CSE2046 Assignment 3

Traveling Salesman Problem (TSP)

Due date: June 12th, 2020 (for the test outputs), June 15th, 2020 (for the report and codes)

The travelling salesman problem (TSP) asks the following question: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?

Inputs are: n cities, with their locations (x and y coordinates) in a 2D grid.

Output is: Ordering (tour) of these cities so that total distance to travel is minimized.

Distance between two cities is defined as the Euclidian distance rounded to the nearest integer. In other words, you will compute distance between two cities $c_1 = (x_1, y_1)$ and $c_2 = (x_2, y_2)$ as follows:

$$d(c_1, c_2) = round\left(\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}\right)$$

For example, if three cities are given with the coordinates $c_1 = (0,2), c_2 = (2,3)$ and $c_3 = (3,0)$, then a tour with ordering $c_1,\ c_2,\ c_3,\ c_1$ has a total distance of :

$$rnd(\sqrt{5}) + rnd(\sqrt{10}) + rnd(\sqrt{13}) = rnd(2{,}23) + rnd(3{,}16) + rnd(3{,}60) = 2 + 3 + 4 = 9$$

Project Specification:

Your team (up to 2 students) is asked to design and implement a method for finding a tour which is as close to the optimal tour as possible. Since TSP problem is an NP-hard problem, it is too hard to find optimal solutions. Your goal is not to design an algorithm for the optimal solution, but you are requested to do your best. This is an open-ended project.

You may do the following:

- Read as much as you want to learn about how to solve the TSP problem. But **you have to cite** any resources you use. You may want to start with some approximation algorithms (such as local search heuristics) in chapter 12.
- You may use whichever programming language you want.

You may **not use** the following:

- Existing implementations or subroutines
- Extensive libraries (if you are not sure, check with the instructor)
- Other people's code.

Input format: Inputs will always be given to you as a text file. Each line defines a city and each line has three numbers seperated by a white space. The first number is the city ID, second number is the city's x-coordinate and third number is the y coordinate.

Output format: You must output your solution into another text file with n+1 lines where n is the number of cities. The first line will include length of the tour you find. The next n lines should contain the city identifiers in the order they are visited by your tour. Each city must be visited exactly once in this list. If your output is not in this valid format, you will not receive any credit from your solutions.

Example instances: You may find example instances in the web site. There are three example inputs and corresponding example outputs (which are not optimal). The optimal tour lengths for example inputs 1, 2 and 3 are 108159, 2579 and 1573084 respectively. You can use these values to have an idea about how good your algorithm is.

Testing: A verifying procedure `tsp-verifier.py` is also given in the same page. We will use it to verify your solutions. For example, the first example instance can be verified as follows:

```
> python tsp-verifier.py sample-input-1.txt sample-output-1.txt
```

The verifier only verifies that your output contains a correct total distance value, not the optimality of the output. Using the verifier, you should test that your program report the correct total distance.

Test instances: On June 11th, we will announce 4 test instances at the web site. By June 12th, 23:59, you will be required to submit 4 seperate output text files (according to the output format) corresponding to each of these test instances. These files should be called test1.txt, test2.txt, test3.txt and test4.txt, and should be submitted to cse246submit@gmail.com. **The deadline is strict.**

Project report: You will submit a project report by June 15th, 2020, 23:59. The project report should describe the ideas behind your algorithm as completely as possible. It should not exceed 3 pages in length in no less than 10pt. **You should also give the "division of labor – who did what?" if you do the project in a team of two students.**

Grading policy:

60% of your grade will be determined by your project report. Clarity and creativity of your work will significantly affect your grade. (Note: If you just implement a simple nearest neighbor algorithm, you can only get half of this grade)

40% of your grade will be determined by your solutions to the test instances. Studies that find solutions closer to the best possible solution will get higher grades.

Come on to the contest: "Hodri Meydan!"

For each of the four test inputs, we will indentify best, second best and third best performing teams. Best, second and third projects receive 3 stars, 2 stars and 1 star, respectively. So if a Project performs best in all four test inputs it will receive 3*4=12 stars.

Stars	1	2	3	4	5	6	7	8	9	10	11	12
Bonus Points	5	7	9	11	13	15	17	19	21	23	25	30