Lab 6 Kruskal’s Algorithm and Topological Sort

Introduction

The learning objective of this lab is to implement Kruskal’s algorithm and Topological Sort. Kruskal's algorithm involves creating a minimum spanning tree based off a given tree. Kruskal’s is utilized with a tree that is not directed and is weighted. Topological sort on the other hand is utilized with a directed graph while the weights play no factor.

Design

Take note that I implemented my these algorithms to works for both AL and AM graphs.

Kruskal’s Algorithm

To start off I created an edge object, the only purpose of this object is to contain data. It contains the source of the edge, the destination, and the weight of the object. The object will be referred to as edge sort object.

The first order is to sort the edges, to do this I call the sort_list function, this function first creates and edge sort object for every edge in the graph. Once all edges are loaded on to this list I sort them using quicksort.

One thing I had to sort out was creating a function to remove the corresponding edge I just loaded into the list. For example being that the graphs we’re analyzing are not directed that means there is a copy of every edge, so I created a function to go in and look for this copy.

Next is to create a disjoint set forest, with the size as the number of vertices in the original graph. Now we get our new graph and start inserting edges, as we do this we make sure to also insert them into the disjointed forest. This way we’re able to detect cycles. I insert the edges from least to greatest.

Topological Sort

Now the majority of this algorithm is given to us by our professor, the only thing we have to code is find the inth degree for every vertex in the graph. We do this by creating an array with the size of the number of vertices in the graph. Then we go through each edge and analyze the destinations, we then just add 1 to the corresponding index.

Result

I did not time my algorithm seeing as we're implementing a known algorithm. I did get all my test cases to pass. My test cases involved testing with both AL and AM graphs, testing utility functions separately, and a full end to end test.

Kruskal's Algorithm

- AM: $N^2 * N * logN$
- AL: $(N + M) * N * logN$ (M being the number of edges)

Topological Sort

- AM: $N^2 * M * N$ (M being the number of vertices with 0 as their inth degree)
- AL: $(N + m) * N * I$ (M being the number of edges, I being number of vertices with 0 as their inth degree).

Conclusion

Overall, the biggest learning from this lab was detecting a cycle within a graph, using path compression is also the most efficient way to test. I also nailed down my understanding of topological sort, when I first learned the algorithm I was a bit confused but actually coding it and creating test cases for it I have a better understanding for it. Overall, I wouldn't run these algorithms often as they can be slow and meant for one purpose.