CSCI 2824 Discrete Structures
Instructor: Hoenigman
Programming Project 2
Due Monday, December 16 by 5pm

Submit your code through Moodle and provide a hardcopy of your code and the output of your code. You can bring the hardcopy to class anytime between now and the end of the semester, or put it in my mailbox in the computer science office on the seventh floor, or slide it under my office door (ECOT 738). Your assignment will not be graded if I don't receive a hardcopy from you.

Your code should be written in Python, or some other obscure language that only a handful of people know. If you choose an obscure language, you need to clear it with me first, and provide significant comments in the code so that I can understand it.

Graphs, circuits, and the traveling salesman problem

One of the classic computer science problems is known as the traveling salesman problem (TSP), where a salesman is tasked with finding the shortest path through a set of cities, visiting each city exactly once, and then returning home to the city of origin. This problem generalizes to any number of situations where there is a set of objects and a distance (weight) between those objects and the objective is to find a circuit through the objects that minimizes that weight.

In lecture, we looked at four algorithms for finding the shortest path through a Hamilton circuit. Those algorithms were exhaustive search, nearest neighbor, repeated nearest neighbor, and cheapest link. In this assignment, you will be implementing those algorithms on a data set of your choice, and reporting on the performance of each algorithm, where performance includes the speed and solution quality. A complete description of those algorithms is available on Moodle in the file called: Lecture21 – Mathematics of Touring.

Data sets

For this assignment, you are welcome to use a data set of your choosing. The only requirement is that the data set has at least 10 nodes. There are several data sets featuring distances between US or European cities available here:

http://people.sc.fsu.edu/~jburkardt/datasets/tsp/tsp.html

A Google search on traveling salesman data sets will yield other data as well. Once you've identified your data set, you should save the data as either a .txt or .csv file. Your program will need to read this file.

Shortest path algorithms

Using the downloaded data, implement three of the four shortest-path algorithms that are discussed in the Lecture21 – Mathematics of Touring file. Depending on the size of your data set, it may be difficult to write an exhaustive search that completes in a reasonable amount of time. If this is the case, add print statements to your code that output the time required to calculate some portion of the total number of solutions. For example, you could display, “Time to calculate 10%: 24 hours.” This statement should be in your code and use a time library, or other method of getting the system time, that starts counting at the beginning of the exhaustive search. You do not need to run the complete search if it becomes clear that it will not complete in time. But, your output should show at least one message of the time to complete some percentage of the search.

Some of the data sets provided at the above-listed website also include the optimal solution. If you select one of those, you will be able to easily compare the results of your other algorithms to the optimal solution without writing an exhaustive search.

Implementation details
In your code, use comments to describe the data set you selected, the website or other source where you found it, the algorithms you used, and an overview of the results.

The results should include a brief explanation of the differences in the solutions for each algorithm, including which algorithm produced the shortest path and which produced the longest path. If you have the optimal solution, then you should also compare your results to the optimal solution. Results should also include a brief discussion of how long it took each algorithm to complete and a comparison to the time to complete the exhaustive search. For the non-exhaustive-search algorithms, you should run your code at least 10 times and calculate the average completion time and report that number.