Rajshahi University of Engineering & Technology

CSE 2202: Sessional Based on CSE 2201

Lab Report 06

Date: January 14, 2019

Submitted to

Biprodip Pal

Instructor, CSE 2201 & CSE 2202

Assistant Professor, Dept. of CSE

Submitted by

Fuad Al Abir

Roll: 1603021

Section: A

Dept. of CSE

Sessional – Cycle 6 – Problem A

Create an undirected graph randomly. Find the minimum spanning tree of the graph

Code:

```
/*-----------------------------
    I N T R O D U C T I O N
-----------------------------
Author:     Fuad Al Abir
Date:       January 13, 2019
Name:       prim.cpp
Objective:  Finding the minimum spanning tree of a graph created randomly using
prim's algorithm.
*/

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

#define INF 0x3f3f3f3f

typedef pair <int, int> iPair;

class Graph
{
       int V;
       list< pair <int, int> > *adj;
public:
       Graph(int V)
       {
        this -> V = V;
        adj = new list <iPair> [V];
    }
       void addEdge(int u, int v, int w)
       {
        adj[u].push_back(make_pair(v, w));
        adj[v].push_back(make_pair(u, w));
    }
       void primMST()
       {
        priority_queue < iPair, vector <iPair> , greater <iPair> > pq;
        int src = 0;
        vector <int> key(V, INF);
        vector <int> parent(V, -1);
        vector <bool> inMST(V, false);
        pq.push(make_pair(0, src));
        key[src] = 0;
        while (!pq.empty())
        {
            int u = pq.top().second;
            pq.pop();
            inMST[u] = true;
            list < pair <int, int> >::iterator i;
            for (i = adj[u].begin(); i != adj[u].end(); ++i)
            {
                int v = (*i).first;
                int weight = (*i).second;
                if (inMST[v] == false && key[v] > weight)
                {
                    key[v] = weight;
                    pq.push(make_pair(key[v], v));
                    parent[v] = u;
                }
            }
        }
        cout << "Minimum spanning tree using Prim's Algorithm having the edges:" <<
endl;
```

```
        for (int i = 1; i < V; ++i)
            cout << parent[i] << " - " << i << endl;
    }
};

int main()
{
    srand(time(0));
    int V;
    cout << "Number of node(s): ";
    cin >> V;
    Graph g(V);
    cout << "\nLet the nodes are: ";
    for (int i = 0; i < V; i++)
        cout << i << " ";
    cout << "\n";
    int temp;
    for (int i = 0; i < V; i++)
    {
        for(int j = i + 1; j < V; j++)
        {
            temp = rand() % 20;
            g.addEdge (i, j, temp);
            cout << i << " -> "<< j << " = " << temp << endl;
        }
    }
    cout << "\n";
       g.primMST();
       return 0;
}
```

Input/Output:

```
Number of node(s): 4
Let the nodes are: 0 1 2 3
0 -> 1 = 7
0 -> 2 = 11
0 -> 3 = 19
1 -> 2 = 11
1 -> 3 = 16
2 -> 3 = 13
Minimum spanning tree using Prim's Algorithm having the edges:
0 - 1
0 - 2
2 - 3
```

```
Number of node(s): 5
Let the nodes are: 0 1 2 3 4
0 -> 1 = 7
0 -> 2 = 5
0 -> 3 = 12
0 -> 4 = 11
1 -> 2 = 8
1 -> 3 = 5
1 -> 4 = 17
2 -> 3 = 3
2 -> 4 = 18
3 -> 4 = 4
Minimum spanning tree using Prim's Algorithm having the edges:
3 - 1
0 - 2
2 - 3
3 - 4
```