Course: CS335
Name: Jianhua Deng

	NearestNeighbor	GreedyTSP	Optimal
Japan (ja9847) Size: 9847 Cities	Distance: 665821 Time: 5102 ms	Distance: 582912 Time: 76286 ms	Distance: 491924 Time: 12 million seconds
Tanzania (tz6117) Size: 6117 Cities	Distance: 501184 Time: 2118 ms	Distance: 463329 Time: 9560 ms	Distance: 394718 Time: 52,196.23 seconds
Canada (ca4663) Size: 4663 Cities	Distance: 1646884 Time: 1195 ms	Distance: 1564514 Time: 11121 ms	Distance: 1,290,319 Time: 560,951 seconds
Oman (mu1979) Size: 1979 Cities	Distance: 120908 Time: 226 ms	Distance: 110799 Time: 1648 ms	Distance: 86891 Time: 9809.20 seconds
Qatar (qa194) Size: 194 Cities	Distance: 11640 Time: 2 ms	Distance: 11601 Time: 8 ms	Distance: 9352 Time: 2.09 seconds

Findings and What I Learned:

The Nearest Neighbor algorithm is an algorithm that finds the shortest path of a tour by using the Greedy Algorithm. First, select a node to start, naming it to be for example, “current”, and storing all the remaining nodes in the list of unvisited so that we can keep track of how many nodes are still unvisited as we go. Then, the second step is to remove the node that produces the smallest distance from your current node, then place the node into the visited list. Next, update the current node to be your new node that is added to the visited list. Then, repeat until there’s no node in the unvisited list. Lastly, manually calculate the distance between the last node and the first node, which is our starting point, so that it becomes a tour.

The GreedyTSP function was implemented using the idea of the Kruskal Algorithm, which also utilized the idea of the greedy algorithm by always choosing the edges with the smallest weight that doesn’t create a cycle. However, because our ultimate goal is to create a tour, we ultimately need to manually add an edge such that it DOES create a cycle. To implement this function, we first need a variable to keep track of open edges, such that whenever a node has come to have no open edges we decrement the open edges by 1, this is to help us manually add the last edge to create a tour. Second, create a list of Edges for every node in the list and store it in a priority queue. Then, repeatedly choose the Edge with the smallest weight that does not cause a cycle. To check this, we would need to make sure that the two nodes connected to the Edge can not have two edges that are already connected to other nodes. And, we have to make sure that either one of the nodes, when we trace their edges, can not return to the adjacent node that is connected with the same Edge as him because if it did, that would cause a cycle. Once that is done, repeatedly pop Edges off of the heap, check if it is addable or if it causes a cycle, if it doesn't cause a cycle and it is addable. add it to the list. Lastly, when open edges are decremented to be equal or less than 2, we break the loop, manually search for the two nodes that are missing an edge, calculate the distance, and add it to the list, thus, a tour is created.

The thing I’ve observed by comparing both of my functions is that NearestNeighbor has a much better time finding the shortest path of a tour than GreedyTSP. While GreedyTSP was able to find a shorter path to complete the tour, but it was due to a significantly higher cost of time. For example, when finding the shortest tour of Japan, the distance difference between the two algorithms is only 82909, but greedyTSP spent around 15 times more time to find the relative optimal tour, which is not a very good deal in terms of how much it costs. In terms of comparing the two functions with the optimal distance from the TSP websites, it is now apparent that to find the most optimal tour, the required time increases exponentially as the preciseness of how optimal the distance is increased. For instance, the optimal tour of Japan, required 12 million seconds, whereas for my greedyTSP it found the tour with a distance of 582912 at a fraction of the cost. Although the distance difference is still large, at a much much much smaller time cost, greedyTSP, and Nearest Neighbor were able to return a relatively optimal path of a tour.