Assignment 5 Problem 1

a) Draw a 2-dimensional range tree of minimal height for the following set of points:

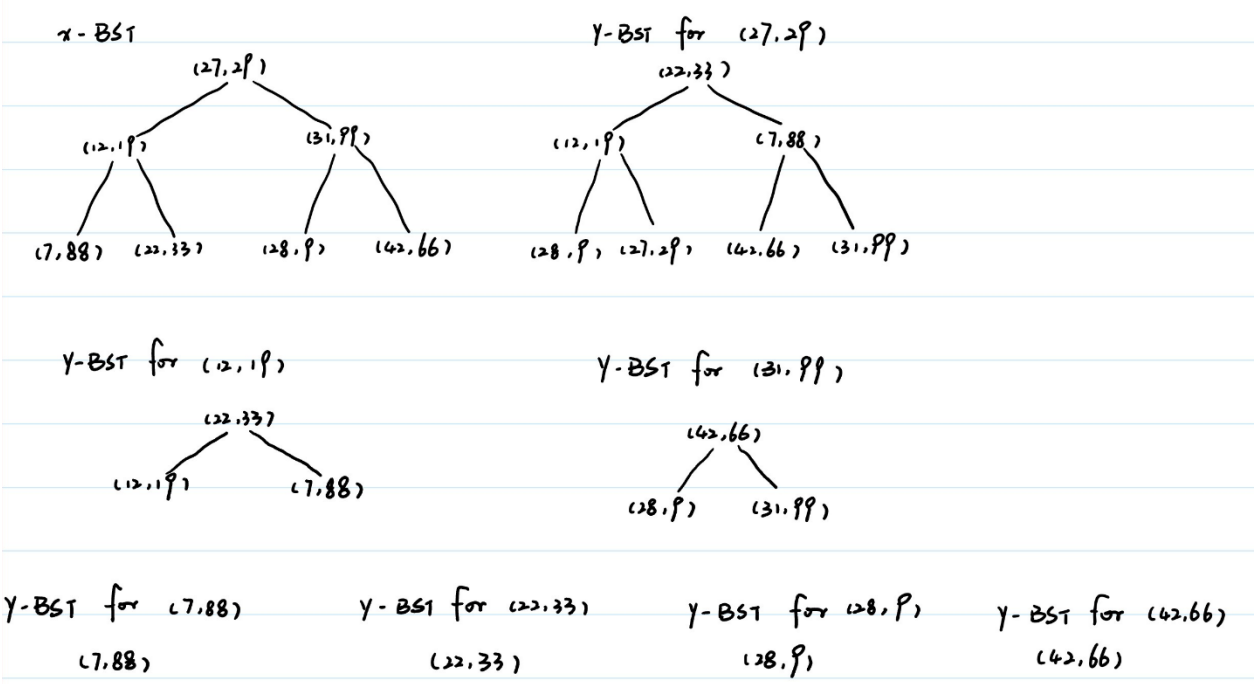
$\{(7, 88), (12, 19), (22, 33), (27, 29), (28, 9), (31, 99), (42, 66)\}$

b) Suppose a two dimensional range tree data structure stores n points, and that the BST ordered by x-coordinates is perfect, i.e., every level is completely filled. Give an exact closed form formula in terms of n for the sum of the number of nodes in the x-ordered BST plus the total number of nodes in all y-ordered BSTs.

c) Assume that we have a set of n numbers (not necessarily integers) and we are interested only in counting the number of points that lie in a range rather than in reporting all of them. Describe how a 1-dimensional range tree (i.e., a balanced BST) can be modified such that a range counting query can be performed in $O(\log n)$ time (independent of s). Briefly justify that your algorithm is within the expected runtime.

d) Next, consider the 2-dimensional case where we have a set of n 2-dimensional points. Given a query rectangle R, we only want to find the number of points inside R, not the points themselves. Explain how to modify the Range Tree data structure and the search algorithm such that counting queries can be performed in $O((\log n)^2)$ time. Briefly justify that your algorithm meets the runtime requirement.

a)



b) For x-ordered BST, there are n nodes. For root of the level 0 of the BST, every node (only root here) contains a BST with n nodes, therefore, for level 0, there are $n_0 = 1 \times n$ nodes in the y-ordered BST in total. For level 1, there are $n_1 = n_0 - 1$ nodes in the y-ordered BST in total. For level 2, there are $n_2 = n_1 - 2$ nodes in the y-ordered BST in total. Etc. For level k, there are $n_{k-1} - 2^{k-1}$

nodes in the y-ordered BST in total. Therefore, the total number of nodes in the y-ordered BSTs are $n + (n-1) + (n-2-1) + \cdots + \left(n - \sum_{i=0}^{k-1} 2^i\right)$
Note that since the BST ordered by x-coordinates is perfect, we have

$$k = log(n+1) - 1$$

Therefore, the total number of nodes in the range tree is

$$n + n + (n-1) + (n-3) + \cdots + \left(n - (2^k - 1)\right) = (2+k)n - (2^{k+1} - 2 - k)$$
$$= 2n + 2 + (n+1)k - 2 * 2^k = (n+1)log(n+1)$$

c) We can more information in each nodes in the range tree. For node N, we store the number of nodes that the subtree rooted at N contains. By doing this, when we doing counting, we do the similar thing as BST range search, find the boundary nodes, and the topmost inside nodes. From lecture, we know that the number of both boundary nodes and topmost inside nodes are in O(log n). For each boundary nodes, check if it is in the range provided, if yes, we add it to count, this operation cost O(1). For each topmost inside nodes, we just add the number of nodes the subtree rooted at such node contains (i.e. the new information we stored in the tree for such node) to the counting variable, also this operation is in O(1). Therefore, the expected running time is O(log n)

d) Similar to the part c, we also add the same information for the y-ordered BST inside each node of the x-ordered BST. When doing counting, we do the similar thing as range search on 2d-range tree. For the x-coordinates, we can find the boundary nodes and topmost inside nodes. As mentioned in the lecture, the number of them are both in O(log n). For the boundary nodes, we add it to count if it is in range (for each node, this cost O(1)). For the topmost inside nodes, we did exactly what we did in c) but this time with y-coordinates. From c) the running time is O(log n) for each topmost inside nodes. Therefore, the total running time is O(log n) + O(log n)O(log n) = $O((logn)^2)$