Outline:

I. Query for the face containing a point

II. Trapezoidal map

III. Geometric complexity of the map

IV. Data structure for search
I. Query for Containing Face

$S$: planar subdivision with $n$ edges

**Query**: Given a point $q$, report face $f$ such that $q \in f$. 
Simple Data Structure

Draw vertical lines through all the vertices.
Simple Data Structure

Draw vertical lines through all the vertices.
Simple Data Structure

Draw vertical lines through all the vertices.

- Sort them by $x$-coordinate.
Simple Data Structure

Draw vertical lines through all the vertices.

- Sort them by $x$-coordinate.
- Determine the slab containing $q$. 
Simple Data Structure

Draw vertical lines through all the vertices.

- Sort them by $x$-coordinate.
- Determine the slab containing $q$.

$O(\log n)$ time
No vertices inside a slab
Slab

No vertices inside a slab

- Bounded by the vertical lines through two adjacent vertices in the sorted list.
Slab

No vertices inside a slab

- Bounded by the vertical lines through two adjacent vertices in the sorted list.
- Edges don't cross each other.
No vertices inside a slab

- Bounded by the vertical lines through two adjacent vertices in the sorted list.

- Edges don’t cross each other.

- Order them from top to bottom.
  - Store edges intersecting a slab in sorted order (e.g., in an array).
  - Label each edge with the face immediately above.
Query Algorithm

1. Determine the slab containing $q$. 
Query Algorithm

1. Determine the slab containing $q$.
   - Binary search with $x$-coordinate of $q$. 
Query Algorithm

1. Determine the slab containing $q$.
   - Binary search with $x$-coordinate of $q$.

$O(\log n)$
Query Algorithm

1. Determine the slab containing $q$.
   - Binary search with $x$-coordinate of $q$.

   $O(\log n)$

2. Binary search within the slab.
Query Algorithm

1. Determine the slab containing $q$.
   - Binary search with $x$-coordinate of $q$.

\[ O(\log n) \]

2. Binary search within the slab.
   - Given a segment $s$ crossing the slab, determine whether $q$ is above, below, or on $s$. 

Query Algorithm

1. Determine the slab containing \( q \).
   - Binary search with \( x \)-coordinate of \( q \).

\[ O(\log n) \]

2. Binary search within the slab.
   - Given a segment \( s \) crossing the slab, determine whether \( q \) is above, below, or on \( s \).

\[ \leq n \text{ segments } \Rightarrow O(\log n) \text{ time.} \]
Query Algorithm

1. Determine the slab containing $q$.
   - Binary search with $x$-coordinate of $q$.
   
   \[ O(\log n) \]

2. Binary search within the slab.
   - Given a segment $s$ crossing the slab, determine whether $q$ is above, below, or on $s$.
   
   \[ \leq n \text{ segments } \Rightarrow O(\log n) \text{ time.} \]

Query time is $O(\log n)$. 
Storage

- An array on $x$-coordinate of vertices. $O(n)$
Storage

• An array on $x$-coordinate of vertices. $O(n)$

• An array for every slab. $O(n)$
Storage

- An array on $x$-coordinate of vertices. $O(n)$
- An array for every slab. $O(n)$
  
  $O(n)$ slabs
Storage

- An array on $x$-coordinate of vertices. $O(n)$
- An array for every slab. $O(n)$

\[
\begin{align*}
O(n) \text{ slabs} & \quad \implies \quad O(n^2)
\end{align*}
\]
Storage

- An array on $x$-coordinate of vertices. $O(n)$
- An array for every slab. $O(n)$

$O(n)$ slabs $\Rightarrow O(n^2)$
Storage

- An array on $x$-coordinate of vertices. $O(n)$
- An array for every slab. $O(n)$

$O(n)$ slabs

$\Theta(n^2)$ storage!
Storage

- An array on $x$-coordinate of vertices. $O(n)$
- An array for every slab. $O(n)$

\[
\begin{align*}
O(n) \text{ slabs} & \quad \Rightarrow \quad O(n^2) \\
\text{n/2 slabs} & \\
\end{align*}
\]

Over-refinement of $S$ for query purpose!
II. Trapezoidal Map

- Makes point location query easier.
- Needs $O(n)$ storage.
II. Trapezoidal Map

- Makes point location query easier.
- Needs $O(n)$ storage.

General position assumption (removable)
No two vertices have the same $x$-coordinate.
II. Trapezoidal Map

- Makes point location query easier.
- Needs $O(n)$ storage.

General position assumption (removable)
No two vertices have the same $x$-coordinate.

- Draw a rectangle bounding all segments in its interior.

axis-parallel rectangle $R$
II. Trapezoidal Map

- Makes point location query easier.
- Needs $O(n)$ storage.

General position assumption (removable)
No two vertices have the same $x$-coordinate.

axis-parallel rectangle $R$

- Draw a rectangle bounding all segments in its interior.
- From every point draw two vertical extensions (up and down).
- Stop when they meet another segment or the boundary of $R$. 

From every point draw two vertical extensions (up and down).
II. Trapezoidal Map

- Makes point location query easier.
- Needs $O(n)$ storage.

General position assumption (removable)
No two vertices have the same $x$-coordinate.

axis-parallel rectangle $R$

- Draw a rectangle bounding all segments in its interior.
- From every point draw two vertical extensions (up and down).
- Stop when they meet another segment or the boundary of $R$. 
II. Trapezoidal Map

- Makes point location query easier.
- Needs $O(n)$ storage.

General position assumption (removable)
No two vertices have the same $x$-coordinate.

axis-parallel rectangle $R$

- Draw a rectangle bounding all segments in its interior.
- From every point draw two vertical extensions (up and down).
- Stop when they meet another segment or the boundary of $R$.

$T(S)$
Face of a Trapezoidal Map

Trapezoid

$f$
Face of a Trapezoidal Map

Trapezoid

Triangle
Every face in $T(s)$ has $\leq 2$ vertical sides and 2 non-vertical sides.
Every face in $T(s)$ has $\leq 2$ vertical sides and 2 non-vertical sides.

A vertical side is one of two cases:
- a vertical extension, or
- a vertical edge of $R$. 
Face of a Trapezoidal Map

Every face in $T(s)$ has $\leq 2$ vertical sides and 2 non-vertical sides.

- A vertical side is one of two cases:
  - a vertical extension, or
  - a vertical edge of $R$. 
Distinguish the two non-vertical sides.
Classification of the Left Side

(a) Degenerating into a point

(b) Lower vertical extension

(c) Upper vertical extension

(d) Upper & lower extension

(e) Left edge of $R$
Classification of the Left Side

(a) Degenerating into a point
(b) Lower vertical extension
(c) Upper vertical extension
Classification of the Left Side

(a) Degenerating into a point
(b) Lower vertical extension
(c) Upper vertical extension
(d) Upper & lower extension
Classification of the Left Side

(a) Degenerating into a point

(b) Lower vertical extension

(c) Upper vertical extension

(d) Upper & lower extension

(e) Left edge of $R$
Defining Endpoint

\[
\text{leftp}(\Delta) \overset{\text{def}}{=} \text{left endpoint of top}(\Delta) \quad \text{(b)} \\
\quad \text{or left endpoint of bottom}(\Delta) \quad \text{(c)} \\
\quad \text{or both} \quad \text{(a)} \\
\quad \text{or right endpoint of a 3rd segment} \quad \text{(d)} \\
\quad \text{or lower left corner of } R \quad \text{(e)}
\]

\[\text{rightp}(\Delta)\] can be similarly defined.
A trapezoid is uniquely determined.
III. Complexity of the Trapezoidal Map

Lemma 1 \( T(S) \) contains \( \leq 6n + 4 \) vertices.
III. Complexity of the Trapezoidal Map

Lemma 1 $T(S)$ contains $\leq 6n + 4$ vertices.

Proof A vertex is one of three types below:
Lemma 1 \( T(S) \) contains \( \leq 6n + 4 \) vertices.

Proof A vertex is one of three types below:

- a vertex of \( R \)
Lemma 1 \( T(S) \) contains \( \leq 6n + 4 \) vertices.

Proof A vertex is one of three types below:

- a vertex of \( R \)
III. Complexity of the Trapezoidal Map

**Lemma 1** \( T(S) \) contains \( \leq 6n + 4 \) vertices.

**Proof**  A vertex is one of three types below:

- a vertex of \( R \)
- an endpoint of a segment
III. Complexity of the Trapezoidal Map

Lemma 1 \( T(S) \) contains \( \leq 6n + 4 \) vertices.

Proof A vertex is one of three types below:

- a vertex of \( R \)
- an endpoint of a segment

\( \leq 2n \)  

\( \geq 4 \)
III. Complexity of the Trapezoidal Map

**Lemma 1** $T(S)$ contains $\leq 6n + 4$ vertices.

**Proof** A vertex is one of three types below:

- a vertex of $R$  \[ \leq 2n \]
- an endpoint of a segment  \[ \leq 2n \]
- (shared endpoints may exist)
III. Complexity of the Trapezoidal Map

Lemma 1 \( T(S) \) contains \( \leq 6n + 4 \) vertices.

Proof  A vertex is one of three types below:

- a vertex of \( R \)
- an endpoint of a segment (\( \leq 2n \))
- the point where the vertical extension line starting in an endpoint reaches another segment or the boundary of \( R \).
Lemma 1  \( T(S) \) contains \( \leq 6n + 4 \) vertices.

Proof  A vertex is one of three types below:

- a vertex of \( R \)
- an endpoint of a segment \( \leq 2n \)
- the point where the vertical extension line starting in an endpoint reaches another segment or the boundary of \( R \).

Every endpoint generates at most two such points.
III. Complexity of the Trapezoidal Map

Lemma 1 $T(S)$ contains $\leq 6n + 4$ vertices.

Proof A vertex is one of three types below:

- a vertex of $R$
- an endpoint of a segment $\leq 2n$ (shared endpoints may exist)
- the point where the vertical extension line starting in an endpoint reaches another segment or the boundary of $R$. $\leq 2 \times 2n$

Every endpoint generates at most two such points.
Lemma 1  $T(S)$ contains $\leq 6n + 4$ vertices.

Proof  A vertex is one of three types below:

- a vertex of $R$

- an endpoint of a segment

- the point where the vertical extension line starting in an endpoint reaches another segment or the boundary of $R$.

Every endpoint generates at most two such points.

$\#\text{vertices} \leq 4 + 2n + 4n = 6n + 4$
Number of Trapezoids

Lemma 2  $T(S)$ contains $\leq 3n + 1$ trapezoids.
Number of Trapezoids

**Lemma 2** $T(S)$ contains $\leq 3n + 1$ trapezoids.

**Proof** Every trapezoid $\Delta$ is represented by $\text{leftp}(\Delta)$. Need only count $\text{leftp}(\Delta)$, including multiplicities (#times for each endpoint).
Lemma 2  $T(S)$ contains $\leq 3n + 1$ trapezoids.

Proof  Every trapezoid $\Delta$ is represented by leftp($\Delta$). Need only count leftp($\Delta$), including multiplicities (#times for each endpoint).

leftp($\Delta$) is one of three possible types:
Lemma 2 \( T(S) \) contains \( \leq 3n + 1 \) trapezoids.

Proof Every trapezoid \( \Delta \) is represented by \( \text{leftp}(\Delta) \). Need only count \( \text{leftp}(\Delta) \), including multiplicities (\#times for each endpoint).

\( \text{leftp}(\Delta) \) is one of three possible types:

- lower left corner of \( R \)
Lemma 2  \( T(S) \) contains \( \leq 3n + 1 \) trapezoids.

Proof  Every trapezoid \( \Delta \) is represented by \( \text{leftp}(\Delta) \). Need only count \( \text{leftp}(\Delta) \), including multiplicities (\#times for each endpoint).

\( \text{leftp}(\Delta) \) is one of three possible types:

- lower left corner of \( R \)

\( \Rightarrow \) it is \( \text{leftp}(\Delta) \) for exactly 1 trapezoid.
Number of Trapezoids

Lemma 2 \( T(S) \) contains \( \leq 3n + 1 \) trapezoids.

Proof Every trapezoid \( \Delta \) is represented by \( \text{leftp}(\Delta) \). Need only count \( \text{leftp}(\Delta) \), including multiplicities (\#times for each endpoint).

\( \text{leftp}(\Delta) \) is one of three possible types:

- lower left corner of \( R \)
  \[ \implies \text{it is leftp}(\Delta) \text{ for exactly 1 trapezoid.} \]

- right endpoint of a segment (\( n \) such points)
Lemma 2 \( T(S) \) contains \( \leq 3n + 1 \) trapezoids.

Proof Every trapezoid \( \Delta \) is represented by \( \text{leftp}(\Delta) \). Need only count \( \text{leftp}(\Delta) \), including multiplicities (#times for each endpoint).

\( \text{leftp}(\Delta) \) is one of three possible types:

- lower left corner of \( R \)
  \[ \Rightarrow \text{it is leftp}(\Delta) \text{ for exactly } 1 \text{ trapezoid}. \]

- right endpoint of a segment \( (n \text{ such points}) \)
  \[ \Rightarrow \text{it is leftp}(\Delta) \text{ for } \leq 1 \text{ trapezoid}. \]
Lemma 2 \( T(S) \) contains \( \leq 3n + 1 \) trapezoids.

Proof Every trapezoid \( \Delta \) is represented by \( \text{leftp}(\Delta) \). Need only count \( \text{leftp}(\Delta) \), including multiplicities (#times for each endpoint).

\( \text{leftp}(\Delta) \) is one of three possible types:

- lower left corner of \( R \)

  \( \Rightarrow \) it is \( \text{leftp}(\Delta) \) for exactly 1 trapezoid.

- right endpoint of a segment (\( n \) such points)

  \( \Rightarrow \) it is \( \text{leftp}(\Delta) \) for \( \leq 1 \) trapezoid.

  \( \uparrow \) shared endpoint
• left endpoint of a segment ($n$ such points)

$\iff \text{leftp} (\Delta)$ for $\leq 2$ trapezoids.

↑

shared endpoint
(cont’d)

- left endpoint of a segment ($n$ such points)

  $\iff$ \text{leftp}(\Delta)$ for $\leq 2$ trapezoids.

  $\uparrow$

  shared endpoint

\[
\# \text{ trapezoids} \leq 1 + 1 \times n + 2 \times n
\]

$= 3n + 1$
Adjacency

Trapezoids $\Delta$ and $\Delta'$ are *adjacent* if they share a *vertical* edge.
Adjacency

Trapezoids $\Delta$ and $\Delta'$ are *adjacent* if they share a *vertical* edge.

$\Delta$ is adjacent to $\Delta_1$, $\Delta_2$, $\Delta_3$, and $\Delta_4$ but not to $\Delta_5$. 
Adjacency

Trapezoids $\Delta$ and $\Delta'$ are *adjacent* if they share a *vertical* edge.

$\Delta$ is adjacent to $\Delta_1$, $\Delta_2$, $\Delta_3$, and $\Delta_4$ but not to $\Delta_5$.

$\Delta_1$: upper left neighbor
$\Delta_2$: lower left neighbor
$\Delta_3$: lower right neighbor
$\Delta_4$: upper right neighbor
A trapezoid has $\leq 4$ neighbors under the general position assumptions.
A trapezoid has $\leq 4$ neighbors under the general position assumptions.

$\Delta'$ adjacent to $\Delta$ along the left vertical edge of $\Delta$. 
A trapezoid has $\leq 4$ neighbors under the general position assumptions.

$\Delta'$ adjacent to $\Delta$ along the left vertical edge of $\Delta$.

$\text{top}(\Delta) = \text{top}(\Delta')$ or $\text{bottom}(\Delta) = \text{bottom}(\Delta')$
A trapezoid has $\leq 4$ neighbors under the general position assumptions.

$\Delta'$ adjacent to $\Delta$ along the left vertical edge of $\Delta$.

$\text{top}(\Delta) = \text{top}(\Delta')$ or $\text{bottom}(\Delta) = \text{bottom}(\Delta')$

$\text{top}(\Delta) = \text{top}(\Delta_1) = \text{top}(\Delta_4)$
A trapezoid has \( \leq 4 \) neighbors under the general position assumptions.

\[ \Delta \] adjacent to \( \Delta \) along the left vertical edge of \( \Delta \).

\[
\begin{align*}
\text{top}(\Delta) &= \text{top}(\Delta') \\
\text{bottom}(\Delta) &= \text{bottom}(\Delta')
\end{align*}
\]

\[
\begin{align*}
\text{top}(\Delta) &= \text{top}(\Delta_1) = \text{top}(\Delta_4) \\
\text{bottom}(\Delta) &= \text{bottom}(\Delta_2) = \text{bottom}(\Delta_3)
\end{align*}
\]
IV. Point Location Data Structure

\(D\) is a directed acyclic graph (DAG)

- one root
- one leaf for every trapezoid
- two types of internal nodes
  - \(x\)-nodes labeled with an endpoint of some segment
  - \(y\)-nodes labeled with a segment

\(A\) \(E\) 
\(p_1\) \(s_1\) \(q_1\)
\(B\) \(D\) 
\(s_2\)
\(p_2\) \(C\) \(q_2\)
\(G\)
IV. Point Location Data Structure

\( D \) is a directed acyclic graph (DAG)

- one root
- one leaf for every trapezoid
- two types of internal nodes
  - \( x \)-nodes labeled with an endpoint of some segment
  - \( y \)-nodes labeled with a segment
$T(S)$ and $D$ cross-referenced

trapezoid $\Delta \iff$ leaf of $D$
$T(S)$ and $D$ cross-referenced

trapezoid $\Delta \Rightarrow$ leaf of $D$

pointer
Query

\[ p_1 \rightarrow s_1 \rightarrow E \rightarrow q_1 \]

\[ p_2 \rightarrow q_2 \rightarrow s_2 \rightarrow C \rightarrow p_1 \rightarrow B \rightarrow E \rightarrow D \rightarrow s_1 \rightarrow F \rightarrow q_1 \rightarrow G \]
Query

\[
\begin{align*}
&p_1 \quad s_1 \quad E \\
&B \quad D \quad q_2 \\
&s_2 \quad C \\
&p_2 \quad q_1 \\
&A \\
&s_1 \quad C \\
&B \\
&E \\
&D \\
&s_1 \quad F \\
&D \\
&E \\
&D \\
&D \\
&G
\end{align*}
\]
Query
Query
Query