MyMap::associate –

associate makes use of a recursively implemented insert function. Each time insert is called, half the tree is eliminated, so associate is O(log(N)) where N is the number of items in the tree.

MyMap::find –

find makes use of a recursively implemented helper function. Each time the helper function is called, half the tree is eliminated, so find is O(log(N)) where N is the number of items in the tree.

AttractionMapper::init –

init visits each of the N street segments (N). If the street segment has attractions, it adds each attraction to a map (log(A) for an insertion, A\*log(A) for all insertions). Visiting N street segments and adding A attractions into a BST results in a time complexity of O(N+A\*log(A)).

AttractionMapper::getGeoCoord –

getGeoCoord uses MyMap::find, so it has a time complexity of O(log(A)) where A is the number attractions.

SegmentMapper::init –

init adds 2 GeoCoords to a map for every street segment and 1 GeoCoord to a map for every attraction. If there are N street segments and A attractions, then the time complexity is O((N+A)\*log(N+A)).

SegmentMapper::getSegments –

getSegments uses MyMap::find, so it has a time complexity of O(log(N+A)), where N is the number of street segments and A is the number of attractions.

Navigator::navigate –

Navigate uses the A\* algorithm with a constant time heuristic. The algorithm’s while loop goes through at most N segments + 2 attractions. For each, neighbors are retrieved using the getSegments function, which has complexity of O(log(N+A)). While there are other loops and function calls within the navigate function, these have smaller or equal complexity or are guaranteed to be small when working with map data, so the time complexity will be O(N\*log(N+A)).