**American International University Bangladesh (AIUB)**

****

**Faculty of science & Technology**

**Department of Computer Science**

**LAB MANUAL Dijksta Algorithm**  
CSC 2211 Algorithms

|  |
| --- |
| **TITLE** |

**Dijksta Algorithm**

|  |  |
| --- | --- |
| **C:\Users\teacher\Downloads\graph (3).png** | E:\Summer 2021 2022\Algorithm\Algorithm F\Algo image\dijkstra.png |

Input for weighted graph

/\*

5 6

0 1 2

0 2 1

0 3 3

1 2 1

3 4 2

4 2 5

/\*

#include<bits/stdc++.h>

using namespace std;

#define N 100

struct node{

int val;

int cost;

};

vector<node> G[N];

int main(){

int nodes;

int edges;

cin>>nodes>>edges;

for(int i=0;i<edges;i++){

int u,v,w;

cin>>u>>v>>w;

G[u].push\_back({v,w});

}

for(int i=0;i<nodes;i++){

cout<<i<<" ";

for(int j=0;j<G[i].size();j++){

cout<<"{"<<G[i][j].val<<","<<G[i][j].cost<<"}"<<" ";

}

cout<<endl;

}

return 0;

}

/\*

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

1 {2,1}

2

3 {4,2}

4 {2,5}

\*/

/\*

5 6

0 1 2

0 2 1

0 3 3

1 2 1

3 4 2

4 2 5

\*/

#include<bits/stdc++.h>

using namespace std;

#define N 100

struct node

{

int val;

int cost;

};

vector<node> G[N];

bool visited[N];

int dis[N];

class cmp

{

public:

bool operator()(node &a, node &b)

{

return a.cost>b.cost?true:false;

}

};

void dijkstra(int source)

{

priority\_queue<node,vector<node>,cmp> PQ;

PQ.push({source,0});

while(!PQ.empty())

{

node current =PQ.top();

PQ.pop();

int v = current.val;

int c = current.cost;

if(visited[v]==1) continue;

dis[v] = c;

visited[v]=1;

for(int i=0; i<G[v].size(); i++)

{

int nxt = G[v][i].val;

int nxtcost = G[v][i].cost;

if(visited[nxt]==0)

{

PQ.push({nxt,c+nxtcost});

}

}

}

}

int main()

{

int nodes;

int edges;

cin>>nodes>>edges;

for(int i=0; i<edges; i++)

{

int u,v,w;

cin>>u>>v>>w;

G[u].push\_back({v,w});

}

int source;

cout<<"enter source: ";

cin>>source;

dijkstra(source);

for(int i=0; i<nodes; i++)

{

cout<<"Node "<<i<<" distance: "<<dis[i]<<endl;

}

return 0;

}

/\*

enter source: 0

Node 0 distance: 0

Node 1 distance: 2

Node 2 distance: 1

Node 3 distance: 3

Node 4 distance: 5

\*/