Report for Project 4:

MyMap:

1. Associate():
   1. Log(n).
      1. The only two functions to consider for the Big Oh are search and insert. Both are standard functions for moving through and inserting in a binary search tree, which is Log(n). Also, since they are not nested, they are separate so if anything it would be 2Log(n) but we drop any coefficient. N represents the number of nodes in our tree.
2. Find():
   1. Log(n).
      1. For find, we are only utilizing the search function, which as we mentioned before is only log(n). N represents the number of nodes in our BST.

Attraction Mapper:

1. Init():
2. N \* A(logA ).
   1. The for loop on top means we are looping through all segments, which we denote as N. The GetSegment function works in constant time so it will not be considered. Then, we have to loop through all the attractions. On average, we believe there are A attractions so we will multiply N by A as we are going through all attractions of all segments. Furthermore, we utilize the associate function, which works in log(A) time. Therefore the big oh is N\*A\*log(A).
3. getGeoCoord:
   1. logA
      1. For getGeocoord, all we are doing is utilizing the find function of our MyMap, which works in log(n) time. For this setup, we have A attractions, so the big oh is logA.

SegmentMapper:

1. Init(): n(Log(n) + A(log(A)))
   1. Since we are looping through all street segments, which we assume is N, we have n there. Then we have the log N as we are using the find and associate functions which we established as log(n). Then we are iterating through all the attractions, hence the A, and associating them, hence the log(A).
2. getSegments(): log(N)
   1. Since I only have to search through the BST, we established that as log(N), which was the size.

Navigator:

1. (N+A) \* Log(N+A);
   1. The big oh will mostly be determined by the while loop that loops at the temp and its successors. First it will loop through the node vector, which has a big Oh of N. Then it will be inserting values with associate which works at log(n). Then it will set the directions vector which will be iterating through the attractions and the nodes to ensure that we will be hitting either end of the end coordinate segment. Hence (a+n)\* Log(a+n).