**Data Structures and Algorithms Design**

(S1-21\_DSECLZG519)

FIRST SEMESTER 2021-22



Assignment No.: 2

**PS2 - Telecom Network**

**SUBMITTED BY**

Varinder Singh - 2021fc04070

Bandaru Raja Sekhar - 2021fc04074

Baskar K - 2021fc04066

**Table of Contents**

[1. Objective](#_heading=h.bacw9fer07q8) **3**

[2. Data Structure](#_heading=h.smmi906b8one) **3**

[2.1 Terminologies used in Graph](#_heading=h.6f869dxh5jwy) 3

[2.2 Justification for data structure](#_heading=h.pgt4zzy662z2) 3

[2.3 Edge Structure](#_heading=h.yfxjqnpwbhva) 4

[3. Kruskal’s Algorithm](#_heading=h.umc1xnwn188p) **4**

[4. Time-Space Complexity of the Operation](#_heading=h.g49ij19r4ov7) **4**

[5. An alternative way to implement the problem](#_heading=h.ka35rbkegbrs) **5**

# **1. Objective**

A client has several offices all across India. The objective is to connect all the client’s offices with each other by leasing phone lines from a network company at a minimal cost. The network company charges different amounts of money to connect different cities. We need to find the best possible way to set up lines that will connect all the offices with minimum total cost.

# **2. Data Structure**

## The problem can be solved by using the data structure of Graph. A Graph consists of a finite set of vertices(or nodes) and a set of edges that connect a pair of nodes. It is a non-linear data structure. The network of offices of the client can be seen as a graph where its nodes(offices) are spread across all of a country. A node of a graph can be denoted as an office or city location and the edge connecting the two nodes can indicate the cost of laying a phone line within cities by its carrying weightage.

## **2.1 Terminologies used in Graph**

* Node – A vertex, sometimes called a node, is a point or circle. It is the fundamental unit from which graphs are made
* Edge – An edge is a line joining a pair of nodes.
* Cycle – A cycle is a closed walk with no repeated vertices (except that the first and last vertices are the same).

## **2.2 Justification for data structure**

The problem is to find the minimum cost to set up lines that will connect all the offices of the client. This can be achieved by finding the minimum spanning tree of the graph. A minimum spanning tree (MST) or minimum weight spanning tree for a weighted, connected, undirected graph is a spanning tree whose sum of weights is minimum. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree. In order to achieve this, we need to create a network of vertices and edges which are in the graph data structure.

In this problem, we are focused on using weighted edges of the graph so that we can get the minimum spanning tree which in return provides the minimum cost required to set up telephone lines between all the offices or city locations.

## **2.3 Edge Structure**

**class** Edge:

**def** \_\_init\_\_(self, source, dest, weight):

self.source = source

self.dest = dest

self.weight = weight

# **3. Kruskal’s Algorithm**

To solve the problem to find minimum cost, we used Kruskal Algorithm to find Minimum Spanning Tree. Kruskal's algorithm uses the greedy approach. This algorithm treats the graph as a forest and every node it is considered as an individual tree. A tree connects to another one and only if it has the least cost among all available options and does not violate Minimum Spanning Tree(MST) properties which mean that there is a greedy choice is to pick the smallest weight edge that does not cause a cycle in the MST constructed so far. Below are the steps for finding MST using Kruskal’s algorithm:

1. Sort all the edges in non-decreasing order of their weight.
2. Pick the smallest edge. Check if it forms a cycle with the spanning-tree formed so far. If a cycle is not formed, include this edge. Else, discard it.
3. Repeat step 2 until there are (V-1) edges in the spanning tree.

# **4. Time-Space Complexity of the Operation**

As in the Kruskal algorithm, we need to sort all the edges in non-decreasing order beforehand. It will have a time complexity of O(E logE) for sorting alone. After this, when we are checking whether the new edge is creating a cycle or not, then it will have a time complexity of O(E logV). Therefore, the overall complexity of the algorithm will be O(E logE) + O(E logV). But the value of E can be utmost O(V2), both O(E logE) and O(E logV) are the same.

The overall average runtime complexity of the Kruskal algorithm is Θ(E logE). The worst time complexity will also be O(E logE). The total space complexity turns out to be O(E+V).

\*\* where E is the number of edges of the graph

and V is the number of vertices of the graph

# 

# **5. An alternative way to implement the problem**

In the Kruskal algorithm, when we are sorting the edges we can use min-heap to sort. The edges can be processed in order of their weights by using a min-heap. This is generally faster than sorting the edges first because in practice we need only to visit a small fraction of the edges before completing the MST(Minimum Spanning Tree).

Another way of implementing the problem to find a minimum spanning tree would be to use another algorithm called Prim’s algorithm. It can start to build the Minimum Spanning Tree from any vertex in the graph which can provide flexibility to start MST whereas in the Kruskal algorithm it will start only from the shortest weighted edge to create MST. Prim’s algorithm runs faster in dense graphs which is good when we are dealing with dense graphs with many more edges than vertices.