

# **Data Structures and Algorithms (CS09203)**

## **Lab Report**

Name: Rehman ullah baig  
Registration #: CSU-F16118  
Lab Report #: 11  
Dated: 2-07-2018  
Submitted To: Mr. Usman Ahmed

The University of Lahore, Islamabad Campus  
Department of Computer Science & Information Technology

# Experiment # 11

## The Kruskal's algorithm

### Objective

To understand and implement The Kruskal's algorithm.

### Software Tool

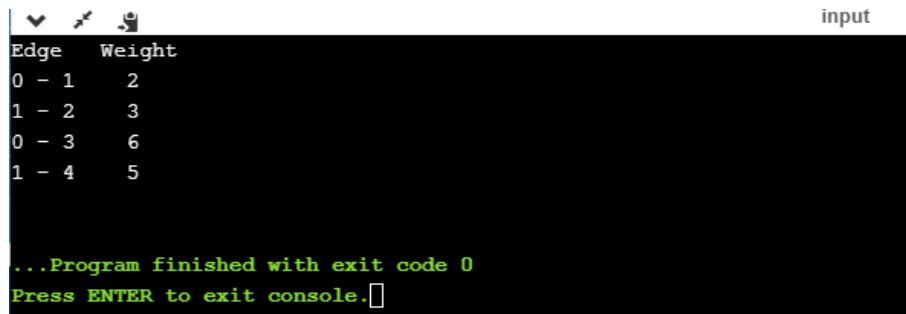
1. Windows 7
2. Dev C++
3. Miktex

## 1 Theory

Kruskal's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which form a tree that includes every vertex has the minimum sum of weights among all the trees that can be formed from the graph. It falls under a class of algorithms called greedy algorithms which find the local optimum in the hopes of finding a global optimum. We start from the edges with the lowest weight and keep adding edges until we reach our goal.

The steps for implementing Kruskal's algorithm are as follows:

1. Sort all the edges from low weight to high.
2. Take the edge with the lowest weight and add it to the spanning tree. If adding the edge created a cycle, then reject this edge.
3. Keep adding edges until we reach all vertices.



```
input
Edge  Weight
0 - 1   2
1 - 2   3
0 - 3   6
1 - 4   5

...Program finished with exit code 0
Press ENTER to exit console.
```

Figure 1: Data entering into different locations

## 2 Task

### 2.1 Procedure: Task 1

Kruskal's algorithm is a minimum-spanning-tree algorithm which finds an edge of the least possible weight that connects any two trees in the forest.[1] It is a greedy algorithm in graph theory as it finds a minimum spanning tree for a connected weighted graph adding increasing cost arcs at each step. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. If the graph is not connected, then it finds a minimum spanning forest (a minimum spanning tree for each connected component).

### 2.2 Procedure: Task 2

```
#include <iostream>
#include <stdio.h>
#include <limits.h>
#define V 5
using namespace std;
```

```

int minKey(int key[], bool mstSet[])
{
    int min = INT_MAX, min_index;

    for (int v = 0; v < V; v++)
        if (mstSet[v] == false && key[v] < min)
            min = key[v], min_index = v;

    return min_index;
}

int printMST(int parent[], int n, int graph[V][V])
{
    printf("Edge\t\tWeight\n");
    for (int i = 1; i < V; i++)
        printf("%d\t-%d\t\t%d\n", parent[i], i, graph[i][parent[i]]);
}

void primMST(int graph[V][V])
{
    int parent[V];
    int key[V];
    bool mstSet[V];

    for (int i = 0; i < V; i++)
        key[i] = INT_MAX, mstSet[i] = false;

    key[0] = 0;
    parent[0] = -1;

    for (int count = 0; count < V-1; count++)
    {
        int u = minKey(key, mstSet);

        mstSet[u] = true;
    }
}

```

```

        for (int v = 0; v < V; v++)

            if (graph[u][v] && mstSet[v] == false && graph[u][v] <
key[v])
                parent[v] = u, key[v] = graph[u][v];
        }

        printMST(parent, V, graph);
    }

    int main()
    {

        int graph[V][V] = {{0, 2, 0, 6, 0},
                            {2, 0, 3, 8, 5},
                            {0, 3, 0, 0, 7},
                            {6, 8, 0, 0, 9},
                            {0, 5, 7, 9, 0},
                            };

        primMST(graph);

        return 0;
    }

```

### 3 Conclusion

weve covered the Intuitive idea behind the Kruskal algorithm and his computation complexity. Kruskal's algorithm is used in real world for map configuration. Kruskals algorithm is an edge based algorithm. Prims algorithm with a heap is faster than Kruskals algorithm. Efficiency of Kruskals algorithm is based on the time needed for sorting the edge weights of a given graph