Alex Chen

CS 32

GooberEats Report

1)

a. bool StreetMap::load(string mapFile)

- if mapFile holds N lines of data, the load function will run in O(N), as each line is visited once, and the processing for each line is of constant time complexity.

b. bool StreetMap::getSegmentsThatStartWith(const GeoCoord& gc, vector<StreetSegment>& segs) const

- The time complexity of the getSegmentsThatStartWith function is dependent upon the ExpandableHashMap’s find function and the assignment operator for vectors. For find, the time complexity in terms of GeoCoords visited is constant on average because searching through the ExpandableHashMap is searching by index, which is constant, and then searching through the linked list in that bucket, which is constant on average. Using the assignment operator for vectors is O(S), if S is the number of elements in the vector. However, the number of elements in the vector is constant compared to the number of lines in the data file. Thus, the overall time complexity is O(1).

2)

a. DeliveryResult PointToPointRouter::generatePointToPointRoute(

const GeoCoord& start,

const GeoCoord& end,

list<StreetSegment>& route,

double& totalDistanceTravelled) const

* If there are N street segments in the mapping data, then in the worst case, my generatePointToPointRoute will run in O(N) time, if all street segments are visited before the final destination is reached. However, because I implemented an A\* algorithm in which the GeoCoords that are generally closer to the final destination are prioritized first, my function runs in much less than O(N) time. In terms of big O however, because coefficients are involved, my function’s time complexity is still considered O(N).

3)

a. void DeliveryOptimizer::optimizeDeliveryOrder(

const GeoCoord& depot,

vector<DeliveryRequest>& deliveries,

double& oldCrowDistance,

double& newCrowDistance) const

* In the optimizeDeliveryOrder function, there are a few factors for time complexity. Assume N is the number of deliveries that need to be made. In calculating each the old and new crow distance, the time complexity is linear as each delivery must be reached (plus the depots). In actually optimizing the delivery order, I used the Greedy Man’s algorithm, which reorders the deliveries by whichever GeoCoord is closest to the current position. This runs in O(N^2) time complexity because for each delivery location, there are N, and then N-1, and then N-2, and so on delivery locations to process. For each delivery location, after comparing it to the rest of the delivery locations left, one delivery location will be pushed back and also erased from a vector. Pushing back to a vector is constant (amortized time), while erasing from a vector is linear on average. Altogether, this O(N^2), on average. Linear and constant time complexities are insignificant.