Catherina Vijay
Lakshman Madhav Kollipara
Shajith Ravi

CS325- Introduction to Algorithms
Project 4 – Traveling Salesman Problem

There is only one tsp.cpp file which has TSP implementation. Refer to README file for more details.

Input: Text file that contains a list of cities and their X & Y coordinates.

- Each line defines a city and has 3 numbers separated by white space.
- The first number is the city id.
- The second number is the city's x coordinate.
- The third number is the city's y coordinate.

Output: Text file with (input_filename.tour) that contains the tour length and a list of cities visited in order.

- The first line is the total length of the TSP tour.
- Each consecutive line is the city visited.

Implementation: There are many different algorithms that aim to improve the tour length of the TSP while not taking a ridiculously long amount of time. Among the most understandable are the greedy and 2opt.

The nearest neighbor algorithm is a greedy algorithm. It works by having the salesman start in a random city, then repeatedly choosing the next closest city (marking those already visited) until all cities have been visited. The salesman then makes the trip back to original city. It will quickly give short tour, but doesn't give most optimal solution. The algorithmic complexity is O(n^2).

The 2opt algorithm improves on the tour produced by the greedy algorithm. It does so by swapping points (pairs of edges) to create shorter and shorter tours. Specifically, for each point in the tour, our algorithm considers swapping it with each other point after it in the tour. In order to keep the algorithm fast, each point only considers points after it in the tour, so while the first point would consider swapping with every other point, the second to last point only considers swapping with the last point since it has already been paired with the other points. The 2-opt optimization is continued until there is no more improvement in the tour distance. Each iteration of 2-opt is O(n^2). Combined, the complexity is O(n^2) + O(n^2) = O(n^2). If we iterate through all possible starting cities, we will gain another n and the complexity becomes O(n^3).

Admittedly this method means that it might produce better results if it were run repeatedly, since any time something gets swapped, new possibilities for improvements may become available. Note that the best swap for a given point (let us call it the focus) is stored until all the swaps for the

Catherina Vijay
Lakshman Madhav Kollipara
Shajith Ravi

focus have been considered; If a swap that improves the tour has been found after the focus has finished its comparisons, then that swap is made and the next point in the tour becomes the focus.

Runtime:

tsp_example_1 – 109975, 0.30277s

tsp_example_2 – 2669, 15.5234s

tsp_example_3 – 1691503, 311s

Testing:

Verified with the python tsp-verifier provided on canvas. Returns "Each item appears to exist in both the input file and the output file.
('solution found of length ', xxxxxx)"

Competition Results:
test-input-1 -- 5373, 0.061075s
test-input-2 -- 7437, 0.541191s
test-input-3 -- 12553, 11.8841s
test-input-4 -- 17603, 88.7663s
test-input-5 -- 24456, 300.668s
test-input-6 -- 34878, 300.224s
test-input-7 -- 54679, 304.237s

References:
1. https://louisville.edu/speed/faculty/sheragu/Research/Intelligent/tsp.PDF
2. http://en.wikipedia.org/wiki/Travelling_salesman_problem
3. http://bardzo.be/0sem/NAI/rozne/Comparison%20of%20TSP%20Algorithms/Comparison%20of %20TSP%20Algorithms.
3. http://en.wikipedia.org/wiki/2-opt
4. http://en.wikipedia.org/wiki/Nearest_neighbour_algorithm
5. http://codereview.stackexchange.com/questions/72265/2-opt-algorithm-for-traveling-salesman