ADVANCED DATA STRUCTURE

GROUP B

ASSIGNMENT 5

BATCH B1

YEAR: 2016-17

COLLEGE: VIIT

DATE OF COMPLETION : 30/03/2018

**Title:**

Dijkstra Algorithm/Shortest Path Method

**Problem Statement:**

Tour operator organizes guided bus trips across the Maharashtra. Tourists may have diﬀerent preferences. Tour operator oﬀers a choice from many diﬀerent routes. Every day the bus moves from starting city S to another city F as chosen by client. On this way, the tourists can see the sights alongside the route travelled from S to F. Client may have preference to choose route. There is a restriction on the routes that the tourists may choose from, the bus has to take a short route from S to F or a route having one distance unit longer than the minimal distance. Two routes from S to F are considered diﬀerent if there is at least one road from a city A to a city B which is part of one route, but not of the other route.

**Objective:**

To understand and implement Dijkstra Algorithm in required Applications.

**Software And Hardware Requirement:**

1. 64-bit Open source Linux or its derivative.

2. Open Source C++ Programming tool like G++/GCC.

**Theory:**

Dijkstras algorithm is very similar to Prims algorithm for minimum spanning tree. Like Prims 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.

1

Below are the detailed steps used in Dijkstras 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 doesnt 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

**Algorithm:**

Dijkstra’s Algorithm

1.Make the distance to source vertex as 0.

2.Set all other distances to infinity.

3.’c[]’, set contains all the vertices.

4.’count’, is the no. of vertices.

5.visited[], set of visited vertices is set to 0 for all vertices.

6.while count is not 0, select element of ’c[]’ with the min distance.

7.set 1 in the visited[] set for the corresponding vertex.

8.If shortest path found set the new value of shortest path.

9.repeat the steps 6,7,8 till count is not equal to 0.

**Code :**

/\*Tour operator organizes guided bus trips across the Maharashtra. Tourists may have different

preferences. Tour operator offers a choice from many different routes. Every day the bus

moves from starting city S to another city F as chosen by client. On this way, the tourists can

see the sights alongside the route travelled from S to F. Client may have preference to choose

route. There is a restriction on the routes that the tourists may choose from, the bus has to take

a short route from S to F or a route having one distance unit longer than the minimal distance.

Two routes from S to F are considered different if there is at least one road from a city A to a

city B which is part of one route, but not of the other route.\*/

#include<iostream>

using namespace std;

#define INFINITY 9999

#define MAX 10

class graph

{

public:

int a[MAX][MAX];

int n;

graph()

{

n=0;

}

void readgraph()

{

int i,j;

cout<<"ENTER THE NO. OF VERTICES\n";

cin>>n;

cout<<"ENTER THE ADJACENCY MATRIX\n";

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

{

cout<<"\n";

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

cin>>a[i][j];

}

}

void printgraph()

{

int i,j;

cout<<"\n NO. OF VERTICES\n";

cout<<n;

cout<<"\nADJACENCY MATRIX\n";

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

{

cout<<"\n";

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

cout<<a[i][j];

}

}

void dijikstra(int snode)

{

int c[MAX][MAX],d[MAX],pred[MAX];

int v[MAX],cnt,min,nnode,i,j;

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

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

if(a[i][j]==0)

c[i][j]=INFINITY;

else

c[i][j]=a[i][j];

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

{

d[i]=c[snode][i];

pred[i]=snode;

v[i]=0;

}

d[snode]=0;

v[snode]=1;

cnt=1;

while(cnt<n-1)

{

min=INFINITY;

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

if(d[i]<min&&!v[i])

{

min=d[i];

nnode=i;

}

v[nnode]=1;

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

if(!v[i])

if(min+c[nnode][i]<d[i])

{

d[i]=min+c[nnode][i];

pred[i]=nnode;

}

cnt++;

}

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

{

if(i!=snode)

{

cout<<"\n distance of node"<<i<<"="<<d[i];

cout<<"\npath="<<i;

j=i;

do

{

j=pred[j];

cout<<"<-"<<j;

}while(j!=snode);

}

}

}

};

int main()

{

graph g;

int u;

g.readgraph();

cout<<"ENTER STARTING NODE\n";

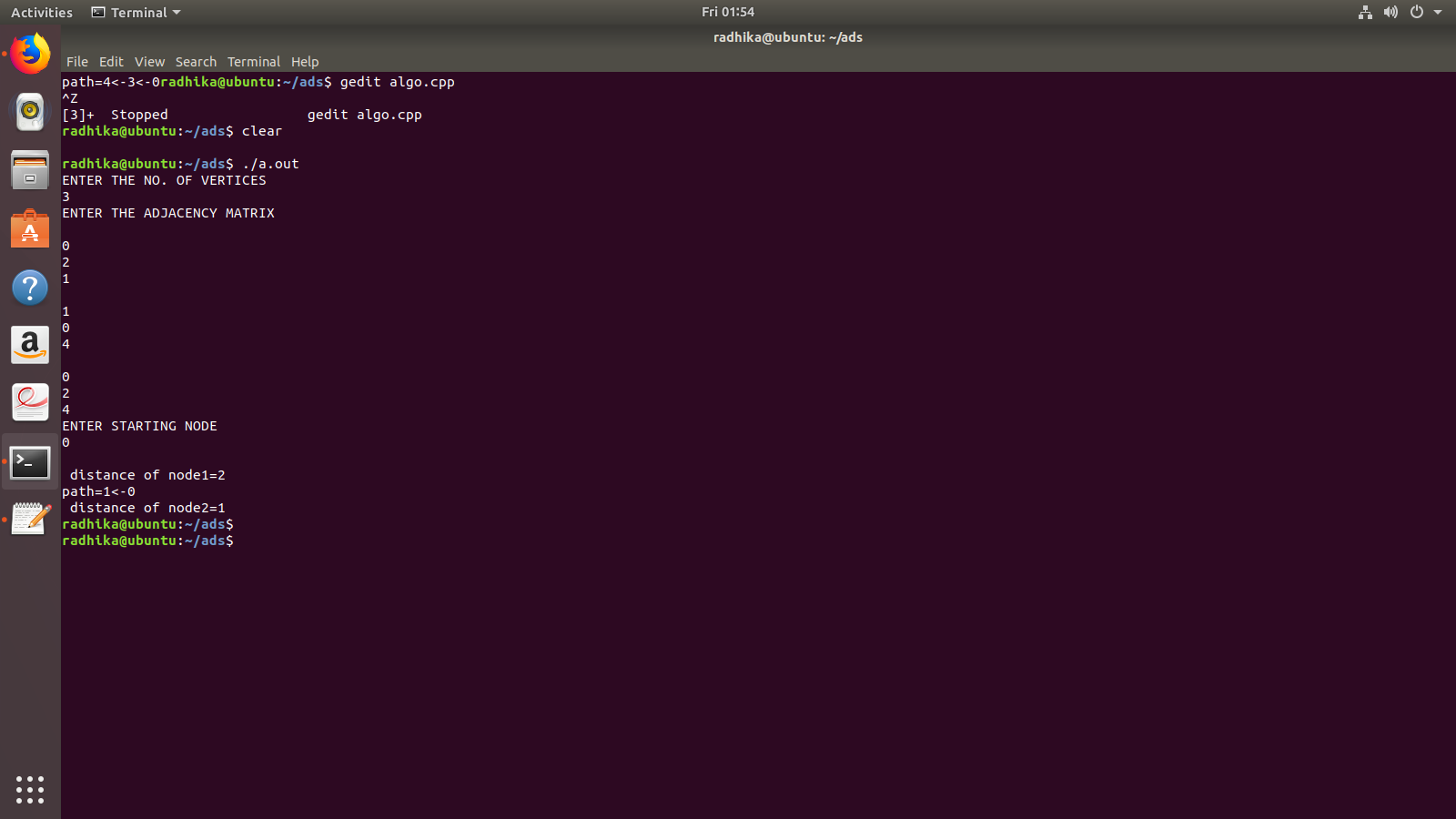
cin>>u;

g.dijikstra(u);

return 0;

}

**Output :**

****

**Conclusion:**

Understood the use of dijkstra algorithm in ﬁnding the shortest path in a graph representation of real life examples like(cities and roads connecting them) and minimum cost.