Windowing Queries

Given a set $S$ of $n$ disjoint line segments in the plane.

Store $S$ in a data structure s.t. given a query rectangle $R$, we can find the segments in $S$ intersecting $R$ efficiently.
Windowing Queries

Given a set $S$ of $n$ disjoint line segments in the plane.

Store $S$ in a data structure s.t. given a query rectangle $R$, we can find the segments in $S$ intersecting $R$ efficiently.
Windowing Queries

Given a set $S$ of $n$ disjoint line segments in the plane.

Store $S$ in a data structure s.t. given a query rectangle $R$, we can find the segments in $S$ intersecting $R$ efficiently.

The segments that intersect $R$

1) have an endpoint in $R$, or

2) intersect the boundary of $R$. 

Windowing Queries

Given a set $S$ of $n$ disjoint line segments in the plane.

Store $S$ in a data structure s.t. given a query rectangle $R$, we can find the segments in $S$ intersecting $R$ efficiently.

The segments that intersect $R$

1) have an endpoint in $R$, or
   find them using a range query with $R$ on the set of end points

2) intersect the boundary of $R$. 
Windowing Queries

Given a set $S$ of $n$ disjoint orthogonal line segments in the plane.

Store $S$ in a data structure s.t. given a query rectangle $R$, we can find the segments in $S$ intersecting $R$ efficiently.

The segments that intersect $R$

1) have an endpoint in $R$, or
   find them using a range query with $R$ on the set of end points

2) intersect the boundary of $R$. 
Windowing Queries

Given a set $S$ of $n$ disjoint horizontal line segments in the plane.

Store $S$ in a data structure s.t. given a vertical query segment $q$, we can find the segments in $S$ intersecting $q$ efficiently.
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query value $q$, we can find the intervals in $S$ intersecting $q$ efficiently.
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query value $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query value $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$

$T$ is a balanced BST on the endpoints

the root of the tree (the median endpoint) $v$

stores the intervals $I(v)$ that contain $v$
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query value $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$

$T$ is a balanced BST on the endpoints

the root of the tree (the median endpoint) $v$ stores the intervals $I(v)$ that contain $v$

The left subtree $\ell$ of $v$ stores the intervals that lie completely left of $v$. 
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query value $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$

$T$ is a balanced BST on the endpoints

the root of the tree (the median endpoint) $v$
stores the intervals $I(v)$ that contain $v$

The left subtree $\ell$ of $v$ stores the intervals that lie completely left of $v$.

The right subtree $r$ of $v$ stores the intervals that lie completely right of $v$. 
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query value $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$

$T$ is a balanced BST on the endpoints

the root of the tree (the median endpoint) $v$
stores the intervals $I(v)$ that contain $v$

store these intervals twice:
1) sorted on increasing left endpoint
2) sorted on decreasing right endpoint
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query value $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$

$T$ is a balanced BST on the endpoints

the root of the tree (the median endpoint) $v$ stores the intervals $I(v)$ that contain $v$

$\text{QUERY}(q, T)$
if $q$ left of $v$ then
  report intervals from $I(v)$ using the list of left-end points, stop at the first interval right of $q$.
$\text{QUERY}(q, \ell)$
else if $q$ right of $v$
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query value $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$

Space usage:

Query time:

Preprocessing time:
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query value $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$

Space usage: $O(n)$

Query time:

Preprocessing time:
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query value $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$

Space usage: $O(n)$

Query time: $O(\log n + k)$

$k = \#\text{intervals reported}$

Preprocessing time:
Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query value $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$

Space usage: $O(n)$
Query time: $O(\log n + k)$

$k = \#\text{intervals reported}$

Preprocessing time: $O(n \log n)$
Segment Stabbing Queries

Given a set $S$ of $n$ disjoint horizontal line segments in the plane.

Store $S$ in a data structure s.t. given a vertical query segment $q$, we can find the segments in $S$ intersecting $q$ efficiently.
Segment Stabbing Queries

Given a set $S$ of $n$ disjoint horizontal line segments in the plane.

Store $S$ in a data structure s.t. given a vertical query segment $q$, we can find the segments in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$

$T$ is a balanced BST on the endpoints

the root of the tree (the median endpoint) $v$
stores the intervals $I(v)$ that contain $v$
Segment Stabbing Queries

Given a set $S$ of $n$ disjoint horizontal line segments in the plane.

Store $S$ in a data structure s.t. given a vertical query segment $q$, we can find the segments in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$

$T$ is a balanced BST on the endpoints

the root of the tree (the median endpoint) $v$

stores the intervals $I(v)$ that contain $v$
Segment Stabbing Queries

Given a set $S$ of $n$ disjoint horizontal line segments in the plane.

Store $S$ in a data structure s.t. given a vertical query segment $q$, we can find the segments in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$

$T$ is a balanced BST on the endpoints

the root of the tree (the median endpoint) $v$ stores the intervals $I(v)$ that contain $v$

store these intervals twice:

1) a range tree on their left endpoints
2) a range tree on the right endpoints
Segment Stabbing Queries

Given a set $S$ of $n$ disjoint horizontal line segments in the plane.

Store $S$ in a data structure s.t. given a vertical query segment $q$, we can find the segments in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$

Space usage:

Query time:

Preprocessing time:
Segment Stabbing Queries

Given a set $S$ of $n$ disjoint horizontal line segments in the plane.

Store $S$ in a data structure s.t. given a vertical query segment $q$, we can find the segments in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$

Space usage: $O(n \log n)$

Query time:

Preprocessing time:
Segment Stabbing Queries

Given a set \( S \) of \( n \) disjoint horizontal line segments in the plane.

Store \( S \) in a data structure s.t. given a vertical query segment \( q \), we can find the segments in \( S \) intersecting \( q \) efficiently.

We store \( S \) in an interval tree \( T \)

Space usage: \( O(n \log n) \)

Query time: \( O(\log^2 n + k) \)

\( k = \#\text{intervals reported} \)

Preprocessing time:
Segment Stabbing Queries

Given a set $S$ of $n$ disjoint horizontal line segments in the plane.

Store $S$ in a data structure s.t. given a vertical query segment $q$, we can find the segments in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$

Space usage: $O(n \log n)$

Query time: $O(\log^2 n + k)$

$k = \#\text{intervals reported}$

Preprocessing time: $O(n \log n)$
Segment Stabbing Queries

Given a set $S$ of $n$ disjoint horizontal line segments in the plane.

Store $S$ in a data structure s.t. given a vertical query segment $q$, we can find the segments in $S$ intersecting $q$ efficiently.

We store $S$ in an interval tree $T$

Space usage: $O(n)$

Query time: $O(\log^2 n + k)$

$k = \#\text{intervals reported}$

Preprocessing time: $O(n \log n)$
Segment Stabbing Queries

Given a set $S$ of $n$ disjoint line segments in the plane.

Store $S$ in a data structure s.t. given a vertical query segment $q$, we can find the segments in $S$ intersecting $q$ efficiently.

Our solution using an interval tree + priority search tree no longer works.
Segment Stabbing Queries

Given a set $S$ of $n$ disjoint line segments in the plane.

Store $S$ in a data structure s.t. given a vertical query segment $q$, we can find the segments in $S$ intersecting $q$ efficiently.

Split the problem into elementary intervals in which a vertical line intersects the same segments.
Segment Stabbing Queries

Given a set $S$ of $n$ disjoint line segments in the plane.

Store $S$ in a data structure s.t. given a vertical query segment $q$, we can find the segments in $S$ intersecting $q$ efficiently.

Split the problem into elementary intervals in which a vertical line intersects the same segments.

Storing all segments in all elementary intervals uses $\Theta(n^2)$ space.
Segment Stabbing Queries

Given a set $S$ of $n$ disjoint line segments in the plane.

Store $S$ in a data structure s.t. given a vertical query segment $q$, we can find the segments in $S$ intersecting $q$ efficiently.

Split the problem into elementary intervals in which a vertical line intersects the same segments.

Project the segments onto the $x$-axis, yielding intervals. We build a different data structure for interval stabbing.
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query point $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

Split the problem into elementary intervals in which a vertical line intersects the same segments.

Store the elementary intervals as leaves in a balanced BST $T$. 
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query point $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

Split the problem into elementary intervals in which a vertical line intersects the same segments.

Store the elementary intervals as leaves in a balanced BST $T$.

Every node $v$ corresponds to an interval $I_v$, which is the union of the elementary intervals stored in its subtree.
Interval Stabbing Queries

Given a set \( S \) of \( n \) intervals in \( \mathbb{R}^1 \)

Store \( S \) in a data structure s.t. given a query point \( q \), we can find the intervals in \( S \) intersecting \( q \) efficiently.

Split the problem into elementary intervals in which a vertical line intersects the same segments.

Store the elementary intervals as leaves in a balanced BST \( T \).

Every node \( v \)
corresponds to an interval \( I_v \), which is the union of the elementary intervals stored in its subtree.

stores a canonical subset \( S(v) \subseteq S \) of intervals s.t. \( s \in S(v) \) if and only if \( I_v \subseteq s \) but \( \text{parent}(v) \cdot I \nsubseteq s \).
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query point $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

Split the problem into elementary intervals in which a vertical line intersects the same segments.

Store the elementary intervals as leaves in a balanced BST $T$.

Every node $v$

corresponds to an interval $I_v$, which is the union of the elementary intervals stored in its subtree.

stores a canonical subset $S(v) \subseteq S$ of intervals s.t. $s \in S(v)$ if and only if $I_v \subseteq s$ but $\text{parent}(v)_1 \not\subseteq s$
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query point $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

Split the problem into elementary intervals in which a vertical line intersects the same segments.

Store the elementary intervals as leaves in a balanced BST $T$.

Every node $v$ corresponds to an interval $I_v$, which is the union of the elementary intervals stored in its subtree.

stores a canonical subset $S(v) \subseteq S$ of intervals s.t. $s \in S(v)$ if and only if $I_v \subseteq s$ but $\text{parent}(v)_I \not\subseteq s$
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query point $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

Split the problem into elementary intervals in which a vertical line intersects the same segments.

Query: find all nodes $v$ s.t. $q \in I_v$, and for each such node report all intervals in $S(v)$.
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query point $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

Split the problem into elementary intervals in which a vertical line intersects the same segments.

Query: find all nodes $v$ s.t. $q \in l_v$, and for each such node report all intervals in $S(v)$.

Query time: $O(\log n + k)$, where $k$ is the output size.
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query point $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

Split the problem into elementary intervals in which a vertical line intersects the same segments.

**Question:** How much storage do we use?
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query point $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

Split the problem into elementary intervals in which a vertical line intersects the same segments.

**Question:** How much storage do we use?

**Claim:** Every interval is stored $O(\log n)$ times; at most twice per level.

$\implies$ space usage is $O(n \log n)$. 

Given a set $S$ of $n$ intervals in $R^1$
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query point $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

Split the problem into elementary intervals in which a vertical line intersects the same segments.

**Question:** How much storage do we use?

**Claim:** Every interval is stored $O(\log n)$ times; at most twice per level.

$\Rightarrow$ space usage is $O(n \log n)$.

**Question:** How do we build $T$?
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query point $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

Split the problem into elementary intervals in which a vertical line intersects the same segments.

**Question**: How much storage do we use?

**Claim**: Every interval is stored $O(\log n)$ times; at most twice per level.

$\implies$ space usage is $O(n \log n)$.

**Question**: How do we build $T$?

Build a BST on the elementary intervals, insert the intervals in $s \in S$ one by one.
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query point $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

Split the problem into *elementary intervals* in which a vertical line intersects the same segments.

**Question:** How much storage do we use?

**Claim:** Every interval is stored $O(\log n)$ times; at most twice per level.

$\Rightarrow$ space usage is $O(n \log n)$.

**Question:** How do we build $T$?

Build a BST on the elementary intervals, insert the intervals in $s \in S$ one by one.

To insert $s$ we visit at most 4 nodes per level
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query point $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

We store $S$ in an segment tree $T$

Space usage: $O(n \log n)$

Query time: $O(\log n + k)$

$k = \#\text{intervals reported}$

Preprocessing time: $O(n \log n)$
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query point $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

Query: find all nodes $v$ s.t. $q \in I_v$, and for each such node report all intervals in $S(v)$. 

Interval Stabbing Queries
Interval Stabbing Queries

Given a set $S$ of $n$ intervals in $\mathbb{R}^1$

Store $S$ in a data structure s.t. given a query point $q$, we can find the intervals in $S$ intersecting $q$ efficiently.

Query: find all nodes $v$ s.t. $q \in I_v$, and for each such node report all intervals in $S(v)$.

$\implies$ we can store $S(v)$ any way we like, since we have to report all intervals in $S(v)$. 
Segment Stabbing Queries

Given a set \( S \) of \( n \) horizontal line segments in the plane.

Store \( S \) in a data structure s.t. given a vertical query segment \( q \), we can find the segments in \( S \) intersecting \( q \) efficiently.

Query: find all nodes \( v \) s.t. \( q \in I_v \), and for each such node report all intervals in \( S(v) \).

\[ \Rightarrow \text{we can store } S(v) \text{ any way we like, since we have to report all intervals in } S(v). \]

Store \( S(v) \) in a balanced BST.

\[ \Rightarrow \text{We can report all segments intersected by } q \text{ in } O(\log^2 n + k) \text{ time}. \]
Segment Stabbing Queries

Given a set $S$ of $n$ disjoint line segments in the plane.

Store $S$ in a data structure s.t. given a vertical query segment $q$, we can find the segments in $S$ intersecting $q$ efficiently.

Query: find all nodes $v$ s.t. $q \in I_v$, and for each such node report all intervals in $S(v)$.

$\implies$ we can store $S(v)$ any way we like, since we have to report all intervals in $S(v)$.

Store $S(v)$ in a balanced BST.

$\implies$

We can report all segments intersected by $q$ in $O(\log^2 n + k)$ time.
Segment Stabbing Queries

Given a set $S$ of $n$ disjoint line segments in the plane.

Store $S$ in a data structure s.t. given a vertical query segment $q$, we can find the segments in $S$ intersecting $q$ efficiently.

We store $S$ in a segment tree $T$

Space usage: $O(n \log n)$

Query time: $O(\log^2 n + k)$

$k = \#\text{intervals reported}$

Preprocessing time: $O(n \log n)$
Windowing Queries

Given a set $S$ of $n$ disjoint line segments in the plane.

Store $S$ in a data structure s.t. given a query rectangle $R$, we can find the segments in $S$ intersecting $R$ efficiently.

The segments that intersect $R$

1) have an endpoint in $R$, or
   find them using a range query with $R$ on the set of end points
   $\implies O(\log^2 n + k)$ query, $O(n \log n)$ space.

2) intersect the boundary of $R$.
   find them using a segment tree
   $\implies O(\log^2 n + k)$ query, $O(n \log n)$ space.
Windowing Queries

Given a set $S$ of $n$ disjoint line segments in the plane.

Store $S$ in a data structure s.t. given a query rectangle $R$, we can find the segments in $S$ intersecting $R$ efficiently.

The segments that intersect $R$

1) have an endpoint in $R$, or
   find them using a range query with $R$ on the set of end points
   $\implies O(\log^2 n + k)$ query, $O(n \log n)$ space.
2) intersect the boundary of $R$.
   find them using a segment tree
   $\implies O(\log^2 n + k)$ query, $O(n \log n)$ space.

Thm. We can solve windowing queries in $O(\log^2 n + k)$ time, using $O(n \log n)$ space after $O(n \log n)$ preprocessing time.
Windowing Queries

Given a set $S$ of $n$ disjoint line segments in the plane.

Store $S$ in a data structure s.t. given a query rectangle $R$, we can find the segments in $S$ intersecting $R$ efficiently.

The segments that intersect $R$

1) have an endpoint in $R$, or
   - find them using a range query with $R$ on the set of end points
     $\Rightarrow O(\log n + k)$ query, $O(n \log n)$ space.
2) intersect the boundary of $R$.
   - find them using a segment tree
     $\Rightarrow O(\log n + k)$ query, $O(n \log n)$ space.

Thm. We can solve windowing queries in $O(\log n + k)$ time, using $O(n \log n)$ space after $O(n \log n)$ preprocessing time.