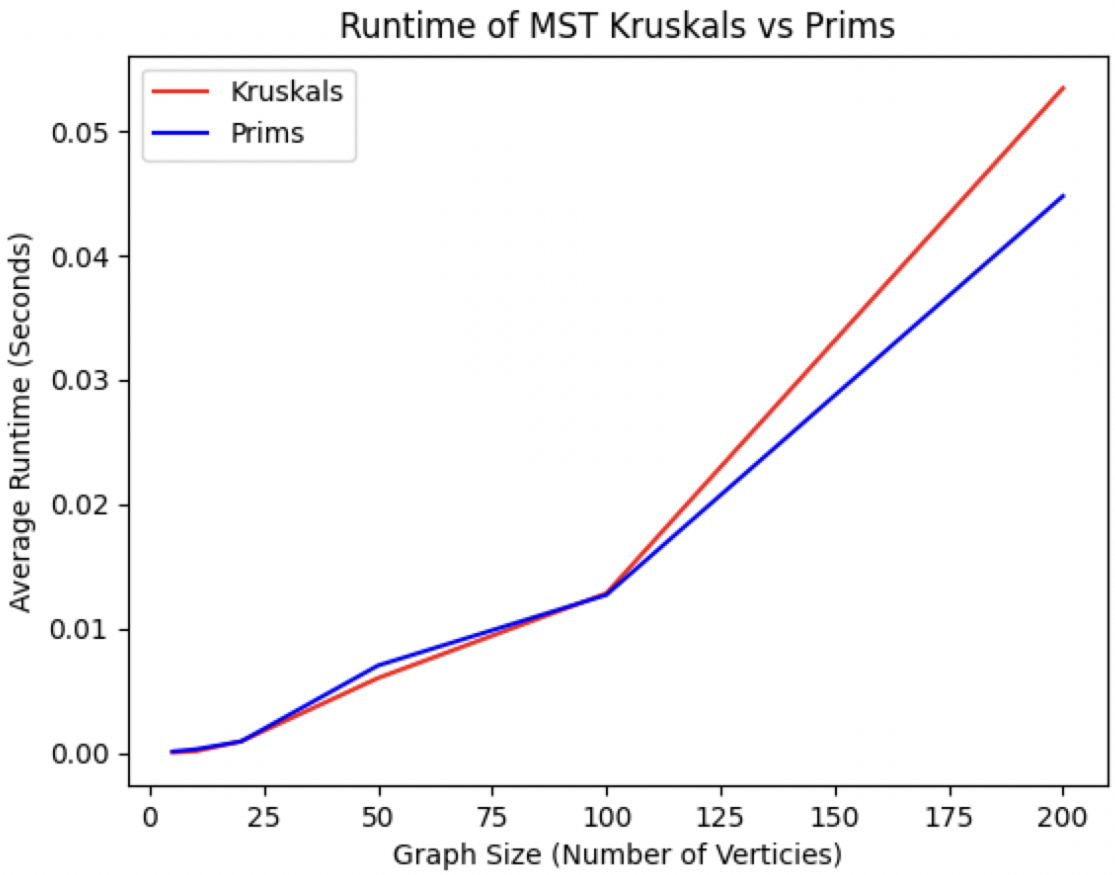
Analysis of Minimum Spanning Tree Algorithms
CSC 349-01
Lab 05
Andrew Okerlund (apokerlu@calpoly.edu)
Noa Kehle (nkhele@calpoly.edu)

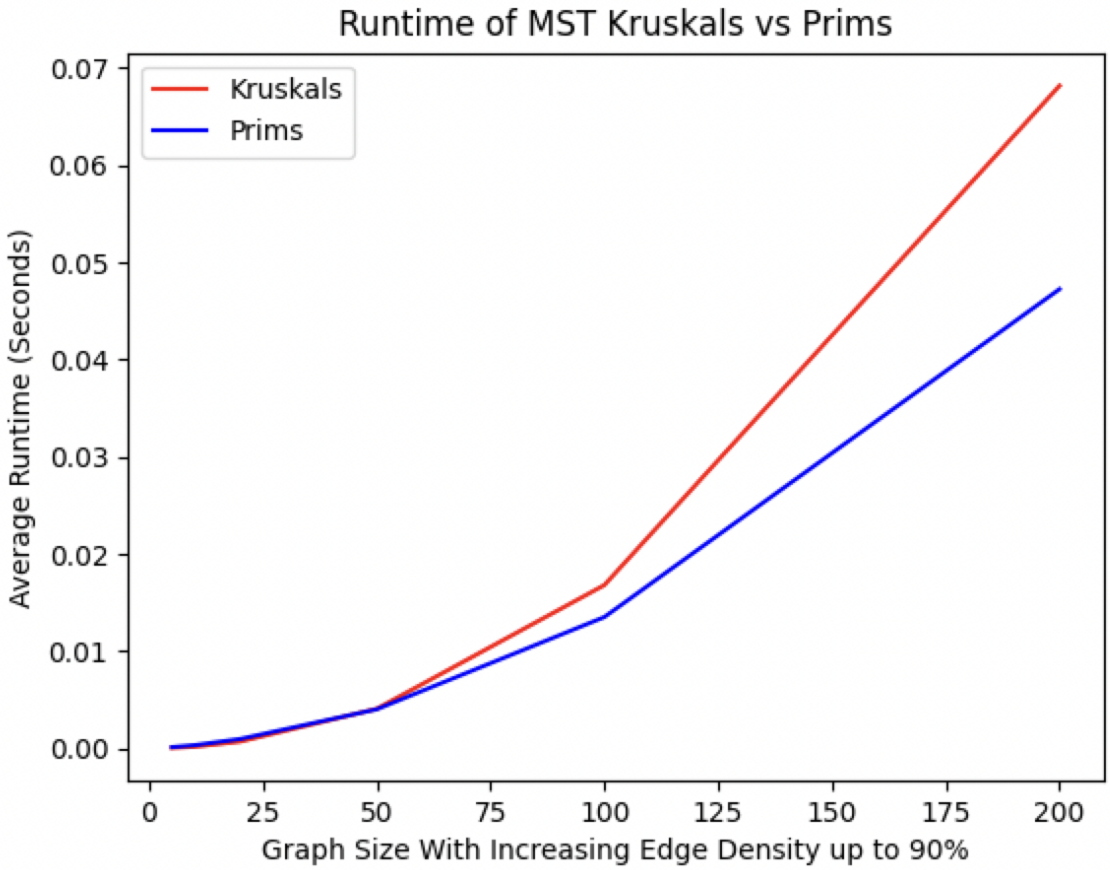
MST Algorithms

In our experiment we implemented Kruskal's and Prim's algorithms to find the minimum spanning tree of given graphs. Our testing trails consisted of graphs of the following size: 5, 10, 20, 50, 100, 200. An additional variable applied to the graphs was the density of vertices in those graphs. Each size graph went through trials with vertice densities of 10%, 25%, 50%, 66%, 75%, and 90%.

Our hypothesis was that Kruskal's would run slower because of our implementation of Union and how the disjoint sets are handled. Additionally, we predicted that Prims would perform better as the density of the graphs increased.

Graphs:





Tables:

```
Kruskals AVG:  [4e-05, 0.00013, 0.00093, 0.00602, 0.01283, 0.05347]
Prims AVG:     [0.00011, 0.00029, 0.00092, 0.00703, 0.01271, 0.04479]
```

```
Kruskals AVG:  [8e-05, 0.00017, 0.00067, 0.00402, 0.01691, 0.06821]
Prims AVG:     [0.00015, 0.00032, 0.00097, 0.00403, 0.01343, 0.04743]
```

Analysis:

Runtimes for each:
Prim's → O(V^2)
Kruskal's → O(E+VlogV)

After the testing trials concluded there was a very clear observable difference in performance between the two mst algorithms. The first graph shows the two compared with random edge densities and increasing sizes. The second has an increasing densities, 10%, 25%, 50%, 66%, 75%, and 90%, while the size also increases. The produced graphs illustrate that Kruskal's runs faster on smaller less dense graphs while Prim's performs measurably slower, but as the size and density of the graphs increase the results are flipped. The new behavior presents Prim's as being faster than Kruskal's in the trials with larger and denser graphs. This is noted on both graphs, but is especially apparent in graph two where the blue line (Prims) takes over, meaning that at this point Prim's performance becomes faster than Kruskal's.

The reason this trend is visible comes down to the mechanics of each algorithm. On smaller less dense graphs Kruskal's is faster because of its sorting procedure. Once the graph is sorted by weight it becomes less work to produce the disjoint sets and unionize them. However, as the graph size increases the initial sorting step begins to take a significantly longer time. Additionally, the way we implemented our Union function in Kruskal's is not ideal and has a worse case of O(n). Another reason for this behavior is that Kruskal's has a time complexity of O(E+VlogV) where E can equal V^2 worse case. This supports our observation that Prim's is a more efficient algorithm in larger denser graphs.