1. Suppose we are given a set $H$ of $n$ horizontal line segments in the plane, each specified by its left $x$-coordinate $h.l$, its right $x$-coordinate $h.r$, and its $y$-coordinate $h.y$. Describe a data structure for $H$ that supports queries of the following form:

   - CrossCount($v$): Given a vertical line segment $v$, specified by its $x$-coordinate $v.x$, its bottom $y$-coordinate $v.b$, and its top $y$-coordinate $v.t$, return the number of horizontal segments in $H$ that intersect $v$.

**Solution (segment-range tree):** The query segment $v$ crosses a horizontal segment $h \in H$ if and only if both of the following conditions are satisfied:

- The $x$-projection of $h$ contains the $x$-coordinate of $v$; that is, $h.l < v.x < h.r$.
- The $y$-coordinate of $h$ stabs the $y$-projection of $v$; that is, $v.b < h.y < v.t$.

We build a two-layer data structure, where each layer is responsible for one of these two conditions.

Specifically, we first build a *segment* tree $T$ over the $x$-projections $[h.l, h.r]$ of segments $h \in H$. For each node $u$ in $T$, let $H_u$ denote the corresponding subset of segments (that is, every segment in $H$ whose canonical partitions include the $x$-range of $u$). For each node $u$ in the primary segment tree, we construct a secondary *range* tree $T_u$ over the $y$-coordinates $h.y$ of segments $h \in H_u$.

The primary segment tree uses $O(n)$ space. For each node $u$, the secondary range tree $T_u$ uses $O(|H_u|)$ space. Each segment in $H$ appears in at most two canonical subsets $H_u$ at each level of the primary range tree, and therefore at most $O(\log n)$ canoncial subsets overall, so $\sum_u |H_u| = O(n \log n)$. Thus, in total, our two-level data structure uses $\sum_u O(|H_u|) = \boldsymbol{O(n \log n)}$ ***space***.

To implement CrossCount($v$), we first find the search path $u_0, u_1, \ldots, u_k$ in the primary segment tree for the $x$-coordinate $v.x$. Here, $u_0$ is the root of the segment tree, and each node $u_{i+1}$ is one of the children of $u_i$. Then for each node $u_i$, we perform a range query for the $y$-projection $[v.b, v.t]$ in the secondary range tree $T_{u_i}$. Finally, we return the sum of the results from all secondary range queries.

If any segment $h \in H$ appeared in two subsets $H_{u_i}$ and $H_{u_j}$, then the canonical partition of $h$ would contain two overlapping intervals, which is impossible. On the other hand, if $h$ does not appear in any subset $H_{u_i}$, then $h$ does not cross the query segment $v$. Thus, each segment $h \in H$ that intersects $v$ is an element of *exactly one* subset $H_{u_i}$. In other words, each secondary range query counts each segment $h \in H$ that crosses $v$ exactly once.

Altogether, we perform $O(\log n)$ secondary range queries. For each node $u$ in the primary range tree, a range query in the corresponding subset $H_u$ takes $O(\log |H_u|) = O(\log n)$ time. Thus, in total, CrossCount runs in $\boldsymbol{O(\log^2 n)}$ ***time***. ■

**Solution (range-segment tree):** The query segment $v$ crosses a horizontal segment $h \in H$ if and only if both of the following conditions are satisfied:

- The $y$-coordinate of $h$ stabs the $y$-projection of $v$; that is, $v.b < h.y < v.t$.
- The $x$-projection of $h$ contains the $x$-coordinate of $v$; that is, $h.l < v.x < h.r$.

We build a two-layer data structure, where each layer is responsible for one of these two conditions.

Specifically, we first build a *range* tree over the $y$-coordinates $h.y$ of segments $h \in H$. For any node $u$ in this range tree, let $H_u$ denote the corresponding canonical subset of segments (those whose $x$-coordinates are stored in leaves in subtree rooted at $u$). For each node $u$ in the primary range tree, we build a secondary segment tree $T_u$ over the $x$-projections $[h.l, h.r]$ of segments $h \in H_u$.

The primary range tree uses $O(n)$ space. For each node $u$, the secondary segment tree $T_u$ uses $O(|H_u|)$ space. Each segment in $H$ is an element of exactly one canonical subset $H_u$ at each level of the primary range tree, so $\sum_u |H_u| = O(n \log n)$. Thus, in total, our two-level data structure uses $\sum_u O(|H_u|) = \boldsymbol{O(n \log n)}$ ***space***.

To implement CrossCount($v$), we first partition the query segment $v$ into $O(\log n)$ canonical segments $v_1, v_2, \ldots, v_k$, each associated with a node in the primary range tree, by performing a binary search for the endpoint coordinates $v.b$ and $v.t$. Then for each canonical segment $v_i$, we perform a stabbing query for $v_i.x = v.x$ in the corresponding canonical subset of segments $H_i$, using the corresponding secondary structure. Finally, we return the sum of the results from all secondary stabbing queries.

The canonical segments $v_i$ defined a partition the query segment $v$. Thus, any horizontal segment $h \in H$ that stabs $v$ stabs *exactly one* of the canonical segments $v_i$. It follows that our secondary stabbing queries count each segment $h \in H$ that crosses $v$ exactly once.

Altogether, we perform $O(\log n)$ secondary stabbing queries. For each node $u$ in the primary range tree, a stabbing query in the canonical subset $H_u$ takes $O(\log |H_u|) = O(\log n)$ time. Thus, in total, CrossCount runs in $\boldsymbol{O(\log^2 n)}$ ***time***. ■