1. Problem Statement: We are required to analyze the below problem statement:
Input: Given a weighted connected graph G = (V, E), where w[i,j] or w(e) is the weight of edge between vertices i and j.
Objective: Find a minimum-weight spanning tree of G

2. Theoretical Analysis

A **minimum spanning tree** (MST) of a graph is a selection of edges that connects all the vertices with the smallest possible total edge weight. **Kruskal's algorithm** is one of the most efficient ways to find an MST because it uses a **greedy approach**—picking edges in the graph based on their smallest weight while avoiding cycles. Here's a breakdown of how the algorithm works:

• **Algorithm**: KruskalsMST (G, W[1:n,1:n], T)
Sort m edges in the ascending order of their weights: e[1], e[2], .. e[m].
Initialize a counter k = 1
Initialize an empty tree T
While (number of edges in Tree < n-1) {
    If adding the edge e[j] does not create a cycle, add edge e[j] to tree T
    k++
}

Now, let's think about a graph **G** with **V** vertices and **E** edges. Kruskal's algorithm uses the graph's edges and their weights to find the MST. The steps of the algorithm involve a few key operations that can be managed efficiently using a **disjoint-set** or **union-find** data structure:
a) **Sorting the edges by weight**: This takes **O(E log V)** time. Sorting ensures that we are always considering the smallest edge first.
b) **Checking for cycles**: Every time we consider adding an edge, we need to check whether it would create a cycle. This is done with a Find(x) operation, which tells us whether two vertices are already connected. This step takes **O(E log V)** time overall, since it is performed for each edge.
c) **Updating connected vertices**: After adding an edge, we need to update the disjoint sets to reflect that two vertices are now connected. This is handled by the Union(x, y) operation, which merges the sets containing the vertices. This takes **O(V log V)** time, and we do it for V-1 edges.
Time Complexity:
The total time complexity of Kruskal's algorithm comes out to **O(E log V + V log V)**. However, since the number of edges E is generally larger than the number of vertices V, this simplifies to **O(E log V)**. Because Kruskal's algorithm consistently selects the smallest available edge that doesn't form a cycle, it ensures that the minimum total weight is achieved—making it an excellent example of a **greedy algorithm** for finding the MST.
Example Test Case:
Let's walk through a small example. Imagine we have a graph G with 4 vertices and 6 edges, where the edges are labeled with their weights as follows:

- (0, 1, 2)

- (0, 3, 7)
- (0, 2, 5)
- (1, 3, 3)
- (1, 2, 1)
- (2, 3, 4)

Kruskal's algorithm begins by sorting these edges by their weights. The sorted list of edges will be:
(1, 2, 1), (0, 1, 2), (1, 3, 3), (2, 3, 4), (0, 2, 5), and (0, 3, 7).
The algorithm now selects the edges with the smallest weights that don't form a cycle. So, it picks the edges (1, 2, 1), (0, 1, 2), and (1, 3, 3), which have weights 1, 2, and 3, respectively. After adding these edges to the tree and confirming no cycles are formed, we have our MST.

3. Experimental Analysis

3.1 Program Listing

• GitHub Link for the code: https://github.com/ichandan2151/daa_project2_7

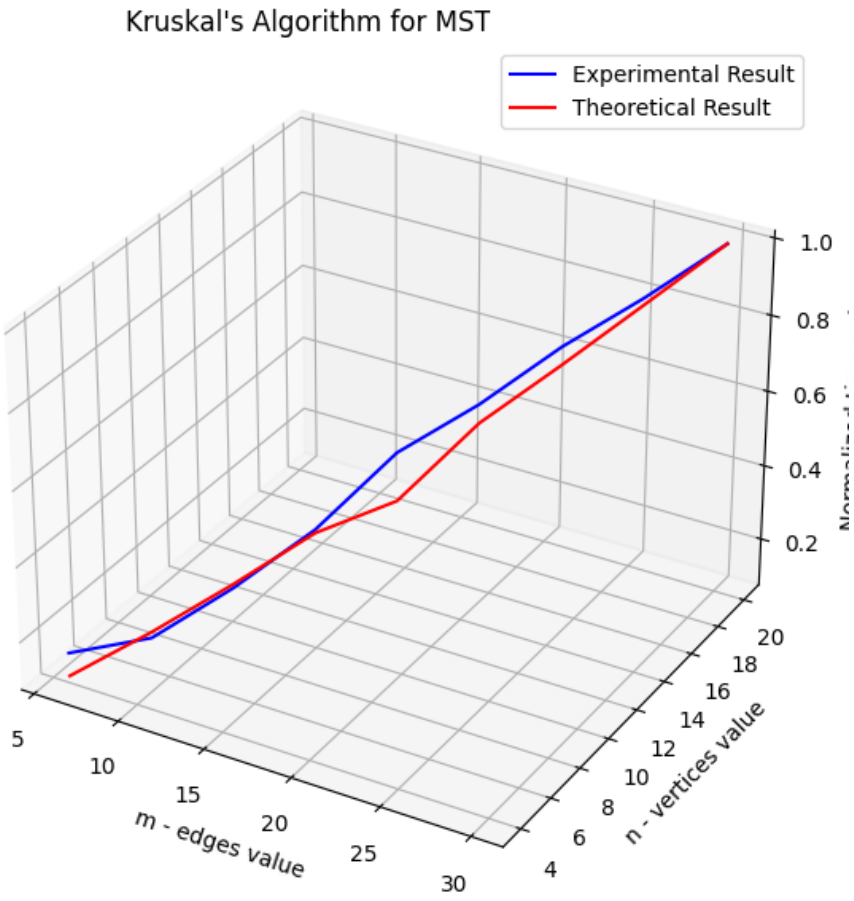
3.2 Data Normalization Notes

Average of Experimental results is: 3034814.41
Average of Theoretical result is: 66.28
Scaling constant = Average of Experimental result / Average of Theoretical result
Therefore, Scaling Constant for normalizing the theoretical results = 32654.726

3.3 Output Numerical Data

Vertices value (v)	Edge Value (e)	Experimental Result in ns	Theoretical Result (e log v)	Scaling Constant	Scaled Theoretical Result
4	6	863384.0	12.00		391856.712
6	9	915359.2	23.27		759875.474
8	12	1494497.0	36.00		1175570.14
10	15	2196097.7	49.83		1627185
12	18	3173597.7	57.06		1863278.67
14	21	3721434.9	79.96		2611071.89
16	24	4402710.8	96.00		3134853.7
18	27	4959169.9	112.78		3682800
20	30	5587078.5	129.66		4234011.77
		3034814.41	66.28	32654.726	

3.4 Graph



z - axis is Normalized time values
x - axis is m - edges value
y - axis is n - vertices value

3.5 Graph Observations

The experimental results closely follow the theoretical trend, both increasing steadily as the number of edges and vertices grows. The rise in time complexity is mainly driven by the increase in edges, and the effect of this growth is more noticeable than the log factor related to the number of vertices.

4. Conclusions

The experimental and theoretical line plots intersect and appear to converge, supporting the conclusion that the time complexity of finding the minimum spanning tree using Kruskal's greedy algorithm is **O(E log V)**, where **E** is the number of edges and **V** is the number of vertices in the graph.