PRIM’S ALGORITHM

#include <iostream>

#include <iomanip>

#define SIZE 100

using namespace std;

class Graph

{

int vertices, edges, cost;

int graph[SIZE][SIZE];

int mst[SIZE][SIZE];

int distance[SIZE];

int visited[SIZE];

int parent[SIZE];

public:

Graph() {}

Graph(int, int);

void create();

void display();

void prims();

void displaymst();

};

Graph::Graph(int v, int e)

{

vertices = v;

edges = e;

cost = 0;

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

{

distance[i] = INT32\_MAX;

visited[i] = 0;

parent[i] = -1;

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

{

graph[i][j] = 0;

mst[i][j] = 0;

}

}

distance[0] = 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()

{

cout << "\n\nThe given tree is:- \n"

<< endl;

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

{

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

cout << setw(4) << graph[i][j];

cout << endl;

}

}

void Graph::prims()

{

int min;

for (int k = 0; k < vertices - 1; k++)

{

min = -1;

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

{

if (!visited[i] && (min == -1 || distance[i] < distance[min]))

min = i;

}

visited[min] = 1;

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

{

if (graph[min][i] != 0 && !visited[i] && graph[min][i] < distance[i])

{

distance[i] = graph[min][i];

parent[i] = min;

}

}

}

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

{

int j = parent[i];

if (j != -1)

{

mst[j][i] = distance[i];

mst[i][j] = distance[i];

cost += distance[i];

}

}

}

void Graph::displaymst()

{

cout << "\n\nThe minimal spanning tree is\n"

<< endl;

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

{

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

cout << setw(4) << 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 Prim'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.prims();

g.displaymst();

break;

case 4:

return 0;

default:

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

}

}

return 0;

}

OUTPUT:

Implementation of Prim'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 between 1 and 2

Enter the source vertex:- 2

Enter the destination vertex:- 3

Enter the weight of the graph:- 30

Inserted edge between 2 and 3

Enter the source vertex:- 3

Enter the destination vertex:- 1

Enter the weight of the graph:- 60

Inserted edge between 3 and 1

Graph created successfully

Implementation of Prim's algorithm

1. Create graph

2. Display graph

3. Find MST using Prim's algorithm

4. Exit the program

Enter your choice:- 2

The given tree is:-

0 10 60

10 0 30

60 30 0

Implementation of Prim's algorithm

1. Create graph

2. Display graph

3. Find MST using Prim's algorithm

4. Exit the program

Enter your choice:- 3

The minimal spanning tree is

0 10 0

10 0 30

0 30 0

The cost of the MST is 40

Implementation of Prim'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 : 42.055 s

Press any key to continue.