Computational Geometry – Exercise

Point Location

LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

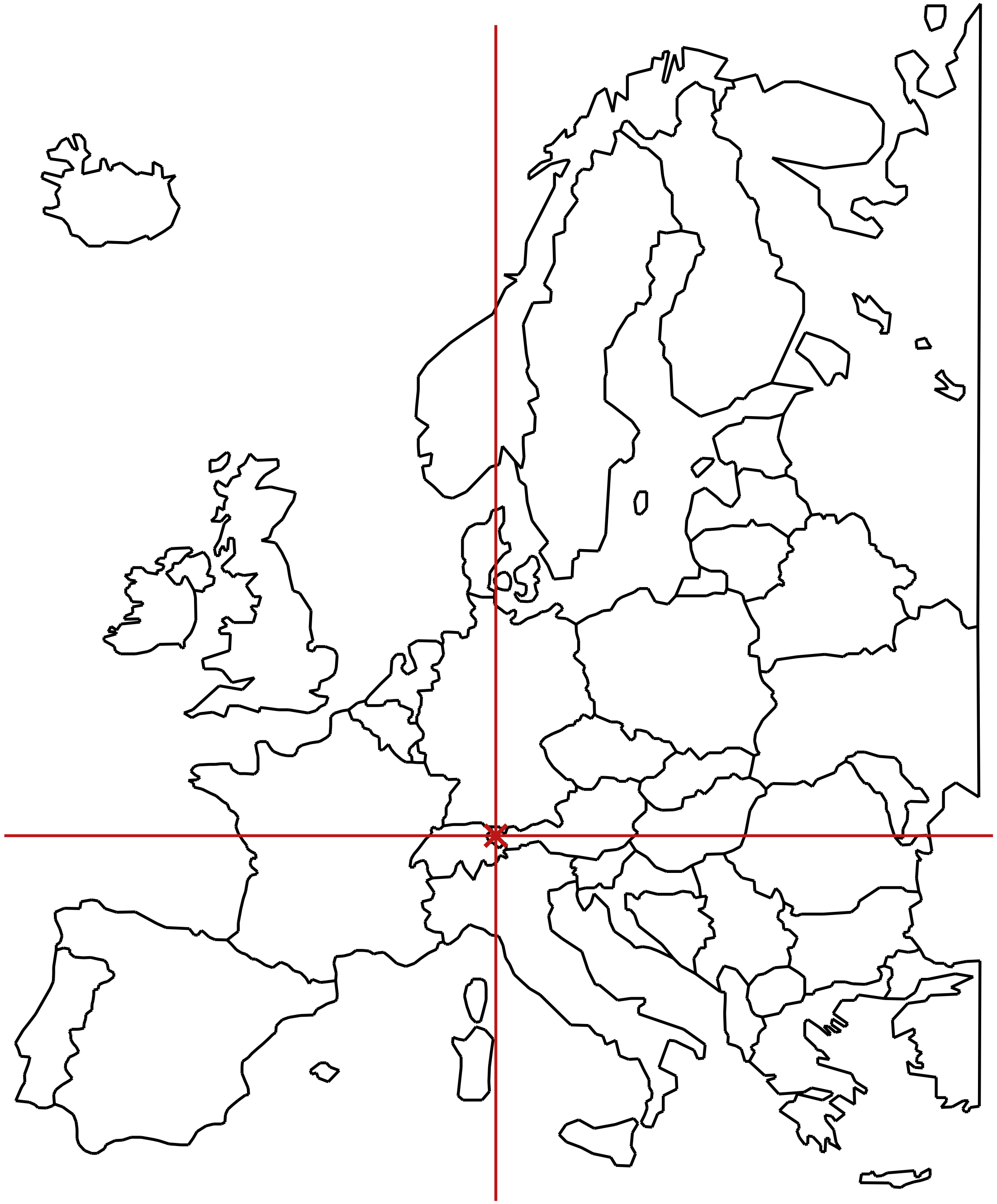
Benjamin Niedermann

10.02.2016

Motivation

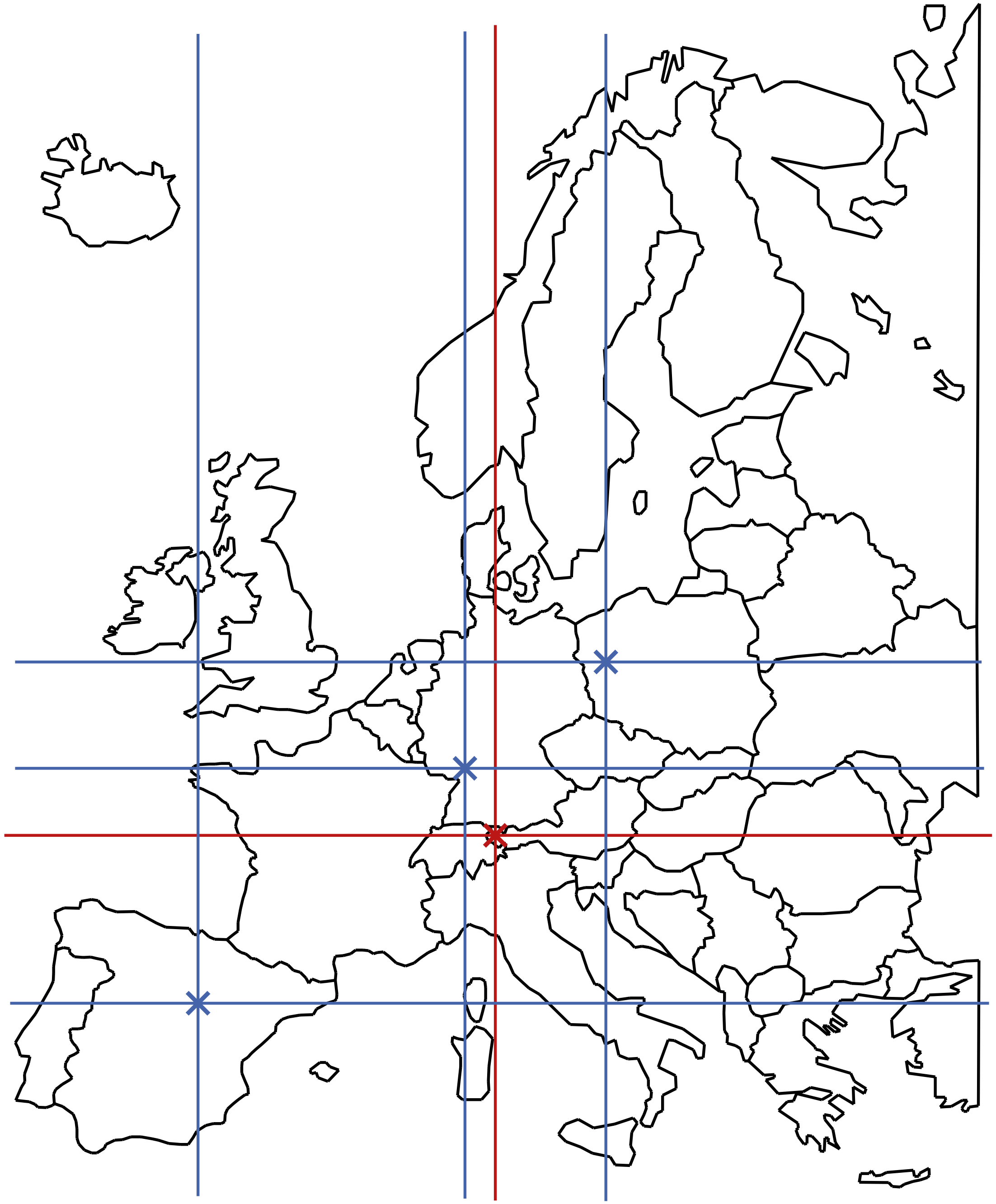
Given a position $p = (p_x, p_y)$ in a map, determine in which country p lies.

Motivation



Given a position $p = (p_x, p_y)$ in a map, determine in which country p lies.

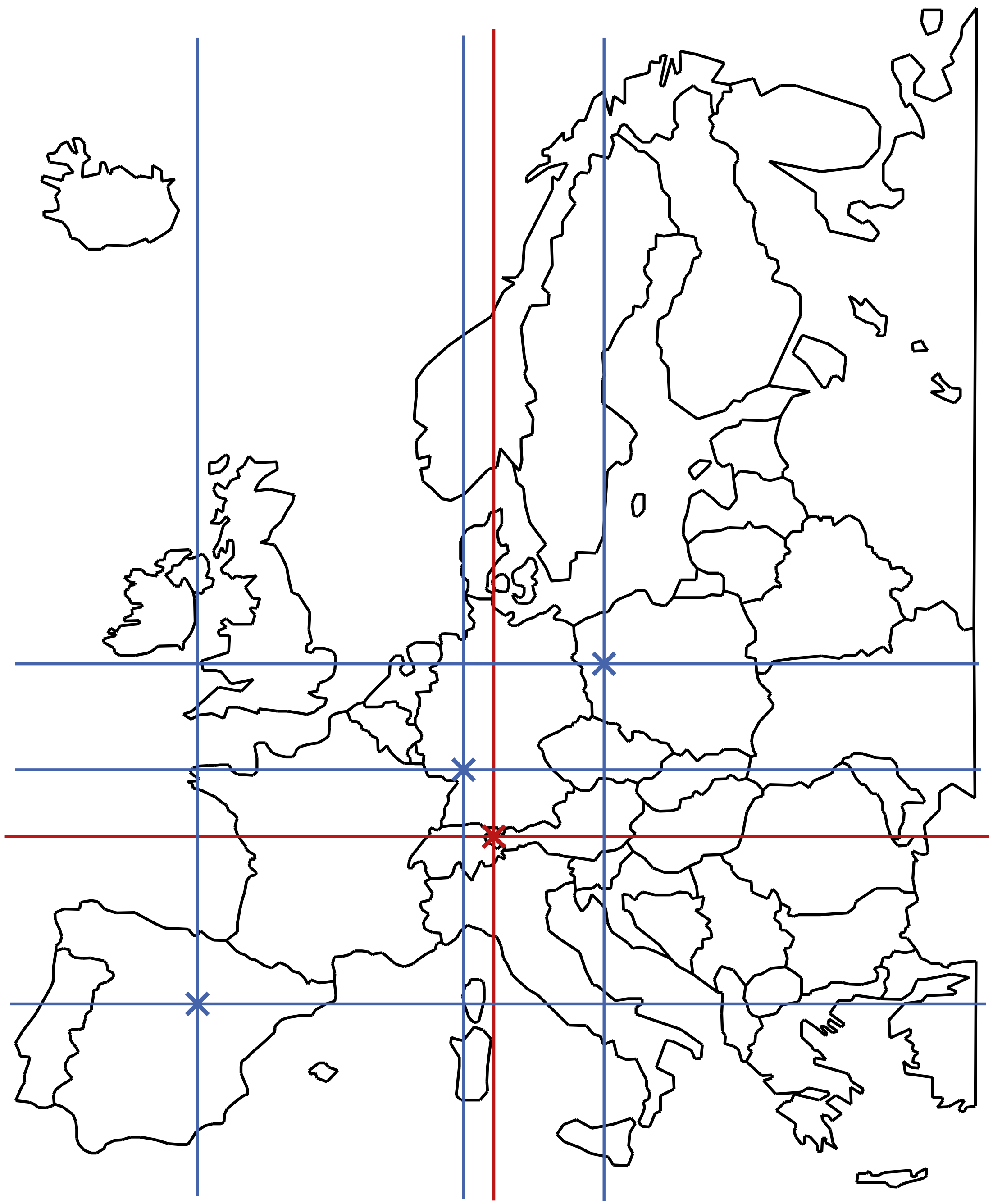
Motivation



Given a position $p = (p_x, p_y)$ in a map, determine in which country p lies.

more precisely:
Find a data structure for efficiently answering such point location queries.

Motivation

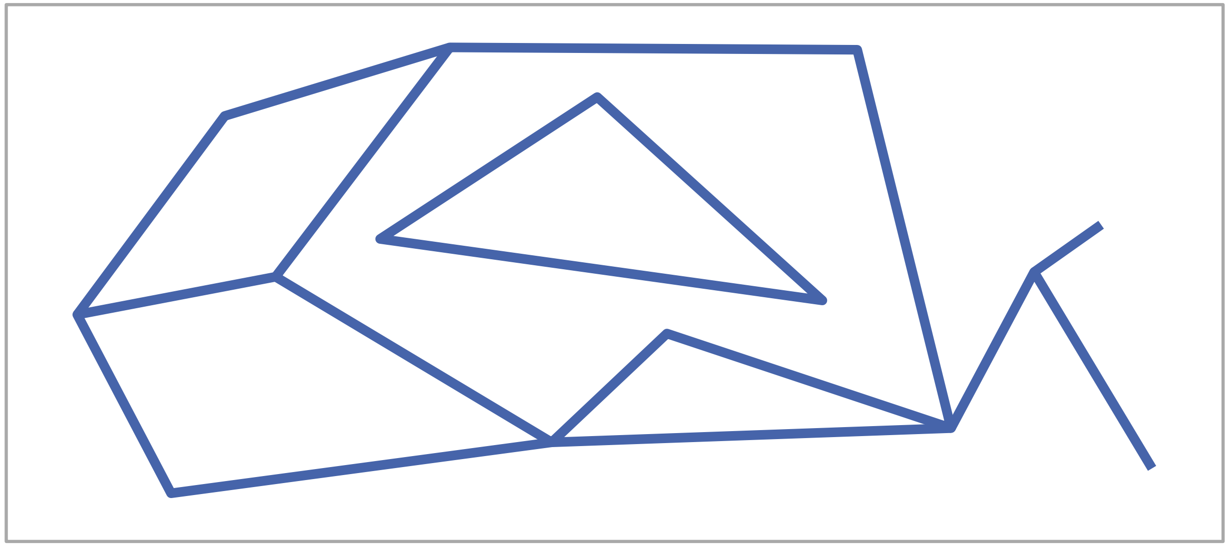


Given a position $p = (p_x, p_y)$ in a map, determine in which country p lies.

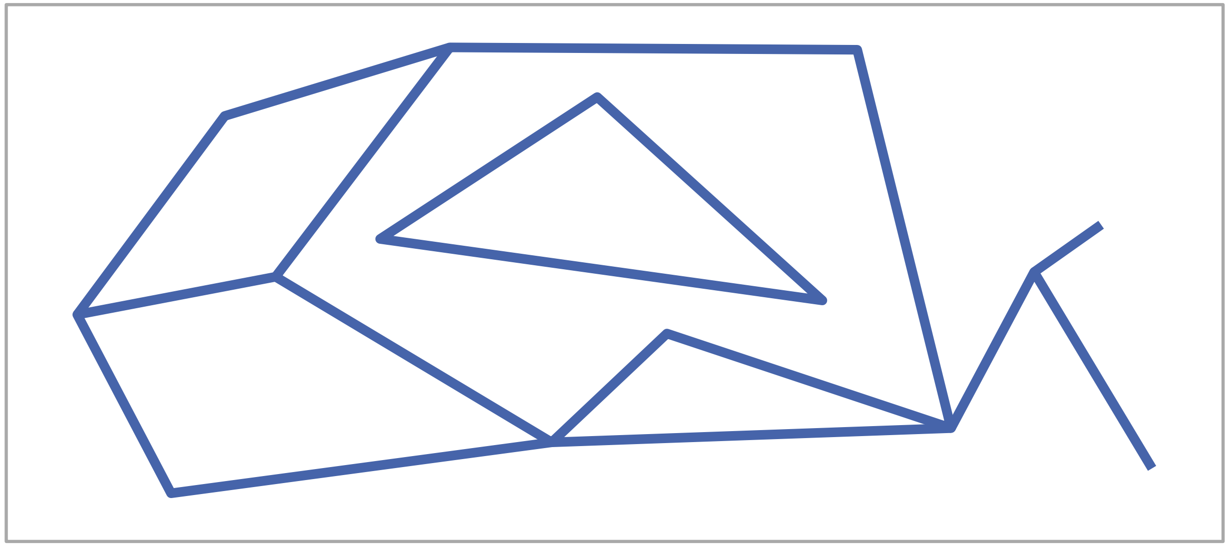
more precisely:
Find a data structure for efficiently answering such point location queries.

The map is modeled as a subdivision of the plane into disjoint polygons.

Problem Setting

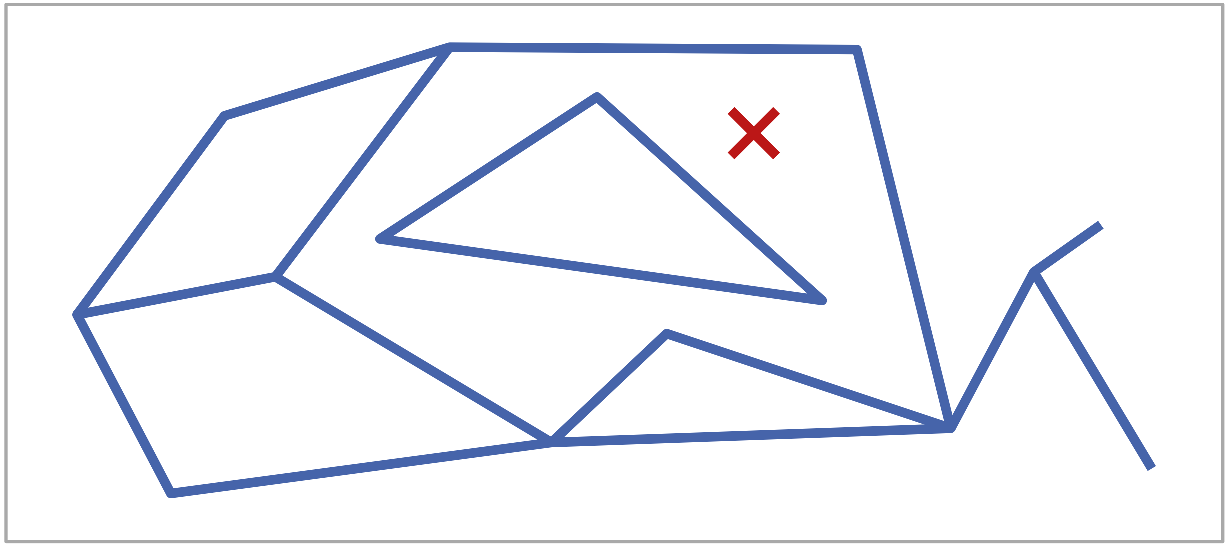


Problem Setting



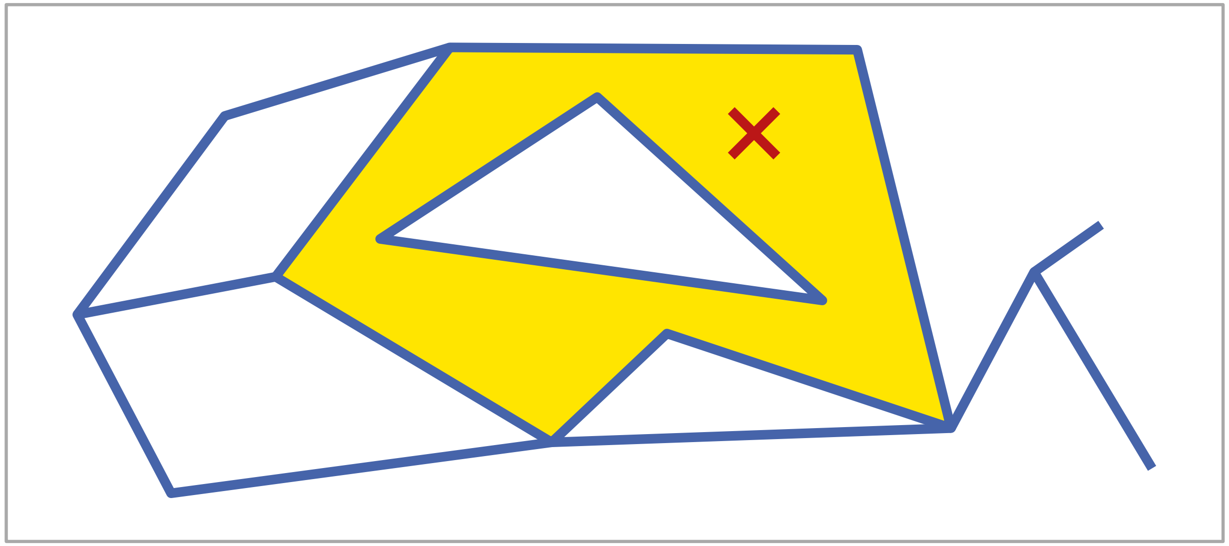
Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Problem Setting



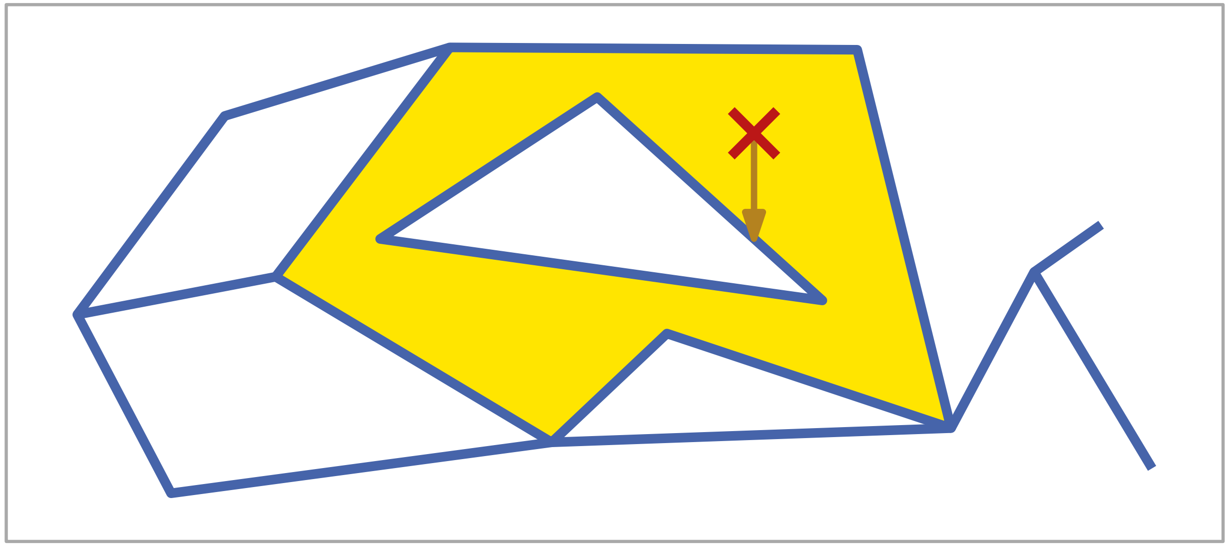
Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Problem Setting



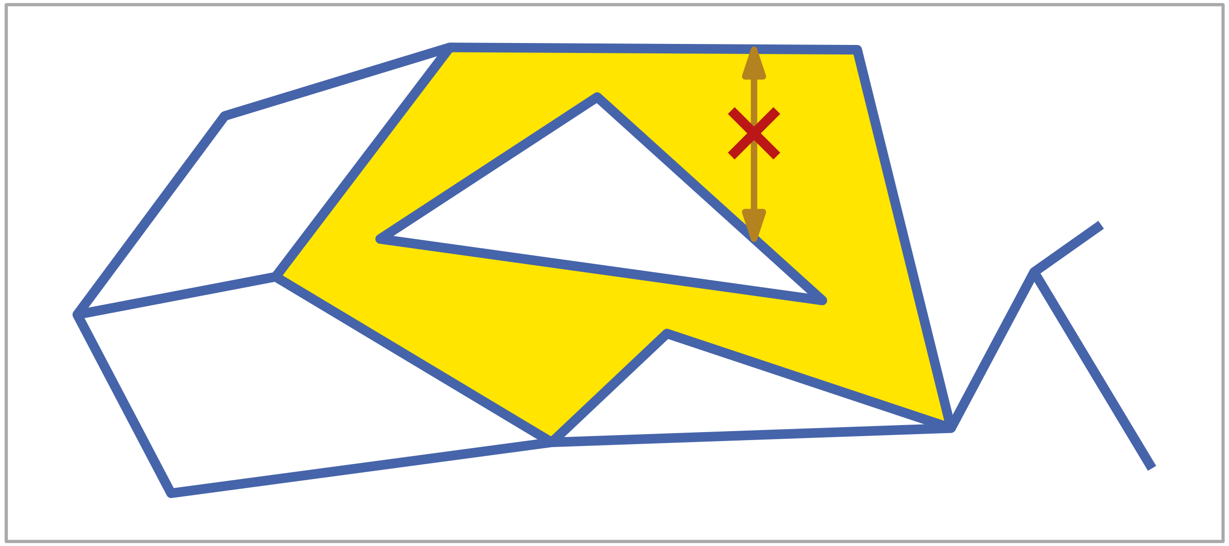
Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Problem Setting



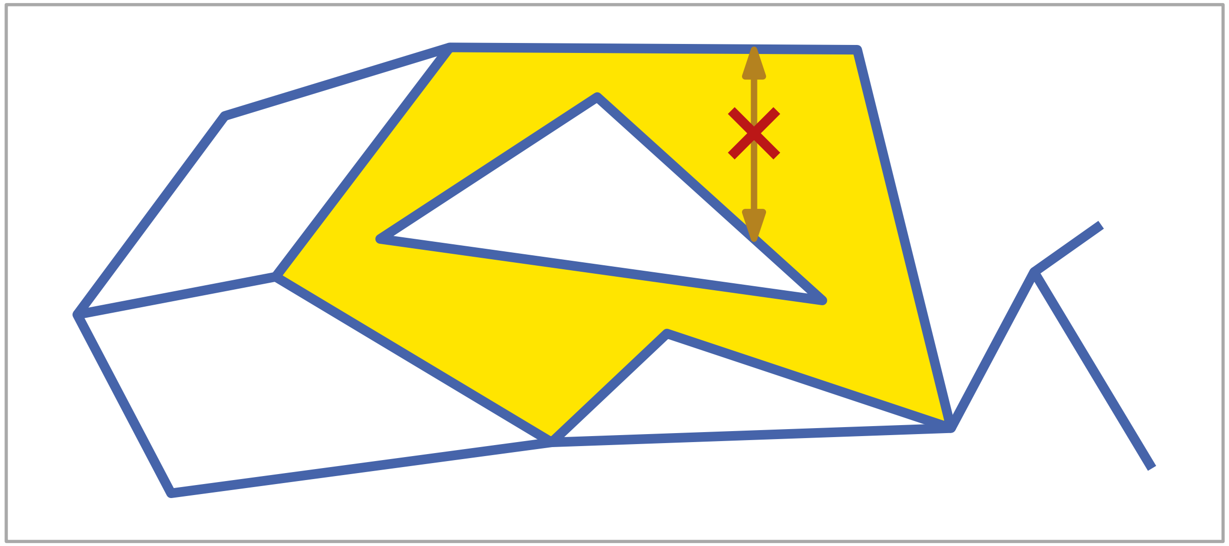
Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Problem Setting



Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

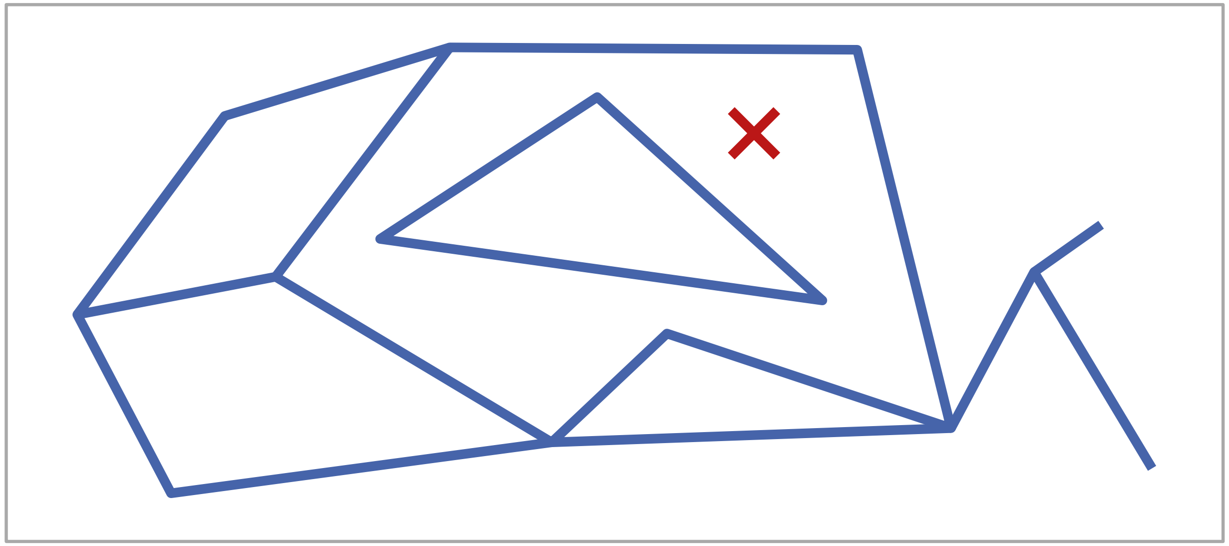
Problem Setting



Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Think for 2 minutes!

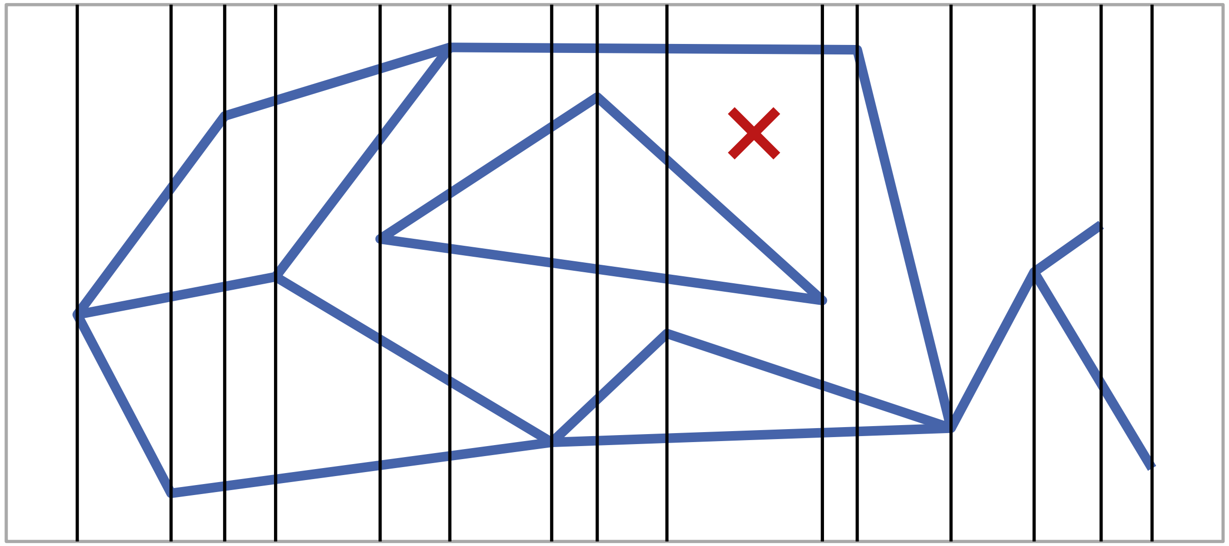
Problem Setting



Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Solution: Partition $\mathcal{S}$ at points into vertical slabs.

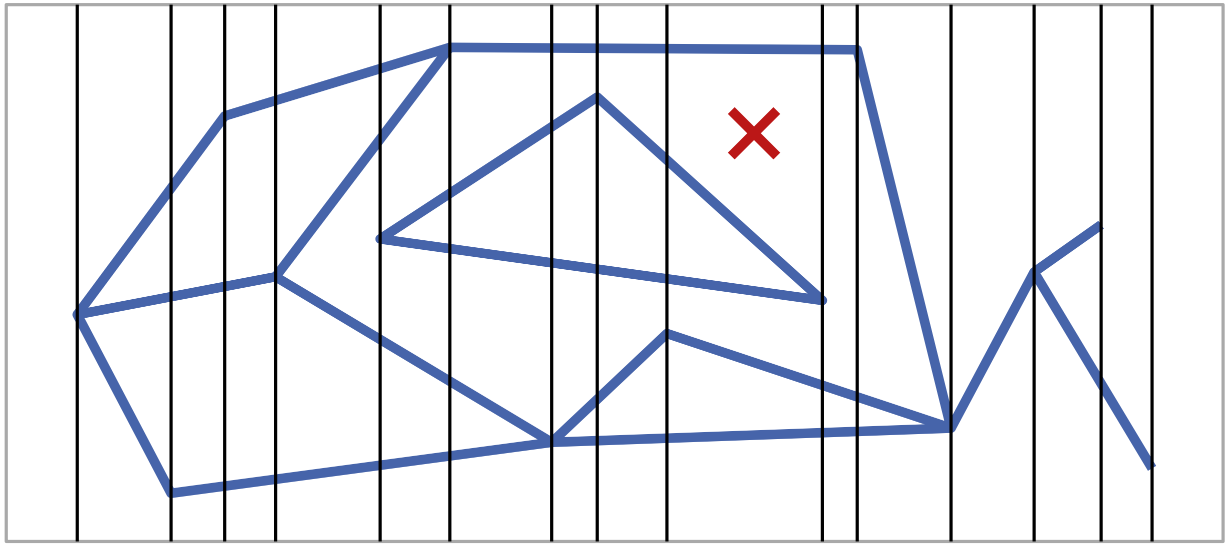
Problem Setting



Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Solution: Partition $\mathcal{S}$ at points into vertical slabs.

Problem Setting

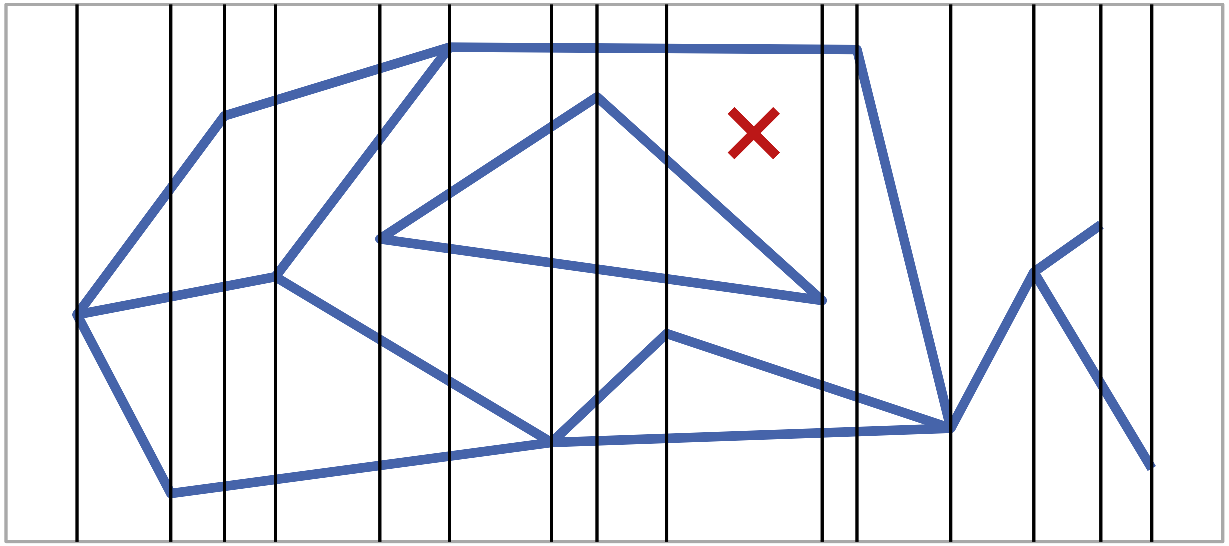


Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Solution: Partition $\mathcal{S}$ at points into vertical slabs.

Query:

Problem Setting

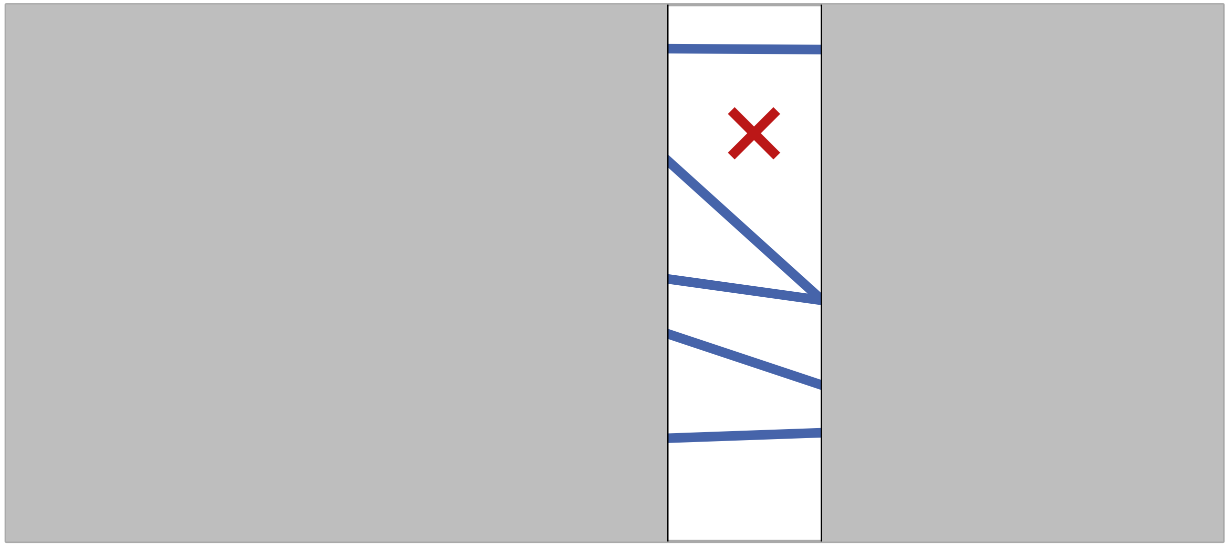


Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Solution: Partition $\mathcal{S}$ at points into vertical slabs.

Query:
- find correct slab

Problem Setting

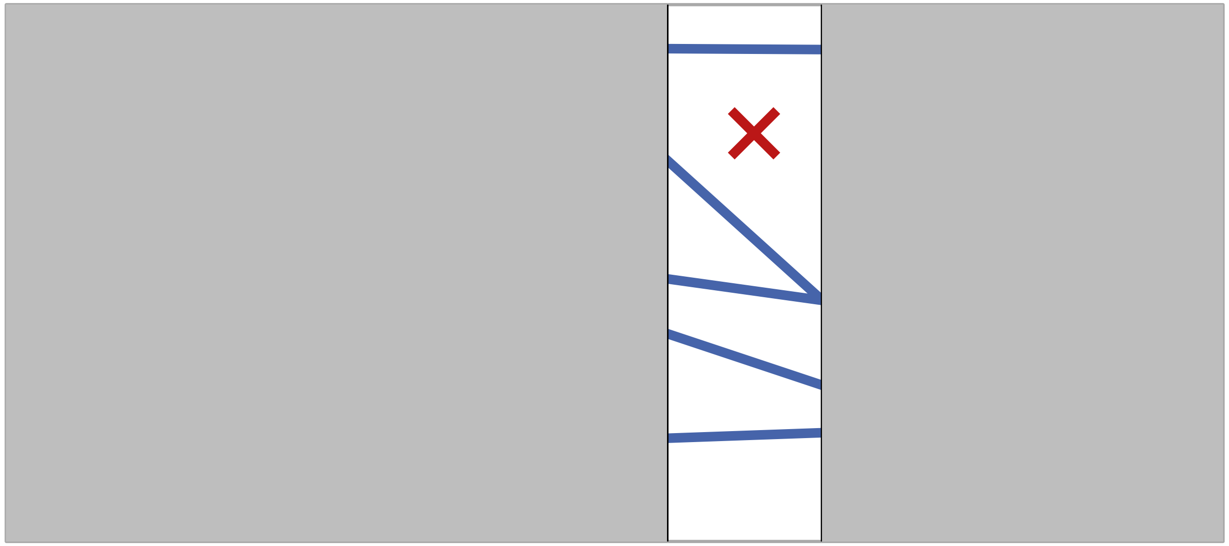


Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Solution: Partition $\mathcal{S}$ at points into vertical slabs.

Query:
- find correct slab

Problem Setting

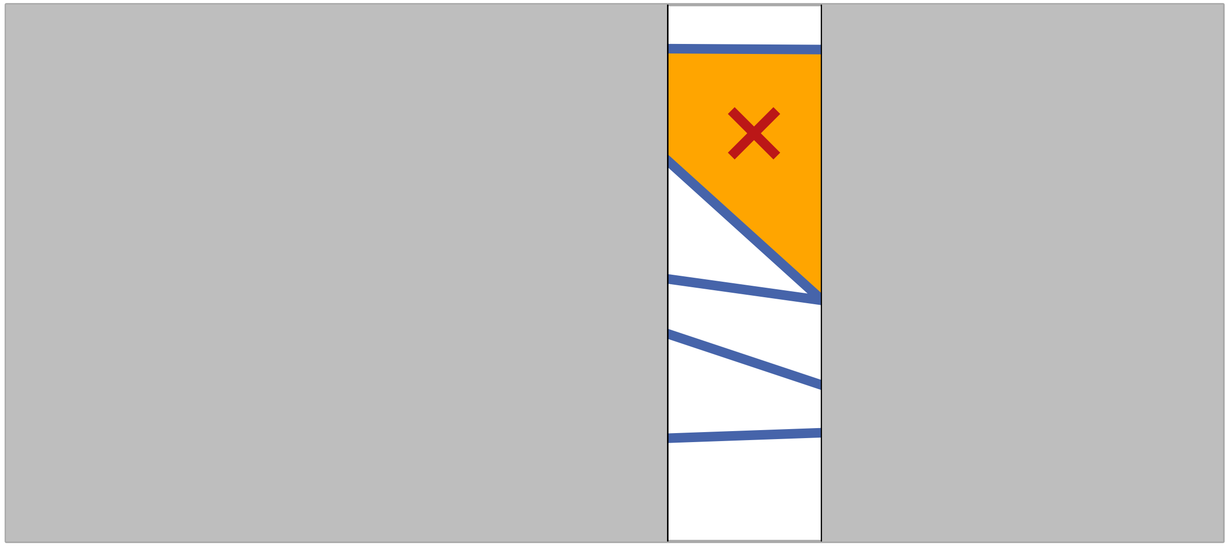


Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Solution: Partition $\mathcal{S}$ at points into vertical slabs.

Query:
- find correct slab
- search this slab

Problem Setting

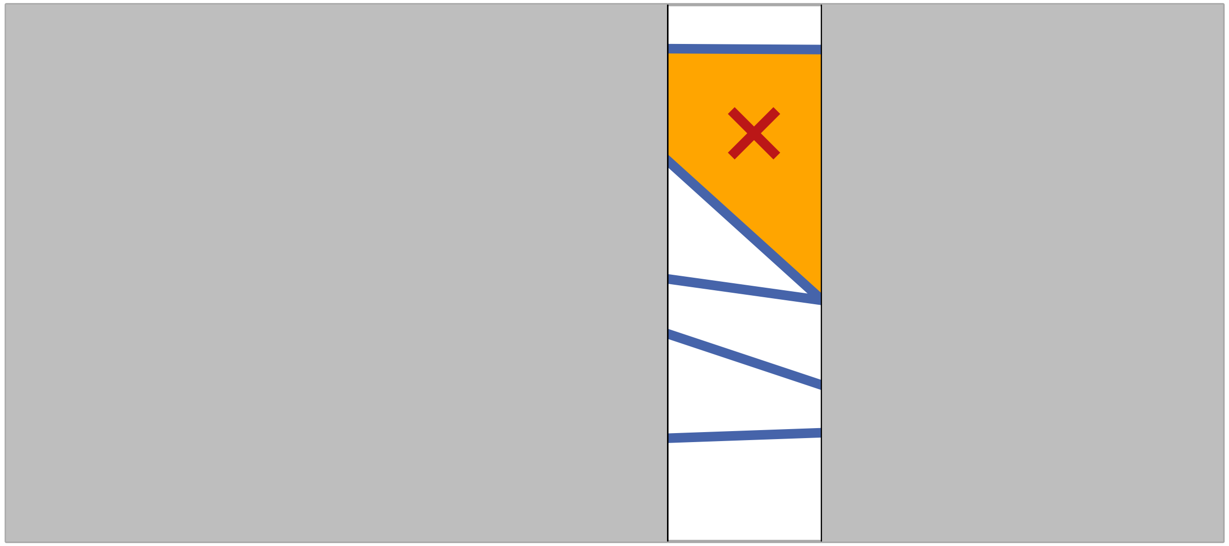


Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Solution: Partition $\mathcal{S}$ at points into vertical slabs.

Query:
- find correct slab
- search this slab

Problem Setting



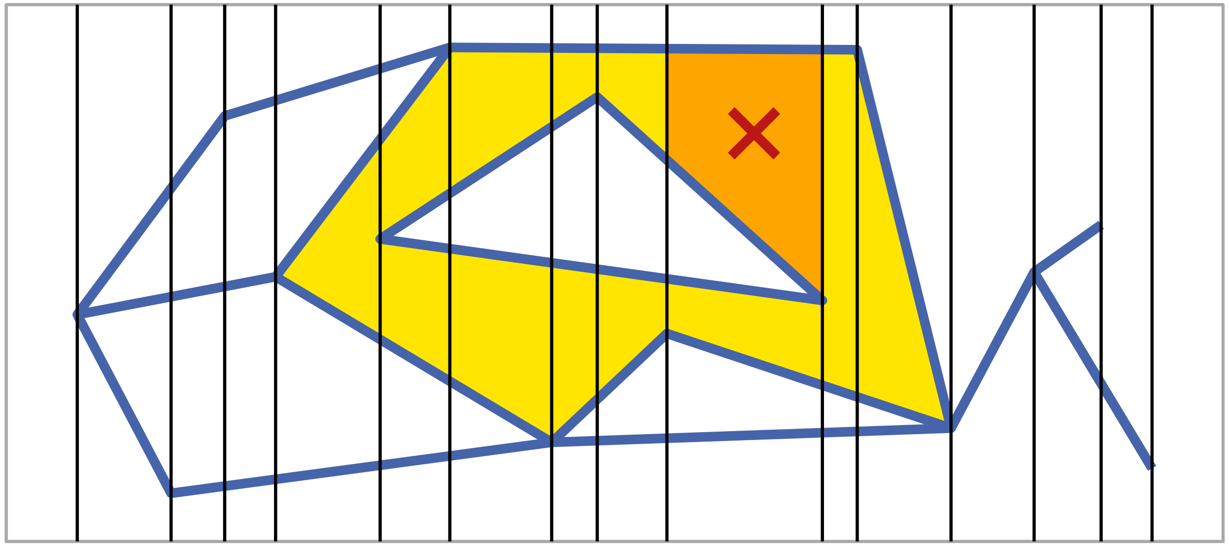
Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Solution: Partition $\mathcal{S}$ at points into vertical slabs.

Query:
- find correct slab
- search this slab

} 2 binary searches

Problem Setting



Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

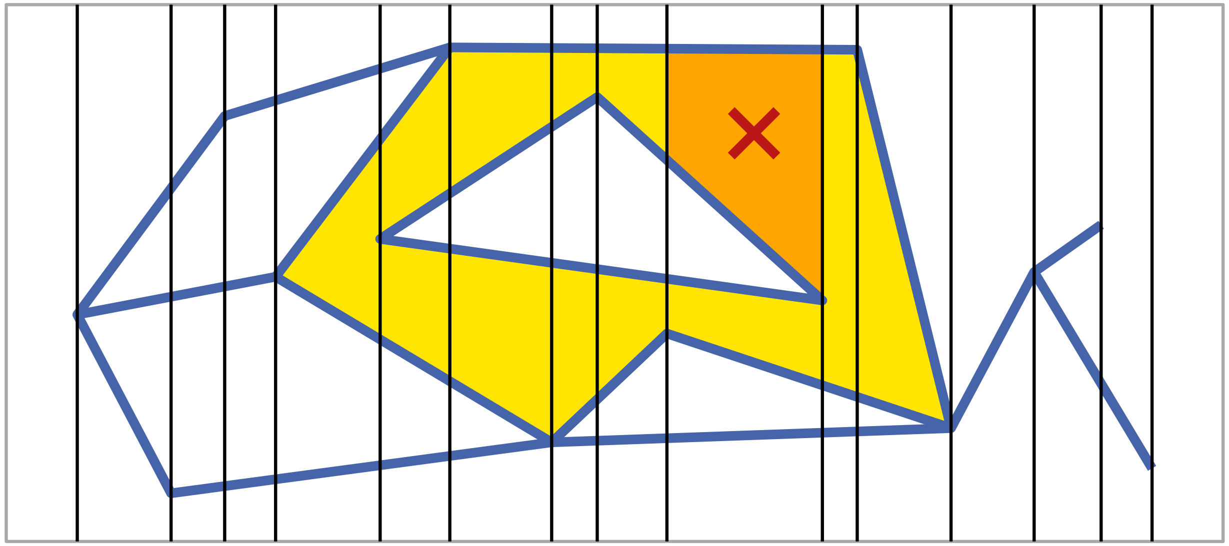
Solution: Partition $\mathcal{S}$ at points into vertical slabs.

$O(\log n)$ time

Query:
- find correct slab
- search this slab

} 2 binary searches

Problem Setting



Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Solution: Partition $\mathcal{S}$ at points into vertical slabs.

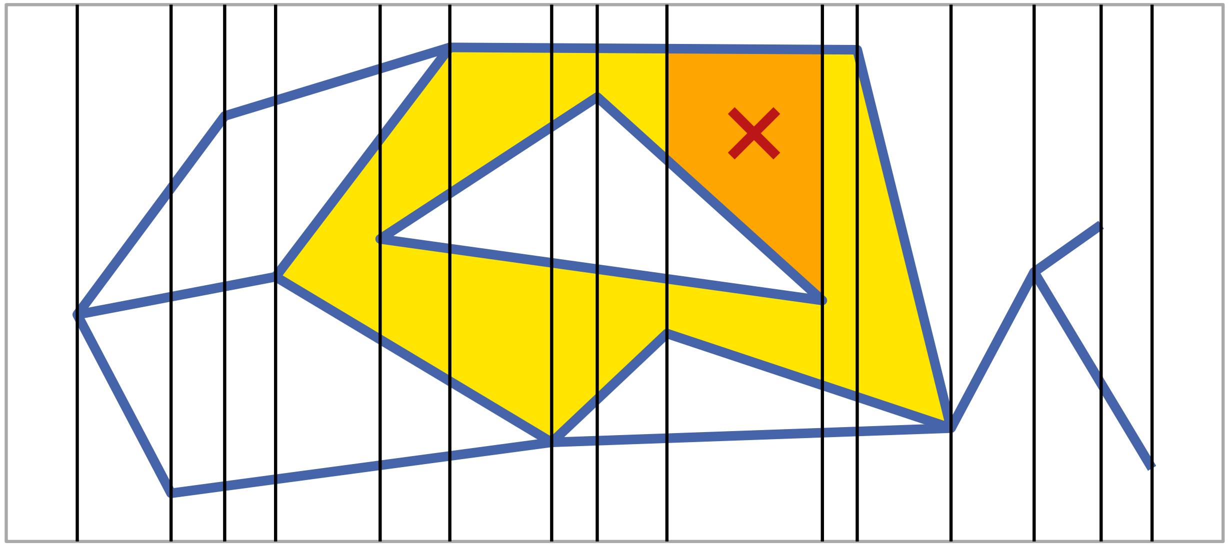
$O(\log n)$ time

Query:
- find correct slab
- search this slab

} 2 binary searches

But:

Problem Setting



Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Solution: Partition $\mathcal{S}$ at points into vertical slabs.

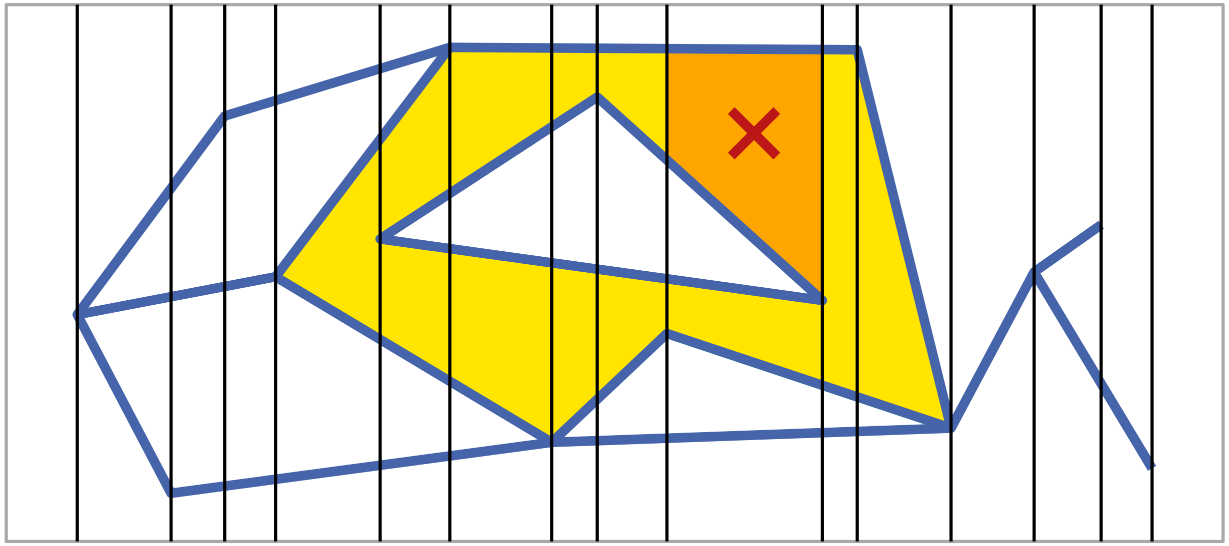
Query:
- find correct slab
- search this slab

} 2 binary searches

$O(\log n)$ time

But: Space?

Problem Setting



Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Solution: Partition $\mathcal{S}$ at points into vertical slabs.

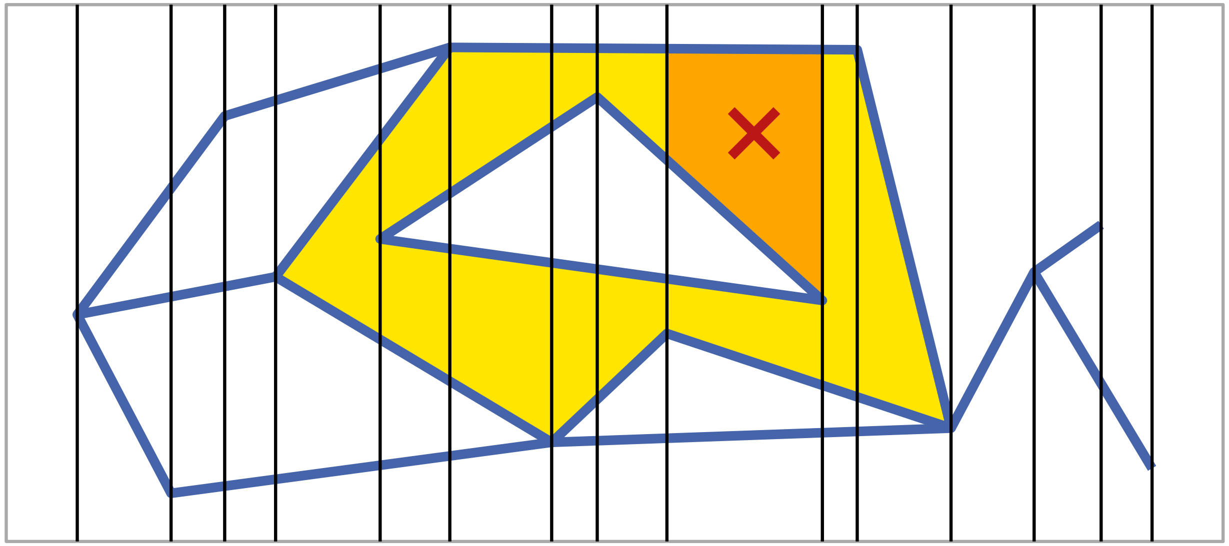
$O(\log n)$ time

Query:
- find correct slab
- search this slab

} 2 binary searches

But: Space? $\Theta(n^2)$

Problem Setting



Goal: Given subdivision $\mathcal{S}$ of the plane with n segments, construct data structure for fast point location queries.

Solution: Partition $\mathcal{S}$ at points into vertical slabs.

$O(\log n)$ time

Query:
- find correct slab
- search this slab

} 2 binary searches

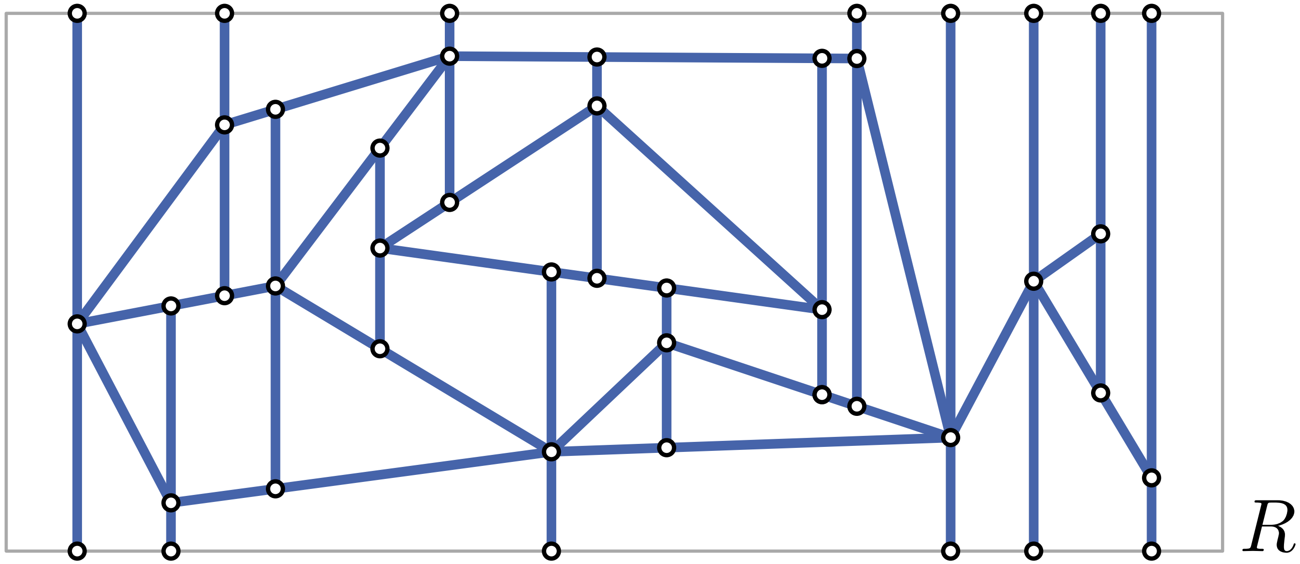
But: Space? $\Theta(n^2)$

Reducing the Complexity

Observation: Slab partition is a refinement $\mathcal{S}'$ of $\mathcal{S}$ into (possibly degenerate) trapezoids.

Goal: Find a suitable refinement of $\mathcal{S}$ with lower complexity!

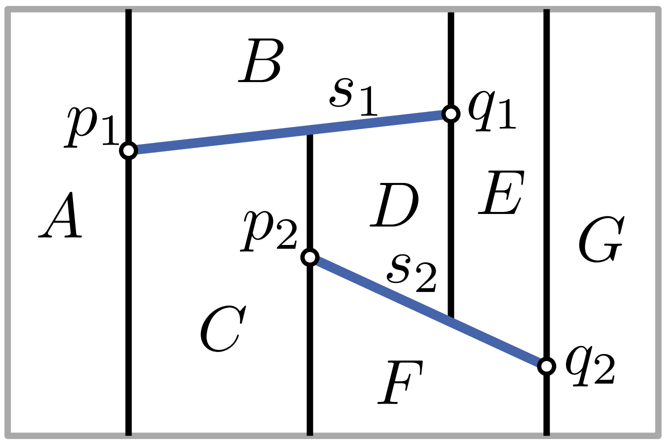
Solution: *Trapezoidal map* $\mathcal{T}(\mathcal{S})$



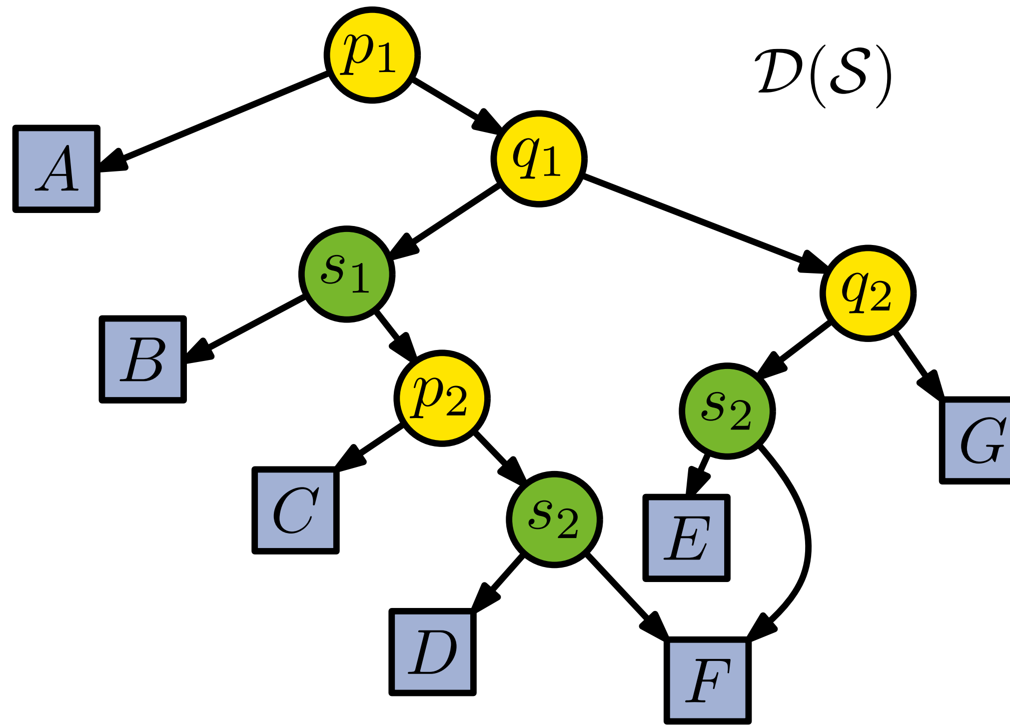
Assumption: $\mathcal{S}$ is in *general position*, i.e., no two segment endpoints have the same x-coordinate

Search Structure

Goal: Compute the trapzoidal map $\mathcal{T}(\mathcal{S})$ and simultaneously a data structure $\mathcal{D}(\mathcal{S})$ for point location in $\mathcal{T}(\mathcal{S})$.



$\mathcal{T}(\mathcal{S})$



$\mathcal{D}(\mathcal{S})$ is a DAG with:

- p — x-node for point p tests left/right of p
- s — y-node for segment s tests above/below s
- Δ — leaf node for trapezoid Δ

Analysis

Thm 1: The algorithm computes the trapezoidal map $\mathcal{T}(\mathcal{S})$ and the search structure $\mathcal{D}$ for a set $\mathcal{S}$ of n segments in *expected* $O(n \log n)$ time. The *expected* size of $\mathcal{D}$ is $O(n)$ and the *expected* query time is $O(\log n)$.

Analysis

Thm 1: The algorithm computes the trapezoidal map $\mathcal{T}(\mathcal{S})$ and the search structure $\mathcal{D}$ for a set $\mathcal{S}$ of n segments in *expected* $O(n \log n)$ time. The *expected* size of $\mathcal{D}$ is $O(n)$ and the *expected* query time is $O(\log n)$.

Observations:

- worst case: size of $\mathcal{D}$ is quadratic and query time is linear
- hope: that happens rarely!
- consider expected time and size over all $n!$ permutations of $\mathcal{S}$
- the theorem holds independently of the input set $\mathcal{S}$

Analysis

Thm 1: The algorithm computes the trapezoidal map $\mathcal{T}(\mathcal{S})$ and the search structure $\mathcal{D}$ for a set $\mathcal{S}$ of n segments in *expected* $O(n \log n)$ time. The *expected* size of $\mathcal{D}$ is $O(n)$ and the *expected* query time is $O(\log n)$.

Observations:

- worst case: size of $\mathcal{D}$ is quadratic and query time is linear
- hope: that happens rarely!
- consider expected time and size over all $n!$ permutations of $\mathcal{S}$
- the theorem holds independently of the input set $\mathcal{S}$

Exercise 1

Given: Point $q \in \mathbb{R}^2$ and polygon P
Question: Is q contained in P?

Algorithm:

1. Start in q a horizontal half-line ϱ.
2. Count the number of intersections of ϱ and edges of P.
   - Number is even: q is not contained in P
   - Number is odd: q is contained in P

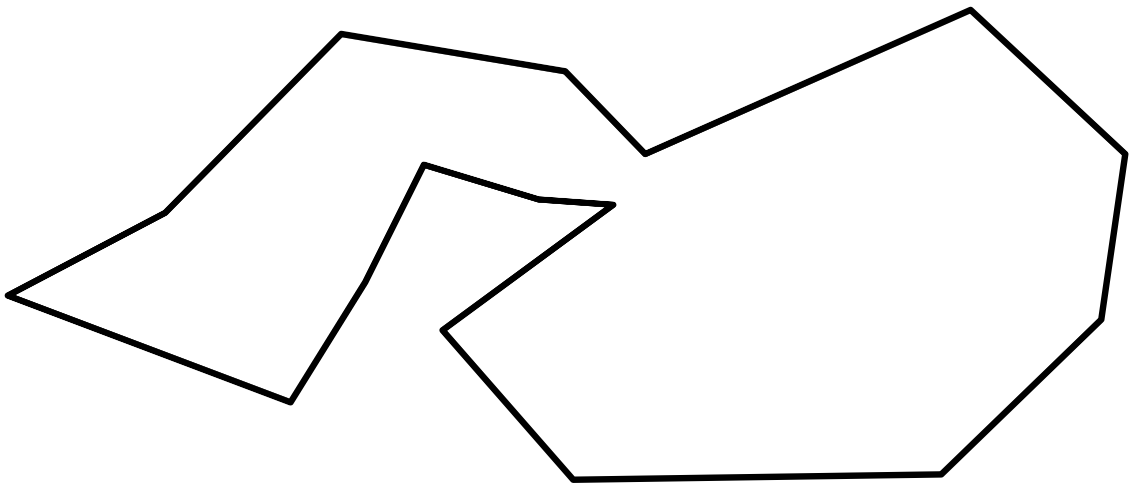
Exercise 1

Given: Point $q \in \mathbb{R}^2$ and polygon P

Question: Is q contained in P?

Algorithm:

1. Start in q a horizontal half-line ϱ.
2. Count the number of intersections of ϱ and edges of P.
   - Number is even: q is not contained in P
   - Number is odd: q is contained in P

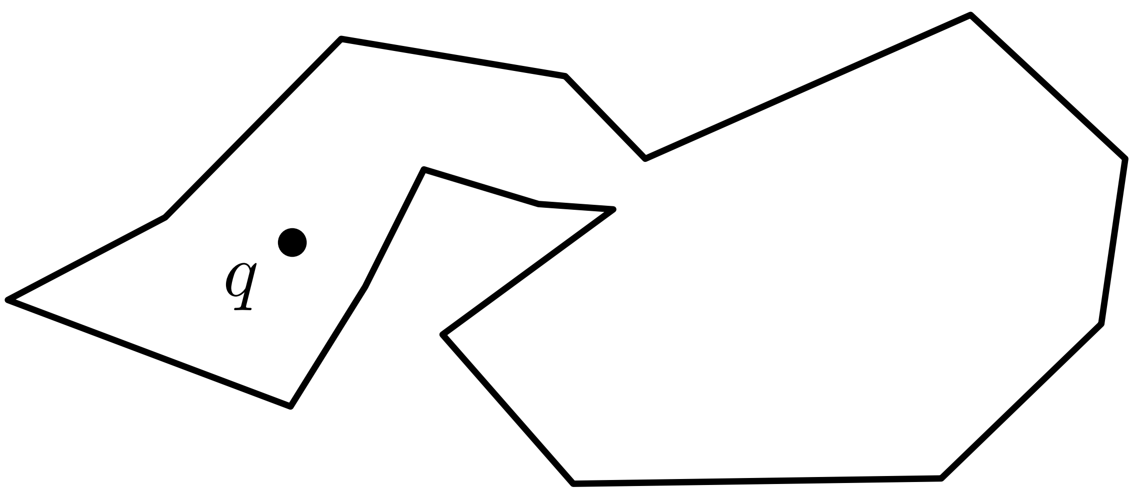


Exercise 1

Given: Point $q \in \mathbb{R}^2$ and polygon P
Question: Is q contained in P?

Algorithm:

1. Start in q a horizontal half-line ϱ.
2. Count the number of intersections of ϱ and edges of P.
   - Number is even: q is not contained in P
   - Number is odd: q is contained in P



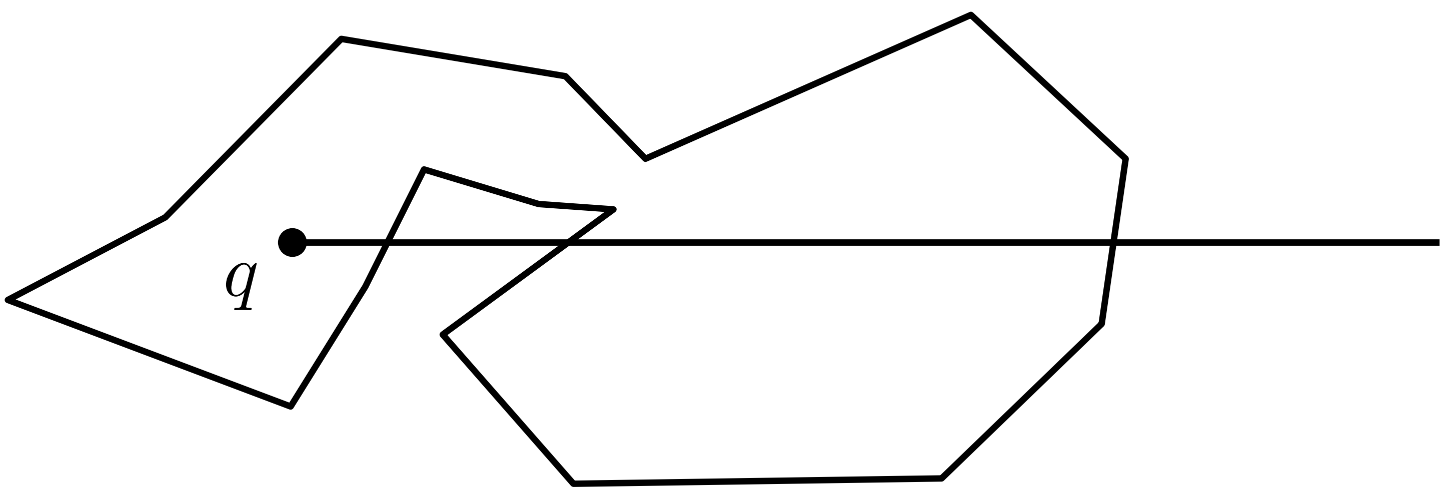
Exercise 1

Given: Point $q \in \mathbb{R}^2$ and polygon P

Question: Is q contained in P?

Algorithm:

1. Start in q a horizontal half-line ϱ.
2. Count the number of intersections of ϱ and edges of P.
   - Number is even: q is not contained in P
   - Number is odd: q is contained in P

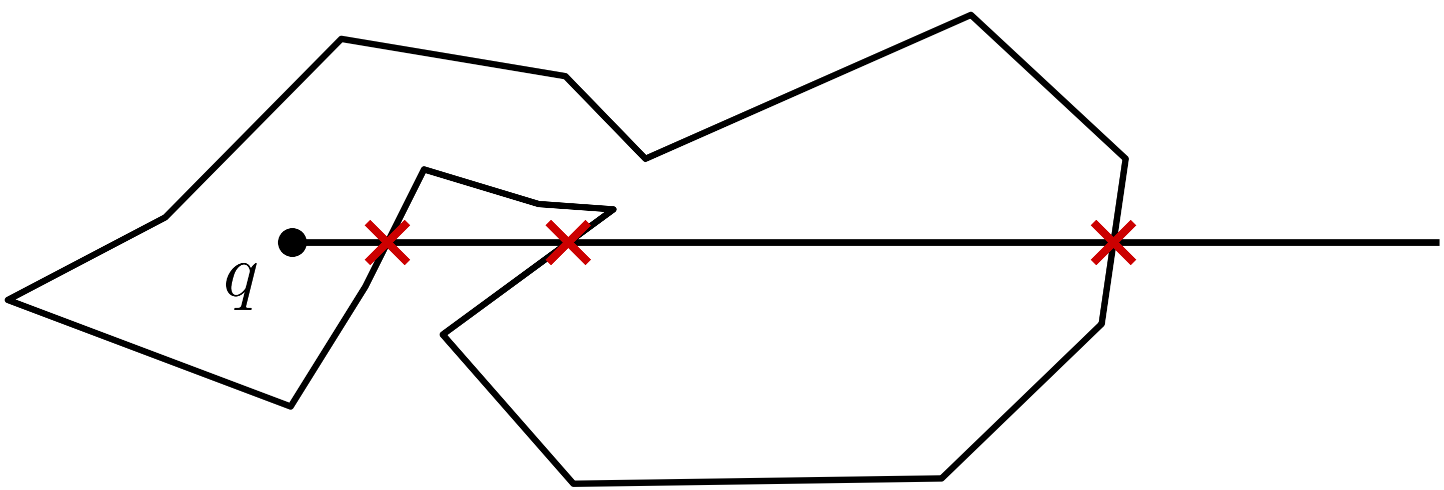


Exercise 1

Given: Point $q \in \mathbb{R}^2$ and polygon P
Question: Is q contained in P?

Algorithm:

1. Start in q a horizontal half-line ϱ.
2. Count the number of intersections of ϱ and edges of P.
   - Number is even: q is not contained in P
   - Number is odd: q is contained in P

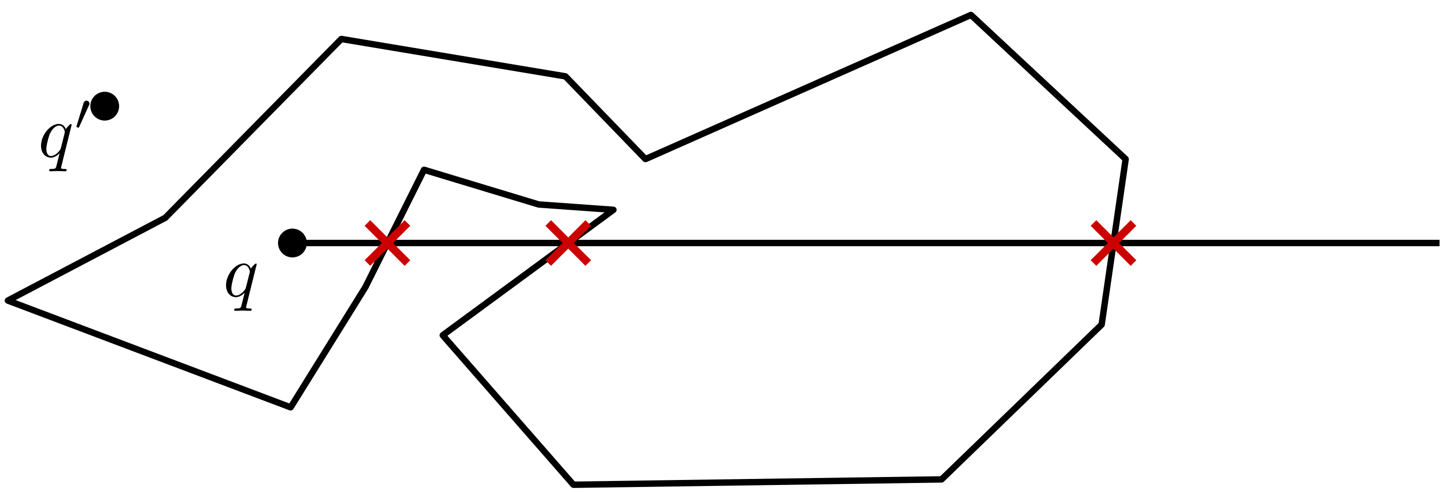


Exercise 1

Given: Point $q \in \mathbb{R}^2$ and polygon P
Question: Is q contained in P?

Algorithm:

1. Start in q a horizontal half-line ϱ.
2. Count the number of intersections of ϱ and edges of P.
   - Number is even: q is not contained in P
   - Number is odd: q is contained in P



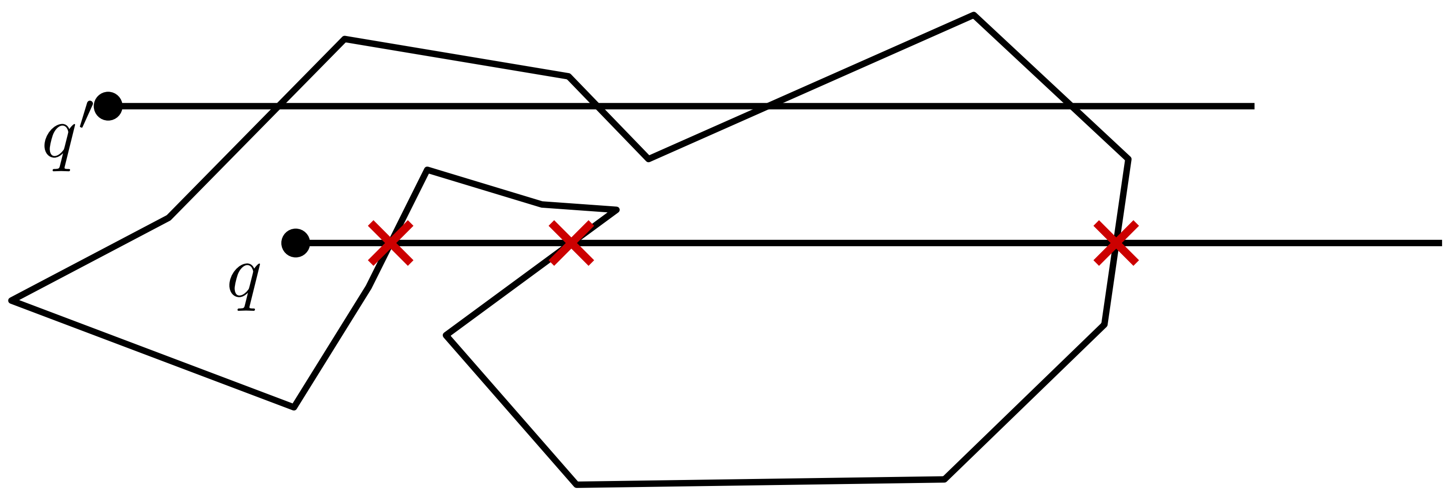
Exercise 1

Given: Point $q \in \mathbb{R}^2$ and polygon P

Question: Is q contained in P?

Algorithm:

1. Start in q a horizontal half-line ϱ.
2. Count the number of intersections of ϱ and edges of P.
   - Number is even: q is not contained in P
   - Number is odd: q is contained in P

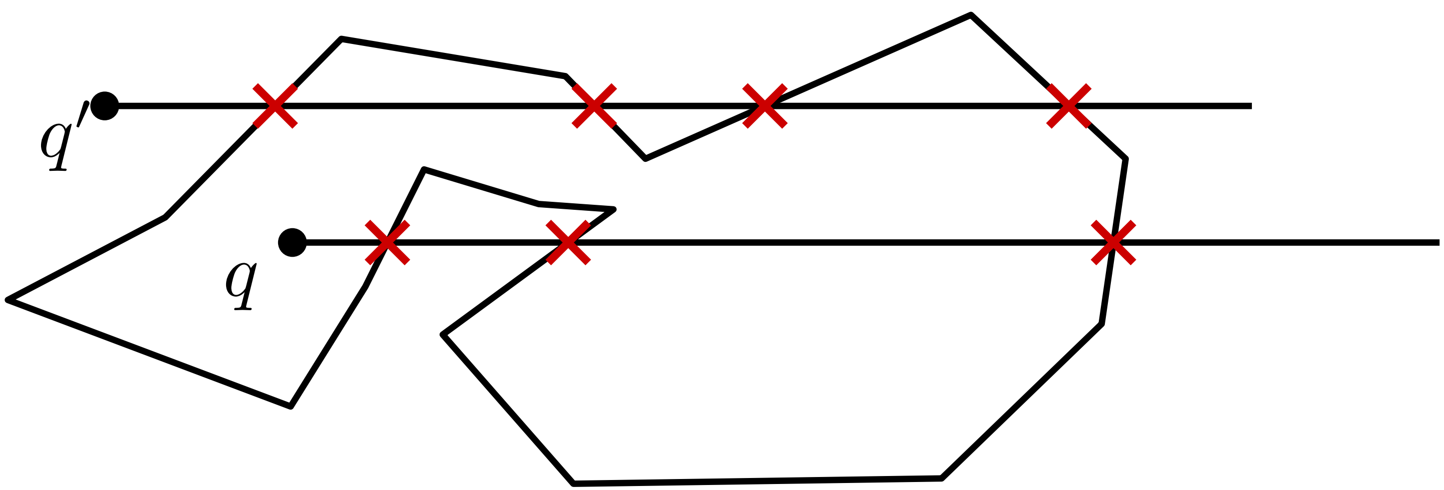


Exercise 1

Given: Point $q \in \mathbb{R}^2$ and polygon P
Question: Is q contained in P?

Algorithm:

1. Start in q a horizontal half-line ϱ.
2. Count the number of intersections of ϱ and edges of P.
   - Number is even: q is not contained in P
   - Number is odd: q is contained in P

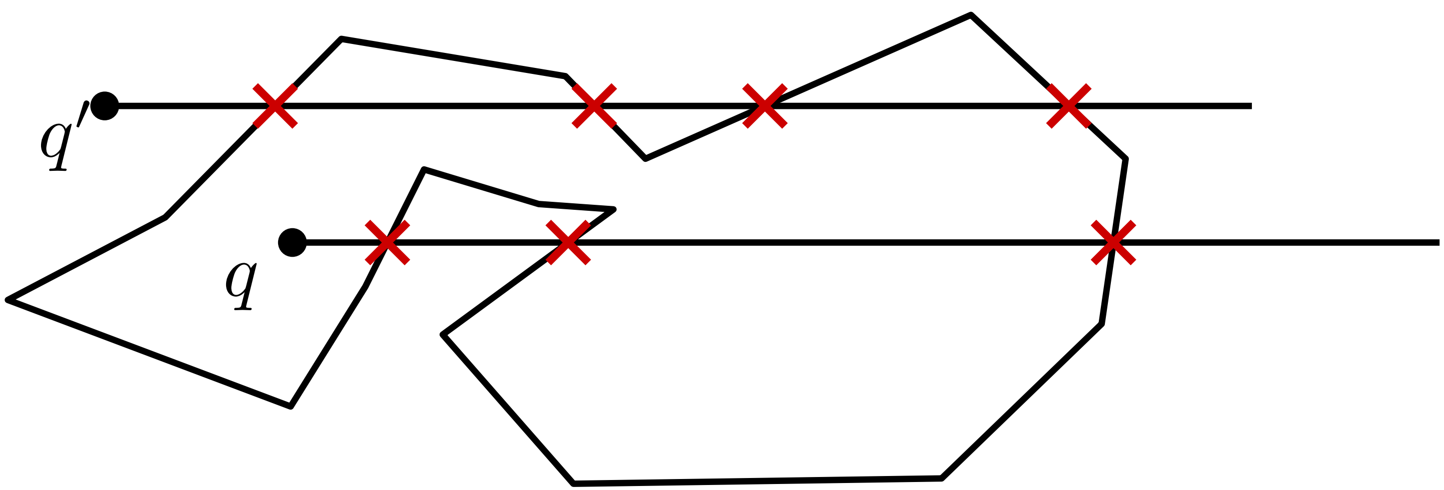


Exercise 1

Given: Point $q \in \mathbb{R}^2$ and polygon P
Question: Is q contained in P?

Algorithm:

1. Start in q a horizontal half-line ϱ.
2. Count the number of intersections of ϱ and edges of P.
   - Number is even: q is not contained in P
   - Number is odd: q is contained in P



a) **Correctness**

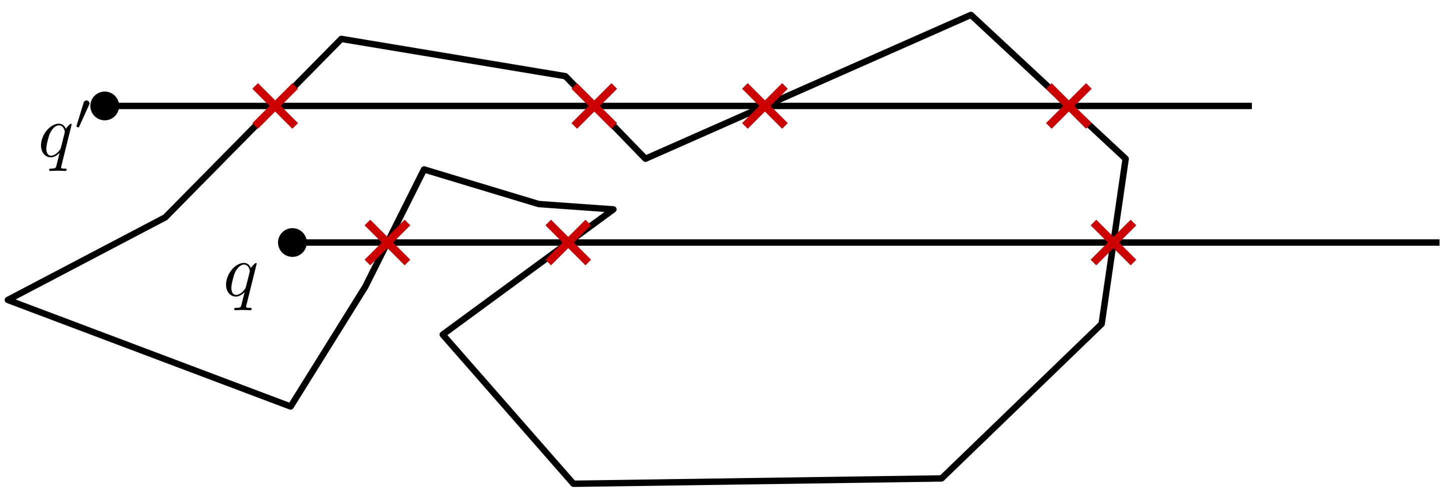
Exercise 1

Given: Point $q \in \mathbb{R}^2$ and polygon P

Question: Is q contained in P?

Algorithm:

1. Start in q a horizontal half-line ϱ.
2. Count the number of intersections of ϱ and edges of P.
   - Number is even: q is not contained in P
   - Number is odd: q is contained in P



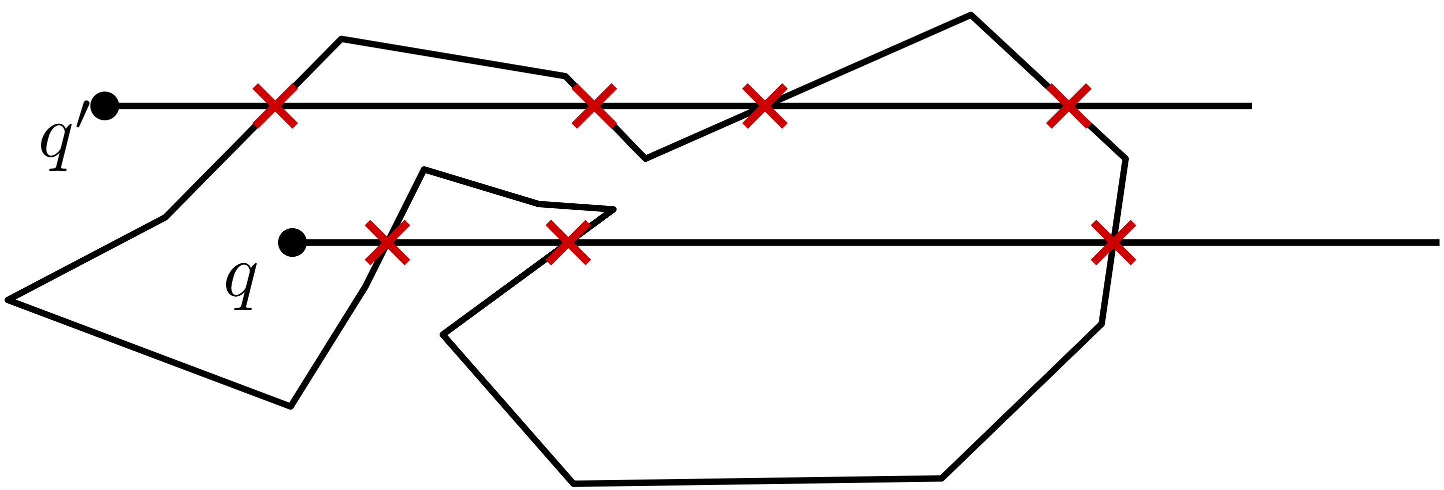
b) **Degenerated cases?**

Exercise 1

Given: Point $q \in \mathbb{R}^2$ and polygon P
Question: Is q contained in P?

Algorithm:

1. Start in q a horizontal half-line ϱ.
2. Count the number of intersections of ϱ and edges of P.
   - Number is even: q is not contained in P
   - Number is odd: q is contained in P



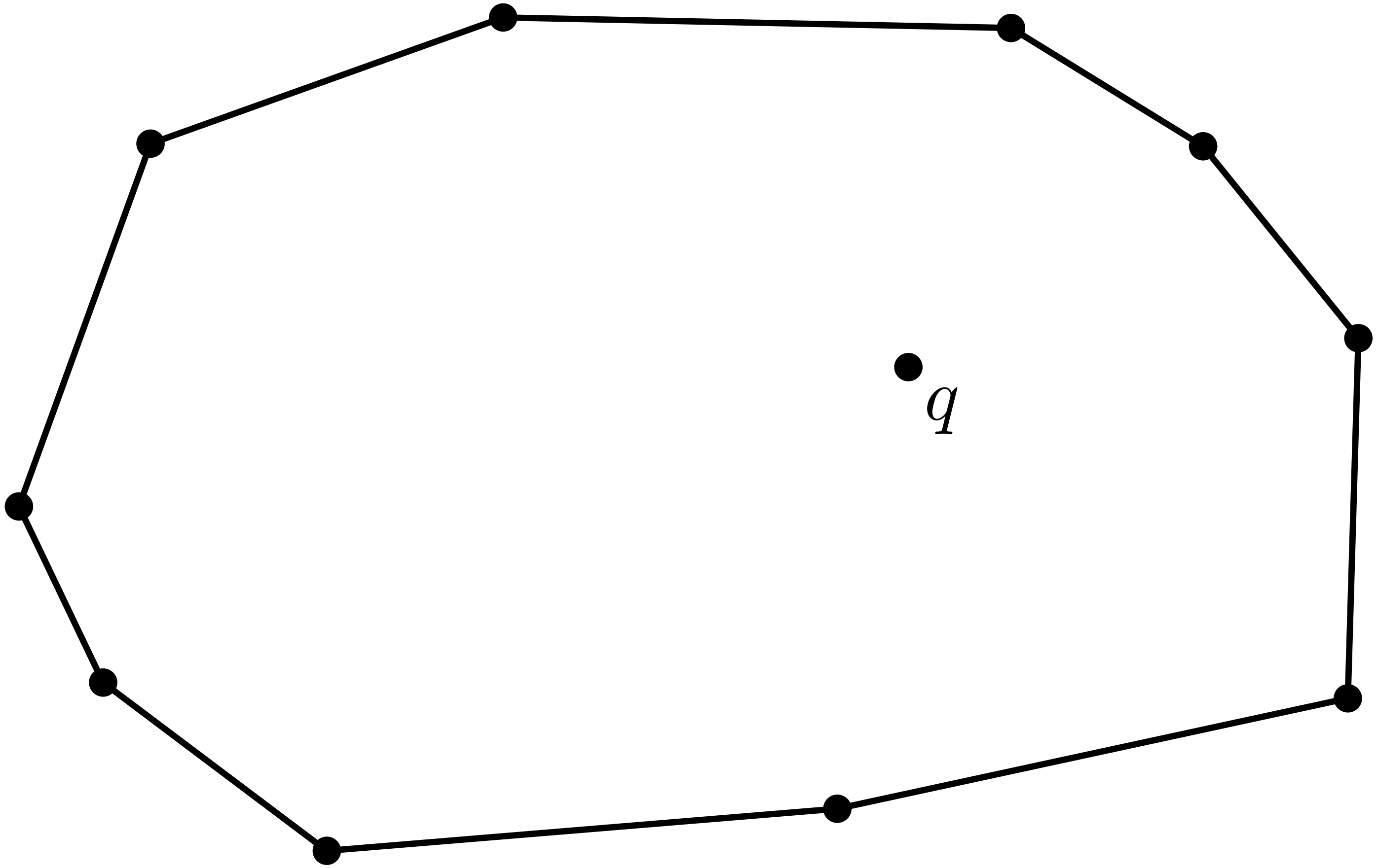
c) **Running time?**

Exercise 2

Given:

- Point q
- convexes polygon P of n points.

a) Is q contained in P? $\mathcal{O}(\log n)$

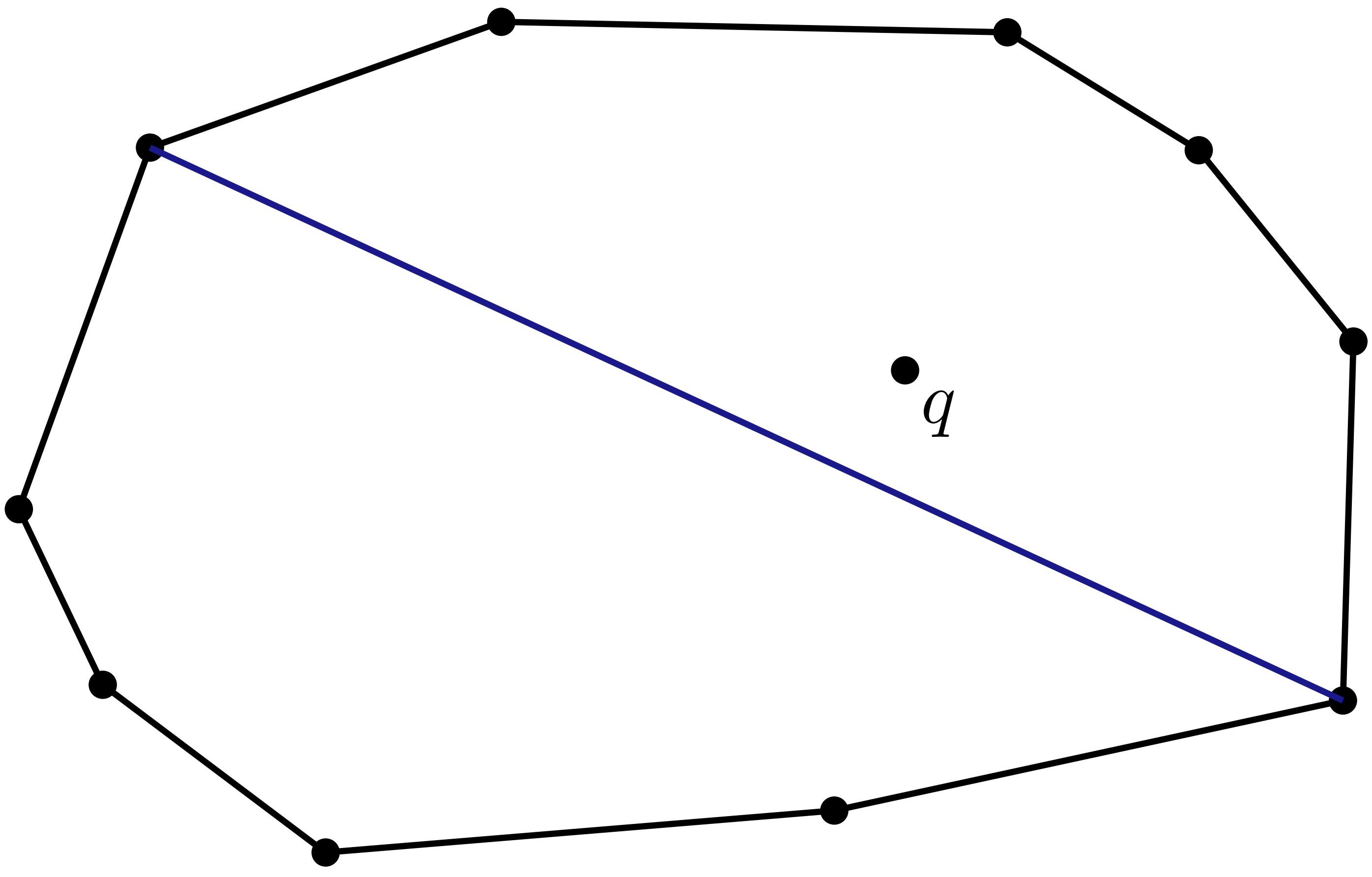


Exercise 2

Given:

- Point q
- convexes polygon P of n points.

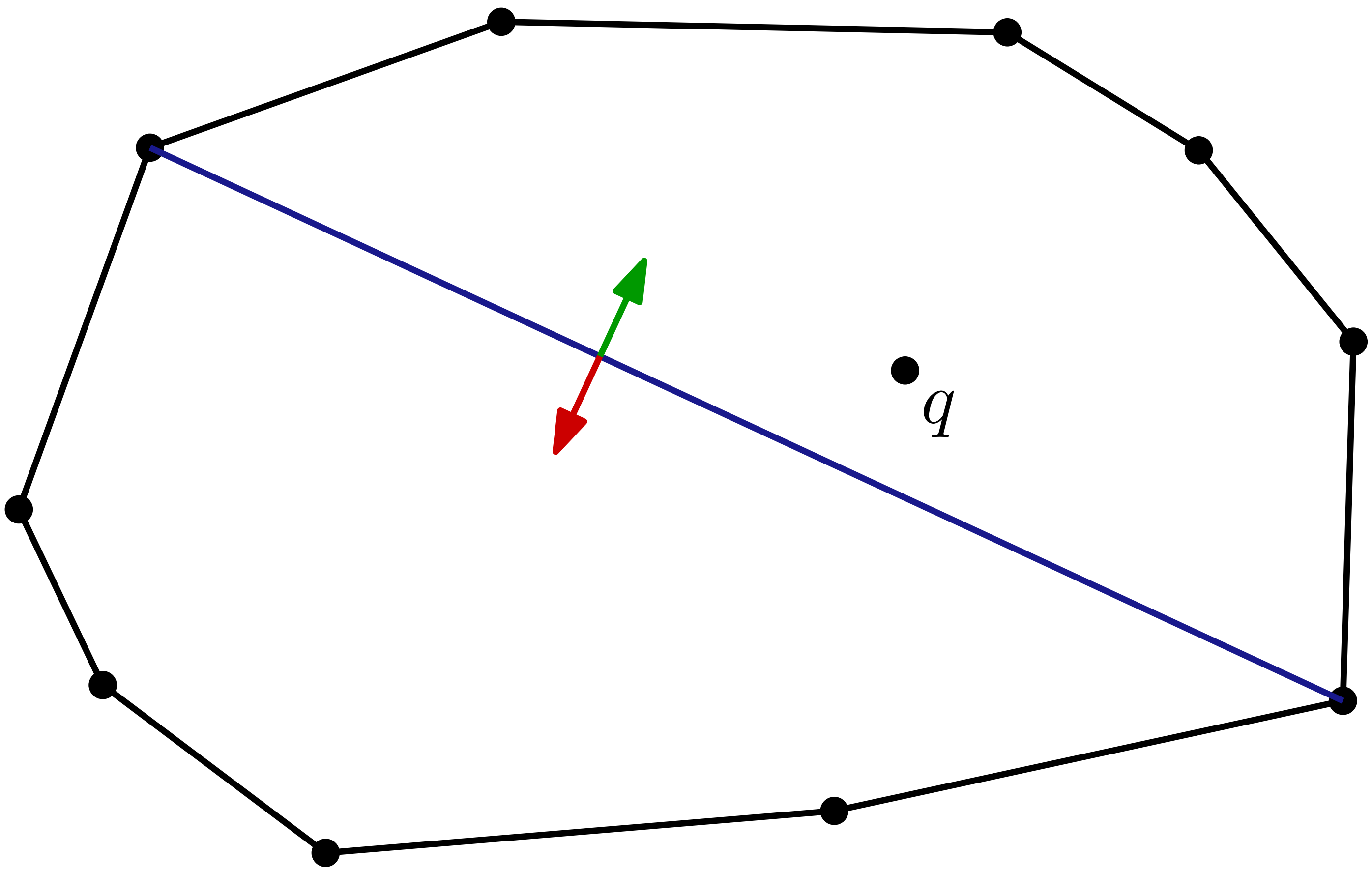
a) Is q contained in P? $\mathcal{O}(\log n)$



Exercise 2

Given:
- Point q
- convexes polygon P of n points.

a) Is q contained in P? $\mathcal{O}(\log n)$

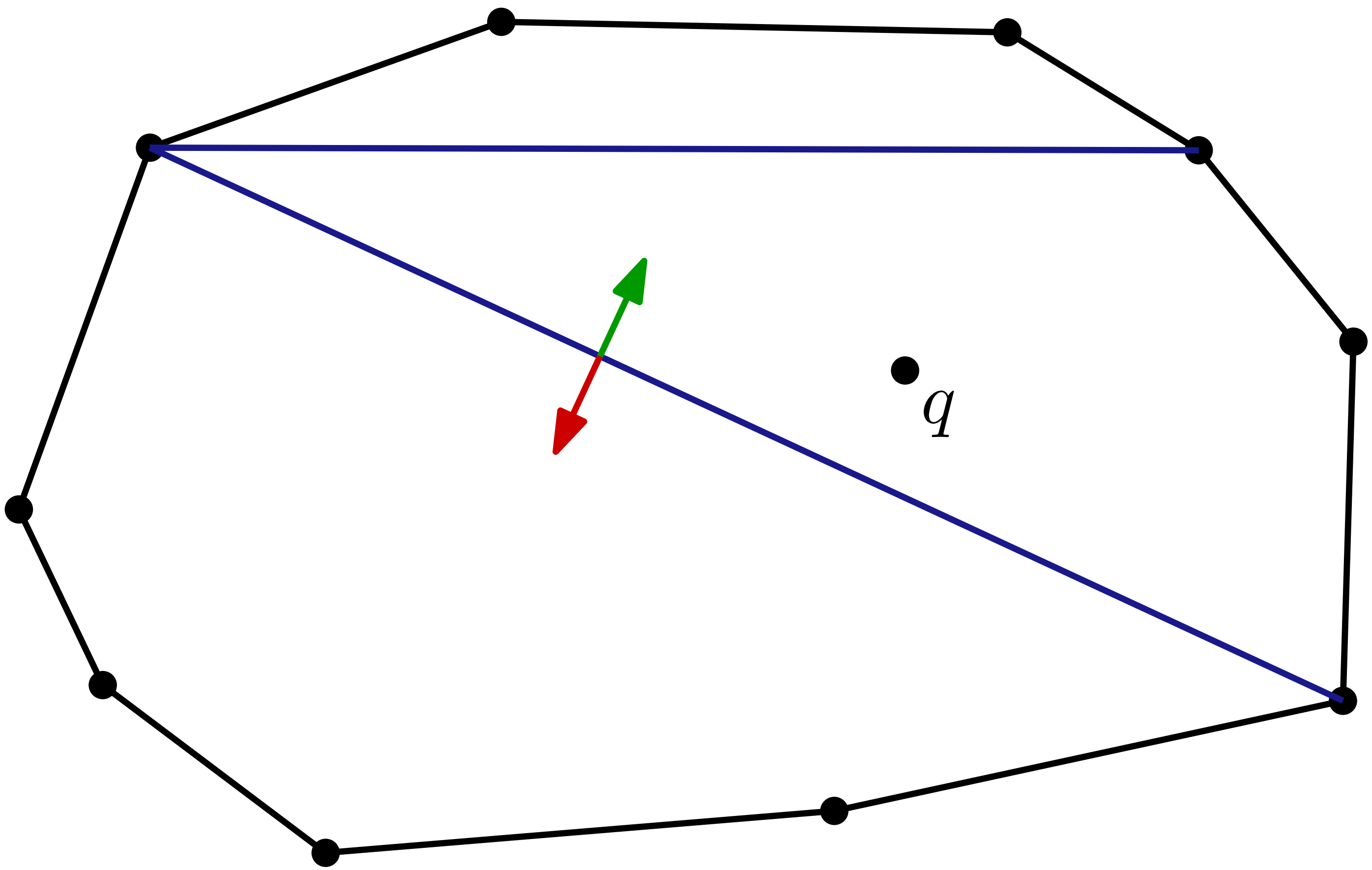


Exercise 2

Given:

- Point q
- convexes polygon P of n points.

a) Is q contained in P? $\mathcal{O}(\log n)$

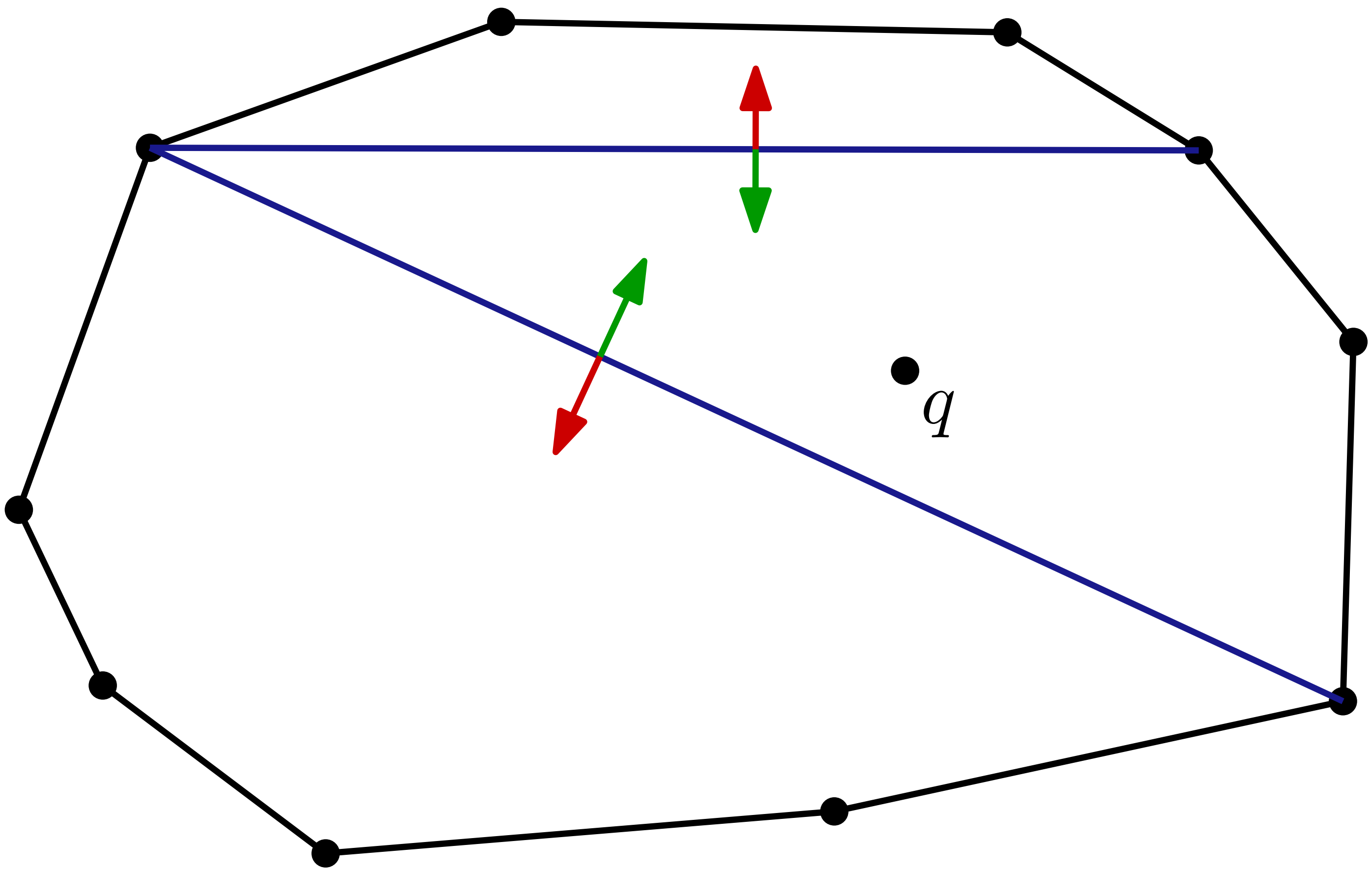


Exercise 2

Given:

- Point q
- convexes polygon P of n points.

a) Is q contained in P? $\mathcal{O}(\log n)$

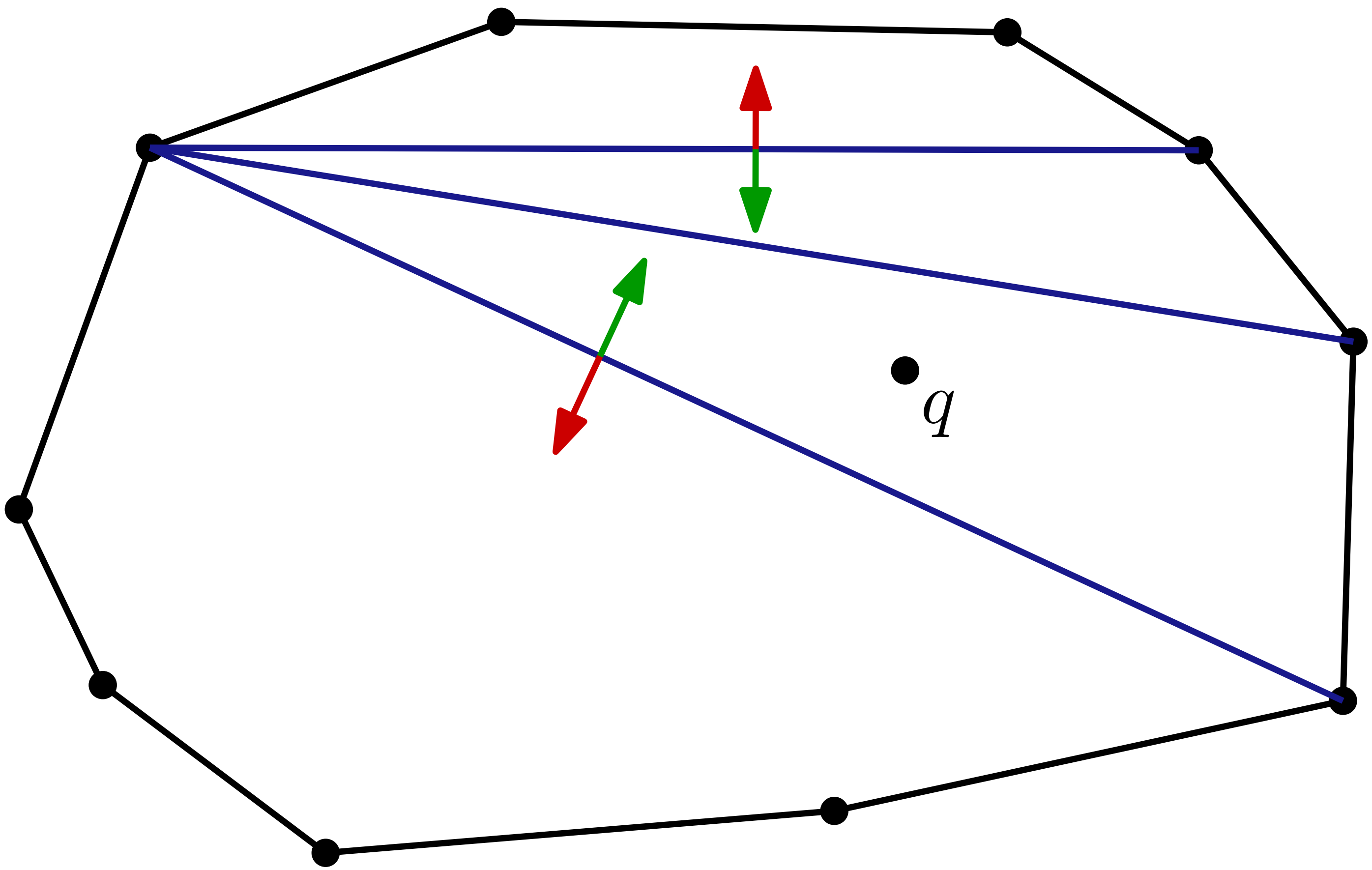


Exercise 2

Given:

- Point q
- convexes polygon P of n points.

a) Is q contained in P? $\mathcal{O}(\log n)$

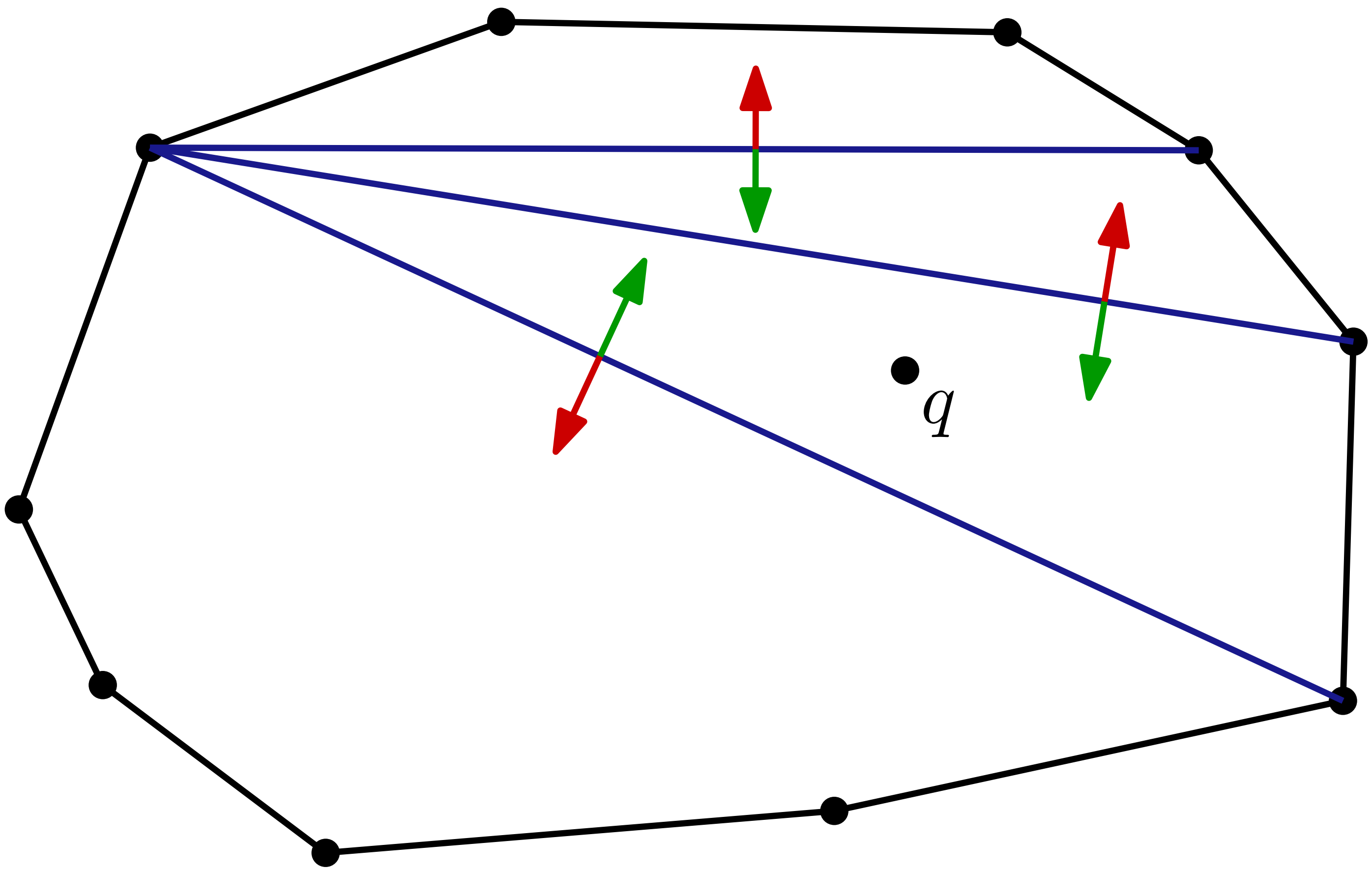


Exercise 2

Given:

- Point q
- convexes polygon P of n points.

a) Is q contained in P? $\mathcal{O}(\log n)$

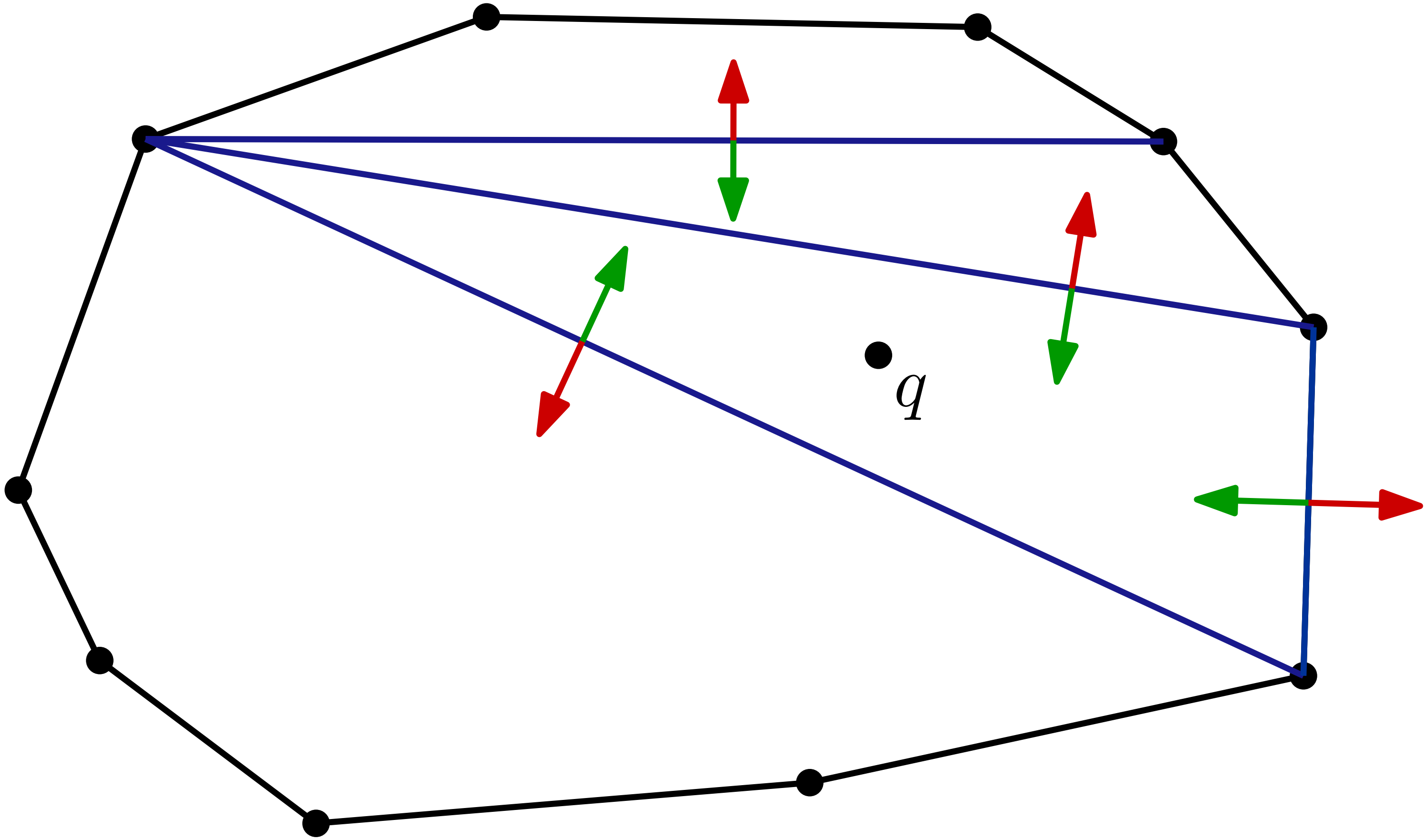


Exercise 2

Given:

- Point q
- convexes polygon P of n points.

a) Is q contained in P? $\mathcal{O}(\log n)$

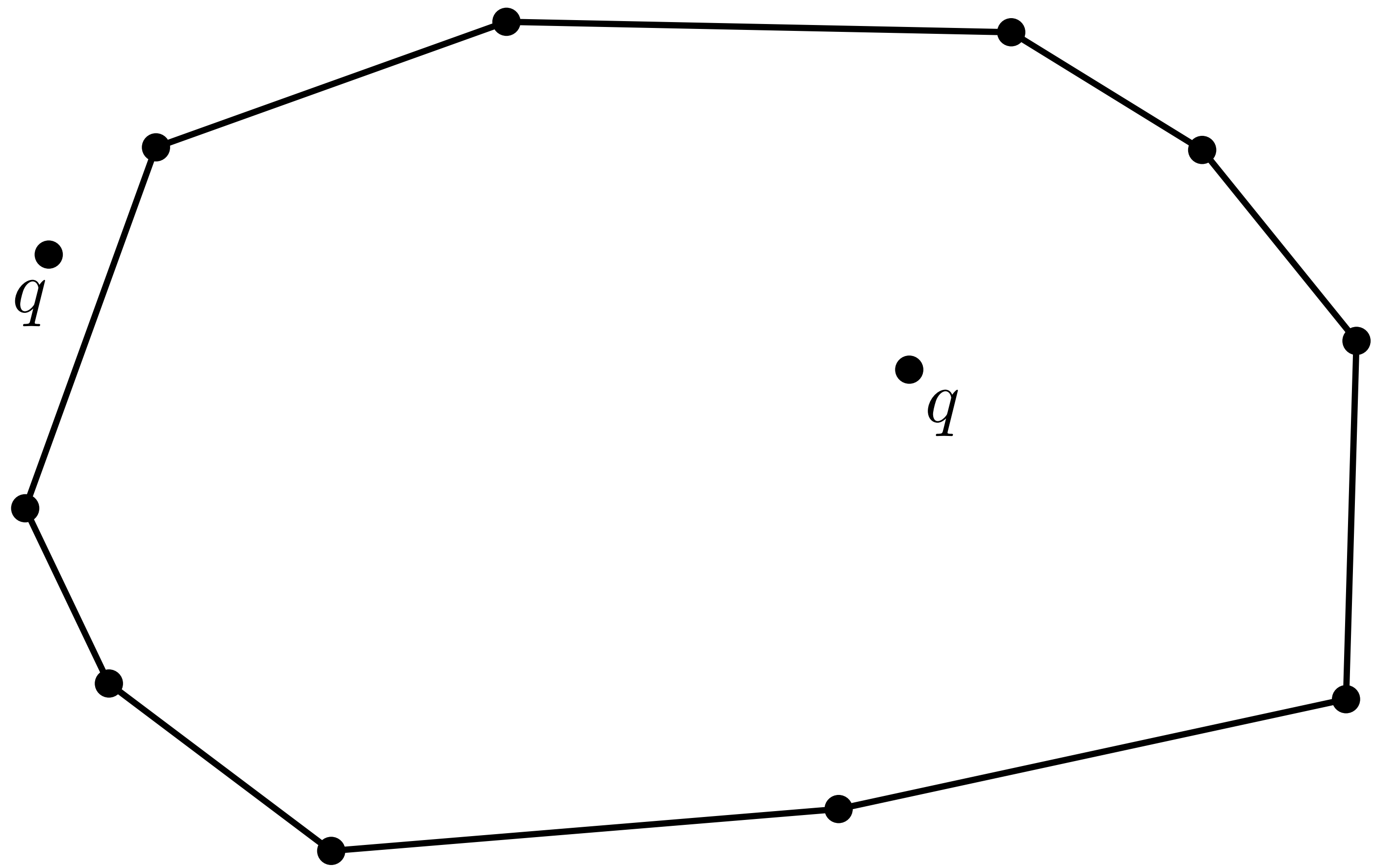


Exercise 2

Given:

- Point q
- convexes polygon P of n points.

a) Is q contained in P? $\mathcal{O}(\log n)$

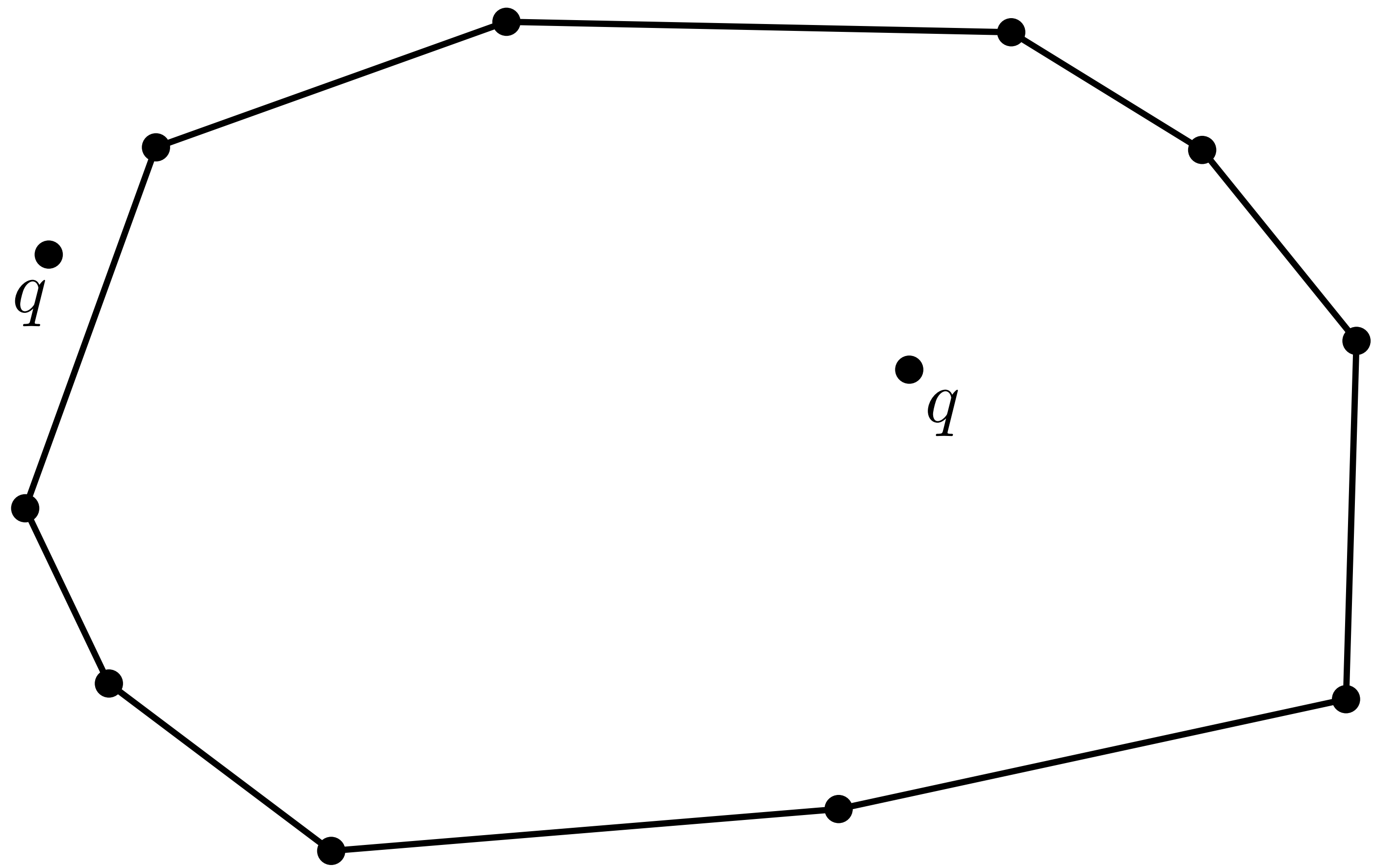


Exercise 2

Given:

- Point q
- convexes polygon P of n points.

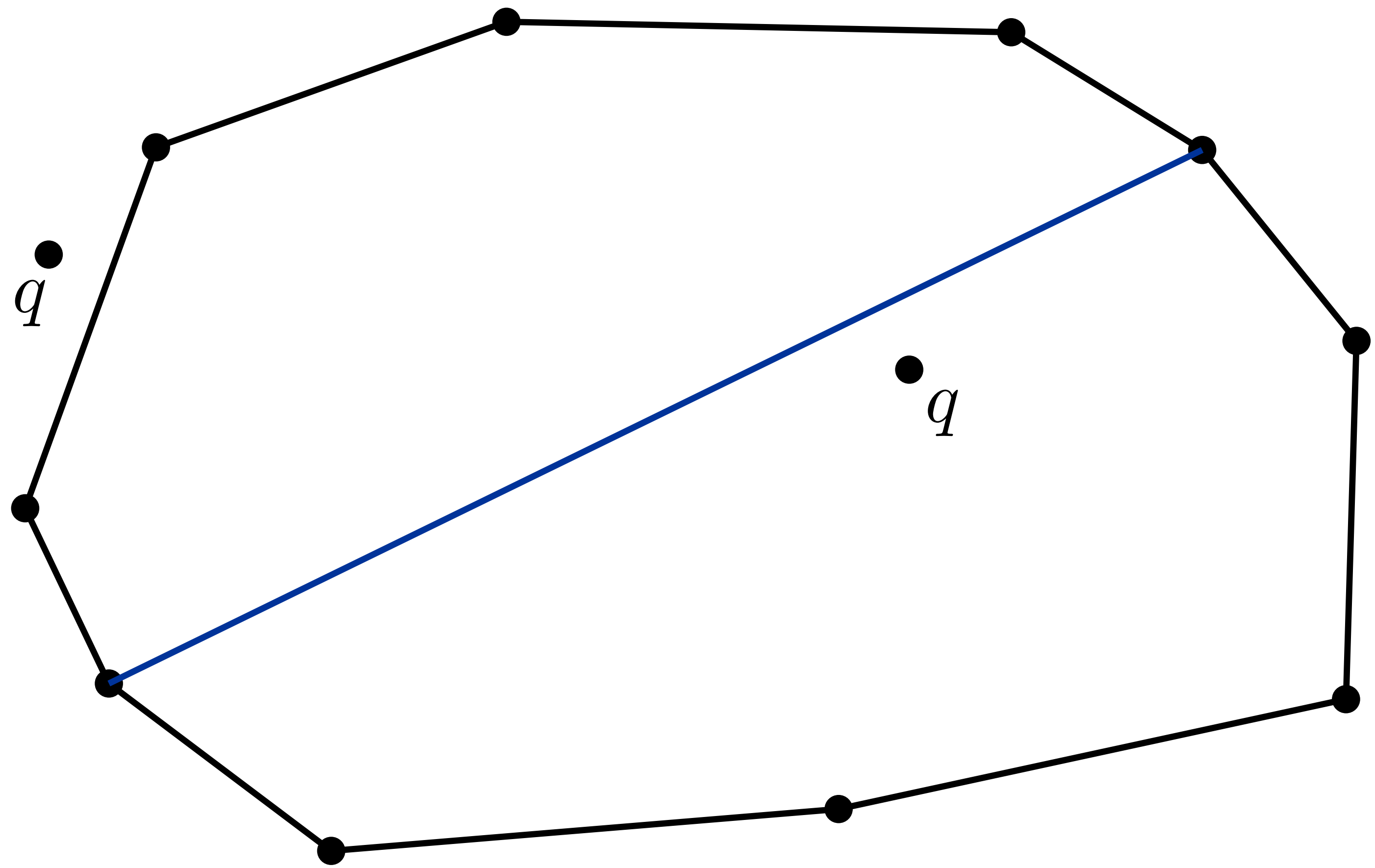
a) Is q contained in P? $\mathcal{O}(\log n)$



Exercise 2

Given:
- Point q
- convexes polygon P of n points.

a) Is q contained in P? $\mathcal{O}(\log n)$

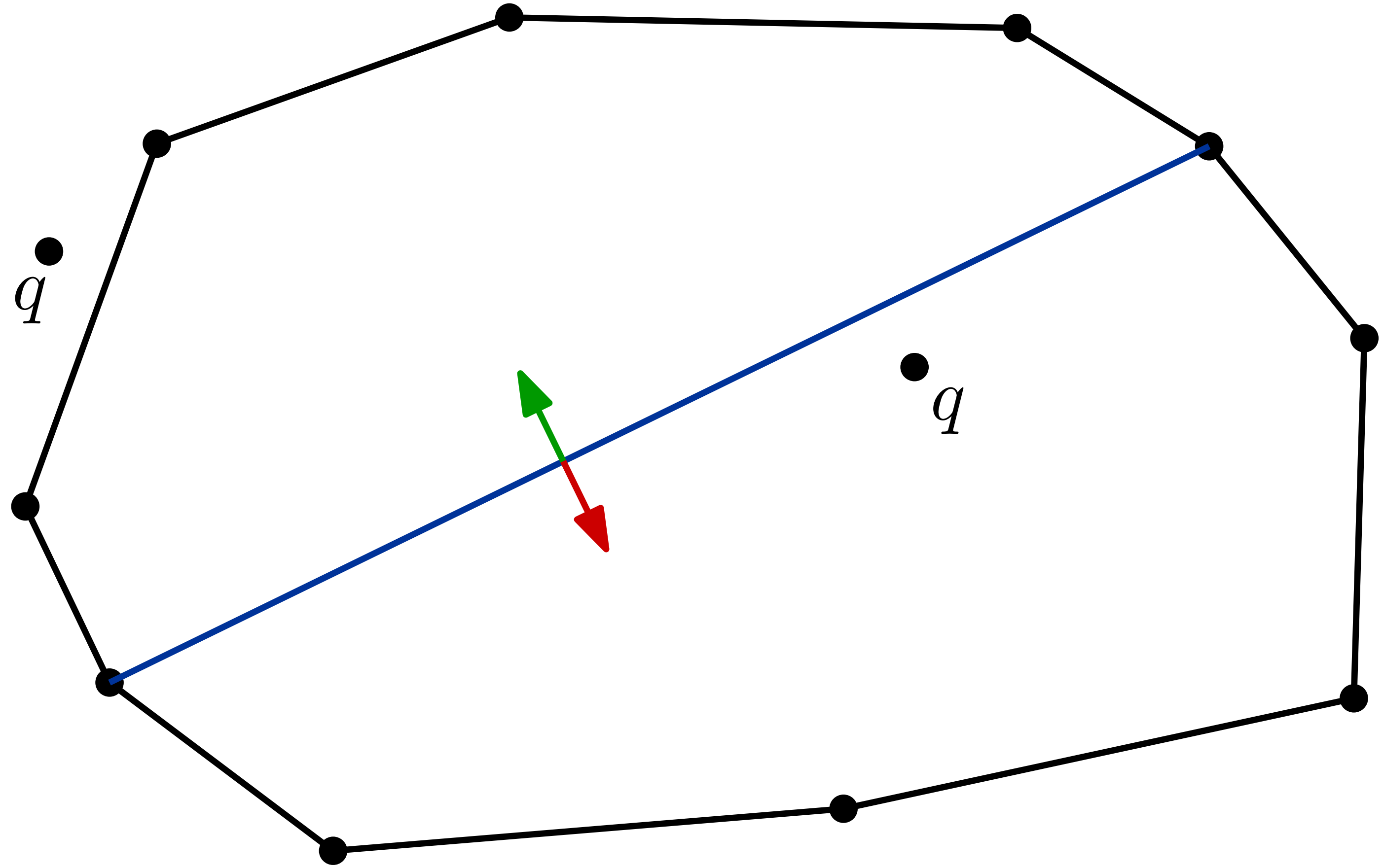


Exercise 2

Given:

- Point q
- convexes polygon P of n points.

a) Is q contained in P? $\mathcal{O}(\log n)$

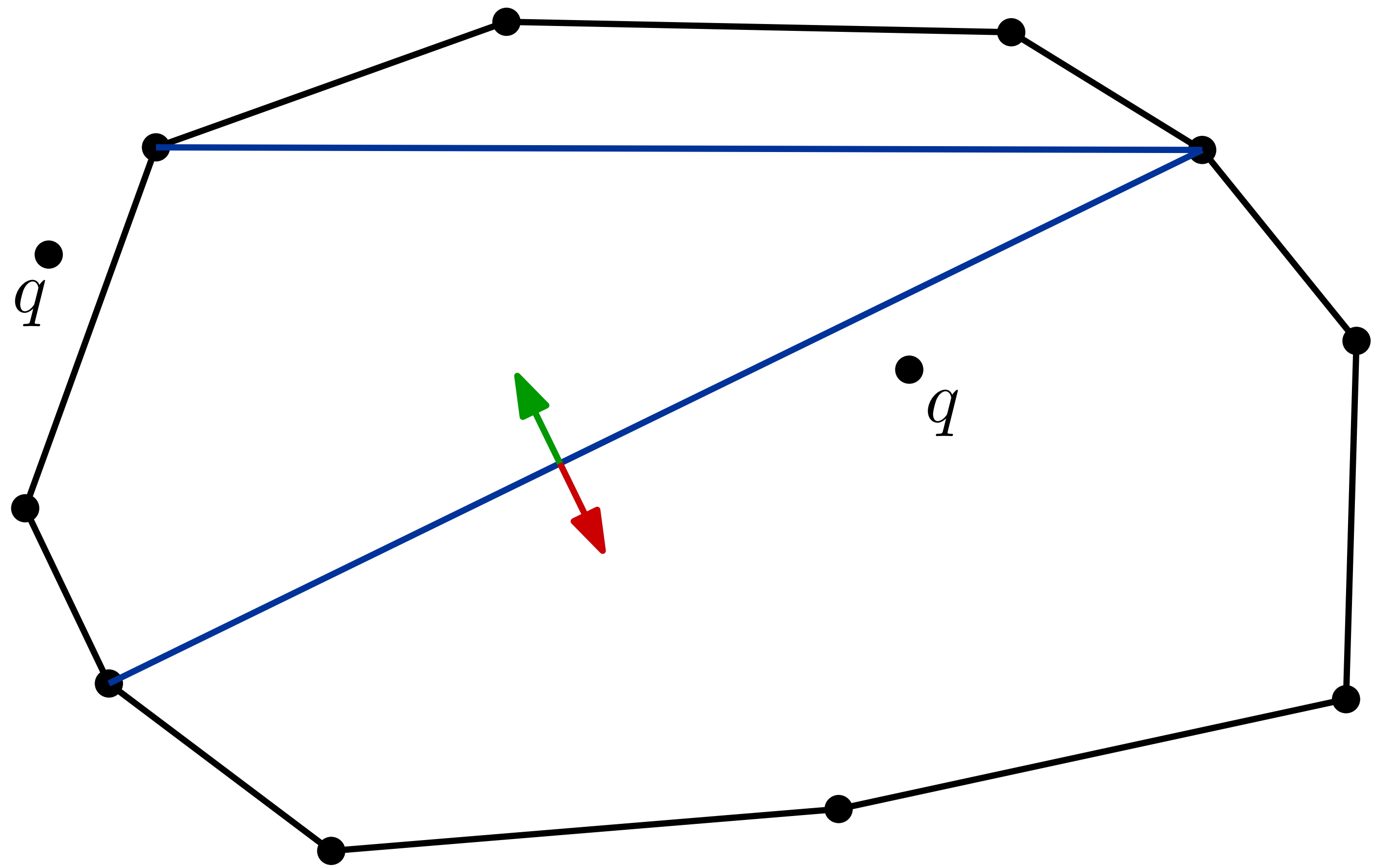


Exercise 2

Given:

- Point q
- convexes polygon P of n points.

a) Is q contained in P? $\mathcal{O}(\log n)$

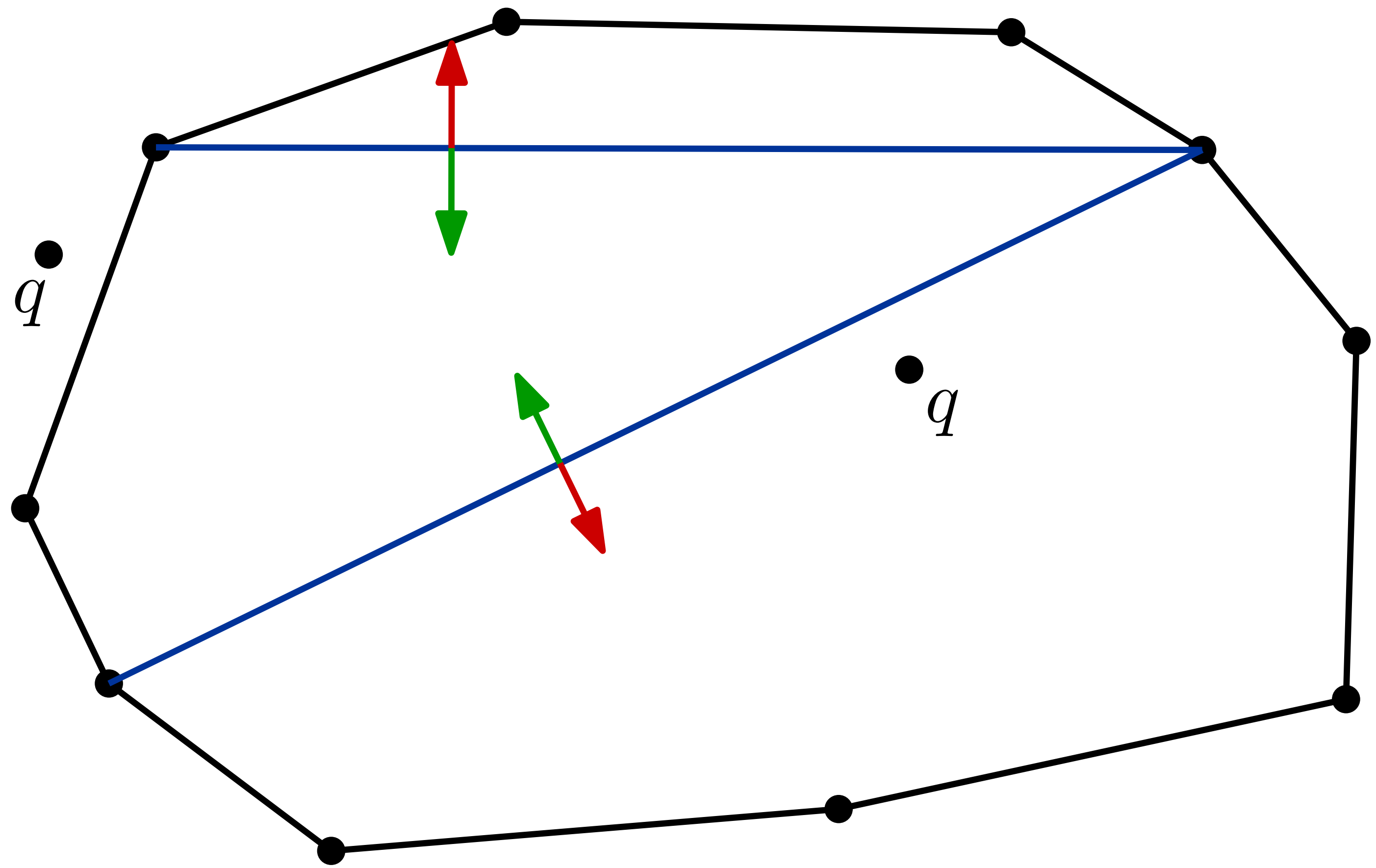


Exercise 2

Given:

- Point q
- convexes polygon P of n points.

a) Is q contained in P? $\mathcal{O}(\log n)$

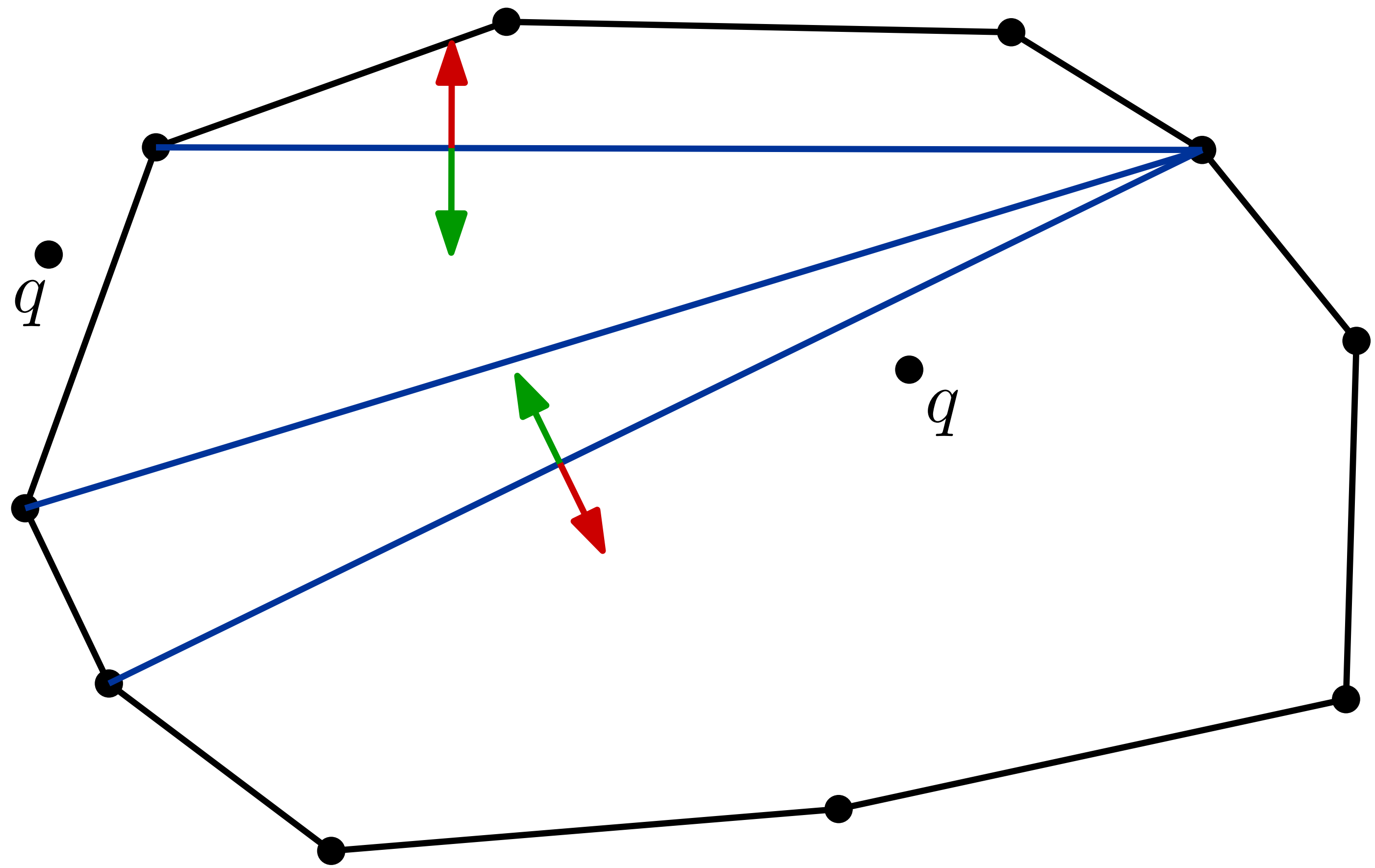


Exercise 2

Given:

- Point q
- convexes polygon P of n points.

a) Is q contained in P? $\mathcal{O}(\log n)$

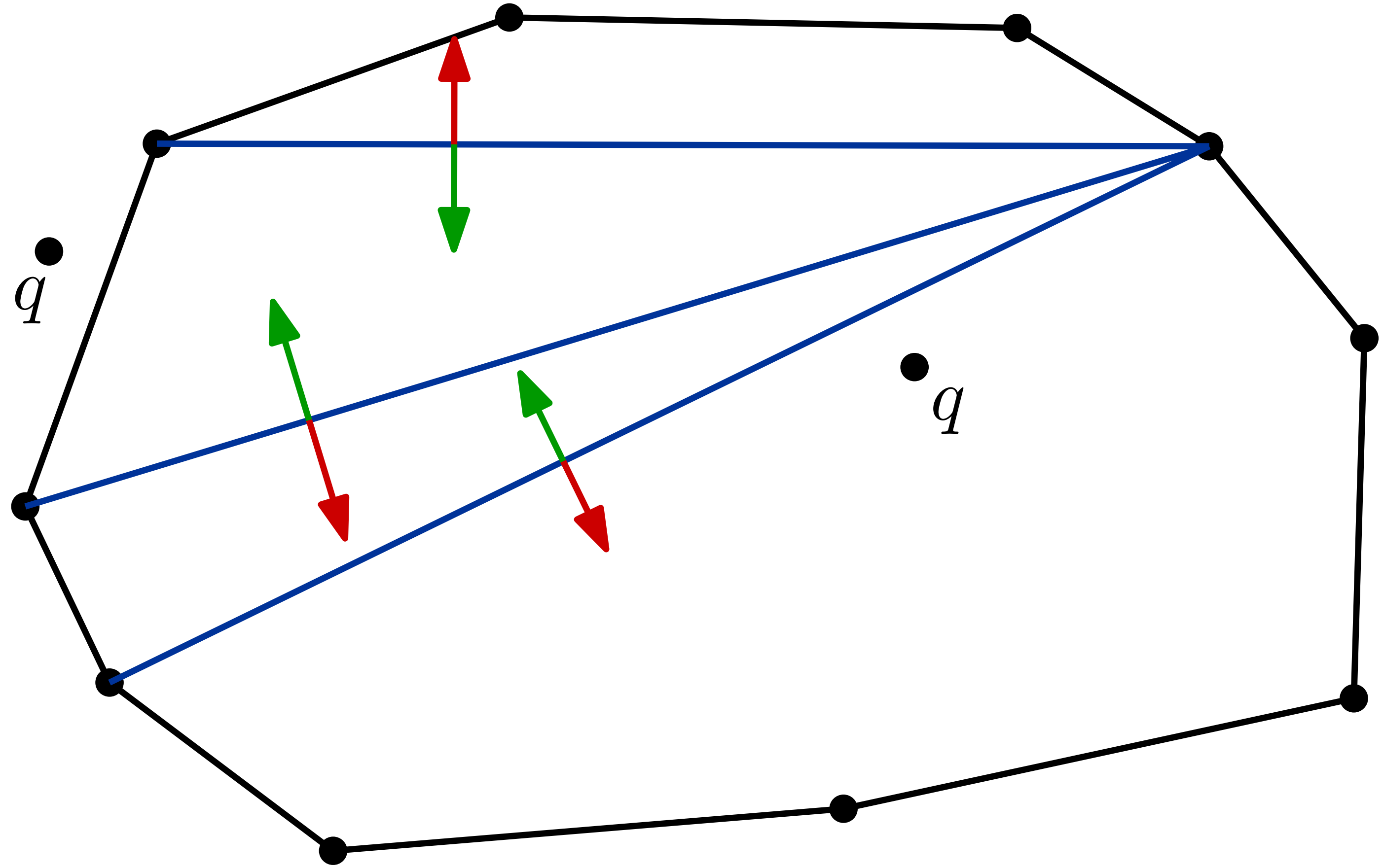


Exercise 2

Given:

- Point q
- convexes polygon P of n points.

a) Is q contained in P? $\mathcal{O}(\log n)$

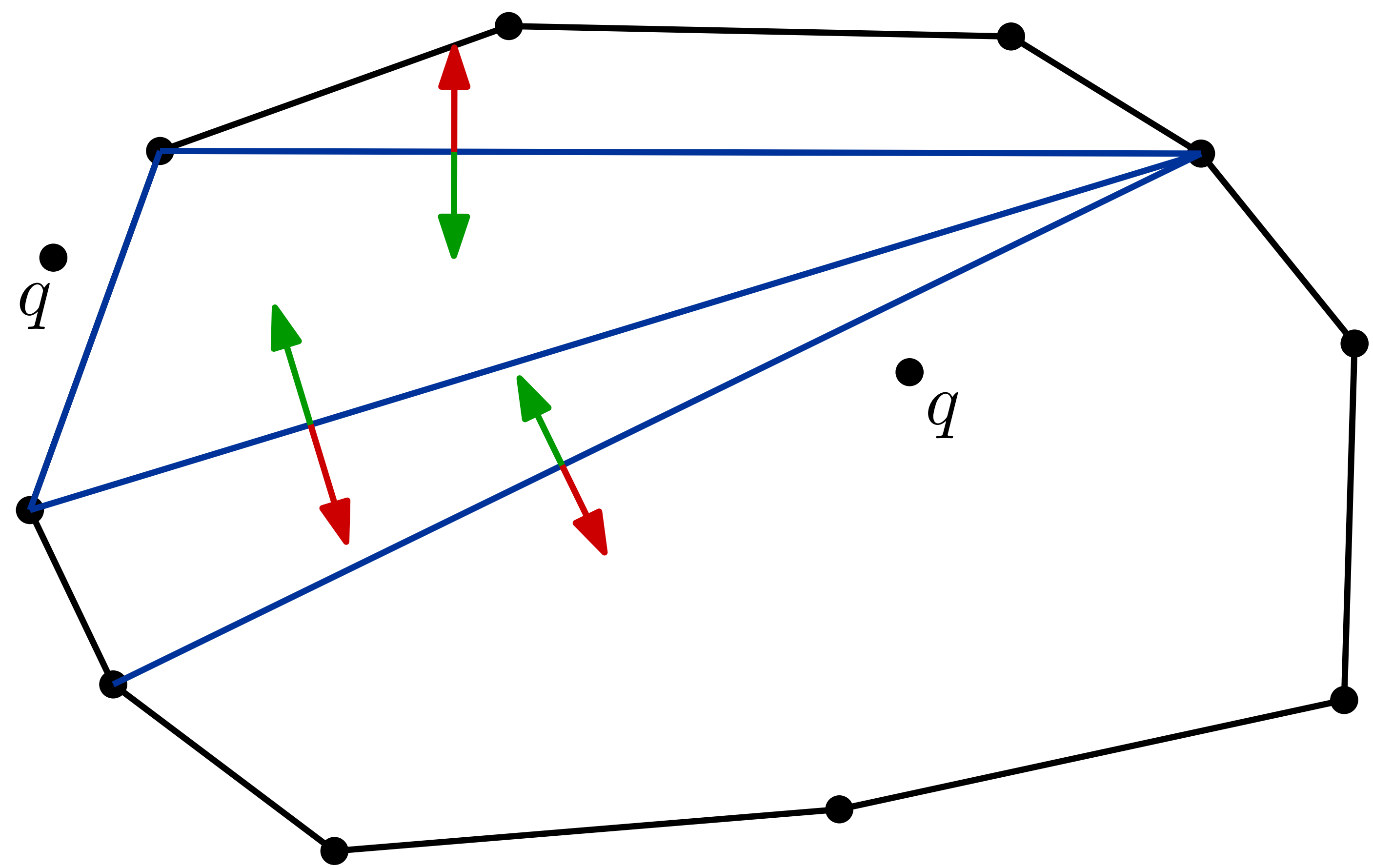


Exercise 2

Given:

- Point q
- convexes polygon P of n points.

a) Is q contained in P? $\mathcal{O}(\log n)$

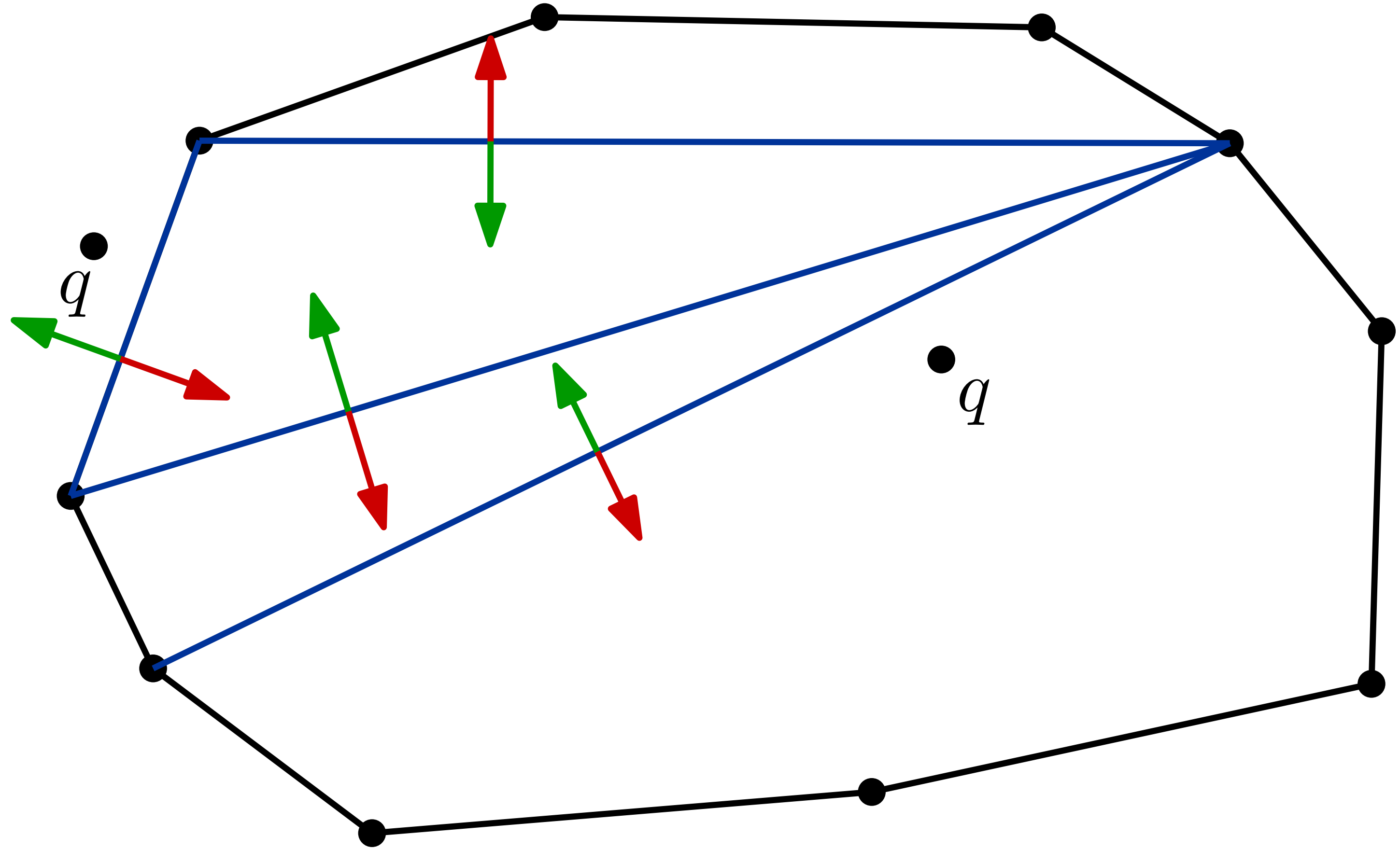


Exercise 2

Given:

- Point q
- convexes polygon P of n points.

a) Is q contained in P? $\mathcal{O}(\log n)$



Exercise 2

Given:

- Point q
- y-monotone polygon P of n points.

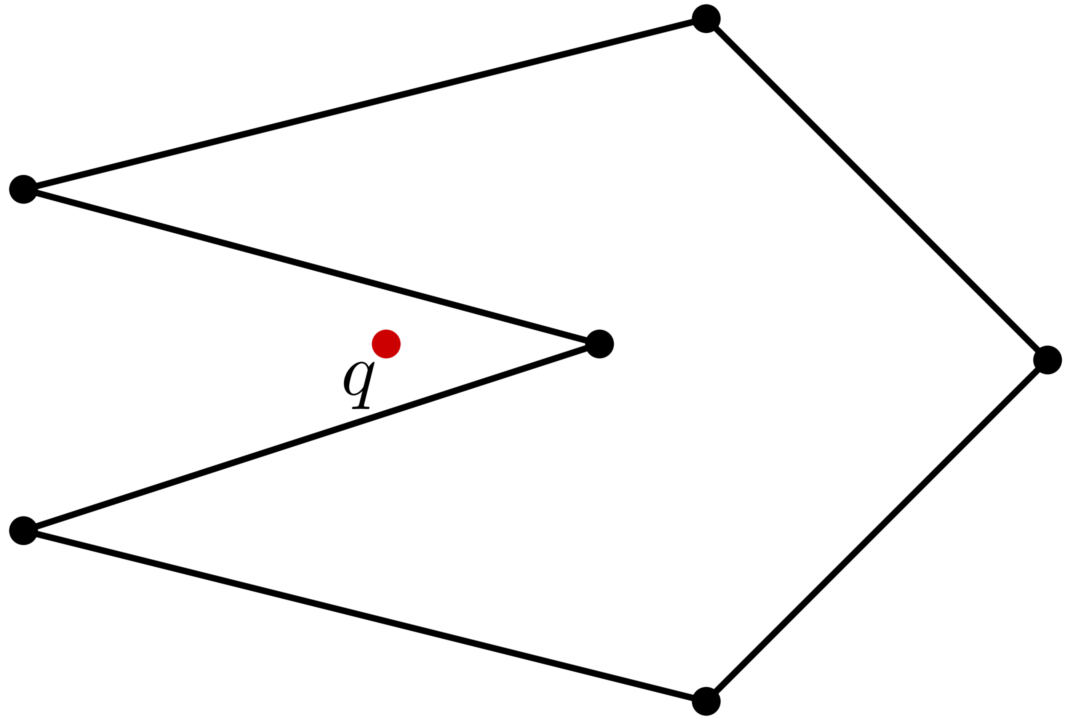
b) Can the procedure be adapted?

Exercise 2

Given:

- Point q
- y-monotone polygon P of n points.

b) Can the procedure be adapted?

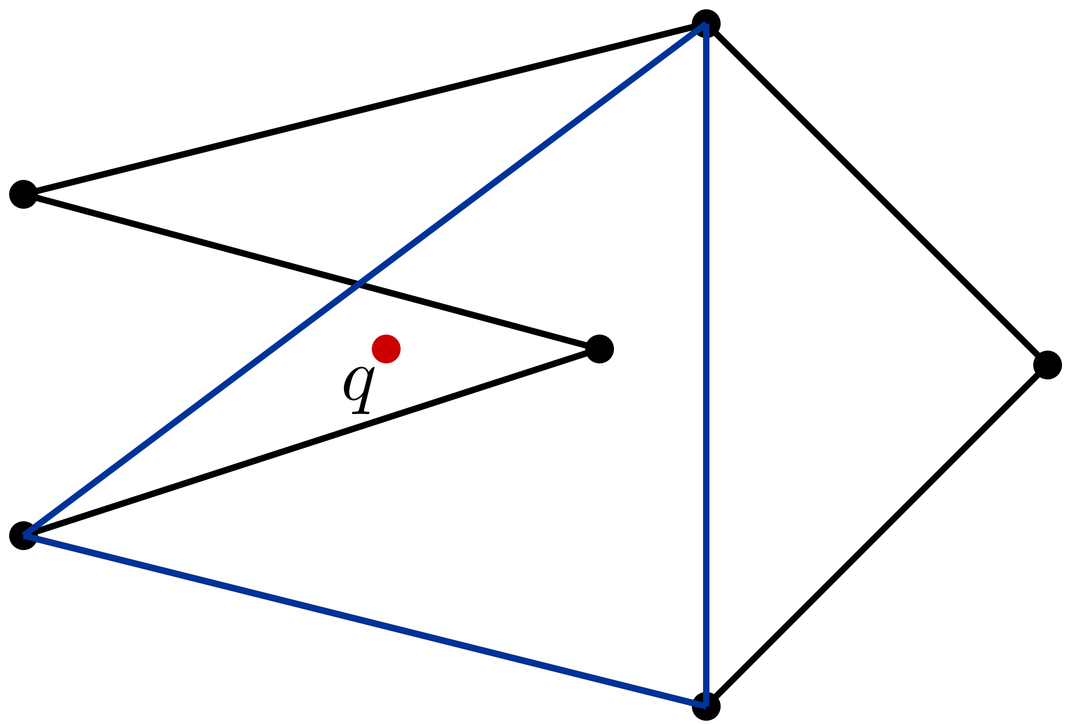


Exercise 2

Given:

- Point q
- y-monotone polygon P of n points.

b) Can the procedure be adapted?



Exercise 3

Given:

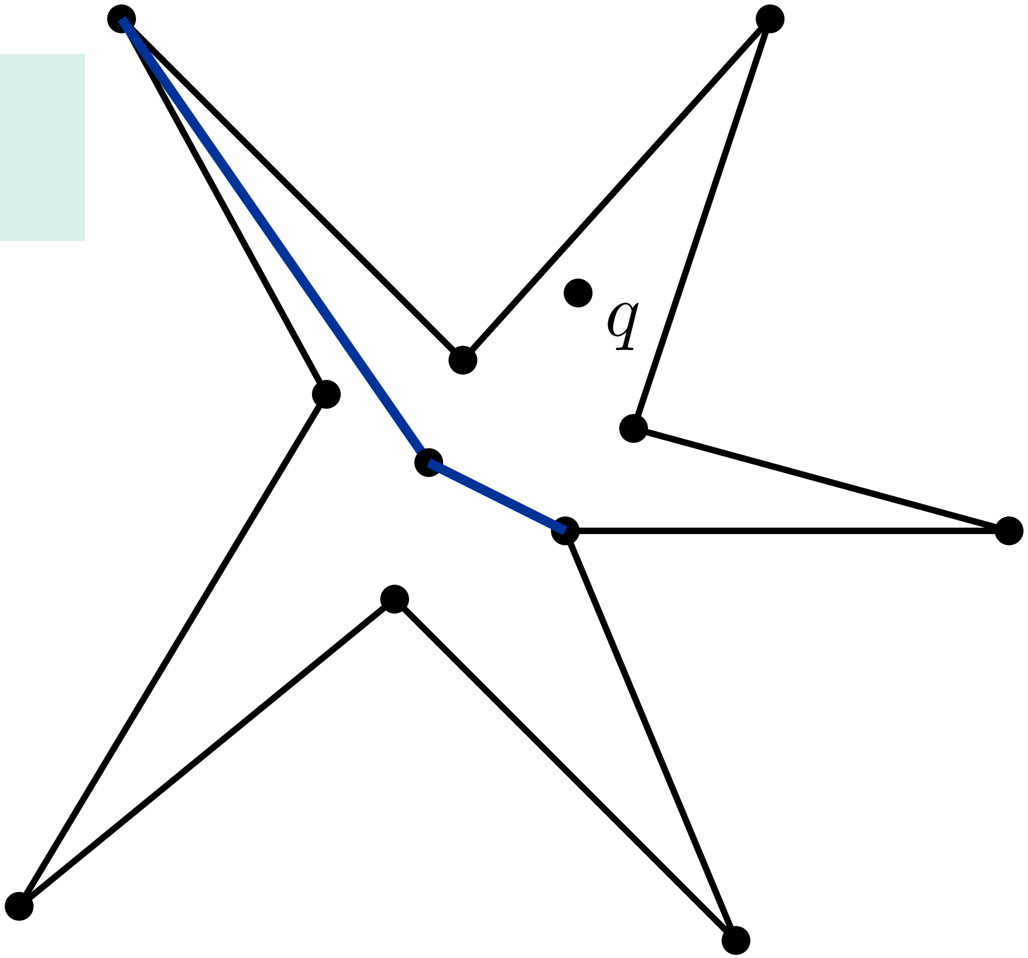
- Point q
- star-shaped polygon P consisting of n points.

a) q is contained in P in $\mathcal{O}(\log n)$?

P is *star-shaped*, if

$$\exists p \in P \text{ s.t. } \forall q \in P\colon \overline{pq} \in P$$

Assumption: p is given.



Exercise 3

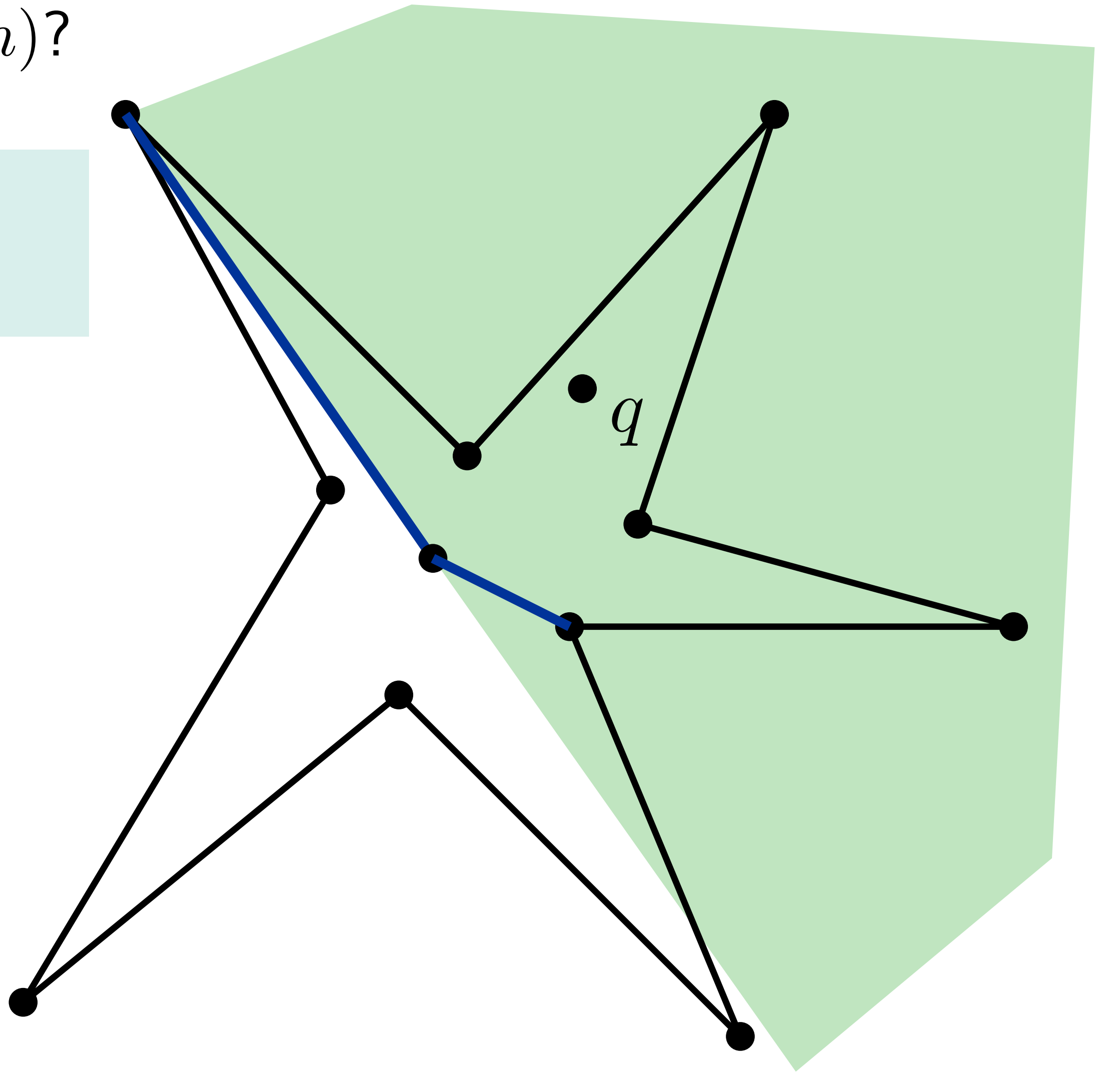
Given:
- Point q
- star-shaped polygon P consisting of n points.

a) q is contained in P in $\mathcal{O}(\log n)$?

P is *star-shaped*, if

$$\exists p \in P \text{ s.t. } \forall q \in P\colon \overline{pq} \in P$$

Assumption: p is given.



Exercise 3

Given:

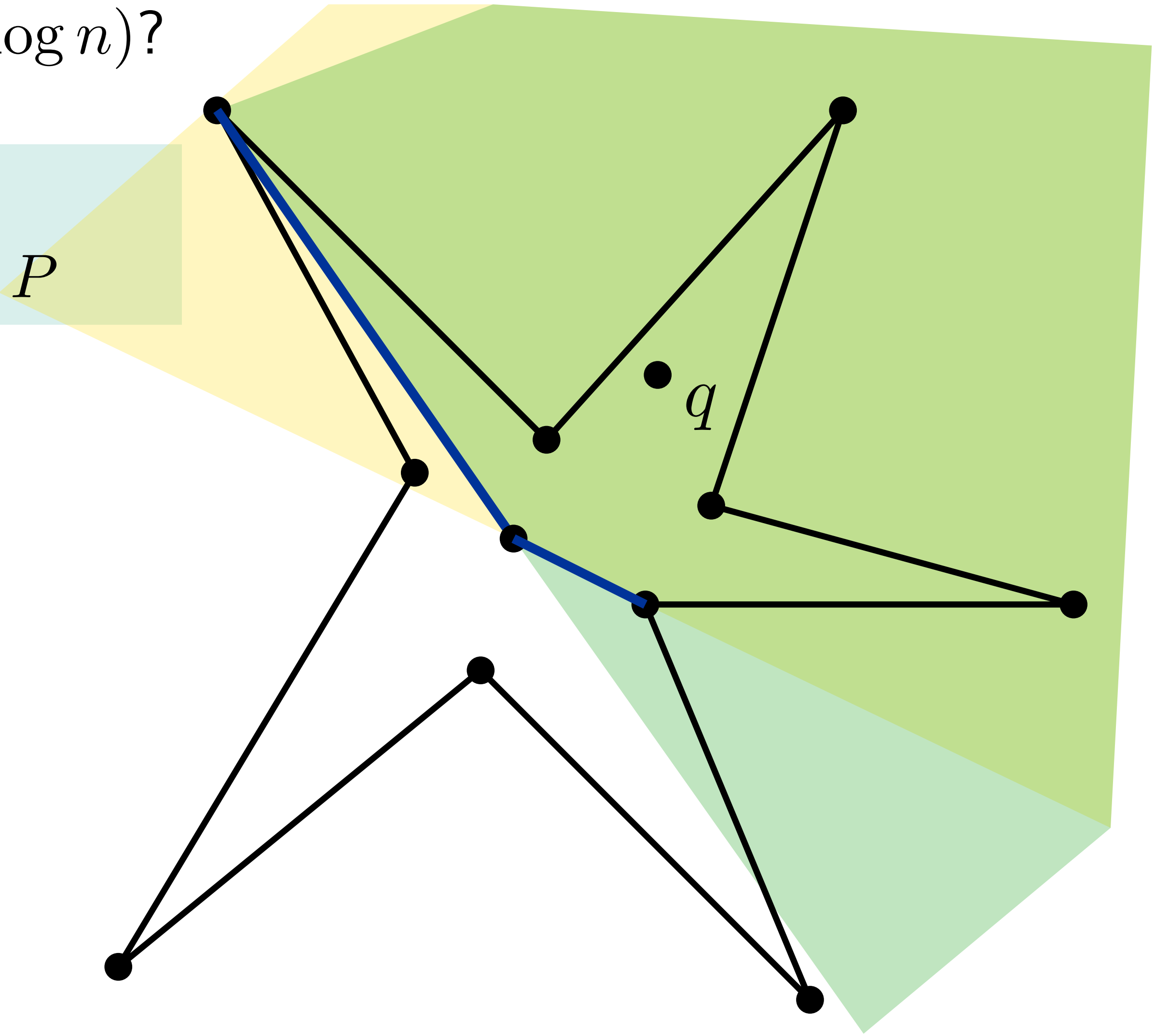
- Point q
- star-shaped polygon P consisting of n points.

a) q is contained in P in $\mathcal{O}(\log n)$?

P is *star-shaped*, if

$\exists p \in P$ s.t. $\forall q \in P$: $\overline{pq} \in P$

Assumption: p is given.



Exercise 3

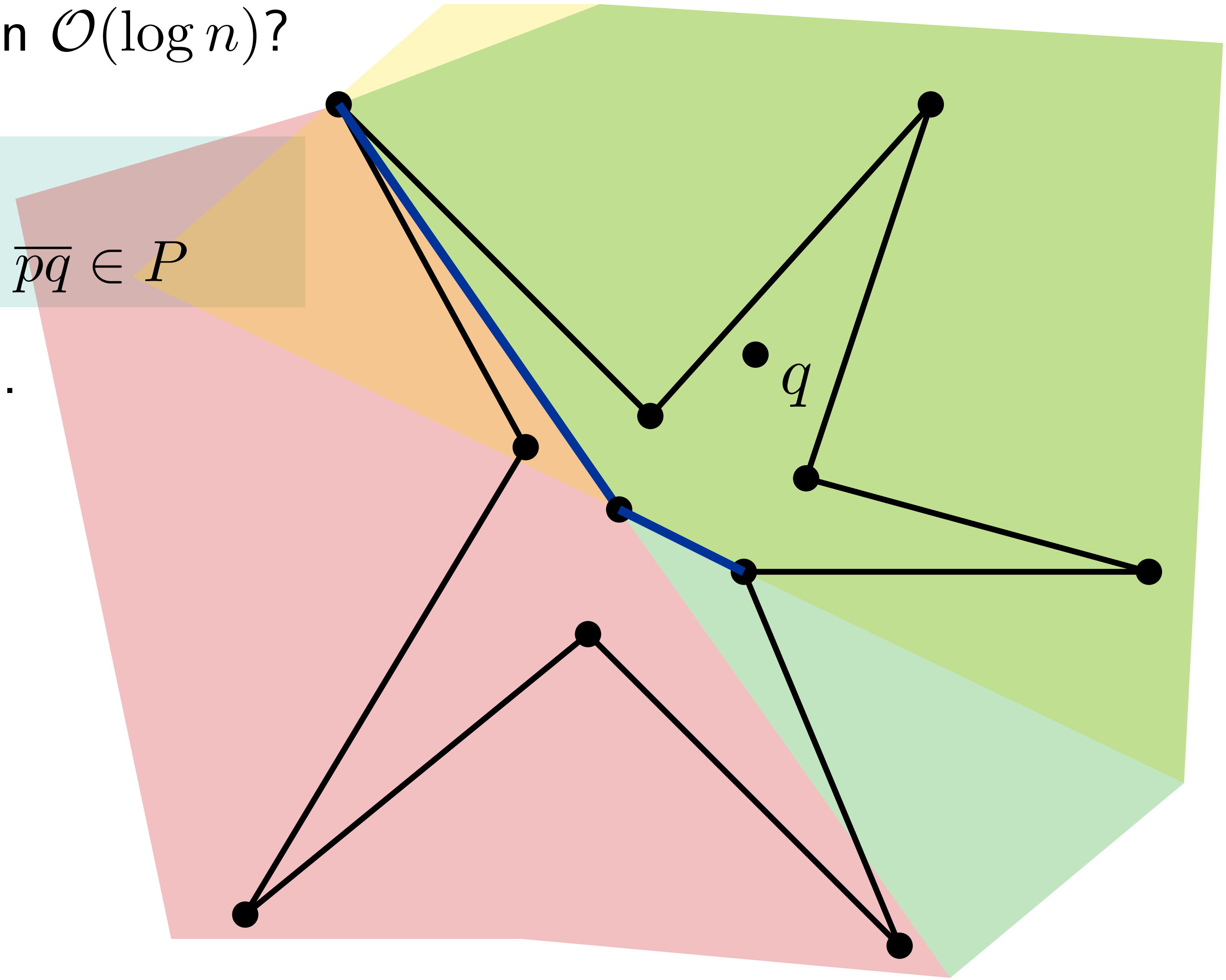
Given:
- Point q
- star-shaped polygon P consisting of n points.

a) q is contained in P in $\mathcal{O}(\log n)$?

P is *star-shaped*, if

$\exists p \in P$ s.t. $\forall q \in P$: $\overline{pq} \in P$

Assumption: p is given.



Exercise 3

Given:

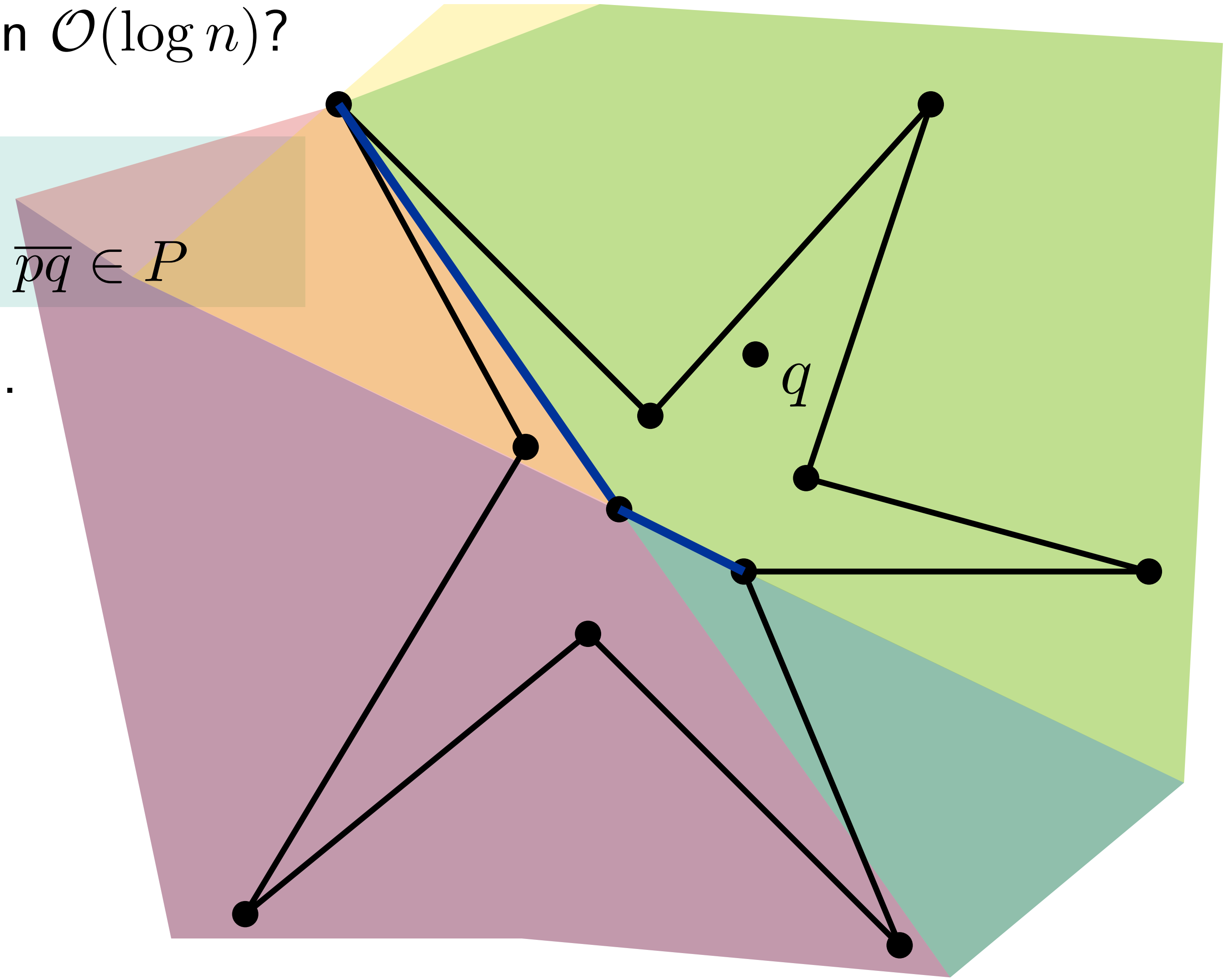
- Point q
- star-shaped polygon P consisting of n points.

a) q is contained in P in $\mathcal{O}(\log n)$?

P is *star-shaped*, if

$\exists p \in P$ s.t. $\forall q \in P$: $\overline{pq} \in P$

Assumption: p is given.



Exercise 3

Given:

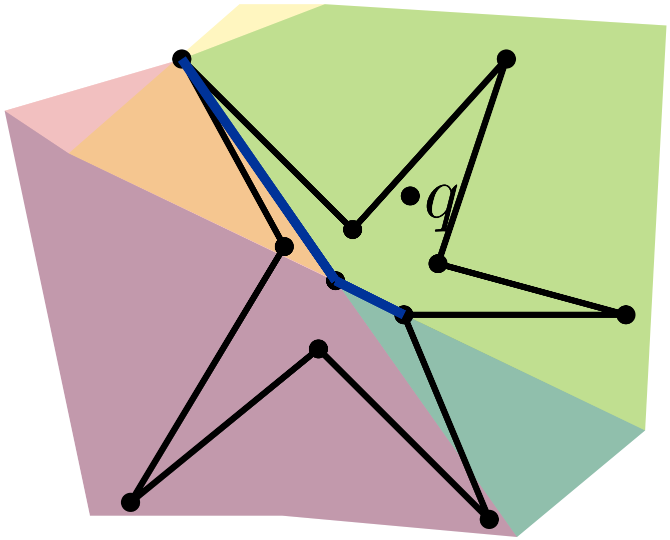
- Point q
- star-shaped polygon P consisting of n points.

a) q is contained in P in $\mathcal{O}(\log n)$?

P is *star-shaped*, if

$$\exists p \in P \text{ s.t. } \forall q \in P\colon \overline{pq} \in P$$

Assumption: p is given.



Exercise 3

Given:

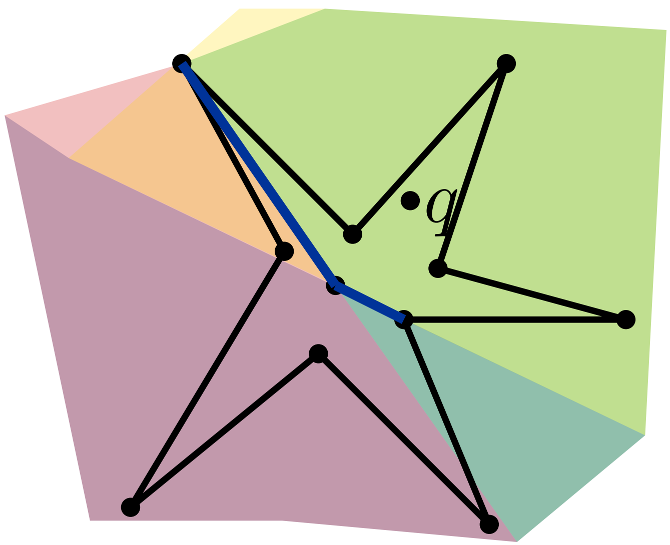
- Point q
- star-shaped polygon P consisting of n points.

a) q is contained in P in $\mathcal{O}(\log n)$?

P is *star-shaped*, if

$$\exists p \in P \text{ s.t. } \forall q \in P\colon \overline{pq} \in P$$

Assumption: p is given.



Exercise 3

Given:

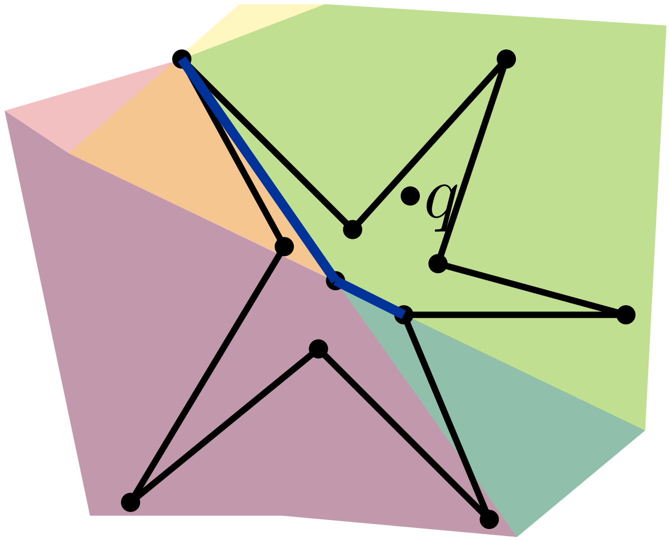
- Point q
- star-shaped polygon P consisting of n points.

b) What, if p is not known?

P is *star-shaped*, if

$$\exists p \in P \text{ s.t. } \forall q \in P\colon \overline{pq} \in P$$

Assumption: p is given.

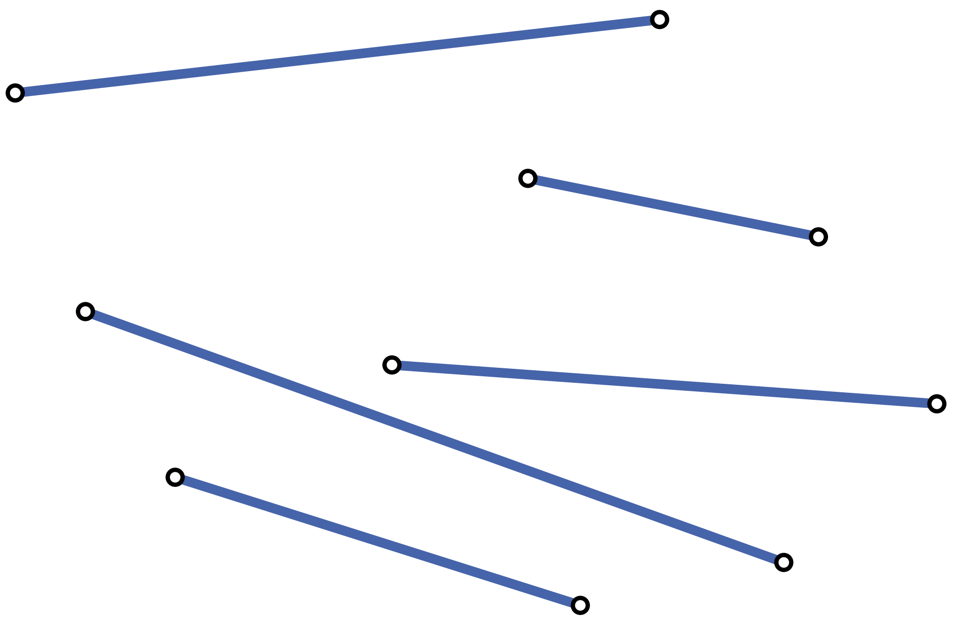


Exercise 4

Ray-Shooting Problem   Here: Simplified.

Point $q \in \mathbb{R}^2$ and n intersecting-free segments are given. Let ϱ be a vertical half-line that *shoots* upwards from q.

Find 'first' segment that intersects ϱ .

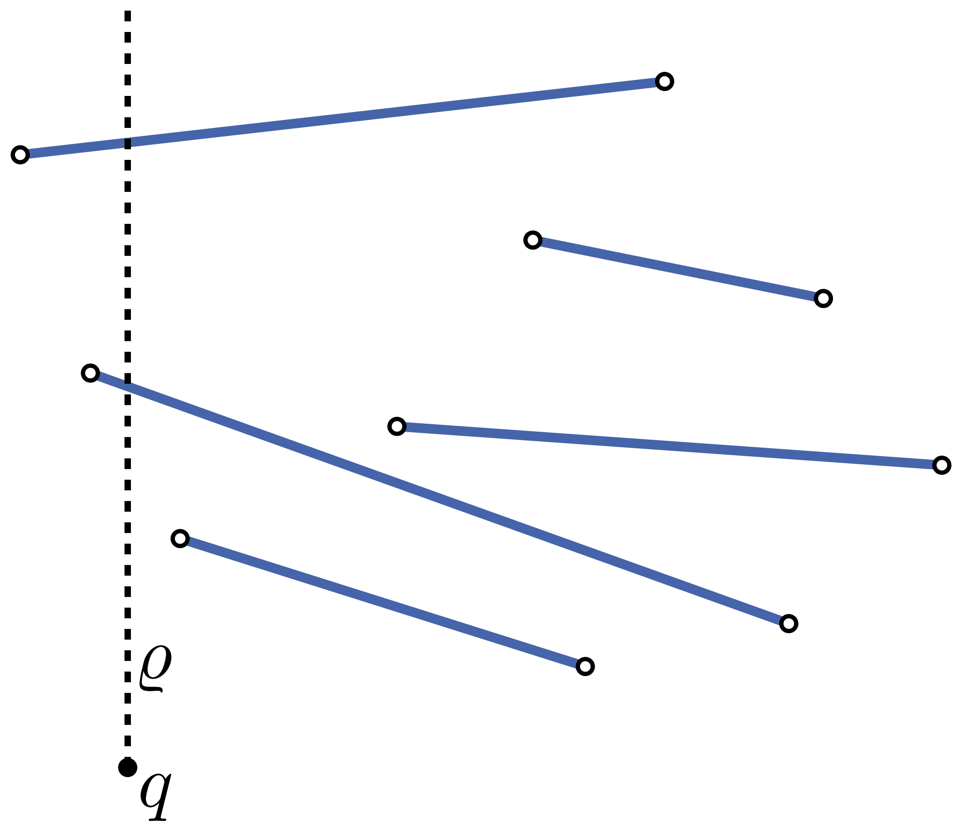


Exercise 4

Ray-Shooting Problem   Here: Simplified.

Point $q \in \mathbb{R}^2$ and n intersecting-free segments are given. Let ϱ be a vertical half-line that *shoots* upwards from q.

Find 'first' segment that intersects ϱ .



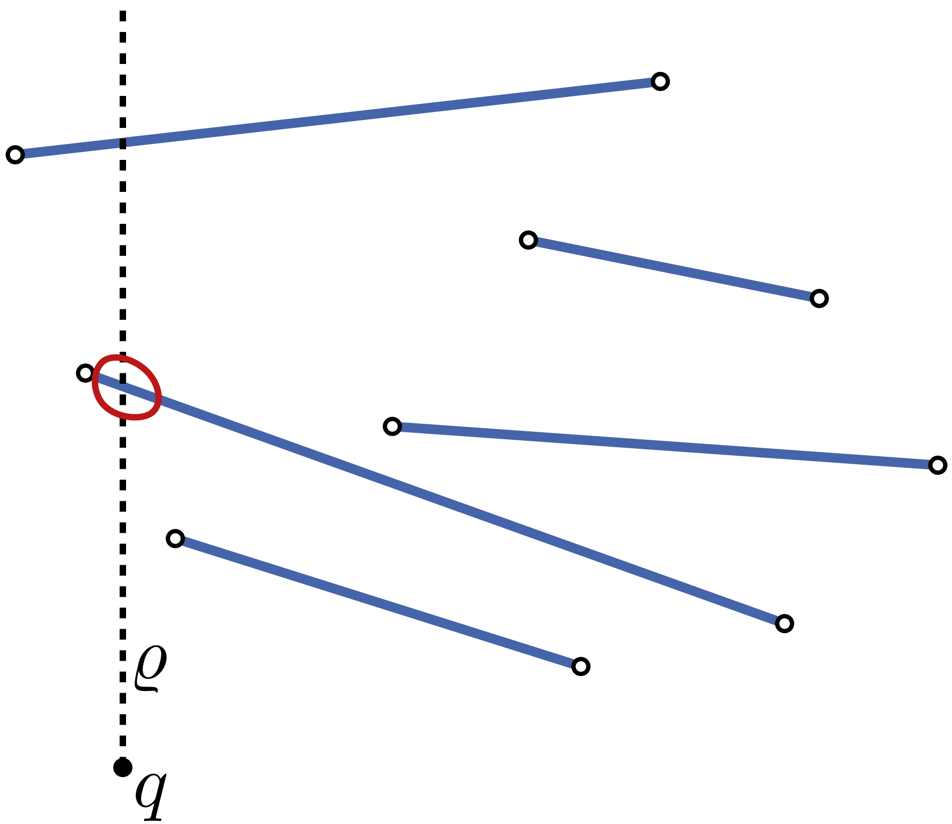
Exercise 4

Ray-Shooting Problem

Here: Simplified.

Point $q \in \mathbb{R}^2$ and n intersecting-free segments are given. Let ϱ be a vertical half-line that *shoots* upwards from q.

Find 'first' segment that intersects ϱ .



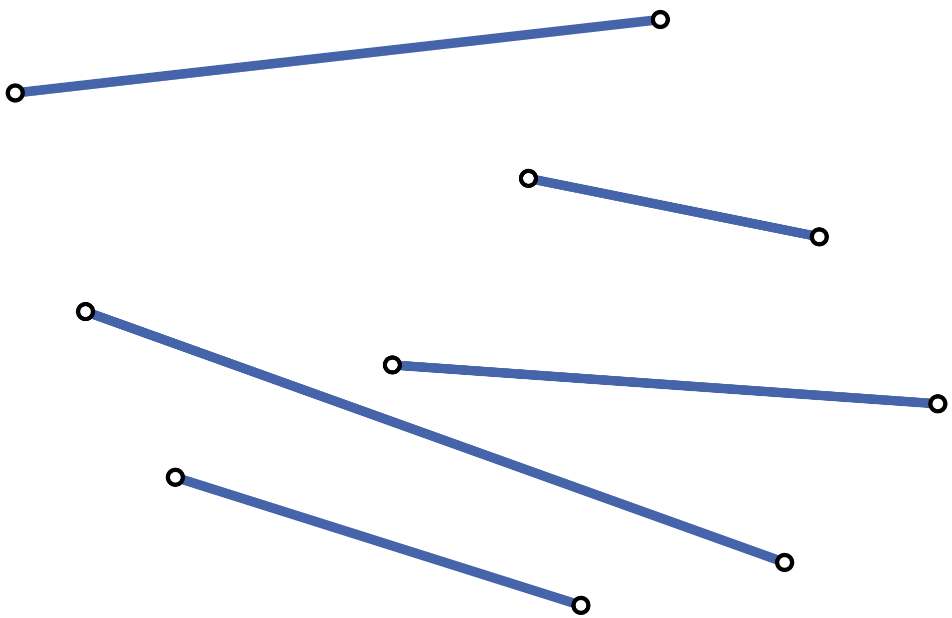
Exercise 4

Ray-Shooting Problem

Here: Simplified.

Point $q \in \mathbb{R}^2$ and n intersecting-free segments are given. Let ϱ be a vertical half-line that *shoots* upwards from q.

Find 'first' segment that intersects ϱ .



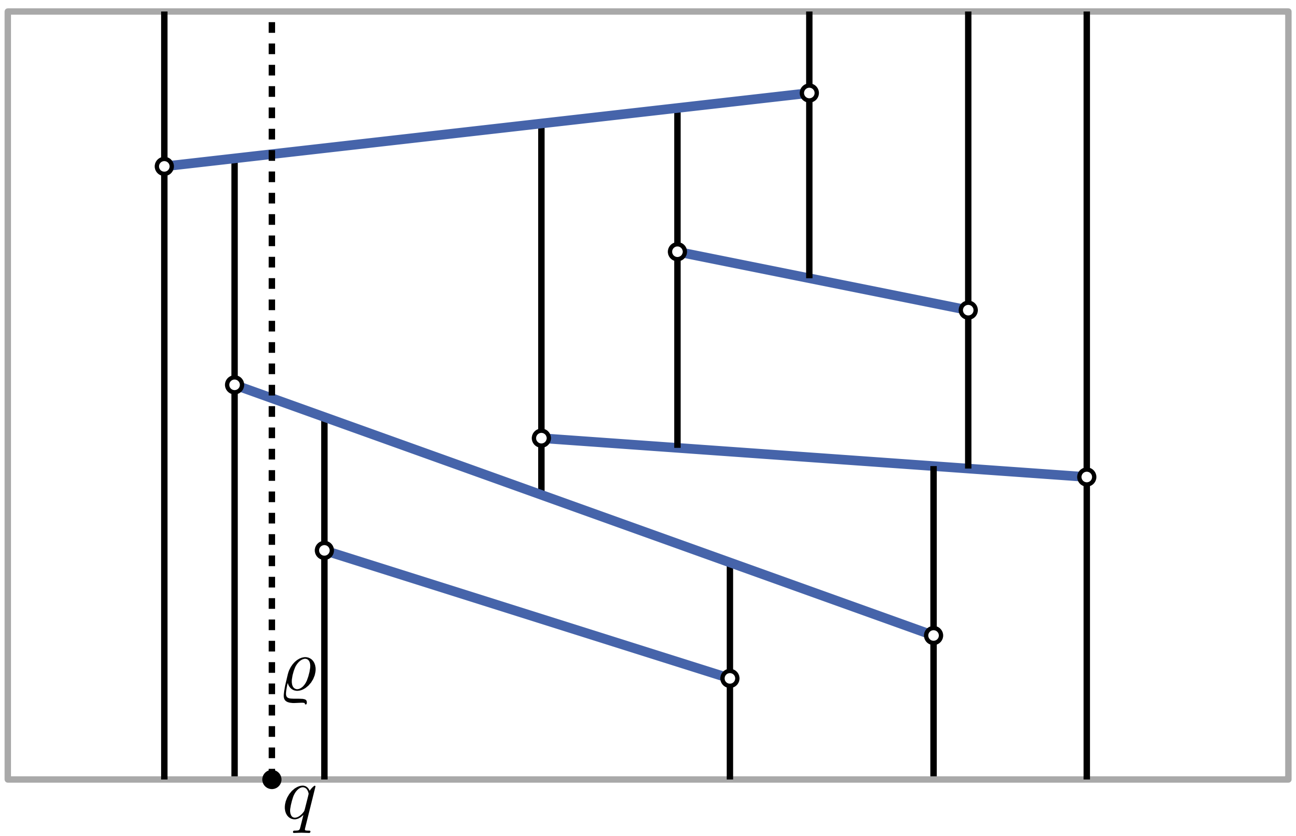
Exercise 4

Ray-Shooting Problem

Here: Simplified.

Point $q \in \mathbb{R}^2$ and n intersecting-free segments are given. Let ϱ be a vertical half-line that *shoots* upwards from q.

Find 'first' segment that intersects ϱ .



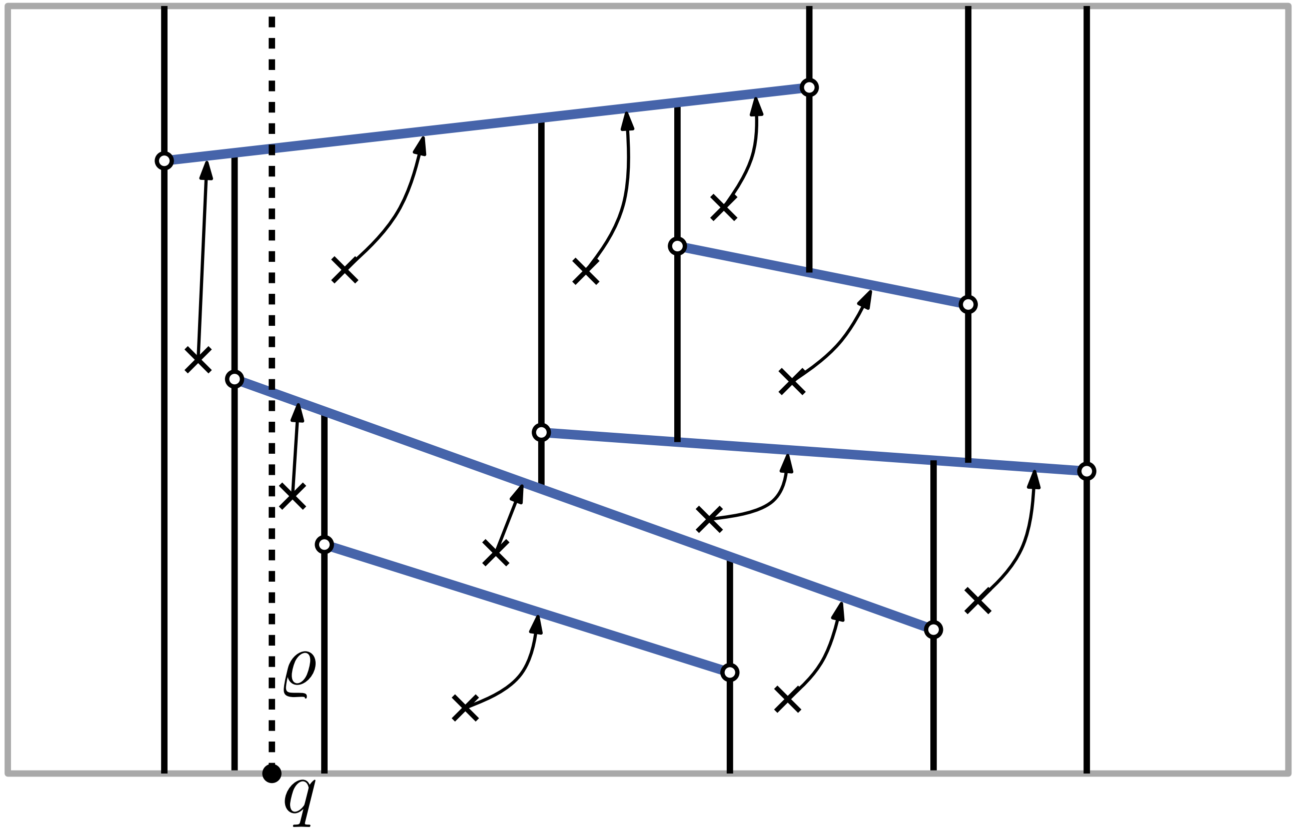
Exercise 4

Ray-Shooting Problem

Here: Simplified.

Point $q \in \mathbb{R}^2$ and n intersecting-free segments are given. Let ϱ be a vertical half-line that *shoots* upwards from q.

Find 'first' segment that intersects ϱ .



Exercise 4

Ray-Shooting Problem   Here: Simplified.

Point $q \in \mathbb{R}^2$ and n **possibly intersecting** segments are given. Let ϱ be a vertical half-line that *shoots* upwards from q.

Find 'first' segment that intersects ϱ .

Exercise 4

Ray-Shooting Problem

Here: Simplified.

Point $q \in \mathbb{R}^2$ and n **possibly intersecting** segments are given. Let ϱ be a vertical half-line that *shoots* upwards from q.

Find 'first' segment that intersects ϱ .

1. Determine intersections of segments.
2. Introduce for each intersection a pseudo-vertex.
3. Use trapezoidal map.

Exercise 4

Ray-Shooting Problem

Here: Simplified.

Point $q \in \mathbb{R}^2$ and n **possibly intersecting** segments are given. Let ϱ be a vertical half-line that *shoots* upwards from q.

Find 'first' segment that intersects ϱ .

1. Determine intersections of segments.
2. Introduce for each intersection a pseudo-vertex.
3. Use trapezoidal map.

Expected Query-Time: $O(\log(n+k))$

Expected time for construction: $O((n+k)\log(n+k))$

Space consumption: $O(n+k)$