**Traveling Salesman Project 2**

Tyler Hamilton

CECS, 1713904

Speed School of Engineering

University of Louisville

jthami05[@louisville.edu](mailto:NameID@louisville.edu)

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

In this project, I implemented two search algorithms, breadth first search (BFS) and depth first search (DFS) as solutions to the Traveling Salesman Problem. The purpose of this project was to compare the two algorithms by speed, distance traveled, and number of traversals executed.

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

Breadth first search runs by exploring each node at each level before moving on to the next level. This is done in code by looping through all nodes adjacent to the current node, and then moves to the child of the leftmost node, and repeats the process until all nodes have been explored.

Depth first search runs by exploring each branch entirely before moving on to the next closest one. This is done in code by visiting nodes at a certain level, visiting each node’s children, and repeating until there are no more children, moving from left to right until all possible branches have been explored.

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

BFS is much more efficient in terms of travel distance than DFS, with a distance of 63.7721 compared to DFS’s 173.871. It is also slightly more efficient than DFS in terms of number of traversals, with 9 traversals rather than the 10 for DFS.

This is in line with my expectations for these algorithms, as BFS will quickly find the shortest distance traversing level by level, as a path with less traversals is likely to cover less distance.

* 1. **Data** (Describe the data you used.)

I used a .tsp file given in the project. The name of the file to be read was passed in as a command line argument. I ignored the first sections of the files and read the remaining lines in, as IDs, x-values, and y-values for *city* objects.

* 1. **Results** (Numerical results and any figures or tables.)

The program works as expected.

Breadth-first, weighted:

printing in order...

city 10

city 7

city 4

city 1

total distance: 63.7721

Time taken: 0.000293s

Breadth-first, unweighted:

printing in order...

city 10

city 7

city 4

city 1

Number of traversals: 9

Time taken: 0.000562s

Depth-first, weighted:

printing in reverse order...

city 6

city 4

city 8

city 10

city 11

city 9

city 7

city 5

city 3

city 2

city 1

Total length traveled is 173.871

Time taken: 0.000827s

Depth-first, unweighted:

printing in reverse order...

city 6

city 4

city 8

city 10

city 11

city 9

city 7

city 5

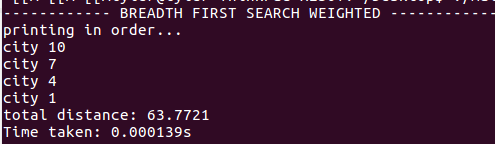
city 3

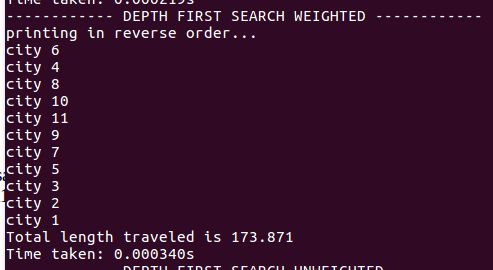
city 2

city 1

Number of traversals is 10

Time taken: 0.001123s





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

compare weighted vs. unweighted.

The basic process of my program is as such: The dataset is read in as a command line argument, and separated into city objects with x and y coordinates. The depth first and breadth first algorithms, both weighted and unweighted, are called on the dataset. The total distance per route is collected during the algorithm’s execution and printed. The list of cities traversed is also logged and printed.

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

Wikipedia page on BFS:

https://en.wikipedia.org/wiki/Breadth-first\_search

Wikipedia page on DFS:

https://en.wikipedia.org/wiki/Depth-first\_search