**Project 2: TSP using**

**DFS and BFS**

Trevor Bright

Computer Engineering/Computer Science

Speed School of Engineering

University of Louisville, USA

[t0brig01@louisville.edu](mailto:t0brig01@louisville.edu)

1. **Introduction** (What did you do in this project and why?)

For this project, I used Python through Visual Studio Code and the command line to run and compile the program. This project is using Breadth First Search and Depth First search to find a path from the first city node to the last city node. I used Python because it is what I am most familiar with for projects like this. This is the same language I used for the first project, so a lot of the code was reused such as the file parsing code and the GUI formatting functions.

1. **Approach** (Describe algorithm you are using for this project)

My approach to the problem was so strip out the data from the files provided, make an object based on a class I made that holds the number, the x, and the y coordinates for each of the cities. I reused the parsing method and functions from the first project. For the breadth first search algorithm, I started at the first city and looked at the cities that it was connected with. I used the hard-coded data provided to implement which cities could go to which cities. I then calculated the distance from the current node to the other nodes it could possibly go to. I then progressed to the next node until I hit the last node, city 11. For the depth first search, it would start at the first city and it would cycle through each of the possibilities until it resulted at the last city. This algorithm is more of a greedy algorithm as opposed to BFS, which is more calculated and thought out. After I found the paths, I used the matplotlib library to return a visual representation of the paths.

1. **Results** (How well did the algorithm perform?)
   1. **Data**

I used the city data provided and implemented my DFS and BFS algorithm to find a path from the first city to the last city. I then used the matplotlib Python library to generate a visual representation of the path.

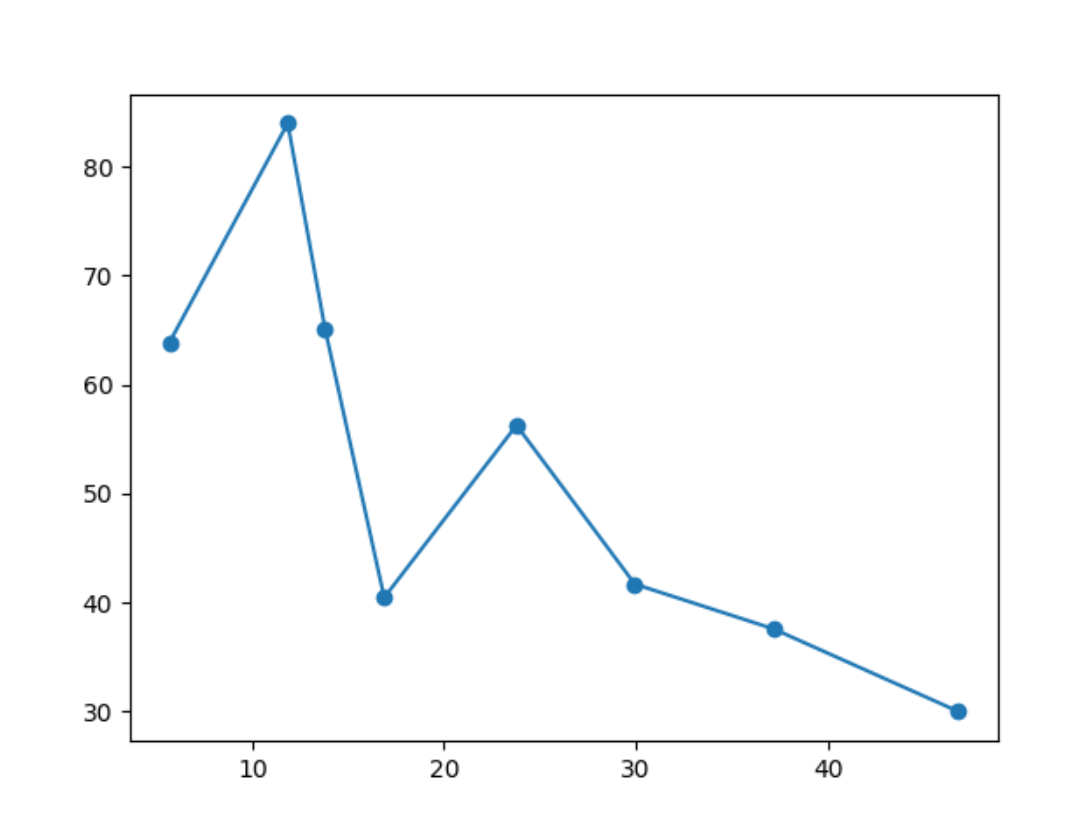
* 1. **Results**

Here is the data I got for depth first search and breadth first search using the 11PointDFSBFS.tsp file.

Depth First Search

----------Path-----------

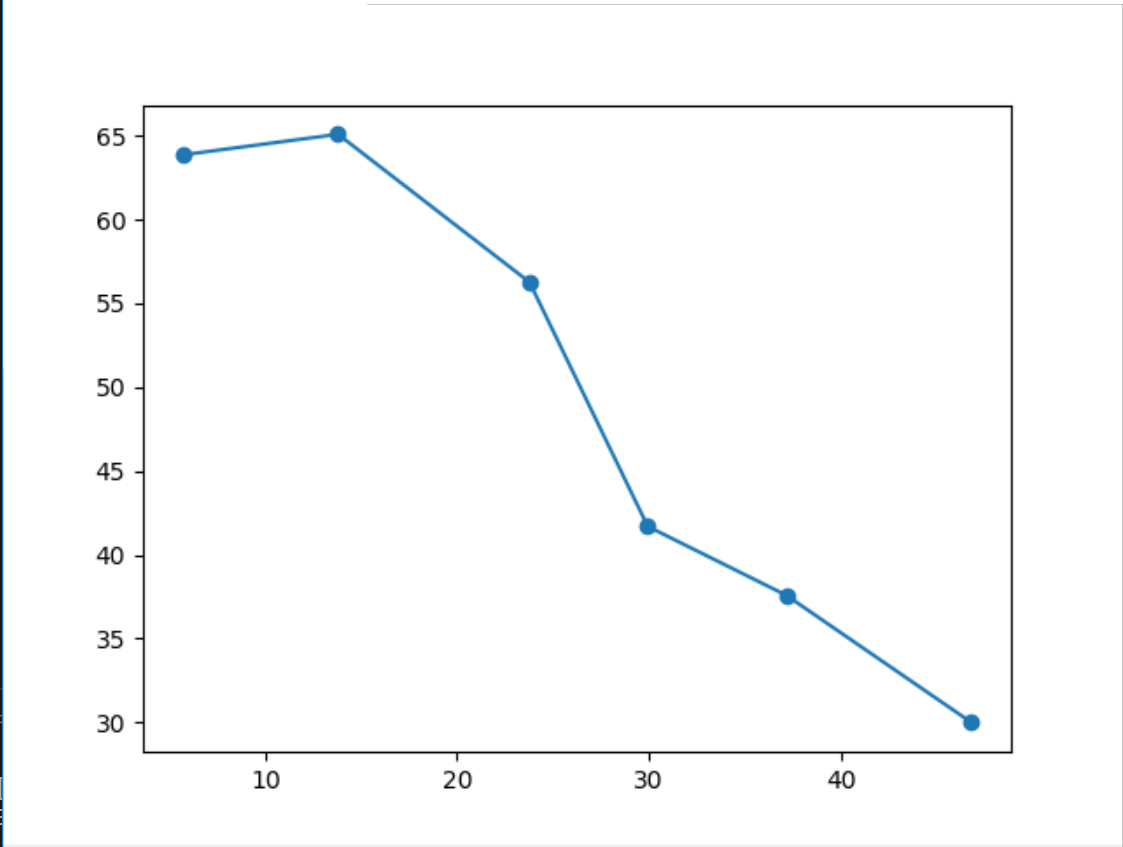
[1, 2, 3, 4, 5, 7, 9, 11]



Breadth First Search

----------Path-----------

[1, 3, 5, 7, 9, 11]



-------Time Elasped-------

0.0160000324249 seconds

1. **Discussion** (Talk about the results you got and answer any specific questions mentioned in the assignment.)

While you can use depth first search and breadth first search to find a path from the first to the last city, it is not feasible. While working with the data set and the algorithm, neither returned the perfect optimal path. This method also does not satisfy the requirements for the Traveling Salesman problem. The problem states that all cities but be visited and returned to the starting city. This method does not even visit every city. It can only find paths from point A to point B. While this is a good algorithm for weighted graphs, it is not good for finding paths.

1. **References** (If you used any sources in addition to lectures please include them here.)

Matplotlib library and documentation: <https://matplotlib.org/>