Generate all minimum spanning trees in a given undirected weighted graph

Pradeep Gangwar (IHM2016501)[1] Prakhar Chaturvedi (IHM2016001)[2] Praphull Mishra (IWM2016005)[3]

***Abstract*— The paper details the algorithm to detect all minimum spanning trees present in an un-directed weighted graph. Given a all the edges and vertex of the graph, the proposed algorithm will output all the edges of minimum spanning trees of graph.**

I. INTRODUCTION

Graphs are mathematical structures that represent pairwise relationship between objects. They perform important role in modeling data. They are useful in solving many real life applications such as social networks, transportation networks, document-link graphs and many more. Let us introduce the important notations and definitions that we will use frequently in this paper.

A. Basic Definitions

1) *Graph:* Graph is a pair G=(V,E) where,
   - V is a set of vertices (or nodes), and
   - E $\subseteq$ (VxV) is a set of edges.

   A graph may be directed or undirected, if graph is undirected, then adjacency relation defined by edges is symmetric.

2) *Spanning trees:* A spanning tree is a subset of Graph G, which has all the vertices covered with minimum possible number of edges. Hence, a spanning tree does not have cycles and it cannot be disconnected.

3) *Minimum spanning trees:* In a weighted graph, a minimum spanning tree is a spanning tree that has minimum weight than all other spanning trees of the same graph. In real-world situations, this weight can be measured as distance, congestion, traffic load or any arbitrary value denoted to the edges.

B. Objective

Here we will propose algorithms to extract all minimum spanning trees present the the graph given. We will make use of Kruskal algorithm to generate a spanning tree and then we will recursively find all other minimum spanning trees.

II. MOTIVATION

Graph are very important data structures. They are used in network related algorithms, routing, finding relation among objects, shortest paths and many more real-life applications. On E-Commerce websites graphs are used to show recommendations.Connecting with friends on social media, where each user is a vertex and the connection between two users is represented by an edge, is also an important application of graphs.

Spanning tree is basically used to find a minimum path to connect all nodes in a graph. Common application of spanning trees are: Civil Network Planning, Computer Network Routing Protocol and Cluster Analysis. Consider, city network as a huge graph and now plans to deploy telephone lines in such a way that in minimum lines we can connect to all city nodes. This is where the spanning tree comes into picture.

III. ALGORITHM DESIGN

This part of the report can be further divided into following sections.

A. Input Format

The user inputs the total number of vertices and the edges. We input all the edges with their corresponding weights and then apply the algorithm.

B. Design of algorithm

Our algorithm can be divided into three parts. In first we will find an MST using kruskal algorithm and then we will use cut-set algorithm to cut the MST at an edge and then use ALL_MST function to find another edge that can replace that edge.

1) Kruskal: Kruskal algorithm is used to find a minimum spanning tree of a graph. Since Kruskal algorithm can only find a single MST we will extend this algorithm to find a MST of graph and use that MST to generate other MST recursively.

2) Cut-set function: Cut-set function removes an edge from the minimum spanning tree and divides the tree into two connected sets, say V1 and V2. After this the two sets are passed to the All-MST function with the weight of the edge removed.

3) All-MST function: All-MST function searches for the edge of weight that was removed by cut-set function, such that the edge creates a relationship between the two disconnected set,i.e., does not form a cycle with with introduction of that edge. This will create a new MST and recursively call the Cut-Set function for that MST, this in turn will create all the MSTs.

C. Algorithm

Algorithm 1 $ALL\_MST(F, R, T)$

1: **for** *each* i *in* $\{k+1, K+2, ..., n-1\}$ **do**
2: $\quad$ *Find the cut* $-$ *set* $Cut(e_i)$
3: $\quad$ *Find, if one exists, a substitute* $e^{i'} \in S(e^i)$
4: **for** *each* i *in* $\{k+1, K+2, ..., n-1\}$ **do**
5: $\quad$ **if** $e^{i'}$ *exists* **then**
6: $\quad\quad$ *Set* $T_i := T \cup e^{i'} \setminus e^i$ *and output* T_i ▷ New tree found
7: $\quad\quad$ *Set* $F_i := F \cup \{e^{k+1}, ..., e^{i-1}\}$ *and* $R_i := R \cup \{e^i\}$
8: $\quad\quad$ *Call* $All\_MST(F, R, T)$ *recursively*

Algorithm 2 $KRUSKAL(G)$

1: $T = \emptyset$
2: **for** *each vertex* $v \in G.V$ **do**
3: $\quad$ $Make - Set(v)$
4: **for** *each edge* $(u, v) \in G.E$ **do**
5: $\quad$ **if** $Find - Set(u) \neq Find - Set(v)$ **then**
6: $\quad\quad$ $T = T \cup \{(u, v)\}$
7: $\quad\quad$ $Union(u, v)$
8: Return T

D. Output Format

The output consists of all the MST that are possible.

IV. ANALYSIS

A. Analysis: All_MST algorithm

The function $All\_MST(F, R, T)$ traverses through the all the cut sets and finds a substitute edge to replace the removed edge. The overall complexity of this algorithm is $O(N * n * m)$. We can now understand the best and worst case complexities from this.

1) Best Case Analysis - (N = 1): When we have only one MST then the complexity $O(N * n * m)$ turns out to be $O(n * m)$

$time_{best(N=1)} \propto N * n * m$ instructions.
$=> time_{best(N=1)} \propto n * m$

$=> O(n * m)$ - A Quadratic Time Computation

2) Worst Case Analysis : The worst case appears when all spanning trees are minimum spanning trees. Hence we can not neglect the valu of N. Hence the complexity appears as $O(N * n * m)$

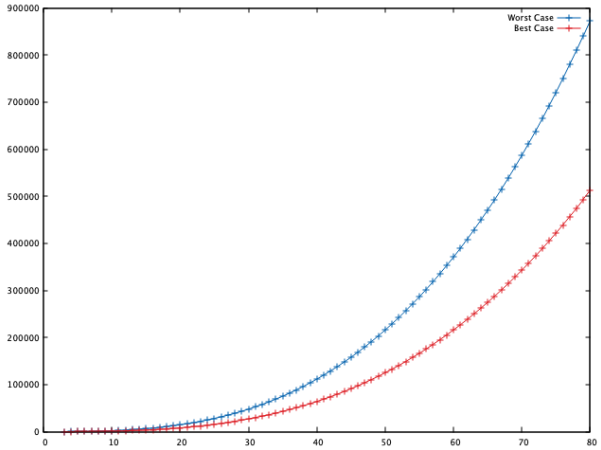
$time_{worst} \propto N * n * m$ instructions.
$=> time_{worst} \propto N * n * m$

$=> O(N * n * m)$ - A Quadratic Time Computation

3) Average Case Analysis : We know that

$$time_{best} <= t_{average} <= t_{worst}$$

So, We can say that
$=> t_{avg} = t_{worst}$



V. DISCUSSION

A. Understanding the minimum spanning tree

A minimum spanning tree (MST) or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted un-directed graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight. That is, it is a spanning tree whose sum of edge weights is as small as possible. More generally, any edge-weighted un-directed graph (not necessarily connected) has a minimum spanning forest, which is a union of the minimum spanning trees for its connected components.

B. About extraction of All Minimum spanning trees

We first extract the minimum spanning tree using Kruskal minimum spanning tree algorithm . Then we remove an edge from that MST and that in turn creates two disconnected sets. Now we try to find an edge of the same weight as of the removed edge such that it creates a relationship between two sets without forming the cycles. As we find that edge we incorporate that edge in between the two sets and create another minimum spanning tree. And now calling the above algorithm again for this formed minimum spanning tree and thus we can find all MST possible in a graph.

VI. CONCLUSION

We have developed an algorithm to list all the minimum spanning trees in an undirected weighted graph, and explored some properties of cut-sets. We have constructed an algorithm, which runs in O(Nmn) time and O(m) space.

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] Introduction to Algorithms English By Thomas H. Cormen , By Charles E. Leiserson , Ronald L. Rivest ,Clifford Stein(3rd edition)