John Fishbein - jhf5my Thursday, April 25, 2019

Post Lab 11 Report

During the first part of this lab, I generated a directed graph using the data given and then performed a topological sort on that graph. My implementation of a topological sort was Θ(v*e) where v is the number of vertices in the graph, and e is the number of edges. This is because I looped through each vertex in the graph, and then for each vertex, I considered each of the nodes connected to it. In terms of space complexity, I created both a graph class and a vertex class. Each vertex consists of a boolean, integer, and string field along with a vector<string> field to hold the outgoing edges of a given vertex. The graph class consists of an int field, a vector<string> field, and a map<string, vertex> which maps a string to the corresponding vertex. For the in-lab, I implemented a version of the traveling salesman algorithm. My implementation ran in $\Theta(v!)$ time where v is the list of vertices (cities) visited along the way. This is the case because in any given situation, in order to find the shortest path to visit each of the cities in the list v, every possible permutation of the list v must be considered.

The in-lab algorithm is extremely inefficient. There are several ways to improve the runtime on large datasets, but some of them come at a slight cost to the accuracy of the algorithm. One of these such changes is a progressive improvement algorithm. Essentially, instead of checking every possible permutation and returning the exact correct answer, a progressive improvement algorithm will continuously search every possible order and only update when new minimum path is found. Another possible improvement that does come at a cost to accuracy would be an approximation algorithm such as the nearest neighbor algorithm. Instead of trying every possible order of visiting cities, this algorithm allows the traveler to choose parts of the order outright with no consideration to length. While this does not always yield the optimal answer, it is much faster and produces an answer relatively close to the true shortest path. One other improvement, called the Ant colony optimization, relies on probability models to pseudo-randomly attempt select paths. Again, this algorithm may not always produce the exact result, but it runs fast enough and is accurate enough to be viable.

Sources:
https://en.wikipedia.org/wiki/Travelling_salesman_problem#Exact_algorithms