Name: Matthew Wen
Login: wen101
/**
 * Explain your overall approach to the problem and a short
 * general summary of your solution and code.
 **/
For reading adjacent nodes, it first creates each node, which takes O(|N|) because while it reads the input file, it already know where to place it inside an array that stores all the node. When storing the edges, it is O(|E|) for directly adjacent node, and it stores the adjacent edges inside each edges. For not directly specified adjacent edge cases, it takes O(|E|), because for each edge cases, it is located directly at the node, then check all current edge cases to determine to add that specified test case or not. Because it is assumed that the number of edge cases stored at each node is small, for best case, it is O(|E|). But worst case, it is O(|E|*|E|). For using dijkstra algorithm, I used a min heap to organized which nodes is visited. From the start, I get the smallest node distance from the start, and then append its edge nodes with setting their distance. In appending a node to the heap, I check if the node is already inside the heap, then append the element, which takes n + nlog(n), leaving the big O being O(nlogn) with n being the number of elements in the heap. For edge cases where a better distance is found, I run it through appending it, but instead of allocating more memory for it, it will update its current position of the heap based off its new distance. This process will continue until the get min function, which gets the smallest distance inside the min heap is equal to the end node, or destination. Performance is 0.138 seconds for usa.txt, and the usa100.txt query.
/**
 * Known bugs / limitations of your program / assumptions made.
 **/
The assumptions I made is that the coordinates are not greater than the greatest value that can be stored inside an int data type depending on which architecture it can run on. In addition, the sum of the coordinates should be less than the greatest value of an integer data type depending on the artitecture it runs on.
/**
 * List whatever help (if any) that you received.
 **/
I looked at YouTube where they recommended using a min heap, and then recording things to be aware about certain edge cases that could happen; specifically when there is a better distance to a node by an alternative path.
/**
 * Describe any serious problems you encountered.
 **/
Some serious issues I ran into is trying to reach the memory threshold. I had to mess around with different data types to store the direct and nondirect edge cases to find all the adjacent nodes. After testing most of the different data types to store the edge cases, I ended up giving up at a total heap of 6.3MB instead of the required 4.9MB for usa.txt.
/**
 * List any other comments/feedback here
 **/
I think the sample test cases should contain no bugs, where there are no repeated nodes and no weird edge cases. I also think that should be no repeated edge cases, and this should be expected as an assumption that the programmer can make. Because these assumptions cannot be made, it made the assignment too tough, and therefore increased the big O complexity when implementing the algorithm.