**Project 2**

Danny Foley

Computer Science and Engineering

Speed School of Engineering

University of Louisville, USA

[danny.foley@louisville.edu](mailto:danny.foley@louisville.edu)

**Introduction**

The Traveling Salesman Problem is a well-known nondeterministic polynomial-time hard problem that has been studied for decades. The salesman is given a list of cities and the locations of said cities. With this information the salesman or program in this case is supposed to find the best (least expensive) route that takes them to all the cities once and returns them to the starting city. For this project, the language, C++, was used to solve the prompt given.

**Approach**

There were two methods to solve this TSP prompt. The first approach was to use breadth-first search (BFS) and the second was to use depth first search (DFS).

**BFS**

Given the edge information from the prompt (Figure 1.1) and the data (Figure 1.2), a method that uses the BFS approach was created. BFS in general is a method that is focused on reaching all the nodes that are adjacent to the current node, before moving on to the next nodes.

At the beginning, the first city, city 1, is used as the starting point with the goal being city 11. Then the BFS algorithm checks the nodes connected to city 1 and adds them to the queue. For each of these cities, the distances are updated and their “citybefore” list is updated to show that city 1 was how they were found. This continues, recursively, until all the nodes are dequeued. Leaving the shortest route saved in city 11’s citybefore list as well as having that distance saved.

**DFS**

The DFS algorithm is used to go into a graph by its depth, so it will search all the way into a graph until there are no more adjacency nodes. When multiple adjacent nodes are found they are added to a queue so that the remaining nodes can be looked at as well.

The same method of saving the route and distance of the route was used as in BFS, the only difference was that the DFS approach was used to traverse the graph of cities.

**Results**

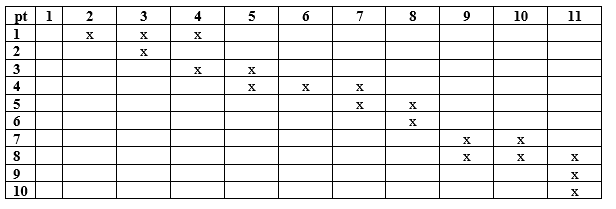
Both approaches were implemented and were able to find the same route. The time each algorithm took was fairly the same. Each took just around 1 ms to complete. These results seem very unusual as normally the DFS algorithm is much faster and the BFS algorithm is usually the algorithm that is used to find the shortest path. The output (Figure 1.3) shows the printed information from the program.

Figure .1 showing the edge information

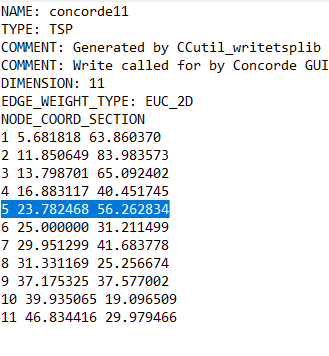
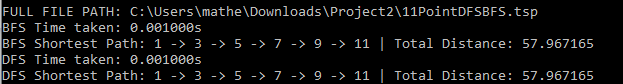


Figure 1.2 Showing the xy-coordinates of the cities (nodes).

Figure 1.3 shows the input/output of the program