Comp 496ALG Fall 2015

Computer Project #2 Traveling Salesman Problem

Points: 35

Due: Nov 3

**Euclidean Traveling Salesman Problem Statement**: In the Euclidean traveling salesman problem, a salesman must visit n cities and return the first city visited, in such a way that the total distanced traveled is minimized. In other words, we are given a set of n points in the plane and wish to find the shortest closed tour that connects n points.

In this project you will implement in Java or C++ three different methods that “solve” the Euclidean Traveling Salesman Problem(TSP) and then you will do a comparison of the runtimes and accuracy of the three TSP methods for various values of n. [n = number of cities visited]. Label the cities 0, 1, 2, .., n-1. In this project ALWAYS start and end the tour at vertex 0, that is the startCity is always city 0. All work must be your own. Do not download solutions from the internet or from other outside sources.

Input: double[] xloc //xloc[k] and yloc[k] are the x and y locations of the city k

double[] yloc

int n //number of cities

String[] cityNames //cityName[k] is name of city k

Output: (See also item 6 for formatting)

1. List of cities that describes the best tours found using each of the described methods, starting and ending with vertex 0.
2. Cost of the best tour found using described method
3. **Brute Force Method. (Generates optimal solution)** Use a permutation algorithm to generate all possible tours of the cities. Find the tour that minimizes the cost (i.e. total distance) of the tour. Use your permutation code from Project 1 if you wish.
4. **Greedy TSP Method (Generates an Approximate Solution):** The salesman always selects the nearest unvisited city as the next city to be visited. Start at the startCity. After visiting all cities, return to the startCity.
5. **Minimum Spanning Tree Method.** **(Generates an Approximate solution).**
   1. Create a distance matrix of distances between all vertices i and j. Consider this as an adjacency matrix for the graph of the city locations. Call this graph G
   2. Find a minimum spanning tree of G using Prim’s algorithm, starting at the startCity. Call the tree T. It should be in adjacency matrix form.
   3. Run a depth-first search of T starting from the startCity. Output the nodes in visited order to an array w. You approximate tour will be w[0], w[1], ..w[n-1], w[0];
6. **Test your TSP methods** using a random placement of n cities in a 300 by 300 grid. Vary n. The Brute Force methods will crawl at n > 10 or 11. Others will be faster. You need to do a thorough analysis of the speed and accuracy of the solutions. Does the accuracy of the approximate methods depend on n in this scenario?
7. **Test your method on your instructor’s test case.** (Available October 27). Format of your test run results for instructor’s test case must be formatted like this:

Optimal Tour: 8 Cities

Min Cost 522.6888976979186

0 3 5 7 6 4 1 2 0

A D F H G E B C A

Best Greedy Tour: 8 Cities

Min Cost 680.815550206437

0 6 2 4 7 1 3 5 0

A G C E H B D F A

MST Tour: 8 Cities

Min Cost 535.2515456298568

0 2 4 1 6 7 3 5 0

A C E B G H D F A

1. **HAND IN:** Write a formal report that describes your experiments and presents your results. Compare the run times and accuracy of the various TSP METHODS. Attach all of your code with the printout of the instructor’s test case runs for each TSP method.