CSCI 232: Lab Three
Due: April 7, 11:59pm

Overview

In this lab, you will write a program to solve the Travelling Salesman Problem (TSP). TSP is one of the classic "hard" problems in CS so we'll explore some *heuristic* approaches to solve it. Briefly, a salesman wishes to visit each city exactly once and return to the city he started at least total cost. (This problem is also called the Hamiltonian cycle problem.)

https://en.wikipedia.org/wiki/Travelling_salesman_problem (for more info)

Notes

- Your program should run from the command line:
  `java TSPcompare 10`
  (you will need `algs4.jar` in your CLASSPATH)

- The argument (`10` above) to your program should be the number of vertices to use. Create this many random vertices in a 100 x 100 box. (Each coordinate is a random double in the range [0.0, 100.0].) Edge weights are given by the normal Euclidean distance between pairs of points.
  Your program should compute a tour that starts and ends at vertex **0** and visits each vertex exactly once. *You should also report the cost of the tour and the time to find it.*
  (if there does not exist a solution, your program should also detect that)
  You can decide the output format, e.g.:

```
greedy tour:
0 -> 5 -> 4 -> 3 -> 1 -> 2 -> 0
total cost: 25.0
time to find: 3 sec
```

- You should implement two TSP-finding strategies:
  1) a *greedy* approach that goes to the nearest unvisited vertex and
  2) the *"twice-around-the-tree"* method; first find a minimum spanning tree **T**, then, starting at root "walk around the tree" (*this can be done using a recursive tree traversal; you will need to figure out the details*) and write down the list of vertices encountered; after the list is constructed remove duplicate vertices. This tour will start and end at the root but you can then shift it so that it also starts at 0:

```
twice-around-the-tree tour:
0 -> 4 -> 5 -> 1 -> 3 -> 2 -> 0
total cost: 24.0
time to find: 8 sec
```

- Besides, your code, you could provide an analysis of the two methods: which algorithm is the best on various graph sizes? How many vertices can each method handle?

- *Bonus A: also implement a "branch-and-bound" exhaustive search algorithm to find an optimal solution.*
  Bonus B: draw the points in the grid and each tour found in a different color.

Submission

Submit using the D2L dropbox prior to the due date. Files to submit:

`TSPcompare.java`
(you can assume that algs4.jar is in our CLASSPATH)

*Only one submission per group is required **BUT** you must put **ALL** group member names in the comments of your program.*