**Assignment No : 7**

Name : Rasika Mahure

PRN No : 72018258B

Div : B

Roll No : S1951093

**Aim :-**

Represent a graph of your college campus using adjacency list /adjacency matrix. Nodes should represent the various departments/institutes and links should represent the distance between them.

Find Minimum Spanning tree

1. Using Kruskal’s Algorithm
2. Using Prims Algorithm.

**//PROGRAM for Kruskal’s Algorithm**

#include <algorithm>

#include <iostream>

using namespace std;

class Edge

{

public:

    int source;

    int dest;

    int wt;

};

bool compare(Edge e1, Edge e2)

{

    return e1.wt < e2.wt;

}

int findParent(int v, int \*parent)

{

    if (parent[v] == v)

        return v;

    return findParent(parent[v], parent);

}

void kruskal(Edge \*input, int v, int e)

{

    //sorting the input array in increasing order based on weights

    //using inbuilt sort function

    sort(input, input + e, compare);

    Edge \*output = new Edge[v - 1];

    int parent[v];

    for (int i = 0; i < v; i++)

    {

        parent[i] = i;

    }

    int i = 0;

    int count = 0;

    while (count < v - 1)

    {

        int sourceParent, destParent;

        Edge currentEdge = input[i];

        sourceParent = findParent(currentEdge.source, parent);

        destParent = findParent(currentEdge.dest, parent);

        //checking if we can add currentnode in MST or not

        if (sourceParent != destParent)

        {

            output[count] = currentEdge;

            count++;

            parent[sourceParent] = destParent;

        }

        i++;

    }

    //displaying the output array

    cout << "\nMinimum spanning tree using kruskal algorithm:-";

    cout << "\nsource destination weight\n ";

    for (i = 0; i < v - 1; i++)

    {

        cout << "\n";

        cout << output[i].source << "\t" << output[i].dest << "\t" << output[i].wt;

    }

}

int main()

{

    int v, e;

    cout << "enter total number of vertices "

         << "\n";

    cin >> v;

    cout << "enter total number of edges "

         << "\n";

    cin >> e;

    Edge \*input = new Edge[e];

    cout << "enter source ,destination and the weights for the edges "

         << "\n";

    for (int i = 0; i < e; i++)

    {

        cin >> input[i].source >> input[i].dest >> input[i].wt;

    }

    kruskal(input, v, e);

}

**//OUTPUT**

enter total number of vertices

7

enter total number of edges

12

enter source ,destination and the weights for the edges

0 1 4

0 2 8

1 4 10

1 3 8

2 3 2

2 5 1

5 3 9

3 4 7

4 6 6

5 5 2

4 5 5

1 2 9

Minimum spanning tree using kruskal algorithm:-

source destination weight

2 5 1

2 3 2

0 1 4

4 5 5

4 6 6

0 2 8

**//Program for Prims Algorithm**

/\*primes\*/

#include<iostream>

using namespace std;

class graph

{

 int G[20][20],n;

 public:

 void accept()

 {

    int i,j,e;

    int src,dest,cost;

      cout<<"\nEnter the no. of vertices: ";

      cin>>n;

      for(i=0;i<n;i++)

      {

      for(j=0;j<n;j++)

      {

      G[i][j]=0;

      }

      }

      cout<<"\nEnter the no. of  Edges: ";

      cin>>e;

      for(i=0;i<e;i++)

      {

      cout<<"\nEnter Source: ";

       cin>>src;

      cout<<"\nDestination: ";

       cin>>dest;

      cout<<"\nCost: ";

      cin>>cost;

      G[src][dest]=cost;

      G[dest][src]=cost;

     }

  }

  void display()

  {

    int i,j;

   for(i=0;i<n;i++)

   {

    cout<<"\n";

    for(j=0;j<n;j++)

    {

        cout<<"\t"<<G[i][j];

    }

   }

  }

  void prims()

  {

    int i,j,R[20][20];

    int src,dest,cost,count,min;

    int total=0;

    int visited[20];

    for(i=0;i<n;i++)

    {

        for(j=0;j<n;j++)

        {

            if(G[i][j]==0)

            {

                R[i][j]=999;

            }

        else

        R[i][j]=G[i][j];

        }

    }

    for(i=0;i<n;i++)

    {

        visited[i]=0;

    }

    cout<<"\nEnter start vertex: ";

    cin>>src;

    visited[src]=1;

    count=0;

    while(count<n-1)

    {

    min=999;

    for(i=0;i<n;i++)

    {

    if(visited[i]==1)

    for(j=0;j<n;j++)

    {

    if(visited[j]!=1)

    {

    if(min>R[i][j])

    {

        min=R[i][j];

        src=i;

        dest=j;

    }

    }

    }

    }

    cout<<"\nEdge from "<<src<<" to "<<dest<<" \twith cost:  "<<min;

    total=total+min;

    visited[dest]=1;

    count++;

    }

        cout<<"\nTotal Cost: "<<total<<"\n";

    }

};

  int main()

  {

  graph g;

  g.accept();

  g.display();

  g.prims();

  }

**//OUTPUT**

Enter the no. of vertices: 7

Enter the no. of Edges: 12

Enter Source: 0

Destination: 1

Cost: 4

Enter Source: 0

Destination: 2

Cost: 8

Enter Source: 1

Destination: 4

Cost: 10

Enter Source: 1

Destination: 3

Cost: 8

Enter Source: 1

Destination: 2

Cost: 9

Enter Source: 2

Destination: 3

Cost: 2

Enter Source: 2

Destination: 5

Cost: 1

Enter Source: 4

Destination: 3

Cost: 7

Enter Source: 4

Destination: 5

Cost: 5

Enter Source: 4

Destination: 6

Cost: 6

Enter Source: 5

Destination: 6

Cost: 2

Enter Source: 5

Destination: 3

Cost: 9

0 4 8 0 0 0 0

4 0 9 8 10 0 0

8 9 0 2 0 1 0

0 8 2 0 7 9 0

0 10 0 7 0 5 6

0 0 1 9 5 0 2

0 0 0 0 6 2 0

Enter start vertex: 0

Edge from 0 to 1 with cost: 4

Edge from 0 to 2 with cost: 8

Edge from 2 to 5 with cost: 1

Edge from 2 to 3 with cost: 2

Edge from 5 to 6 with cost: 2

Edge from 5 to 4 with cost: 5

Total Cost: 22