KRUSKAL’S ALGORITHM

#include <iostream>

#define SIZE 100

using namespace std;

class Graph

{

int vertices,edges;

int graph[SIZE][SIZE];

int mst[SIZE][SIZE];

int selected[SIZE][SIZE];

int parent[SIZE];

int cost;

public:

Graph(){}

Graph (int ,int);

void create();

void display();

int findparent(int v);

void kruskal();

void displaymst();

};

Graph::Graph(int v,int e)

{

vertices=v;

edges=e;

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

{

parent[i]=i;

for (int j=0; j<vertices; j++)

{

graph[i][j]=0;

selected[i][j]=0;

mst[i][j]=0;

}

}

}

void Graph::create()

{

int source,destination,weight;

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

{

cout<<"\nEnter the source vertex:-";

cin>>source;

cout <<"Enter the destination vertex:-";

cin>>destination;

if(source !=destination)

{

if(graph[source-1][destination-1]==0&& graph[source-1][destination-1]==0)

{

cout<<"Enter the weight of the graph:-";

cin>>weight;

graph[source-1][destination-1]=weight;

graph[destination-1][source-1]=weight;

cout<<"Inserted edge between"<<source<<"and"<<destination<<endl;

}

else

{

cout<<"\nEdge already exists.Please select a new edge"<<endl;

i--;

continue;

}

}

else

{

cout<<"\nSource and destination cannot be the same\n";

i--;

continue;

}

}

cout<<"\n\nGraph created successfully"<<endl;

}

void Graph::display()

{

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

{

for (int j=0; j<vertices; j++)

cout<<graph[i][j]<<" ";

cout<<endl;

}

}

int Graph::findparent(int v)

{

if(parent[v]==v)

return v;

return findparent(parent[v]);

}

void Graph::kruskal()

{

int min\_weight,min\_source,min\_destination;

int k=1;

cost=0;

while (k!=vertices)

{

min\_weight=100;

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

{

for(int j=0; j<vertices;j++)

{

if(graph[i][j]&& !selected[i][j] && graph[i][j] <=min\_weight)

{

min\_weight=graph[i][j];

min\_source=i;

min\_destination=j;

}

}

}

if(findparent(min\_source) !=findparent(min\_destination))

{

mst[min\_source][min\_destination]=min\_weight;

mst[min\_destination][min\_source]=min\_weight;

parent[min\_destination]=min\_source;

cost+=mst[min\_source][min\_destination];

selected[min\_source][min\_destination]=1;

selected[min\_destination][min\_source]=1;

k++;

}

}

}

void Graph::displaymst()

{

cout<<"\n\nThe minimum spanning tree is:-";

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

{

for (int j=0; j<vertices; j++)

cout<<mst[i][j]<<" ";

cout<<endl;

}

cout<<"\nThe cost of the MST is: "<<cost <<endl;

}

int main()

{

Graph g;

int choice, e, v;

while(1)

{

cout<<"\nImplementation of kruskal's algorithm"<<endl;

cout<<"1.Create graph"<<endl;

cout<<"2. Display graph"<<endl;

cout<<"3.Find MST using Prim's algorithm"<<endl;

cout<<"4.Exit the program"<<endl;

cout<<"\nEnter your choice:-";

cin>>choice;

switch(choice)

{

case 1:

cout<<"\nEnter the number of vertices:-";

cin>>v;

cout<<"\nEnter the number of edges:-";

cin>>e;

g=Graph(v,e);

g.create();

break;

case 2:

g.display();

break;

case 3:

g.kruskal();

g.displaymst();

break;

case 4:

return 0;

default:

cout<<"\nEnter in choice,try again"<<endl;

}

}

return 0;

}

OUTPUT:

Implementation of kruskal's algorithm

1.Create graph

2. Display graph

3.Find MST using Prim's algorithm

4.Exit the program

Enter your choice:-1

Enter the number of vertices:-3

Enter the number of edges:-3

Enter the source vertex:-1

Enter the destination vertex:-2

Enter the weight of the graph:-10

Inserted edge between1and2

Enter the source vertex:-2

Enter the destination vertex:-3

Enter the weight of the graph:-30

Inserted edge between2and3

Enter the source vertex:-3

Enter the destination vertex:-1

Enter the weight of the graph:-60

Inserted edge between3and1

Graph created successfully

Implementation of kruskal's algorithm

1.Create graph

2. Display graph

3.Find MST using Prim's algorithm

4.Exit the program

Enter your choice:-2

0 10 60

10 0 30

60 30 0

Implementation of kruskal's algorithm

1.Create graph

2. Display graph

3.Find MST using Prim's algorithm

4.Exit the program

Enter your choice:-3

The minimum spanning tree is:-0 10 0

10 0 30

0 30 0

The cost of the MST is: 40

Implementation of kruskal's algorithm

1.Create graph

2. Display graph

3.Find MST using Prim's algorithm

4.Exit the program

Enter your choice:-4

Process returned 0 (0x0) execution time : 41.291 s

Press any key to continue.