Yash Hegde
COP4533
Programming Problem Write-up

My program aims to find the minimum path cost from the source station **s** to any number of stations in array **B**, routing a path through every station. This path is calculated via the Bellman-Ford algorithm, which calculates the shortest path distances from any arbitrary **s** to every other station, with the heights in the three-dimensional coordinates being considered as edge weights in the constructed grid. My dynamic programming problem involves visiting all of **B** exactly once, with the recursive equation being the following:

BF(r, B) = min_(t is an element of B) (BF(t,B-B.difference({r})) + cost(t, r))

This recursive solution represents the Optimal Substructure (OPT) that is essentially done in the program, with **BF(r, B)** representing the min cost to reach station **r** from any **s**, with the base case of one station being the cost of two stations routed to each other. **r** is constructed by combining the minimum cost paths from the source station to all previous stations in the permutation. This property allows us to build the solution to the larger problem (finding the minimum cost path) by recursively solving smaller subproblems (finding the minimum cost paths to each station in the permutation). **t** is simply an element of **B**.

As such, the overall time complexity of my program would be **O(V * E * C)**, where **V** and **E** are the vertices and edges respectively. **C** would be the number of stations in array **B** that are gathered from the reconstructed stations graph and further permutations being completed. The initial dynamic programming problem's time complexity of **O(n^2 * 2^n)** where **n** is the number of stations is equivalent to this time complexity, as **V * E** is equivalent to **n^2** in the scope of the problem, where it's possible for denser graphs to exist, while **2^n** is equivalent to **C**, as this expression correlates with the number of subsets of **B** with **n** elements.