CSCI3383 Algorithms: Final Project Report

Maximilian Bahar

Due Friday, May 14th 2021

How much did I achieve?

I worked on the default final project of extending the Graph API implemented in Homework 5.

I added all of the following graph algorithms:

1. Strongly Connected Components (SCC) for directed graphs using Kosaraju-Sharir's Algorithm.
2. Minimum Spanning Tree (MST) for weighted, connected, undirected graphs using Prim's Algorithm.
3. Single-source Shortest Path (SSSP) for weighted, directed graphs using Bellman-Ford's Algorithm.

How many lines of code?

The entire graph API is implemented in `myGraphs.py` spanning 281 lines of code (including format spacing and other helper functions).

Documentation is available in `README.md`.

Lessons learned?

I learned several things by taking on this project.

First, I learned that it is extremely important to plan out the structure of code before actually implementing it. This is especially critical when implementing APIs where components are designed to interact and interface with one another. Making sure that functions and classes are streamlined and well-structured is easier said than done, but greatly helps the usability of the end product.

Second, I learned that much of the work involving algorithms takes place way before coding. Being able to see the building blocks and construction of really elegant algorithms for SCC, MST, and SSSP, and the different ways of solving algorithmic problems has been a great way to think critically.

Analysis of important algorithms?

For a graph $G = (V, E)$, where V are the vertices and E are the edges, the algorithms implemented have the following runtime complexity:

1. Breadth-first Search : $O(|V| + |E|)$
2. Depth-first Search : $O(|V| + |E|)$
3. Kosaraju-Sharir's SCC : $O(|V| + |E|)$
4. Prim's MST : $O(|E| \log |V|)$
5. Bellman-Ford's MST : $O(|V| * |E|)$