PROJECT-2

1 Problem Statement

Kruskal's Algorithm for Minimum Spanning Tree

2 Theoretical Analysis

- Sorting all the edges requires O(E * logE) time
- Once sorted, we process each edge, using the find-union algorithm. Both the find and union operations take up to O(logV) time per edge. Thus, for all edges, this results in O(ElogV) time.
- The total time complexity is O(E * logE + E * logV), but since E can be as large as O(V^2), logV and logE are asymptotically equivalent.
- **Therefore, the overall time complexity simplifies to O(E * logV).**
- Additionally, the Kruskal's algorithm is particularly efficient for sparse graphs, where the number of edges E is much smaller than V^2, making the O(E * logV) complexity favorable.

3 Experimental Analysis

3.1 Program Listing

```
def KruskalMST(self):
    result = []
    edge_count = 0
    i = 0
    self.graph.sort(key=lambda edge: edge[2])
    parent = list(range(self.V))
    rank = [0] * self.V  # Initialize rank array
    start_time = time.perf_counter_ns()
    while edge_count < self.V - 1 and i < len(self.graph):
        u, v, weight = self.graph[i]
        i += 1
        root_u = self.find(parent, u)
        root_v = self.find(parent, v)
        if root_u != root_v:
            edge_count += 1
            result.append([u, v, weight])
            self.union(parent, rank, root_u, root_v)
    end_time = time.perf_counter_ns()
    experimental_time = end_time - start_time
    E = len(self.graph)
    V = self.V
    theoretical_time = E * math.log2(V)
    return experimental_time, theoretical_time, E, V
```

3.2 Data Normalization Notes

Average of Experimental results = 39603480 ns

Average of Theoretical results = 725948.006130544 ns

Scaling factor = Avg. (Exp result) / Avg. (Theo. result)

= 39603480 / 725948.006130544

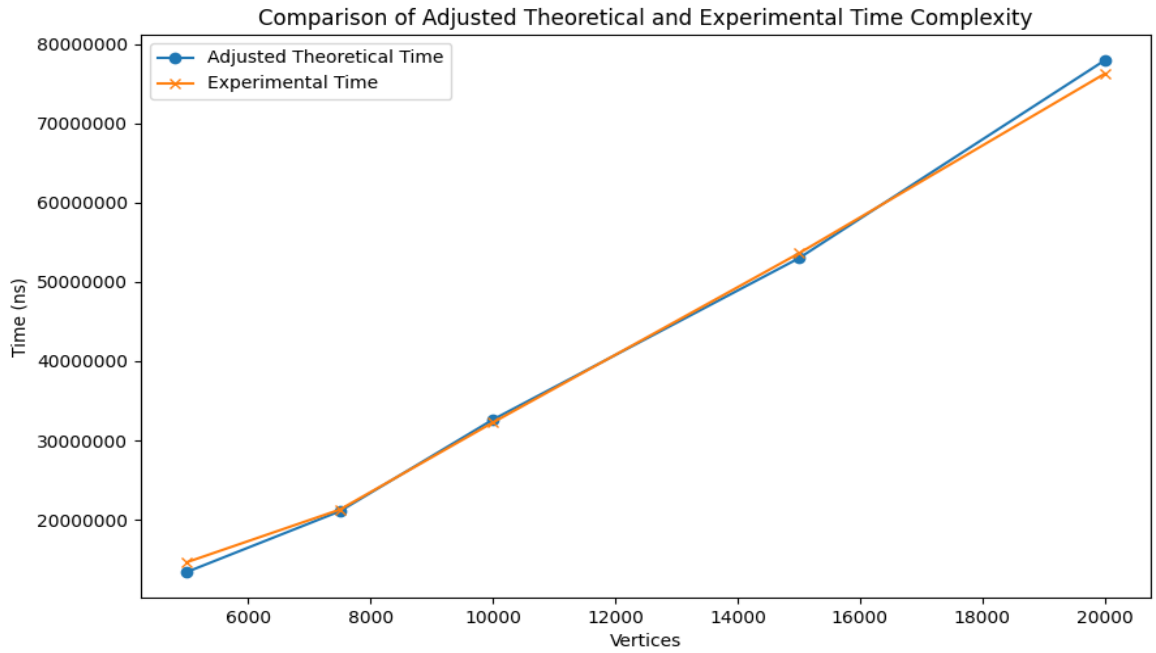
= 54.55415493334695

It was observed that the theoretical results were off by a factor of **54.55415493334695.** Therefore, to scale the values we multiply all theoretical results by **scaling factor** to get the adjusted theoretical value.

3.3 Output Numerical Data

Vertices	Edges	Experimental Time (ns)	Theoretical Time (ns)	Adjusted Theoretical Time (ns)
5000	20000	14638700	245754.24759098902	13406915.298606921
7500	30000	21275800	386180.2464081182	21067736.994746584
10000	45000	32253900	597947.0570797253	32620496.39386619
15000	70000	53591800	971087.2416189425	52976843.83307631
20000	100000	76257200	1428771.2379549448	77945407.47970398

3.4 Graph



3.5 Graph Observation

Both the adjusted theoretical time (blue line) and experimental time (orange line) show a positive linear relationship with the number of vertices, indicating that as the number of vertices increases, the time taken also increases. The experimental time closely follows the adjusted theoretical time, suggesting that the experimental results are consistent with the theoretical predictions. The graph demonstrates that the algorithm scales predictably with the number of vertices, which is crucial for understanding its performance in larger datasets.

4 Conclusion

The graph shows a linear increase in both the adjusted theoretical and experimental time complexities as the number of vertices increases. Interestingly, the experimental times are consistently lower than the adjusted theoretical estimates, indicating that real-world implementations may benefit from optimizations or practical factors not fully captured by theoretical models. This suggests that while the theoretical analysis offers a close approximation, actual performance tends to be more efficient in practice. Ultimately, we can conclude that the theoretical time complexity estimate of O(E * log V) for the algorithm is accurate.

GITHUB LINK: https://github.com/ruthvik279/Kruskals-Algorithm-for-Minimum-Spanning-Tree