**TRAVEL ROUTE GENERATOR**

**=======================================================**

**Team members:**

Tejeshwar-AP19110010262

Laasya-AP19110010248

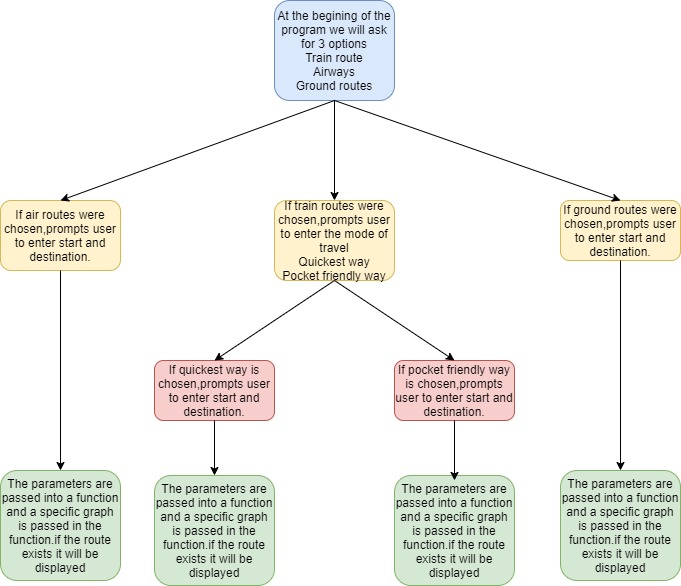
Siva sathvik-AP19110010271

**=======================================================**

**Statement:** Guiding the users the best suitable paths, ground and flight routes and time taken by each of these ways to reach the destination.

**Explanation of the problem:** The problem is to generate a route using the existing data which we will store in form of graphs. The vertices represent the each city that we choose. We cannot consider every city. So we considered 29 cities from 29 states. We have prepared graphs in such a way that each graph contains data of different aspects that we have to consider for generating the paths.

**Flowchart:**

****

**Algorithm:**

It has two parts.One is the path calculator function and other is the main function

**Pathcalculator(graph,vertices,start,end)**

**Step 1:** Create cost matrix C[ ][] from adjacency matrix adj[ ][ ]. C[i][j]) is the cost of going from vertex i to vertex j. If there is no edge between vertices i and j then C[i][j] is infinity**.**

**Step 2:** Array visited[ ] is initialized to zero.

**Step 3:**  If the vertex o is the source vertex then visited[0] is marked as 1.

**Step 4:** Create the distance matrix, by storing the cost of vertices from vertex no. o to n-1 from the source vertex o. Initially, distance of source vertex is taken as o. i.e. distance[0]=0;

**Step 5:** for(i = 1; i < n; i++) – Choose a vertex w, such that distance[w] is minimum and visited[w] is 0. Mark visited[w] as 1

- Recalculate the shortest distance of remaining vertices from the source.

-Only, the vertices not marked as 1 in array visited[ ] should be considered for recalculation of distance.

i.e. for each vertex v if(visited[v]==0) distance[v]=min(distance[v], distance[w]+cost[w][v])

**Main()**

**Step 1:**Defining different graphs that contain different data as their edge weights

**Step 2:**Asking user for the input.

**Step 3:**Taking input from the user.

User has 3 options to give input

**Step 4:**Based on the user input a graph will be selected and passed into the Pathcalculator()

Pathcalculator(graph,vertices,start,end)

**Code:**

**#include<iostream>**

**#include<map>**

**using namespace std;**

**#define MAX 29**

**#define INF 99999**

**int dijk(int G[MAX][MAX],int n,int start,int dest)**

**{**

**int cost[MAX][MAX],dist[MAX],visited[MAX],pred[MAX];**

**int i,j,count,mindist,nextno;**

**map<int,string>m;**

**m[0] = "Amaravati";**

**m[1] = "Itanagar";**

**m[2] = "Dispur";**

**m[3] = "Patna";**

**m[4] = "Raipur";**

**m[5] = "Panaji";**

**m[6] = "Gandhinagar";**

**m[7] = "Chandigarh";**

**m[8] = "Shimla";**

**m[9] = "Srinagar";**

**m[10] = "Ranchi";**

**m[11] = "Banglore";**

**m[12] = "Thiruvananthapuram";**

**m[13] = "Bhopal";**

**m[14] = "Mumbai";**

**m[15] = "Imphal";**

**m[16] = "Shillong";**

**m[17] = "Aizwal";**

**m[18] = "Kohima";**

**m[19] = "Bhubaneswar";**

**m[20] = "Chandigarh";**

**m[21] = "Jaipur";**

**m[22] = "Gangtok";**

**m[23] = "Chennai";**

**m[24] = "Hyderabad";**

**m[25] = "Agartala";**

**m[26] = "Lucknow";**

**m[27] = "Dehradun";**

**m[28] = "Kolkata";**

**for(i = 0;i<n;i++)**

**{**

**for(j = 0;j<n;j++)**

**{**

**if(G[i][j] == 0)**

**{**

**cost[i][j] = INF;**

**}**

**else**

**{**

**cost[i][j] = G[i][j];**

**}**

**}**

**}**

**for(i=0;i<n;i++)**

**{**

**dist[i] = cost[start][i];**

**pred[i] = start;**

**visited[i] = 0;**

**}**

**dist[start] = 0;**

**visited[start] = 1;**

**count = 1;**

**while(count<n-1)**

**{**

**mindist = INF;**

**for(i=0;i<n;i++)**

**{**

**if(dist[i]<mindist && !visited[i])**

**{**

**mindist = dist[i];**

**nextno = i;**

**}**

**}**

**visited[nextno] = 1;**

**for(i=0;i<n;i++)**

**{**

**if(!visited[i])**

**{**

**if((mindist+cost[nextno][i])<dist[i])**

**{**

**dist[i] = mindist+cost[nextno][i];**

**pred[i] = nextno;**

**}**

**}**

**}**

**count++;**

**}**

**int net=0;**

**string s = " ";**

**for(i=0;i<n;i++)**

**{**

**if(i!=start&& i==dest)**

**{**

**net = dist[i];**

**s.append(m[i]);**

**s.append("<-");**

**j = i;**

**do{**

**j = pred[j];**

**s.append(m[j]);**

**s.append("<-");**

**}while(j!=start);**

**}**

**}**

**int len = s.length();**

**s.erase(len-2);**

**cout << s << endl;**

**if(net>0){**

**return net;**

**}**

**else**

**{**

**cout << "This route doesnt exist" << endl;**

**}**

**return 0;**

**}**

**int main()**

**{**

**map<int,string>m;**

**m[0] = "Amaravati";**

**m[1] = "Itannagar";**

**m[2] = "Dispur";**

**m[3] = "Patna";**

**m[4] = "Raipur";**

**m[5] = "Panaji";**

**m[6] = "Gandhinagar";**

**m[7] = "Chandigarh";**

**m[8] = "Shimla";**

**m[9] = "Srinagar";**

**m[10] = "Ranchi";**

**m[11] = "Banglore";**

**m[12] = "Thiruvananthapuram";**

**m[13] = "Bhopal";**

**m[14] = "Mumbai";**

**m[15] = "Imphal";**

**m[16] = "Shillong";**

**m[17] = "Aizwal";**

**m[18] = "Kohima";**

**m[19] = "Bhubaneswar";**

**m[20] = "Chandigarh";**

**m[21] = "Jaipur";**

**m[22] = "Gangtok";**

**m[23] = "Chennai";**

**m[24] = "Hyderabad";**

**m[25] = "Agartala";**

**m[26] = "Lucknow";**

**m[27] = "Dehradun";**

**m[28] = "Kolkata";**

**int G[MAX][MAX] ={{0,0,0,0,735,985,0,0,0,0,0,0,659,1171,1130,1002,0,0,0,0,0,0,0,447,272,0,0,0,0**

**},{0,0,320,1185,0,0,0,0,0,0,0,1354,0,0,0,0,461,388,674,328,0,0,793,0,0,759,0,0,1285**

**},{0,0,934,0,0,0,0,0,0,0,1103,0,0,0,0,496,91,462,364,1459,0,542,0,0,538,1382,0,1033**

**},{0,1185,934,0,742,0,0,1334,0,1433,0,327,0,0,949,0,1406,1001,1372,1274,768,1110,570,0,1470,1448,538,1286,550**

**},{735,0,0,742,0,1353,1152,1486,0,0,0,589,1370,0,629,1124,0,0,0,0,549,1213,1338,1174,779,0,813,1438,923**

**},{985,0,0,0,1353,0,1104,0,0,0,0,0,647,970,1222,574,0,0,0,0,0,0,0,1000,658,0,0,0,0**

**},{0,0,0,0,1152,1104,0,1134,0,1233,0,0,0,0,601,545,0,0,0,0,0,649,0,0,1204,0,1223,1153,0**

**},{0,0,0,1334,1486,0,1134,0,0,113,567,0,0,0,1023,0,0,0,0,0,0,517,0,0,0,0,804,223,0**

**},{0,0,0,1433,0,0,1233,113,0,0,635,0,0,0,1120,0,0,0,0,0,0,614,0,0,0,0,901,226,0**

**},{0,0,0,0,0,0,0,567,0,635,0,0,0,0,0,0,0,0,0,0,0,1272,0,0,0,0,0,980,0**

**},{0,1354,1103,327,589,0,0,0,0,0,0,0,0,0,1020,0,0,1181,0,1454,456,1255,750,0,1368,0,738,1430,408**

**},{659,0,0,0,1370,647,0,0,0,0,0,0,0,683,1443,981,0,0,0,0,1439,0,0,347,569,0,0,0,0**

**},{1171,0,0,0,0,970,0,0,0,0,0,0,683,0,0,0,0,0,0,0,0,0,0,725,1251,0,0,0,0**

**},{1130,0,0,949,629,1222,601,1023,0,1120,0,1020,1443,0,0,776,0,0,0,0,1179,593,0,1476,852,0,647,996,1380**

**},{1002,0,0,0,1124,574,545,0,0,0,0,0,981,0,776,0,0,0,0,0,0,1147,0,1337,721,0,1377,0,0**

**},{0,461,496,0,0,0,0,0,0,0,0,0,0,0,0,0,0,545,425,136,0,0,1018,0,0,539,0,0,0**

**},{0,388,91,0,0,0,0,0,0,0,0,0,0,0,0,0,545,0,375,431,0,0,632,0,0,451,0,0,0**

**},{0,674,462,1372,0,0,0,0,0,0,0,0,0,0,0,0,425,375,0,562,0,0,1002,0,0,336,0,0,1494**

**},{0,328,364,1274,0,0,0,0,0,0,0,1454,0,0,0,0,136,431,562,0,0,0,886,0,0,616,0,0,1378**

**},{0,0,1459,768,549,0,0,0,0,0,0,456,1439,0,1179,0,0,0,0,0,0,0,1097,1232,1053,0,1183,0,440**

**},{0,0,0,1110,1213,0,649,517,0,614,1272,1255,0,0,593,1147,0,0,0,0,0,0,0,0,0,0,575,508,0**

**},{0,0,542,570,0,0,0,0,0,0,0,0,0,0,0,0,1018,632,1002,886,0,0,0,0,0,1056,0,0,670**

**},{447,0,0,0,1174,1000,0,0,0,0,0,0,347,725,1476,1337,0,0,0,0,1232,0,0,0,627,0,0,0,0**

**},{272,0,0,1470,779,658,1204,0,0,0,0,1368,569,1251,852,721,0,0,0,0,1053,0,0,627,0,0,1343,0,1490**

**},{0,759,538,0,0,0,0,0,0,0,0,0,0,0,0,0,539,451,336,616,0,0,1056,0,0,0,0,0,0**

**},{0,0,1382,538,813,0,1223,804,0,901,0,738,0,0,647,1377,0,0,0,0,1183,575,0,0,1343,0,0,550,995**

**},{0,0,0,1286,1438,0,1153,223,0,226,980,1430,0,0,996,0,0,0,0,0,0,508,0,0,0,0,550,0,0**

**},{0,1285,1033,550,923,0,0,0,0,0,0,408,0,0,1380,0,0,0,1494,1378,440,0,670,0,1490,0,995,0,0**

**}};**

**int FT[MAX][MAX] = {{0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,2,1,0,0,0,0},**

**{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3},**

**{0,0,0,2,0,0,0,0,3,0,0,3,0,0,3,1,0,1,0,0,0,1,3,3,3,4,0,1},**

**{0,0,2,0,0,0,3,0,2,0,1,3,0,0,3,0,0,0,0,0,0,0,2,2,0,0,0,1},**

**{0,0,0,0,0,0,2,0,2,0,0,2,0,0,2,0,0,0,0,1,0,0,0,2,0,0,0,0},**

**{0,0,0,0,0,0,2,3,3,0,0,2,0,0,1,0,0,0,0,0,0,0,2,1,0,0,0,3},**

**{0,0,0,2,2,2,0,2,2,0,0,2,5,0,2,0,0,0,0,3,2,0,2,2,0,2,0,3},**

**{0,0,0,0,0,3,2,0,0,1,0,3,0,0,3,0,0,0,0,0,0,0,3,3,0,2,0,3},**

**{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},**

**{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},**

**{0,0,0,1,0,0,0,0,0,0,0,3,0,0,3,0,0,0,0,0,0,0,0,2,0,0,0,1},**

**{2,0,3,3,2,1,2,3,3,0,2,0,1,2,2,0,0,0,0,2,3,3,1,1,3,2,3,2},**

**{0,0,0,0,0,0,3,0,0,0,0,2,0,0,2,0,0,0,0,0,0,0,2,0,0,0,0,4},**

**{0,0,0,0,0,0,0,0,0,0,0,2,0,0,2,0,0,0,0,0,0,0,0,3,0,0,0,0},**

**{0,0,3,2,2,1,1,2,3,3,2,2,2,2,0,0,0,0,0,2,2,3,2,1,0,2,2,2},**

**{0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,4,1,0,0,2},**

**{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},**

**{0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2},**

**{0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,2},**

**{0,0,0,0,0,0,3,0,0,0,0,2,0,0,3,0,0,0,0,0,0,0,2,2,0,0,0,1},**

**{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},**

**{0,0,3,4,0,0,1,0,0,0,0,3,0,0,2,0,0,0,0,0,0,0,3,2,0,0,3,2},**

**{2,0,1,0,0,0,3,0,0,0,0,3,0,0,3,0,0,0,0,0,0,0,2,2,0,0,0,1},**

**{1,0,3,2,0,2,2,0,0,0,0,1,2,0,2,0,0,0,0,2,3,2,0,1,0,3,0,2},**

**{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},**

**{0,0,1,0,0,0,0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1},**

**{0,0,0,0,2,3,2,1,2,0,0,3,0,0,2,0,0,0,0,0,0,0,3,2,0,1,1,1},**

**{0,0,0,0,0,0,2,0,0,0,0,3,0,0,2,0,0,0,0,0,2,0,0,4,0,1,0,0},**

**{0,2,1,1,2,0,3,3,3,0,0,2,0,0,3,2,0,1,2,1,2,1,2,0,1,1,0,0}};**

**int TT[MAX][MAX] = {{0,0,0,34,16,0,0,0,0,0,21,11,0,13,22,0,0,0,0,11,30,0,6,5,0,0,0,75,75**

**},{0,0,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{39,6,0,15,0,0,0,0,0,0,0,52,0,35,44,0,0,0,0,25,0,0,45,45,0,0,0,18,18**

**},{33,0,15,0,0,42,0,21,0,0,0,47,0,18,27,0,0,0,0,16,18,9,41,0,29,0,0,9,9**

**},{16,0,0,0,0,0,0,0,0,0,9,25,0,10,18,0,0,0,0,11,0,0,24,11,0,0,0,14,14**

**},{0,0,0,41,0,0,0,0,0,0,0,0,0,28,12,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{0,0,0,21,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10,0,0,0**

**},{0,0,0,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{21,0,0,7,9,0,0,0,0,0,0,34,0,0,27,0,0,0,0,0,0,0,0,20,0,0,0,7,7**

**},{11,0,0,47,25,0,0,0,0,0,34,0,17,24,23,0,0,0,0,22,41,45,5,10,0,41,0,21,21**

**},{22,0,0,0,40,0,0,0,0,0,0,18,0,39,23,0,0,74,0,37,0,54,16,29,0,52,0,44,44**

**},{14,0,35,18,11,29,0,0,0,0,0,0,0,0,13,0,0,0,0,23,12,27,24,16,52,10,0,0,0**

**},{22,0,44,27,69,11,0,23,0,0,27,24,24,13,0,0,0,0,0,30,17,36,21,14,0,24,0,29,0**

**},{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{0,0,0,0,0,0,0,0,0,0,0,0,74,0,0,0,0,0,0,0,0,18,0,0,0,0,0,28,0**

**},{0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,10,0,0,0,0,0,0,0**

**},{13,0,0,18,11,0,0,0,0,0,0,23,38,23,30,0,0,0,0,0,0,17,18,19,0,25,0,6,0**

**},{0,0,0,18,0,0,0,0,0,0,0,0,0,0,24,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{0,0,0,0,0,0,0,0,0,0,0,41,0,11,17,0,0,0,0,0,0,0,36,28,0,11,0,23,0**

**},{0,0,7,0,0,0,0,0,0,0,0,52,17,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{7,0,0,47,22,0,0,0,0,0,0,5,16,19,21,0,0,0,0,17,39,37,0,13,0,34,0,26,0**

**},{6,0,0,30,12,0,0,0,0,0,23,12,30,0,14,0,0,0,0,18,28,37,14,0,0,27,0,26,0**

**},{0,0,12,0,0,0,0,0,0,0,0,0,0,51,0,0,0,0,0,0,0,19,0,0,0,0,0,37,0**

**},{0,0,0,28,0,0,0,0,0,0,0,0,0,51,0,0,0,0,0,0,0,0,0,0,0,0,0,37,0**

**},{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{20,0,0,11,13,0,0,0,0,0,8,12,0,0,10,0,0,0,0,8,25,0,22,1,38,0,0,0,0**

**}};**

**int TC[MAX][MAX] = {{0,0,0,940,640,0,0,0,0,0,715,130,0,805,525,0,0,0,0,155,260,0,185,415,0,0,0,505,505**

**},{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{745,0,0,775,0,0,0,0,0,0,0,510,0,860,330,0,0,0,0,570,0,0,355,60,0,0,0,750,750**

**},{765,0,755,0,0,40,0,715,0,0,0,710,0,590,820,0,0,0,0,565,715,145,850,0,635,0,0,435,435**

**},{640,0,0,0,0,0,0,0,0,0,435,745,0,680,695,0,0,0,0,505,0,0,790,490,0,0,0,580,580**

**},{0,0,0,40,0,0,0,0,0,0,0,0,0,595,550,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{0,0,0,715,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,495,0,0,0**

**},{0,0,0,0,0,0,0,630,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{715,0,0,170,465,0,0,0,0,0,0,940,0,0,885,0,0,0,0,0,0,0,0,705,0,0,0,355,355**

**},{575,0,0,95,805,0,0,0,0,0,940,0,520,255,480,0,0,0,0,910,850,335,585,485,0,835,0,35,35**

**},{545,0,0,0,540,0,0,0,0,0,0,520,0,770,240,0,0,140,0,985,0,995,375,755,0,140,0,70,70**

**},{170,0,925,670,515,595,0,0,0,0,0,0,0,0,425,0,0,0,0,590,415,815,630,505,35,465,0,0,0**

**},{0,0,330,840,695,550,0,780,0,0,885,480,980,420,0,0,0,0,0,840,560,155,750,445,0,575,0,945,0**

**},{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{0,0,0,0,0,0,0,0,0,0,0,0,140,0,0,0,0,0,0,0,0,445,0,0,0,0,0,695,0**

**},{0,0,465,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,465,0,0,0,0,0,0,0**

**},{0,0,0,465,505,0,0,0,0,0,0,5,785,590,905,0,0,0,0,0,0,510,505,500,0,605,0,765,0**

**},{0,0,0,565,0,0,0,0,0,0,0,0,0,0,780,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{0,0,0,0,0,0,0,0,0,0,0,850,0,415,510,0,0,0,0,0,0,0,520,695,0,425,0,595,0**

**},{0,0,280,0,0,0,0,0,0,0,0,265,560,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{0,0,0,10,790,0,0,0,0,0,0,635,490,990,750,0,0,0,0,365,520,965,0,570,0,980,0,825,0**

**},{0,0,0,765,530,0,0,0,0,0,760,485,755,0,405,0,0,0,0,500,700,985,570,0,0,670,0,600,0**

**},{0,0,465,0,0,0,0,0,0,0,0,0,0,955,0,0,0,0,0,0,0,70,0,0,0,0,0,745,0**

**},{0,0,0,635,0,0,0,0,0,0,0,0,0,955,0,0,0,0,0,0,0,0,0,0,0,0,0,745,0**

**},{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0**

**},{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,889,0,0,0,0,0**

**}};**

**int i,j,n,start,dest;**

**cout << "===================================================================================================" << endl;**

**cout << "=============================WELCOME TO TRAVEL ROUTE PLANNER=======================================" << endl;**

**cout << "===================================================================================================" << endl;**

**cout << "\n";**

**cout << " ->Based on your mode of travel,we will suggest you the best path" << endl;**

**for(i=0;i<29;i++)**

**{**

**if(i%5 == 0)**

**{**

**cout << "\n";**

**cout << i+1 << " - " << m[i] << "\t";**

**}**

**else**

**{**

**cout << i+1 << " - " << m[i] << "\t";**

**}**

**}**

**cout << "\n\n";**

**int a,b,c,d;**

**cout << " Enter mode of travel." << endl;**

**cout << "1.Air 2.Trains 3.Roadway" << endl;**

**cin >> a;**

**if(a==1)**

**{**

**cout << "Flight mode of travel selected" << endl;**

**cout << "Enter the starting point[number] : ";**

**cin >> c;**

**cout << "Enter the destination point[number] : ";**

**cin >> d;**

**int count = 0;**

**int dist = dijk(FT,29,c-1,d-1);**

**cout << "This is the best path between the choosen cities" << endl;**

**cout << "Total flight time approximately is : " << dist << " hours"<< endl;**

**cout << "\n\n";**

**}**

**else if(a == 2)**

**{**

**cout << "Train mode of travel selected." << endl;**

**cout << "Enter your choice[number] : ";**

**cout << "Enter the way of travel" << endl;**

**cout << "1.Quickest way 2.Economically benificial way" << endl;**

**cin >> b;**

**if(b==1)**

**{**

**cout << "Enter the starting point[number] : ";**

**cin >> c;**

**cout << "Enter the destination point[number] : ";**

**cin >> d;**

**int count = 0;**

**int dist = dijk(TT,29,c-1,d-1);**

**cout << "Total time taken approximately is : " << dist << " hours"<< endl;**

**cout << "\n\n";**

**}**

**else if(b==2)**

**{**

**cout << "Enter the starting point[number] : ";**

**cin >> c;**

**cout << "Enter the destination point[number] : ";**

**cin >> d;**

**int count = 0;**

**int dist = dijk(TC,29,c-1,d-1);**

**cout << "Total cost is approximately: " << dist << " rupees"<< endl;**

**cout << "\n\n";**

**}**

**}**

**else if(a==3)**

**{**

**cout << "Roadways selected." << endl;**

**cout << "Enter the starting point[number] : ";**

**cin >> c;**

**cout << "Enter the destination point[number] : ";**

**cin >> d;**

**int count = 0;**

**int dist = dijk(G,29,c-1,d-1);**

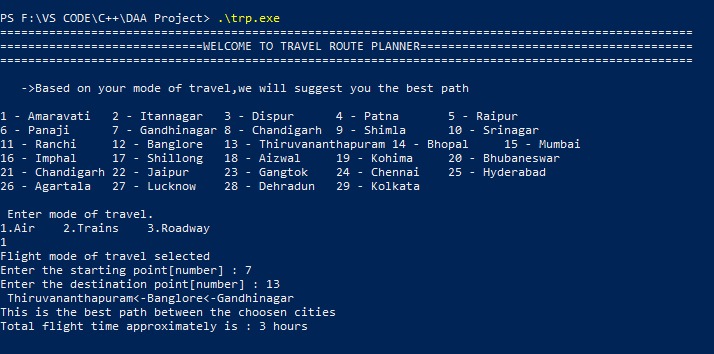
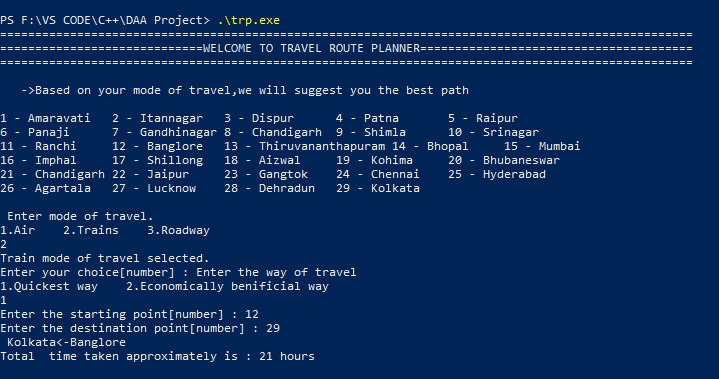
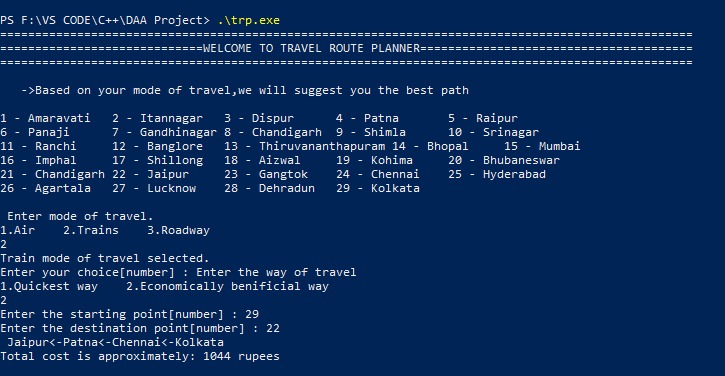
**cout << "Total distance approximately is : " << dist << " kilometers."<< endl;**

**cout << "\n\n";**

**}**

**return 0;**

**}**

**Sample Input and Output:**

**Input and Output Explanation:**

In each input user has 3 choices.

-Air

-Trains

-Roadways

Once the user choses air or roadways,user is prompted to enter the start and destination.Id user chooses trains user has to again enter another input i.e to enter the mode of travel.user has two choices between quick routes and economically beneficial way.Once he enters the values,

If the route exists,the route will be displayed and also the respective cost or distance or time will be displayed.

-For airways,along with the route flight time will be displayed

-For train routes in quick way,along with the route travel time will be displayed.

-For train routes in economically beneficial way,along with the route travel expense will be displayed

-For roadways along with the route total kilometres will be displayed.