Elad Ohayon

3/1/15

My algorithm models the problem of determining which segment minimizes the total sum of every vertex-city pair by representing the collection of segments between each city as an undirected edge weighted graph. Afterwards, we run an algorithm to calculate the distance between every vertex pairing through Floyd-Warshalls Algorithm, and then use it determine which segment from possibilities will minimize the total duration. This is a version of the "Transitive Closure" problem, but we also care about graph durations, rather than simply whether a vertex **j** is reachable from **i.**

The "true" brute force solution to this algorithm is to construct a graph for every segment in possibilities, run Warshalls algorithm on said graph, and determine the total duration of the graph by summing up all the weights in the distTo matrix. We then return the segment which generated the graph with the smallest total duration.

My algorithm solves the problem more efficiently as its run time is $V^3 + 2QV^2 + V^2$ compared to brute force's $(E(V^3 + 2V^2)) + V^3 + V^2$. V is the number of cities (vertices in the graph) and Q is qualifying segments. The algorithm works by conducting Warshall-Floydd only on the original graph (as compared to the brute force solution which does it possibilities.length times). It then loops through all the segments and determines if they qualify by comparing the duration of each segment with the distance between x-y computed in Warshall which takes O(1) time. If the duration in Warshall is shorter than the segment, then we ignore the segment as it won't reduce the graph's duration. If the segment's duration is less, we need to update our distTo matrix, my algorithm does this in V^2 time by looping through the distTo matrix and updating it, rather rerunning Warshall's algorithm. As this changes the values in the distTo matrix, we reset it at the end of the iteration which takes V^2 time, to minimize costs, we sum up the duration in the same loop to reduce 2 V^2 operations into a singular V^2. We keep track of which segment has minimized the total cost and return it once we have examined every qualifying segment. This is the segment which minimizes the total cost the most.