Lab 6
Mark Williams
CS 2302

Introduction: In this lab, I implement the topological sort algorithm and Kruskal's algorithm. I test my algorithms on hard-coded graphs.

Design Implementation: In this lab, I implement graph algorithms (Kruskal's, topological sort). To do so, I follow the algorithm implementations discussed in class adapted for graphs that have an adjacency list implementation. For topological sort, I use Python's deque as a queue instead of creating a whole new Queue class and I create the function compute_indegrees_every_vertex using the knowledge that the graphs given will be implemented using an adjacency list. Other than that, the algorithm is nearly identical to the one given in class. For my implementation of Kruskal's algorithm, I first collect all the edges in the graph and sort them based on their weight. I add edges to a list if the edge to be added does not create a cycle. To check for cycles, I use a discrete set forest. Once all the valid edges have been added to the list, it is returned to the user.

Results: I tested the topological sort algorithm on three unique hard-coded graphs. The visual representation of the graphs created can be found in the code. The results of topological sort on the three graphs are below:

```
The topological order of vertices in  graph 1 is as follows:
[0, 3, 2, 1, 6, 5, 4, 9, 8, 7, 10]
The topological order of vertices in graph 2 is as follows:
None
The topological order of vertices in  graph 3 is as follows:
[0, 3, 1, 2]
```

I tested Kruskal's algorithm on three unique hard-coded graphs. The visual representation of the graphs created can be found in the code. The results of Kruskal's algorithm on three graphs is below:

```
A minimum spanning tree for graph 1 can be created from the following edges:
[0, 2, 1.0]
[2, 3, 2.0]
[1, 3, 4.0]
A minimum spanning tree for graph 2 can be created from the following edges:
[0, 2, 1.0]
[1, 4, 2.0]
[1, 2, 3.0]
[1, 3, 4.0]
A minimum spanning tree for graph 3 can be created from the following edges:
[2, 3, 1.0]
[1, 3, 2.0]
[3, 4, 3.0]
[0, 1, 7.0]
```

The time complexity of running topological sort on each of the graphs in seconds:

	Topological Sort Time Complexity
Graph 1	0.000097813
Graph 2	0.000005434
Graph 3	0.000006712

The time complexity of running Kruskal's algorithm on each of the graphs in seconds:

	Kruskal's Time Complexity
Graph 1	0.000996113
Graph 2	0.000623028
Graph 3	0.000084412

Conclusion: In this project, I learned how to implement graph algorithms and how to use them on graphs.

Appendix: Code for this lab is located at: https://github.com/mawilliams7/lab6

"I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provide inappropriate assistance to any student in the class."