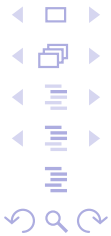
Point Location

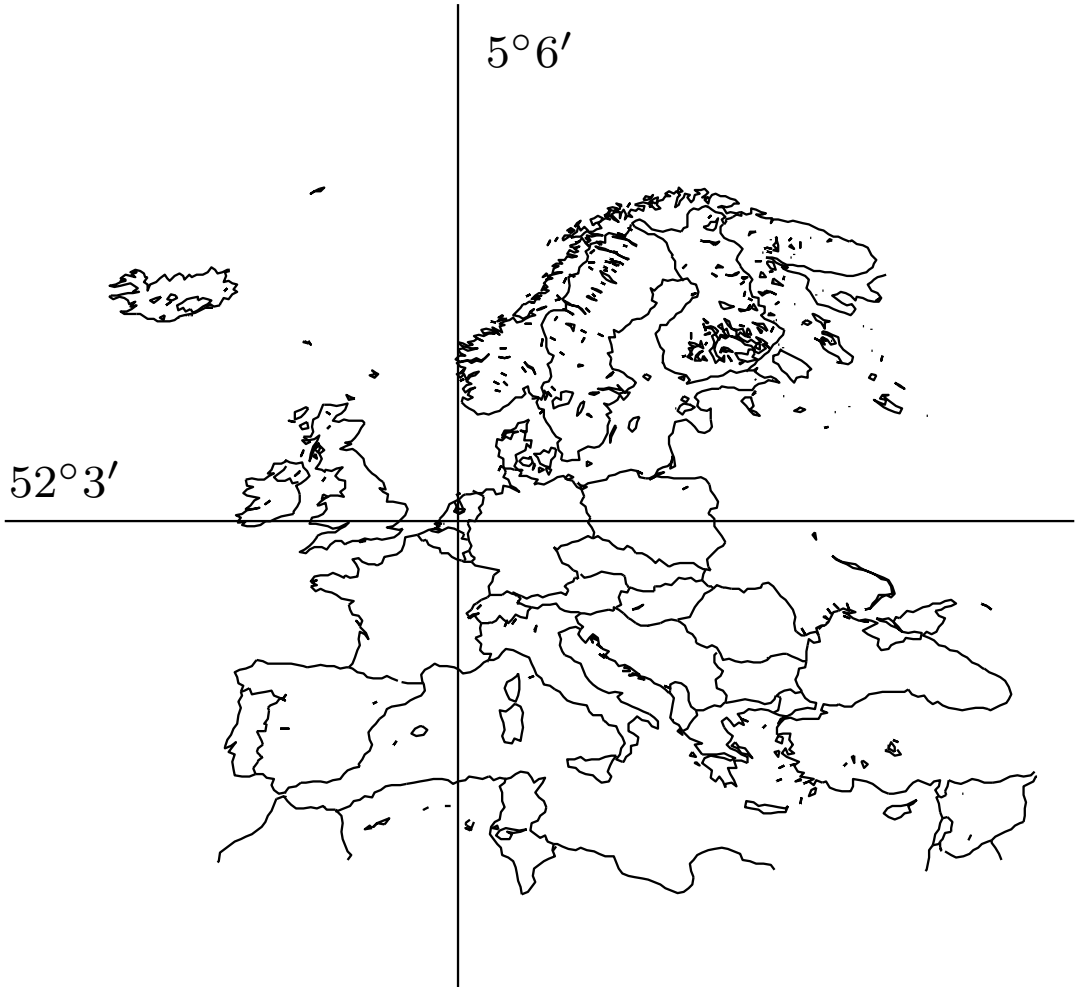
1397-2



Motivation:

Point location in a map

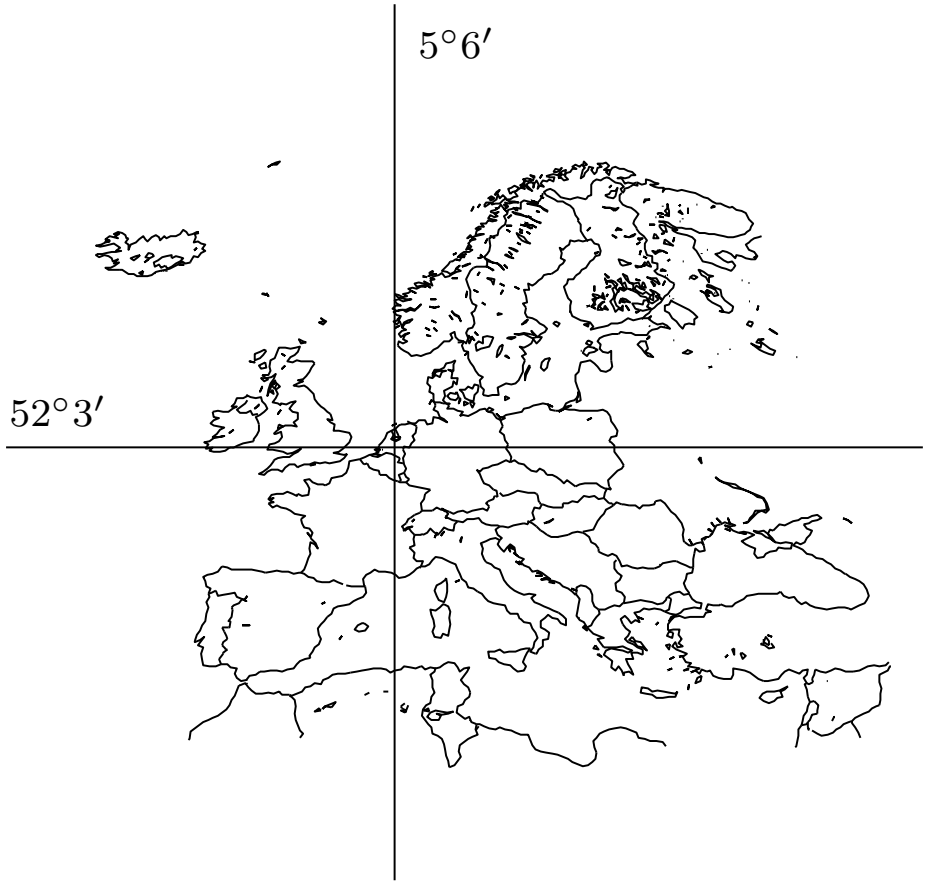
Point location query: Given a map and a query point q specified by its coordinates, find the region of the map containing q.



Motivation:

Point location in a map

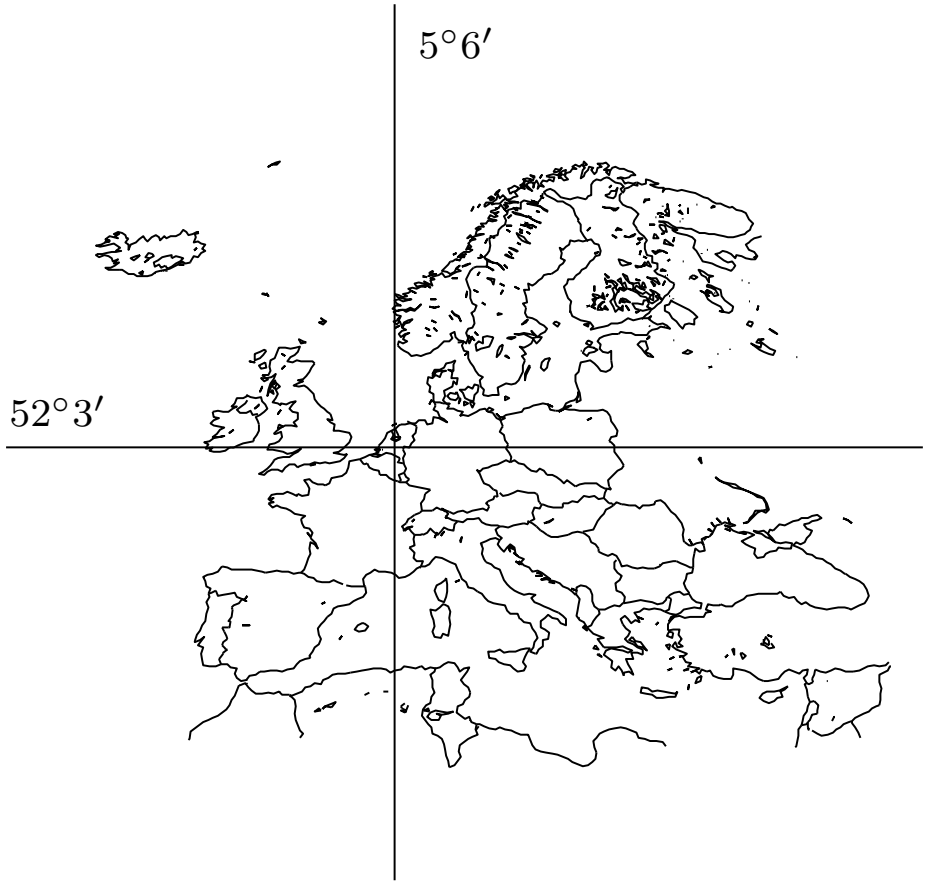
- Store the map electronically, and let the computer do the point location for you.
- Preprocess the maps, and to store them in a data structure that makes it possible to answer point location queries fast.



Motivation:

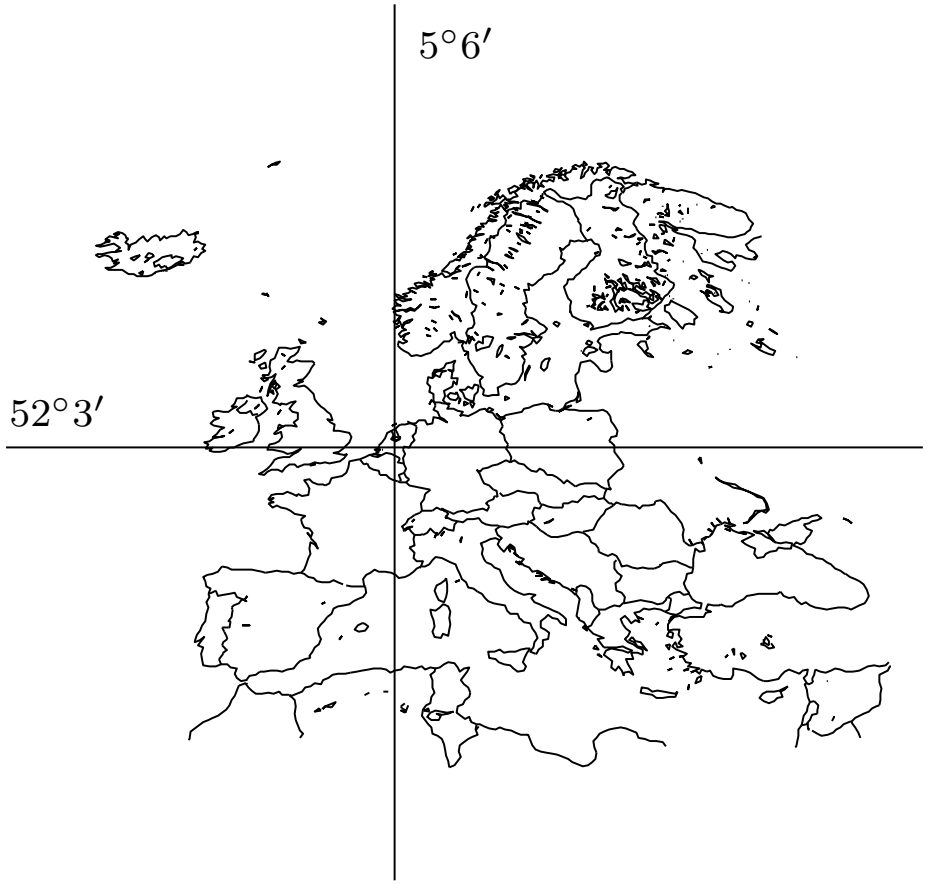
Point location in a map

- Store the map electronically, and let the computer do the point location for you.
- Preprocess the maps, and to store them in a data structure that makes it possible to answer point location queries fast.



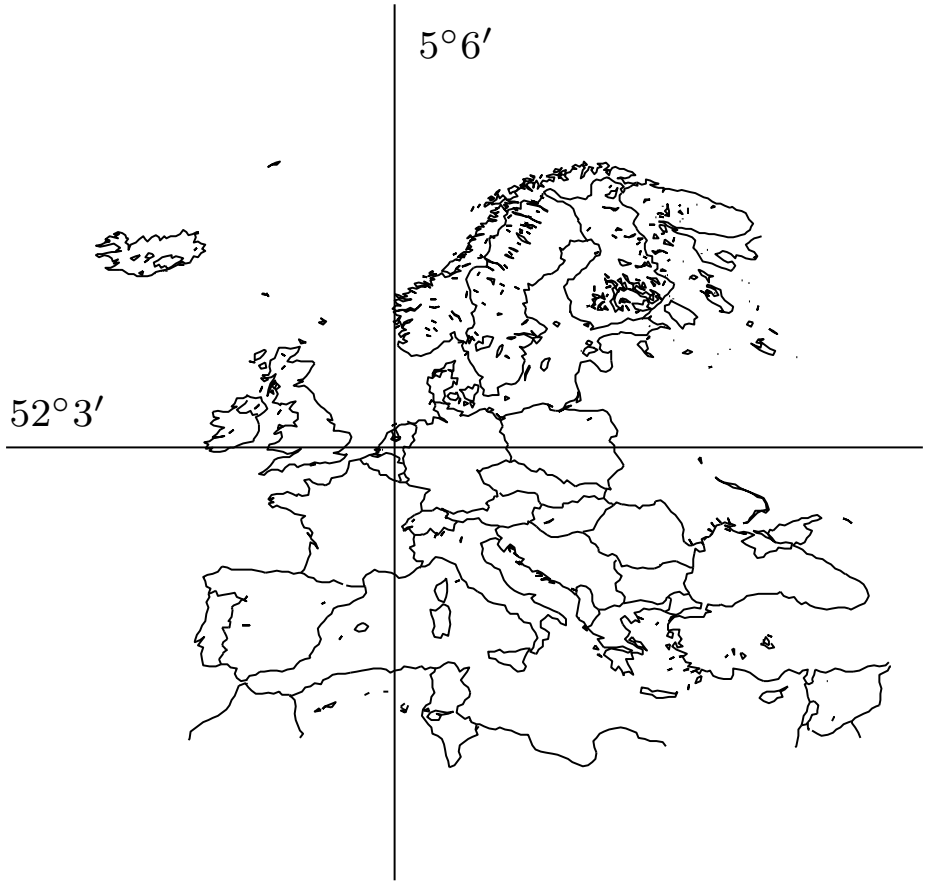
Trapezoidal Maps:

- Store the map electronically, and let the computer do the point location for you.
- Preprocess the maps, and to store them in a data structure that makes it possible to answer point location queries fast.



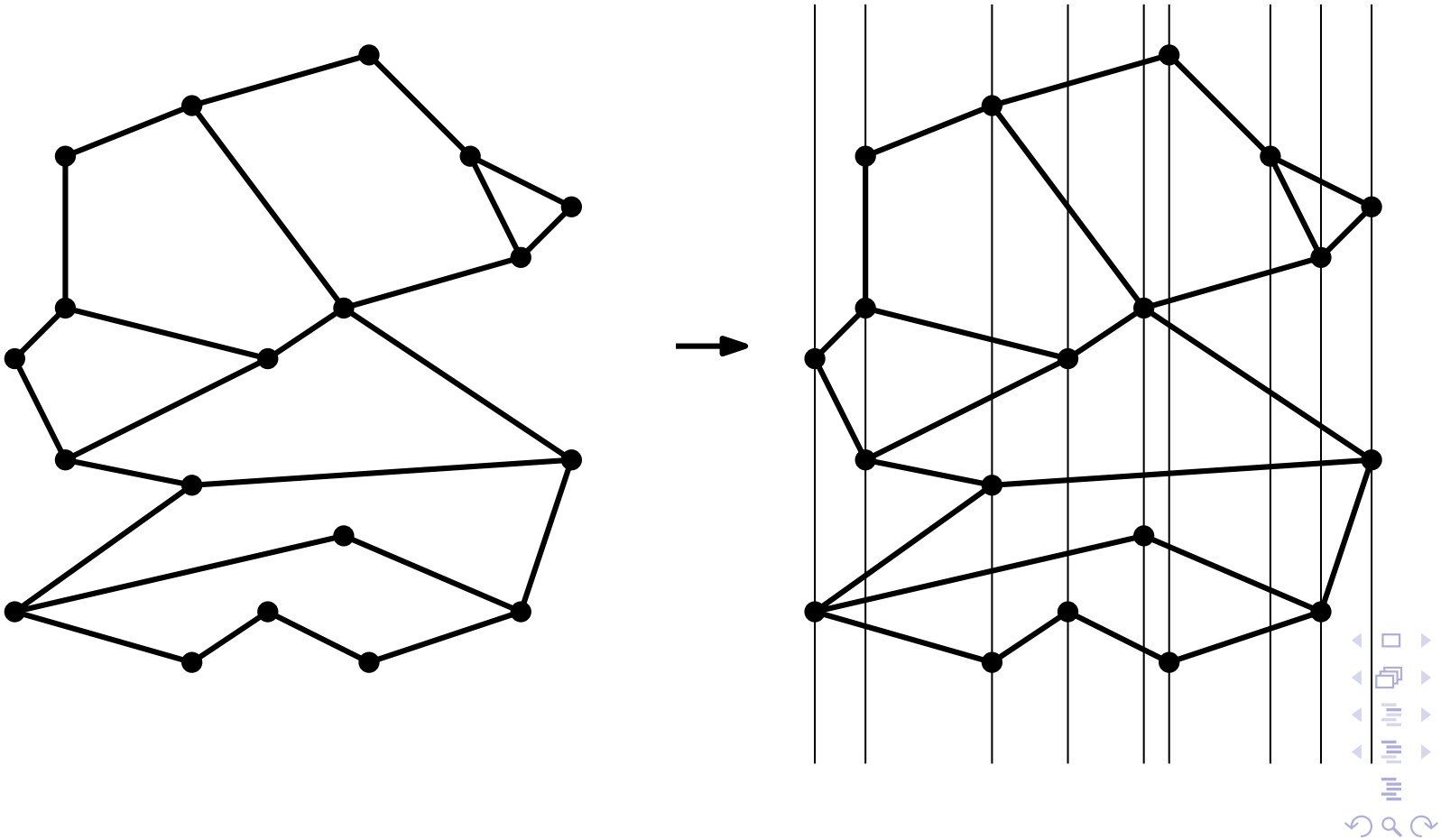
Trapezoidal Maps:

- Store the map electronically, and let the computer do the point location for you.
- Preprocess the maps, and to store them in a data structure that makes it possible to answer point location queries fast.



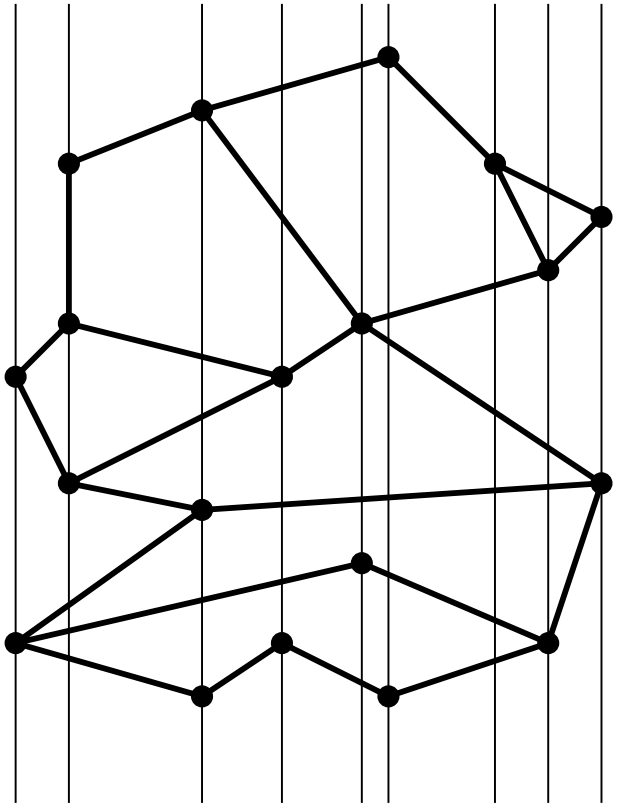
Trapezoidal Maps:

- Given a planar subdivision S with n edges, store S in such a way that we can answer queries of the following type: Given a query point q, report the face f of S that contains q.



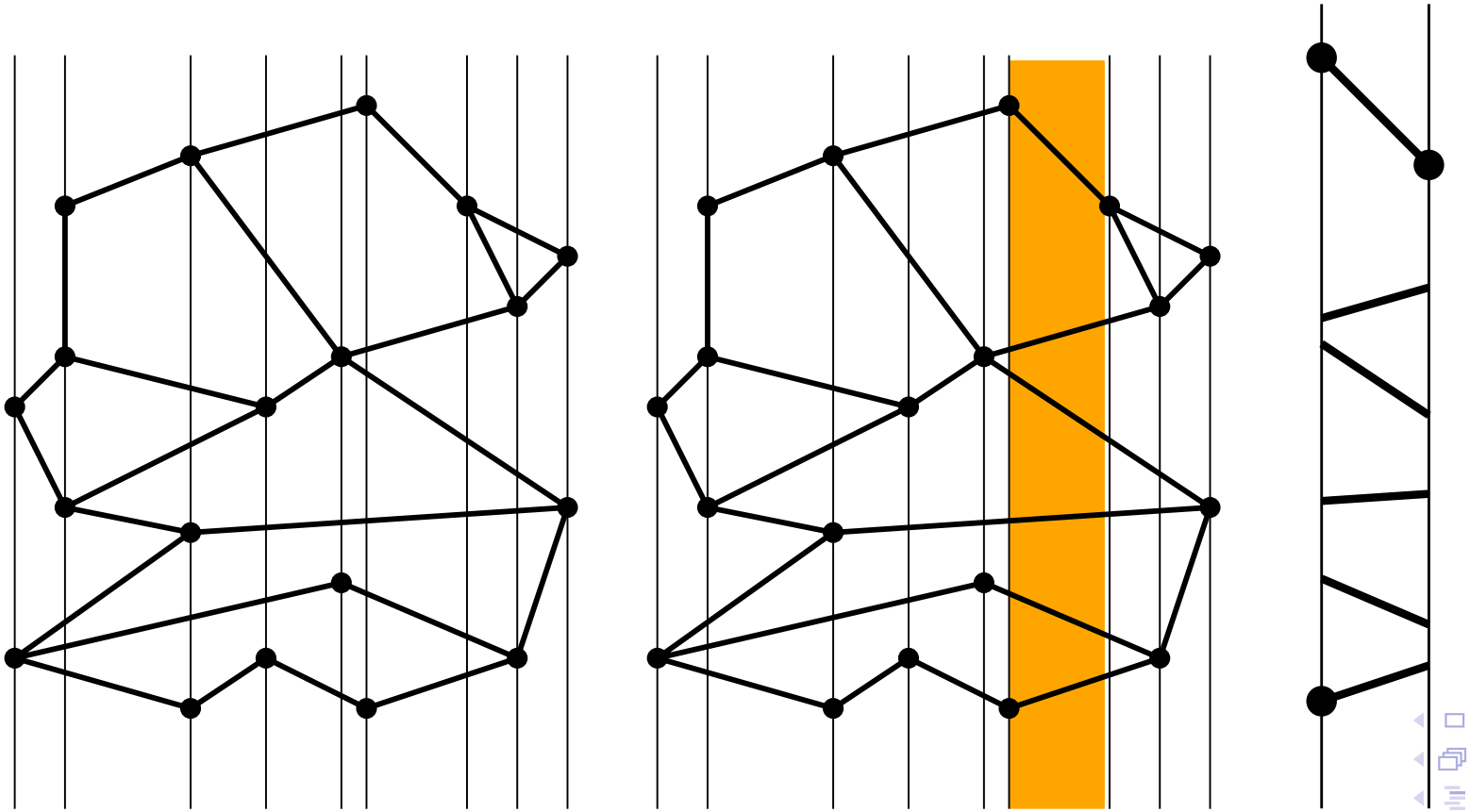
Trapezoidal Maps:

- Find the slab containing the query point: $\mathcal{O}(\log n)$ time.
- Query the slab: $\mathcal{O}(\log n)$ time.



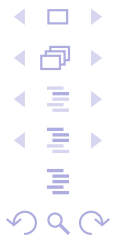
Trapezoidal Maps:

- Find the slab containing the query point: $\mathcal{O}(\log n)$ time.
- Query the slab: $\mathcal{O}(\log n)$ time.



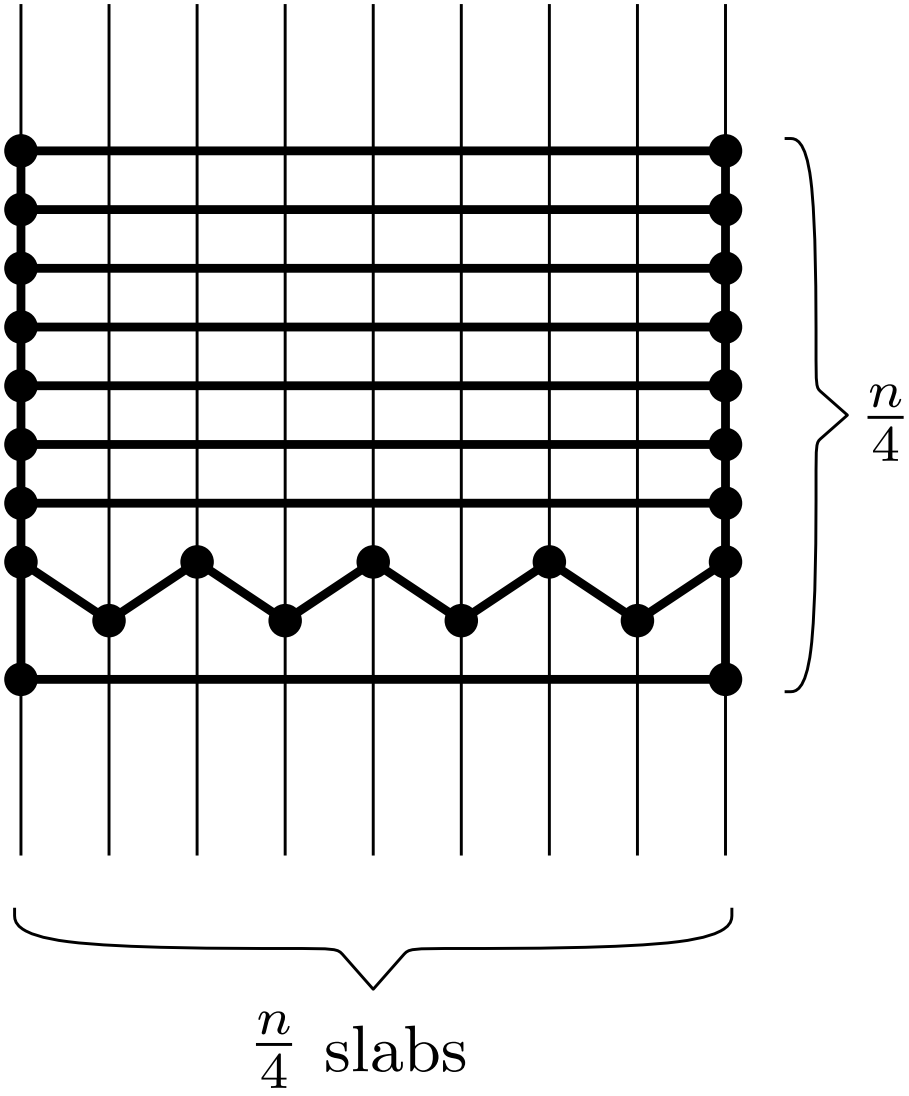
Trapezoidal Maps:

- What about the storage requirements?



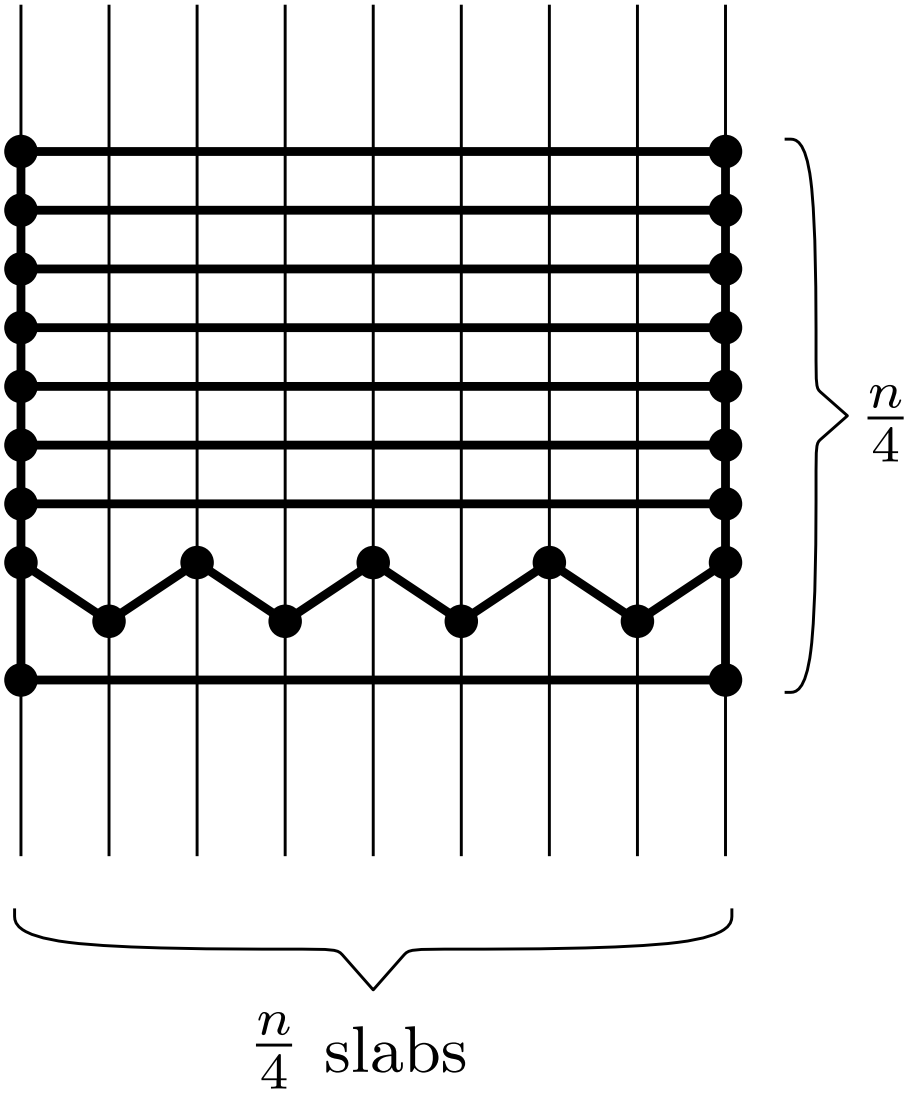
Trapezoidal Maps:

- What about the storage requirements?



Trapezoidal Maps:

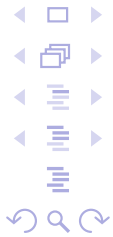
- What about the storage requirements? $\mathcal{O}(n^2)$



Trapezoidal Maps:

Two Simplifications:

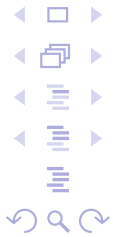
- Bounding Box.
- No two distinct points on a vertical line (general position).



Trapezoidal Maps:

Two Simplifications:

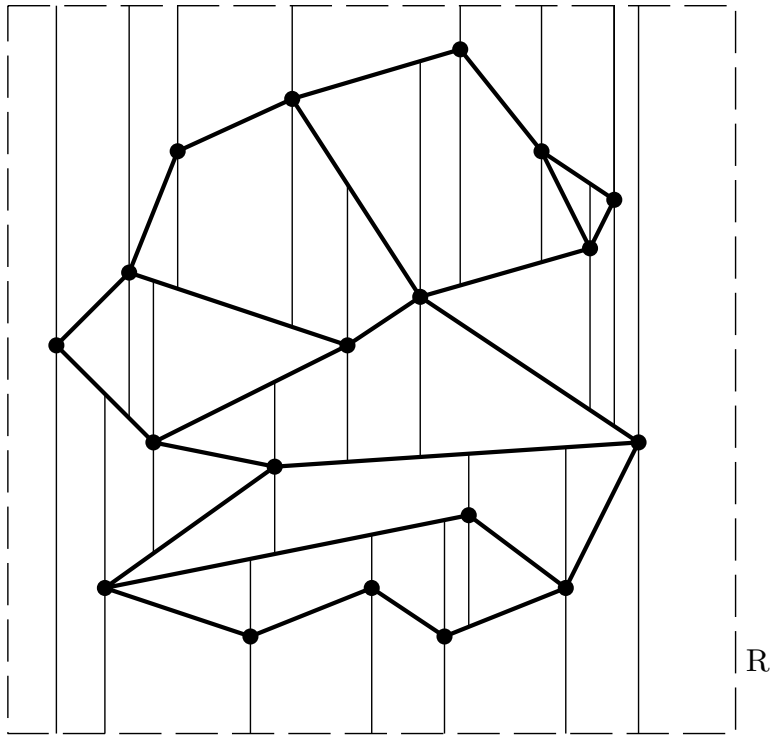
- Bounding Box.
- No two distinct points on a vertical line (general position).



Trapezoidal Maps:

Trapezoidal Map:

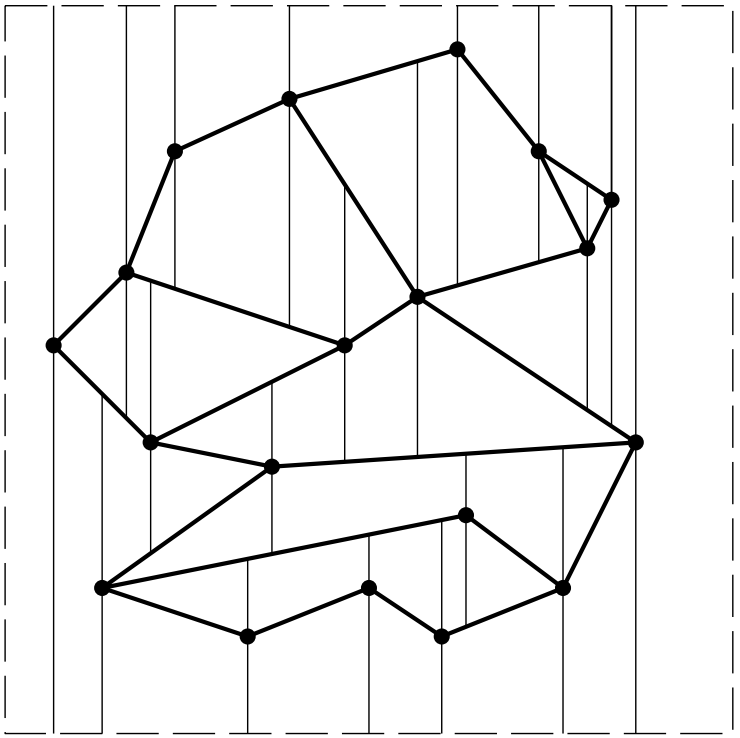
- drawing two vertical extensions from every endpoint
- Sides of a face

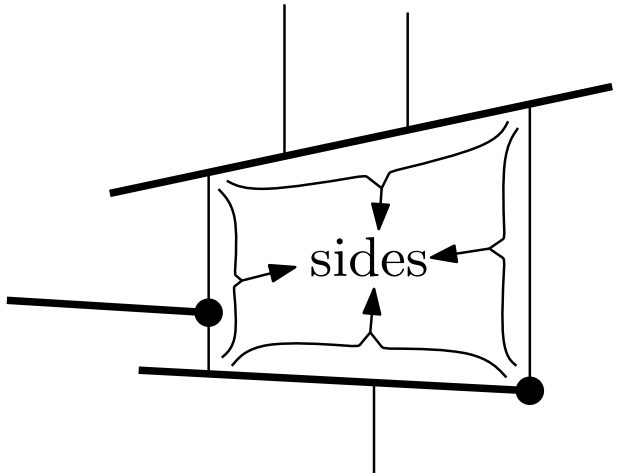


Trapezoidal Maps:

Trapezoidal Map:

- drawing two vertical extensions from every endpoint
- Sides of a face





Trapezoidal Maps:

Lemma 6.1 *Each face in a trapezoidal map of a set S of line segments in general position has one or two vertical sides and exactly two non-vertical sides.*

Proof.

- Each face is convex: all angles are at most $180°$
- Because of the convexity, each face has at most 2 vertical sides.
- If there is more than two non-vertical sides, then at least two of them must meet at a vertex which contradict the definition of a face.
- Since a face is bounded, it cannot have less than two non-vertical sides and that it must have at least one vertical side.

Trapezoidal Maps:

Lemma 6.1 *Each face in a trapezoidal map of a set S of line segments in general position has one or two vertical sides and exactly two non-vertical sides.*

Proof.

- Each face is convex: all angles are at most $180°$
- Because of the convexity, each face has at most 2 vertical sides.
- If there is more than two non-vertical sides, then at least two of them must meet at a vertex which contradict the definition of a face.
- Since a face is bounded, it cannot have less than two non-vertical sides and that it must have at least one vertical side.

Trapezoidal Maps:

Lemma 6.1 *Each face in a trapezoidal map of a set S of line segments in general position has one or two vertical sides and exactly two non-vertical sides.*

Proof.

- Each face is convex: all angles are at most $180°$
- Because of the convexity, each face has at most 2 vertical sides.
- If there is more than two non-vertical sides, then at least two of them must meet at a vertex which contradict the definition of a face.
- Since a face is bounded, it cannot have less than two non-vertical sides and that it must have at least one vertical side.

Trapezoidal Maps:

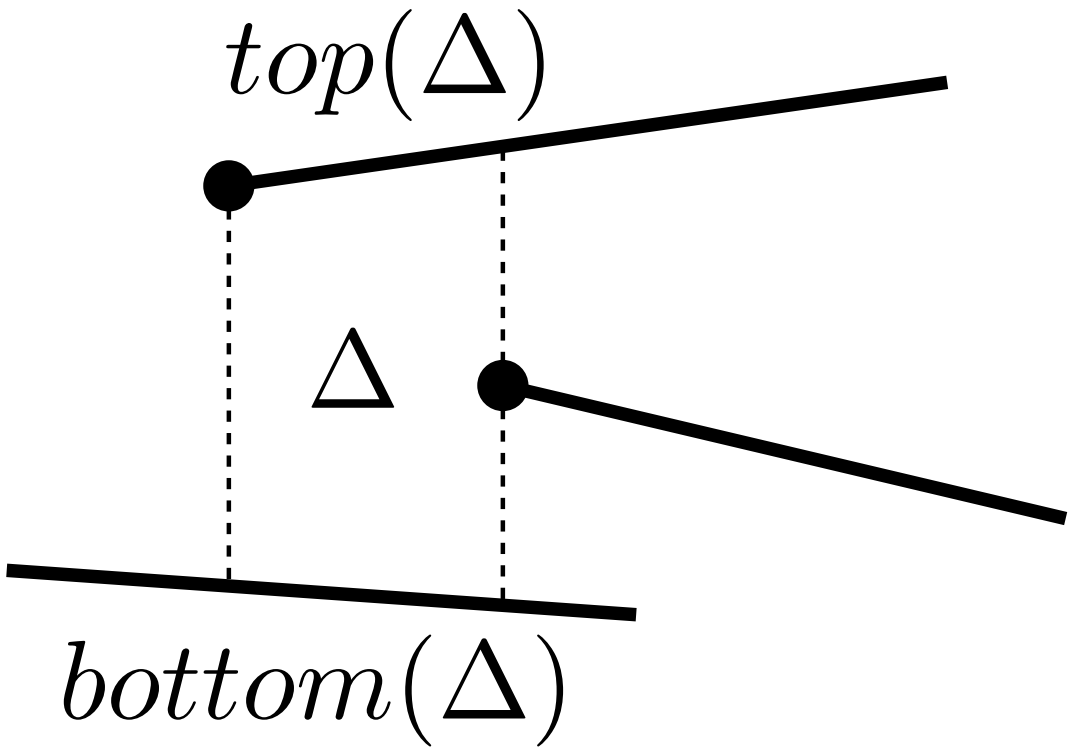
Lemma 6.1 *Each face in a trapezoidal map of a set S of line segments in general position has one or two vertical sides and exactly two non-vertical sides.*

Proof.

- Each face is convex: all angles are at most $180°$
- Because of the convexity, each face has at most 2 vertical sides.
- If there is more than two non-vertical sides, then at least two of them must meet at a vertex which contradict the definition of a face.
- Since a face is bounded, it cannot have less than two non-vertical sides and that it must have at least one vertical side.

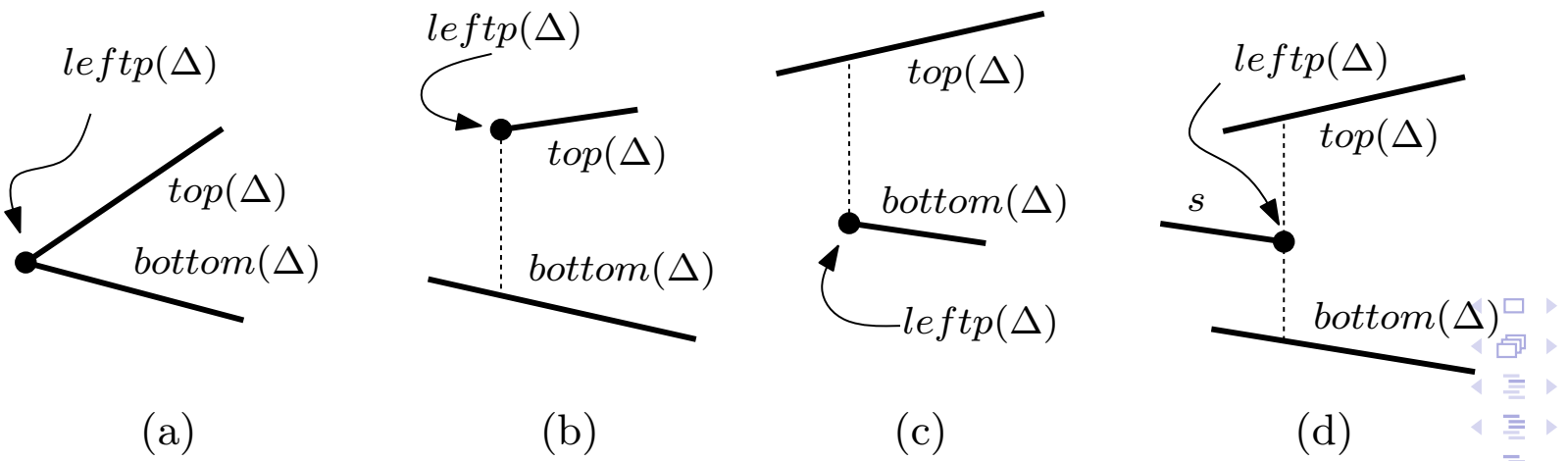
Trapezoidal Maps:

Lemma 6.1 *Each face in a trapezoidal map of a set S of line segments in general position has one or two vertical sides and exactly two non-vertical sides.*



Trapezoidal Maps: Cases for the left side:

(a) It degenerates to a point, which is the common left endpoint of $top(\Delta)$ and $bottom(\Delta)$.
(b) It is the lower vertical extension of the left endpoint of $top(\Delta)$ that abuts on $bottom(\Delta)$.
(c) It is the upper vertical extension of the left endpoint of $bottom(\Delta)$ that abuts on $top(\Delta)$.
(d) It consists of the upper and lower extension of the right endpoint p of a third segment s. These extensions abut on $top(\Delta)$ and $bottom(\Delta)$, respectively.
(e) It is the left edge of R. This case occurs for a single trapezoid of $\mathcal{T}(S)$ only, namely the unique leftmost trapezoid of $\mathcal{T}(S)$.



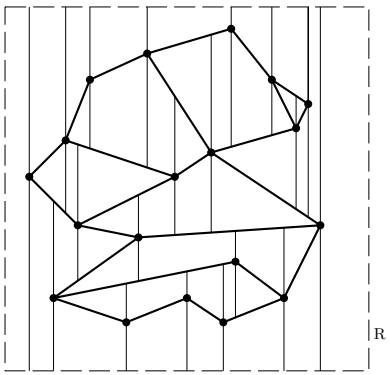
(a) (b) (c) (d)

Trapezoidal Maps:

Lemma 6.2 *The trapezoidal map $\mathcal{T}(S)$ of a set S of n line segments in general position contains at most $6n+4$ vertices and at most $3n+1$ trapezoids.*

Proof.

- #vertices: Bounding box+endpoints of line segments+new vertices generated by extensions= $4 + 2n + 2(2n) = 6n + 4$.
- Left endpoint of a segment can be $leftp(\Delta)$ of two trapezoids. (Note coincide endpoints) .
- Right endpoint of a segment can be $leftp(\Delta)$ of one trapezoid.
- #trapezoids= $3n + 1$.

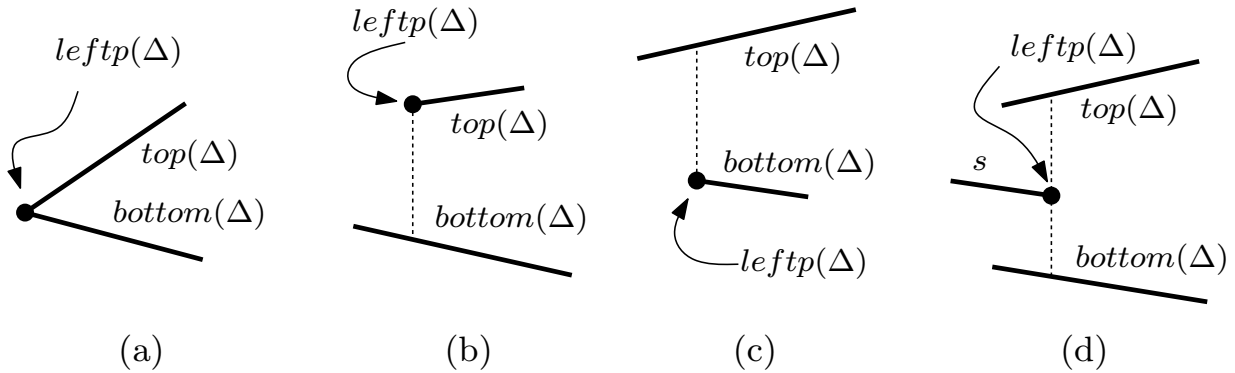


Trapezoidal Maps:

Lemma 6.2 *The trapezoidal map $\mathcal{T}(S)$ of a set S of n line segments in general position contains at most $6n+4$ vertices and at most $3n+1$ trapezoids.*

Proof.

- #vertices: Bounding box+endpoints of line segments+new vertices generated by extensions= $4+2n+2(2n)=6n+4$.
- Left endpoint of a segment can be $leftp(\Delta)$ of two trapezoids. (Note coincide endpoints) .
- Right endpoint of a segment can be $leftp(\Delta)$ of one trapezoid.
- #trapezoids= $3n+1$.

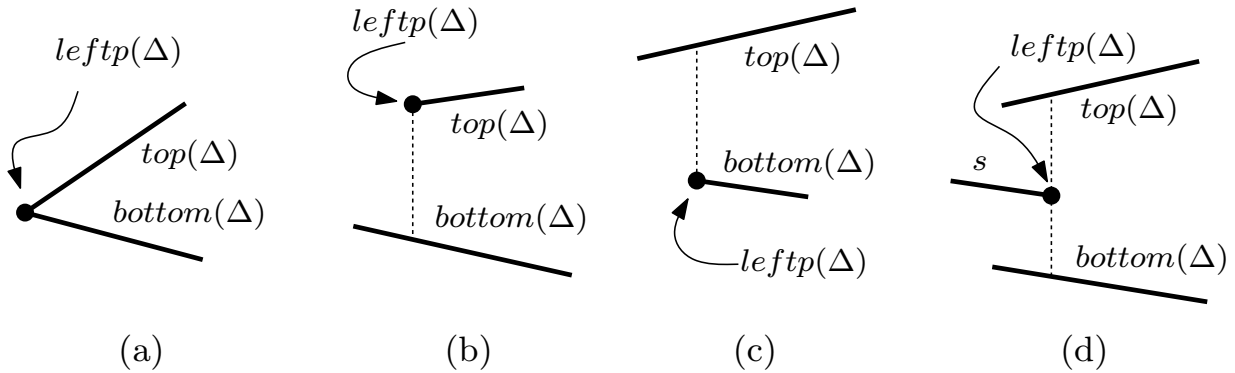


Trapezoidal Maps:

Lemma 6.2 *The trapezoidal map $\mathcal{T}(S)$ of a set S of n line segments in general position contains at most $6n+4$ vertices and at most $3n+1$ trapezoids.*

Proof.

- #vertices: Bounding box+endpoints of line segments+new vertices generated by extensions= $4+2n+2(2n)=6n+4$.
- Left endpoint of a segment can be $leftp(\Delta)$ of two trapezoids. (Note coincide endpoints) .
- Right endpoint of a segment can be $leftp(\Delta)$ of one trapezoid.
- #trapezoids= $3n+1$.



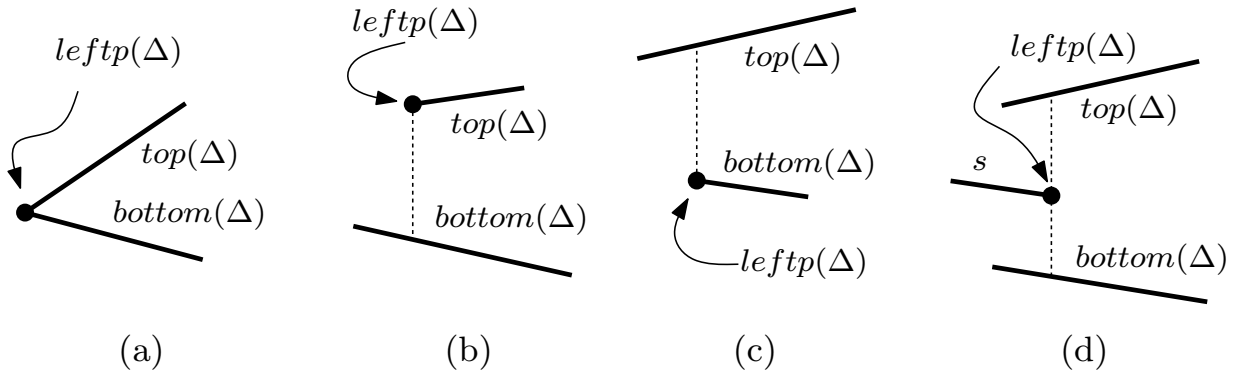
(a) (b) (c) (d)

Trapezoidal Maps:

Lemma 6.2 *The trapezoidal map $\mathcal{T}(S)$ of a set S of n line segments in general position contains at most $6n+4$ vertices and at most $3n+1$ trapezoids.*

Proof.

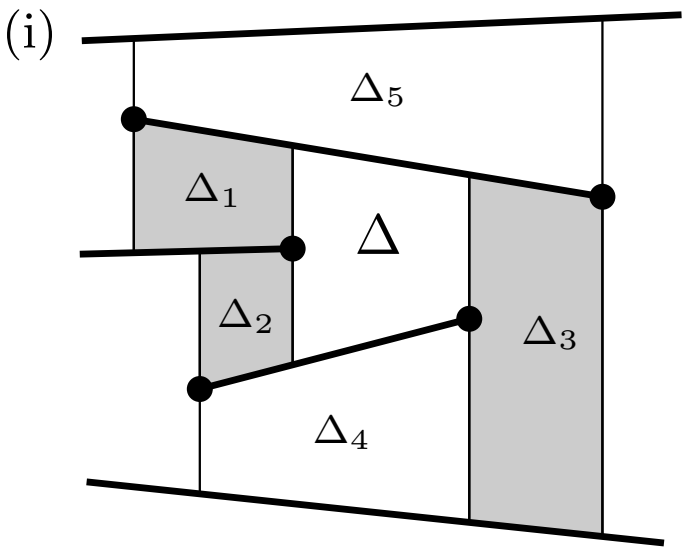
- #vertices: Bounding box+endpoints of line segments+new vertices generated by extensions= $4 + 2n + 2(2n) = 6n + 4$.
- Left endpoint of a segment can be $leftp(\Delta)$ of two trapezoids. (Note coincide endpoints) .
- Right endpoint of a segment can be $leftp(\Delta)$ of one trapezoid.
- #trapezoids= $3n + 1$.

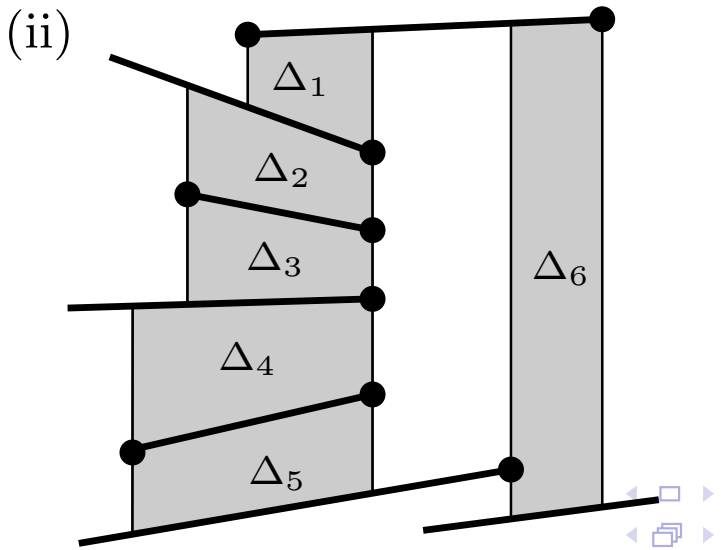


Trapezoidal Maps:

Definition: Adjacent

- Two trapezoids are adjacent if they meet along a vertical edge.
- Since line segments are in general position, a trapezoid has at most 4 adjacent trapezoids.

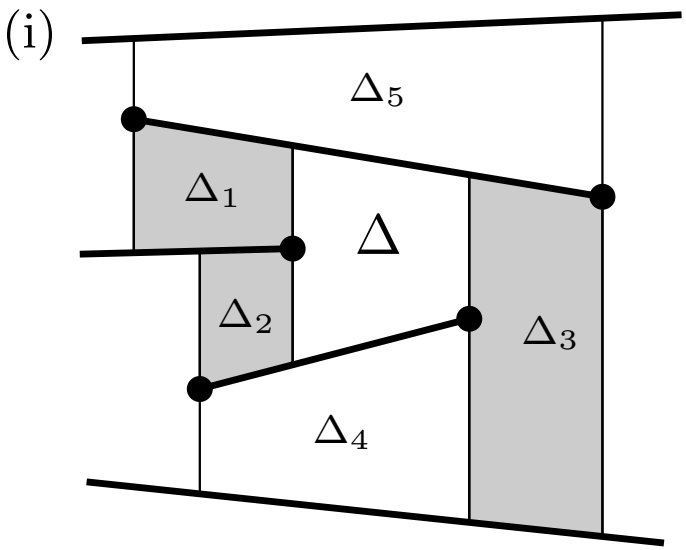


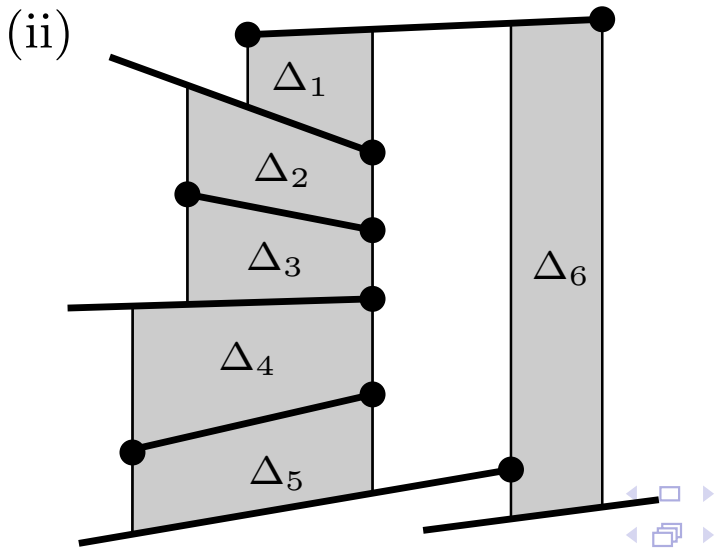


Trapezoidal Maps:

Definition: Adjacent

- Two trapezoids are adjacent if they meet along a vertical edge.
- Since line segments are in general position, a trapezoid has at most 4 adjacent trapezoids.

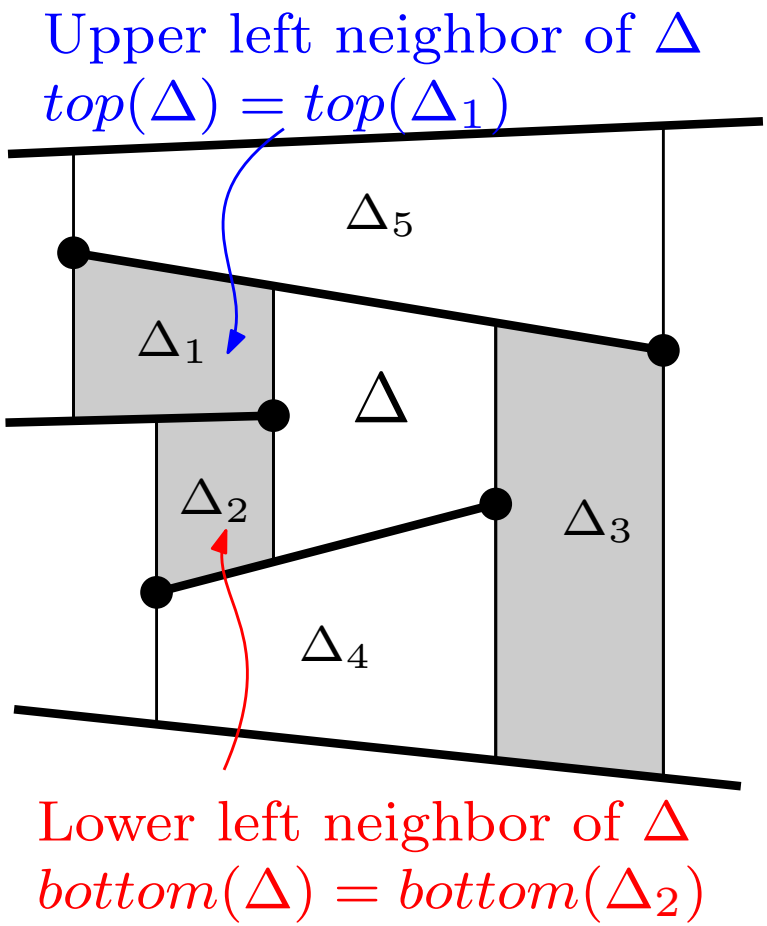




Trapezoidal Maps:

Definition: Upper and Lower left neighbor

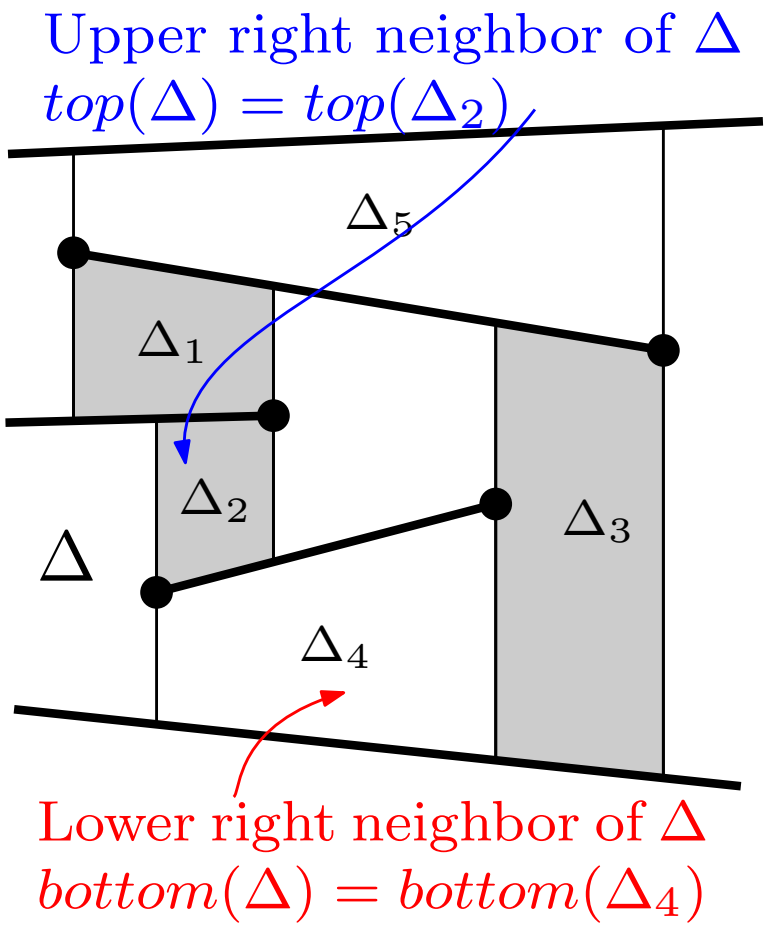
- If Δ' is adjacent to Δ along the left vertical of Δ:



Trapezoidal Maps:

Definition: Upper and Lower right neighbor

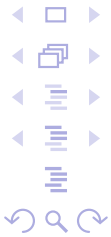
- If Δ' is adjacent to Δ along the right vertical of Δ:



Trapezoidal Maps:

How to store a trapezoidal map:

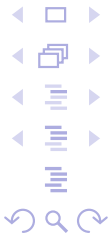
- In a DCEL
- Specialize structure:
  Records for line segments and their endpoints
  For each trapezoid: store pointer to top, bottom, leftp and rightp and neighbors of the trapezoid.



Trapezoidal Maps:

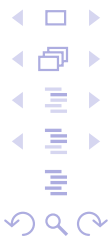
How to store a trapezoidal map:

- In a DCEL
- Specialize structure:
  Records for line segments and their endpoints
  For each trapezoid: store pointer to top, bottom, leftp and rightp and neighbors of the trapezoid.



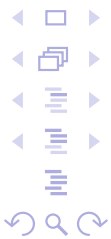
Randomized Incremental Algorithm

- Randomized: See it later.
- Incremental: We add segments one by one.



Randomized Incremental Algorithm

- Randomized: See it later.
- Incremental: We add segments one by one.

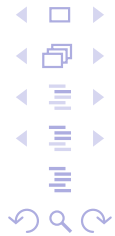


How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Point Location Data Structure (search structure)

- Directed Acyclic Graph
- Exactly one leaf for each trapezoid
- Inner nodes have out-degree 2.
- Inner nodes: x-nodes: labeled with an endpoint.
- Inner nodes: y-nodes: labeled with a segment.
- A query point starts at the root and traverse the structure until it reaches a leaf that contains q.

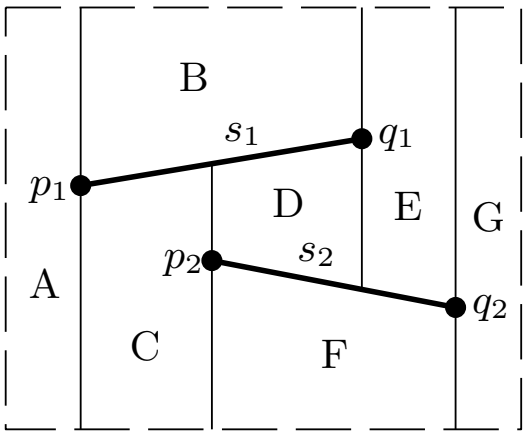


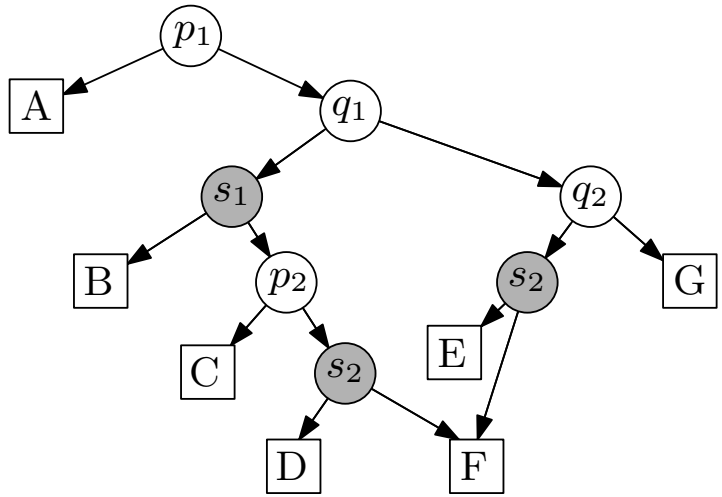
How to construct a Trapezoidal Map?

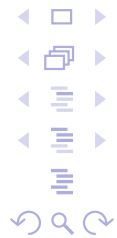
A Randomized Incremental Algorithm

Point Location Data Structure (search structure)

- Directed Acyclic Graph
- Exactly one leaf for each trapezoid
- Inner nodes have out-degree 2.
- Inner nodes: x-nodes: labeled with an endpoint.
- Inner nodes: y-nodes: labeled with a segment.
- A query point starts at the root and traverse the structure until it reaches a leaf that contains q.





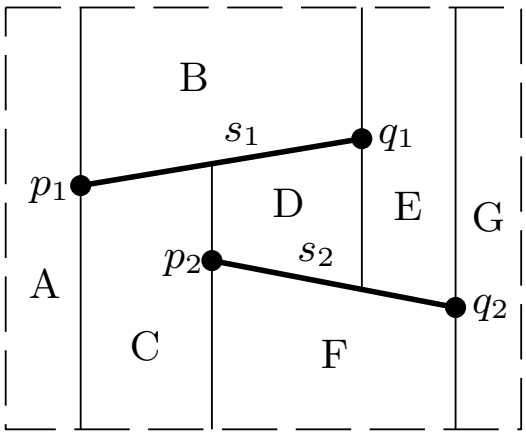


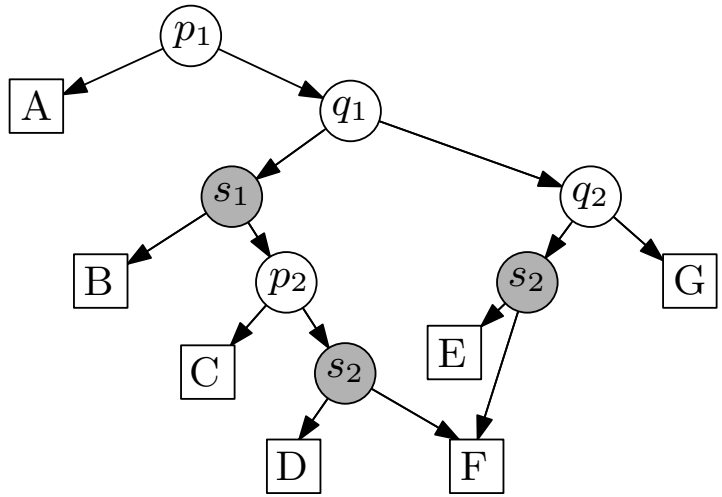
How to construct a Trapezoidal Map?

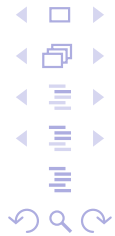
A Randomized Incremental Algorithm

Point Location Data Structure (search structure)

- Directed Acyclic Graph
- Exactly one leaf for each trapezoid
- Inner nodes have out-degree 2.
- Inner nodes: x-nodes: labeled with an endpoint.
- Inner nodes: y-nodes: labeled with a segment.
- A query point starts at the root and traverse the structure until it reaches a leaf that contains q.





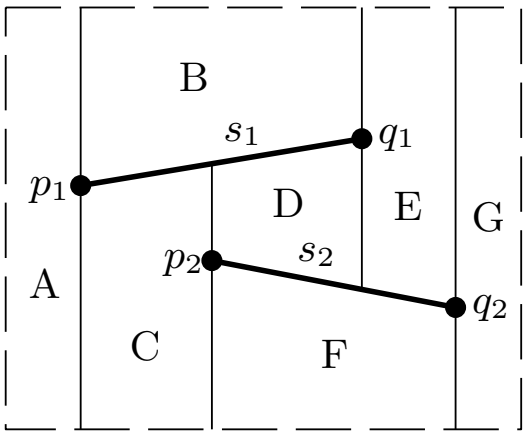


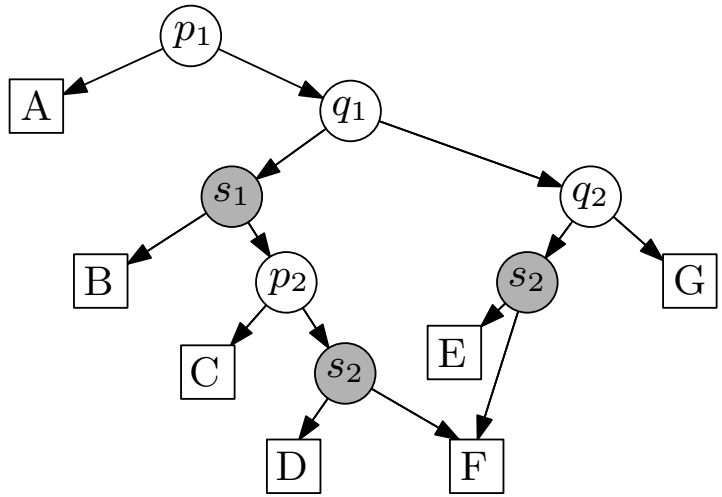
How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Point Location Data Structure (search structure)

- Directed Acyclic Graph
- Exactly one leaf for each trapezoid
- Inner nodes have out-degree 2.
- Inner nodes: x-nodes: labeled with an endpoint.
- Inner nodes: y-nodes: labeled with a segment.
- A query point starts at the root and traverse the structure until it reaches a leaf that contains q.



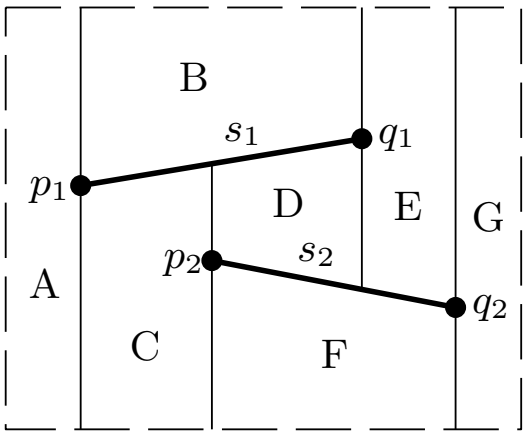


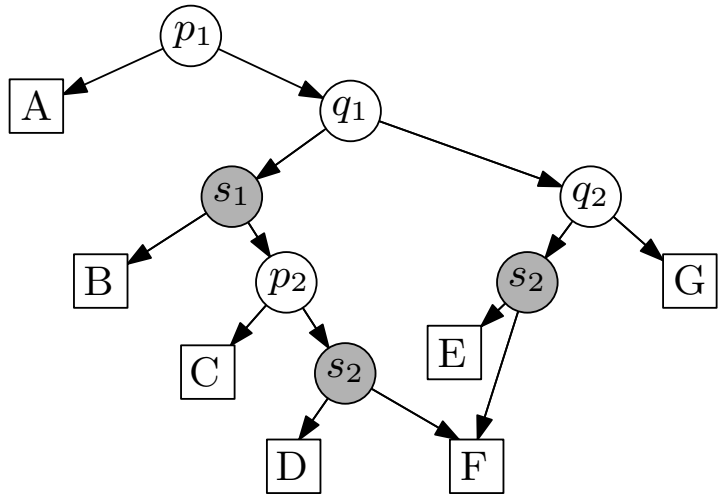
How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Point Location Data Structure (search structure)

- Directed Acyclic Graph
- Exactly one leaf for each trapezoid
- Inner nodes have out-degree 2.
- Inner nodes: x-nodes: labeled with an endpoint.
- Inner nodes: y-nodes: labeled with a segment.
- A query point starts at the root and traverse the structure until it reaches a leaf that contains q.



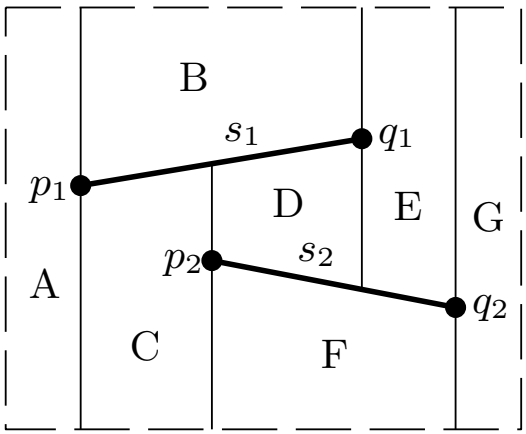


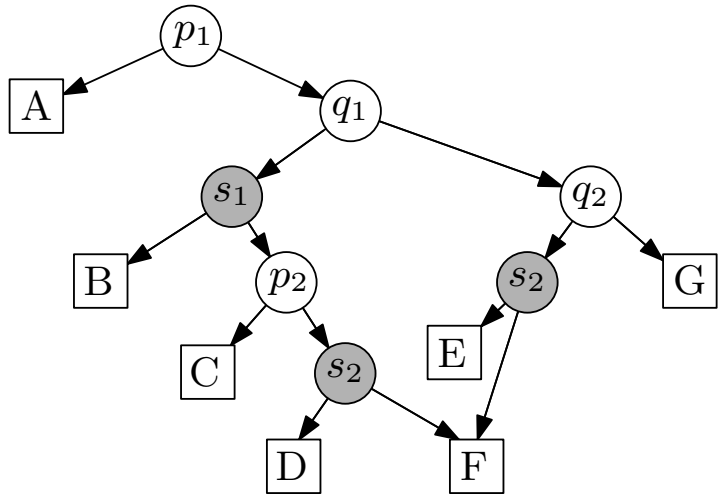
How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Point Location Data Structure (search structure)

- Directed Acyclic Graph
- Exactly one leaf for each trapezoid
- Inner nodes have out-degree 2.
- Inner nodes: x-nodes: labeled with an endpoint.
- Inner nodes: y-nodes: labeled with a segment.
- A query point starts at the root and traverse the structure until it reaches a leaf that contains q.



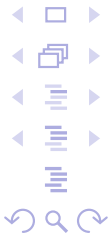


How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

The Trapezoid Construction Algorithm:

- Incremental: Add segments one-by-one and update the structure.
- Randomized: Add segments in Random orderings.



How to construct a Trapezoidal Map?

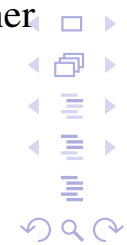
A Randomized Incremental Algorithm

The Trapezoid Construction Algorithm:

- Incremental: Add segments one-by-one and update the structure.
- Randomized: Add segments in Random orderings.

Algorithm TRAPEZOIDALMAP(S)
Input. A set S of n non-crossing line segments.
Output. The trapezoidal map $\mathcal{T}(S)$ and a search structure $\mathcal{D}$ for $\mathcal{T}(S)$ in a bounding box.

1. Determine a bounding box R that contains all segments of S, and initialize the trapezoidal map structure $\mathcal{T}$ and search structure $\mathcal{D}$ for it.
2. Compute a random permutation $s_1, s_2, \ldots, s_n$ of the elements of S.
3. **for** $i \leftarrow 1$ **to** n
4. **do** Find the set $\Delta_0, \Delta_1, \ldots, \Delta_k$ of trapezoids in $\mathcal{T}$ properly intersected by s_i.
5. Remove $\Delta_0, \Delta_1, \ldots, \Delta_k$ from $\mathcal{T}$ and replace them by the new trapezoids that appear because of the insertion of s_i.
6. Remove the leaves for $\Delta_0, \Delta_1, \ldots, \Delta_k$ from $\mathcal{D}$, and create leaves for the new trapezoids. Link the new leaves to the existing inner nodes by adding some new inner nodes, as explained below.

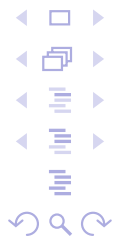


How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Algorithm TRAPEZOIDALMAP(S)
Input. A set S of n non-crossing line segments.
Output. The trapezoidal map $\mathcal{T}(S)$ and a search structure $\mathcal{D}$ for $\mathcal{T}(S)$ in a bounding box.

1. Determine a bounding box R that contains all segments of S, and initialize the trapezoidal map structure $\mathcal{T}$ and search structure $\mathcal{D}$ for it.
2. Compute a random permutation $s_1, s_2, \ldots, s_n$ of the elements of S.
3. **for** $i \leftarrow 1$ **to** n
4. **do** Find the set $\Delta_0, \Delta_1, \ldots, \Delta_k$ of trapezoids in $\mathcal{T}$ properly intersected by s_i.
5. Remove $\Delta_0, \Delta_1, \ldots, \Delta_k$ from $\mathcal{T}$ and replace them by the new trapezoids that appear because of the insertion of s_i.
6. Remove the leaves for $\Delta_0, \Delta_1, \ldots, \Delta_k$ from $\mathcal{D}$, and create leaves for the new trapezoids. Link the new leaves to the existing inner nodes by adding some new inner nodes, as explained below.

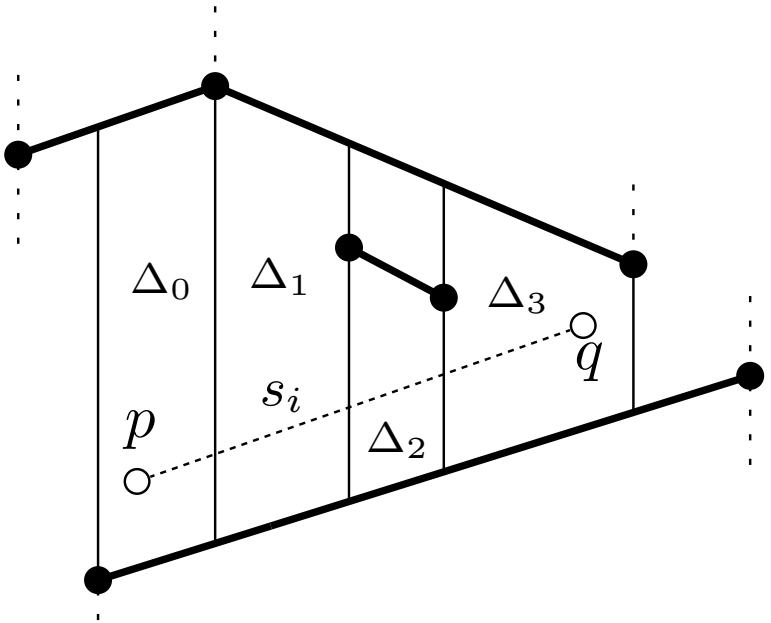


How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Algorithm TRAPEZOIDALMAP(S)
Input. A set S of n non-crossing line segments.
Output. The trapezoidal map $\mathcal{T}(S)$ and a search structure $\mathcal{D}$ for $\mathcal{T}(S)$ in a bounding box.

1. Determine a bounding box R that contains all segments of S, and initialize the trapezoidal map structure $\mathcal{T}$ and search structure $\mathcal{D}$ for it.
2. Compute a random permutation $s_1, s_2, \ldots, s_n$ of the elements of S.
3. **for** $i \leftarrow 1$ **to** n
4. **do** Find the set $\Delta_0, \Delta_1, \ldots, \Delta_k$ of trapezoids in $\mathcal{T}$ properly intersected by s_i.
5. Remove $\Delta_0, \Delta_1, \ldots, \Delta_k$ from $\mathcal{T}$ and replace them by the new trapezoids that appear because of the insertion of s_i.
6. Remove the leaves for $\Delta_0, \Delta_1, \ldots, \Delta_k$ from $\mathcal{D}$, and create leaves for the new trapezoids. Link the new leaves to the existing inner nodes by adding some new inner nodes, as explained below.

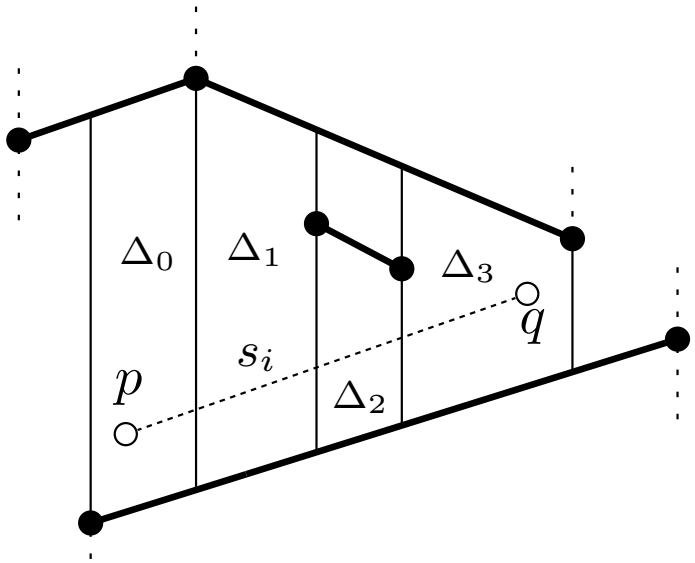


How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

The Trapezoid Construction Algorithm:

- Δ_0 is the trapezoid that contains the left endpoint of s_i, say p.
- Δ_{j+1} is one of right neighbors of Δ_j.
- In searching p in the structure:
  If p lie on a vertical line: go to the right.
  If p lies on a segment s of a y-node: If $slope(s_i) > slope(s)$ then p is above s.

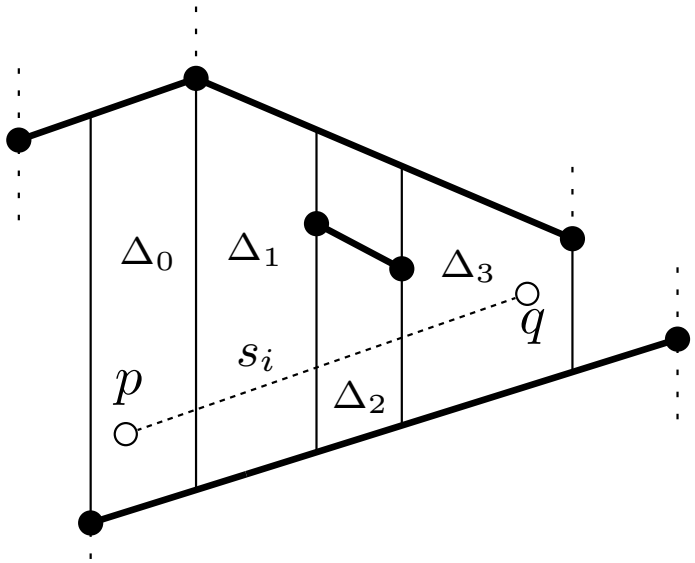


How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

The Trapezoid Construction Algorithm:

- Δ_0 is the trapezoid that contains the left endpoint of s_i, say p.
- Δ_{j+1} is one of right neighbors of Δ_j.
- In searching p in the structure:
  If p lie on a vertical line: go to the right.
  If p lies on a segment s of a y-node: If $slope(s_i) > slope(s)$ then p is above s.

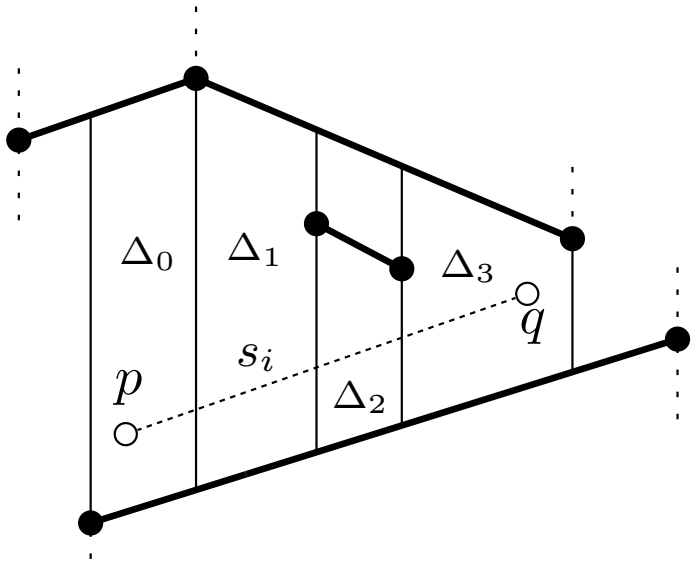


How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

The Trapezoid Construction Algorithm:

- Δ_0 is the trapezoid that contains the left endpoint of s_i, say p.
- Δ_{j+1} is one of right neighbors of Δ_j.
- In searching p in the structure:
  If p lie on a vertical line: go to the right.
  If p lies on a segment s of a y-node: If $slope(s_i) > slope(s)$ then p is above s.

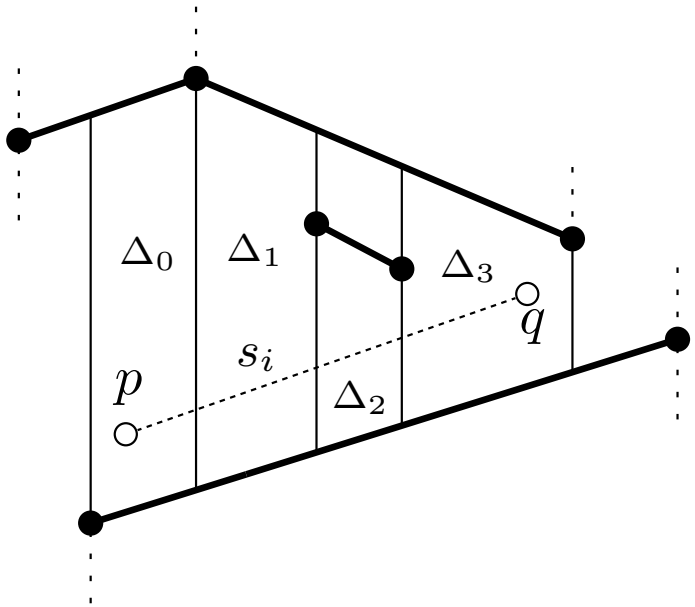


How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

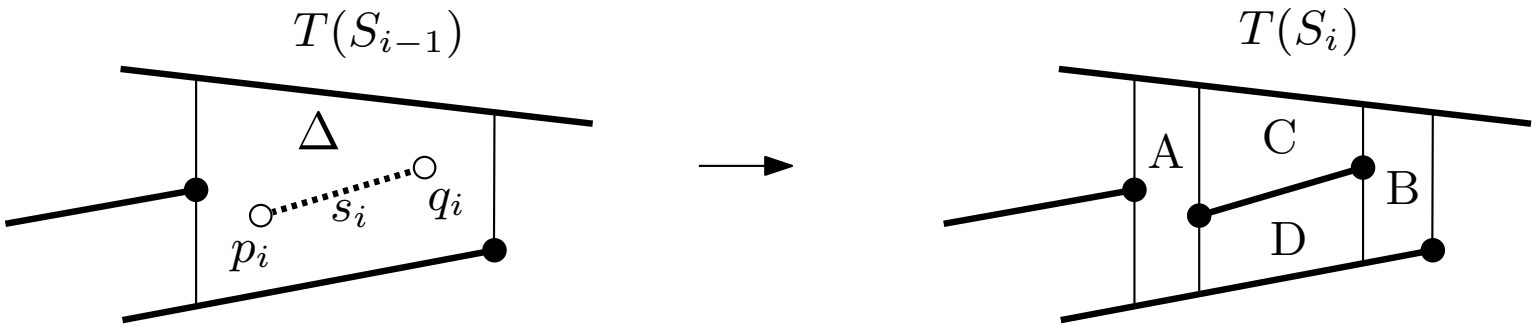
Algorithm FOLLOWSEGMENT($\mathcal{T}, \mathcal{D}, s_i$)
Input. A trapezoidal map $\mathcal{T}$, a search structure $\mathcal{D}$ for $\mathcal{T}$, and a new segment s_i.
Output. The sequence $\Delta_0, \ldots, \Delta_k$ of trapezoids intersected by s_i.

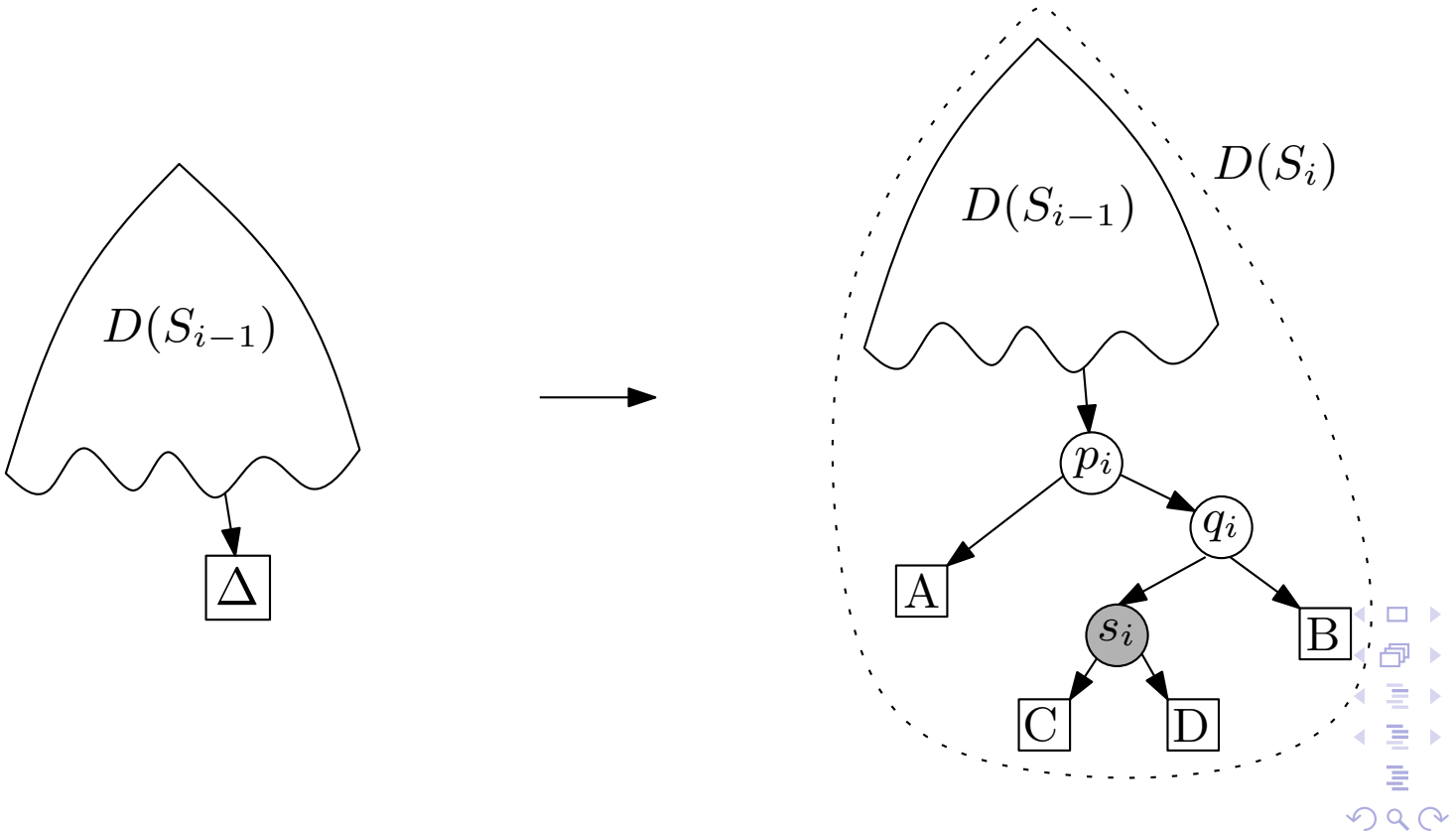
1. Let p and q be the left and right endpoint of s_i.
2. Search with p in the search structure $\mathcal{D}$ to find Δ_0.
3. $j \leftarrow 0$;
4. **while** q lies to the right of $rightp(\Delta_j)$
5. **do if** $rightp(\Delta_j)$ lies above s_i
6. **then** Let Δ_{j+1} be the lower right neighbor of Δ_j.
7. **else** Let Δ_{j+1} be the upper right neighbor of Δ_j.
8. $j \leftarrow j+1$
9. **return** $\Delta_0, \Delta_1, \ldots, \Delta_j$



How to construct a Trapezoidal Map?

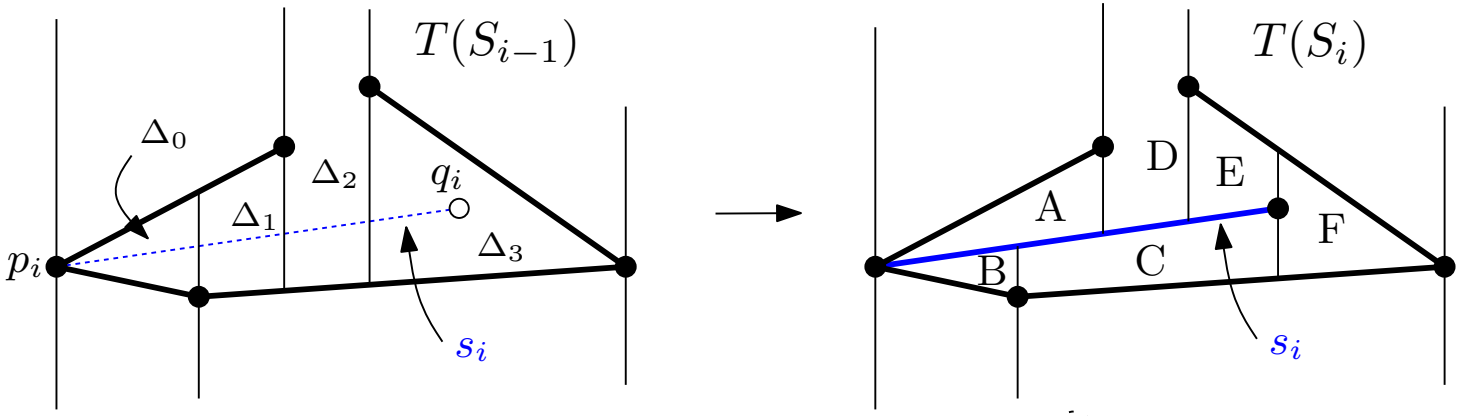
A Randomized Incremental Algorithm

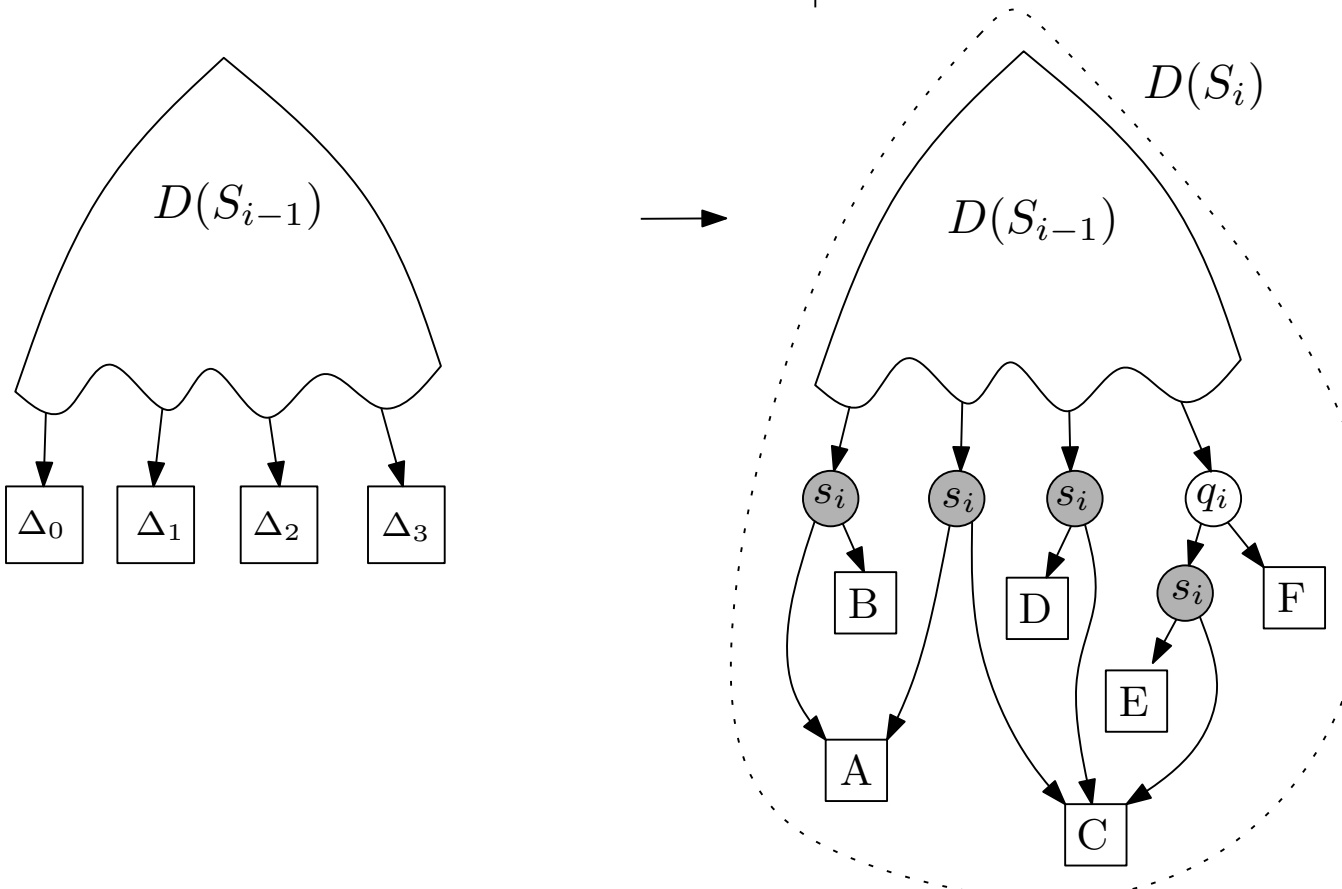




How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm





Is the ordering important? Yes!

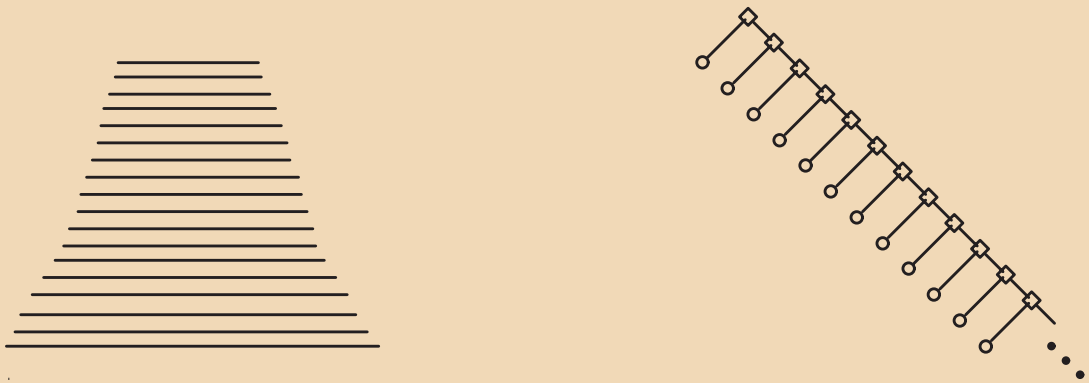


Figure 9: (Left) A set of segments in T which, if inserted bottom to top, leads to a worst-case query time. (Right) The resulting query structure.

Here the Random ordering helps!

Is the ordering important? Yes!

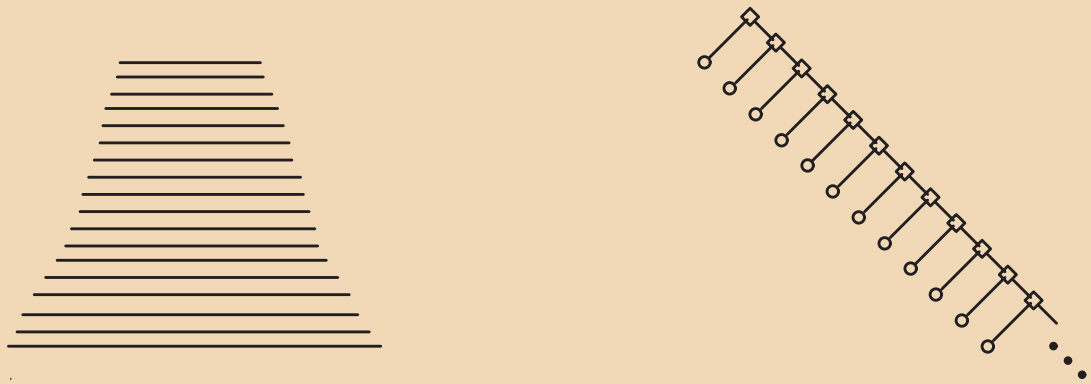


Figure 9: (Left) A set of segments in T which, if inserted bottom to top, leads to a worst-case query time. (Right) The resulting query structure.

Here the Random ordering helps!

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof.

- The query Complexity: Worst case: Depends on the depth of the DAG ($O(n)$).
- Expected case: Average on all orders of inserting segments!
- P: the search path.
  X_i: # vertices on P that created in step i of the algorithm.
- X_i is a random variable (its value depends on the order that the algorithm adds segments).
- $|P| = \sum_{i=1}^{n} X_i$

Yazd Univ.

Computational Geometry

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof.

- The query Complexity: Worst case: Depends on the depth of the DAG ($O(n)$).
- Expected case: Average on all orders of inserting segments!
- P: the search path.
  X_i: # vertices on P that created in step i of the algorithm.
- X_i is a random variable (its value depends on the order that the algorithm adds segments).
- $|P| = \sum_{i=1}^{n} X_i$

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof.

- The query Complexity: Worst case: Depends on the depth of the DAG ($O(n)$).
- Expected case: Average on all orders of inserting segments!
- P: the search path.
  X_i: # vertices on P that created in step i of the algorithm.
- X_i is a random variable (its value depends on the order that the algorithm adds segments).
- $|P| = \sum_{i=1}^{n} X_i$

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof.

- The query Complexity: Worst case: Depends on the depth of the DAG ($O(n)$).
- Expected case: Average on all orders of inserting segments!
- P: the search path.
  X_i: # vertices on P that created in step i of the algorithm.
- X_i is a random variable (its value depends on the order that the algorithm adds segments).
- $|P| = \sum_{i=1}^{n} X_i$

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof.

- The query Complexity: Worst case: Depends on the depth of the DAG ($O(n)$).
- Expected case: Average on all orders of inserting segments!
- P: the search path.
  X_i: # vertices on P that created in step i of the algorithm.
- X_i is a random variable (its value depends on the order that the algorithm adds segments).
- $|P| = \sum_{i=1}^{n} X_i$

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

$$
\begin{aligned}
Exp(|P|) &= Exp\left(\sum_{i=1}^{n} X_i\right) \\
&= \sum_{i=1}^{n} Exp(X_i) \quad (P_i = \text{Possibility of having a node in iteration } i) \\
&\leq \sum_{i=1}^{n} 3P_i \quad (\text{Since } X_i \leq 3)
\end{aligned}
$$

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

What is the bound for P_i?

- Notation: $\Delta_q(S_i)$: the trapezoid containing q in $T(S_i)$
- $P_i = Pr[\Delta_q(S_i) \neq \Delta_q(S_{i-1})]$.
- If $\Delta_q(S_i) \neq \Delta_q(S_{i-1})$, then $\Delta_q(S_i)$ is one of the trapezoids that generated in step i.
- all trapezoids Δ created in iteration i are adjacent to s_i.
- the segment that is inserted in iteration i: either $top(\Delta)$ or $bottom(\Delta)$ is s_i, or $leftp(\Delta)$ or $rightp(\Delta)$ is an endpoint of s_i.

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

What is the bound for P_i?

- Notation: $\Delta_q(S_i)$: the trapezoid containing q in $T(S_i)$
- $P_i = Pr[\Delta_q(S_i) \neq \Delta_q(S_{i-1})]$.
- If $\Delta_q(S_i) \neq \Delta_q(S_{i-1})$, then $\Delta_q(S_i)$ is one of the trapezoids that generated in step i.
- all trapezoids Δ created in iteration i are adjacent to s_i.
- the segment that is inserted in iteration i: either $top(\Delta)$ or $bottom(\Delta)$ is s_i, or $leftp(\Delta)$ or $rightp(\Delta)$ is an endpoint of s_i.

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

What is the bound for P_i?

- Notation: $\Delta_q(S_i)$: the trapezoid containing q in $T(S_i)$
- $P_i = Pr[\Delta_q(S_i) \neq \Delta_q(S_{i-1})]$.
- If $\Delta_q(S_i) \neq \Delta_q(S_{i-1})$, then $\Delta_q(S_i)$ is one of the trapezoids that generated in step i.
- all trapezoids Δ created in iteration i are adjacent to s_i.
- the segment that is inserted in iteration i: either $top(\Delta)$ or $bottom(\Delta)$ is s_i, or $leftp(\Delta)$ or $rightp(\Delta)$ is an endpoint of s_i.

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

What is the bound for P_i?

- Notation: $\Delta_q(S_i)$: the trapezoid containing q in $T(S_i)$
- $P_i = Pr[\Delta_q(S_i) \neq \Delta_q(S_{i-1})]$.
- If $\Delta_q(S_i) \neq \Delta_q(S_{i-1})$, then $\Delta_q(S_i)$ is one of the trapezoids that generated in step i.
- all trapezoids Δ created in iteration i are adjacent to s_i.
- the segment that is inserted in iteration i: either $top(\Delta)$ or $bottom(\Delta)$ is s_i, or $leftp(\Delta)$ or $rightp(\Delta)$ is an endpoint of s_i.

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

What is the bound for P_i?

- Notation: $\Delta_q(S_i)$: the trapezoid containing q in $T(S_i)$
- $P_i = Pr[\Delta_q(S_i) \neq \Delta_q(S_{i-1})]$.
- If $\Delta_q(S_i) \neq \Delta_q(S_{i-1})$, then $\Delta_q(S_i)$ is one of the trapezoids that generated in step i.
- all trapezoids Δ created in iteration i are adjacent to s_i.
- the segment that is inserted in iteration i: either $top(\Delta)$ or $bottom(\Delta)$ is s_i, or $leftp(\Delta)$ or $rightp(\Delta)$ is an endpoint of s_i.

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

What is the bound for P_i?

- For a fixed set $S_i \subset S$, $T(S_i)$, and therefore $\Delta_q(S_i)$, are uniquely defined (does not depend on the order).
- To bound the probability that the trapezoid containing q has changed due to the insertion of s_i, we use backward analysis.
- Backward Analysis: Consider $T(S_i)$ and look at the probability that $\Delta_q(S_i)$ disappears from the trapezoidal map when we remove the segment s_i.

Yazd Univ.

Computational Geometry

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

What is the bound for P_i?

- For a fixed set $S_i \subset S$, $T(S_i)$, and therefore $\Delta_q(S_i)$, are uniquely defined (does not depend on the order).
- To bound the probability that the trapezoid containing q has changed due to the insertion of s_i, we use backward analysis.
- Backward Analysis: Consider $T(S_i)$ and look at the probability that $\Delta_q(S_i)$ disappears from the trapezoidal map when we remove the segment s_i.

Yazd Univ.

Computational Geometry

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

What is the bound for P_i?

- For a fixed set $S_i \subset S$, $T(S_i)$, and therefore $\Delta_q(S_i)$, are uniquely defined (does not depend on the order).
- To bound the probability that the trapezoid containing q has changed due to the insertion of s_i, we use backward analysis.
- Backward Analysis: Consider $T(S_i)$ and look at the probability that $\Delta_q(S_i)$ disappears from the trapezoidal map when we remove the segment s_i.

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

What is the bound for P_i?

- Backward Analysis: Consider $T(S_i)$ and look at the probability that $\Delta_q(S_i)$ disappears from the trapezoidal map when we remove the segment s_i.
- $\Delta_q(S_i)$ disappears $\iff$ one of $top(\Delta_q(S_i))$, $bottom(\Delta_q(S_i))$, $leftp(\Delta_q(S_i))$, or $rightp(\Delta_q(S_i))$ disappears with the removal of s_i.
- What is the probability that $top(\Delta_q(S_i))$ disappears? S_i inserted in random order, so every segment in S_i is equally likely to be s_i. So, the probability that s_i happens to be $top(\Delta_q(S_i))$ is $\frac{1}{i}$.

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

What is the bound for P_i?

- Backward Analysis: Consider $T(S_i)$ and look at the probability that $\Delta_q(S_i)$ disappears from the trapezoidal map when we remove the segment s_i.
- $\Delta_q(S_i)$ disappears $\iff$ one of $top(\Delta_q(S_i))$, $bottom(\Delta_q(S_i))$, $leftp(\Delta_q(S_i))$, or $rightp(\Delta_q(S_i))$ disappears with the removal of s_i.
- What is the probability that $top(\Delta_q(S_i))$ disappears? S_i inserted in random order, so every segment in S_i is equally likely to be s_i. So, the probability that s_i happens to be $top(\Delta_q(S_i))$ is $\frac{1}{i}$.

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

What is the bound for P_i?

- Backward Analysis: Consider $T(S_i)$ and look at the probability that $\Delta_q(S_i)$ disappears from the trapezoidal map when we remove the segment s_i.
- $\Delta_q(S_i)$ disappears $\iff$ one of $top(\Delta_q(S_i))$, $bottom(\Delta_q(S_i))$, $leftp(\Delta_q(S_i))$, or $rightp(\Delta_q(S_i))$ disappears with the removal of s_i.
- What is the probability that $top(\Delta_q(S_i))$ disappears? S_i inserted in random order, so every segment in S_i is equally likely to be s_i. So, the probability that s_i happens to be $top(\Delta_q(S_i))$ is $\frac{1}{i}$.

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

What is the bound for P_i?

- What is the probability that $top(\Delta_q(S_i))$ disappears? S_i inserted in random order, so every segment in S_i is equally likely to be s_i. So, the probability that s_i happens to be $top(\Delta_q(S_i))$ is $\frac{1}{i}$.
- Same argument works for $bottom, rightp, leftp$.

$$\begin{aligned} P_i &= Pr[\Delta_q(S_i) \neq \Delta_q(S_{i-1})] \\ &= Pr[\Delta_q(S_i) \notin T(S_{i-1})] \\ &\leq \frac{4}{i}. \end{aligned}$$

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

What is the bound for P_i?

- What is the probability that $top(\Delta_q(S_i))$ disappears? S_i inserted in random order, so every segment in S_i is equally likely to be s_i. So, the probability that s_i happens to be $top(\Delta_q(S_i))$ is $\frac{1}{i}$.
- Same argument works for $bottom, rightp, leftp$.

$$\begin{aligned} P_i &= Pr[\Delta_q(S_i) \neq \Delta_q(S_{i-1})] \\ &= Pr[\Delta_q(S_i) \notin T(S_{i-1})] \\ &\leq \frac{4}{i}. \end{aligned}$$

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

What is the bound for P_i?

- What is the probability that $top(\Delta_q(S_i))$ disappears? S_i inserted in random order, so every segment in S_i is equally likely to be s_i. So, the probability that s_i happens to be $top(\Delta_q(S_i))$ is $\frac{1}{i}$.
- Same argument works for $bottom, rightp, leftp$.

$$\begin{aligned} P_i &= Pr[\Delta_q(S_i) \neq \Delta_q(S_{i-1})] \\ &= Pr[\Delta_q(S_i) \notin T(S_{i-1})] \\ &\leq \frac{4}{i}. \end{aligned}$$

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

$$
\begin{aligned}
Exp(|P|) &= Exp\left(\sum_{i=1}^{n} X_i\right) \\
&= \sum_{i=1}^{n} Exp(X_i) \\
&\leq \sum_{i=1}^{n} 3P_i \\
&\leq \sum_{i=1}^{n} \frac{12}{i} \quad (\text{Since } P_i \leq 4/i) \\
&\leq 12(\ln n + 1). \quad \textit{Done!}
\end{aligned}
$$

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

Space Complexity:

- Size of D: # leaves + # internal nodes.
- # leaves= # trapezoids= $O(n)$.
- # internal nodes: $k_i - 1$ (k_i is # leaves generated in step i).
- Worst case: $k_i = O(i) \Rightarrow$
  Size of $D = O(n) + \sum_{i=1}^{n} i = O(n^2)$.
- Expected Size:
  $O(n) + E\left[\sum_{i=1}^{n}(k_i - 1)\right] = O(n) + E\left[\sum_{i=1}^{n}(k_i)\right]$.

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

Space Complexity:

- Size of D: # leaves + # internal nodes.
- # leaves= # trapezoids= $O(n)$.
- # internal nodes: $k_i - 1$ (k_i is # leaves generated in step i).
- Worst case: $k_i = O(i) \Rightarrow$
  Size of $D = O(n) + \sum_{i=1}^{n} i = O(n^2)$.
- Expected Size:
  $O(n) + E\left[\sum_{i=1}^{n}(k_i - 1)\right] = O(n) + E\left[\sum_{i=1}^{n}(k_i)\right]$.

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

Space Complexity:

- Size of D: # leaves + # internal nodes.
- # leaves= # trapezoids= $O(n)$.
- # internal nodes: $k_i - 1$ (k_i is # leaves generated in step i).
- Worst case: $k_i = O(i) \Rightarrow$
  Size of $D = O(n) + \sum_{i=1}^{n} i = O(n^2)$.
- Expected Size:
  $O(n) + E\left[\sum_{i=1}^{n}(k_i - 1)\right] = O(n) + E\left[\sum_{i=1}^{n}(k_i)\right]$.

Yazd Univ.

Computational Geometry

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

Space Complexity:

- Size of D: # leaves + # internal nodes.
- # leaves= # trapezoids= $O(n)$.
- # internal nodes: $k_i - 1$ (k_i is # leaves generated in step i).
- Worst case: $k_i = O(i) \Rightarrow$
  Size of $D = O(n) + \sum_{i=1}^{n} i = O(n^2)$.
- Expected Size:
  $O(n) + E\left[\sum_{i=1}^{n}(k_i - 1)\right] = O(n) + E\left[\sum_{i=1}^{n}(k_i)\right]$.

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

Space Complexity:

- Size of D: # leaves + # internal nodes.
- # leaves= # trapezoids= $O(n)$.
- # internal nodes: $k_i - 1$ (k_i is # leaves generated in step i).
- Worst case: $k_i = O(i) \Rightarrow$
  Size of $D = O(n) + \sum_{i=1}^{n} i = O(n^2)$.
- Expected Size:
  $O(n) + E\left[\sum_{i=1}^{n}(k_i - 1)\right] = O(n) + E\left[\sum_{i=1}^{n}(k_i)\right]$.

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

Space Complexity: (Cont.)

- Upper bound for $E(k_i)$?
- For $\Delta \in T(S_i)$ and $s \in S_i$,

$$\delta(\Delta, s) = \begin{cases} 1 & \text{if } \Delta \text{ disappears when } s \text{ is removed.} \\ 0 & \text{otherwise.} \end{cases}$$

- Since at most 4 segments can cause disappearance of a trapezoid

$$\sum_{s \in S_i} \sum_{\Delta \in T(S_i)} \delta(\Delta, s) \leq 4|T(S_i)| = O(i).$$

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

Space Complexity: (Cont.)

- Upper bound for $E(k_i)$?
- For $\Delta \in T(S_i)$ and $s \in S_i$,

$$\delta(\Delta, s) = \begin{cases} 1 & \text{if } \Delta \text{ disappears when } s \text{ is removed.} \\ 0 & \text{otherwise.} \end{cases}$$

- Since at most 4 segments can cause disappearance of a trapezoid

$$\sum_{s \in S_i} \sum_{\Delta \in T(S_i)} \delta(\Delta, s) \leq 4|T(S_i)| = O(i).$$

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

Space Complexity: (Cont.)

- Upper bound for $E(k_i)$?
- For $\Delta \in T(S_i)$ and $s \in S_i$,

$$\delta(\Delta, s) = \begin{cases} 1 & \text{if } \Delta \text{ disappears when } s \text{ is removed.} \\ 0 & \text{otherwise.} \end{cases}$$

- Since at most 4 segments can cause disappearance of a trapezoid

$$\sum_{s \in S_i} \sum_{\Delta \in T(S_i)} \delta(\Delta, s) \leq 4|T(S_i)| = O(i).$$

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

Space Complexity: (Cont.)

- k_i is # leaves generated in step i
- Upper bound for $E(k_i)$?
- For $E(k_i)$ we take average on all cases:

$$E(k_i) = \frac{1}{i} \sum_{s \in S_i} \sum_{\Delta \in T(S_i)} \delta(\Delta, s) \leq \frac{O(i)}{i} = O(1).$$

- Size= $O(n) + \sum_{i=1}^{n} E(k_i) = O(n)$.

Yazd Univ.

Computational Geometry

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

Space Complexity: (Cont.)

- k_i is # leaves generated in step i
- Upper bound for $E(k_i)$?
- For $E(k_i)$ we take average on all cases:

$$E(k_i) = \frac{1}{i} \sum_{s \in S_i} \sum_{\Delta \in T(S_i)} \delta(\Delta, s) \leq \frac{O(i)}{i} = O(1).$$

- Size= $O(n) + \sum_{i=1}^{n} E(k_i) = O(n)$.

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

Space Complexity: (Cont.)

- k_i is # leaves generated in step i
- Upper bound for $E(k_i)$?
- For $E(k_i)$ we take average on all cases:

$$E(k_i) = \frac{1}{i} \sum_{s \in S_i} \sum_{\Delta \in T(S_i)} \delta(\Delta, s) \leq \frac{O(i)}{i} = O(1).$$

- Size= $O(n) + \sum_{i=1}^{n} E(k_i) = O(n)$.

How to construct a Trapezoidal Map?

A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

Space Complexity: (Cont.)

- k_i is # leaves generated in step i
- Upper bound for $E(k_i)$?
- For $E(k_i)$ we take average on all cases:

$$E(k_i) = \frac{1}{i} \sum_{s \in S_i} \sum_{\Delta \in T(S_i)} \delta(\Delta, s) \leq \frac{O(i)}{i} = O(1).$$

- Size= $O(n) + \sum_{i=1}^{n} E(k_i) = O(n)$.

How to construct a Trapezoidal Map?

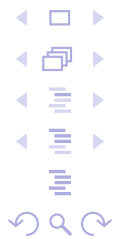
A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

Construction Time:

- Construction time: Time to insert s_i+Time to locate endpoint of $s_i = O(k_i) + O(\log i)$.
- Construction time: $O(1) + \sum_{i=1}^{n} (O(\log i) + O(E(k_i))) = O(n \log n)$.



How to construct a Trapezoidal Map?

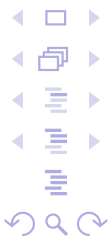
A Randomized Incremental Algorithm

Theorem 6.3 *Algorithm* TRAPEZOIDALMAP *computes the trapezoidal map* $\mathcal{T}(S)$ *of a set* S *of* n *line segments in general position and a search structure* $\mathcal{D}$ *for* $\mathcal{T}(S)$ *in* $O(n \log n)$ *expected time. The expected size of the search structure is* $O(n)$ *and for any query point* q *the expected query time is* $O(\log n)$.

Proof. (Cont.)

Construction Time:

- Construction time: Time to insert s_i+Time to locate endpoint of $s_i = O(k_i) + O(\log i)$.
- Construction time: $O(1) + \sum_{i=1}^{n} (O(\log i) + O(E(k_i))) = O(n \log n)$.



END.