Range tree construction.

1. If we can construct a \( d - 1 \)-dimensional range tree in \( O(n \log^{d-2} n) \) time, then we can easily construct a \( d \)-dimensional range tree in \( O(n \log^{d-1} n) \) time. To do this, we construct the tree from the root down. First, construct a \( d - 1 \) dimensional range tree over all elements for the root, then divide on the \( d \)th dimension and pass down the relevant elements to each subtree and recursively build them. The total running time is then

\[
T(n, d) = 2T(n/2, d) + T(n, d - 1) = 2T(n/2, d) + O(n \log^{d-2} n)
\]

which solves to \( O(n \log^{d-1} n) \) by the Master theorem. All that remains then, is to show how to construct a 2-dimensional range tree in \( O(n \log n) \) time. To do this, first sort the elements by \( y \) in \( O(n \log n) \) time. We then build the tree over \( x \) as before, except since the elements are sorted by \( y \) (which we can easily maintain as we work down the tree) the time to build each balanced tree over \( y \) is \( O(n) \), for a total of

\[
T(n, 2) = 2T(n/2, 2) + T(n, 1) = 2T(n/2, 2) + O(n) = O(n \log n).
\]

2. The procedure for a constructing a layered range tree is identical to the previous section, except instead of building the tree over \( y \) in \( O(n) \) time, we must instead compute the pointers between adjacent layers in \( O(n) \) time. This can be done by with a merge-like operation on the child and parent arrays. Keep a pointer to the head of each array, if the \( y \) value at the child head is less than the \( y \) value at the parent head, then increment the child head. Otherwise, the parent head points to the child head and we increment the parent head.