Yan-Bin Jia, Iowa State University

Point Location & Trapezoidal Maps

Outline:

I. Query for the face containing a point

II. Trapezoidal map

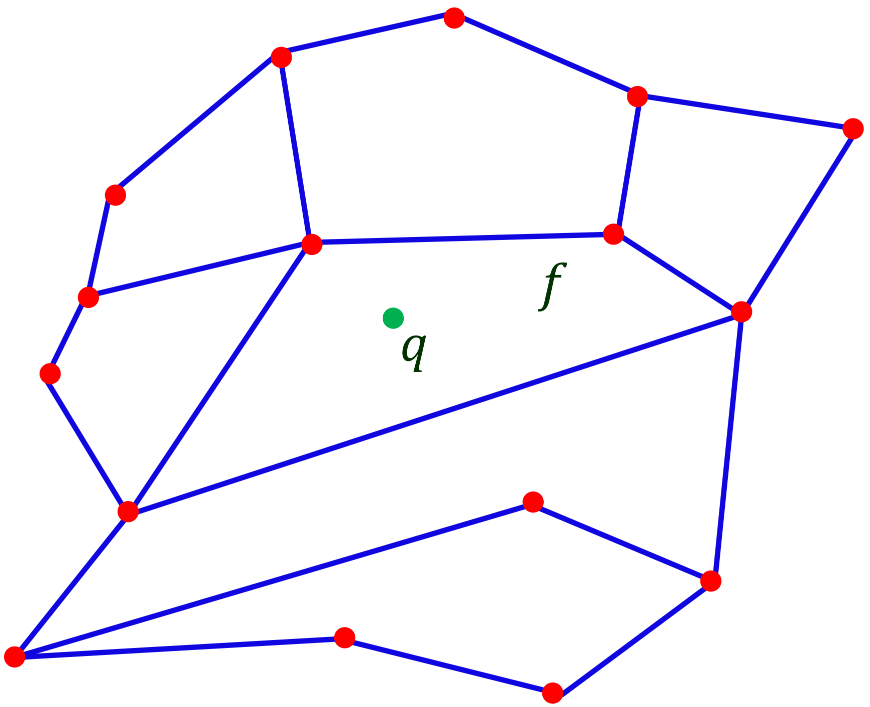
III. Geometric complexity of the map

IV. Data structure for search

I. Query for Containing Face

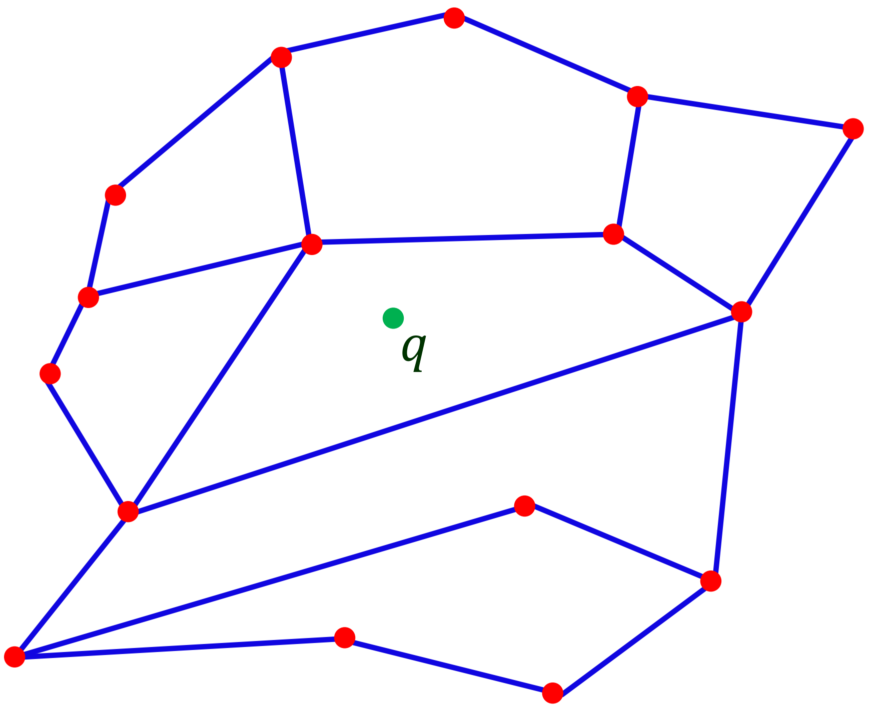
S: planar subdivision with n edges

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



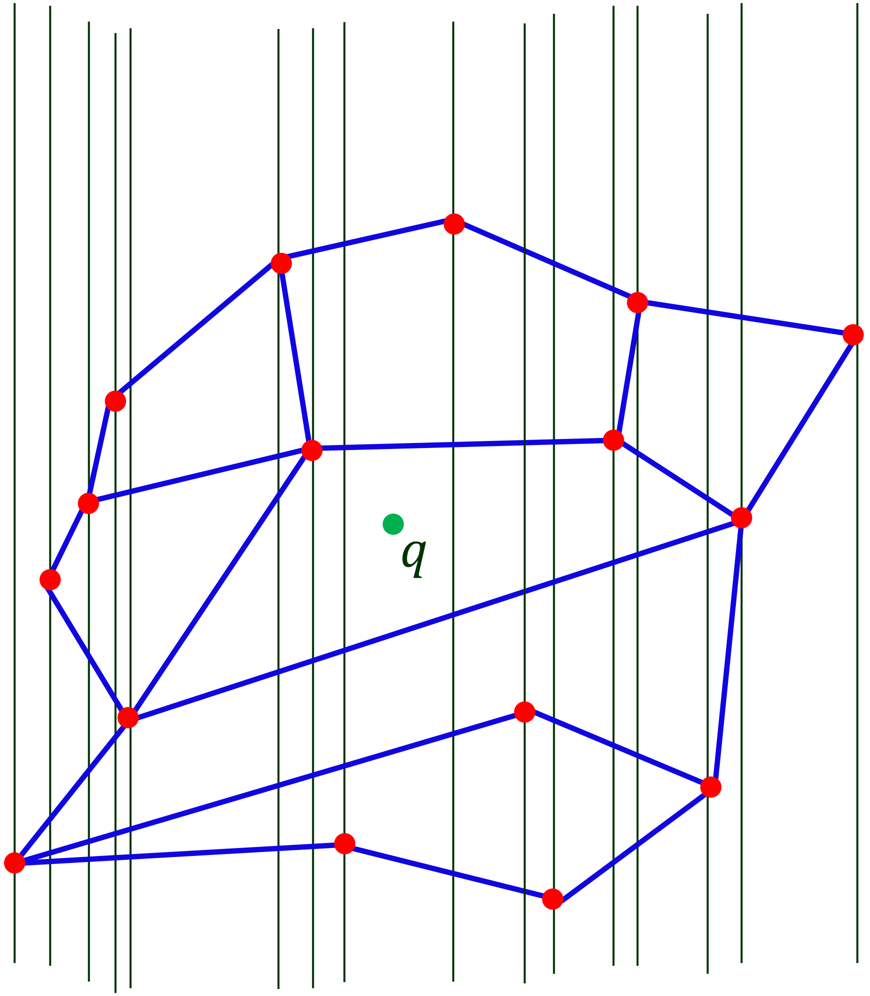
Simple Data Structure

Draw vertical lines through all the vertices.



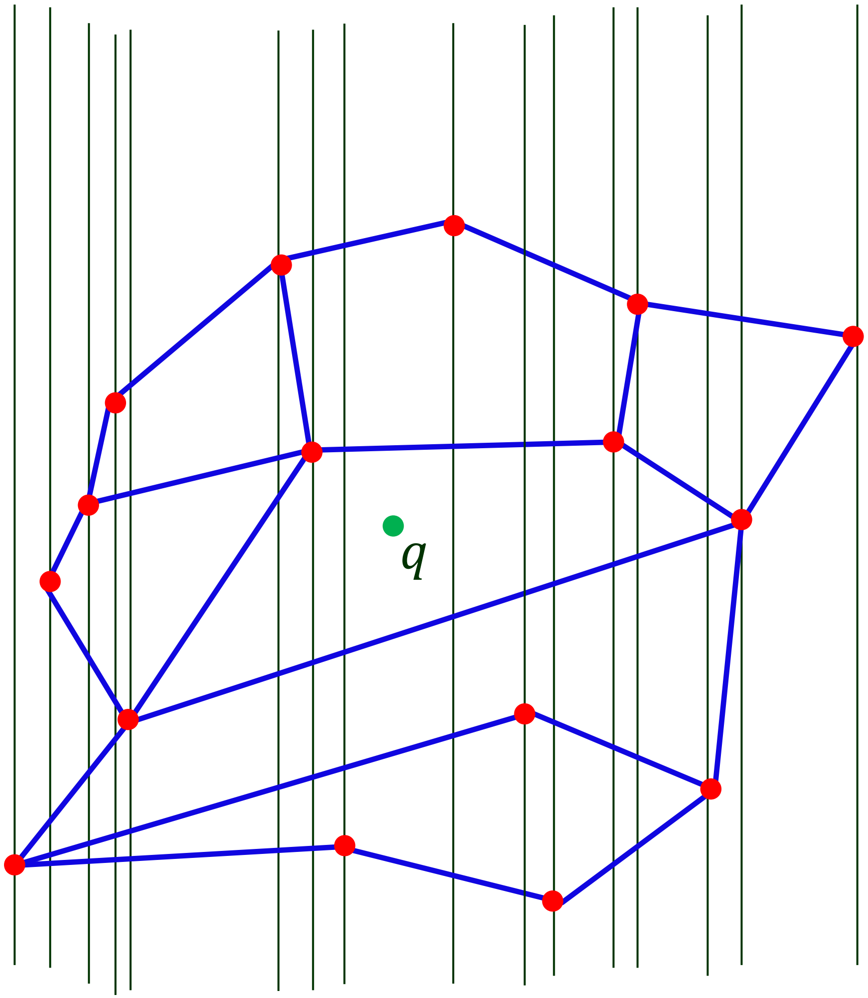
Simple Data Structure

Draw vertical lines through all the vertices.



Simple Data Structure

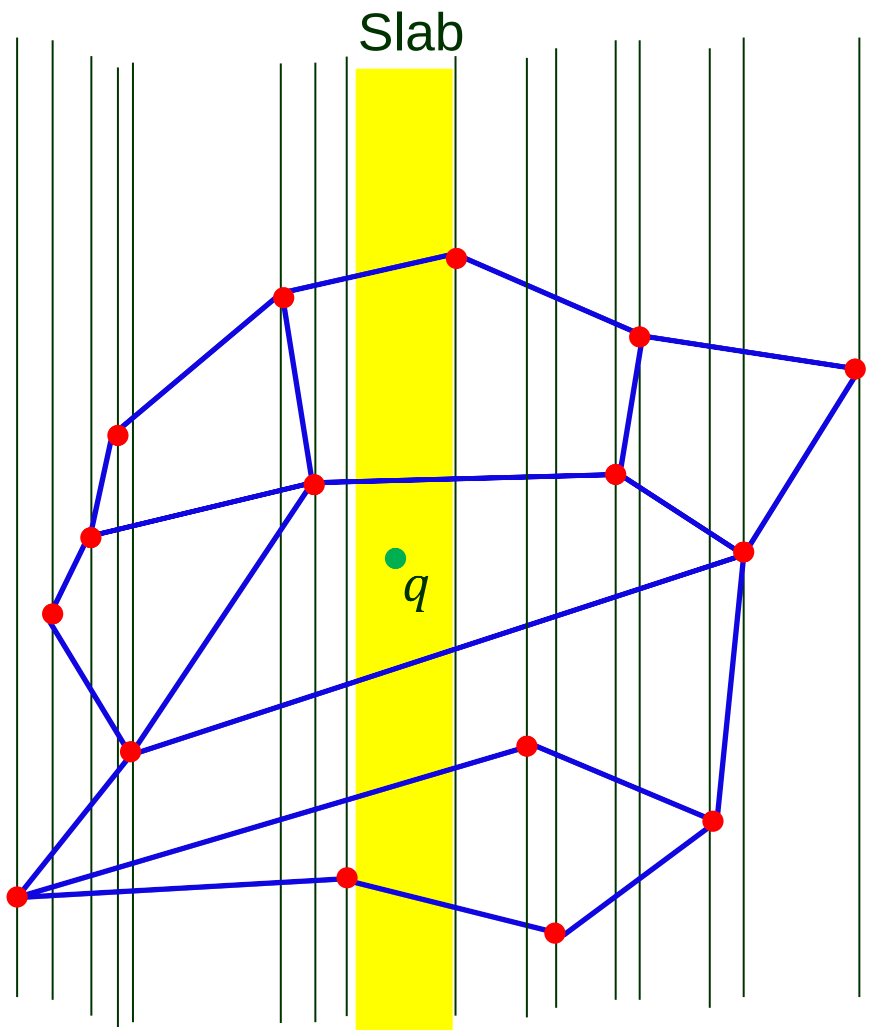
Draw vertical lines through all the vertices.



- Sort them by x-coordinate.

Simple Data Structure

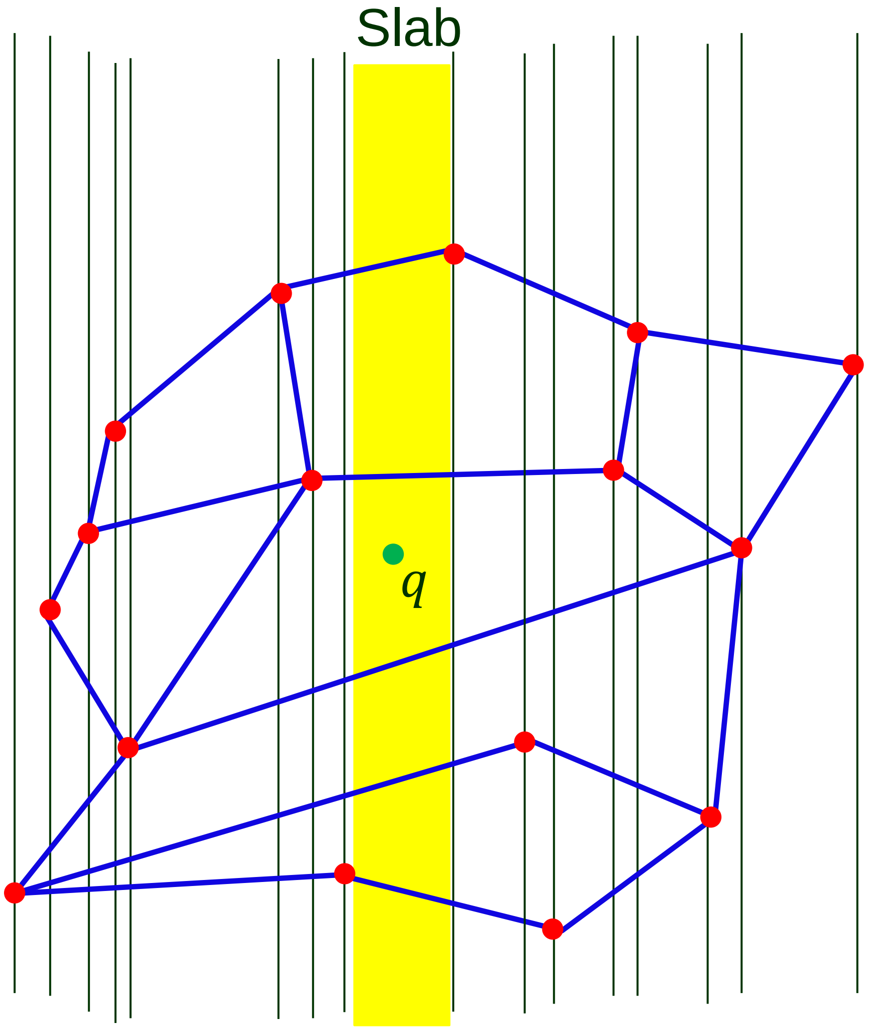
Draw vertical lines through all the vertices.



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

Simple Data Structure

Draw vertical lines through all the vertices.

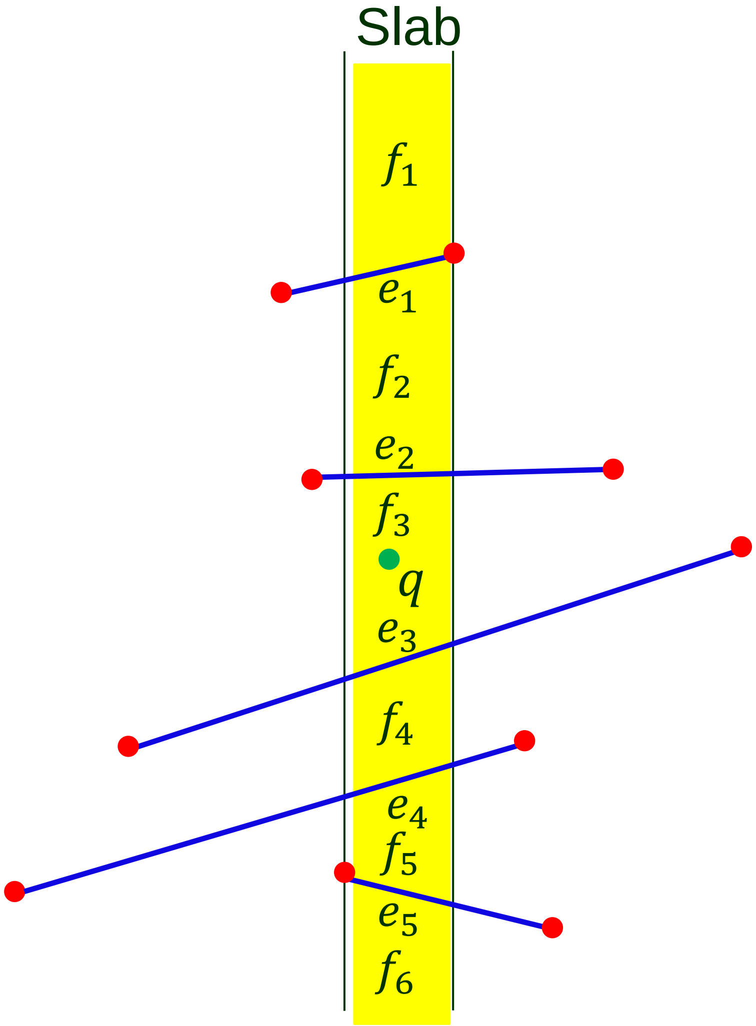


- Sort them by x-coordinate.
- Determine the slab containing q.

$O(\log n)$ time

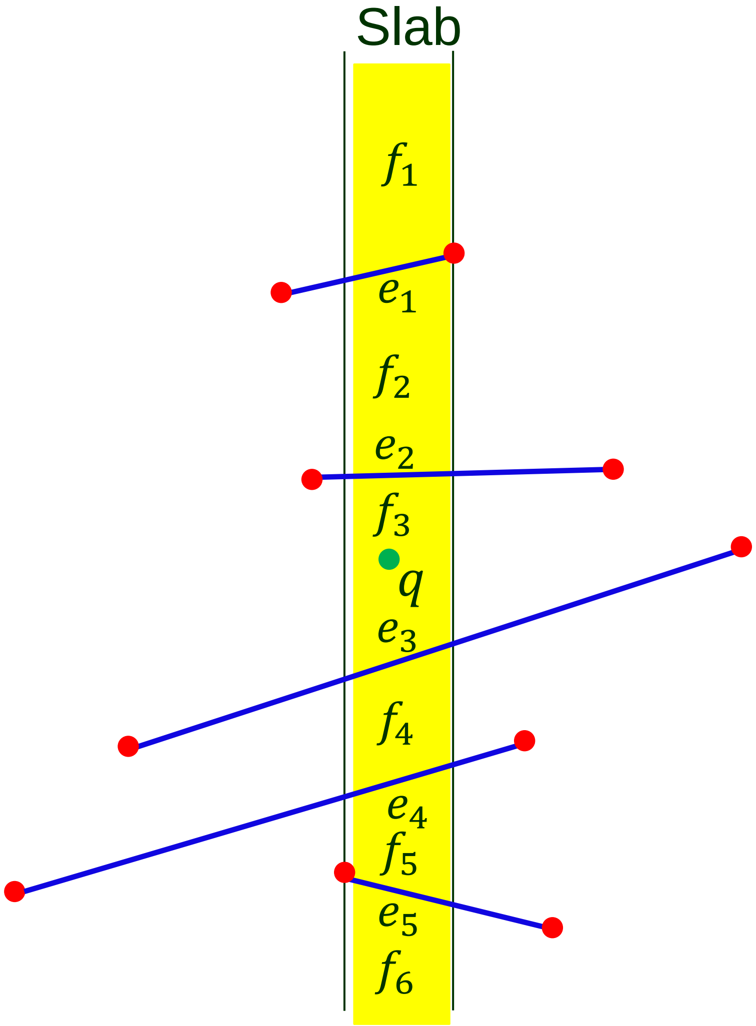
Slab

No vertices inside a slab



Slab

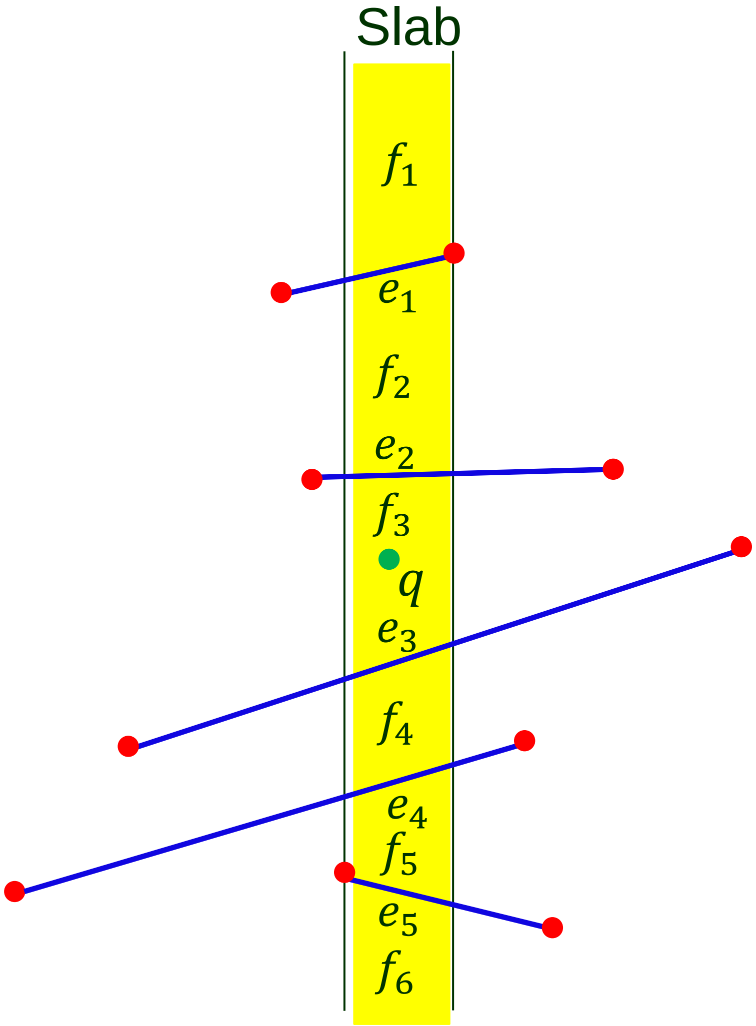
No vertices inside a slab



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

Slab

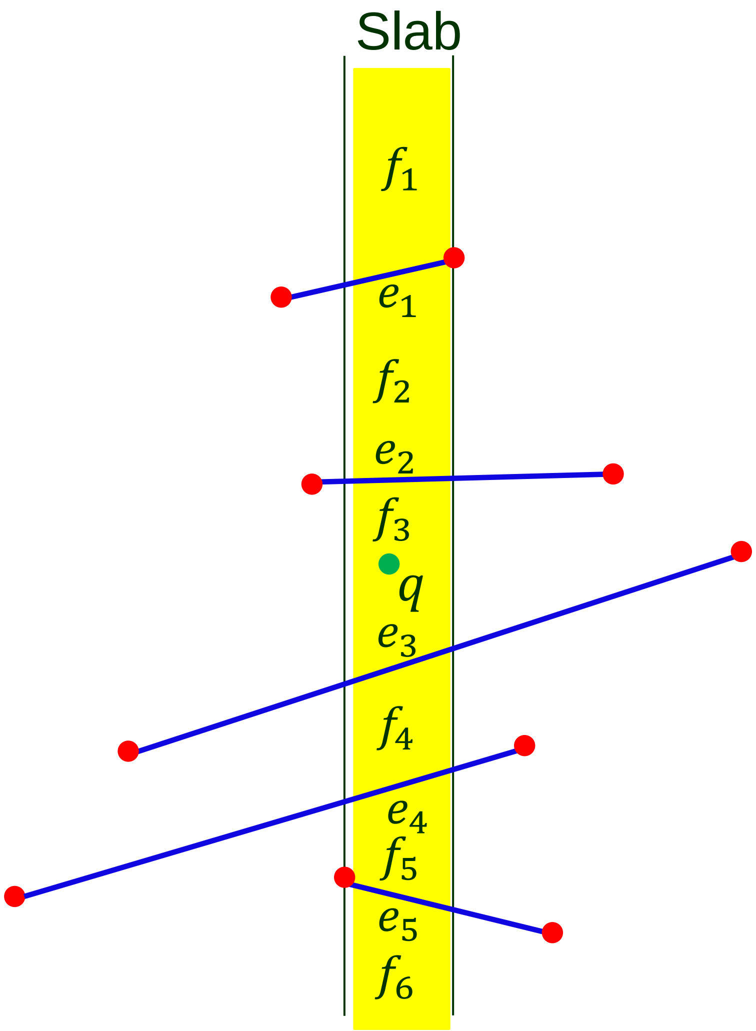
No vertices inside a slab



- Bounded by the vertical lines through two adjacent vertices in the sorted list.
- Edges don't cross each other.

Slab

No vertices inside a slab



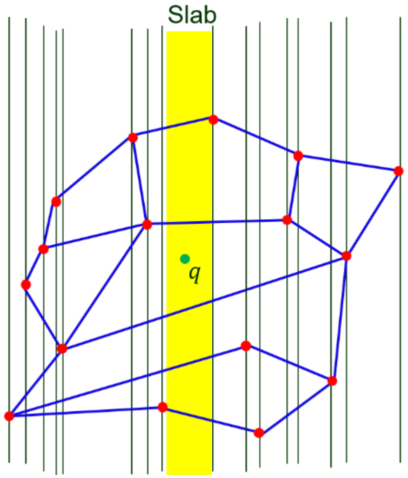
- Bounded by the vertical lines through two adjacent vertices in the sorted list.
- Edges don't cross each other.

♦ 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

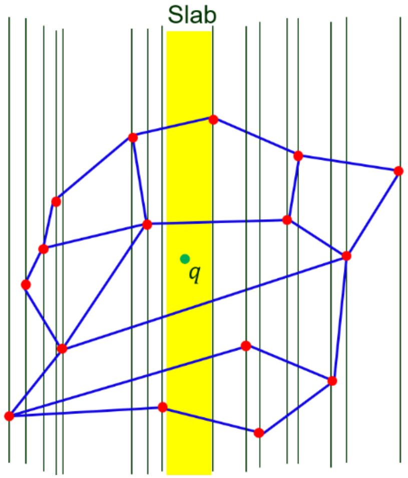
1. Determine the slab containing q.



Query Algorithm

1. Determine the slab containing q.

- Binary search with x-coordinate of q.

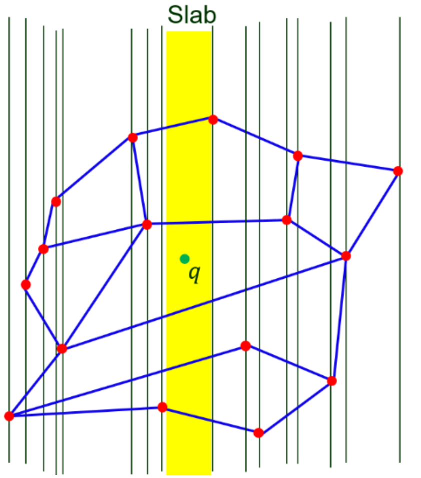


Query Algorithm

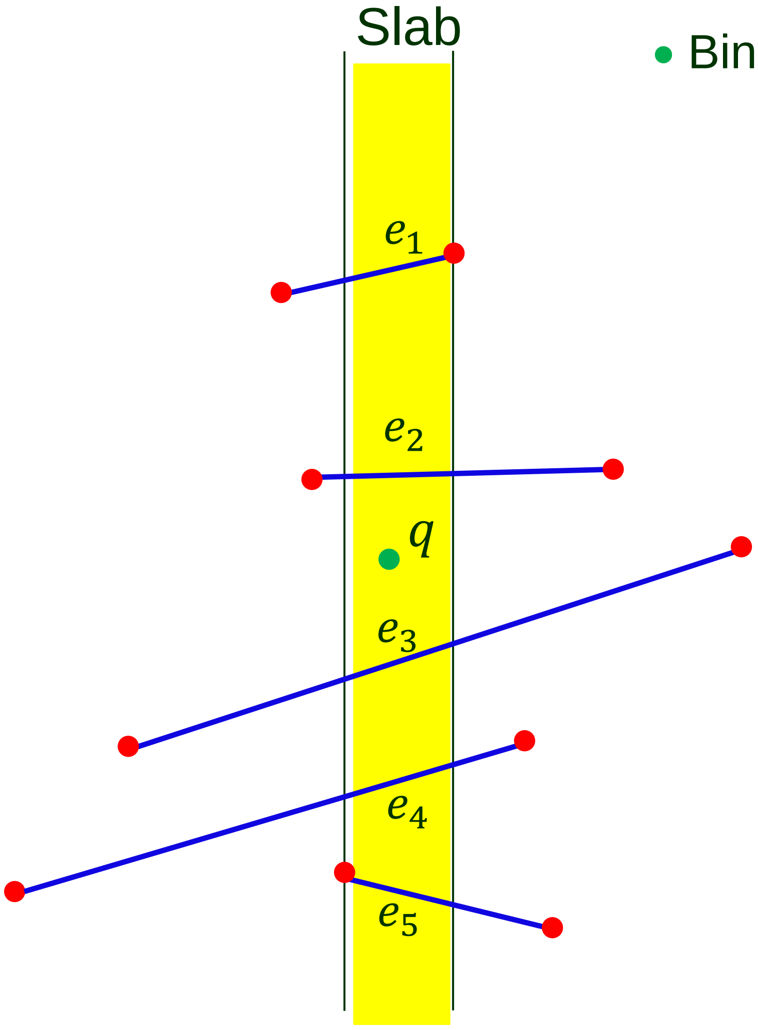
1. Determine the slab containing q.

- Binary search with x-coordinate of q.

$$O(\log n)$$

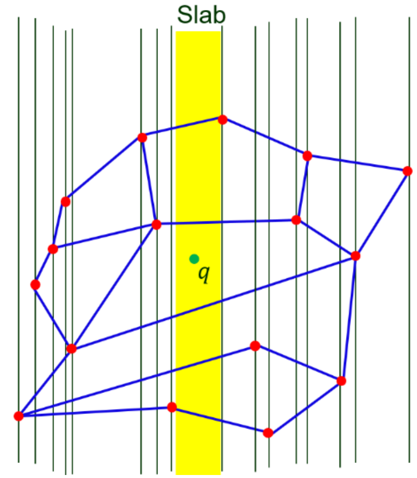


Query Algorithm



1. Determine the slab containing q.

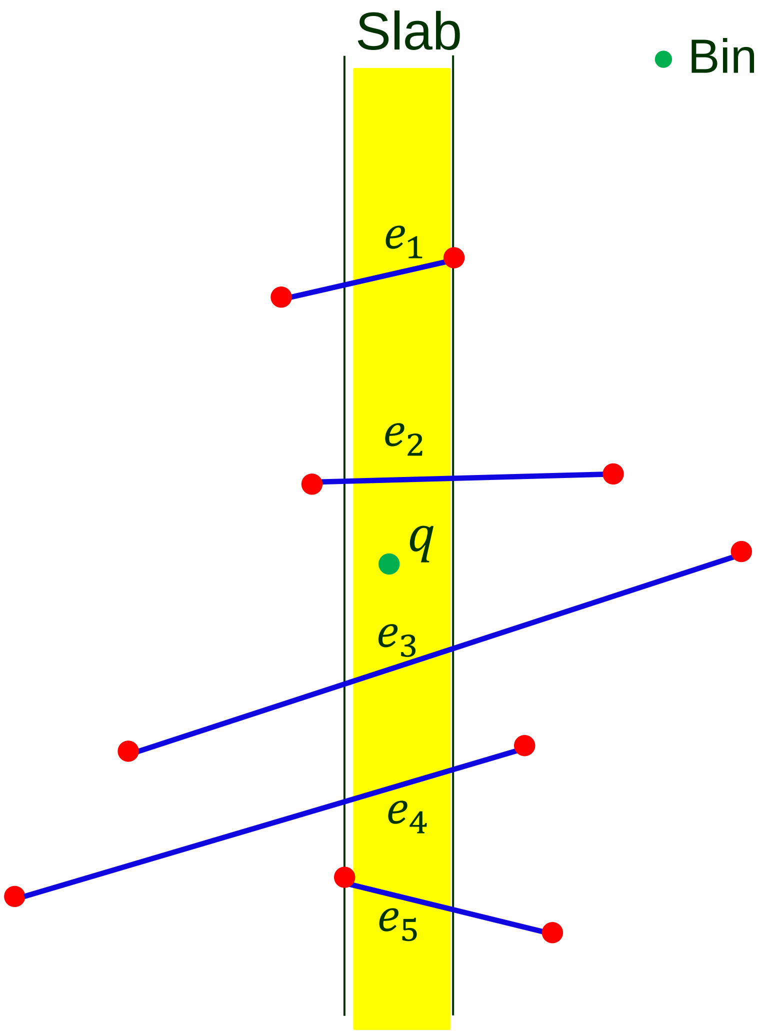
- Binary search with x-coordinate of q.



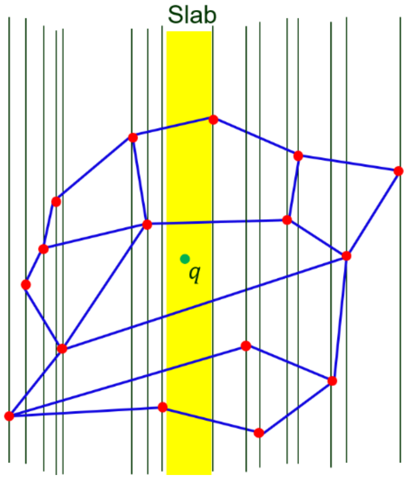
$O(\log n)$

2. Binary search within the slab.

Query Algorithm



1. Determine the slab containing q.



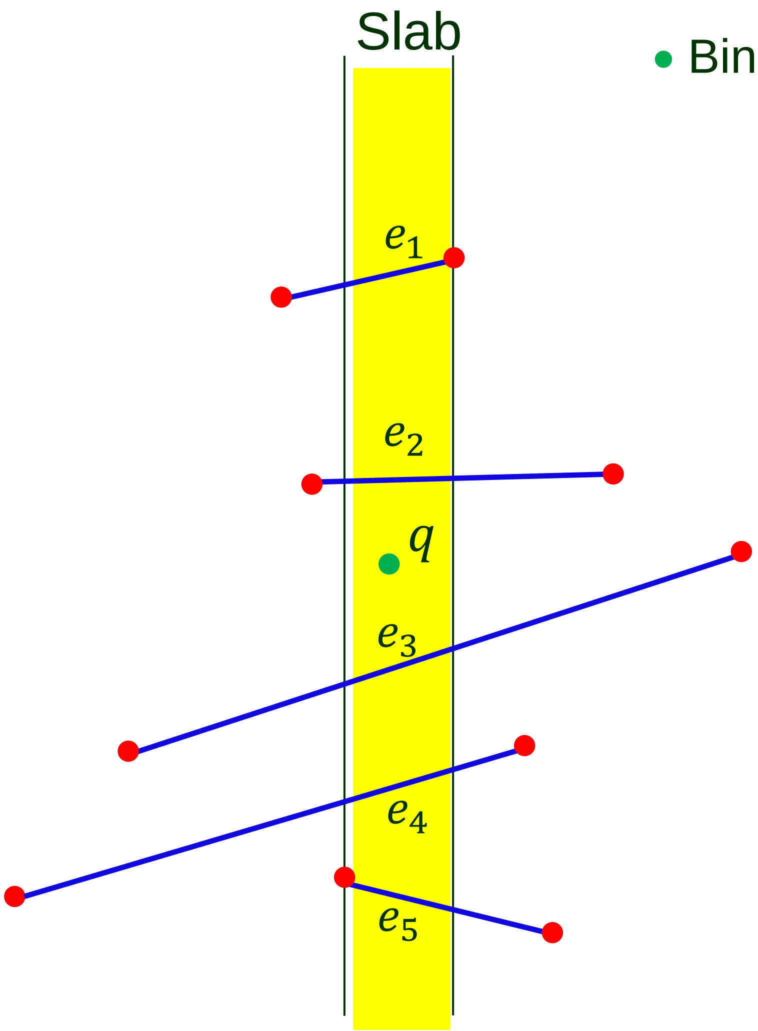
- Binary search with x-coordinate of q.

$O(\log n)$

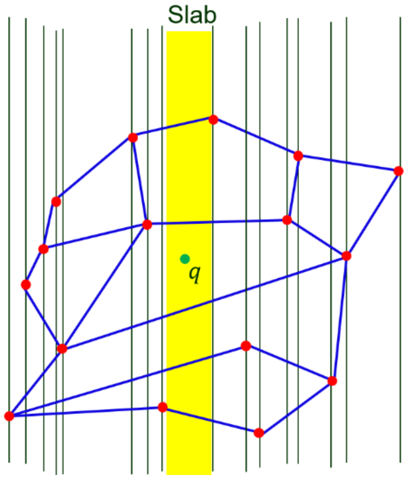
2. Binary search within the slab.

- 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.

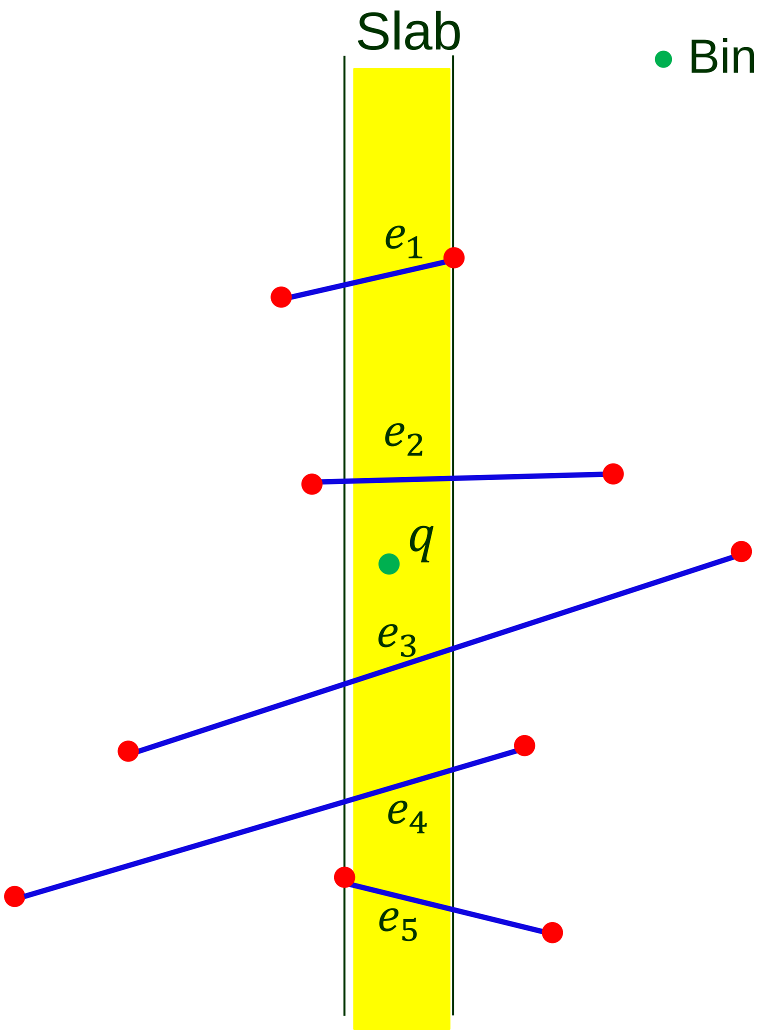
$O(\log n)$

2. Binary search within the slab.

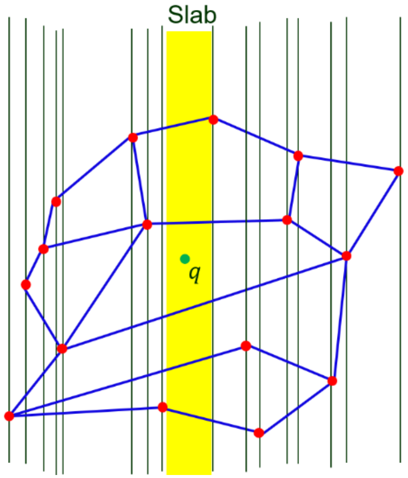
- Given a segment s crossing the slab, determine whether q is above, below, or on s.

$\leq n$ segments $\Rightarrow$ $O(\log n)$ 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.

$\leq n$ segments $\Rightarrow$ $O(\log n)$ 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. $O(n)$

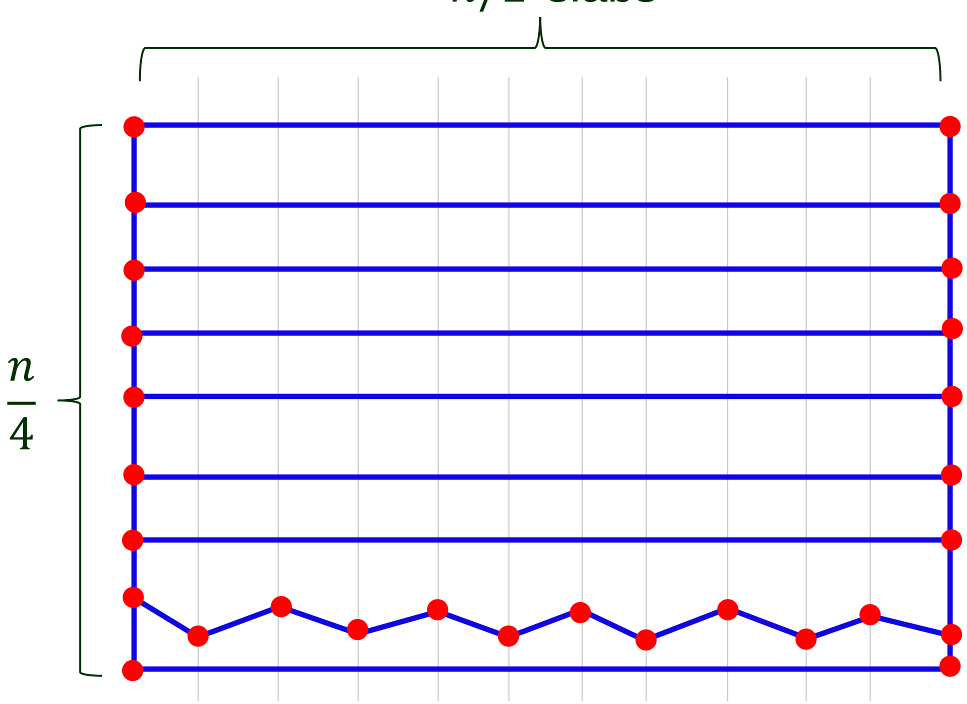
$O(n)$ slabs

$$\left.\begin{array}{r} \text{An array for every slab. } O(n) \\ O(n) \text{ slabs} \end{array}\right\} \Rightarrow O(n^2)$$

Storage

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

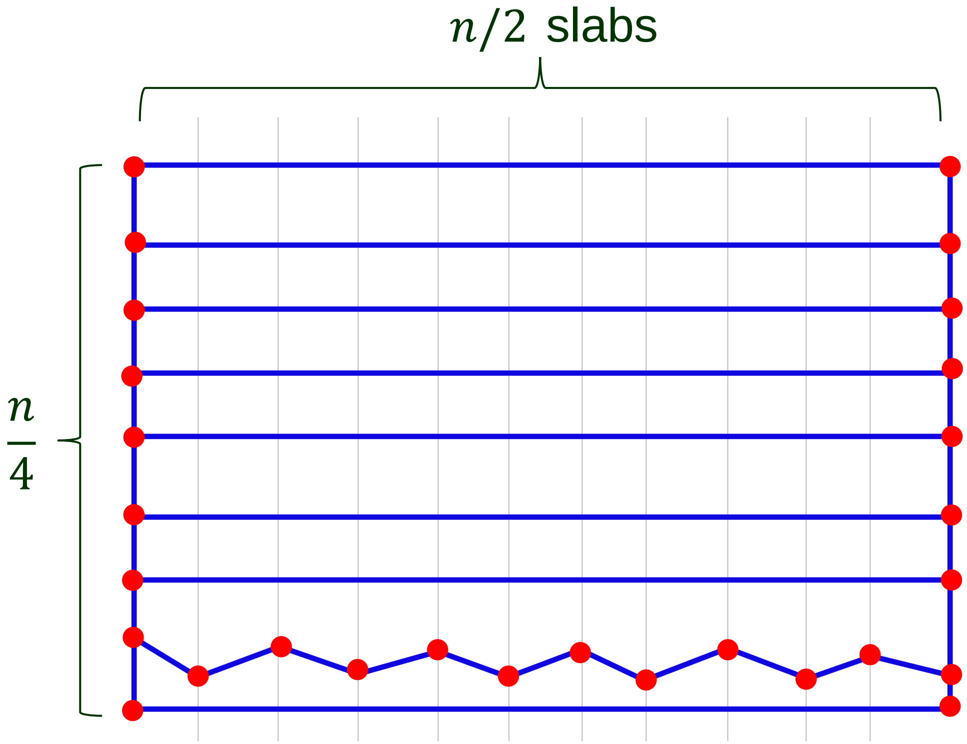
$$\left.\begin{array}{r} \text{An array for every slab. } O(n) \\ O(n) \text{ slabs} \end{array}\right\} \Rightarrow O(n^2)$$



Storage

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

$$\left.\begin{array}{r} O(n) \\ O(n) \text{ slabs} \end{array}\right\} \Rightarrow O(n^2)$$

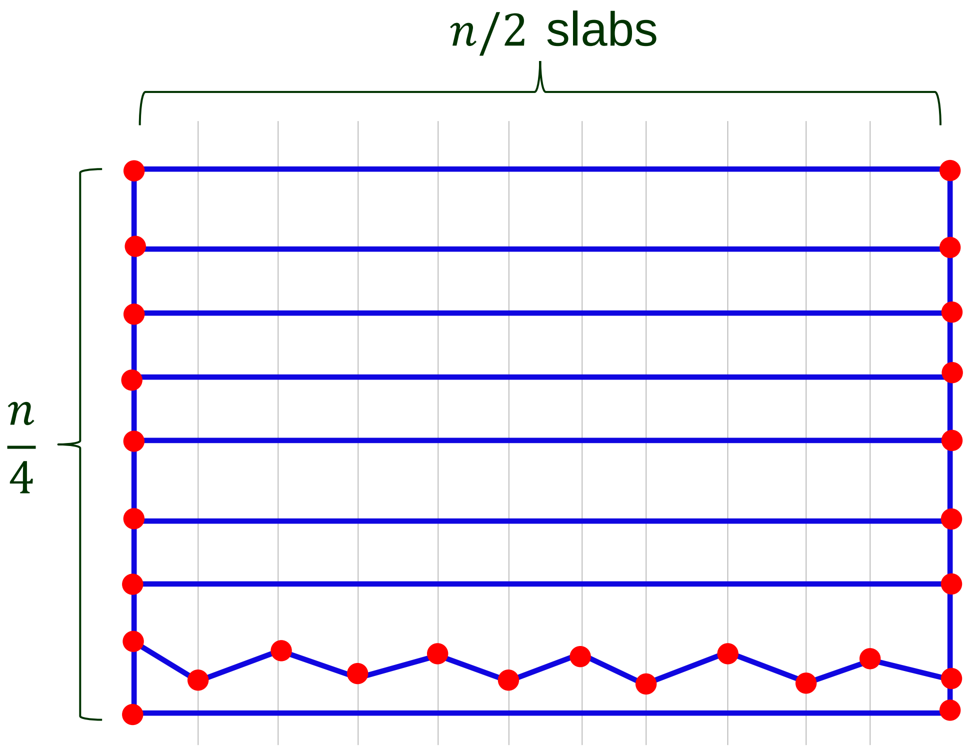


$\Theta(n^2)$ storage!

Storage

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

$$\left.\begin{array}{r}\text{An array for every slab. } O(n)\\ O(n) \text{ slabs}\end{array}\right\} \Rightarrow O(n^2)$$

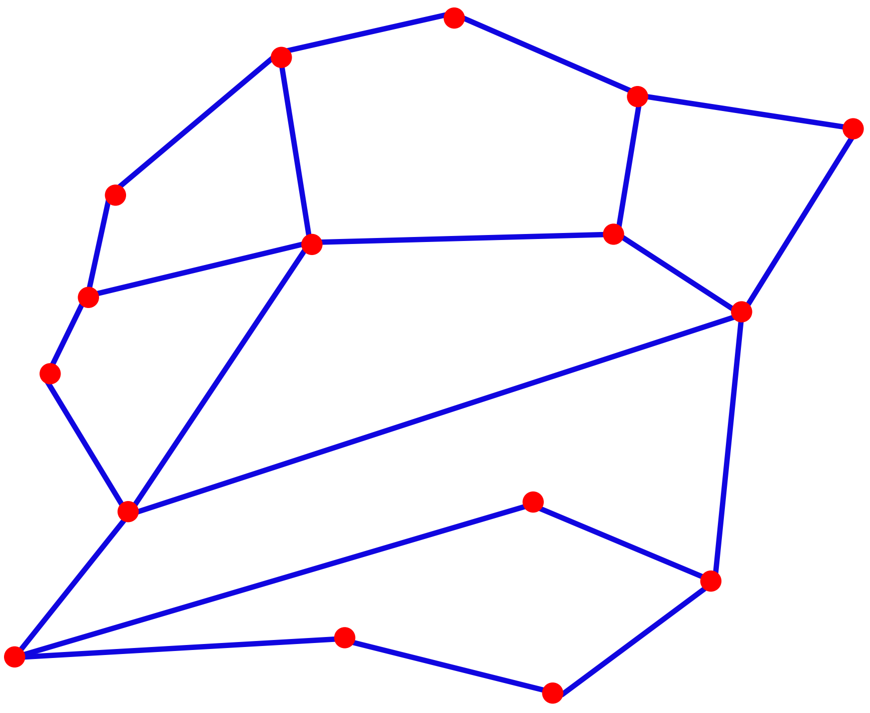


$\Theta(n^2)$ storage!

Over-refinement of S for query purpose!

II. Trapezoidal Map

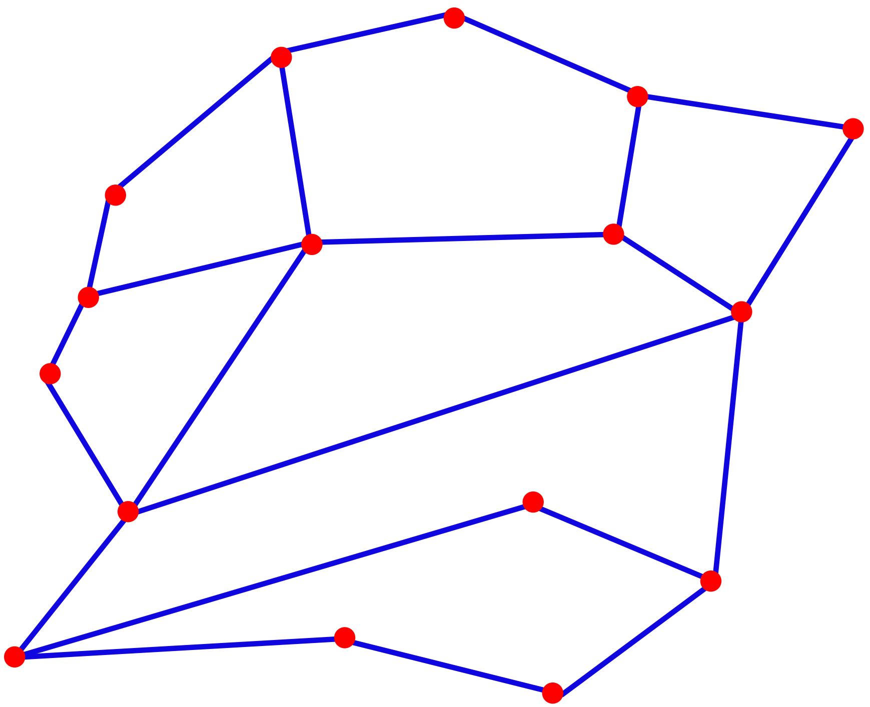
- Makes point location query easier.
- Needs $O(n)$ storage.



II. Trapezoidal Map

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

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

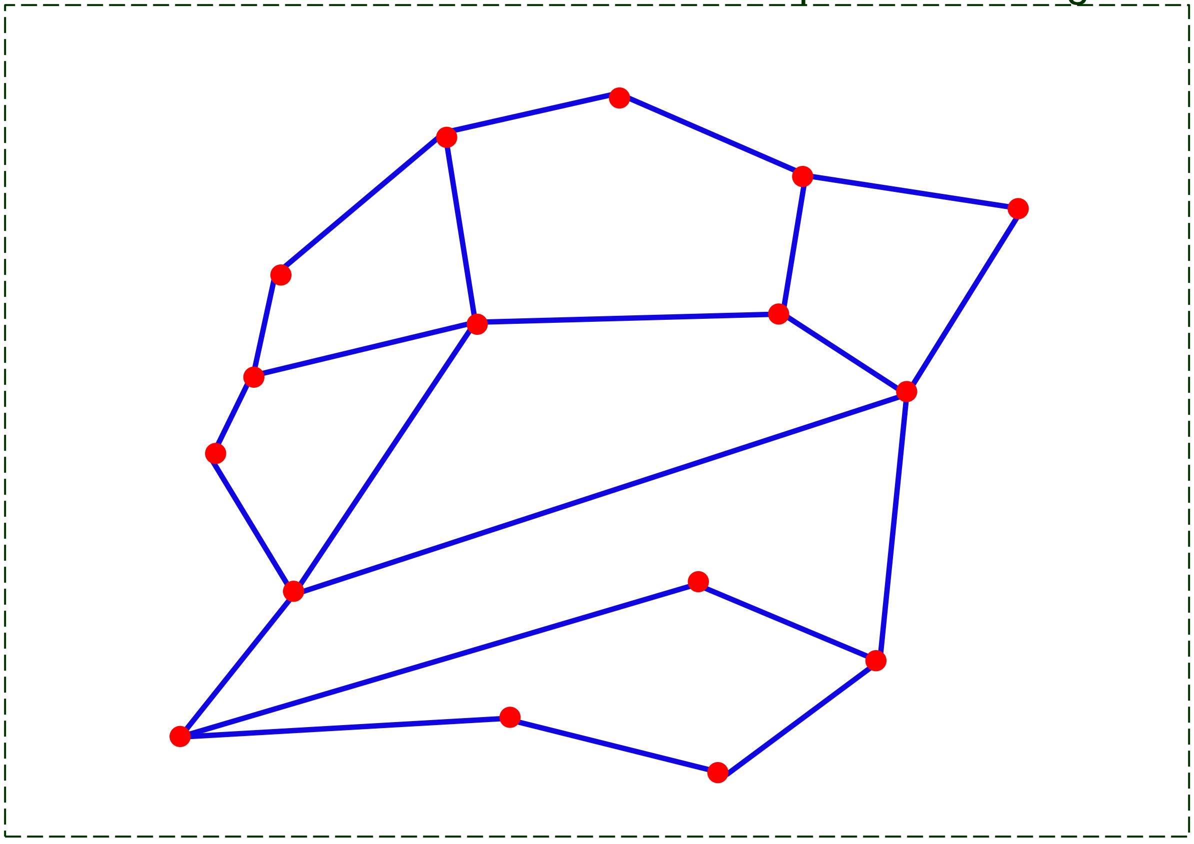


II. Trapezoidal Map

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

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

axis-parallel rectangle R



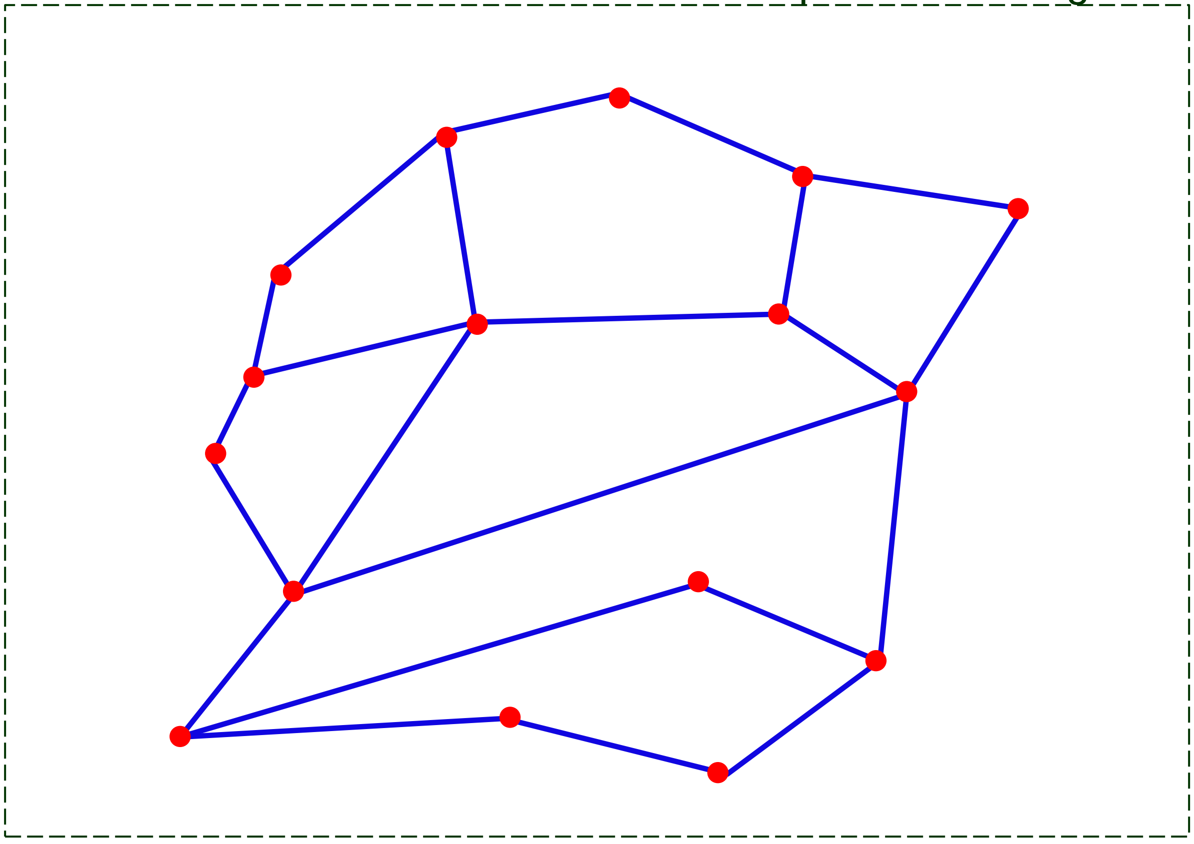
- Draw a rectangle bounding all segments in its interior.

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



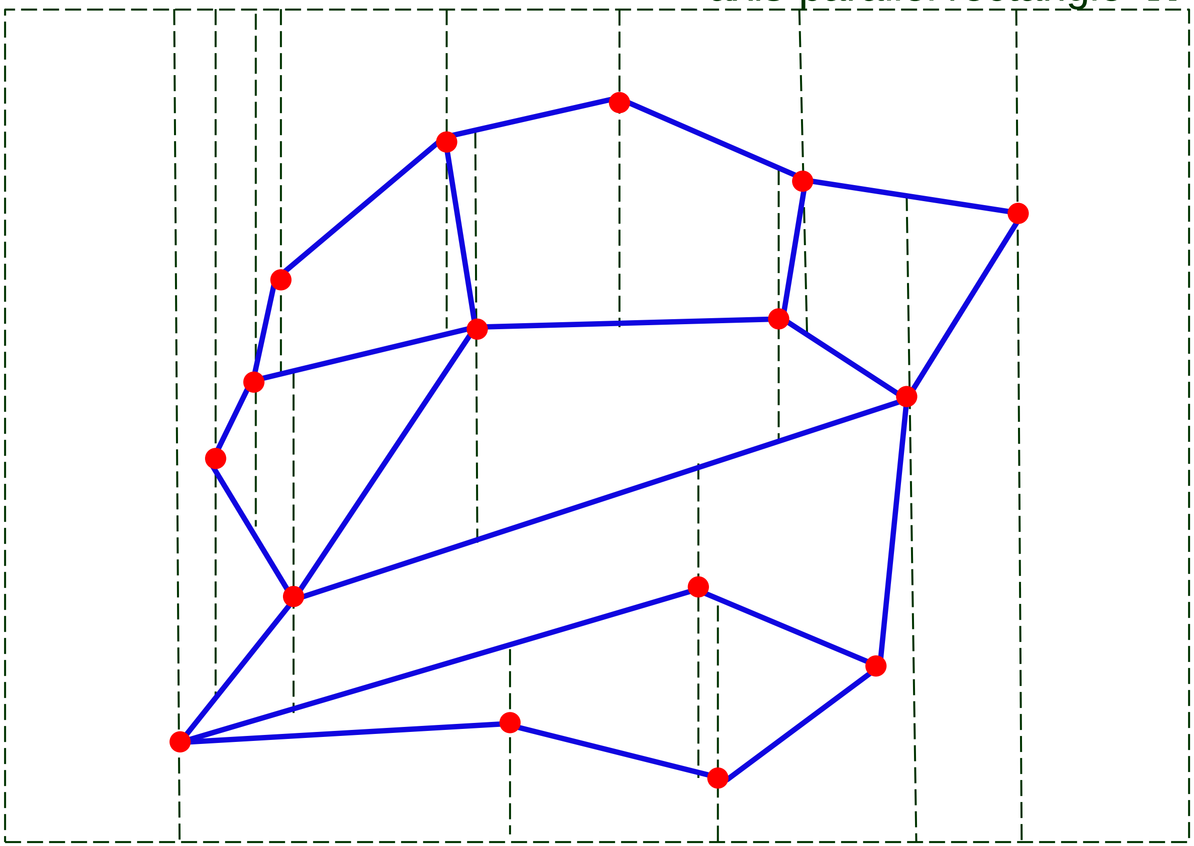
- Draw a rectangle bounding all segments in its interior.
- From every point draw two vertical extensions (up and down).
- Stop when they meet another segment or the boundary of R.

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



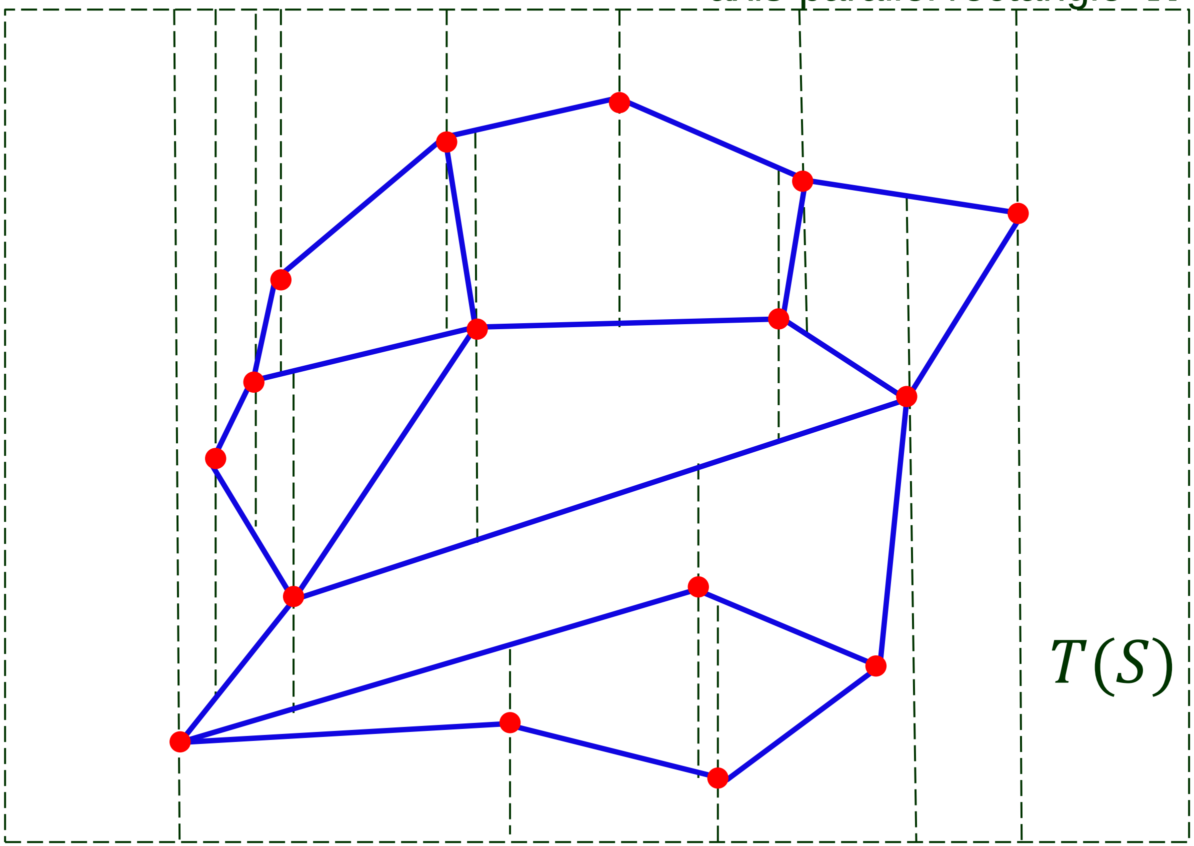
- Draw a rectangle bounding all segments in its interior.
- From every point draw two vertical extensions (up and down).
- Stop when they meet another segment or the boundary of R.

II. Trapezoidal Map

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

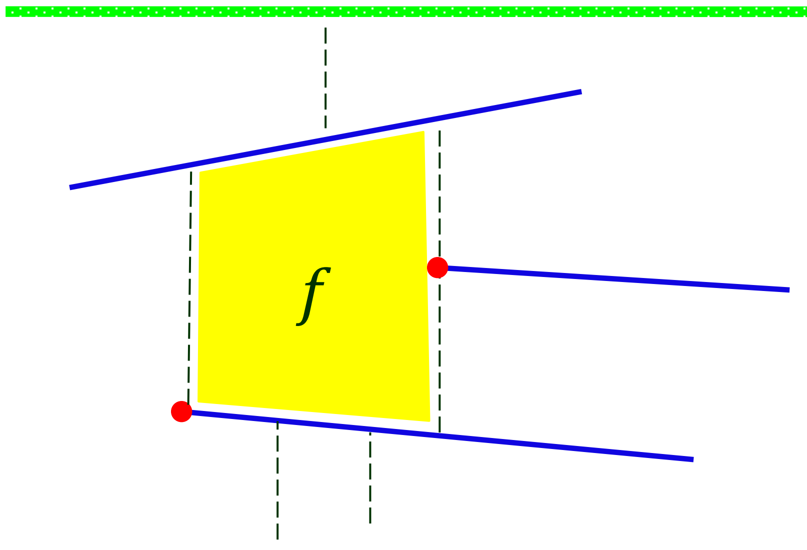
General position assumption (removable)
No two vertices have the same x-coordinate.

axis-parallel rectangle R



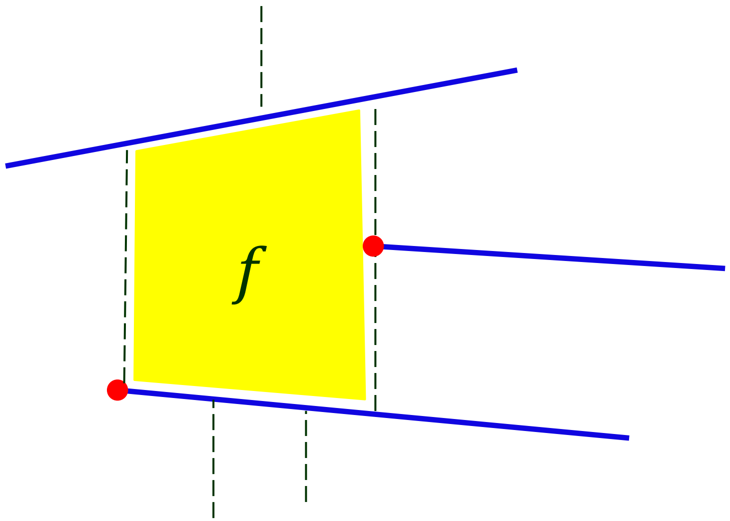
- Draw a rectangle bounding all segments in its interior.
- From every point draw two vertical extensions (up and down).
- Stop when they meet another segment or the boundary of R.

Face of a Trapezoidal Map

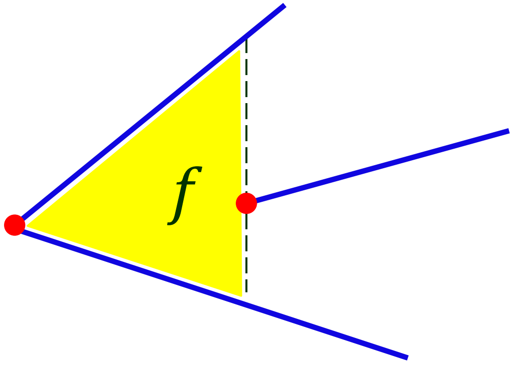


Trapezoid

Face of a Trapezoidal Map

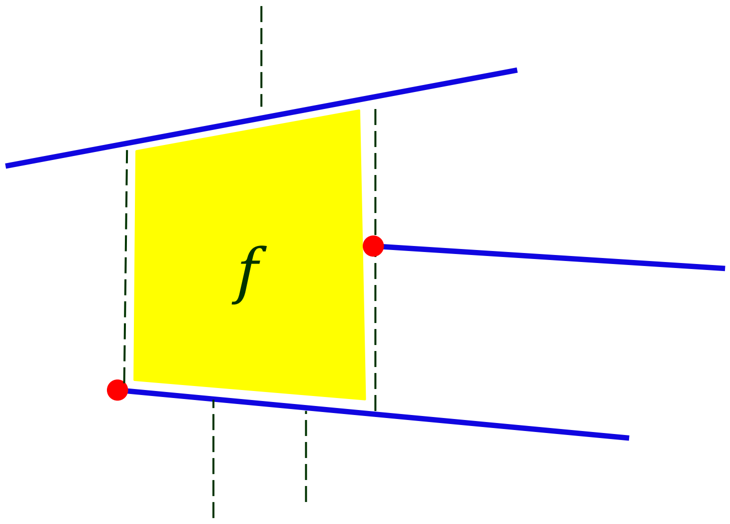


Trapezoid

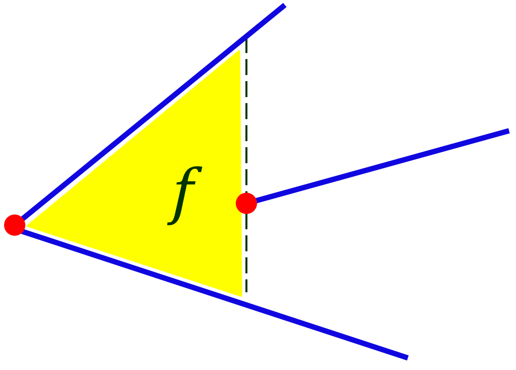


Triangle

Face of a Trapezoidal Map



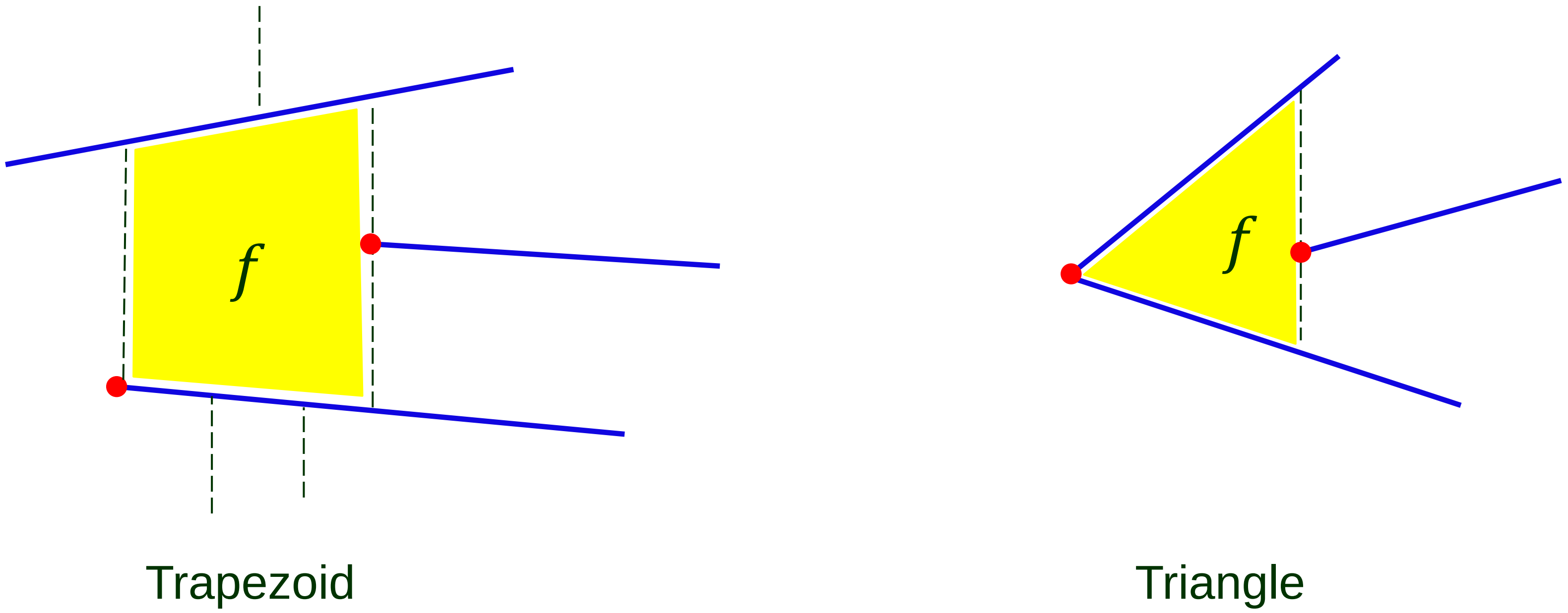
Trapezoid



Triangle

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

Face of a Trapezoidal Map

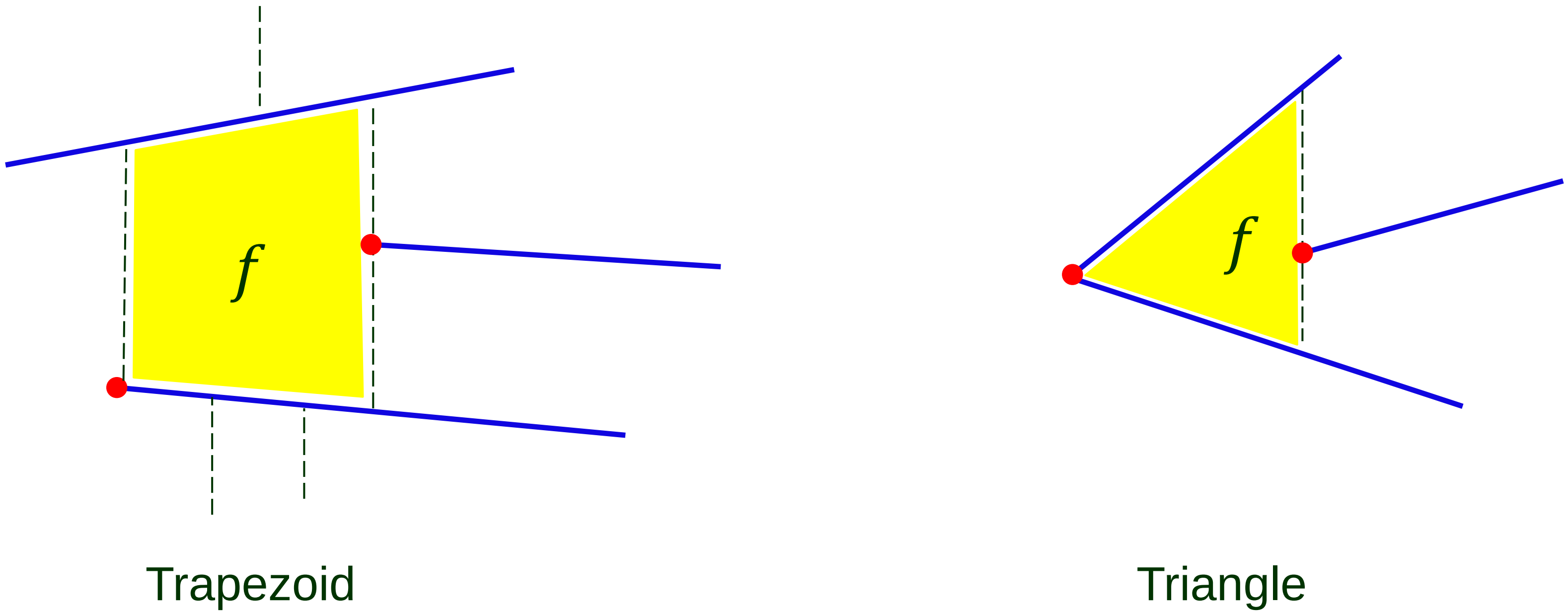


- Every face in $T(s)$ has ≤ 2 vertical sides and 2 non-vertical sides.
- A vertical side is one of two cases:

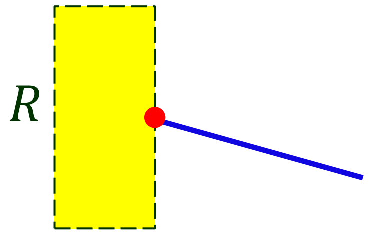
  - a vertical extension, or

  - a vertical edge of R.

Face of a Trapezoidal Map

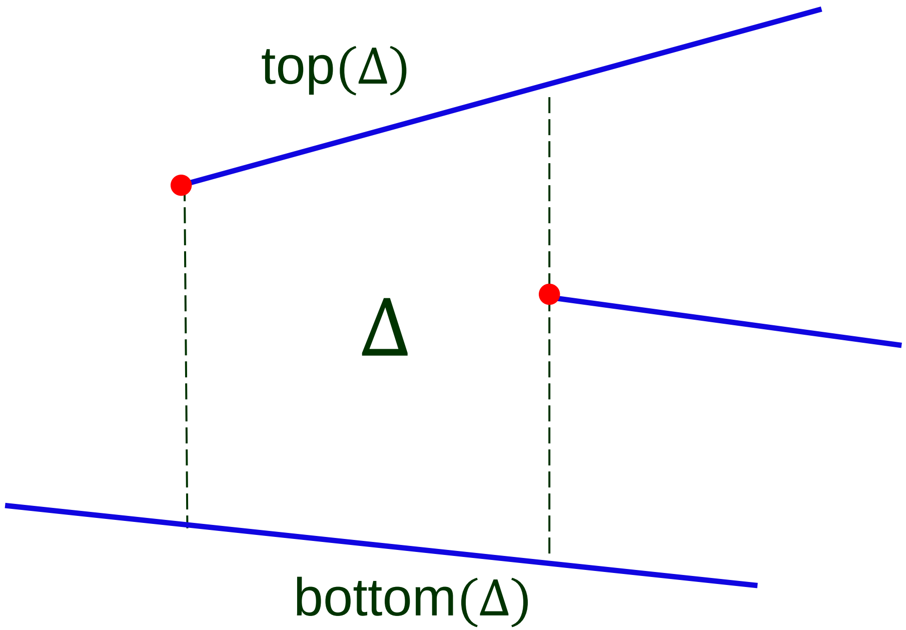


- Every face in $T(s)$ has ≤ 2 vertical sides and 2 non-vertical sides.
- A vertical side is one of two cases:
  - a vertical extension, or
  - a vertical edge of R.

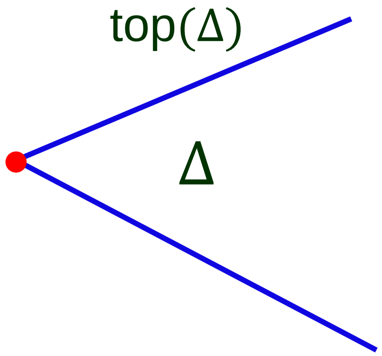


Top & Bottom

Distinguish the two non-vertical sides.

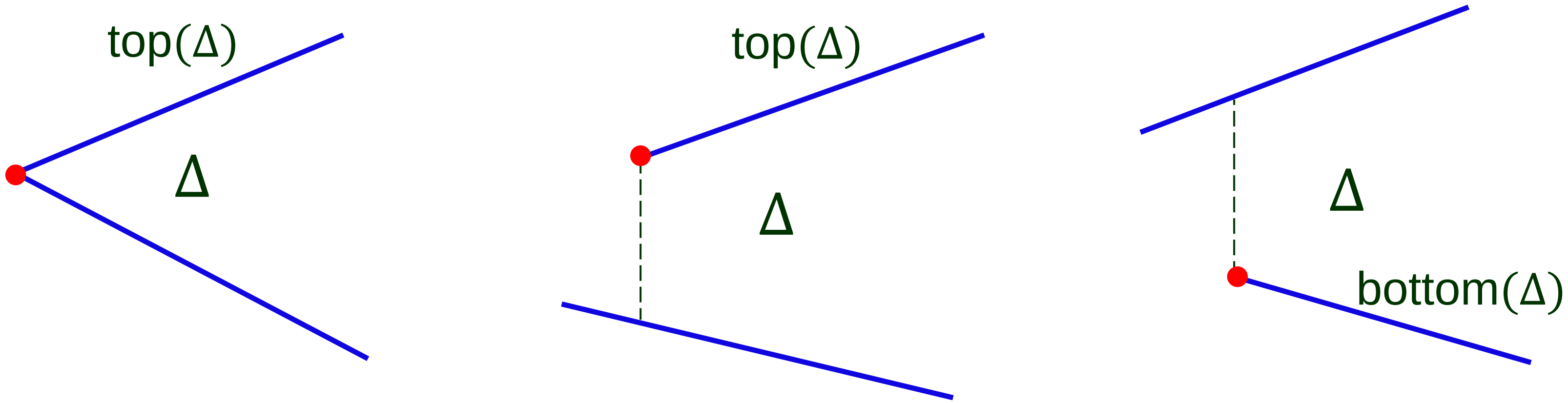


Classification of the Left Side



(a) Degenerating into a point

Classification of the Left Side

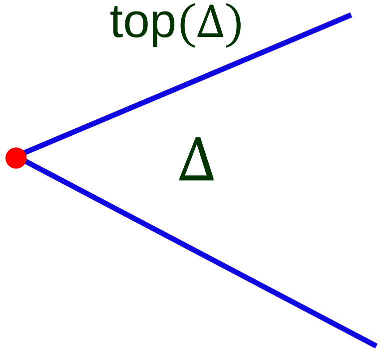


(a) Degenerating into a point

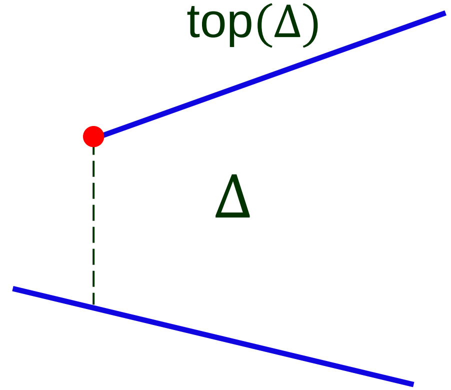
(b) Lower vertical extension

(c) Upper vertical extension

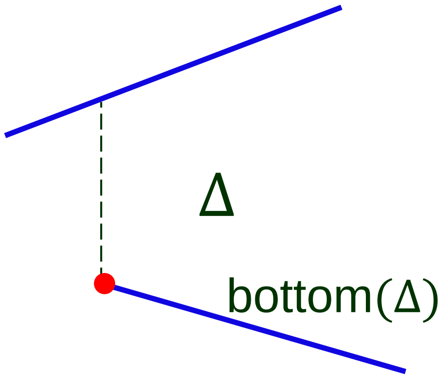
Classification of the Left Side



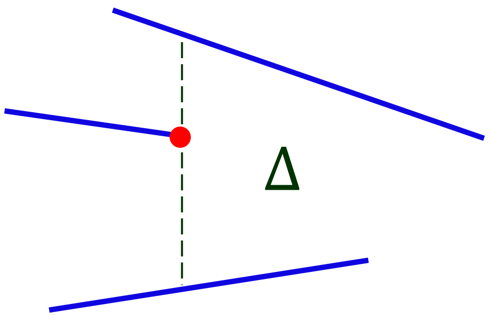
(a) Degenerating into a point



(b) Lower vertical extension

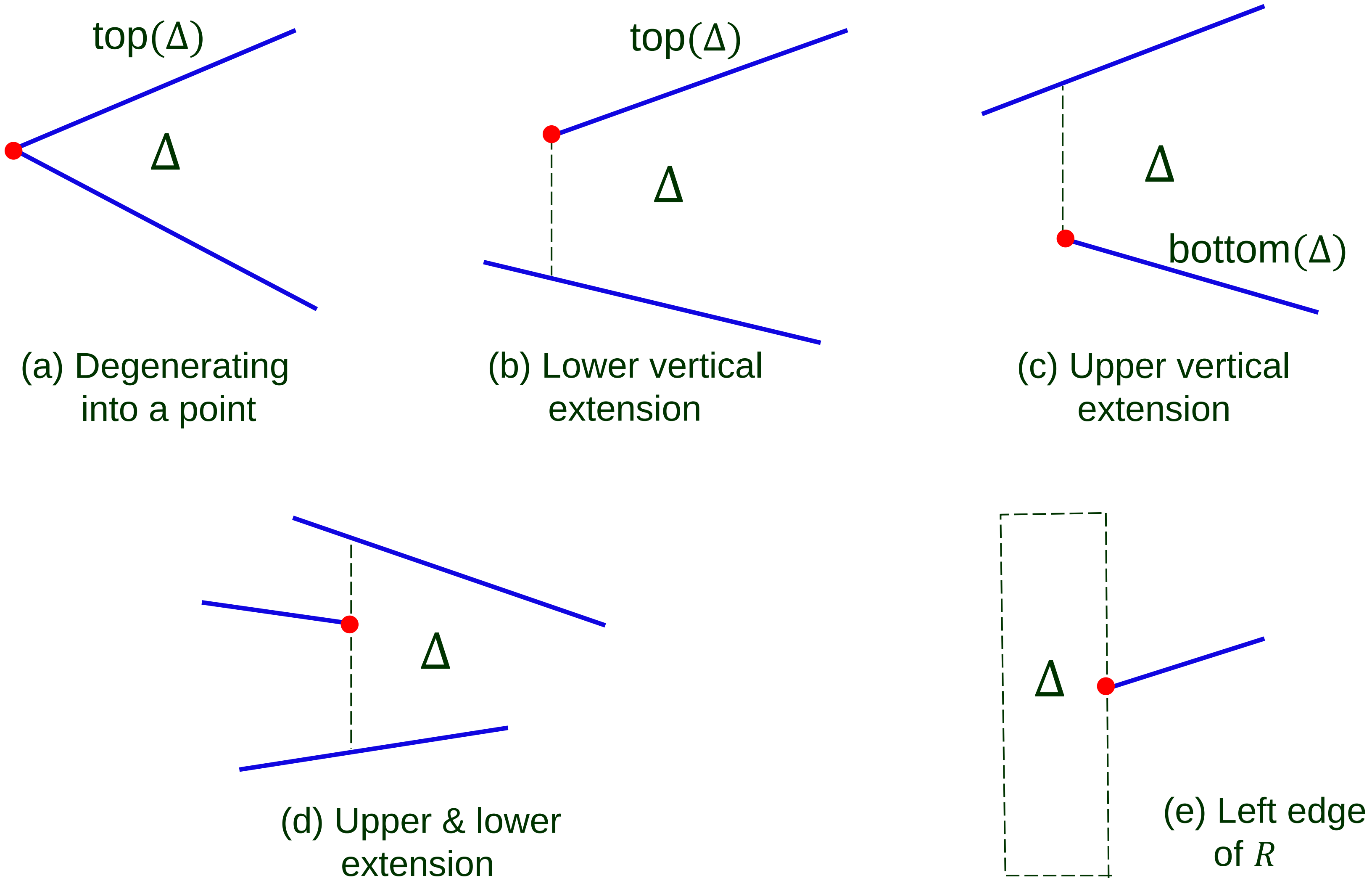


(c) Upper vertical extension

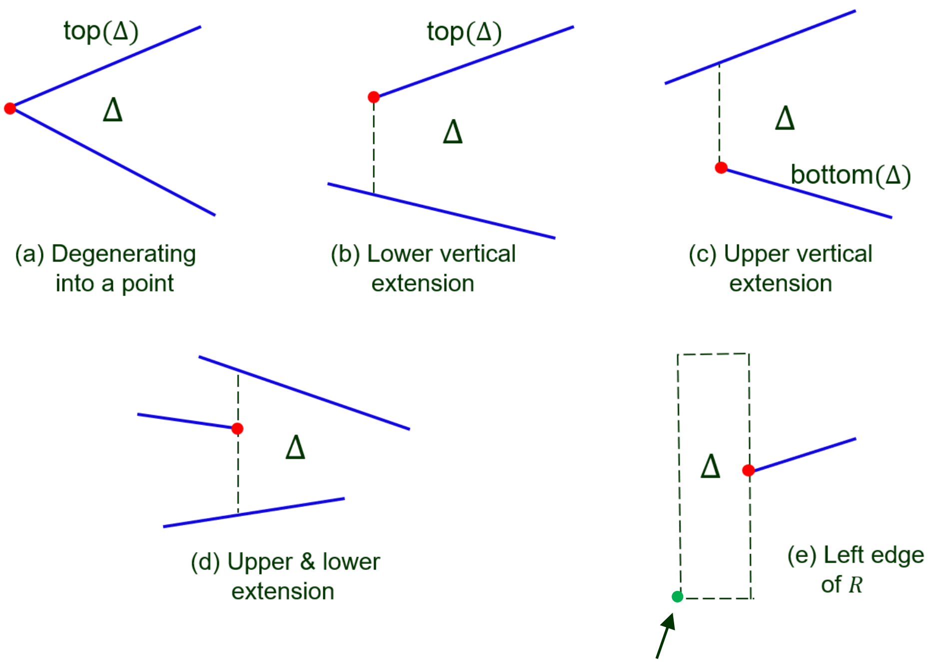


(d) Upper & lower extension

Classification of the Left Side



Defining Endpoint

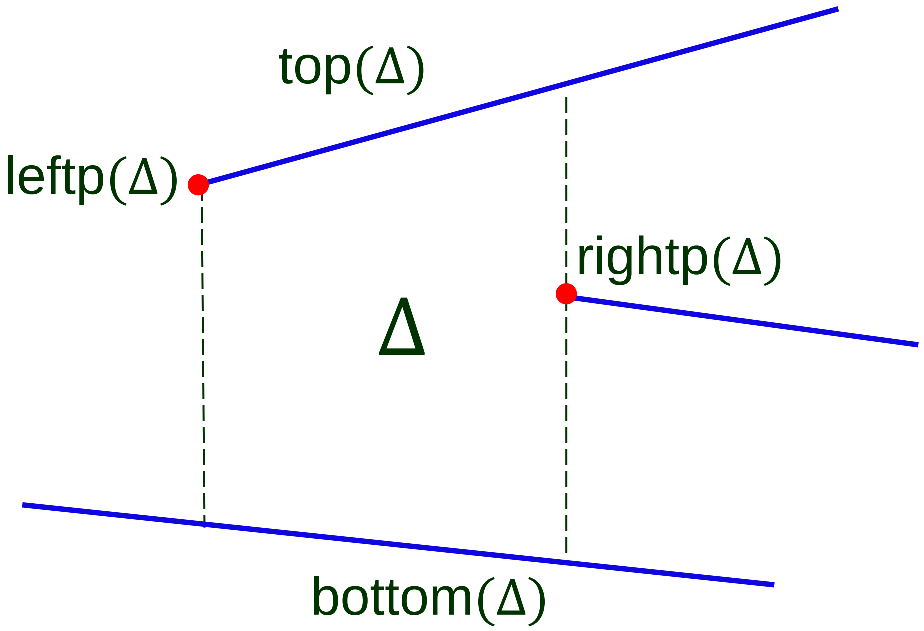


leftp(Δ) $\stackrel{\text{def}}{=}$ left endpoint of top(Δ) (b)
or left endpoint of bottom(Δ) (c)
or both (a)
or right endpoint of a 3rd segment (d)
or lower left corner of R (e)

rightp(Δ) can be similarly defined.

Face Description

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:

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

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

4

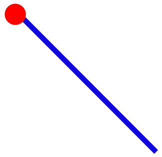
III. Complexity of the Trapezoidal Map

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

Proof   A vertex is one of three types below:

- a vertex of R

4



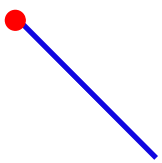
- an endpoint of a segment

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 4



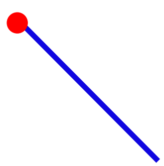
- an endpoint of a segment $\leq 2n$

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 4



- an endpoint of a segment $\leq 2n$

(shared endpoints may exist)

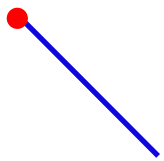
III. Complexity of the Trapezoidal Map

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

Proof   A vertex is one of three types below:

- a vertex of R

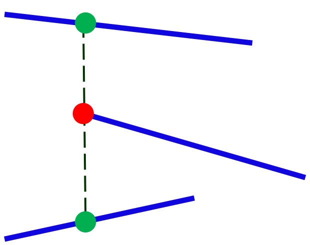
4



- 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.

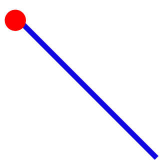
III. Complexity of the Trapezoidal Map

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

Proof A vertex is one of three types below:

- a vertex of R

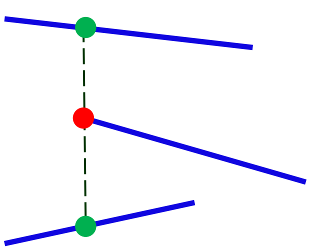
4



- 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.

Every endpoint generates at most two such points.

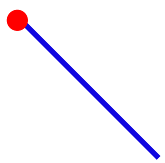
III. Complexity of the Trapezoidal Map

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

Proof A vertex is one of three types below:

- a vertex of R

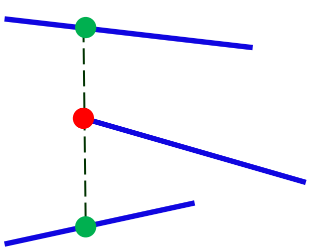
4



- 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.

Every endpoint generates at most two such points.

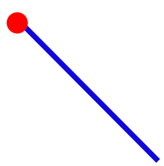
$\leq 2 \times 2n$

III. Complexity of the Trapezoidal Map

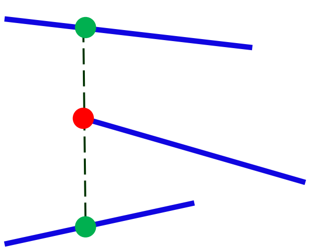
Lemma 1 $T(S)$ contains $\leq 6n + 4$ vertices.

Proof A vertex is one of three types below:

- a vertex of R — 4



- 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.

#vertices $\leq 4 + 2n + 4n$

$= 6n + 4$ $\square$

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 Δ is represented by leftp(Δ). Need only count leftp(Δ), including multiplicities (#times for each endpoint).

Number of Trapezoids

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

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

leftp(Δ) is one of three possible types:

Number of Trapezoids

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

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

leftp(Δ) is one of three possible types:

- lower left corner of R

Number of Trapezoids

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

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

leftp(Δ) is one of three possible types:

- lower left corner of R

$\Longrightarrow$ it is leftp(Δ) for exactly 1 trapezoid.

Number of Trapezoids

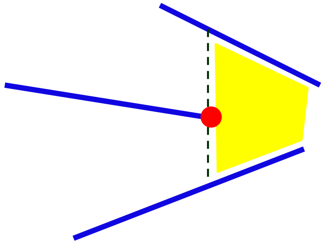
Lemma 2 $T(S)$ contains $\leq 3n+1$ trapezoids.

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

leftp(Δ) is one of three possible types:

- lower left corner of R

$\Longrightarrow$ it is leftp(Δ) for exactly 1 trapezoid.



- right endpoint of a segment (n such points)

Number of Trapezoids

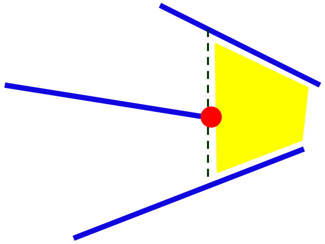
Lemma 2 $T(S)$ contains $\leq 3n + 1$ trapezoids.

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

leftp(Δ) is one of three possible types:

- lower left corner of R

$\Longrightarrow$ it is leftp(Δ) for exactly 1 trapezoid.



- right endpoint of a segment (n such points)

$\Longrightarrow$ it is leftp(Δ) for ≤ 1 trapezoid.

Number of Trapezoids

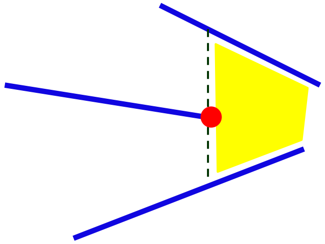
Lemma 2 $T(S)$ contains $\leq 3n + 1$ trapezoids.

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

leftp(Δ) is one of three possible types:

- lower left corner of R

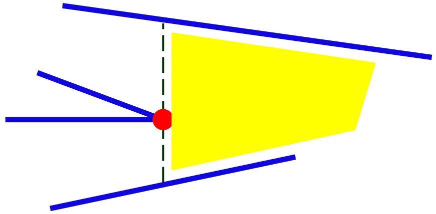
$\Longrightarrow$ it is leftp(Δ) for exactly 1 trapezoid.



- right endpoint of a segment (n such points)

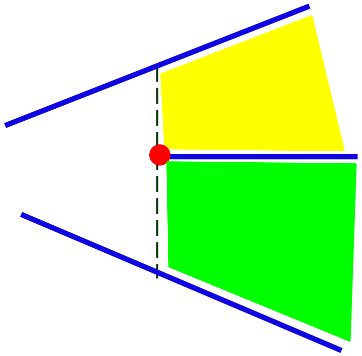
$\Longrightarrow$ it is leftp(Δ) for ≤ 1 trapezoid.

shared endpoint



(cont'd)

- left endpoint of a segment (n such points)

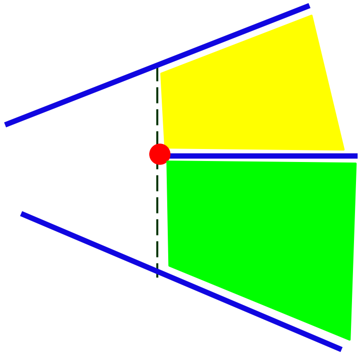


$\Longrightarrow$ leftp(Δ) for ≤ 2 trapezoids.

shared endpoint

(cont'd)

- left endpoint of a segment (n such points)



$\Longrightarrow$ leftp(Δ) for ≤ 2 trapezoids.

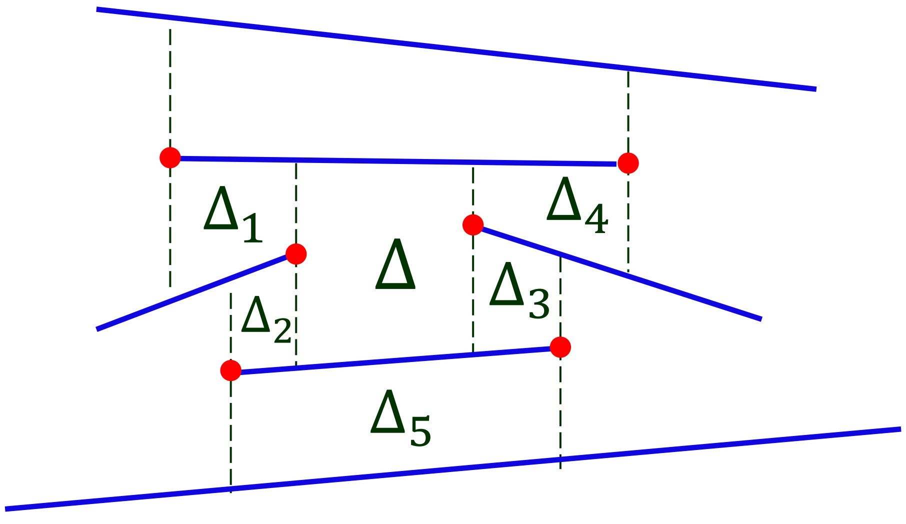
shared endpoint

$$\# \text{ trapezoids} \leq 1 + 1 \times n + 2 \times n$$
$$= 3n + 1$$

$\square$

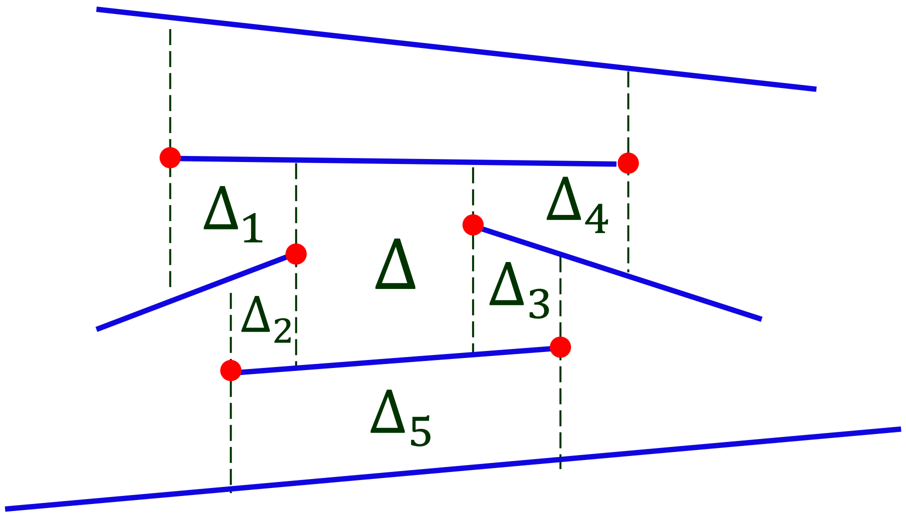
Adjacency

Trapezoids Δ and Δ' are *adjacent* if they share a *vertical* edge.



Adjacency

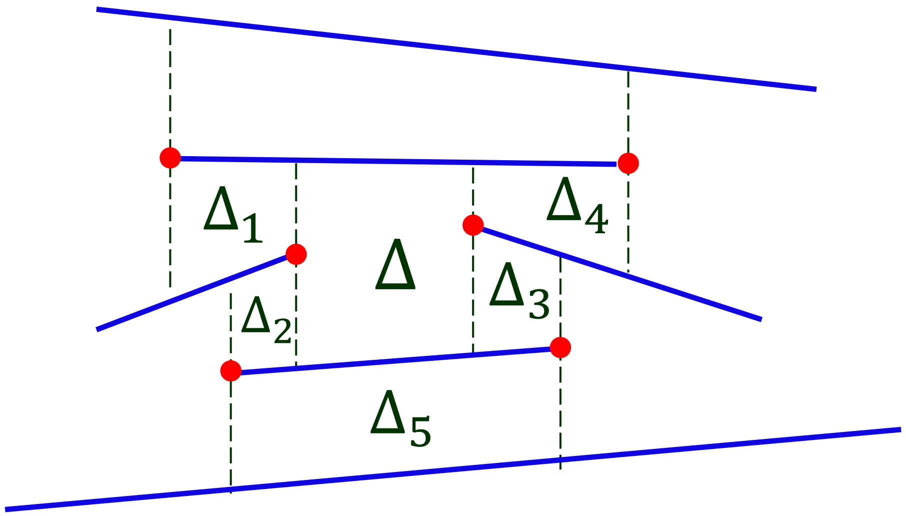
Trapezoids Δ and Δ' are *adjacent* if they share a *vertical* edge.



Δ is adjacent to Δ_1, Δ_2, Δ_3, and Δ_4 but not to Δ_5.

Adjacency

Trapezoids Δ and Δ' are *adjacent* if they share a *vertical* edge.



Δ is adjacent to Δ_1, Δ_2, Δ_3, and Δ_4 but not to Δ_5.

Δ_1: upper left neighbor

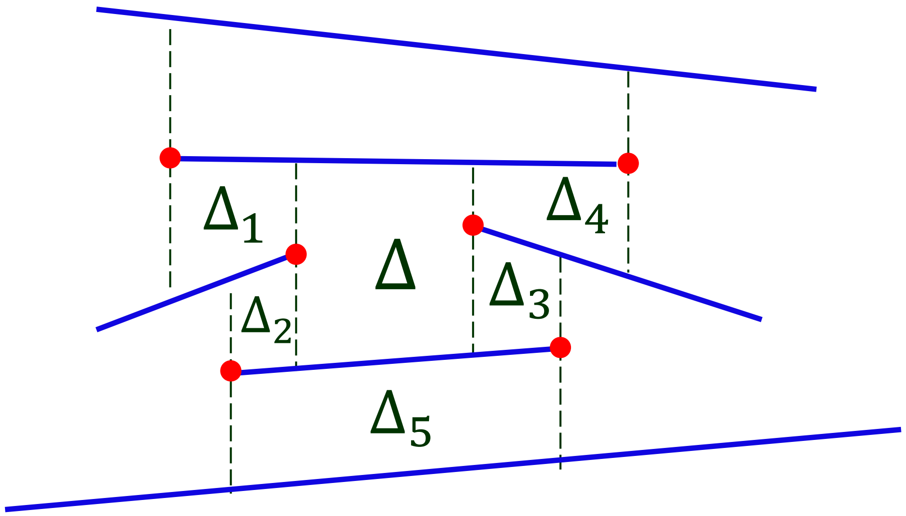
Δ_2: lower left neighbor

Δ_3: lower right neighbor

Δ_4: upper right neighbor

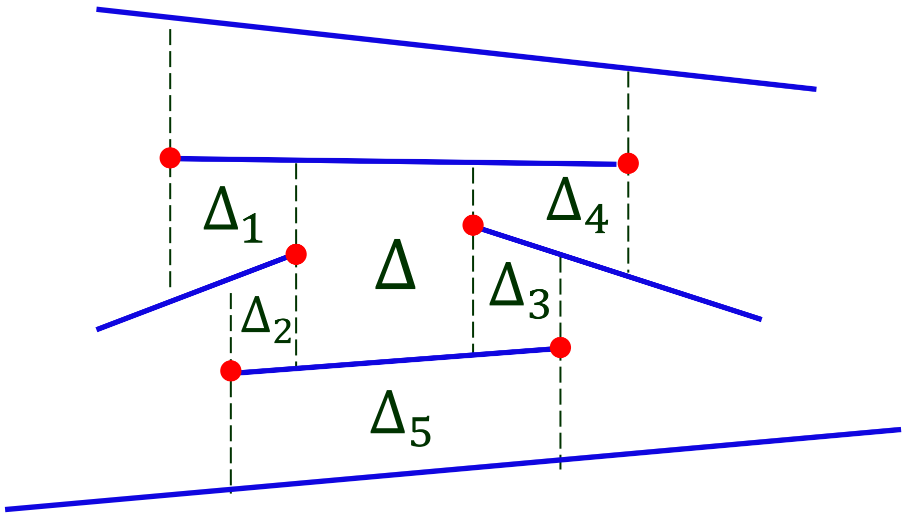
Adjacency (cont'd)

A trapezoid has ≤ 4 neighbors under the general position assumptions.



Adjacency (cont'd)

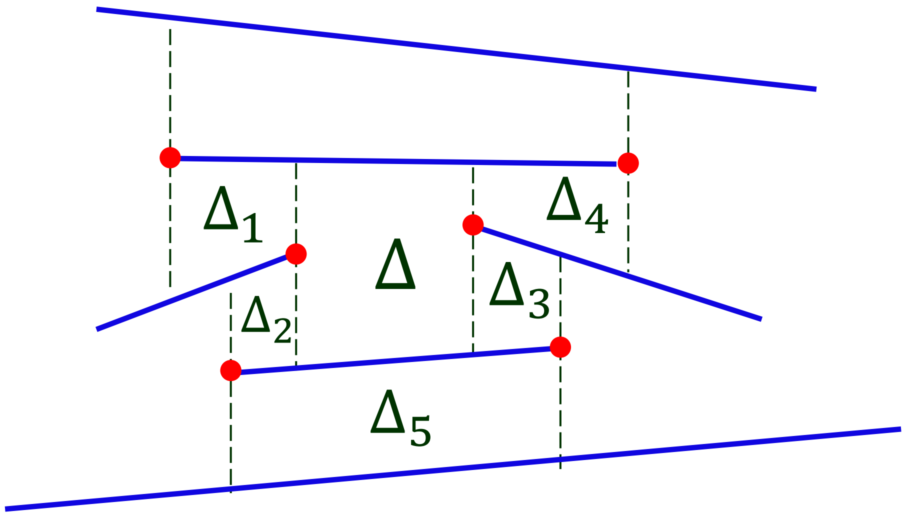
A trapezoid has ≤ 4 neighbors under the general position assumptions.



Δ' adjacent to Δ along the left vertical edge of Δ.

Adjacency (cont'd)

A trapezoid has ≤ 4 neighbors under the general position assumptions.



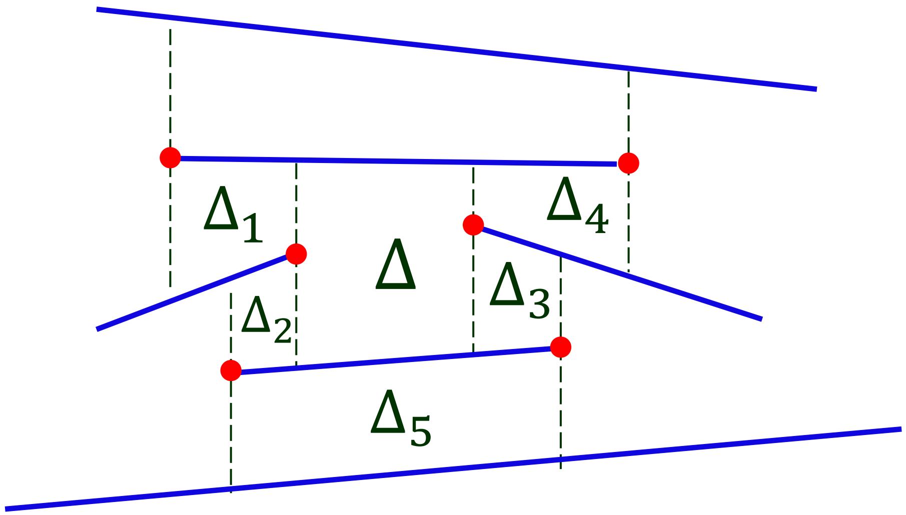
Δ' adjacent to Δ along the left vertical edge of Δ.

$\Downarrow$

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

Adjacency (cont'd)

A trapezoid has ≤ 4 neighbors under the general position assumptions.



Δ' adjacent to Δ along the left vertical edge of Δ.

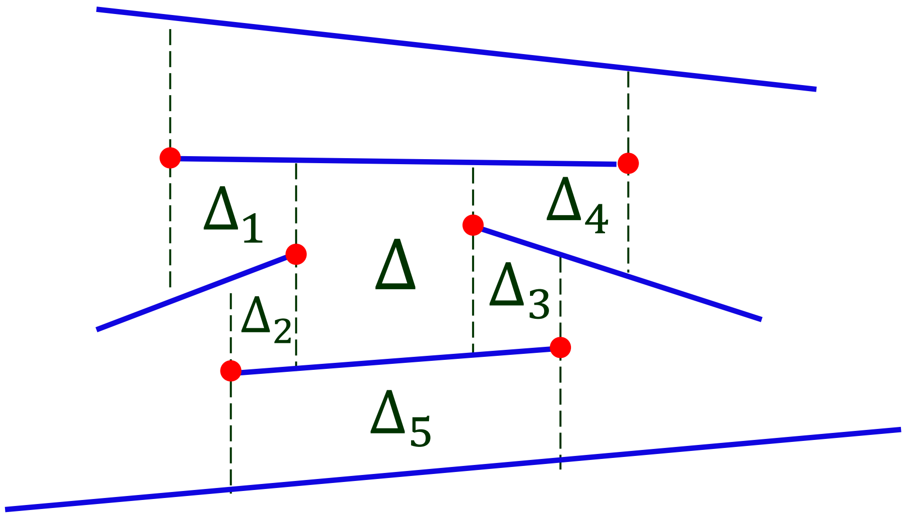
$\Downarrow$

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

$\text{top}(\Delta) = \text{top}(\Delta_1) = \text{top}(\Delta_4)$

Adjacency (cont'd)

A trapezoid has ≤ 4 neighbors under the general position assumptions.



Δ' adjacent to Δ along the left vertical edge of Δ.

$\Downarrow$

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

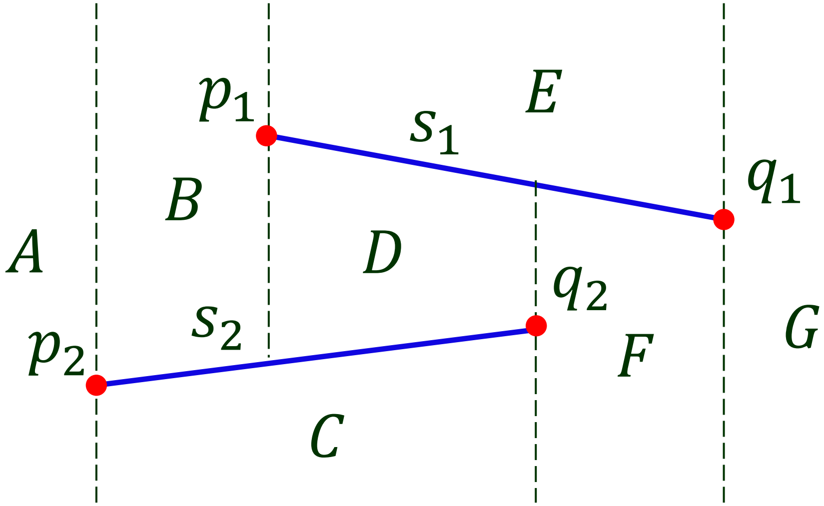
$\text{top}(\Delta) = \text{top}(\Delta_1) = \text{top}(\Delta_4)$

$\text{bottom}(\Delta) = \text{bottom}(\Delta_2) = \text{bottom}(\Delta_3)$

IV. Point Location Data Structure

D is a directed acyclic graph (DAG)

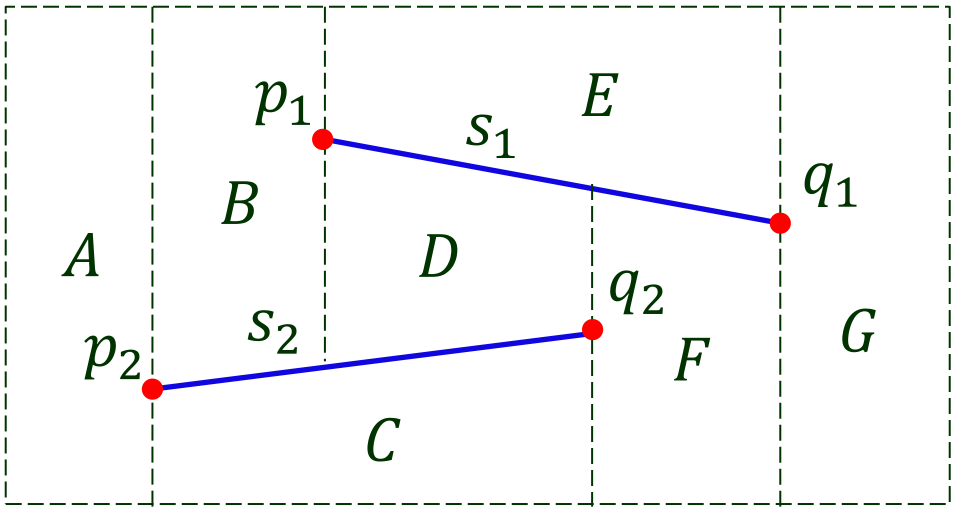
- one root
- one leaf for every trapezoid
- two types of internal nodes
  - x-nodes labeled with an endpoint of some segment
  - y-nodes labeled with a segment



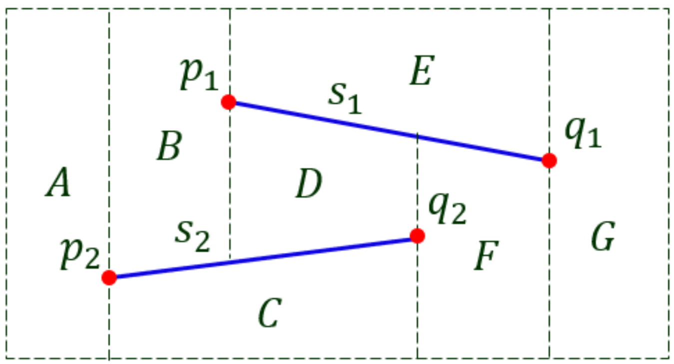
IV. Point Location Data Structure

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

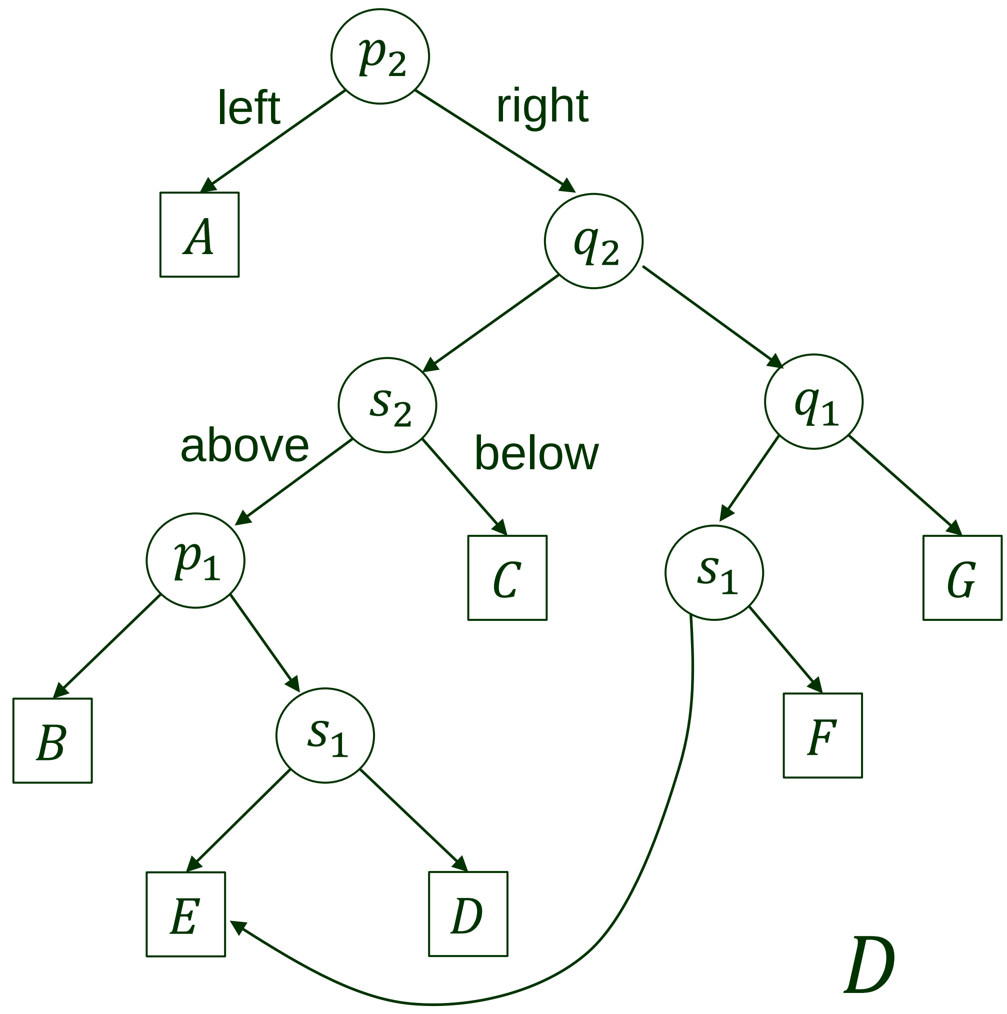


DAG



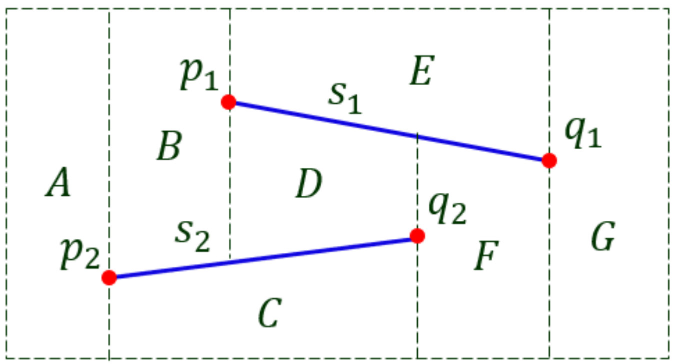
p_1
s_1
E
q_1
A
B
D
q_2
s_2
p_2
F
G
C

$T(S)$

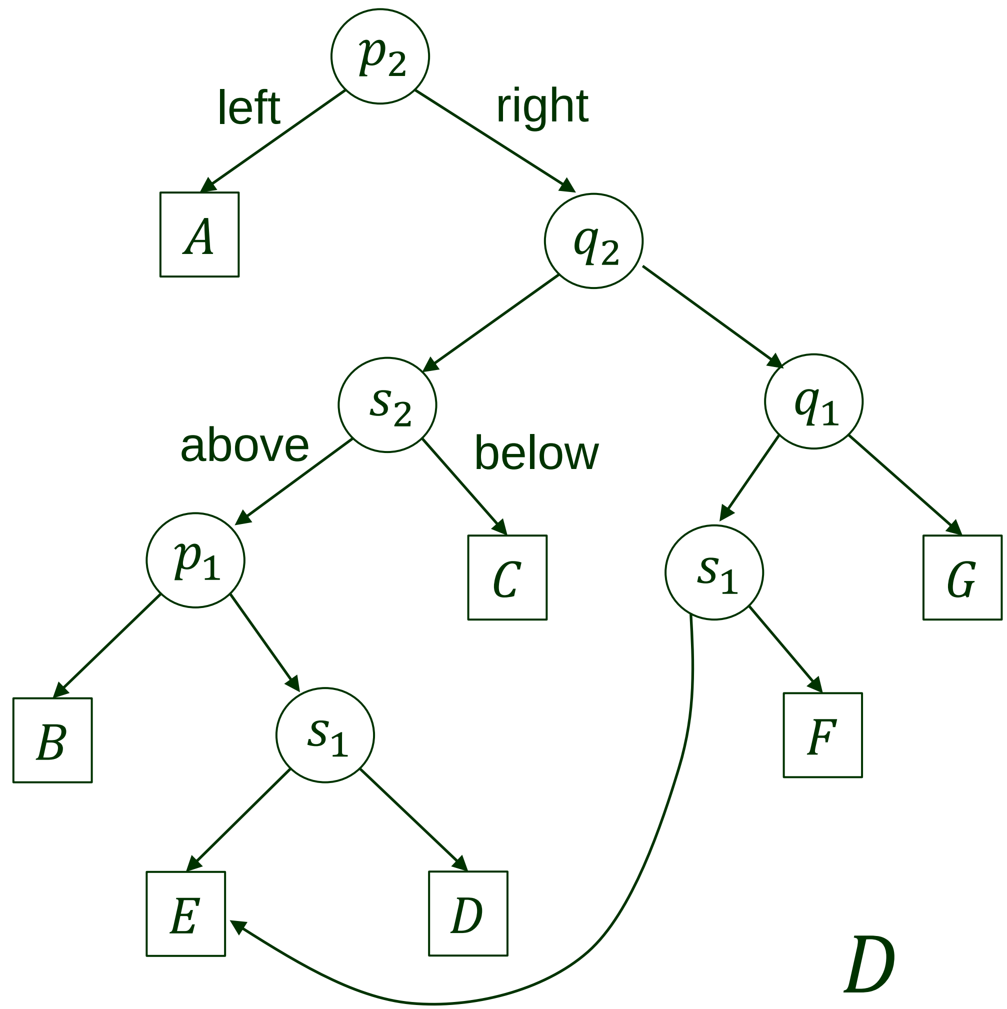


p_2
left
right
A
q_2
s_2
q_1
above
below
p_1
C
s_1
G
B
s_1
F
E
D
D

DAG



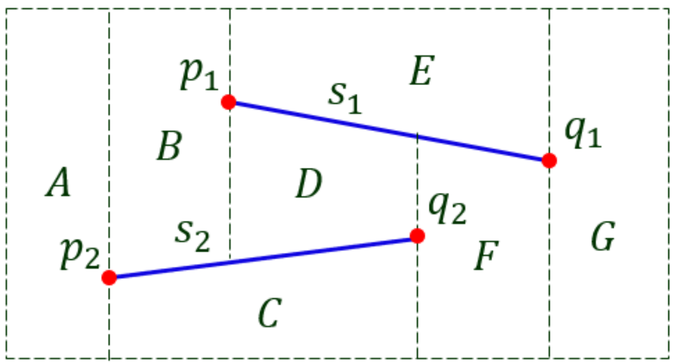
$T(S)$



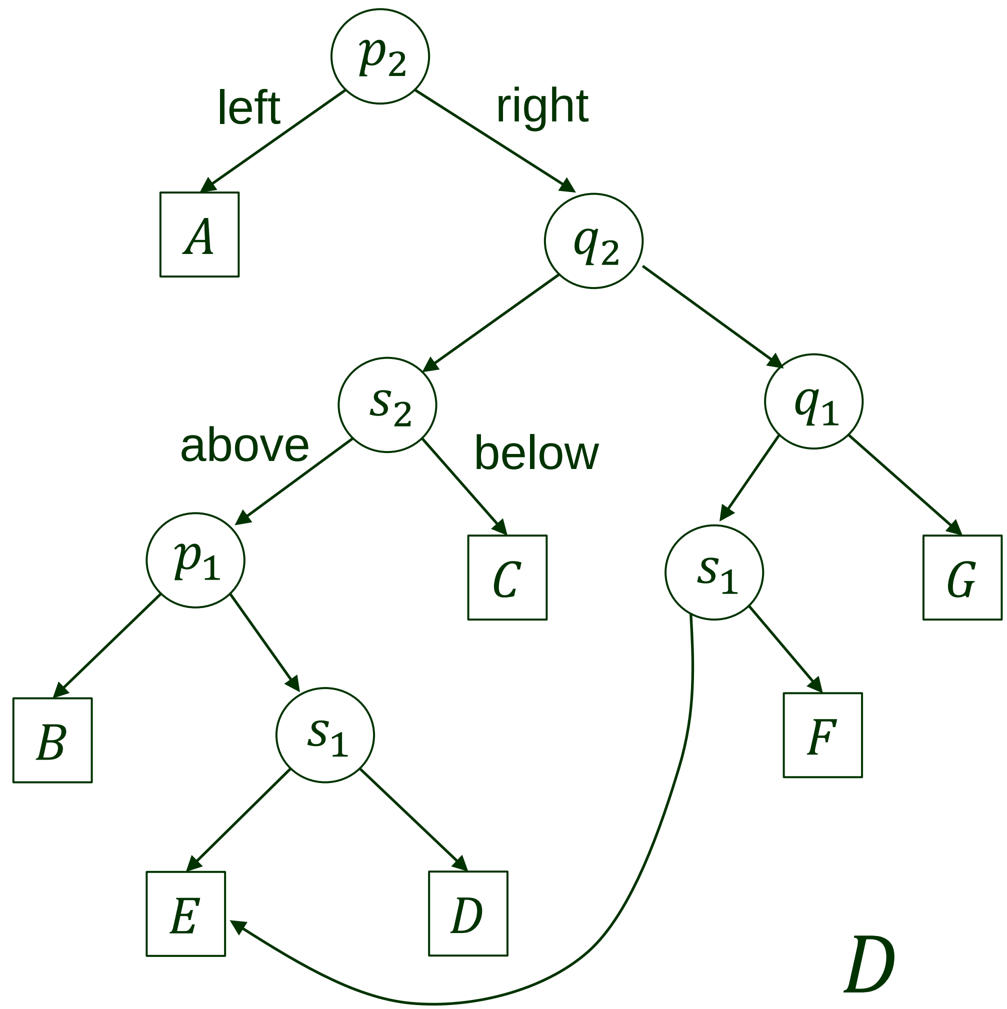
$T(S)$ and D cross-referenced

trapezoid Δ $\rightleftharpoons$ leaf of D

DAG



$T(S)$

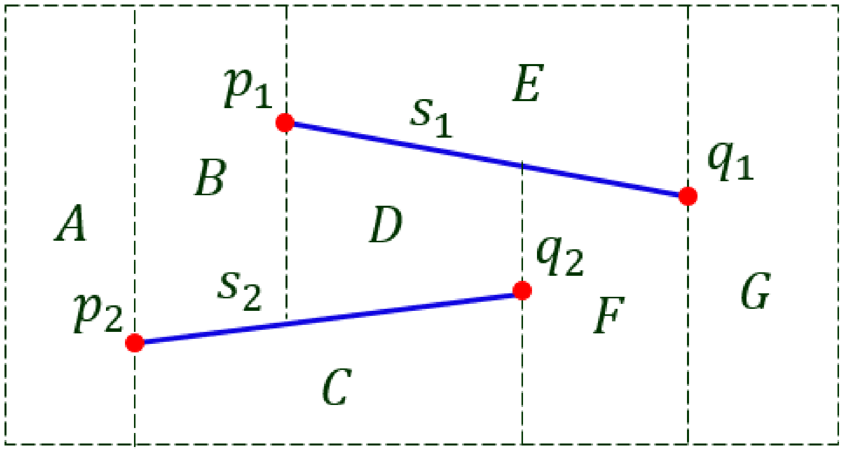


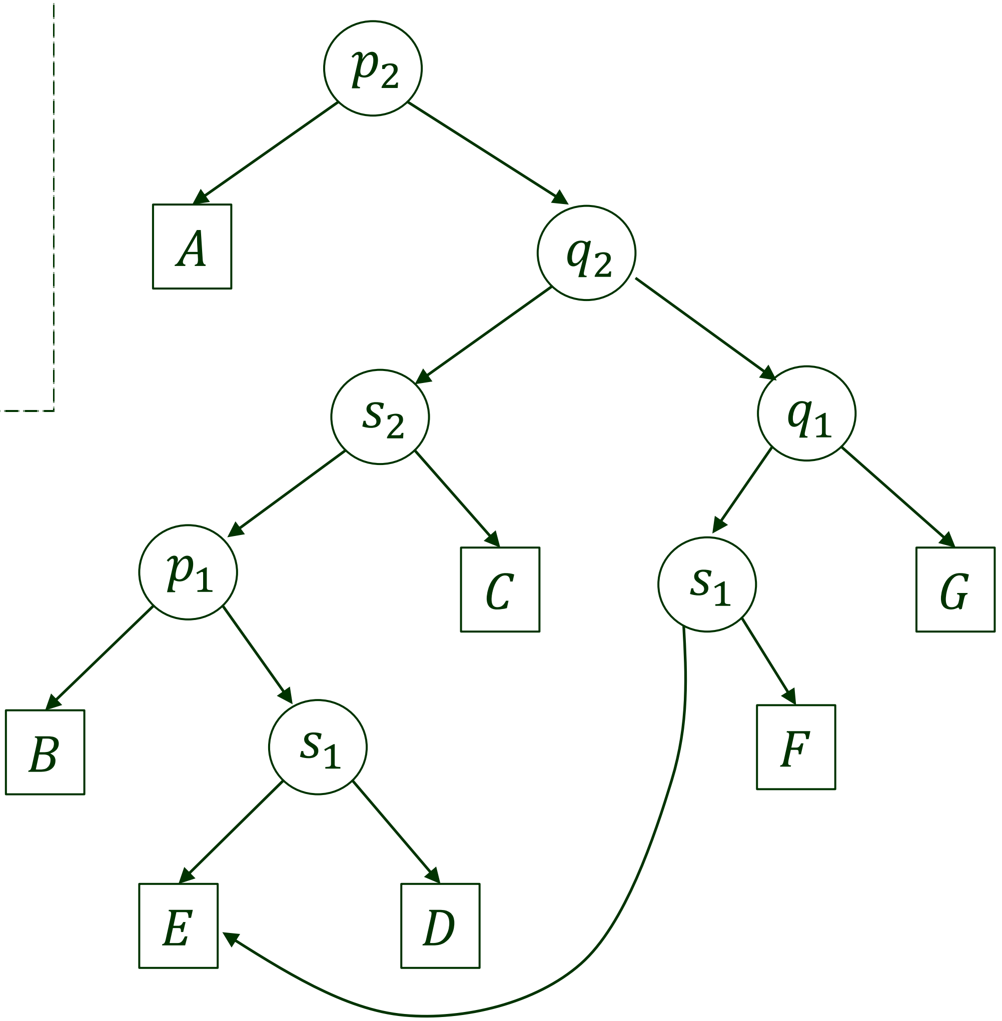
$T(S)$ and D cross-referenced

trapezoid Δ $\rightleftharpoons$ leaf of D

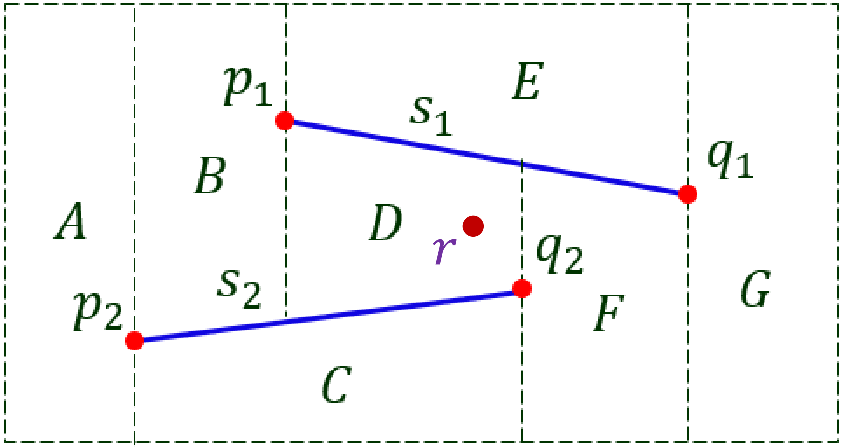
pointer

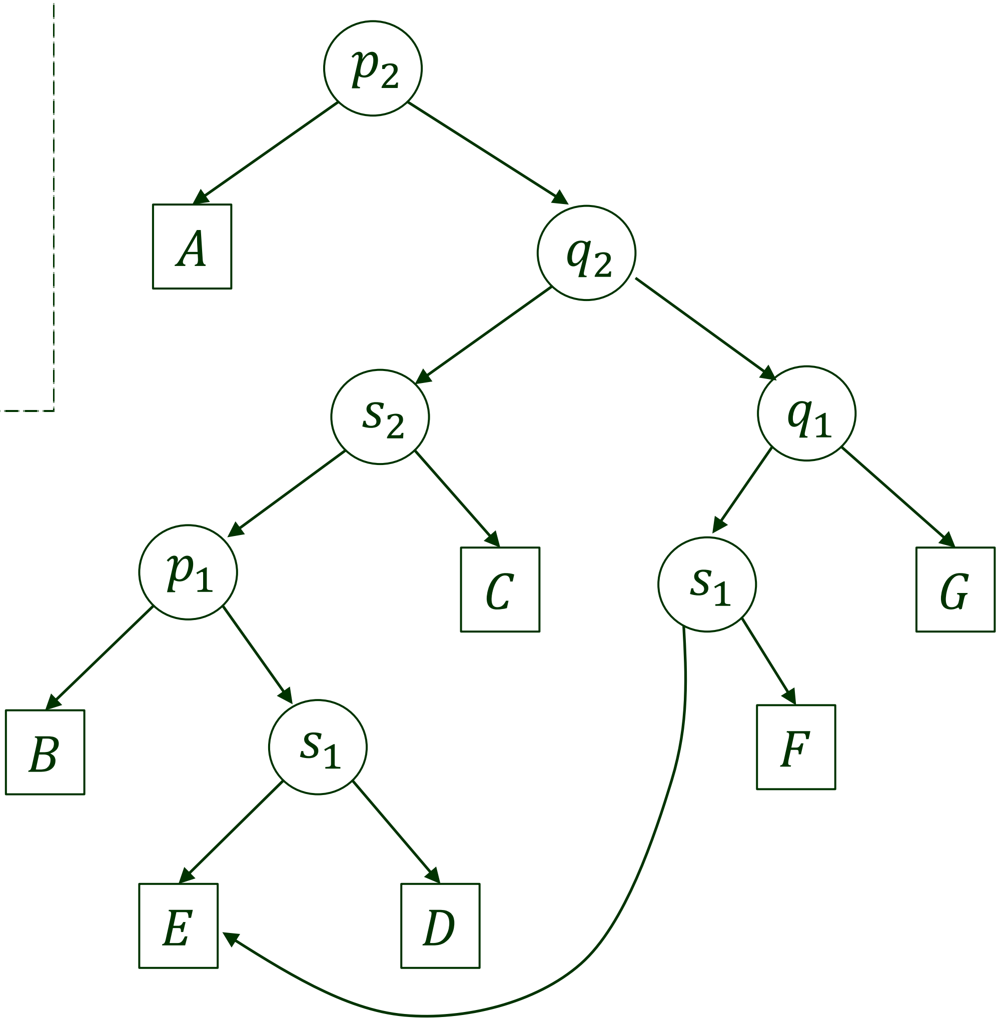
Query



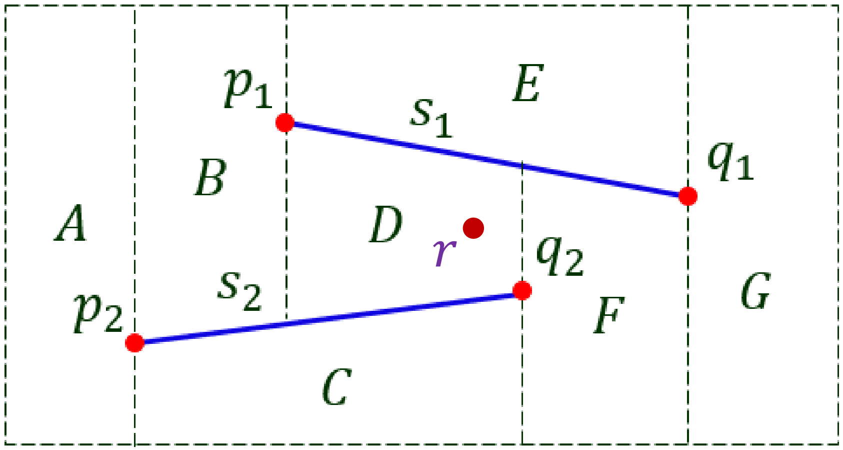


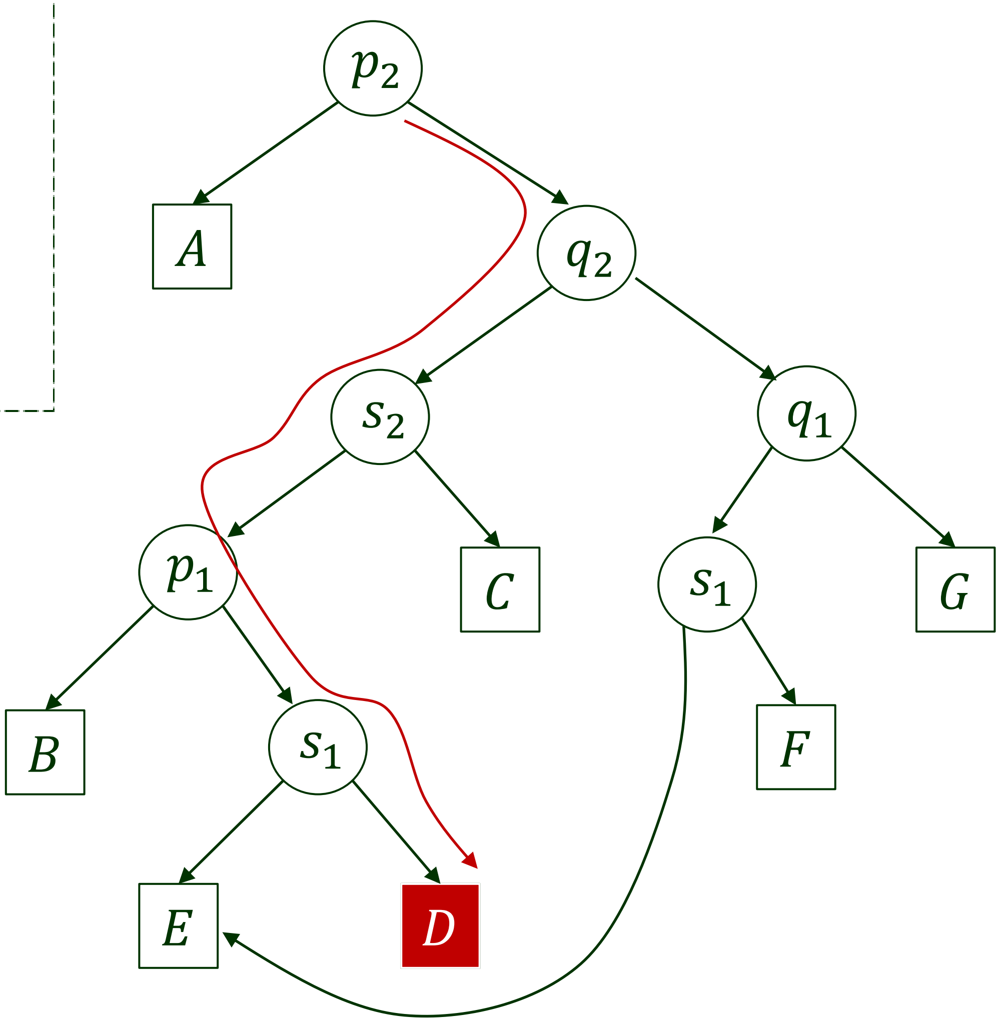
Query



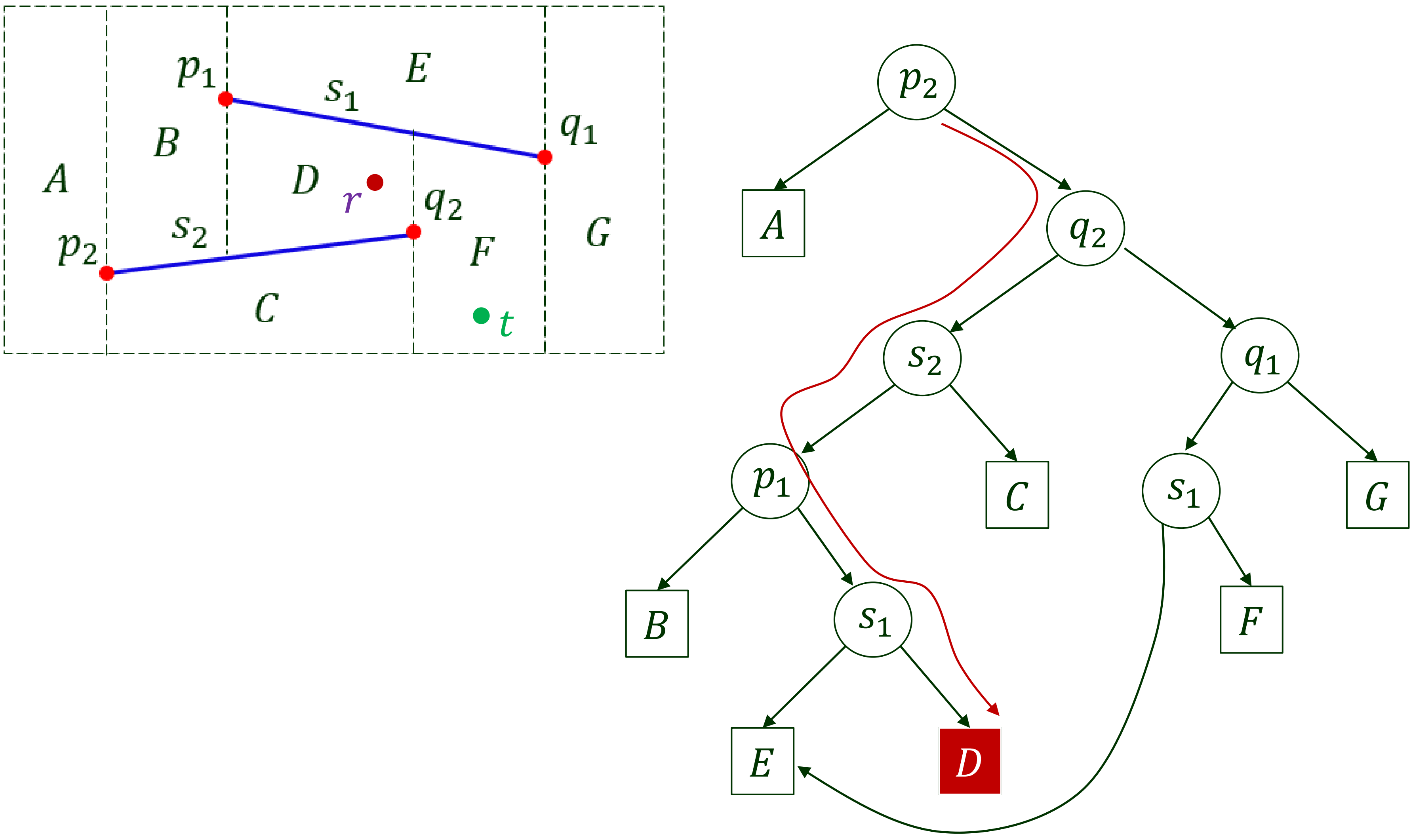


Query





Query



Query