COL106 Assignment 3 Regrade Request

**Complexity of PointDatabase Object Creation:**

1. The given pointlist is sorted—O(nlogn)
2. A balanced Binary Search Tree is created, ordering the points according to their x coordinates.

As L is sorted, we can make such a tree in O(n) time as follows:

1. Select the median of the list and make it the root
2. Recursively do the same for left and right sublists making them the left and right subtrees respectively

Let T1(n) be the time taken for a list of size n. Then

Where O(logn) is the time taken for the insertion step. On solving this recursion we obtain that T1(n) is O(n).

1. For every node of the above tree, a corresponding YTree is created. It is done recursively for any node by merging the two YTrees of its children and the node itself. Let T(n) be the time taken for creating all the YTrees for a tree of size n. Then

Where O(n) is the time taken to merge the two trees (This is done by merging the level order traversals of both the trees using a algorithm similar to the merge function of mergesort, which takes O(sum of sizes of both trees) time).

On solving this recursion, we obtain T(n) ~ O(nlogn)

Thus, the total time taken is O(nlogn)+O(n)+O(nlogn)=**O(nlogn)**

**Complexity of searchNearby(q,d)**

First, the correct range of x is found. To do this, find the corresponding node for the maximum possible and minimum possible x-coordinates of the tree (O(logn) time), and their lowest common anscestor(lca) (O(logn) time). Now starting from the minimum x-coordinate node:

1. If the node’s y coordinate falls in the range, append it to the answer list. If it has a right YTree, search for the corresponding range of y in a similar way and append the answers to the answer list.
2. If the node is the left child of its parent then follow the same process for the parent node. Move above to the anscestors till the lca.

A similar process is followed for the maximum x-coordinate node, but with the left and right flipped.

There may be O(log n) ancestors on the way to the lca and for each such node, a similar search on the YTree takes O(logn) time, so total complexity becomes O((logn)2). Further, appending the m answers to the answer list takes O(m) time so net complexity becomes **O((logn)2+m)**