Computational Geometry

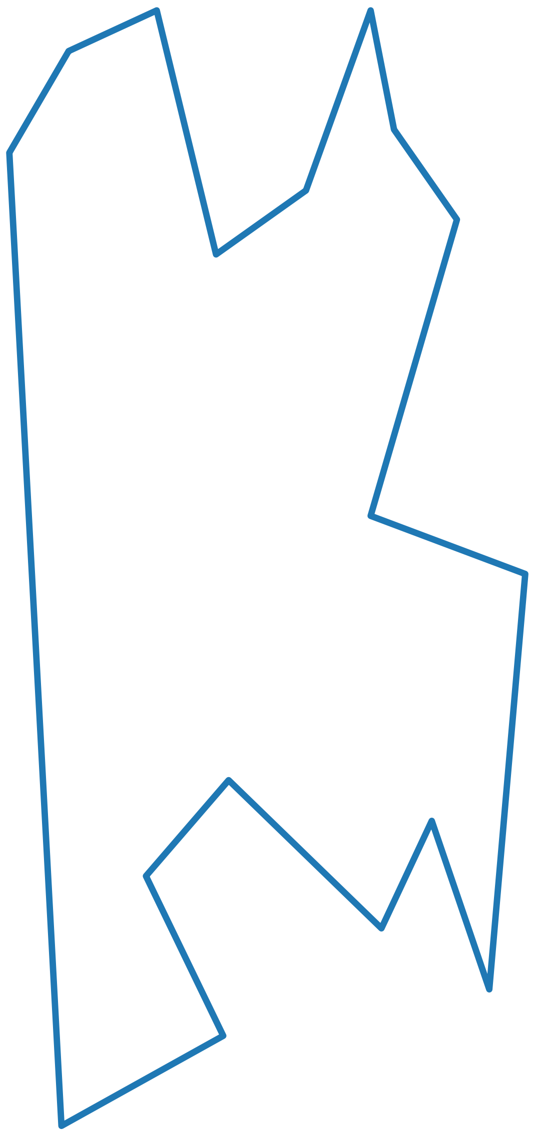
Lecture 12: Seidel's Triangulation Algorithm

Part I: General Idea

Philipp Kindermann

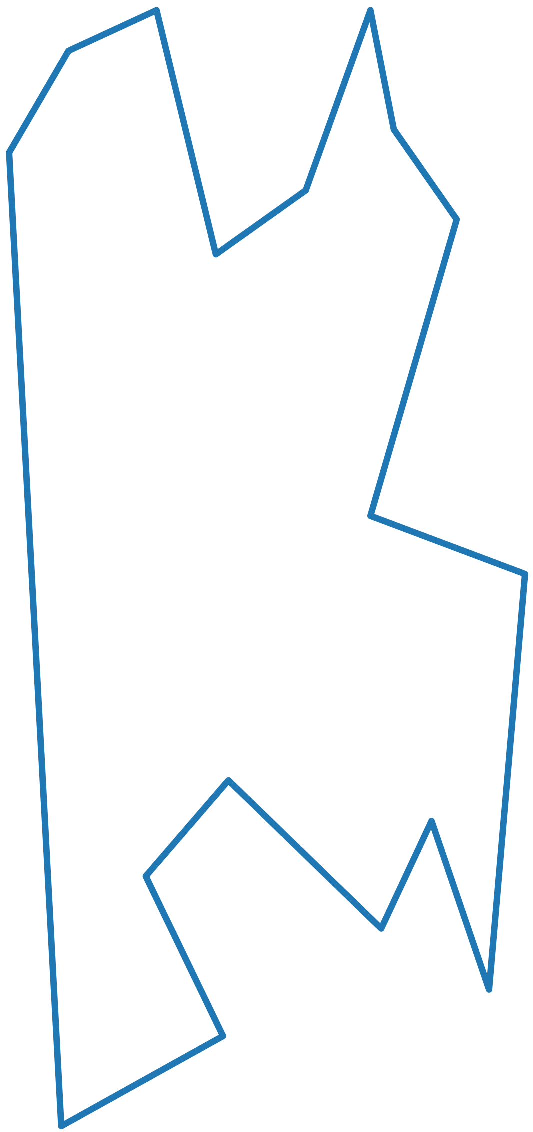
Winter Semester 2020

Triangulating a Polygon



Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

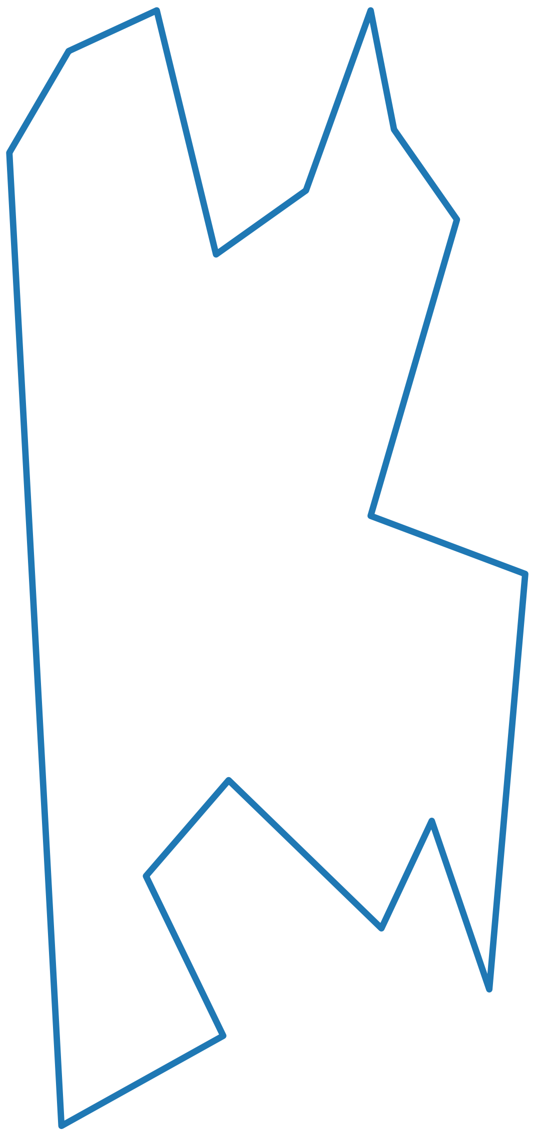
Triangulating a Polygon



Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P

Triangulating a Polygon

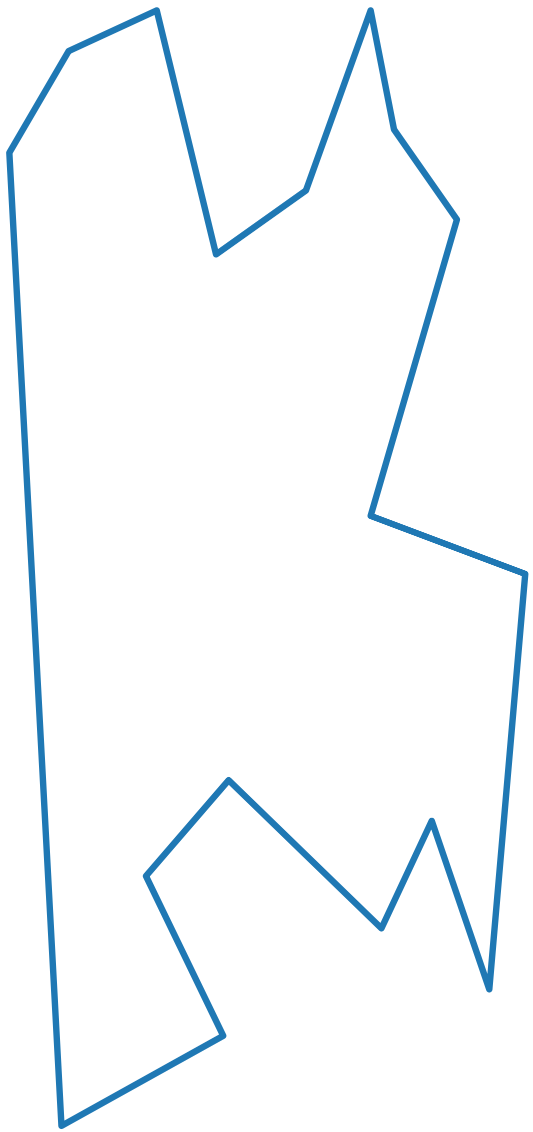


Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P

i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Triangulating a Polygon

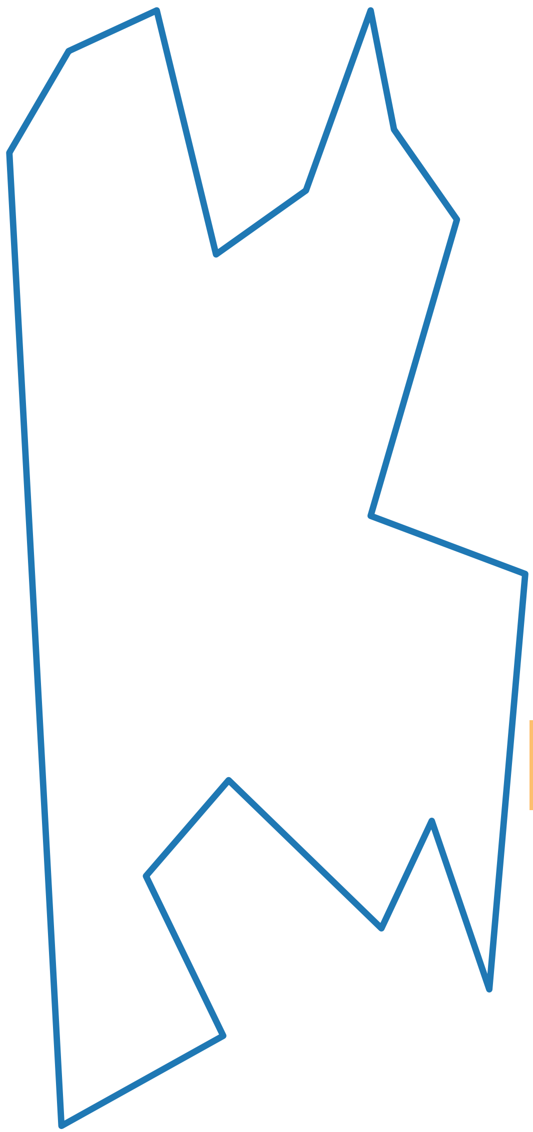


Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

Triangulating a Polygon



Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

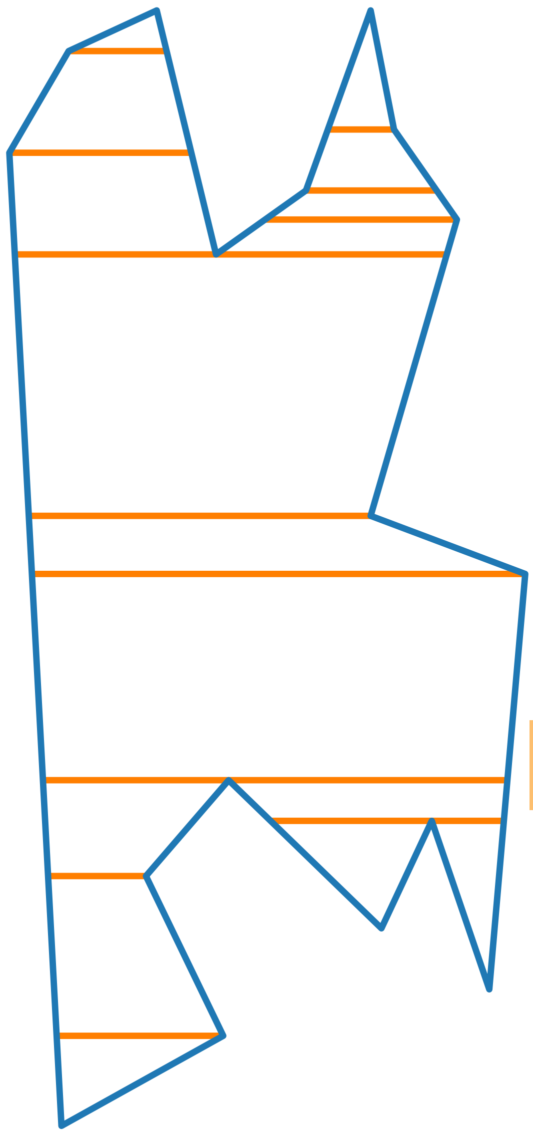
Find: Triangulation of P

i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

1. Trapezoidize interior of P.

Triangulating a Polygon



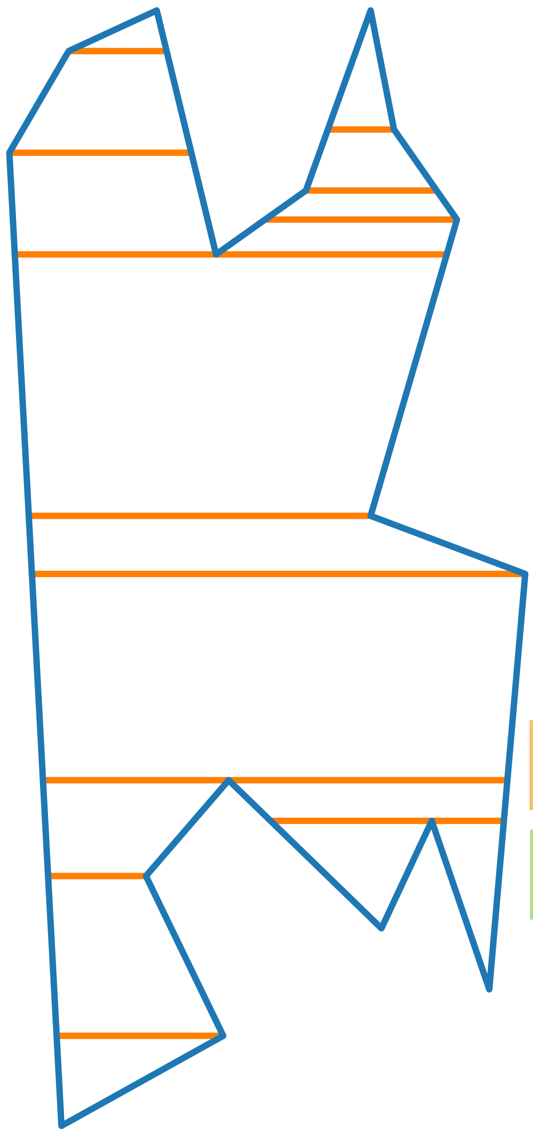
Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

1. Trapezoidize interior of P.

Triangulating a Polygon



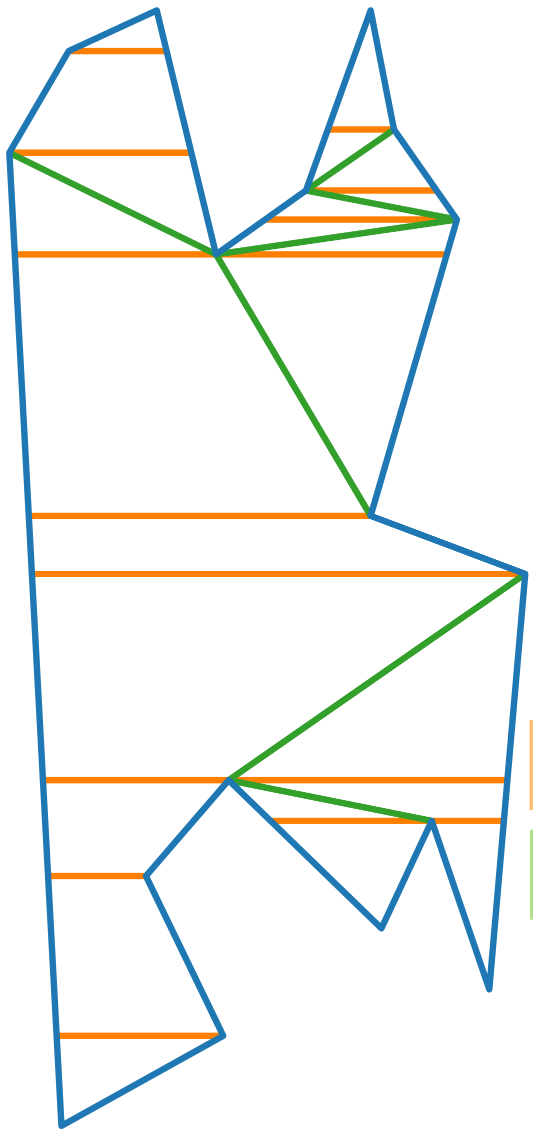
Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

1. Trapezoidize interior of P.
2. Draw diagonals inside trapezoids.

Triangulating a Polygon



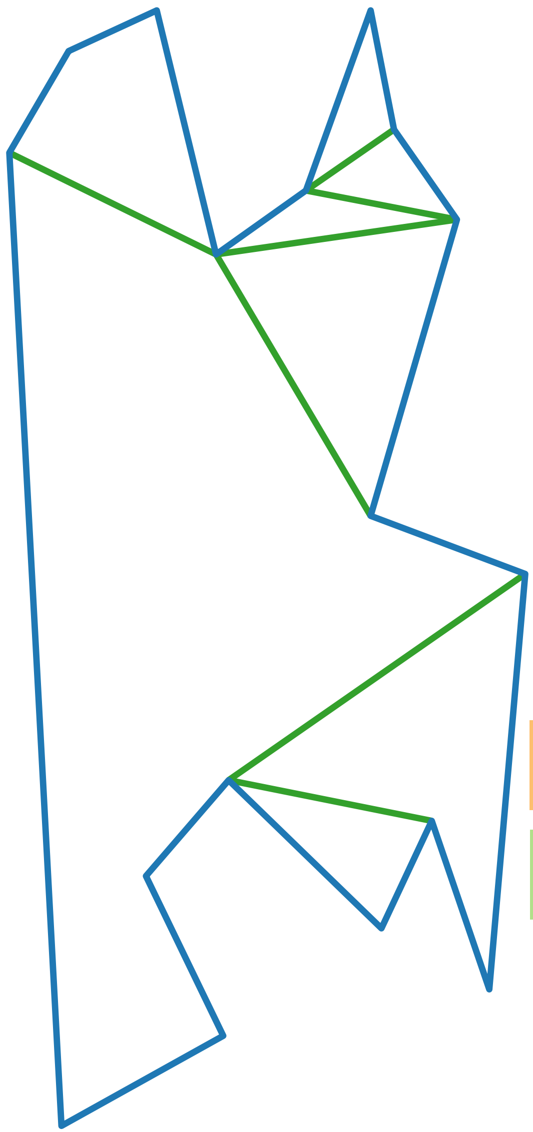
Given: Polygon $P = \langle p_1, \dots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

1. Trapezoidize interior of P.
2. Draw diagonals inside trapezoids.

Triangulating a Polygon



Given: Polygon $P = \langle p_1, \dots, p_n \rangle$
(list of vertices in cw order)

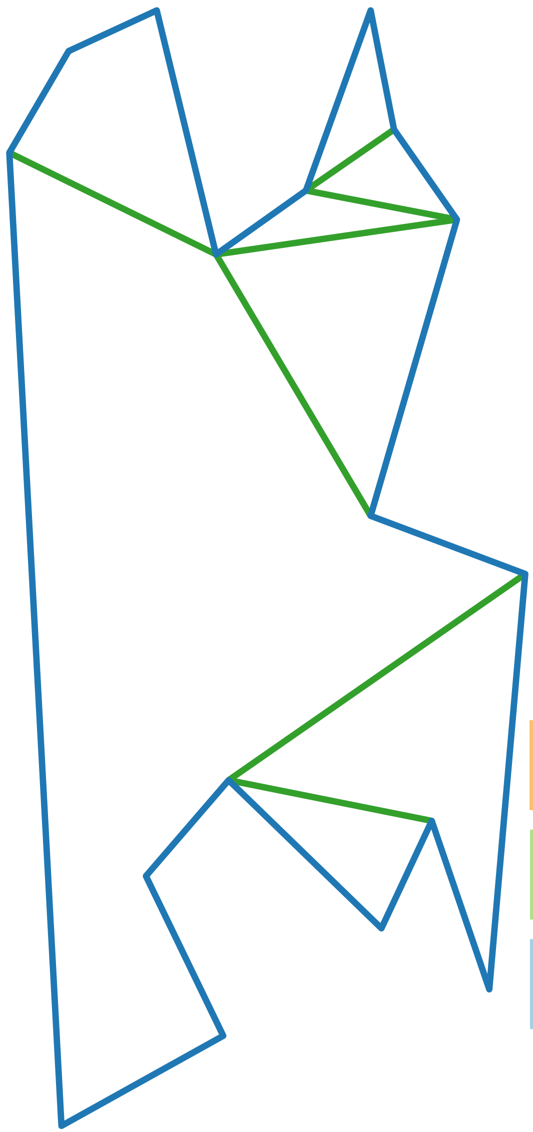
Find: Triangulation of P

i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

1. Trapezoidize interior of P.
2. Draw diagonals inside trapezoids.

Triangulating a Polygon



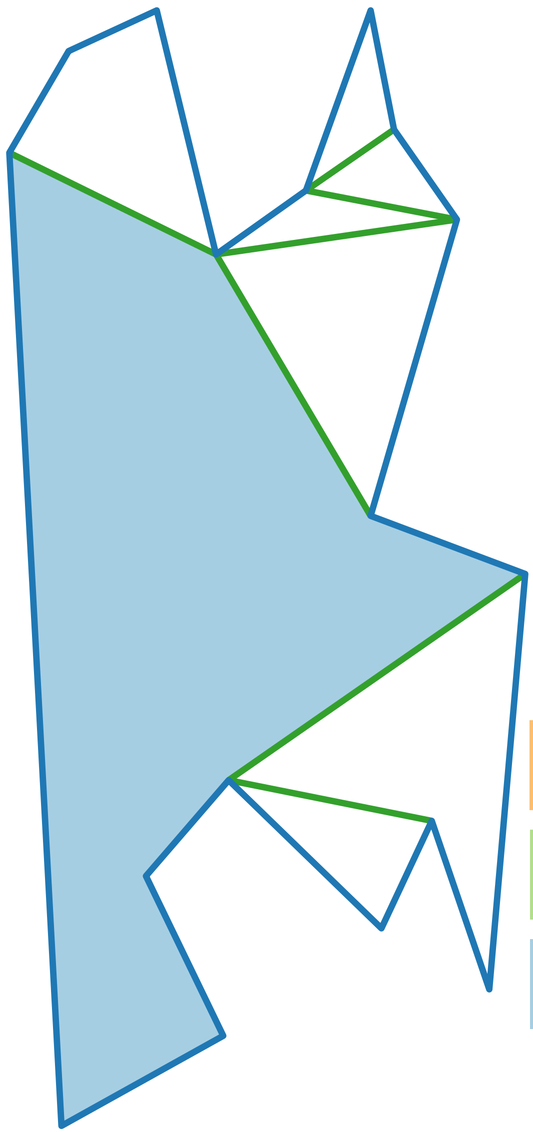
Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

1. Trapezoidize interior of P.
2. Draw diagonals inside trapezoids.
3. Triangulate y-monotone subpolygons.

Triangulating a Polygon



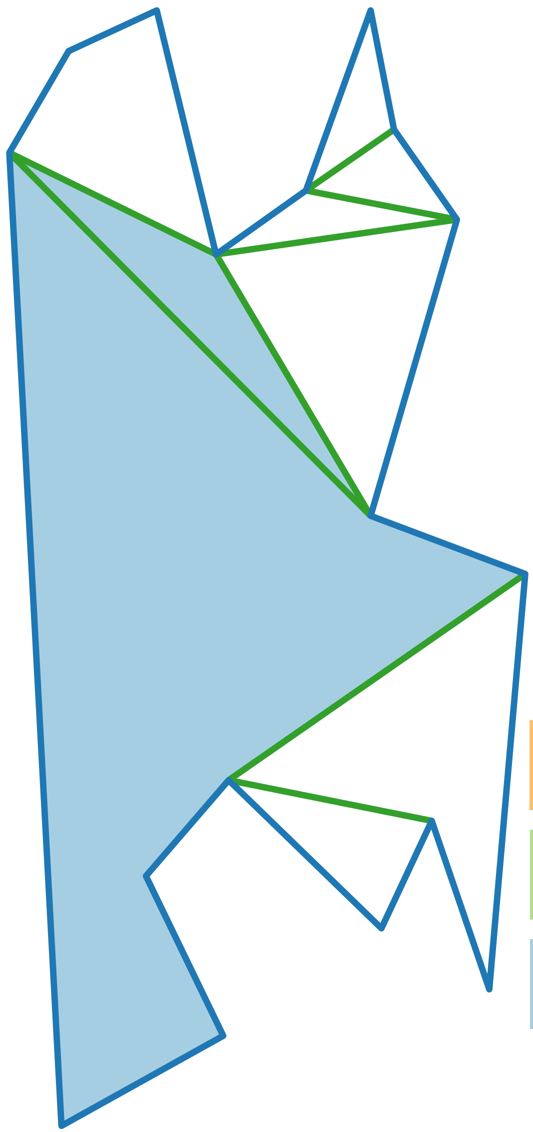
Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

1. Trapezoidize interior of P.
2. Draw diagonals inside trapezoids.
3. Triangulate y-monotone subpolygons.

Triangulating a Polygon



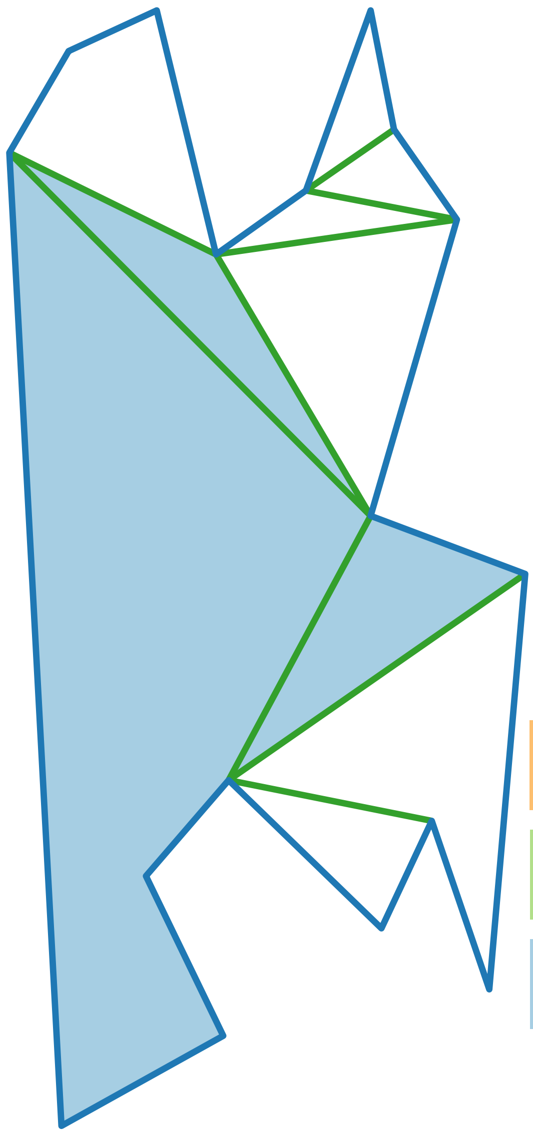
Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

1. Trapezoidize interior of P.
2. Draw diagonals inside trapezoids.
3. Triangulate y-monotone subpolygons.

Triangulating a Polygon



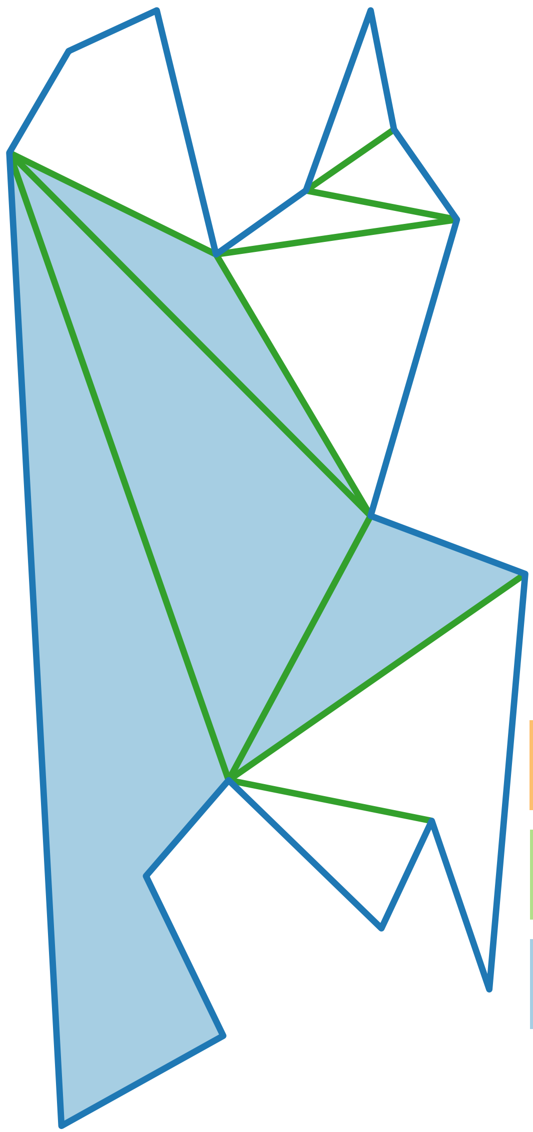
Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

1. Trapezoidize interior of P.
2. Draw diagonals inside trapezoids.
3. Triangulate y-monotone subpolygons.

Triangulating a Polygon



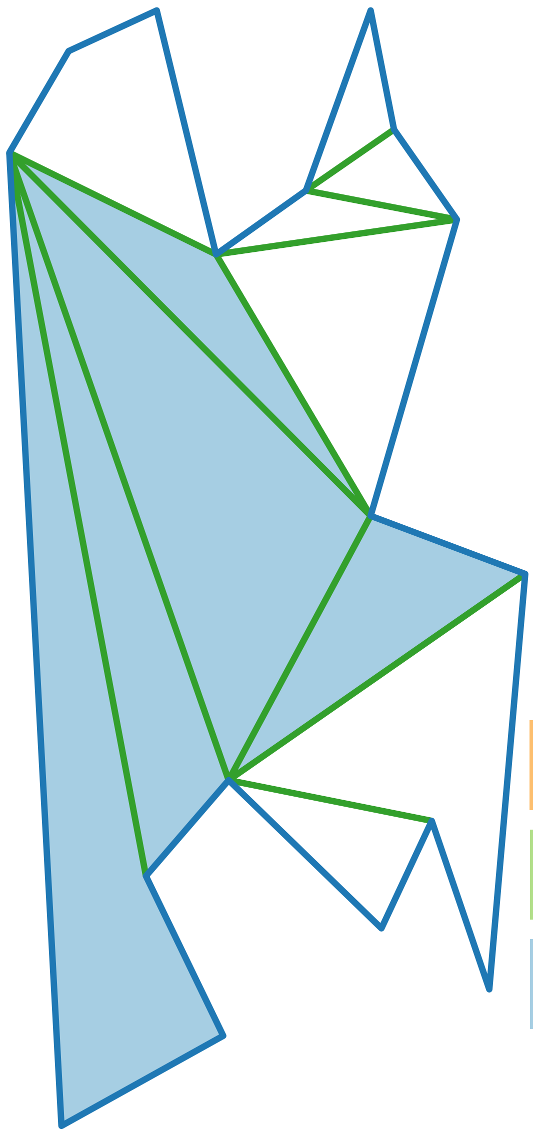
Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

1. Trapezoidize interior of P.
2. Draw diagonals inside trapezoids.
3. Triangulate y-monotone subpolygons.

Triangulating a Polygon



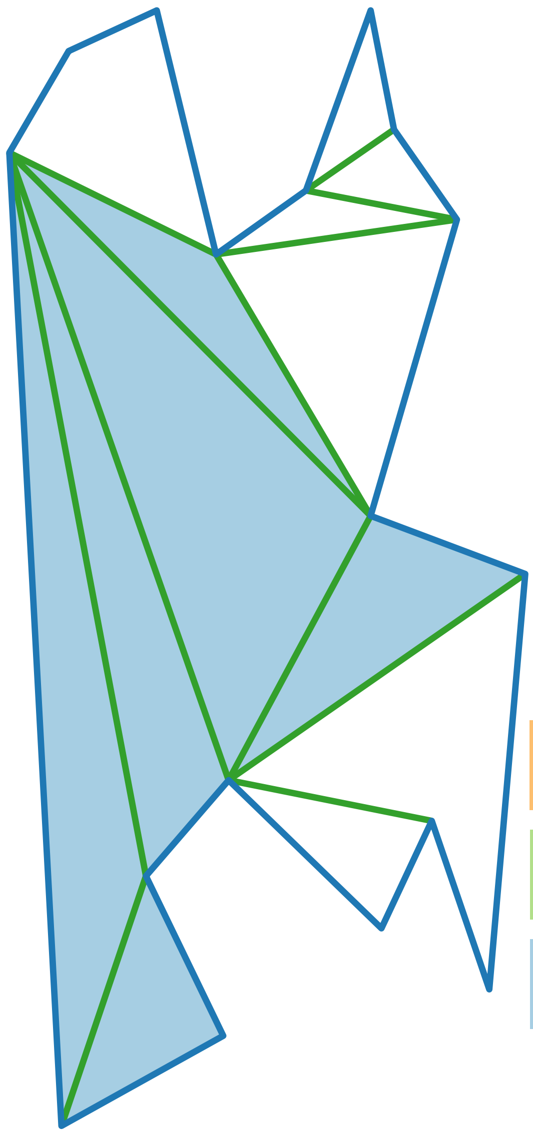
Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

1. Trapezoidize interior of P.
2. Draw diagonals inside trapezoids.
3. Triangulate y-monotone subpolygons.

Triangulating a Polygon



Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

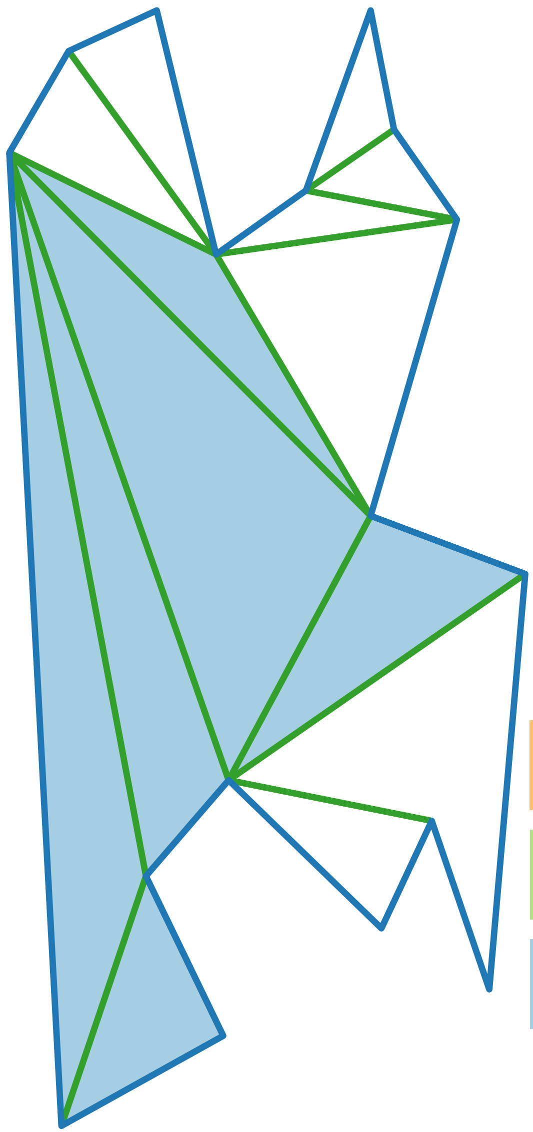
Find: Triangulation of P

i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

1. Trapezoidize interior of P.
2. Draw diagonals inside trapezoids.
3. Triangulate y-monotone subpolygons.

Triangulating a Polygon



Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

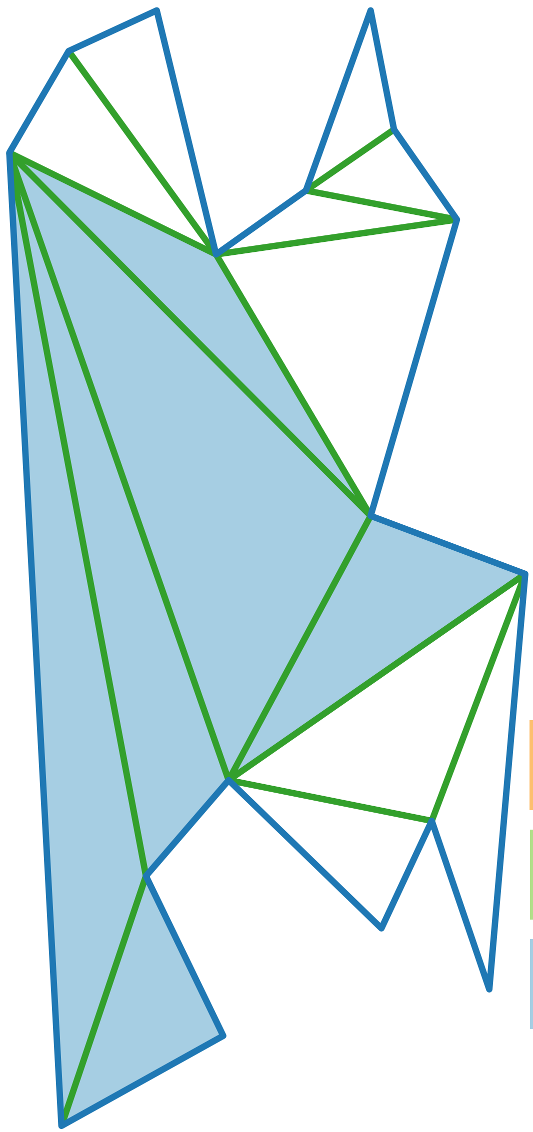
Find: Triangulation of P

i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

1. Trapezoidize interior of P.
2. Draw diagonals inside trapezoids.
3. Triangulate y-monotone subpolygons.

Triangulating a Polygon



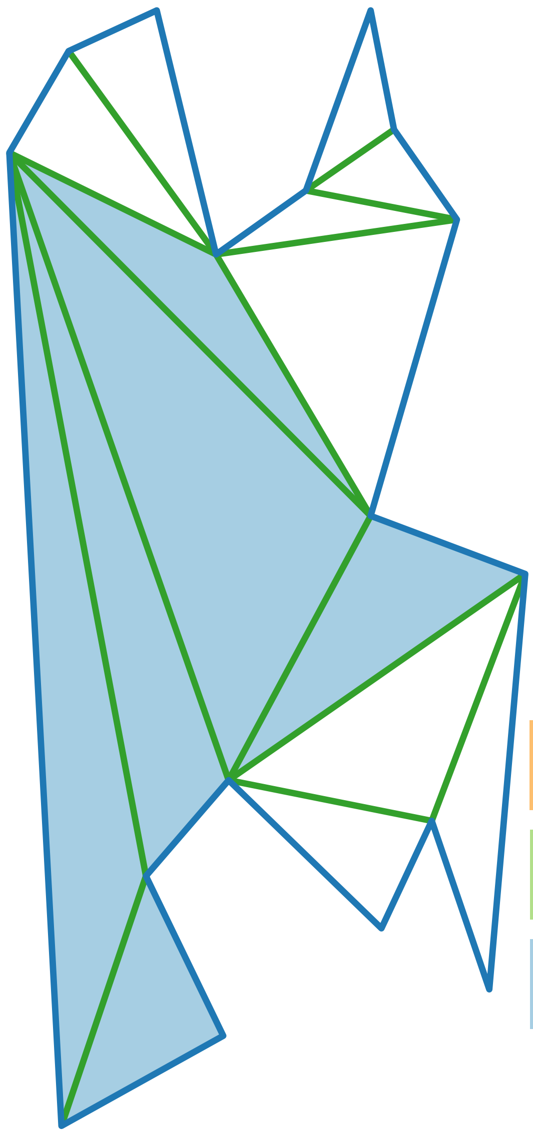
Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

1. Trapezoidize interior of P.
2. Draw diagonals inside trapezoids.
3. Triangulate y-monotone subpolygons.

Triangulating a Polygon



Given: Polygon $P = \langle p_1, \dots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P

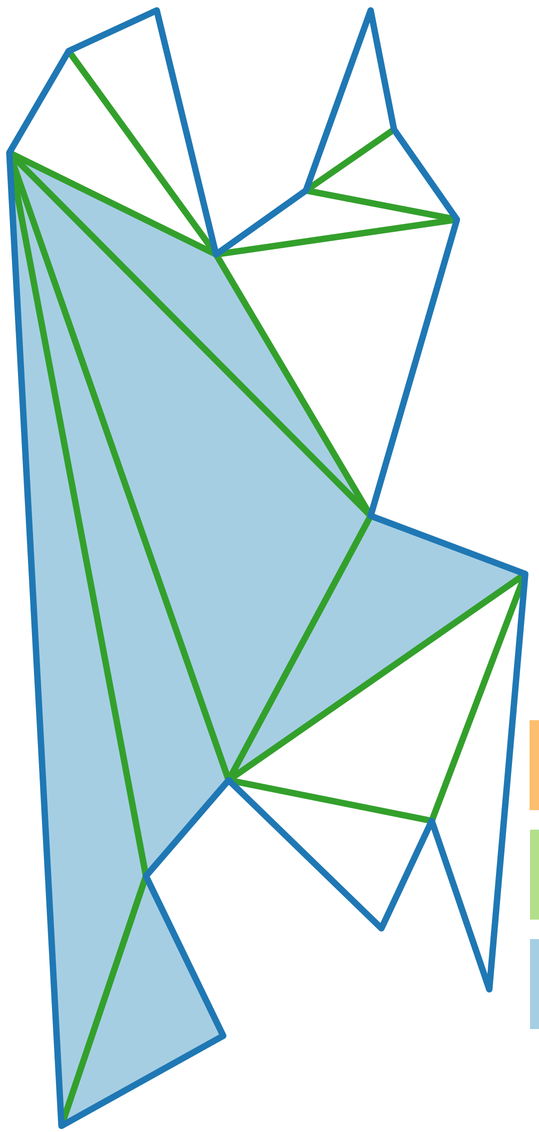
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

Running time:

1. Trapezoidize interior of P.
2. Draw diagonals inside trapezoids.
3. Triangulate y-monotone subpolygons.

Triangulating a Polygon



Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P

i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

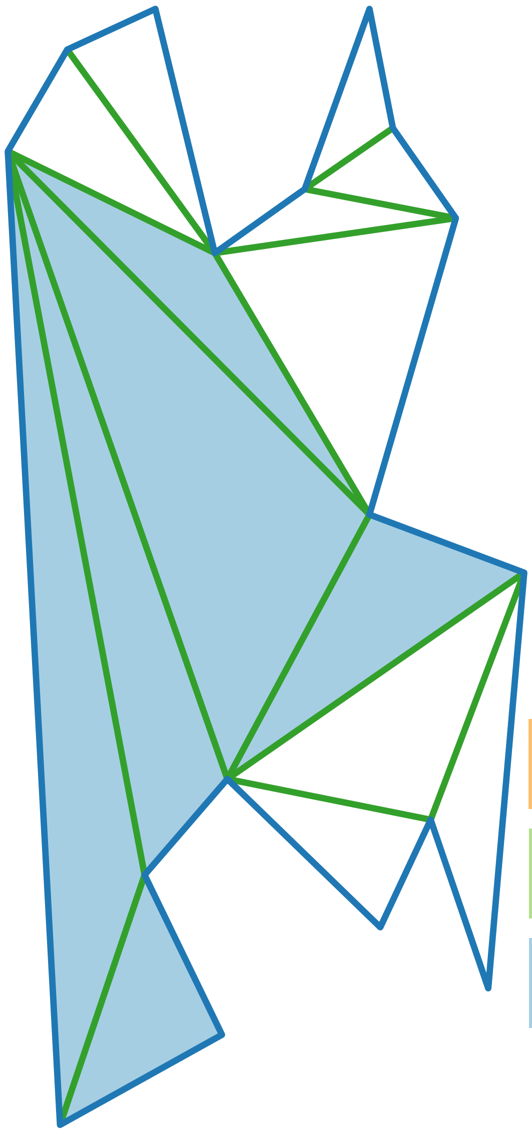
Running time:

1. Trapezoidize interior of P.

$O(n \log n)$

2. Draw diagonals inside trapezoids.
3. Triangulate y-monotone subpolygons.

Triangulating a Polygon



Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P

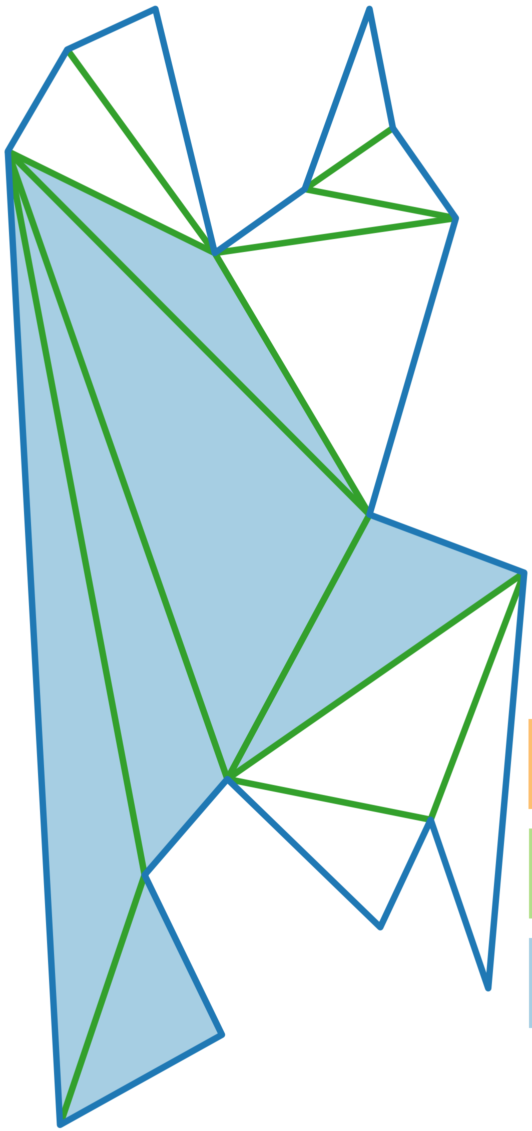
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

Running time:

1. Trapezoidize interior of P. $O(n \log n)$
2. Draw diagonals inside trapezoids. $O(n)$
3. Triangulate y-monotone subpolygons.

Triangulating a Polygon



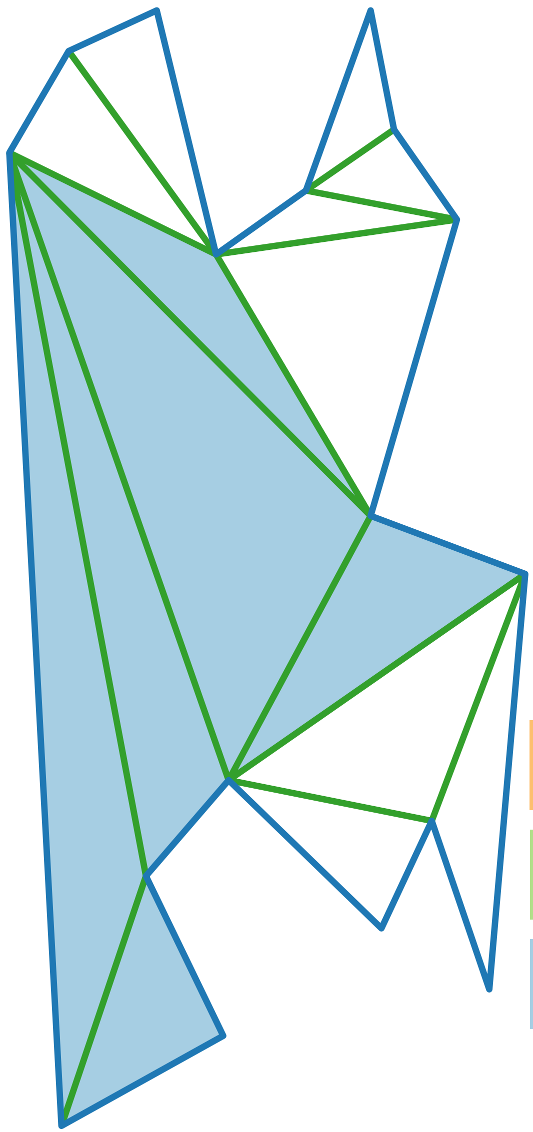
Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach: *Running time:*

1. Trapezoidize interior of P. $O(n \log n)$
2. Draw diagonals inside trapezoids. $O(n)$
3. Triangulate y-monotone subpolygons. $O(n)$

Triangulating a Polygon



Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

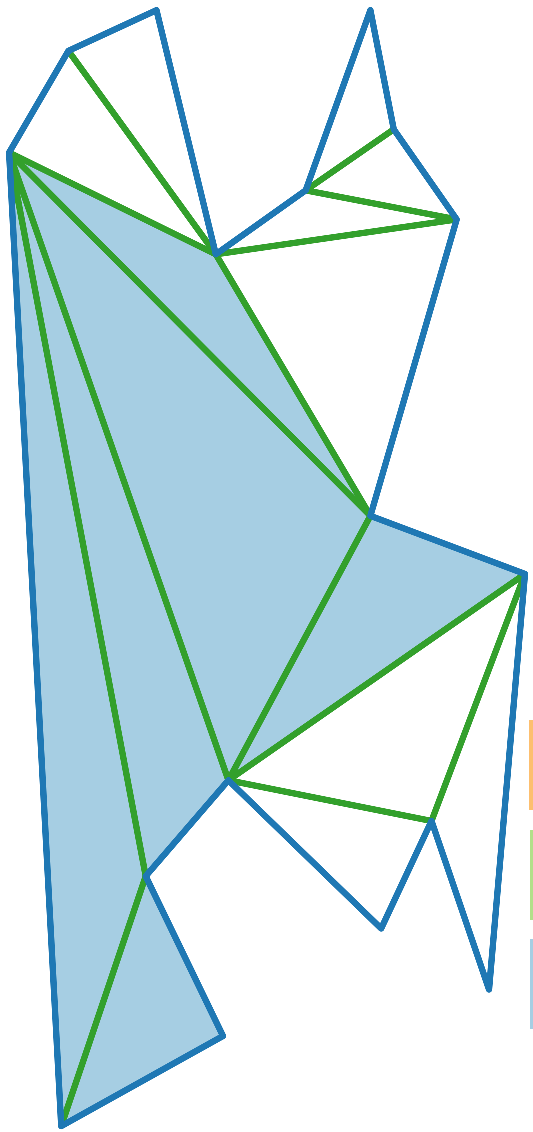
Find: Triangulation of P
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach:

Running time:

1. Trapezoidize interior of P. $O(n \log n)$
2. Draw diagonals inside trapezoids. $O(n)$
3. Triangulate y-monotone subpolygons. $O(n)$

Triangulating a Polygon



Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

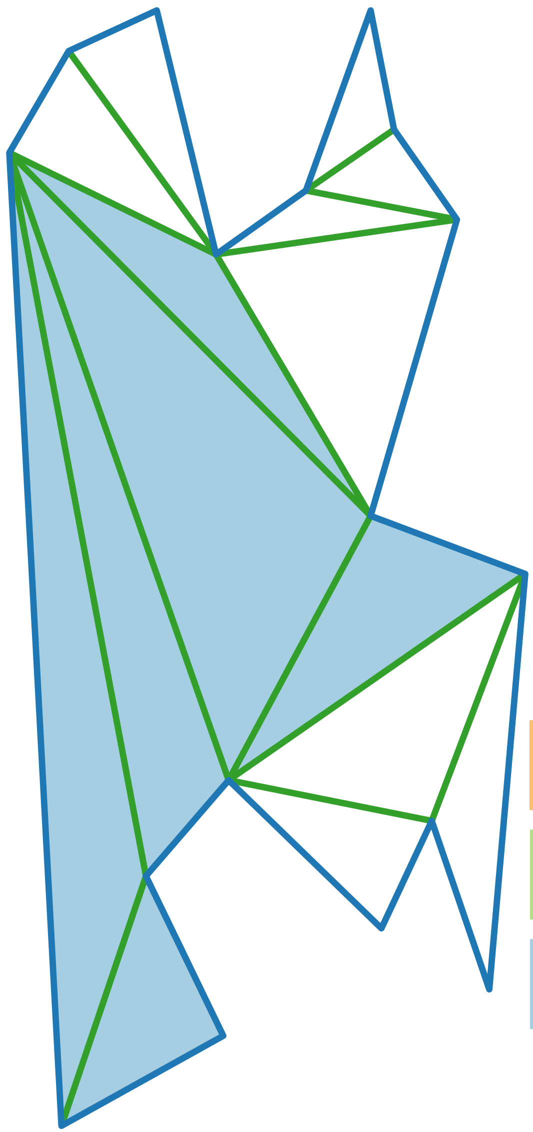
Approach:

Running time:

1. Trapezoidize interior of P. $O(n \log n)$
2. Draw diagonals inside trapezoids. $O(n)$
3. Triangulate y-monotone subpolygons. $O(n)$

$O(n \log n)$

Triangulating a Polygon



Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P

i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

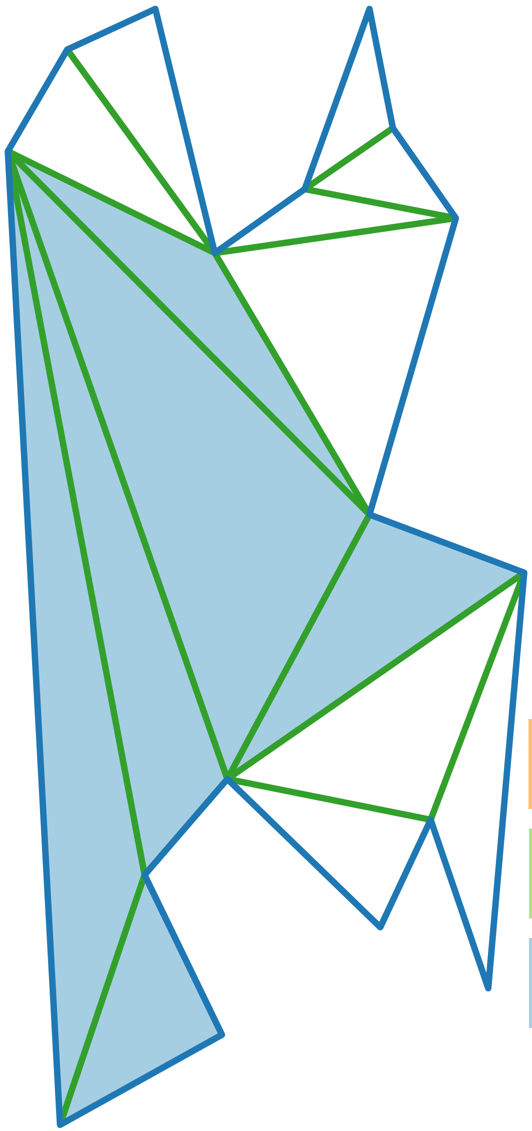
Approach:

Running time:

1. Trapezoidize interior of P. $O(n \log n)$
2. Draw diagonals inside trapezoids. $O(n)$
3. Triangulate y-monotone subpolygons. $O(n)$

$O(n \log n)$

Triangulating a Polygon



Given: Polygon $P = \langle p_1, \ldots, p_n \rangle$
(list of vertices in cw order)

Find: Triangulation of P
i.e., a partition of P into triangles by *diagonals* (segments of type $\overline{p_i p_j} \subset P$)

Approach: *Running time:*

1. Trapezoidize interior of P. $O(n \log n)$
2. Draw diagonals inside trapezoids. $O(n)$
3. Triangulate y-monotone subpolygons. $O(n)$

$O(n \log n)$

Lemma 1. Given a trapezoidation, a polygon can be triangulated in linear time.

General Idea

Let S be a set of n non-crossing segments

General Idea

Let S be a set of n non-crossing segments

WANTED:

General Idea

Let S be a set of n non-crossing segments

WANTED: – trapezoidation $\mathcal{T}(S)$ of S

General Idea

Let S be a set of n non-crossing segments

WANTED: – trapezoidation $\mathcal{T}(S)$ of S
– point-location data structure $\mathcal{Q}(S)$

General Idea

Let S be a set of n non-crossing segments

WANTED: – trapezoidation $\mathcal{T}(S)$ of S
– point-location data structure $\mathcal{Q}(S)$

Our construction is randomized-incremental:

General Idea

Let S be a set of n non-crossing segments

WANTED: – trapezoidation $\mathcal{T}(S)$ of S
– point-location data structure $\mathcal{Q}(S)$

Our construction is randomized-incremental:

Trapezoidation (set S of n non-crossing line segments)

General Idea

Let S be a set of n non-crossing segments

WANTED: – trapezoidation $\mathcal{T}(S)$ of S
– point-location data structure $\mathcal{Q}(S)$

Our construction is randomized-incremental:

Trapezoidation (set S of n non-crossing line segments)
$\langle s_1, s_2, \ldots, s_n \rangle \leftarrow$ random ordering of S

General Idea

Let S be a set of n non-crossing segments

WANTED: – trapezoidation $\mathcal{T}(S)$ of S
– point-location data structure $\mathcal{Q}(S)$

Our construction is randomized-incremental:

Trapezoidation (set S of n non-crossing line segments)

$\langle s_1, s_2, \ldots, s_n \rangle \leftarrow$ random ordering of S

$S_0 \leftarrow \emptyset$

General Idea

Let S be a set of n non-crossing segments

WANTED: – trapezoidation $\mathcal{T}(S)$ of S
– point-location data structure $\mathcal{Q}(S)$

Our construction is randomized-incremental:

Trapezoidation (set S of n non-crossing line segments)

$\langle s_1, s_2, \ldots, s_n \rangle \leftarrow$ random ordering of S
$S_0 \leftarrow \emptyset$
for $i = 1$ **to** n **do**

General Idea

Let S be a set of n non-crossing segments

WANTED: – trapezoidation $\mathcal{T}(S)$ of S
– point-location data structure $\mathcal{Q}(S)$

Our construction is randomized-incremental:

Trapezoidation (set S of n non-crossing line segments)

$\langle s_1, s_2, \ldots, s_n \rangle \leftarrow$ random ordering of S
$S_0 \leftarrow \emptyset$
for $i = 1$ **to** n **do**
  $S_i \leftarrow S_{i-1} \cup \{s_i\}$

General Idea

Let S be a set of n non-crossing segments

WANTED: – trapezoidation $\mathcal{T}(S)$ of S
– point-location data structure $\mathcal{Q}(S)$

Our construction is randomized-incremental:

Trapezoidation (set S of n non-crossing line segments)

$\langle s_1, s_2, \ldots, s_n \rangle \leftarrow$ random ordering of S
$S_0 \leftarrow \emptyset$
for $i = 1$ **to** n **do**
  $S_i \leftarrow S_{i-1} \cup \{s_i\}$
  use $\mathcal{T}(S_{i-1})$ and $\mathcal{Q}(S_{i-1})$ to construct $\mathcal{T}(S_i)$ and $\mathcal{Q}(S_i)$

General Idea

Let S be a set of n non-crossing segments

WANTED: – trapezoidation $\mathcal{T}(S)$ of S
– point-location data structure $\mathcal{Q}(S)$

Our construction is randomized-incremental:

Trapezoidation (set S of n non-crossing line segments)

$\langle s_1, s_2, \ldots, s_n \rangle \leftarrow$ random ordering of S
$S_0 \leftarrow \emptyset$
for $i = 1$ **to** n **do**
  $S_i \leftarrow S_{i-1} \cup \{s_i\}$
  use $\mathcal{T}(S_{i-1})$ and $\mathcal{Q}(S_{i-1})$ to construct $\mathcal{T}(S_i)$ and $\mathcal{Q}(S_i)$

Total cost of one step:

General Idea

Let S be a set of n non-crossing segments

WANTED: – trapezoidation $\mathcal{T}(S)$ of S
– point-location data structure $\mathcal{Q}(S)$

Our construction is randomized-incremental:

Trapezoidation (set S of n non-crossing line segments)

$\langle s_1, s_2, \ldots, s_n \rangle \leftarrow$ random ordering of S
$S_0 \leftarrow \emptyset$
for $i = 1$ **to** n **do**
    $S_i \leftarrow S_{i-1} \cup \{s_i\}$
    use $\mathcal{T}(S_{i-1})$ and $\mathcal{Q}(S_{i-1})$ to construct $\mathcal{T}(S_i)$ and $\mathcal{Q}(S_i)$

Total cost of one step: – location time

General Idea

Let S be a set of n non-crossing segments

WANTED: – trapezoidation $\mathcal{T}(S)$ of S
– point-location data structure $\mathcal{Q}(S)$

Our construction is randomized-incremental:

Trapezoidation (set S of n non-crossing line segments)

$\langle s_1, s_2, \ldots, s_n \rangle \leftarrow$ random ordering of S
$S_0 \leftarrow \emptyset$
for $i = 1$ **to** n **do**
  $S_i \leftarrow S_{i-1} \cup \{s_i\}$
  use $\mathcal{T}(S_{i-1})$ and $\mathcal{Q}(S_{i-1})$ to construct $\mathcal{T}(S_i)$ and $\mathcal{Q}(S_i)$

Total cost of one step: – location time
– "threading" (updating) time

Computational Geometry

Lecture 12:
Seidel's Triangulation Algorithm

Part II:
Location & Threading Time

Philipp Kindermann

Winter Semester 2020

Threading Time

We assume general position
(no two points have the same y-coordinate).

Threading Time

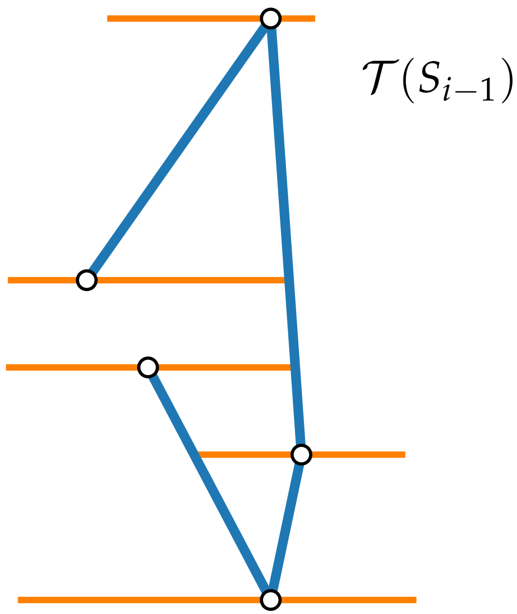
We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

Threading Time

Use lexicographic order!

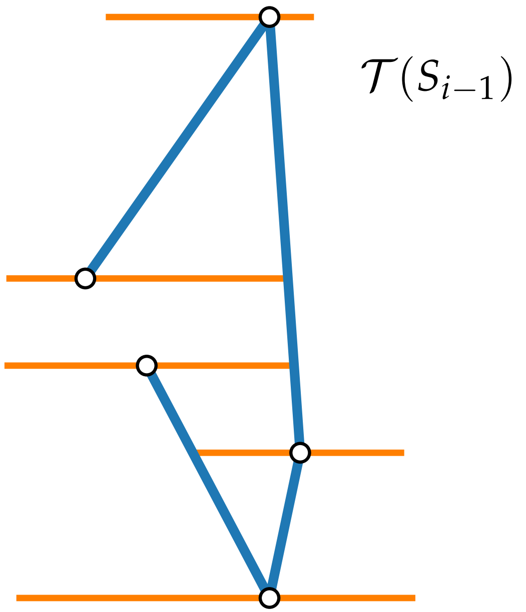
We assume general position
(no two points have the same y-coordinate).

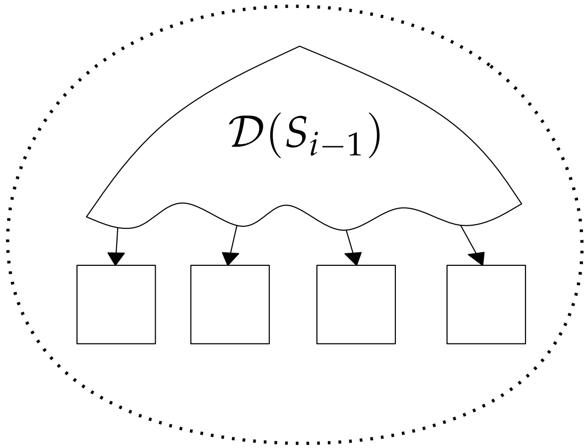


Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

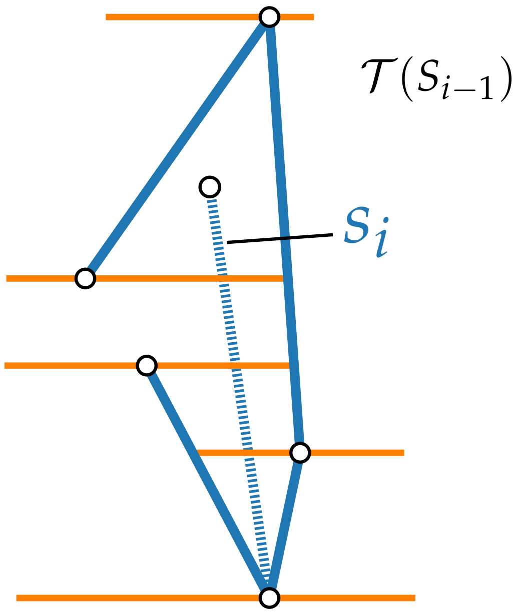


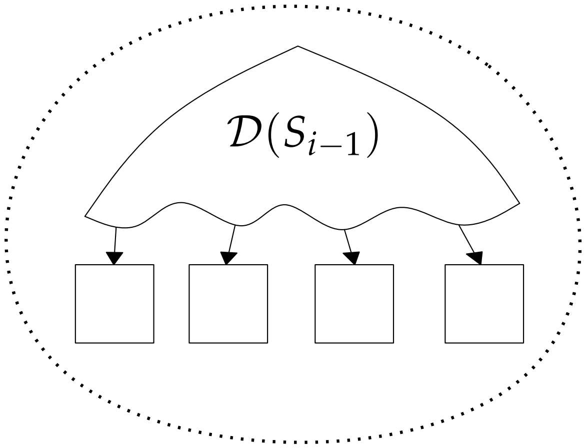


Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

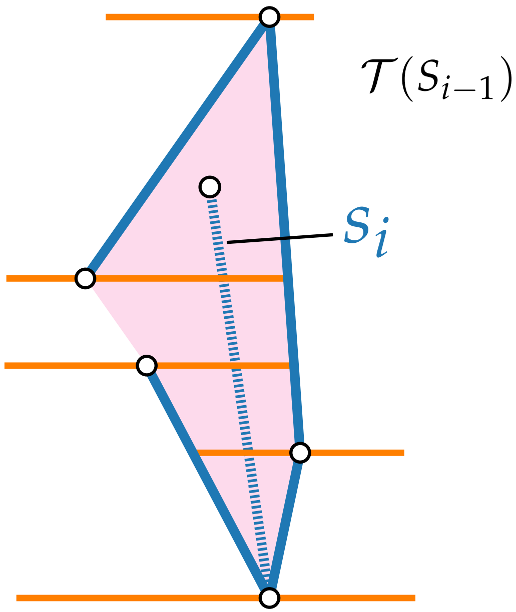


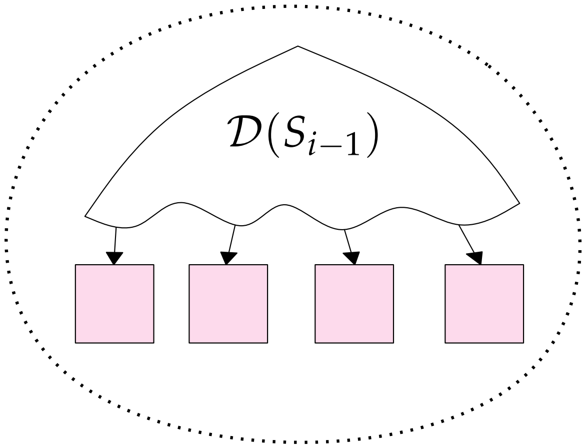


Threading Time

Use lexicographic order!

We assume general position
(no two points have the same y-coordinate).

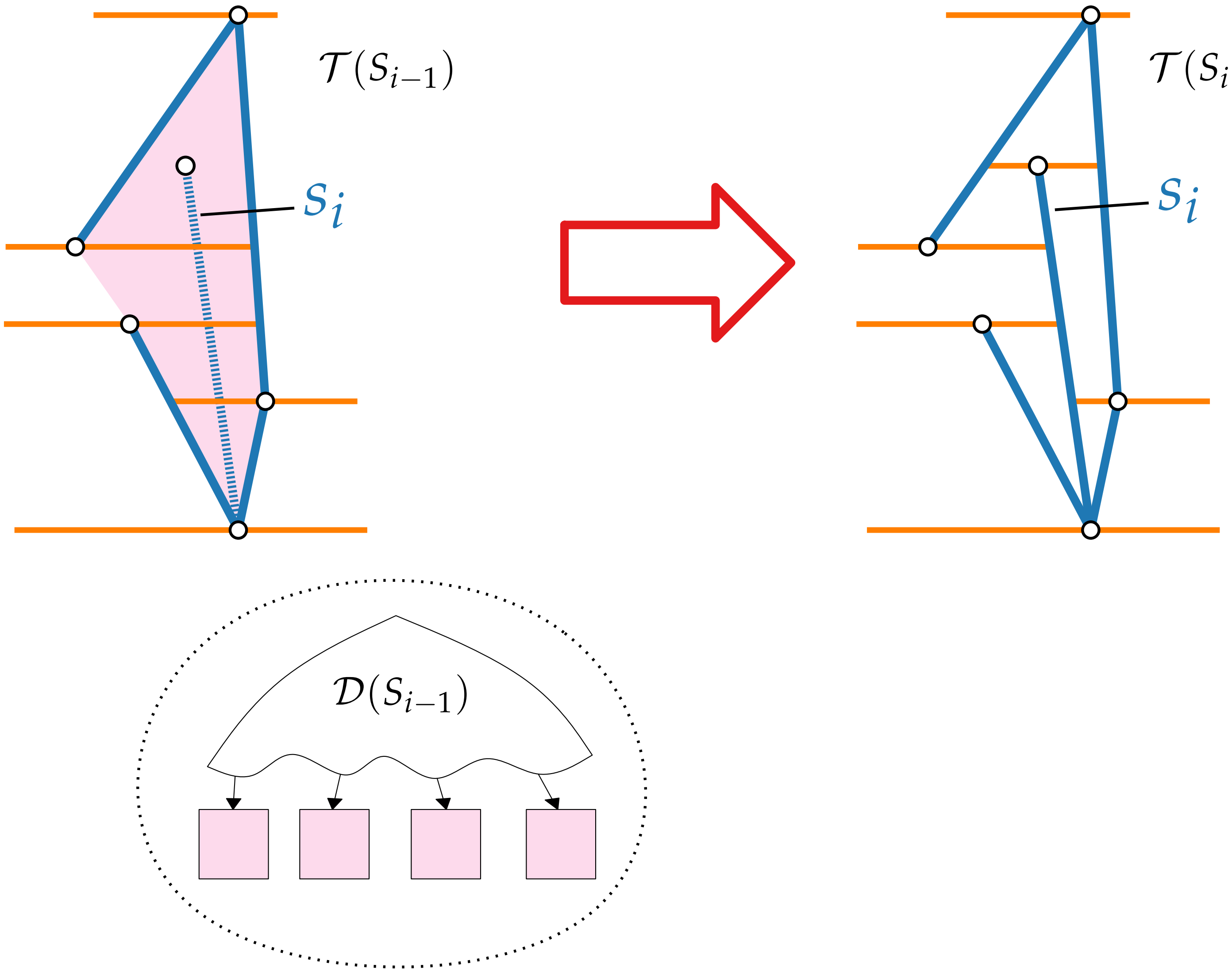




Threading Time

We assume general position
(no two points have the same y-coordinate).

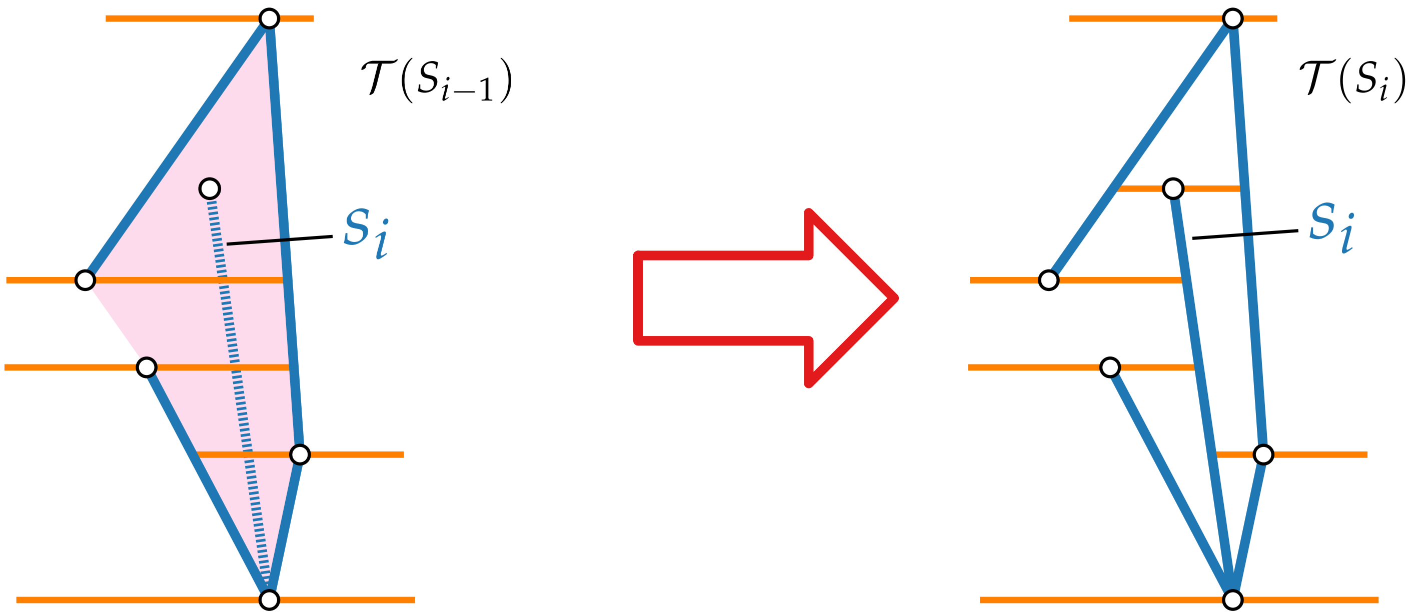
Use lexicographic order!

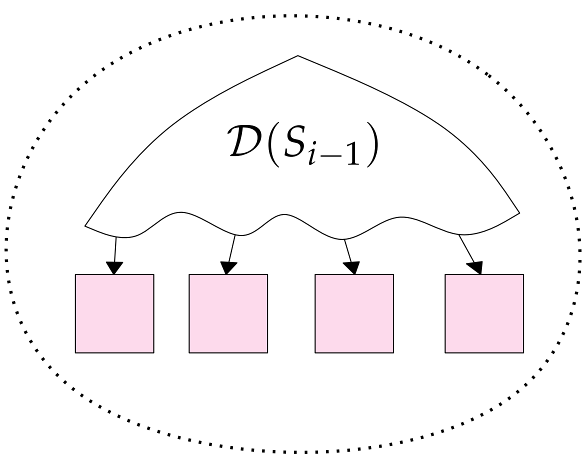


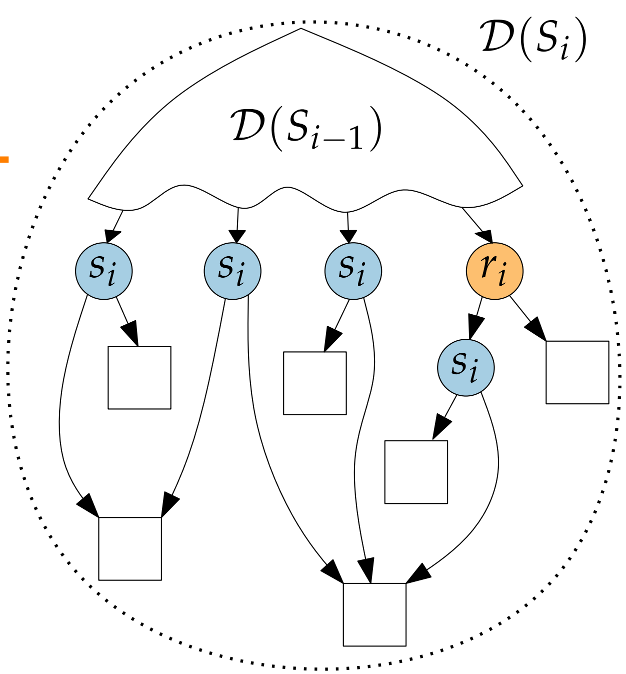
Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!



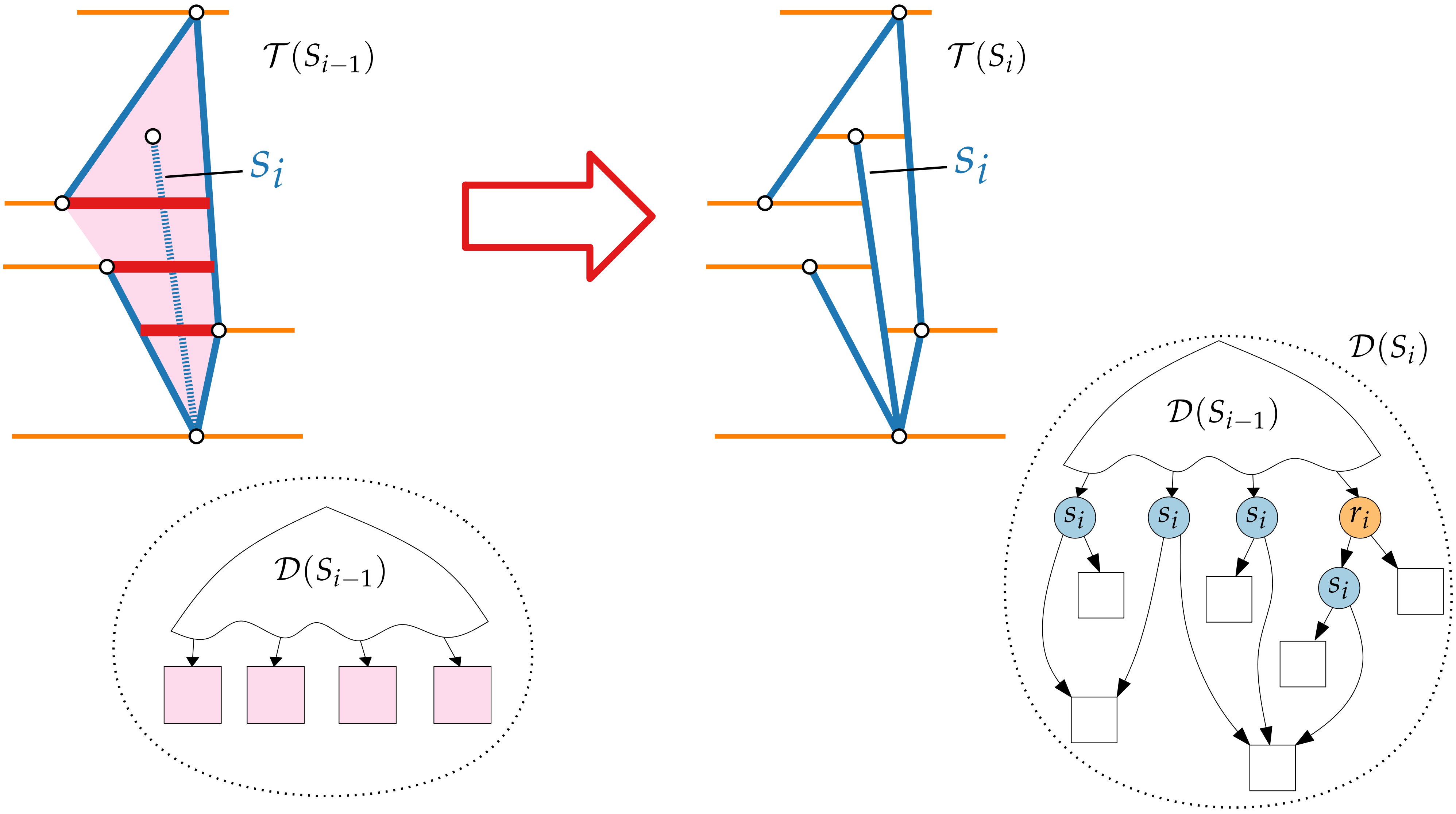




Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!



Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Proof.

Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Proof. For $s \in S_i$, let $\deg(s, \mathcal{T}(S_i)) = \#$ rays of $\mathcal{T}(S_i)$ that hit the relative interior of s.

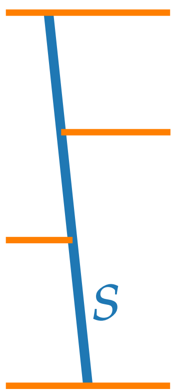
Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Proof. For $s \in S_i$, let $\deg(s, \mathcal{T}(S_i)) = \#$ rays of $\mathcal{T}(S_i)$ that hit the relative interior of s.



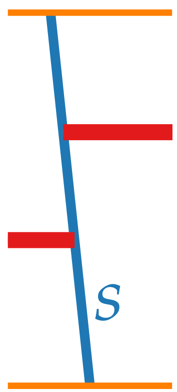
Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Proof. For $s \in S_i$, let $\deg(s, \mathcal{T}(S_i))$ = # rays of $\mathcal{T}(S_i)$ that hit the relative interior of s.



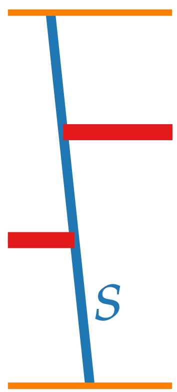
Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

Lemma 2. For $i = 1, \dots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Proof.



For $s \in S_i$, let $\deg(s, \mathcal{T}(S_i))$ = # rays of $\mathcal{T}(S_i)$ that hit the relative interior of s.

rays of $\mathcal{T}(S_{i-1})$ intersected by $s_i =$

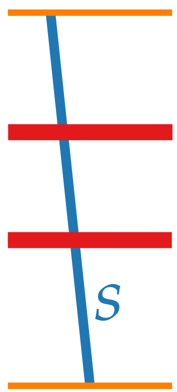
Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Proof. For $s \in S_i$, let $\deg(s, \mathcal{T}(S_i))$ = # rays of $\mathcal{T}(S_i)$ that hit the relative interior of s.



rays of $\mathcal{T}(S_{i-1})$ intersected by $s_i = \deg(s_i, \mathcal{T}(S_i))$

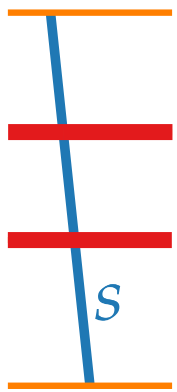
Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Proof. For $s \in S_i$, let $\deg(s, \mathcal{T}(S_i))$ = # rays of $\mathcal{T}(S_i)$ that hit the relative interior of s.



rays of $\mathcal{T}(S_{i-1})$ intersected by $s_i = \deg(s_i, \mathcal{T}(S_i))$

rays in $\mathcal{T}(S_i) \leq$

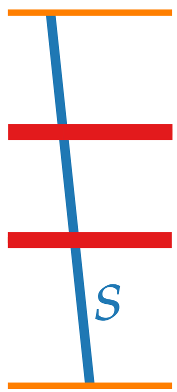
Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

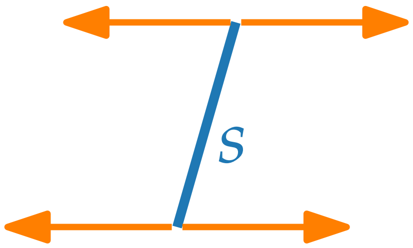
Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Proof. For $s \in S_i$, let $\deg(s, \mathcal{T}(S_i))$ = # rays of $\mathcal{T}(S_i)$ that hit the relative interior of s.



rays of $\mathcal{T}(S_{i-1})$ intersected by $s_i = \deg(s_i, \mathcal{T}(S_i))$

rays in $\mathcal{T}(S_i) \leq$



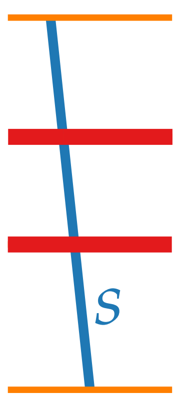
Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

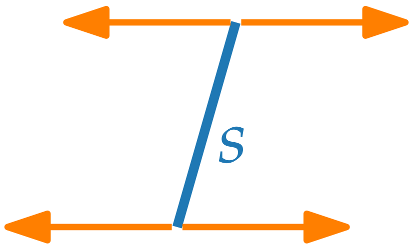
Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Proof. For $s \in S_i$, let $\deg(s, \mathcal{T}(S_i))$ = # rays of $\mathcal{T}(S_i)$ that hit the relative interior of s.



rays of $\mathcal{T}(S_{i-1})$ intersected by $s_i = \deg(s_i, \mathcal{T}(S_i))$

rays in $\mathcal{T}(S_i) \leq 4i$



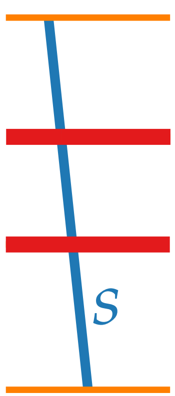
Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Proof.

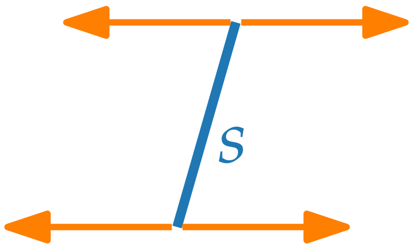


For $s \in S_i$, let $\deg(s, \mathcal{T}(S_i))$ = # rays of $\mathcal{T}(S_i)$ that hit the relative interior of s.

rays of $\mathcal{T}(S_{i-1})$ intersected by $s_i = \deg(s_i, \mathcal{T}(S_i))$

rays in $\mathcal{T}(S_i) \leq 4i$

$\Rightarrow \sum_{s \in S_i} \deg(s, \mathcal{T}(S_i)) \leq$



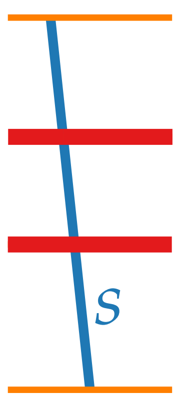
Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

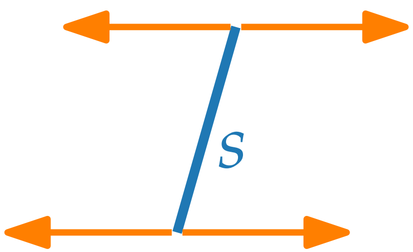
Proof. For $s \in S_i$, let $\deg(s, \mathcal{T}(S_i))$ = # rays of $\mathcal{T}(S_i)$ that hit the relative interior of s.



rays of $\mathcal{T}(S_{i-1})$ intersected by $s_i = \deg(s_i, \mathcal{T}(S_i))$

rays in $\mathcal{T}(S_i) \leq 4i$

$\Rightarrow \sum_{s \in S_i} \deg(s, \mathcal{T}(S_i)) \leq 4i$



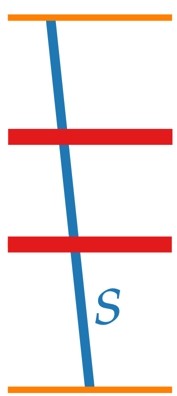
Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

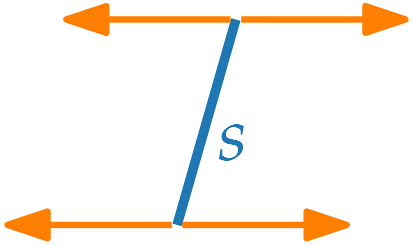
Proof. For $s \in S_i$, let $\deg(s, \mathcal{T}(S_i))$ = # rays of $\mathcal{T}(S_i)$ that hit the relative interior of s.



rays of $\mathcal{T}(S_{i-1})$ intersected by $s_i = \deg(s_i, \mathcal{T}(S_i))$

rays in $\mathcal{T}(S_i) \leq 4i$

$\Rightarrow \sum_{s \in S_i} \deg(s, \mathcal{T}(S_i)) \leq 4i$



Ordering of S_i random $\Rightarrow$

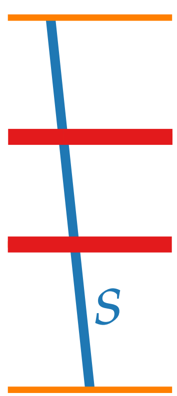
Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

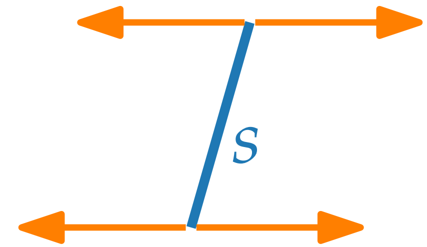
Proof. For $s \in S_i$, let $\deg(s, \mathcal{T}(S_i))$ = # rays of $\mathcal{T}(S_i)$ that hit the relative interior of s.



rays of $\mathcal{T}(S_{i-1})$ intersected by $s_i = \deg(s_i, \mathcal{T}(S_i))$

rays in $\mathcal{T}(S_i) \leq 4i$

$\Rightarrow \sum_{s \in S_i} \deg(s, \mathcal{T}(S_i)) \leq 4i$



Ordering of S_i random $\Rightarrow E[\deg(s_i, \mathcal{T}(S_i))] \leq$

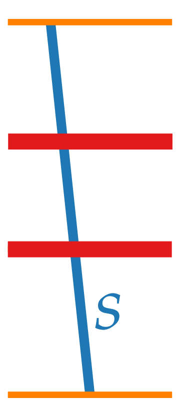
Threading Time

We assume general position
(no two points have the same y-coordinate).

Use lexicographic order!

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

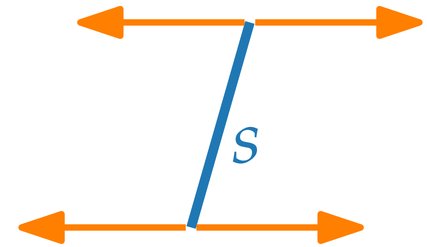
Proof. For $s \in S_i$, let $\deg(s, \mathcal{T}(S_i))$ = # rays of $\mathcal{T}(S_i)$ that hit the relative interior of s.



rays of $\mathcal{T}(S_{i-1})$ intersected by $s_i = \deg(s_i, \mathcal{T}(S_i))$

rays in $\mathcal{T}(S_i) \leq 4i$

$\Rightarrow \sum_{s \in S_i} \deg(s, \mathcal{T}(S_i)) \leq 4i$



Ordering of S_i random $\Rightarrow E[\deg(s_i, \mathcal{T}(S_i))] \leq 4$ $\square$

Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\qquad)$

Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$

Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$

More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Proof. Let $T_i(q)$ be the length of the search path of q in $\mathcal{Q}(S_i)$

Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Proof. Let $T_i(q)$ be the length of the search path of q in $\mathcal{Q}(S_i)$
Let $t_i(q)$ be the trapezoid in $\mathcal{T}(S_i)$ that contains q

Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Proof. Let $T_i(q)$ be the length of the search path of q in $\mathcal{Q}(S_i)$
Let $t_i(q)$ be the trapezoid in $\mathcal{T}(S_i)$ that contains q
$t_i(q) = t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1})$

Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Proof. Let $T_i(q)$ be the length of the search path of q in $\mathcal{Q}(S_i)$
Let $t_i(q)$ be the trapezoid in $\mathcal{T}(S_i)$ that contains q
$t_i(q) = t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1})$
$t_i(q) \neq t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1}) + \ldots$

Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

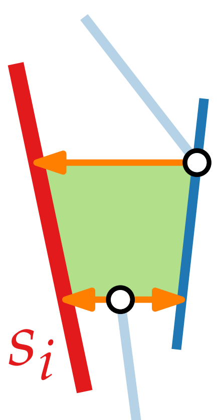
Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Proof. Let $T_i(q)$ be the length of the search path of q in $\mathcal{Q}(S_i)$
Let $t_i(q)$ be the trapezoid in $\mathcal{T}(S_i)$ that contains q

$t_i(q) = t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1})$

$t_i(q) \neq t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1}) + \ldots$

1:

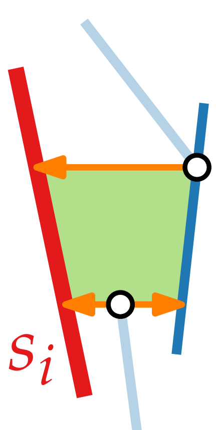


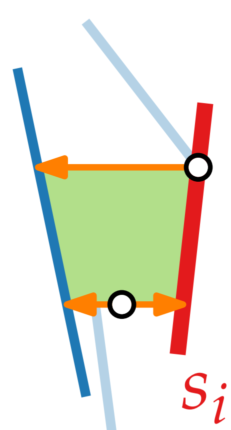
Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Proof. Let $T_i(q)$ be the length of the search path of q in $\mathcal{Q}(S_i)$
Let $t_i(q)$ be the trapezoid in $\mathcal{T}(S_i)$ that contains q
$t_i(q) = t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1})$
$t_i(q) \neq t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1}) + \ldots$

1:


1:


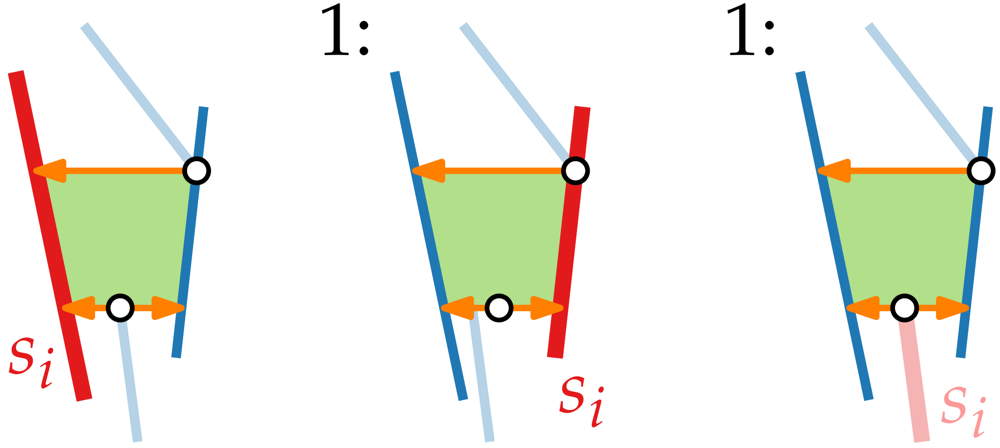
Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Proof. Let $T_i(q)$ be the length of the search path of q in $\mathcal{Q}(S_i)$
Let $t_i(q)$ be the trapezoid in $\mathcal{T}(S_i)$ that contains q
$t_i(q) = t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1})$
$t_i(q) \neq t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1}) + \ldots$

1: 1: 1:



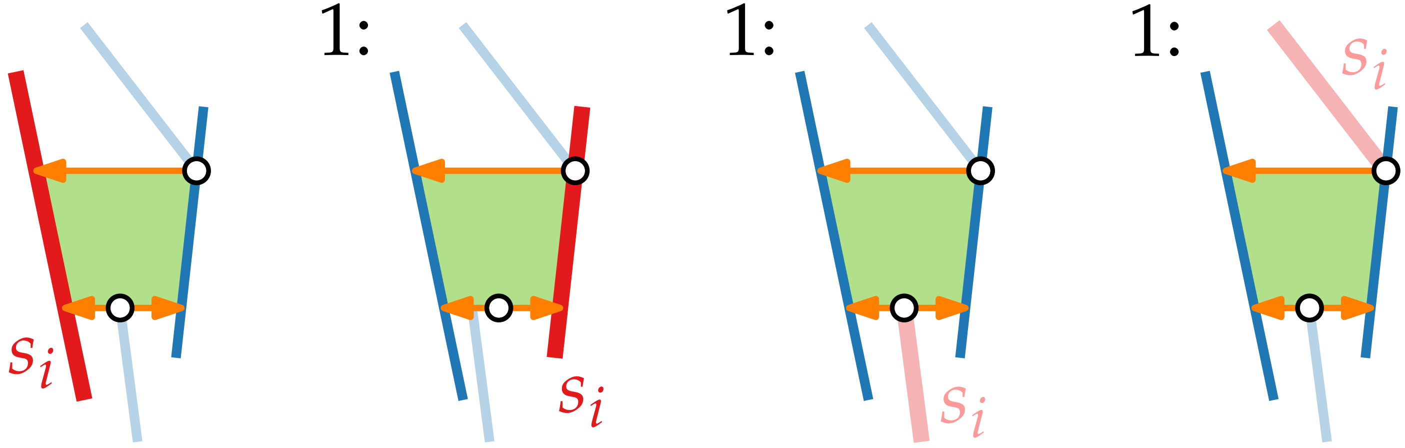
Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Proof. Let $T_i(q)$ be the length of the search path of q in $\mathcal{Q}(S_i)$
Let $t_i(q)$ be the trapezoid in $\mathcal{T}(S_i)$ that contains q
$t_i(q) = t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1})$
$t_i(q) \neq t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1}) + \ldots$

1: 1: 1: 1:

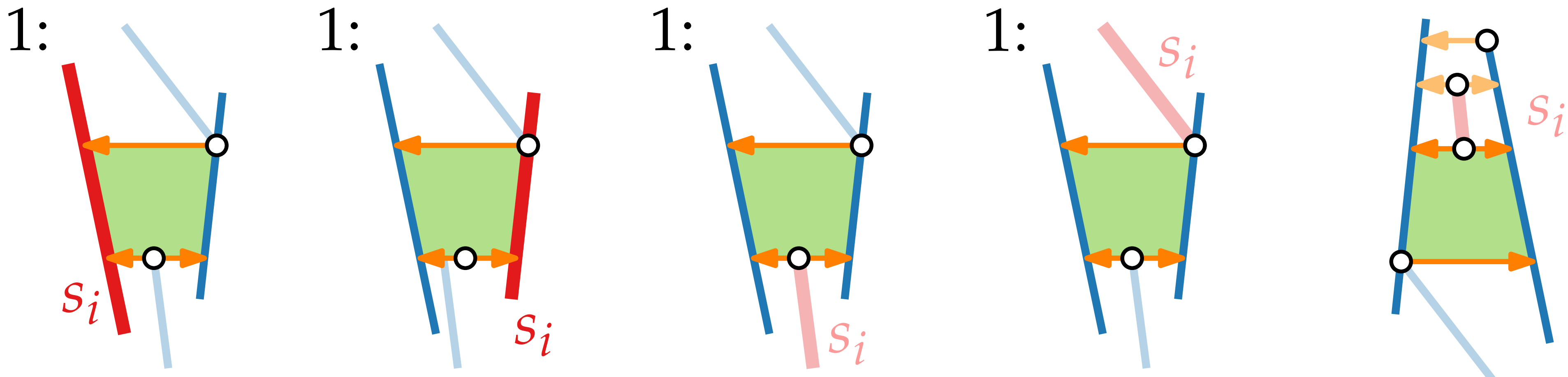


Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Proof. Let $T_i(q)$ be the length of the search path of q in $\mathcal{Q}(S_i)$
Let $t_i(q)$ be the trapezoid in $\mathcal{T}(S_i)$ that contains q
$t_i(q) = t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1})$
$t_i(q) \neq t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1}) + \ldots$

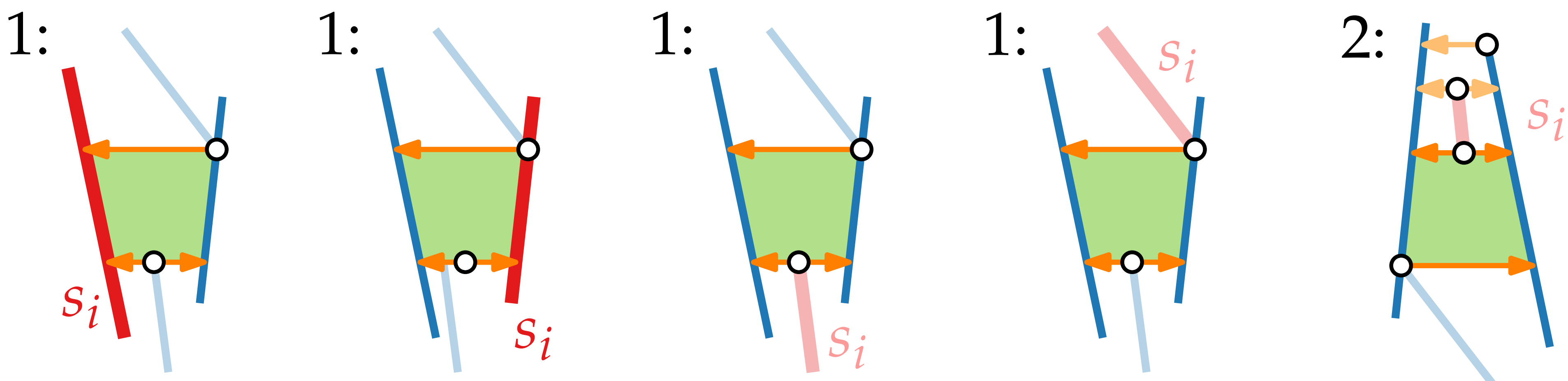


Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Proof. Let $T_i(q)$ be the length of the search path of q in $\mathcal{Q}(S_i)$
Let $t_i(q)$ be the trapezoid in $\mathcal{T}(S_i)$ that contains q
$t_i(q) = t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1})$
$t_i(q) \neq t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1}) + \ldots$

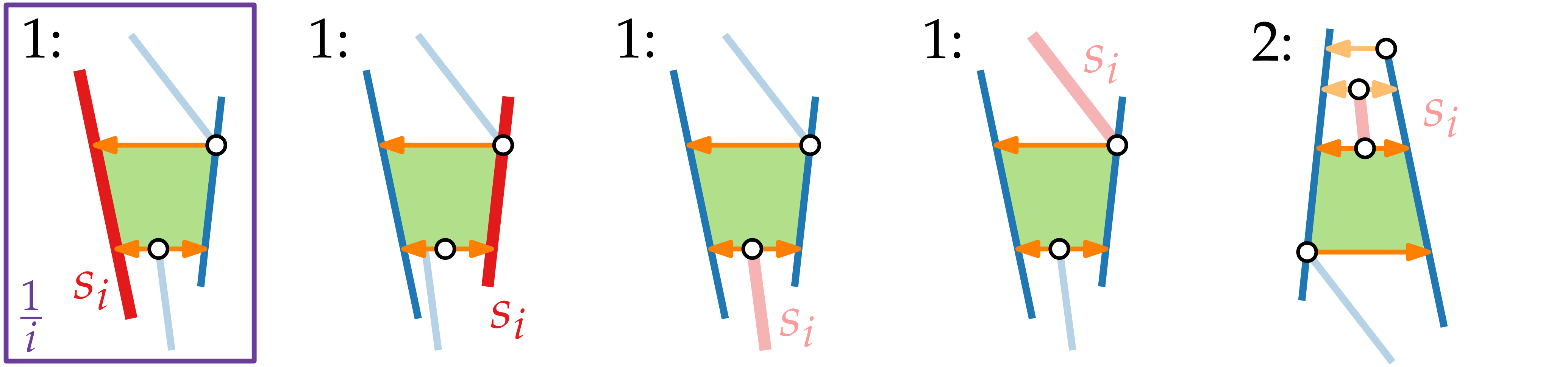


Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Proof. Let $T_i(q)$ be the length of the search path of q in $\mathcal{Q}(S_i)$
Let $t_i(q)$ be the trapezoid in $\mathcal{T}(S_i)$ that contains q
$t_i(q) = t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1})$
$t_i(q) \neq t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1}) + \ldots$

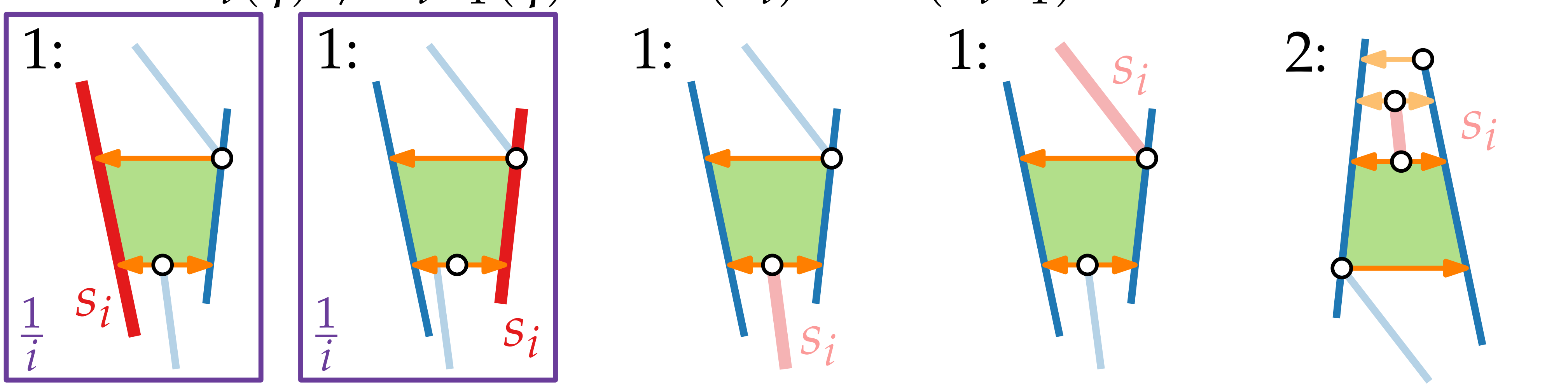


Location Time

Recall: $H_n := 1 + \frac{1}{2} + \dots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Proof. Let $T_i(q)$ be the length of the search path of q in $\mathcal{Q}(S_i)$
Let $t_i(q)$ be the trapezoid in $\mathcal{T}(S_i)$ that contains q
$t_i(q) = t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1})$
$t_i(q) \neq t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1}) + \dots$



Location Time

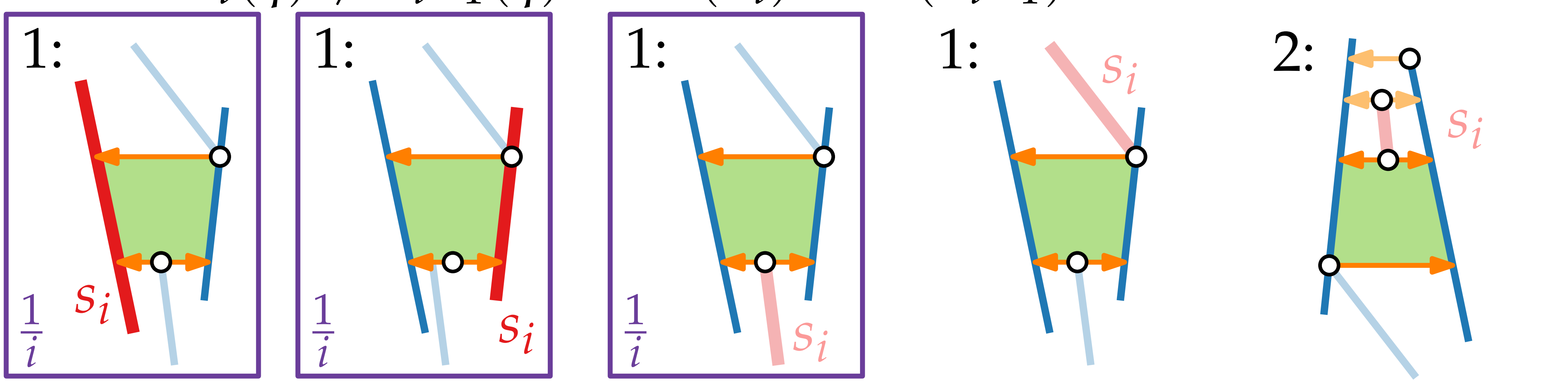
Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Proof. Let $T_i(q)$ be the length of the search path of q in $\mathcal{Q}(S_i)$
Let $t_i(q)$ be the trapezoid in $\mathcal{T}(S_i)$ that contains q

$t_i(q) = t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1})$

$t_i(q) \neq t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1}) + \ldots$

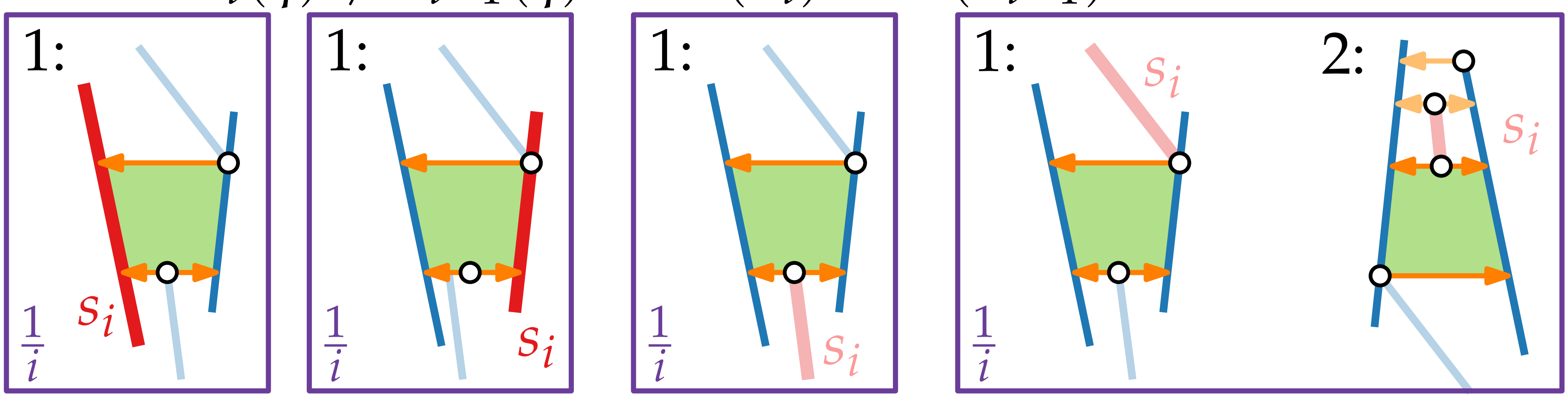


Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Proof. Let $T_i(q)$ be the length of the search path of q in $\mathcal{Q}(S_i)$
Let $t_i(q)$ be the trapezoid in $\mathcal{T}(S_i)$ that contains q
$t_i(q) = t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1})$
$t_i(q) \neq t_{i-1}(q) \Rightarrow T(S_i) = T(S_{i-1}) + \ldots$



Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$

More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Theorem. Let S be a set of n non-crossing line segments.

Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Theorem. Let S be a set of n non-crossing line segments.

- We can build $\mathcal{T}(S)$ and $\mathcal{Q}(S)$ in $O(n \log n)$ expected time.

Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Theorem. Let S be a set of n non-crossing line segments.

- We can build $\mathcal{T}(S)$ and $\mathcal{Q}(S)$ in $O(n \log n)$ expected time.
- The expected size of $\mathcal{Q}(S)$ is $O(n)$.

Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Theorem. Let S be a set of n non-crossing line segments.

- We can build $\mathcal{T}(S)$ and $\mathcal{Q}(S)$ in $O(n \log n)$ expected time.
- The expected size of $\mathcal{Q}(S)$ is $O(n)$.
- The expected time for locating a point in $\mathcal{T}(S)$ via $\mathcal{Q}(S)$ is $O(\log n)$.

Location Time

Recall: $H_n := 1 + \frac{1}{2} + \dots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Theorem. Let S be a set of n non-crossing line segments.

- We can build $\mathcal{T}(S)$ and $\mathcal{Q}(S)$ in $O(n \log n)$ expected time.
- The expected size of $\mathcal{Q}(S)$ is $O(n)$.
- The expected time for locating a point in $\mathcal{T}(S)$ via $\mathcal{Q}(S)$ is $O(\log n)$.

Location Time

Recall: $H_n := 1 + \frac{1}{2} + \cdots + \frac{1}{n} \in \Theta(\log n)$
More precisely, $\ln n < H_n < 1 + \ln n$ for $n > 1$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

Theorem. Let S be a set of n non-crossing line segments.

- We can build $\mathcal{T}(S)$ and $\mathcal{Q}(S)$ in $O(n \log n)$ expected time.
- The expected size of $\mathcal{Q}(S)$ is $O(n)$.
- The expected time for locating a point in $\mathcal{T}(S)$ via $\mathcal{Q}(S)$ is $O(\log n)$.

Aim: Speed-up construction for simple polygons.

Computational Geometry

Lecture 12:
Seidel's Triangulation Algorithm

Part III:
New Approach

Philipp Kindermann

Winter Semester 2020

New Approach

Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

New Approach

Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

Idea: Exploit polygon structure!

New Approach

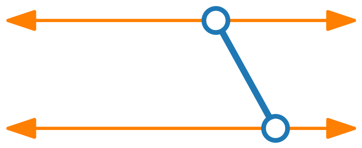
Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

Idea: Exploit polygon structure!
Locate once, then follow polygon.

New Approach

Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

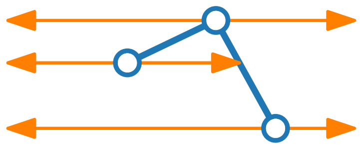
Idea: Exploit polygon structure!
Locate once, then follow polygon.



New Approach

Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

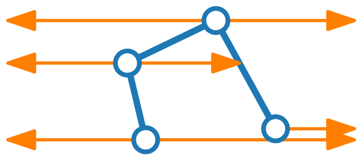
Idea: Exploit polygon structure!
Locate once, then follow polygon.



New Approach

Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

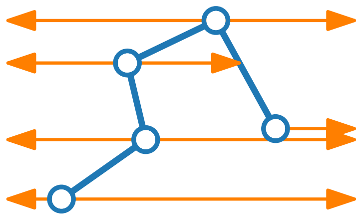
Idea: Exploit polygon structure!
Locate once, then follow polygon.



New Approach

Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

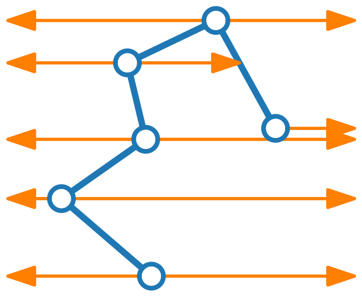
Idea: Exploit polygon structure!
Locate once, then follow polygon.



New Approach

Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

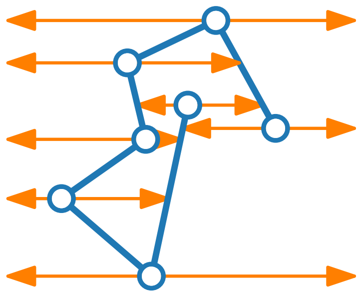
Idea: Exploit polygon structure!
Locate once, then follow polygon.



New Approach

Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

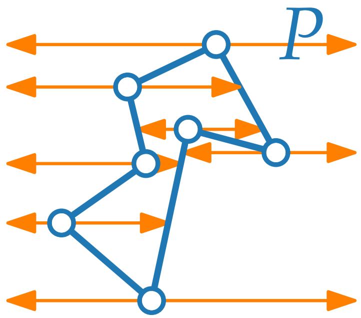
Idea: Exploit polygon structure!
Locate once, then follow polygon.



New Approach

Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

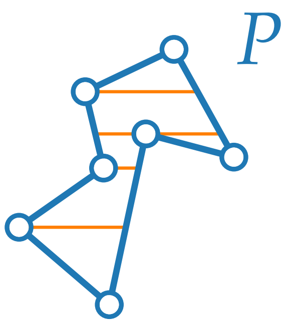
Idea: Exploit polygon structure!
Locate once, then follow polygon.



New Approach

Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

Idea: Exploit polygon structure!
Locate once, then follow polygon.

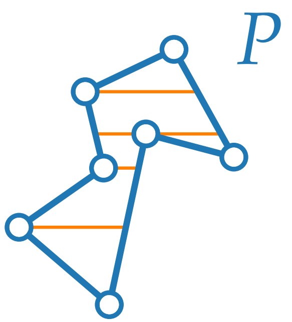


Problem: This way, we lose the random structure!

New Approach

Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

Idea: Exploit polygon structure!
Locate once, then follow polygon.

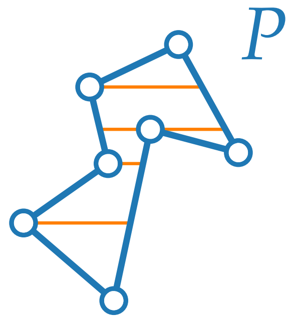


Problem: This way, we lose the random structure!
$\Rightarrow$ threading becomes more expensive

New Approach

Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

Idea: Exploit polygon structure!
Locate once, then follow polygon.

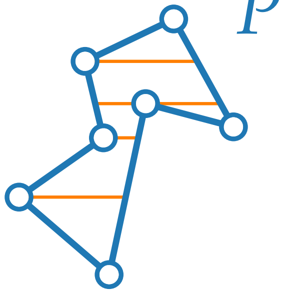


Problem: This way, we lose the random structure!
$\Rightarrow$ threading becomes more expensive
$\Rightarrow$ $\Theta(n^2)$-time algorithm :-(

New Approach

Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

Idea: Exploit polygon structure!
Locate once, then follow polygon.



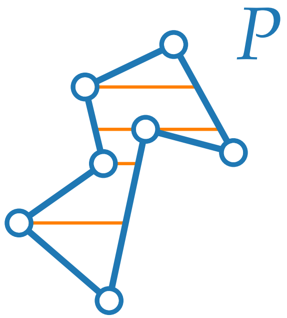
Problem: This way, we lose the random structure!
$\Rightarrow$ threading becomes more expensive
$\Rightarrow$ $\Theta(n^2)$-time algorithm :-(

Solution:

New Approach

Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

Idea: Exploit polygon structure!
Locate once, then follow polygon.



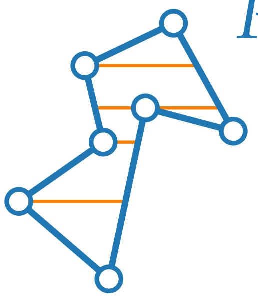
Problem: This way, we lose the random structure!
$\Rightarrow$ threading becomes more expensive
$\Rightarrow$ $\Theta(n^2)$-time algorithm :-(

Solution:
- insert segments in random order

New Approach

Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

Idea: Exploit polygon structure!
Locate once, then follow polygon.



Problem: This way, we lose the random structure!
$\Rightarrow$ threading becomes more expensive
$\Rightarrow$ $\Theta(n^2)$-time algorithm :-(

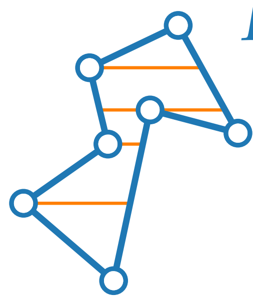
Solution:

- insert segments in random order
- every now and then, locate *all* polygon vertices in the current trapezoidation

New Approach

Observe: in $\mathcal{Q}(S_i)$,
– point location takes $O(\log i)$ expected time
– threading s_{i+1} takes $O(1)$ expected time

Idea: Exploit polygon structure!
Locate once, then follow polygon.



Problem: This way, we lose the random structure!
$\Rightarrow$ threading becomes more expensive
$\Rightarrow$ $\Theta(n^2)$-time algorithm :-(

Solution:

- insert segments in random order
- every now and then, locate *all* polygon vertices in the current trapezoidation *by walking along the polygon!*

The Two Main New Technical Ingredients

Questions:

The Two Main New Technical Ingredients

Questions:

- How much does the intermediate location information help later?

The Two Main New Technical Ingredients

Questions:

- How much does the intermediate location information help later?
- How expensive is it to walk along the polygon in the current trapezoidation?

The Two Main New Technical Ingredients

Questions:

- How much does the intermediate location information help later?
- How expensive is it to walk along the polygon in the current trapezoidation?

Lemma 4. Let $1 \leq j \leq k \leq n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time

The Two Main New Technical Ingredients

Questions:

- How much does the intermediate location information help later?
- How expensive is it to walk along the polygon in the current trapezoidation?

Lemma 4. Let $1 \leq j \leq k \leq n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

The Two Main New Technical Ingredients

Questions:

- How much does the intermediate location information help later?
- How expensive is it to walk along the polygon in the current trapezoidation?

Lemma 4. Let $1 \leq j \leq k \leq n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\qquad)$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

The Two Main New Technical Ingredients

Questions:

- How much does the intermediate location information help later?
- How expensive is it to walk along the polygon in the current trapezoidation?

Lemma 4. Let $1 \leq j \leq k \leq n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Lemma 3. For any query point q, the expected length of the search path of q in $\mathcal{Q}(S_n)$ is at most $5H_n \in O(\log n)$.

The Two Main New Technical Ingredients

Questions:

- How much does the intermediate location information help later?
- How expensive is it to walk along the polygon in the current trapezoidation?

Lemma 4. Let $1 \leq j \leq k \leq n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Lemma 5. S as before, $R \subseteq S$ random subset, $r := |R|$.

The Two Main New Technical Ingredients

Questions:

- How much does the intermediate location information help later?
- How expensive is it to walk along the polygon in the current trapezoidation?

Lemma 4. Let $1 \leq j \leq k \leq n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Lemma 5. S as before, $R \subseteq S$ random subset, $r := |R|$.
Let I be the number of intersections between rays of $\mathcal{T}(R)$ and segments in $S \setminus R$.

The Two Main New Technical Ingredients

Questions:

- How much does the intermediate location information help later?
- How expensive is it to walk along the polygon in the current trapezoidation?

Lemma 4. Let $1 \leq j \leq k \leq n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Lemma 5. S as before, $R \subseteq S$ random subset, $r := |R|$. Let I be the number of intersections between rays of $\mathcal{T}(R)$ and segments in $S \setminus R$. Then $E[I] \leq$, where the expectation is over all size-r subsets of S.

The Two Main New Technical Ingredients

Questions:

- How much does the intermediate location information help later?
- polygon in the current trapezoidation?

Lemma 2. For $i = 1, \dots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Lemma 4. Let $1 \le j \le k \le n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Lemma 5. S as before, $R \subseteq S$ random subset, $r := |R|$. Let I be the number of intersections between rays of $\mathcal{T}(R)$ and segments in $S \setminus R$. Then $E[I] \le$, where the expectation is over all size-r subsets of S.

The Two Main New Technical Ingredients

Questions:
- How much does the intermediate location information help later?

  polygon in the current trapezoidation?

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Lemma 4. Let $1 \leq j \leq k \leq n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Lemma 5. S as before, $R \subseteq S$ random subset, $r := |R|$. Let I be the number of intersections between rays of $\mathcal{T}(R)$ and segments in $S \setminus R$. Then $E[I] \leq 4(n - r)$, where the expectation is over all size-r subsets of S.

The Two Main New Technical Ingredients

Questions:

- How much does the intermediate location information help later?
- How expensive is it to walk along the polygon in the current trapezoidation?

Lemma 4. Let $1 \leq j \leq k \leq n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Lemma 5. S as before, $R \subseteq S$ random subset, $r := |R|$. Let I be the number of intersections between rays of $\mathcal{T}(R)$ and segments in $S \setminus R$. Then $E[I] \leq 4(n-r)$, where the expectation is over all size-r subsets of S.

Computational Geometry

Lecture 12:
Seidel's Triangulation Algorithm

Part IV:
The Algorithm

Philipp Kindermann

Winter Semester 2020

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

Examples. $\log^{(0)} 2^{2^{2^2}} =$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

Examples. $\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

Examples. $\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$

$\log^{(1)} 2^{2^{2^2}} =$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

Examples. $\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$

$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} =$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

Examples. $\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$

$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

Examples. $\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$

$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$

$\log^{(2)} 2^{2^{2^2}} =$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

Examples.

$$\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$$

$$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$$

$$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} =$$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

Examples.

$$\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$$

$$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$$

$$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2$$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

Examples.

$$\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$$

$$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$$

$$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2$$

$$\log^{(3)} 2^{2^{2^2}} =$$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

Examples.

$$\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$$

$$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$$

$$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2$$

$$\log^{(3)} 2^{2^{2^2}} = 2;$$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

Examples. $\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$

$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$

$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2$

$\log^{(3)} 2^{2^{2^2}} = 2; \quad \log^{(4)} 2^{2^{2^2}} =$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

Examples.

$$\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$$

$$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$$

$$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2$$

$$\log^{(3)} 2^{2^{2^2}} = 2; \quad \log^{(4)} 2^{2^{2^2}} = 1$$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

For $n > 0$, let $\log^{\star} n := \max\{i \mid \log^{(i)} n \geq 1\}$.

Examples.

$$\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$$

$$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$$

$$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2$$

$$\log^{(3)} 2^{2^{2^2}} = 2; \quad \log^{(4)} 2^{2^{2^2}} = 1$$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

For $n > 0$, let $\log^{\star} n := \max\{i \mid \log^{(i)} n \geq 1\}$.

Examples.

$$\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$$

$$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$$

$$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2$$

$$\log^{(3)} 2^{2^{2^2}} = 2; \quad \log^{(4)} 2^{2^{2^2}} = 1 \;\Rightarrow\; \log^{\star} 2^{2^{2^2}} =$$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

For $n > 0$, let $\log^\star n := \max\{i \mid \log^{(i)} n \geq 1\}$.

Examples.

$$\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$$

$$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$$

$$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2$$

$$\log^{(3)} 2^{2^{2^2}} = 2; \quad \log^{(4)} 2^{2^{2^2}} = 1 \quad \Rightarrow \log^\star 2^{2^{2^2}} = 4$$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

For $n > 0$, let $\log^\star n := \max\{i \mid \log^{(i)} n \geq 1\}$.

Examples.

$$\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$$

$$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$$

$$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2$$

$$\log^{(3)} 2^{2^{2^2}} = 2; \quad \log^{(4)} 2^{2^{2^2}} = 1 \quad \Rightarrow \log^\star \underbrace{2^{2^{2^2}}}_{65{,}536} = 4$$

Logs All Over the Place

Input:

$2^{2^{2^{2^2}}}$

Open code

Result:

2 003 529 930 406 846 464 979 072 351 560 255 750 447 825 475 569 751 419 265 016 ⋱
973 710 894 059 556 311 453 089 506 130 880 933 348 101 038 234 342 907 263 ⋱
181 822 949 382 118 812 668 869 506 364 761 547 029 165 041 871 916 351 587 ⋱
966 347 219 442 930 927 982 084 309 104 855 990 570 159 318 959 639 524 863 ⋱
372 367 203 002 916 969 592 156 108 764 948 889 254 090 805 911 457 037 675 ⋱
208 500 206 671 563 702 366 126 359 747 144 807 111 774 815 880 914 135 742 ⋱
720 967 190 151 836 282 560 618 091 458 852 699 826 141 425 030 123 391...

More digits

Decimal approximation:

$2.0035299304068464649790723515602557504478254755697 5\ldots \times 10^{19728}$

More digits

Number length:

19 729 decimal digits

n of n be defined by

if $i = 0$,

$n)$ if $i > 0$.

$x\{i \mid \log^{(i)} n \geq 1\}$.

$$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2$$

65,536

$$\log^{(3)} 2^{2^{2^2}} = 2; \quad \log^{(4)} 2^{2^{2^2}} = 1 \quad \Rightarrow \log^{\star} 2^{2^{2^2}} = 4$$

Logs All Over the Place

Input:

$2^{2^{2^{2^2}}}$

Open code

Result:

More digits

2 003 529 930 406 846 464 979 072 351 560 255 750 447 825 475 569 751 419 265 016 ⋱
973 710 894 059 556 311 453 089 506 130 880 933 348 101 038 234 342 907 263 ⋱
181 822 949 382 118 812 668 869 506 364 761 547 029 165 041 871 916 351 587 ⋱
966 347 219 442 930 927 982 084 309 104 855 990 570 159 318 959 639 524 863 ⋱
372 367 203 002 916 969 592 156 108 764 948 889 254 090 805 911 457 037 675 ⋱
208 500 206 671 563 702 366 126 359 747 144 807 111 774 815 880 914 135 742 ⋱
720 967 190 151 836 282 560 618 091 458 852 699 826 141 425 030 123 391…

Decimal approximation:

More digits

$2.0035299304068464649790723515602557504478254755 6975\ldots \times 10^{19728}$

Number length:

19 729 decimal digits

Input interpretation:

estimated number of atoms in the universe

Result:

1×10^{80} atoms

$$\text{x}\{i \mid \log^{(i)} n \geq 1\}.$$

$$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2$$

$$\log^{(3)} 2^{2^{2^2}} = 2; \quad \log^{(4)} 2^{2^{2^2}} = 1 \quad \Rightarrow \log^{\star} 2^{2^{2^2}} = 4$$

65,536

Logs All Over the Place

Input:

$2^{2^{2^{2^{2}}}}$

Open code

Result: More digits

2 003 529 930 406 846 464 979 072 351 560 255 750 447 825 475 569 751 419 265 016 ⋱
973 710 894 059 556 311 453 089 506 130 880 933 348 101 038 234 342 907 263 ⋱
181 822 949 382 118 812 668 869 506 364 761 547 029 165 041 871 916 351 587 ⋱
966 347 219 442 930 927 982 084 309 104 855 990 570 159 318 959 639 524 863 ⋱
372 367 203 002 916 969 592 156 108 764 948 889 254 090 805 911 457 037 675 ⋱
208 500 206 671 563 702 366 126 359 747 144 807 111 774 815 880 914 135 742 ⋱
720 967 190 151 836 282 560 618 091 458 852 699 826 141 425 030 123 391…

Decimal approximation: More digits

$2.0035299304068464649790723515602557504478254755697 5\ldots \times 10^{19728}$

Number length:

19 729 decimal digits

Input interpretation:

estimated number of atoms in the universe

Result:

1×10^{80} atoms

Input interpretation:

$$\frac{2^{2^{2^{2^{2}}}}}{\text{estimated number of atoms in the universe}}$$

Result:

2×10^{19648} per atom

$$\log^{(2)} 2^{2^{2^{2}}} = \log_2 \log^{(1)} 2^{2^{2^{2}}} = 2^2$$

$$\log^{(3)} 2^{2^{2^{2}}} = 2; \quad \log^{(4)} 2^{2^{2^{2}}} = 1 \quad \Rightarrow \log^{\star} 2^{2^{2^{2}}} = 4$$

65,536

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

For $n > 0$, let $\log^{\star} n := \max\{i \mid \log^{(i)} n \geq 1\}$.

For $0 \leq h \leq \log^{\star} n$, let $N(h) := \lceil n / \log^{(h)} n \rceil$.

Examples.

$$\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$$

$$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$$

$$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2$$

$$\log^{(3)} 2^{2^{2^2}} = 2; \quad \log^{(4)} 2^{2^{2^2}} = 1 \quad \Rightarrow \log^{\star} 2^{2^{2^2}} = 4$$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

For $n > 0$, let $\log^\star n := \max\{i \mid \log^{(i)} n \geq 1\}$.

For $0 \leq h \leq \log^\star n$, let $N(h) := \lceil n / \log^{(h)} n \rceil$.

Examples.

$$\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$$

$$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$$

$$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2$$

$$\log^{(3)} 2^{2^{2^2}} = 2; \quad \log^{(4)} 2^{2^{2^2}} = 1 \quad \Rightarrow \log^\star 2^{2^{2^2}} = 4$$

$$N(0) = 1$$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

For $n > 0$, let $\log^\star n := \max\{i \mid \log^{(i)} n \geq 1\}$.

For $0 \leq h \leq \log^\star n$, let $N(h) := \lceil n / \log^{(h)} n \rceil$.

Examples.

$$\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$$

$$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$$

$$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2$$

$$\log^{(3)} 2^{2^{2^2}} = 2; \quad \log^{(4)} 2^{2^{2^2}} = 1 \quad \Rightarrow \log^\star 2^{2^{2^2}} = 4$$

$$N(0) = 1,\ N(1) = \lceil n / \log n \rceil$$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

For $n > 0$, let $\log^\star n := \max\{i \mid \log^{(i)} n \geq 1\}$.

For $0 \leq h \leq \log^\star n$, let $N(h) := \lceil n / \log^{(h)} n \rceil$.

Examples.

$$\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$$

$$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$$

$$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2$$

$$\log^{(3)} 2^{2^{2^2}} = 2; \quad \log^{(4)} 2^{2^{2^2}} = 1 \quad \Rightarrow \log^\star 2^{2^{2^2}} = 4$$

$$N(0) = 1, N(1) = \lceil n / \log n \rceil, \ldots$$

Logs All Over the Place

Definition. Let the *i-th iterated logarithm* of n be defined by

$$\log^{(i)} n := \begin{cases} n & \text{if } i = 0, \\ \log_2(\log^{(i-1)} n) & \text{if } i > 0. \end{cases}$$

For $n > 0$, let $\log^\star n := \max\{i \mid \log^{(i)} n \geq 1\}$.

For $0 \leq h \leq \log^\star n$, let $N(h) := \lceil n / \log^{(h)} n \rceil$.

Examples.

$$\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$$

$$\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} = 2^{2^2}$$

$$\log^{(2)} 2^{2^{2^2}} = \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2$$

$$\log^{(3)} 2^{2^{2^2}} = 2; \quad \log^{(4)} 2^{2^{2^2}} = 1 \quad \Rightarrow \log^\star 2^{2^{2^2}} = 4$$

$$N(0) = 1, N(1) = \lceil n / \log n \rceil, N(\log^\star n) > n/2.$$

The Algorithm

PolygonTrapezoidation ((edges along) simple polygon P)

The Algorithm

PolygonTrapezoidation ((edges along) simple polygon P)

1. $\langle s_1, s_2, \ldots, s_n \rangle :=$ random ordering of the edges of P

The Algorithm

PolygonTrapezoidation ((edges along) simple polygon P)

1. $\langle s_1, s_2, \ldots, s_n \rangle :=$ random ordering of the edges of P
2. Compute $\mathcal{T}_1$ and $\mathcal{Q}_1$ for $\{s_1\}$.

The Algorithm

PolygonTrapezoidation ((edges along) simple polygon P)

1. $\langle s_1, s_2, \ldots, s_n \rangle :=$ random ordering of the edges of P
2. Compute $\mathcal{T}_1$ and $\mathcal{Q}_1$ for $\{s_1\}$.
   foreach $v \in P$ **do** $\pi(v) \leftarrow$ ptr to the leaf of $\mathcal{Q}_1$ that contains v.

The Algorithm

PolygonTrapezoidation ((edges along) simple polygon P)

1. $\langle s_1, s_2, \ldots, s_n \rangle :=$ random ordering of the edges of P
2. Compute $\mathcal{T}_1$ and $\mathcal{Q}_1$ for $\{s_1\}$.
   foreach $v \in P$ **do** $\pi(v) \leftarrow$ ptr to the leaf of $\mathcal{Q}_1$ that contains v.
   for $h = 1$ **to** $\log^\star n$ **do** // phase h

3.1

3.2

The Algorithm

PolygonTrapezoidation ((edges along) simple polygon P)

1. $\langle s_1, s_2, \ldots, s_n \rangle :=$ random ordering of the edges of P
2. Compute $\mathcal{T}_1$ and $\mathcal{Q}_1$ for $\{s_1\}$.

foreach $v \in P$ **do** $\pi(v) \leftarrow$ ptr to the leaf of $\mathcal{Q}_1$ that contains v.

for $h = 1$ **to** $\log^\star n$ **do** // phase h

3.1 **for** $i = N(h-1) + 1$ **to** $N(h)$ **do**

3.2

The Algorithm

PolygonTrapezoidation ((edges along) simple polygon P)

1. $\langle s_1, s_2, \ldots, s_n \rangle :=$ random ordering of the edges of P
2. Compute $\mathcal{T}_1$ and $\mathcal{Q}_1$ for $\{s_1\}$.

foreach $v \in P$ **do** $\pi(v) \leftarrow$ ptr to the leaf of $\mathcal{Q}_1$ that contains v.

for $h = 1$ **to** $\log^\star n$ **do** // phase h

3.1 **for** $i = N(h-1)+1$ **to** $N(h)$ **do**

insert $s_i = v_i w_i$ in $\mathcal{T}_{i-1}$ using $\pi(v_i)$ (node in $\mathcal{Q}_{N(h-1)}$)

3.2

The Algorithm

PolygonTrapezoidation ((edges along) simple polygon P)

1. $\langle s_1, s_2, \ldots, s_n \rangle :=$ random ordering of the edges of P
2. Compute $\mathcal{T}_1$ and $\mathcal{Q}_1$ for $\{s_1\}$.

foreach $v \in P$ **do** $\pi(v) \leftarrow$ ptr to the leaf of $\mathcal{Q}_1$ that contains v.

for $h = 1$ **to** $\log^\star n$ **do** // phase h

3.1 **for** $i = N(h-1)+1$ **to** $N(h)$ **do**

insert $s_i = v_i w_i$ in $\mathcal{T}_{i-1}$ using $\pi(v_i)$ (node in $\mathcal{Q}_{N(h-1)}$)

3.2 walk along P through $\mathcal{T}_{N(h)}$:

The Algorithm

PolygonTrapezoidation ((edges along) simple polygon P)

1. $\langle s_1, s_2, \ldots, s_n \rangle :=$ random ordering of the edges of P
2. Compute $\mathcal{T}_1$ and $\mathcal{Q}_1$ for $\{s_1\}$.

foreach $v \in P$ **do** $\pi(v) \leftarrow$ ptr to the leaf of $\mathcal{Q}_1$ that contains v.

for $h = 1$ **to** $\log^\star n$ **do** // phase h

3.1 **for** $i = N(h-1)+1$ **to** $N(h)$ **do**

insert $s_i = v_i w_i$ in $\mathcal{T}_{i-1}$ using $\pi(v_i)$ (node in $\mathcal{Q}_{N(h-1)}$)

3.2 walk along P through $\mathcal{T}_{N(h)}$:

foreach vertex v **do**

The Algorithm

PolygonTrapezoidation ((edges along) simple polygon P)

1. $\langle s_1, s_2, \ldots, s_n \rangle :=$ random ordering of the edges of P
2. Compute $\mathcal{T}_1$ and $\mathcal{Q}_1$ for $\{s_1\}$.

foreach $v \in P$ **do** $\pi(v) \leftarrow$ ptr to the leaf of $\mathcal{Q}_1$ that contains v.

for $h = 1$ **to** $\log^\star n$ **do** // phase h

3.1 **for** $i = N(h-1)+1$ **to** $N(h)$ **do**

insert $s_i = v_i w_i$ in $\mathcal{T}_{i-1}$ using $\pi(v_i)$ (node in $\mathcal{Q}_{N(h-1)}$)

3.2 walk along P through $\mathcal{T}_{N(h)}$:

foreach vertex v **do**

$\Delta \leftarrow$ the trapezoid in $\mathcal{T}_{N(h)}$ that contains v

The Algorithm

PolygonTrapezoidation ((edges along) simple polygon P)

1. $\langle s_1, s_2, \ldots, s_n \rangle :=$ random ordering of the edges of P
2. Compute $\mathcal{T}_1$ and $\mathcal{Q}_1$ for $\{s_1\}$.

foreach $v \in P$ **do** $\pi(v) \leftarrow$ ptr to the leaf of $\mathcal{Q}_1$ that contains v.

for $h = 1$ **to** $\log^\star n$ **do** // phase h

3.1 **for** $i = N(h-1)+1$ **to** $N(h)$ **do**

insert $s_i = v_i w_i$ in $\mathcal{T}_{i-1}$ using $\pi(v_i)$ (node in $\mathcal{Q}_{N(h-1)}$)

3.2 walk along P through $\mathcal{T}_{N(h)}$:

foreach vertex v **do**

$\Delta \leftarrow$ the trapezoid in $\mathcal{T}_{N(h)}$ that contains v

$\pi(v) \leftarrow$ the node in $\mathcal{Q}_{N(h)}$ corresponding to Δ

The Algorithm

PolygonTrapezoidation ((edges along) simple polygon P)

1. $\langle s_1, s_2, \ldots, s_n \rangle :=$ random ordering of the edges of P
2. Compute $\mathcal{T}_1$ and $\mathcal{Q}_1$ for $\{s_1\}$.

foreach $v \in P$ **do** $\pi(v) \leftarrow$ ptr to the leaf of $\mathcal{Q}_1$ that contains v.

for $h = 1$ **to** $\log^\star n$ **do** // phase h

3.1 **for** $i = N(h-1) + 1$ **to** $N(h)$ **do**

insert $s_i = v_i w_i$ in $\mathcal{T}_{i-1}$ using $\pi(v_i)$ (node in $\mathcal{Q}_{N(h-1)}$)

3.2 walk along P through $\mathcal{T}_{N(h)}$:

foreach vertex v **do**

$\Delta \leftarrow$ the trapezoid in $\mathcal{T}_{N(h)}$ that contains v

$\pi(v) \leftarrow$ the node in $\mathcal{Q}_{N(h)}$ corresponding to Δ

4. **for** $i = N(\log^\star n) + 1$ **to** n **do**

The Algorithm

PolygonTrapezoidation ((edges along) simple polygon P)

1. $\langle s_1, s_2, \ldots, s_n \rangle :=$ random ordering of the edges of P
2. Compute $\mathcal{T}_1$ and $\mathcal{Q}_1$ for $\{s_1\}$.

   foreach $v \in P$ **do** $\pi(v) \leftarrow$ ptr to the leaf of $\mathcal{Q}_1$ that contains v.

   for $h = 1$ **to** $\log^\star n$ **do** // phase h

3.1 **for** $i = N(h-1) + 1$ **to** $N(h)$ **do**

   insert $s_i = v_i w_i$ in $\mathcal{T}_{i-1}$ using $\pi(v_i)$ (node in $\mathcal{Q}_{N(h-1)}$)

3.2 walk along P through $\mathcal{T}_{N(h)}$:

   foreach vertex v **do**

   $\Delta \leftarrow$ the trapezoid in $\mathcal{T}_{N(h)}$ that contains v

   $\pi(v) \leftarrow$ the node in $\mathcal{Q}_{N(h)}$ corresponding to Δ

4. **for** $i = N(\log^\star n) + 1$ **to** n **do**

   insert $s_i = v_i w_i$ in $\mathcal{T}_{i-1}$ using $\pi(v_i)$ (node in $\mathcal{Q}_{N(\log^\star n)}$)

The Algorithm

PolygonTrapezoidation ((edges along) simple polygon P)

1. $\langle s_1, s_2, \ldots, s_n \rangle :=$ random ordering of the edges of P
2. Compute $\mathcal{T}_1$ and $\mathcal{Q}_1$ for $\{s_1\}$.

foreach $v \in P$ **do** $\pi(v) \leftarrow$ ptr to the leaf of $\mathcal{Q}_1$ that contains v.

for $h = 1$ **to** $\log^\star n$ **do** // phase h

3.1 **for** $i = N(h-1) + 1$ **to** $N(h)$ **do**

insert $s_i = v_i w_i$ in $\mathcal{T}_{i-1}$ using $\pi(v_i)$ (node in $\mathcal{Q}_{N(h-1)}$)

3.2 walk along P through $\mathcal{T}_{N(h)}$:

foreach vertex v **do**

$\Delta \leftarrow$ the trapezoid in $\mathcal{T}_{N(h)}$ that contains v

$\pi(v) \leftarrow$ the node in $\mathcal{Q}_{N(h)}$ corresponding to Δ

4. **for** $i = N(\log^\star n) + 1$ **to** n **do**

insert $s_i = v_i w_i$ in $\mathcal{T}_{i-1}$ using $\pi(v_i)$ (node in $\mathcal{Q}_{N(\log^\star n)}$)

return $(\mathcal{T}_n, \mathcal{Q}_n)$

Computational Geometry

Lecture 12:
Seidel's Triangulation Algorithm

Part V:
Time Complexity

Philipp Kindermann

Winter Semester 2020

Time Complexity

Step 1: Random permutation

Time Complexity

Step 1: Random permutation $O(n)$

Time Complexity

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$

Time Complexity

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Time Complexity

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$

Time Complexity

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Time Complexity

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon

Time Complexity

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$

Lemma 5. S as before, $R \subseteq S$ random subset, $r := |R|$. Let I be the number of intersections between rays of $\mathcal{T}(R)$ and segments in $S \setminus R$. Then $E[I] \leq 4(n - r)$, where the expectation is over all size-r subsets of S.

Time Complexity

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Lemma 5. S as before, $R \subseteq S$ random subset, $r := |R|$. Let I be the number of intersections between rays of $\mathcal{T}(R)$ and segments in $S \setminus R$. Then $E[I] \leq 4(n-r)$, where the expectation is over all size-r subsets of S.

Time Complexity

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

Time Complexity

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost:

– locating cost:

Time Complexity

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost:

– locating cost:

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Time Complexity

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost:Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

– locating cost:

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Time Complexity

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost:Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

– locating cost: Know the location of v_i in $\mathcal{Q}_{N(h-1)}$.

Time Complexity

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost: Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

– locating cost: Know the location of v_i in $\mathcal{Q}_{N(h-1)}$.

Lem. 4 $\Rightarrow$ expected location cost

Lemma 4. Let $1 \le j \le k \le n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Time Complexity

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost: Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

– locating cost: Know the location of v_i in $\mathcal{Q}_{N(h-1)}$.

Lem. 4 $\Rightarrow$ expected location cost

$O(\log(i/N(h-1))) \subseteq$

Lemma 4. Let $1 \leq j \leq k \leq n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost: Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

– locating cost: Know the location of v_i in $\mathcal{Q}_{N(h-1)}$.

Lem. 4 $\Rightarrow$ expected location cost

$O(\log(i/N(h-1))) \subseteq$

Lemma 4. Let $1 \le j \le k \le n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost: Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

– locating cost: Know the location of v_i in $\mathcal{Q}_{N(h-1)}$.

Lem. 4 $\Rightarrow$ expected location cost

$O(\log(i/N(h-1))) \subseteq O(\log^{(h)} n)$

Lemma 4. Let $1 \leq j \leq k \leq n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost: Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

– locating cost: Know the location of v_i in $\mathcal{Q}_{N(h-1)}$.

Lem. 4 $\Rightarrow$ expected location cost $O(\log(i/N(h-1))) \subseteq O(\log^{(h)} n)$

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost: Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

– locating cost: Know the location of v_i in $\mathcal{Q}_{N(h-1)}$

Lem. 4 $\Rightarrow$ expected location cost

$O(\log(i/N(h-1))) \subseteq O(\log^{(h)} n)$

$\Big\} \cdot N(h) =$

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost: Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

– locating cost: Know the location of v_i in $\mathcal{Q}_{N(h-1)}$

Lem. 4 $\Rightarrow$ expected location cost

$O(\log(i/N(h-1))) \subseteq O(\log^{(h)} n)$

$\Big\} \cdot N(h) = O(n)$

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost: Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

– locating cost: Know the location of v_i in $\mathcal{Q}_{N(h-1)}$

Lem. 4 $\Rightarrow$ expected location cost

$O(\log(i/N(h-1))) \subseteq O(\log^{(h)} n)$

$\Bigg\} \cdot N(h) = O(n)$

Step 4: Inserting s_i (for $N(\log^\star n) < i \leq n$) using $\mathcal{Q}_{N(\log^\star n)}$

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost: Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

– locating cost: Know the location of v_i in $\mathcal{Q}_{N(h-1)}$

Lem. 4 $\Rightarrow$ expected location cost

$O(\log(i/N(h-1))) \subseteq O(\log^{(h)} n)$

$\Big\} \cdot N(h) = O(n)$

Step 4: Inserting s_i (for $N(\log^\star n) < i \leq n$) using $\mathcal{Q}_{N(\log^\star n)}$

– threading cost:

– locating cost:

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost:Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

– locating cost: Know the location of v_i in $\mathcal{Q}_{N(h-1)}$ $\cdot$ $N(h)$ $O(n)$

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Step 4: Inserting s_i (for $N(\log^\star n) < i \leq n$) using $\mathcal{Q}_{N(\log^\star n)}$

– threading cost:

– locating cost:

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost:Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

– locating cost: Know the location of v_i in $\mathcal{Q}_{N(h-1)}$ $\cdot$ $N(h)$ $O(n)$

Lemma 2. For $i = 1, \ldots, n$, the expected number of rays of $\mathcal{T}(S_{i-1})$ that are intersected by s_i is at most 4.

Step 4: Inserting s_i (for $N(\log^\star n) < i \leq n$) using $\mathcal{Q}_{N(\log^\star n)}$

– threading cost:Lem. 2 $\Rightarrow$ $O(1)$

– locating cost:

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost:Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

Lemma 4. Let $1 \leq j \leq k \leq n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Step 4: Inserting s_i (for $N(\log^\star n) < i \leq n$) using $\mathcal{Q}_{N(\log^\star n)}$

– threading cost:Lem. 2 $\Rightarrow$ $O(1)$

– locating cost: Lem. 4 $\Rightarrow$

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost:Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

Lemma 4. Let $1 \le j \le k \le n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Step 4: Inserting s_i (for $N(\log^\star n) < i \le n$) using $\mathcal{Q}_{N(\log^\star n)}$

– threading cost:Lem. 2 $\Rightarrow O(1)$

– locating cost: Lem. 4 $\Rightarrow O(\log n / N(\log^\star n)) =$

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost:Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

Lemma 4. Let $1 \leq j \leq k \leq n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Step 4: Inserting s_i (for $N(\log^\star n) < i \leq n$) using $\mathcal{Q}_{N(\log^\star n)}$

– threading cost:Lem. 2 $\Rightarrow$ $O(1)$

– locating cost: Lem. 4 $\Rightarrow$ $O(\log n / \underbrace{N(\log^\star n)}) =$

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost:Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

Lemma 4. Let $1 \leq j \leq k \leq n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Step 4: Inserting s_i (for $N(\log^\star n) < i \leq n$) using $\mathcal{Q}_{N(\log^\star n)}$

– threading cost:Lem. 2 $\Rightarrow O(1)$

– locating cost: Lem. 4 $\Rightarrow O(\log n / \underbrace{N(\log^\star n)}_{> n/2}) =$

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost:Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

Lemma 4. Let $1 \le j \le k \le n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Step 4: Inserting s_i (for $N(\log^\star n) < i \le n$) using $\mathcal{Q}_{N(\log^\star n)}$

– threading cost:Lem. 2 $\Rightarrow O(1)$

– locating cost: Lem. 4 $\Rightarrow O(\log n / \underbrace{N(\log^\star n)}_{> n/2}) = O(1)$

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost:Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

Lemma 4. Let $1 \leq j \leq k \leq n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Step 4: Inserting s_i (for $N(\log^\star n) < i \leq n$) using $\mathcal{Q}_{N(\log^\star n)}$

– threading cost:Lem. 2 $\Rightarrow O(1)$

– locating cost: Lem. 4 $\Rightarrow O(\log n / \underbrace{N(\log^\star n)}_{> n/2}) = O(1)$

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost:Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

Lemma 4. Let $1 \le j \le k \le n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Step 4: Inserting s_i (for $N(\log^\star n) < i \le n$) using $\mathcal{Q}_{N(\log^\star n)}$

– threading cost:Lem. 2 $\Rightarrow$ $O(1)$

– locating cost: Lem. 4 $\Rightarrow$ $O(\log n / \underbrace{N(\log^\star n)}_{> n/2}) = O(1)$

$\Big\} \cdot O(n) =$

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost:Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

Lemma 4. Let $1 \le j \le k \le n$ and $q \in \mathbb{R}^2$. Suppose location of q in $\mathcal{Q}(S_j)$ is known, then q can be located in $\mathcal{Q}(S_k)$ in expected time $5(H_k - H_j) \in O(\log k/j)$.

Step 4: Inserting s_i (for $N(\log^\star n) < i \le n$) using $\mathcal{Q}_{N(\log^\star n)}$ $O(n)$

– threading cost:Lem. 2 $\Rightarrow O(1)$

– locating cost: Lem. 4 $\Rightarrow O(\log n / \underbrace{N(\log^\star n)}_{> n/2}) = O(1)$

$\Big\} \cdot O(n) = O(n)$

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ $O(n)$

Step 3: Phases 1 to $\log^\star n$ $(\log^\star n) \cdot$

Step 3.2: Walking the polygon Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost: Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

– locating cost: Know the location of v_i in $\mathcal{Q}_{N(h-1)}$

Lem. 4 $\Rightarrow$ expected location cost $O(\log(i / N(h-1))) \subseteq O(\log^{(h)} n)$

$\cdot N(h) = O(n)$

Step 4: Inserting s_i (for $N(\log^\star n) < i \leq n$) using $\mathcal{Q}_{N(\log^\star n)}$ $O(n)$

– threading cost: Lem. 2 $\Rightarrow O(1)$

– locating cost: Lem. 4 $\Rightarrow O(\log n / \underbrace{N(\log^\star n)}_{> n/2}) = O(1)$

$\cdot O(n) =$

Time Complexity

$N(h) := \lceil n / \log^{(h)} n \rceil$

Step 1: Random permutation — $O(n)$

Step 2: Setting up $\mathcal{T}_1$, $\mathcal{Q}_1$, and $\pi(v)$ — $O(n)$

Step 3: Phases 1 to $\log^\star n$ — $(\log^\star n) \cdot$

Step 3.2: Walking the polygon — Lemma 5 $\Rightarrow$ $O(n)$

Step 3.1: Inserting $s_i = v_i w_i$ using $\mathcal{Q}_{N(h-1)}$

– threading cost: Lem. 2 $\Rightarrow$ expected $O(1)$ per segm.

– locating cost: Know the location of v_i in $\mathcal{Q}_{N(h-1)}$

Lem. 4 $\Rightarrow$ expected location cost $O(\log(i/N(h-1))) \subseteq O(\log^{(h)} n)$

$\Big\} \cdot N(h) = O(n)$

Step 4: Inserting s_i (for $N(\log^\star n) < i \leq n$) using $\mathcal{Q}_{N(\log^\star n)}$ — $O(n)$

– threading cost: Lem. 2 $\Rightarrow O(1)$

– locating cost: Lem. 4 $\Rightarrow O(\log n / N(\log^\star n)) = O(1)$, where $N(\log^\star n) > n/2$

$\Big\} \cdot O(n) = O(n)$

Total: $O(n \log^\star n)$

The Results

Theorem. Let S be the edge set of a polygon, $|S| = n$.

- We can build $\mathcal{T}(S)$ and $\mathcal{Q}(S)$ in $O(n \log^{\star} n)$ expected time.
- The expected size of $\mathcal{Q}(S)$ is $O(n)$.
- The expected time for locating a point in $\mathcal{T}(S)$ via $\mathcal{Q}(S)$ is $O(\log n)$.

The Results

Theorem. Let S be the edge set of a polygon, $|S| = n$.

- We can build $\mathcal{T}(S)$ and $\mathcal{Q}(S)$ in $O(n \log^{\star} n)$ expected time.
- The expected size of $\mathcal{Q}(S)$ is $O(n)$.
- The expected time for locating a point in $\mathcal{T}(S)$ via $\mathcal{Q}(S)$ is $O(\log n)$.

Theorem. Let S be the edge set of a plane straight-line graph with k connected components, $|S| = n$.

- We can build $\mathcal{T}(S)$ and $\mathcal{Q}(S)$ in $O(n \log^{\star} n + k \log n)$ expected time.
- The expected size of $\mathcal{Q}(S)$ is $O(n)$.
- The expected time for locating a point in $\mathcal{T}(S)$ via $\mathcal{Q}(S)$ is $O(\log n)$.