Rajshahi University of Engineering & Technology

CSE 2202: Sessional Based on CSE 2201

Lab Report 07

Date: January 21, 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

Implement Kruskal’s algorithm for finding Minimum spanning tree from a weighted undirected graph. Use disjoint set to implement union and set finding operations. These operations should use the concept of parent child relationship to represent elements of a set.

Code:

```
/*-----------------------------
    I N T R O D U C T I O N
-------------------------------
Author:     Fuad Al Abir
Date:       January 20, 2019
Name:       kruskal.cpp
Objective:  Finding the minimum spanning tree of a graph created randomly using
Kruskal’s algorithm.
*/

#include <iostream>
#include <algorithm>

using namespace std;

/*
1. Create Edge class
2. Input graph
*** KRUSKAL'S ALGO STARTS HERE ***
3. Sort edges with respect to weight (compare is for this)
4. Create output array of type Edge, size n - 1
5. Create parent array and initialize them with themselves
6. count, i
7. Check if we can add currentEdge in MST or not (Check by finding the parent of
the edges)
8. if parents are not equal, add edge to MST and update parent
*/

class Edge
{
    public:
        int source;
        int dest;
        int weight;
};

bool compare(Edge e1, Edge e2)
{
    return e1.weight < e2.weight;
}

int findParent(int v, int *parent)
{
    if(parent[v] == v)
    {
        return v;
    }
    return findParent(parent[v], parent);
}

void printMST(Edge *output, int n)
{
    cout << "\nMinimum Spanning Tree using Kruskals Algorithm\n";
    for(int i = 0; i < n - 1; i++)
    {
        if(output[i].source < output[i].dest)
        {
            cout << output[i].source << " ";
            cout << output[i].dest << " ";
            cout << output[i].weight;
            cout << endl;
        }
        else
        {
            cout << output[i].dest << " ";
            cout << output[i].source << " ";
            cout << output[i].weight;
            cout << endl;
        }
    }
}

void kruskals(Edge *input, int n, int E)
{
    sort(input, input + E, compare);

    Edge *output = new Edge[n - 1];

    int *parent = new int[n];
    for(int i = 0; i < n; i++)
    {
        parent[i] = i;
    }

    int count = 0;
    int i = 0;
    while(count != n - 1)
    {
        Edge currentEdge = input[i];
        int sourceParent = findParent(currentEdge.source, parent);
        int destParent = findParent(currentEdge.dest, parent);

        if(sourceParent != destParent)
        {
            output[count] = currentEdge;
            count++;
            parent[sourceParent] = destParent;
        }
        i++;
    }
    printMST(output, n);
}

int main()
{
    int n, E;
    cout << "Enter the number of Nodes and Edges: ";
    cin >> n >> E;
    // Creating input array of size E, type Edge
    Edge *input = new Edge[E];

    for(int i = 0; i < E; i++)
    {
        int s, d, w;
        cin >> s >> d >> w;
        input[i].source = s;
        input[i].dest = d;
        input[i].weight = w;
    }
    kruskals(input, n, E);
}
```

Input/Output:

```
Enter the number of Nodes and Edges: 6 11
1 3 1
0 1 2
0 3 3
0 2 4
4 5 5
2 3 6
3 5 7
2 1 8
2 4 9
2 5 10
3 4 11

Minimum Spanning Tree using Kruskals Algorithm
1 3 1
0 1 2
0 2 4
4 5 5
3 5 7
```

```
Enter the number of Nodes and Edges: 5 4
0 1 1
1 2 3
2 3 4
3 4 1

Minimum Spanning Tree using Kruskals Algorithm
0 1 1
3 4 1
1 2 3
2 3 4
```