Big O Times

StreetMap:

load():

If there are N lines in the mapdata file, the time complexity of load() will be O(N).

getSegmentsThatStartWith():

If there are an average of S segments associated with each GeoCoord, this function will run in O(S) time. However, the number S is likely to be very low because a segment can only have so many different intersections, usually ranging from 1 to 4. Thus, it can also be seen as O(1).

PointToPointRouter

generatePointToPointRoute():

I used what I think was A\*, or a deviation of it. I used a priority queue of tuples to generate my route. The tuple contained a double that held the total distance travelled so far on the route it was building, a list of StreetSegments detailing the route for that specific path, and a GeoCoord that held the destination coordinate. I made my own comparator function for that priority queue such that the shortest route would go to the front of the queue. I also used an ExpandableHashMap mapping StreetSegments to bools to keep track of which StreetSegments I had visited, although the data type it was mapped to didn’t matter.

DeliveryOptimizer:

optimizeDeliveryOrder():

This function runs at O(N^2) assuming there are N deliveries to be delivered.