1. Suppose we are given a set S of n line segments in the plane, each of which is either horizontal or vertical. Each horizontal segment $h \in S$ is specified by its left x-coordinate $h.l$, its right x-coordinate $h.r$, and its y-coordinate $h.y$. Each vertical segment $v \in S$ is specified by its x-coordinate $v.x$, its bottom y-coordinate $v.b$, and its top y-coordinate $v.t$. Assume that all x- and y-coordinates are distinct.

   Describe and analyze an algorithm to compute the number of pairs of segments in S that intersect. (Because all coordinates are distinct, if two segments in S intersect, one must be horizontal and the other vertical.)

   [Hint: You can do better than blindly applying Homework 9.]

Solution (Blindly applying HW9): Partition the input segments S into horizontal segments H and vertical segments V, preprocess H into either of the data structures described in the HW9 solutions, and then perform a CrossCount query for each segment in V using the corresponding query algorithm.

For both data structures, the total time to build the data structure is

$$\sum_u O\left(|H_u| \log |H_u|\right) \le \left(\sum_u |H_u|\right) \cdot O(\log n) = O(n \log^2 n),$$

and the total time to perform at most n CrossCount queries is at most $n \cdot O(\log^2 n)$ $= O(n \log^2 n)$. So the overall algorithm runs in ***$O(n \log^2 n)$ time***. ■

Solution (Sweep with dynamic range tree): Partition the input segments S into horizontal segments H and vertical segments V.

We count segment intersections using a sweepline algorithm. Intuitively, we sweep a vertical line ℓ from left to right across the segments, maintaining the y-coordinates of the intersection $S = \ell \cap H$ in a one-dimensional range tree. Whenever ℓ touches the left endpoint of a horizontal segment $h \in H$, we insert $h.y$ into the range tree; when ℓ touches the right endpoint of a horizontal segment $h \in H$, we delete $h.y$ from the range tree; when ℓ reaches a vertical segment $v \in V$, we perform a range query for the range $[v.b, v.t]$.

Our description of range trees as *exogenous* binary search trees (for example, in the HW7 solutions) did not include insertion and deletion algorithms. There are at least two ways to make range trees dynamic:

- Use a standard (endogenous) balanced binary search tree, where each node u stores the number of nodes $u.size$ in its subtree. We have already seen how to use such a tree tree to answer Rank queries—How many points in S are less than a query value q?—in $O(\log n)$ time, and how to maintain the tree in $O(\log n)$ time per iteration (using, for example, the AVL balancing algorithms). The number of points in S in any range $[v.b, v.t]$ is exactly $\text{Rank}(v.t) - \text{Rank}(v.b)$.
- Store the y-coordinates of *all* horizontal segments in the leaves of a *static* balanced binary tree. Additionally, give each segment $h \in H$ a *weight*, which is 1 if h intersects ℓ and 0 otherwise; each node u in the range tree stores the total

weight of its leaves in a new field *u.totalwt*.

- When the sweepline ℓ touches a vertical segment $v \in V$, we use the range tree to decompose v into $O(\log n)$ canonical segments, and we return the sum of *u.totalwt* over the $O(\log n)$ corresponding nodes, in $O(\log n)$ time. (See the MINIMUM algorithm in the HW7 solutions.)
- When the sweepline ℓ touches either endpoint of a horizontal segment $h \in H$, we change the weight of h and update *u.totalwt* for every ancestor of h in the tree, in $O(\log n)$ time. (See the SHIFT algorithm in the HW7 solutions.)

Using either of these strategies, the overall algorithm runs in $\boldsymbol{O(n \log n)}$ ***time***. ■