# Greedy Algorithm

The greedy algorithm has a time complexity of O(n^2). The algorithm starts from an initial city, picks the shortest path available from that city to an unvisited city, and continues this process until all the cities have been visited. Each city is considered exactly once. The complexity of considering a single city is O(n) because each outgoing edge must be considered to find the smallest path. This results in an overall running time of O(n^2). We did add one enhancement, so the greedy algorithm did not get stuck. If it ever has an infinite cost for the path, we just restart at a new arbitrary start city. There is no complex data structure used within the greedy algorithm, so the space complexity is simply the space needed to store the problem, which is O(n).

# Fancy Algorithm

For our fancy algorithm we took a local-search-based approach. To fulfill local search’s requirement of always having an initial route, we had to design a better greedy algorithm that would always arrive at a solution.

The updated greedy algorithm works similarly to the basic greedy algorithm described above but includes backtracking. It keeps track of which routes have been tried from a node in a n by n table (n is the number of cities) and tries new paths in order of shortest distance. If there is not a valid complete tour, this algorithm will try every possible route.

It has space complexity of O(n2), best-case time complexity of O(n2), and worst-case time complexity of O(n!\*n2). It is possible to reduce the worst-case complexity to O(n!\*n) by first checking for a valid path without taking the time to compare distances, but the graphs we’re working with are sufficiently dense that we never found one without at least one complete tour. We also felt it was best to leave that feature out because for best-case runs, which are common, it doubles the time taken.

We spent a decent amount of time optimizing the updated greedy algorithm, and it finds routes for around 7000 cities within the 60 second maximum time. City counts below 900 almost always take less than a second to process.

<http://www.eng.uwaterloo.ca/~sjayaswa/projects/MSCI703_project.pdf>

# Results Table

# Discussion of Table

We took the average of 5 trials for city sizes of 15, 30, 60, 100, 200, 500, 100, 1500, and 2000 for each algorithm (Random, Greedy, Branch and Bound, and Local Search). Of all the algorithms, Greedy was by far the fastest across all city sizes. Greedy finished with an average run time of 10 seconds for 2000 cities. The only other algorithm that could finish a solution before the 10-minute limit was the Local Search algorithm with an average run time of 424 seconds. Nonetheless, the Local Search algorithm had the best average cost of tour for each tried city size. The Local Search algorithm on average had a cost of tour score 6% lower than Greedy. Therefore, Local Search is the more effective algorithm at finding the best minimum cost solution while Greedy finds good, low cost solutions quickly.