**CSE 545 – Artificial Intelligence – Project 2 – TSP BFS DFS Problem**

Jordan Richardson

BA Computer Science

Speed School of Engineering

University of Louisville, USA

[Jkrich05@louisville.edu](mailto:Jkrich05@louisville.edu)

**Introduction**

This project revolved around some specific implementations of the TSP problem, as well as opting for more optimized search methods, and altering the base problem to only require the arrival at a goal city, using a possible paths table, with no need for a return to the initial city. This allows us to further understand that the method in which we search, and the basis of the problem itself, can make massive differences in performance if brought to a large scale. For this problem, we used DFS and BFS search methods, which both perform the same task, however the accomplish it using different approaches. DFS explores each path by navigating to the end, before coming back to the start and trying other any other possible paths, which allows it to eventually test all paths and find the shortest distance for the salesman. BFS works similar in that it navigates the branches possible, however it approaches it differently. BFS explores each possibility in a branch, before moving to the next depth, and exploring its possibilities considering what it has searched already.

**Approach**

The two search methods we are using for this problem are Depth-First-Search and Breadth-First-Search. These specific algorithms are not the best by any means; however, they perform much better than brute forcing all possible permutations of the path, and while in this problem we do not return to the initial city, you could simply reverse the path, or do a backwards search using either method to find the shortest path back to the starting city. While in lecture, we were told we could consider the problem in many ways, even taking the shortest path to be the least city leaps instead of distance. However, I had already finished my problem due to a misunderstanding of the due date, so I have stayed with shortest distance just like in the first problem, which allowed me to reuse some of my code from project 1.

**Results**

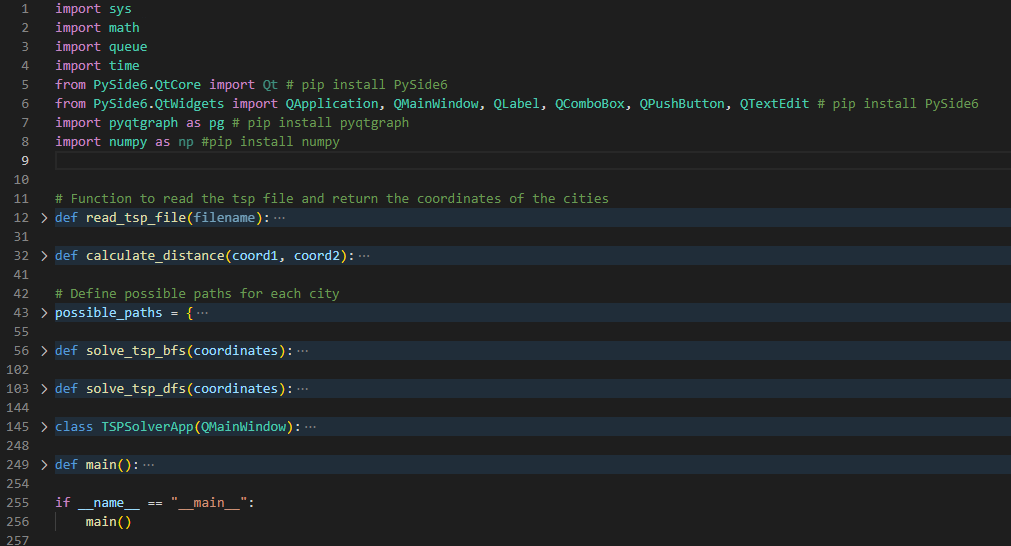
So, I manually coded a lot of the search methods instead of using a library like Dijkstra’s to perform my searches. While I am not sure how much this specifically affects performance overall, I am able to conclude that BFS is more efficient given this specific problem when both methods are trying to find the shortest distance to the goal city. This could be because a variety of reasons, such as requiring each method to find the shortest path, which may cause the traversal to take longer than necessary, as well BFS works better when it is closer to the source, while DFS is better when further from the source and we are only traversing so many nodes at minimum and maximum given the directed path. BFS is also optimal for finding the shortest path while DFS is not optimal for finding shortest path.

**Results – Data**

For this project, we were given a single .TSP file that contained some basic information about how it was generated up to line 7, and a list of cities with city number, longitude, and latitude. Our problem had 11 cities in this tsp problem that we were able to navigate between, along with a document that given us information about the points as it is a directed graph. This means we have set locations to travel to, depending on what city we are currently residing in. This means we must hard code set locations(edges) in to complete the problem which adds another level of complexity to TSP which I found super interesting.

**Results – Results**

This is the structure of my code, it helped to separate the problem into sections.



This is what my application outputs for the BFS algorithm.

A screenshot of a graph

Description automatically generated

This is what my application outputs for the DFS algorithm.

A screenshot of a graph

Description automatically generated

As you can see, both formulas were able to find the path with the shortest distance through tracking information as it traverses the nodes.

However, like I stated above, BFS is faster, when DFS is generally considered more optimal search method. I believe this is generally because BFS is more suited for finding the shortest distance, while DFS is not optimized for finding the shortest distance, but I am forcing it to find it.

**Discussion**

During this assignment I learned a ton about algorithm as I read online about BFS and DFS, I did not have a lot of knowledge regarding algorithms prior to this class and I am very grateful for the opportunity. I also further refined my graphing and application skills even more compared to the last assignment, or really compared to ever before. This assignment itself is probably my most sophisticated program so far in my undergrad pursuit, and it excites me to think about where the class, and myself can develop from here. I would like to possibly see an assignment where we graph a TSP considering the curvature of the earth, or TSP in a 3D environment as it would challenge my graphing and application skills and I want to get as much as I can out of this class.

**References**

I did not use any references for my code other than talking to other students about libraries for graphing, however in reading about DFS and BFS I did access a few websites, and two specifically I visited a few times because of the visual example for DFS and BFS as I wrote my code.

<https://www.geeksforgeeks.org/difference-between-bfs-and-dfs/>

<https://nicksypark.github.io/2017-01-09/BFS-and-DFS/>