find(arr[j]);
        cout<<itr->second<<" -> ";
    }
    itr=metro.find(dest);
    cout<<itr->second<<endl;
    cout<<endl<<endl;
    cout<<"The total fare is: "<<fare<<" Rupees";
    cout<<endl;
    cout<<"The total time is: "<<timetaken<<" minutes"<<endl<<endl;
   return 0;
}
```

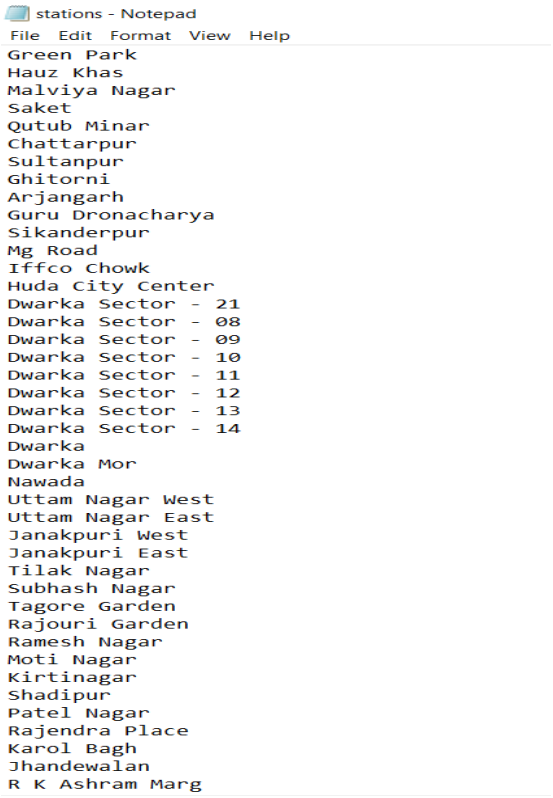
Text files

Distance





Stations



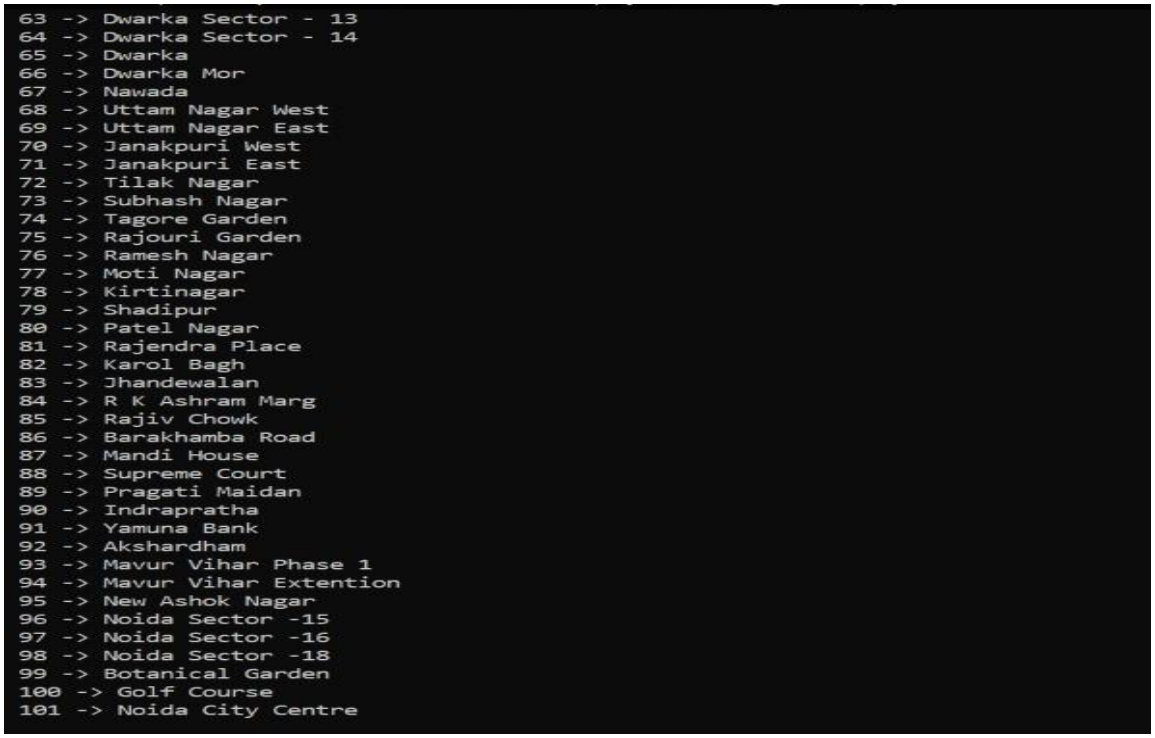
Rajiv Chowk
Barakhamba Road
Mandi House
Supreme Court
Pragati Maidan
Indrapratha
Yamuna Bank
Akshardham
Mavur Vihar Phase 1
Mavur Vihar Extention
New Ashok Nagar
Noida Sector -15
Noida Sector -16
Noida Sector -18
Botanical Garden
Golf Course
Noida City Centre

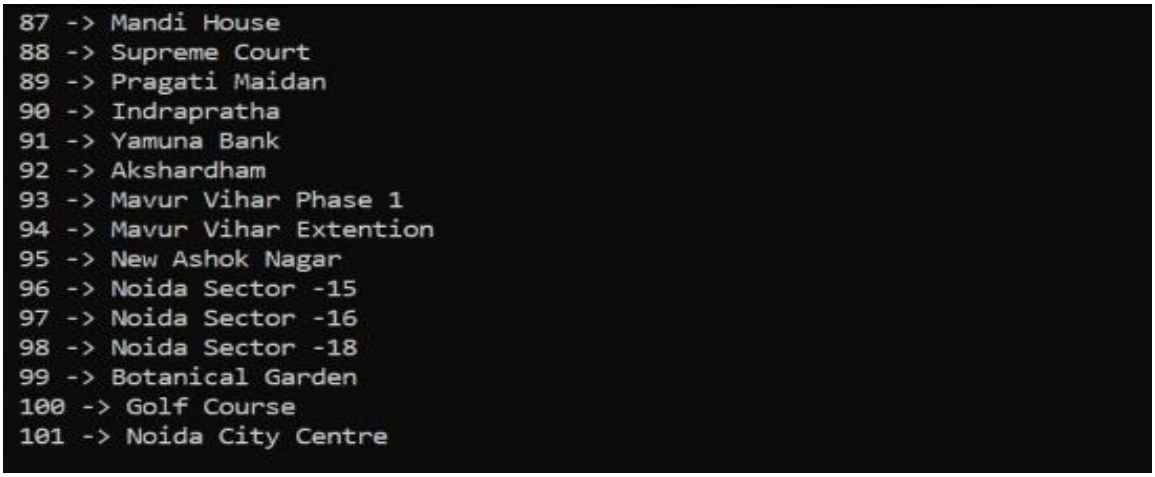
Results and Discussion

As the user enters the home page, list of all the lines are shown according to stations in a proper manner.

```
***WELCOME TO DMRC WEBSITE***

List of functioning metro stations:


RED LINE-
 1 -> Dilshad Garden
 2 -> Jhilmil
 3 -> Mansarover Park
 4 -> Shahdra
 5 -> Welcome
 6 -> Seelampur
 7 -> Shastri Park
 8 -> Kashmere Gate
 9 -> Tis Hazari
 10 -> Pulbangesh
 11 -> Pratap Nagar
 12 -> Shastri Nagar
 13 -> Inderlok Line-1
 14 -> Kanhaiya Naar
 15 -> Keshav Puram
 16 -> Netaji Subhash Place
 17 -> Kohat Enclave
 18 -> Pitampura
 19 -> Rohini East
 20 -> Rohini West
 21 -> Rithala

YELLOW LINE-
 22 -> Jahangir Puri
 23 -> Adarsh Nagar
 24 -> Azadpur
 25 -> Modeltown
 26 -> Gtb Nagar
 27 -> Vishwavidyalaya
 28 -> Vidhan Sabha
 29 -> Civil Lines
 30 -> Kashmere Gate
 31 -> Chandni Chowk
 32 -> Chwari Bazar
 33 -> New Delhi (Dmrc)
```

```
 31 -> Chandni Chowk
 32 -> Chwari Bazar
 33 -> New Delhi (Dmrc)
 34 -> New Delhi (Airport Line)
 35 -> Rajiv Chowk
 36 -> Patel Chowk
 37 -> Central Secretariat
 38 -> Udyog Bhawan
 39 -> Race Course
 40 -> Jor Bagh
 41 -> Ina
 42 -> Aiims
 43 -> Green Park
 44 -> Hauz Khas
 45 -> Malviya Nagar
 46 -> Saket
 47 -> Qutub Minar
 48 -> Chattarpur
 49 -> Sultanpur
 50 -> Ghitorni
 51 -> Arjangarh
 52 -> Guru Dronacharya
 53 -> Sikanderpur
 54 -> Mg Road
 55 -> Iffco Chowk
 56 -> Huda City Center

BLUE LINE-
 57 -> Dwarka Sector - 21
 58 -> Dwarka Sector - 08
 59 -> Dwarka Sector - 09
 60 -> Dwarka Sector - 10
 61 -> Dwarka Sector - 11
 62 -> Dwarka Sector - 12
 63 -> Dwarka Sector - 13
 64 -> Dwarka Sector - 14
 65 -> Dwarka
 66 -> Dwarka Mor
 67 -> Nawada
 68 -> Uttam Nagar West
 69 -> Uttam Nagar East
```





Then it asks for the station number according to the list that from where to start and where to end. Once done, then it shows the most efficient path to reach destination along with the interchanges required.

```
Enter your starting metro station number: 95


Enter your Destination station number: 3


This is the most efficient path:

New Ashok Nagar->Mavur Vihar Extention -> Mavur Vihar Phase 1 -> Akshardham -> Yamuna Bank -> Indrapratha -> Pragati Maidan -> Supreme Court -> Mandi House -> Barakhamba Ro
ad -> Rajiv Chowk ->

CHANGE FOR YELLOW LINE
Rajiv Chowk -> New Delhi (Airport Line) -> New Delhi (Dmrc) -> Chwari Bazar -> Chandni Chowk -> Kashmere Gate ->

CHANGE FOR RED LINE
Kashmere Gate -> Shastri Park -> Seelampur -> Welcome -> Shahdra -> Mansarover Park
```

It also shows the total time according to 2 minutes/km and total fare of the journey according to 3 Rs. / km.

```
The total fare is: 150 Rupees
The total time is: 100 minutes


Process returned 0 (0x0)   execution time : 8.664 s
Press any key to continue.
```

Conclusion

"The Urban Subway" is a project that can help people get the shortest path to reach their destination economically and in minimum time. We made this using C++and put our dense effort to make this work perfectly so that we don't leave any stone unturned. We have tried our level best to make this project authentically with proper data of metro lines and stations. With the help of our torchbearers, we have created this project and hope the results to be fruitful.

References

https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/

https://www.tutorialspoint.com/data_structures_algorithms/graph_data_structure.htm

https://www.programiz.com/dsa/graph

https://en.wikipedia.org/wiki/Graph_(abstract_data_type)