DSA LAB (Prims, Kruskal & Dijkstra) - 17/11/2021

Name: Vishal Teotia

Reg No.: 19BME1133

Link to source code:
https://gist.github.com/vashuteotia123/345a05cbfcd7a02644f83d37d76d77a2

Prims Code:

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

using namespace std;



void prims(int **edges, int n)

{

bool* visited = new bool[n];

int* parent = new int[n];

int* weight = new int[n];

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

{

weight[i] = INT_MAX; visited[i]=false;

}

parent[0] = -1;

weight[0] = 0;

for (int i = 0; i < n - 1; i++)

{

int min_vertex = -1;

for (int k = 0; k < n; k++)

{

if (!visited[k] && (min_vertex==-1||(weight[k]<weight[min_vertex])))

{
```

```
min_vertex = k;
}
}
visited[min_vertex] = true;
cout << "Current Node: "<< (char)(97+min_vertex) << endl ;
for (int j = 0; j < n; j++)
{


if (!visited[j] && edges[min_vertex][j] !=0)
{
cout << "Option : " << (char)(97+min_vertex) << " - " << (char)(97+j) << " -> " << edges[min_vertex][j];
if(weight[j] > edges[min_vertex][j]){
weight[j] = edges[min_vertex][j]; parent[j] = min_vertex;
cout << endl;
} else {
cout << " Rejected"<<endl;
}
}
}
cout << endl;
}
cout<<"\nMinimum spanning tree will be: \n"<<endl;
for (int i = 1; i < n; i++)
{
if (parent[i] < i)
{
cout << (char)(97 + parent[i]) << " - " << (char)(97 + i) <<" == "<< weight[i]<<endl;
}
```

```
else

{

cout << (char)(97 + i) << " - " << (char)(97 + parent[i]) << " == " << weight[i] << endl;

}

}

}



int main() {

cout <<
"\033[1;31m================================================================================
================\033[0m" << endl;

cout << "\033[1;31m| Topic: Prims's Alogrithm - DSA LAB WORK 17/11/2021 - Submitted by: Vishal Teotia
(19BME1133) |\033[0m" << endl;

cout <<
"\033[1;31m================================================================================
================\033[0m" << endl;

int n, e;

cout << "Enter number of nodes: " << endl;

cin >> n;

cout << "Enter number of edges: " << endl;

cin >> e;

int** edges = new int*[n];

cout << "Enter edges connection in format [node1 node2 weight]: " << endl;

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

edges[i] = new int[n];

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

edges[i][j] = 0;

}

}

for (int i = 0; i < e; i++)
```

```
{
int f, s, weight;
cin >> f >> s >> weight;
edges[f][s] = weight; edges[s][f] = weight;
}


// cout << "\nMatrix of given data: " << endl;
// for (int i = 0; i < n; i++) {
// for (int j = 0; j < n; j++) {
// cout << edges[i][j] << " ";
// }
// cout << endl;
// }
cout << endl;
prims(edges, n);


cout << "\nClearing dynamically allocated memory..." << endl;
for (int i = 0; i < n; i++) {
delete[] edges[i];
}


delete[]edges;
cout << "Exiting the program..." << endl;
return 0;
}
```

Output:

```
================================================================================
================
| Topic: Prims's Alogrithm - DSA LAB WORK 17/11/2021 - Submitted by: Vishal Teotia
(19BME1133) |
================================================================================
================
Enter number of nodes:
7
Enter number of edges:
12
Enter edges connection in format [node1 node2 weight]:
0 1 3
0 4 4
0 3 2
0 2 10
1 4 7
2 3 12
2 5 15
3 5 4
3 4 6
4 5 5
4 6 2
5 6 3

Current Node: a
Option : a - b -> 3
```

Option : a - c -> 10

Option : a - d -> 2

Option : a - e -> 4

Current Node: d

Option : d - c -> 12 Rejected

Option : d - e -> 6 Rejected

Option : d - f -> 4

Current Node: b

Option : b - e -> 7 Rejected

Current Node: e

Option : e - f -> 5 Rejected

Option : e - g -> 2

Current Node: g

Option : g - f -> 3

Current Node: f

Option : f - c -> 15 Rejected

Minimum spanning tree will be:

a - b == 3

a - c == 10

a - d == 2

a - e == 4

f - g == 3

e - g == 2

Clearing dynamically allocated memory...

Exiting the program...

Kruskal Code:

```
#include <bits/stdc++.h>
using namespace std;


class Edge{
public:
int source;
int dest;
int weight;
};
bool mysort(Edge x,Edge y){
return x.weight < y.weight;
}
int find_parent(int v1,int *parent){
int pv1=v1;
while(parent[pv1] !=pv1){
pv1=parent[pv1];
}
return pv1;
```

```
}

int main(){

cout <<
"\033[1;31m================================================================================
================\033[0m" << endl;

cout << "\033[1;31m| Topic: Kruskal Algorithm - DSA LAB WORK 17/11/2021 - Submitted by: Vishal Teotia
(19BME1133) |\033[0m" << endl;

cout <<
"\033[1;31m================================================================================
================\033[0m" << endl;

int n,e;

cout << "Enter no. of Nodes: " << endl;

cin >> n;

cout << "Enter no. of Edges: " << endl;

cin >> e;

cout << "Enter edges connection in format [node1 node2 weight]: " << endl;

Edge input[e], output[n - 1];

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

int source, dest, weight;

cin >> source >> dest >> weight;

input[i].source = source;

input[i].dest = dest;

input[i].weight = weight;

}

int parent[n];

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

parent[i]=i;

}

sort(input,input+e,mysort);

int count=0,i=0,v1,v2;

while(count<n-1){
```

```
v1=input[i].source;
v2=input[i].dest;
// cout << "Source: "<< v1 << "Dest: "<< v2 << endl;
int pv1=find_parent(v1,parent);
int pv2=find_parent(v2,parent);
if(pv1!=pv2){
parent[pv2]=pv1;
output[count].source=input[i].source;
output[count].dest=input[i].dest;
output[count].weight=input[i].weight;
count++;
}
i++;
}
cout << "\nCost optimal solution for the given graph: " << endl;
for (int i = 0; i < n - 1; i++) {
int minVertex = min(output[i].source, output[i].dest);
int maxVertex = max(output[i].source, output[i].dest);
cout << (char)(minVertex+97) << " - " << (char)(maxVertex+97) << " -> " << output[i].weight << endl;
}
return 0;
}
```

Output :

```
================================================================================
| Topic: Kruskal Algorithm - DSA LAB WORK 17/11/2021 - Submitted by: Vishal Teotia (19BME1133) |
================================================================================'
Enter no. of Nodes:
6
```

Enter no. of Edges:

9

Enter edges connection in format [node1 node2 weight]:

0 1 13

0 2 8

0 3 1

1 2 15

2 3 5

2 4 3

3 4 4

3 5 5

4 5 2

Cost optimal solution for the given graph:

a - d -> 1

e - f -> 2

c - e -> 3

d - e -> 4

a - b -> 13

Dijkstra Code:

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

using namespace std;

void dijkstra(int **edge,int n,bool *visited,int *dist){

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

int minDist=INT_MAX;
```

```
int vertexMinDist=-1;
for(int j=0;j<n;j++){
if(!visited[j] && minDist > dist[j]){
minDist=dist[j];
vertexMinDist=j;
}
}
visited[vertexMinDist]=1;
for(int j=0;j<n;j++){
if(!visited[j] && dist[j]>dist[vertexMinDist]+edge[vertexMinDist][j] && edge[vertexMinDist][j]>0){
dist[j]=dist[vertexMinDist]+edge[vertexMinDist][j];
}
}
}
}
int main(){
cout <<
"\033[1;31m================================================================================
================\033[0m" << endl;
cout << "\033[1;31m| Topic: Dijkstra Alogrithm - DSA LAB WORK 17/11/2021 - Submitted by: Vishal Teotia
(19BME1133) |\033[0m" << endl;
cout <<
"\033[1;31m================================================================================
================\033[0m" << endl;
int n,e;
cout << "Number of Nodes: " << endl;
cin >> n;
cout << "Number of Edges : " << endl;
cin >> e;
int **edge = new int *[n];
for(int i=0;i<n;i++){
```

```
edge[i]=new int[n];
for(int j=0;j<n;j++){
edge[i][j]=0;
}
}
cout << "Enter edges connection in format [node1 node2 weight]: " << endl;
for(int i=0;i<e;i++){
int f,s,w;
cin>>f>>s>>w;
edge[f][s]=w;
edge[s][f]=w;
}
bool *visited=new bool[n];
for(int i=0;i<n;i++){
visited[i]=0;
}
int *dist=new int[n];
for(int i=0;i<n;i++){
dist[i]=INT_MAX;
}
int source_node;


cout << "Enter the source Node: " << endl;
cin >> source_node;
dist[source_node] = 0;
dijkstra(edge,n,visited,dist);
cout << "Minimum Distance to every node form Source Node: " << endl;
for (int i = 0; i < n; i++) {
```

```
cout <<"From " << (char)(source_node+65) << " to "<< (char)(i+65) << " == " << dist[i] << endl;
}


cout <<"\nClearing the graph..."<<endl;
for(int i=0;i<n;i++){
delete [] edge[i];
}
delete [] edge;
delete [] visited;
delete [] dist;
cout <<"Exiting the program..."<<endl;
}
```

Output :

```
================================================================================
| Topic: Dijkstra Alogrithm - DSA LAB WORK 17/11/2021 - Submitted by: Vishal Teotia (19BME1133) |
================================================================================
Number of Nodes:
8
Number of Edges :
14
Enter edges connection in format [node1 node2 weight]:
0 1 2
0 3 7
0 6 2
0 5 12
1 6 5
```

1 3 4

1 2 1

1 4 3

2 6 4

2 4 4

3 4 1

3 7 5

4 7 7

5 7 3

Enter the source Node:

6

Minimum Distance to every node form Source Node:

From G to A == 2

From G to B == 4

From G to C == 4

From G to D == 8

From G to E == 7

From G to F == 14

From G to G == 0

From G to H == 13

Clearing the graph...

Exiting the program...