Part 1: Questions on Complexity

a) What is the complexity class of the TSP? (find this via a literature search. Cite your source in the literature. Please give authors' names, name of paper/book chapter, year published, and a link if available.)

The Traveling Salesman Problem is classified as NP-Hard.

Brucato, Corinne. The traveling salesman problem. Diss. University of Pittsburgh, 2013.

The Traveling Salesman Problem (pitt.edu)

b) What is the largest possible number of tours on n cities? Support your answer with a proof. (Hint: since we are looking for the largest possible number of tours, you will have to assume that every city is connected to every other city, since having more roads increases the number of tours)

For one city, there is only one route, since the city started in is the city you end in for the entire "tour."

For two cities, there are now two routes, since we can start in either city and travel to the other one and return to the original.

For 3+ cities, we can now create a formula to represent the maximum number of tours.

We can start in n possible cities. Since we assume every city is connected to another city, we can travel to $n - 1$ cities, and from there we can travel to $n - 2$ cities, and so on until we ultimately reach all n cities and return to the starting one. This gives us the following result.

$$n!$$

However, a tour is defined by a cycle in the graph. If we consider each permutation of the same cycle to be the same tour, then there are n possible rotations of a tour (Ex. $c_1, c_2, c_3, \ldots, c_n$ is the same as $c_n, c_1, c_2, \ldots, c_{n-1}$ and $c_{n-1}, c_n, c_1, \ldots, c_{n-2}$). Similarly if we consider every permutation in reverse to also be the same tour, we should divide the equation found earlier by $2n$.

$$n! / 2n = (n - 1)! / 2$$

So the largest possible number of unique tours on n cities is $(n - 1)! / 2$

Part 2: Data Structure

d) Given a tour generated from matrix M, how do you compute its cost? (If this question feels easy, it is)

Since every road that exists between two cities i and j is given in the matrix as $M(i, j)$ and we are given an ordered list of n cities $[c_1, c_2, c_3, ..., c_n]$, we can sum the following obtained from the matrix: $M(c_1, c_2) + M(c_2, c_3) + M(c_3, c_4) + ... + M(c_{n-1}, c_n) + M(c_n, c_1)$ to get the cost of the tour.

e) Just in case your algorithm gets a matrix M that does not contain any tours (e.g., it is not possible to come back to the starting city 1), it's a good idea to be able to detect that and output some message like "This matrix does not contain any tours" before you attempt to find non-existent tours. How do you detect that?

One possible way for there not to be a complete tour is if a city is not connected to any other city by a road or if the city is only connected to one other city. If the first case were to occur, there is no tour possible since we cannot access all cities in any way. If the second case were to occur, we would have to go through a city twice since there is only one road to one city. We can double check this before performing any computations by traversing each row in the matrix and ensuring that each row has at least two non-negative values.