

# Computational Geometry

Winter semester 2014/15

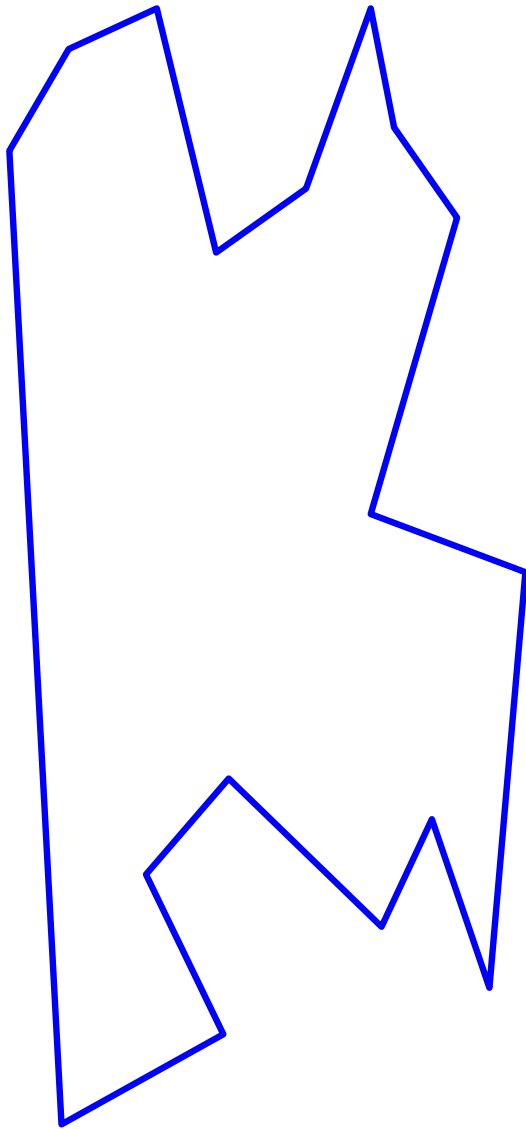
## Seidel's Triangulation Algorithm

Lecture #14

*Dipl.-Inf. Philipp Kindermann,  
Prof. Dr. Alexander Wolff*

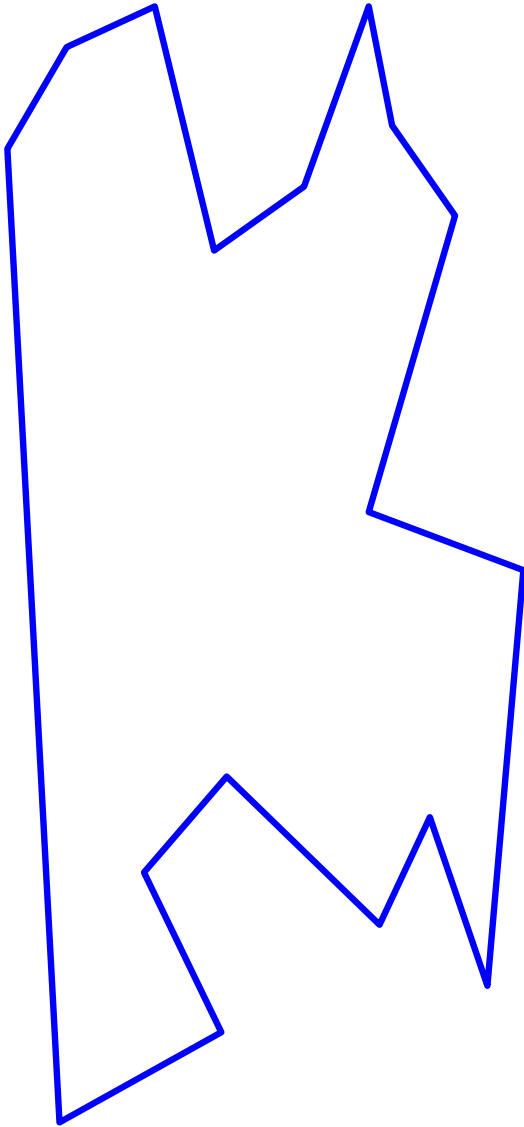
*Chair for Computer Science I*

# Triangulating a Polygon



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

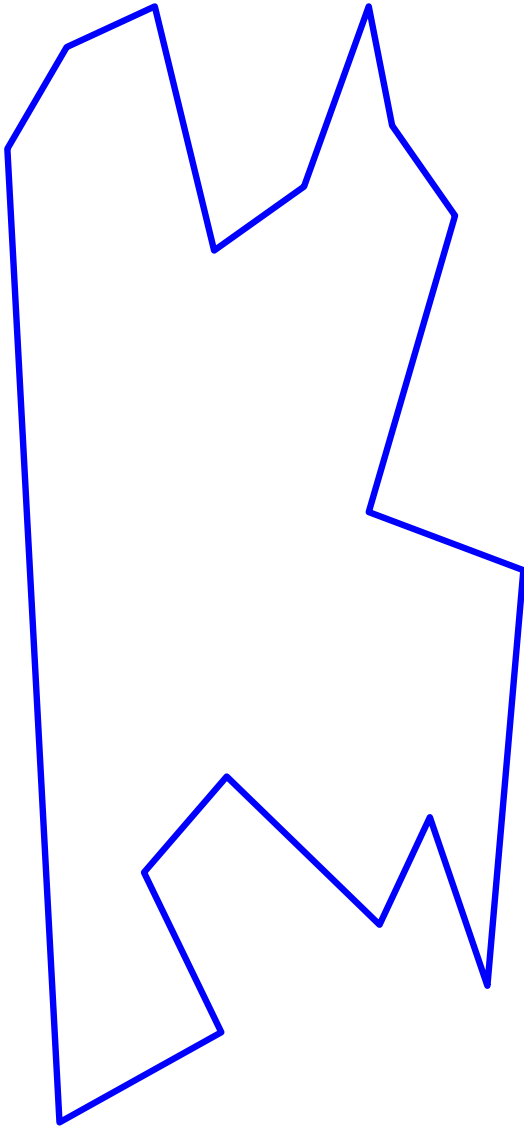
# Triangulating a Polygon



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

**Find:** Triangulation of  $P$

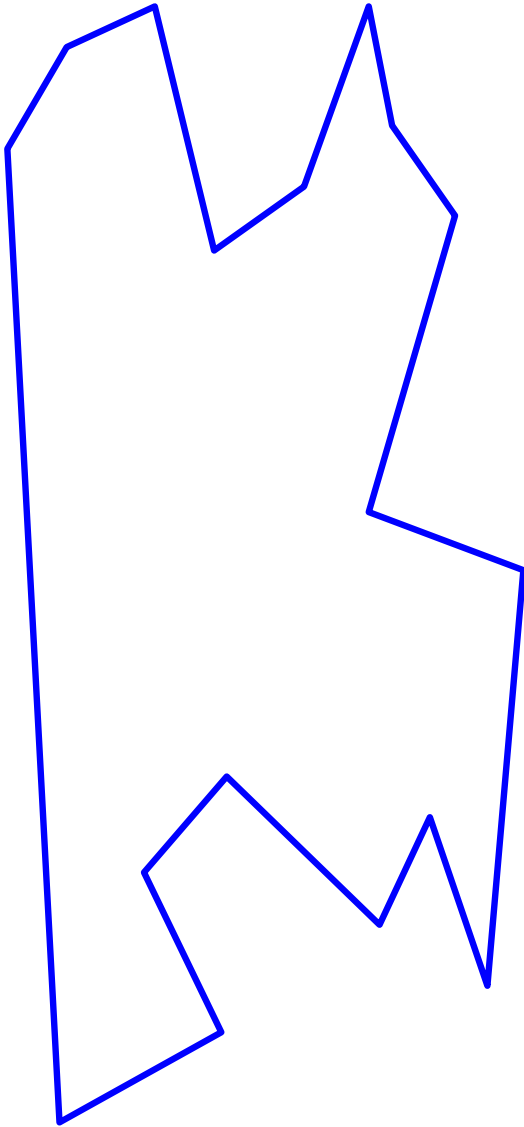
# Triangulating a Polygon



**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  $\overline{p_i p_j} \subset P$ )

# Triangulating a Polygon

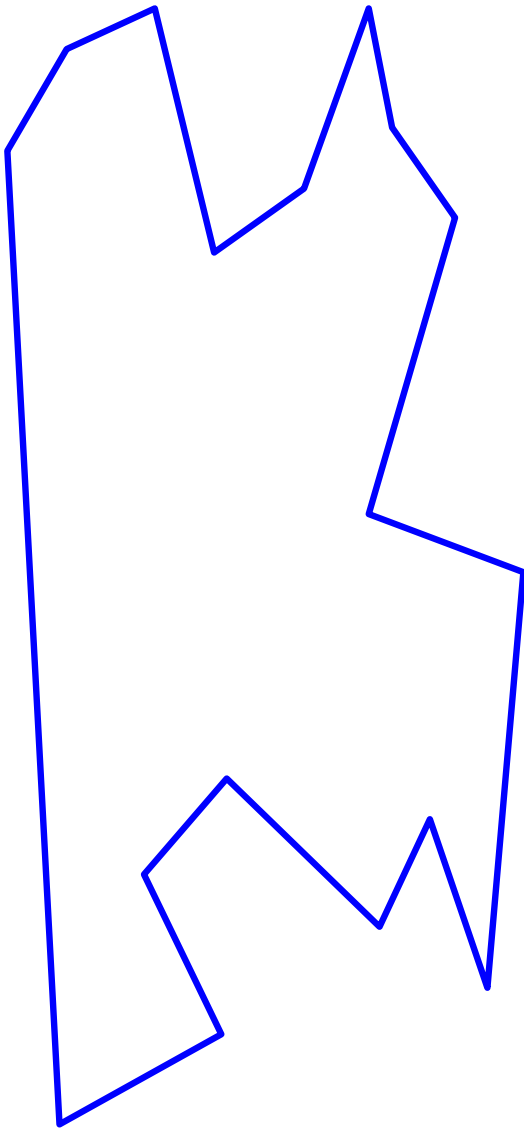


**Given:** Polygon  $P = \langle p_1, \dots, p_n \rangle$   
(list of vertices in cw order)

**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:**

# Triangulating a Polygon



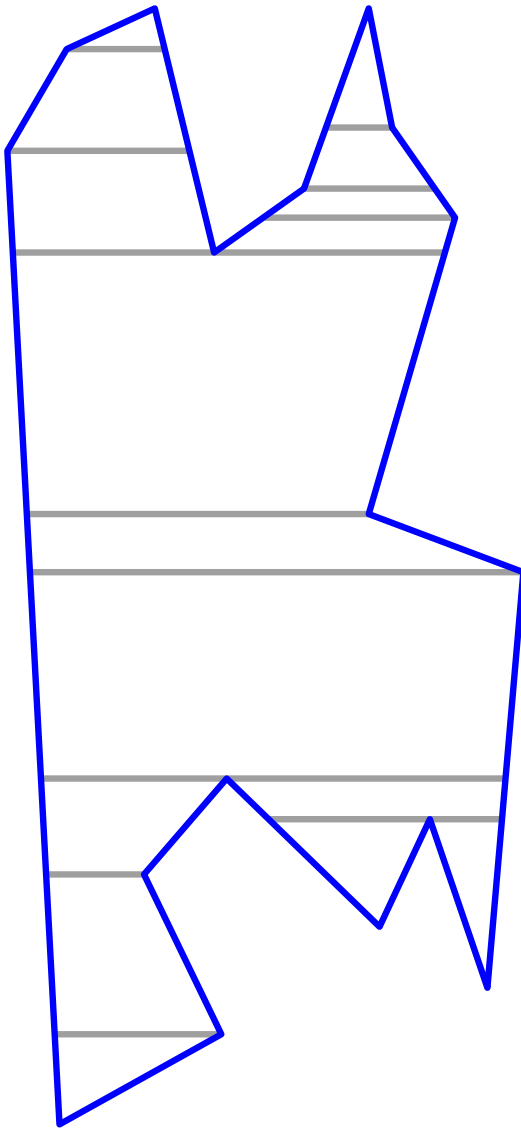
**Given:** Polygon  $P = \langle p_1, \dots, p_n \rangle$   
(list of vertices in cw order)

**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  $P$ .

# Triangulating a Polygon



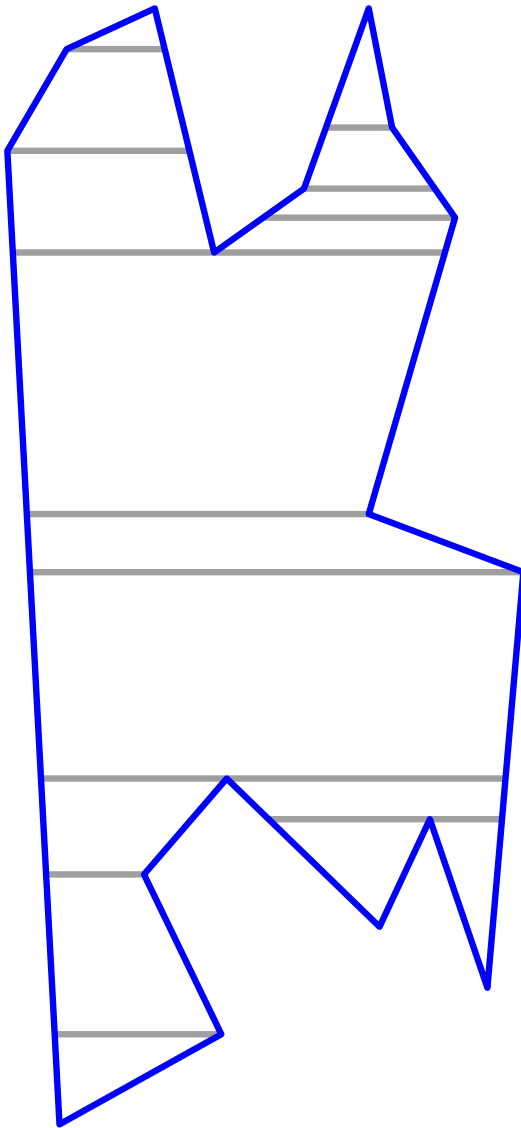
**Given:** Polygon  $P = \langle p_1, \dots, p_n \rangle$   
(list of vertices in cw order)

**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  $P$ .

# Triangulating a Polygon



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

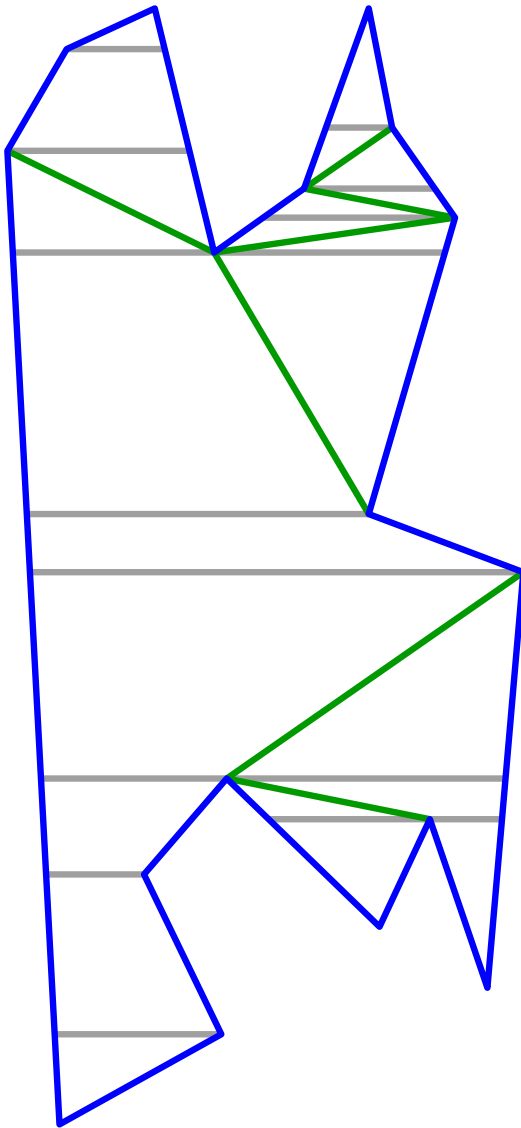
**Find:** Triangulation of  $P$   
i.e., a partition of  $P$  into triangles by  
*diagonals* (segments of type  $\overline{p_i p_j} \subset P$ )

**Approach:**

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



# Triangulating a Polygon



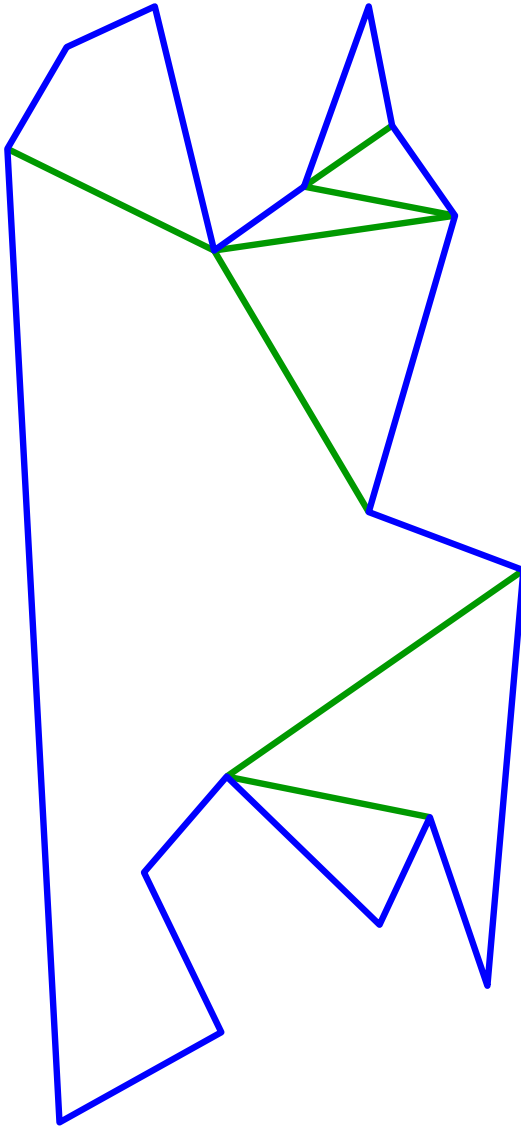
**Given:** Polygon  $P = \langle p_1, \dots, p_n \rangle$   
(list of vertices in cw order)

**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



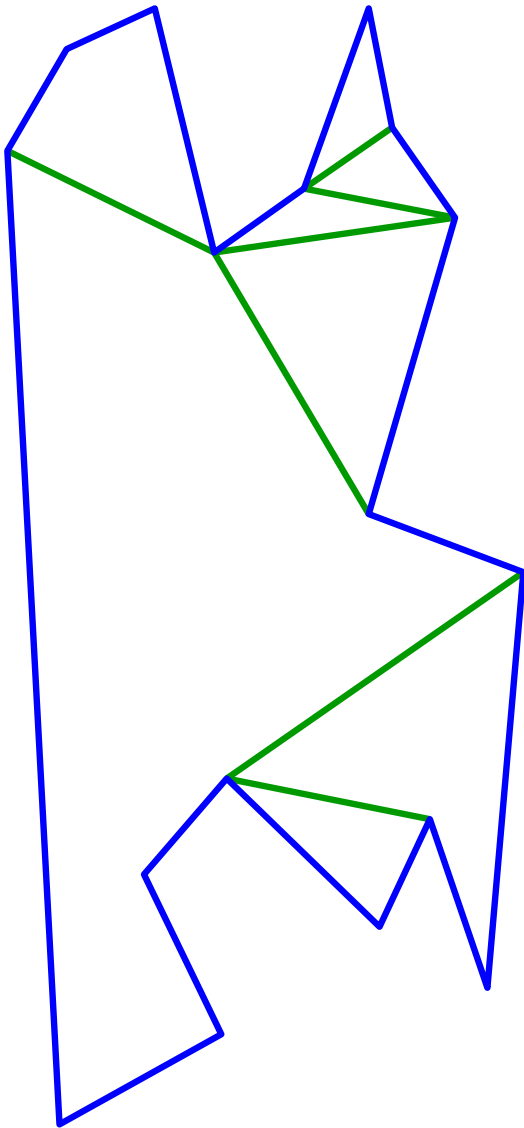
**Given:** Polygon  $P = \langle p_1, \dots, p_n \rangle$   
(list of vertices in cw order)

**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



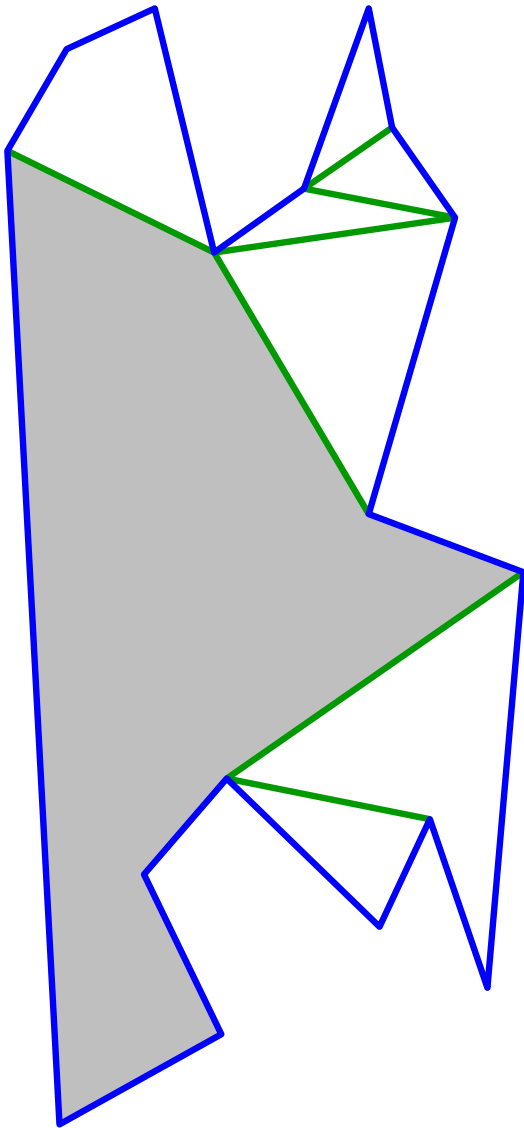
**Given:** Polygon  $P = \langle p_1, \dots, p_n \rangle$   
(list of vertices in cw order)

**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.
3. Triangulate  $y$ -monotone subpolygons

# Triangulating a Polygon



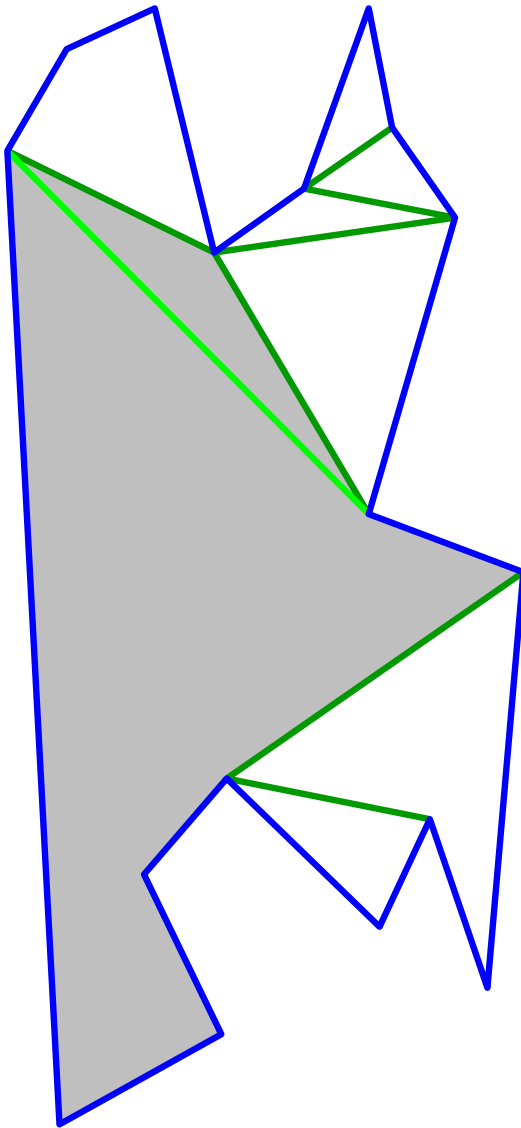
**Given:** Polygon  $P = \langle p_1, \dots, p_n \rangle$   
(list of vertices in cw order)

**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:**

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

# Triangulating a Polygon



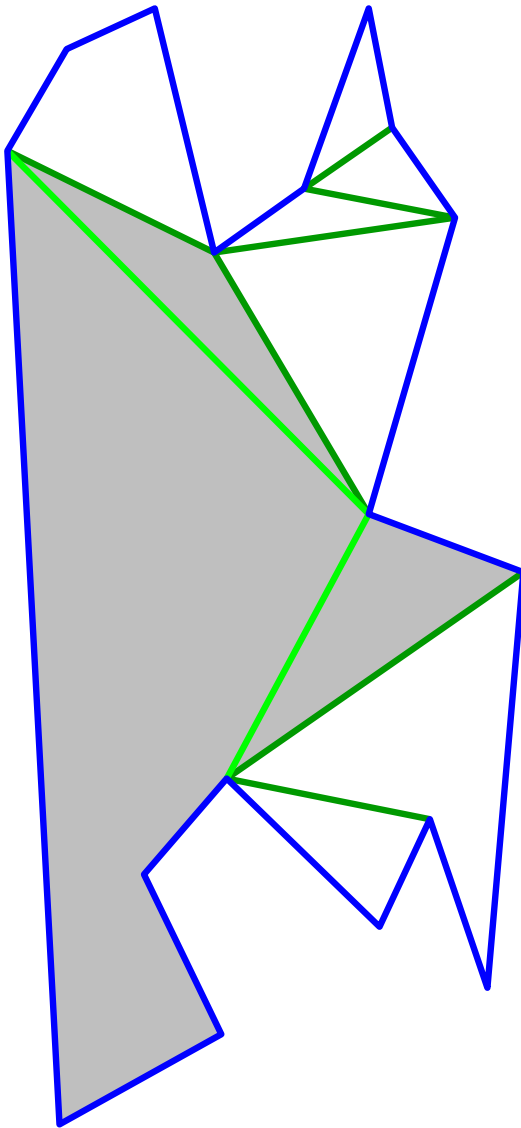
**Given:** Polygon  $P = \langle p_1, \dots, p_n \rangle$   
(list of vertices in cw order)

**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:**

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

# Triangulating a Polygon



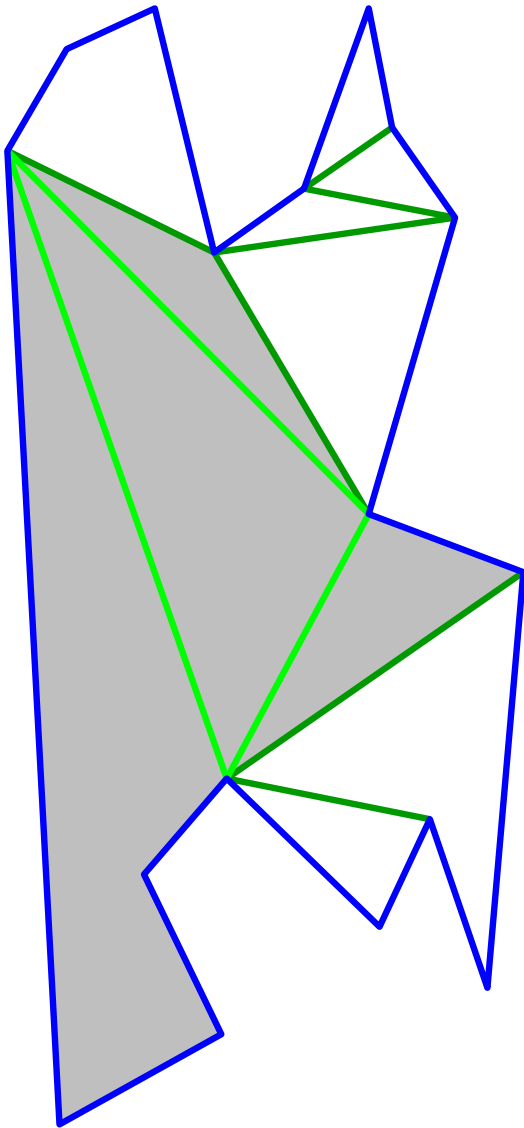
**Given:** Polygon  $P = \langle p_1, \dots, p_n \rangle$   
(list of vertices in cw order)

**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:**

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

# Triangulating a Polygon



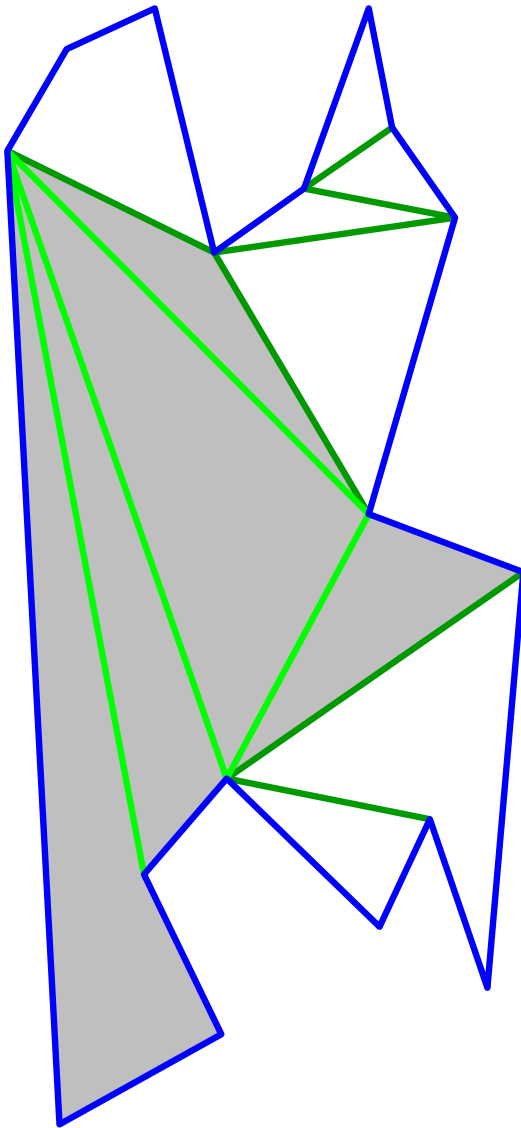
**Given:** Polygon  $P = \langle p_1, \dots, p_n \rangle$   
(list of vertices in cw order)

**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:**

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)

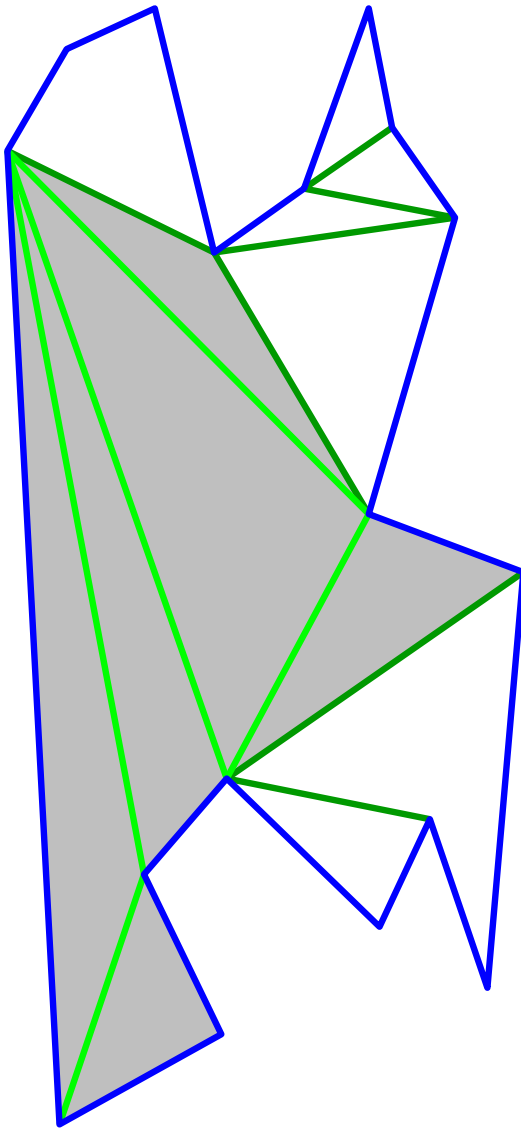
**Find:** Triangulation of  $P$   
i.e., a partition of  $P$  into triangles by  
*diagonals* (segments of type  $\overline{p_i p_j} \subset P$ )

**Approach:**

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



# Triangulating a Polygon



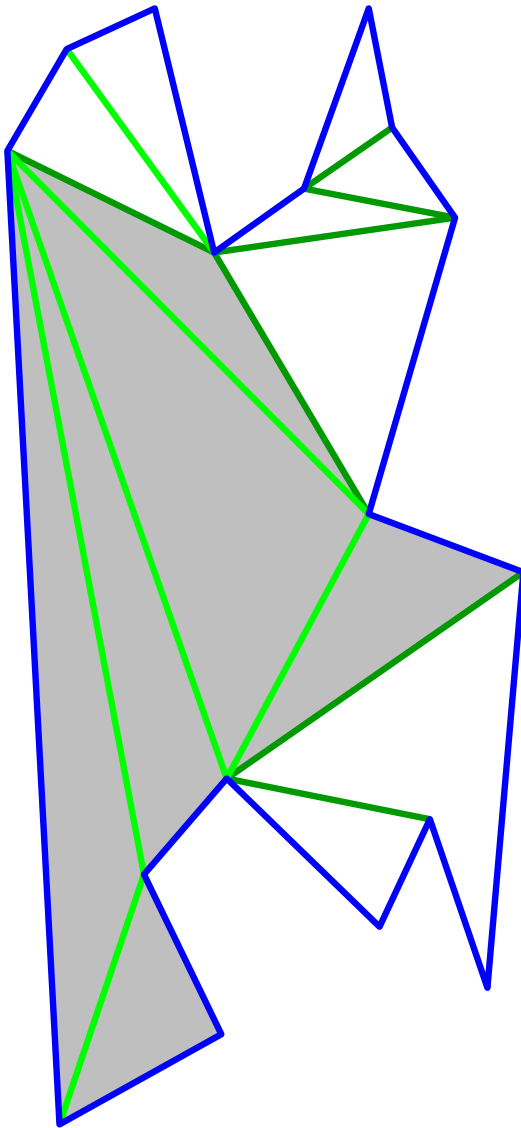
**Given:** Polygon  $P = \langle p_1, \dots, p_n \rangle$   
(list of vertices in cw order)

**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:**

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

# Triangulating a Polygon



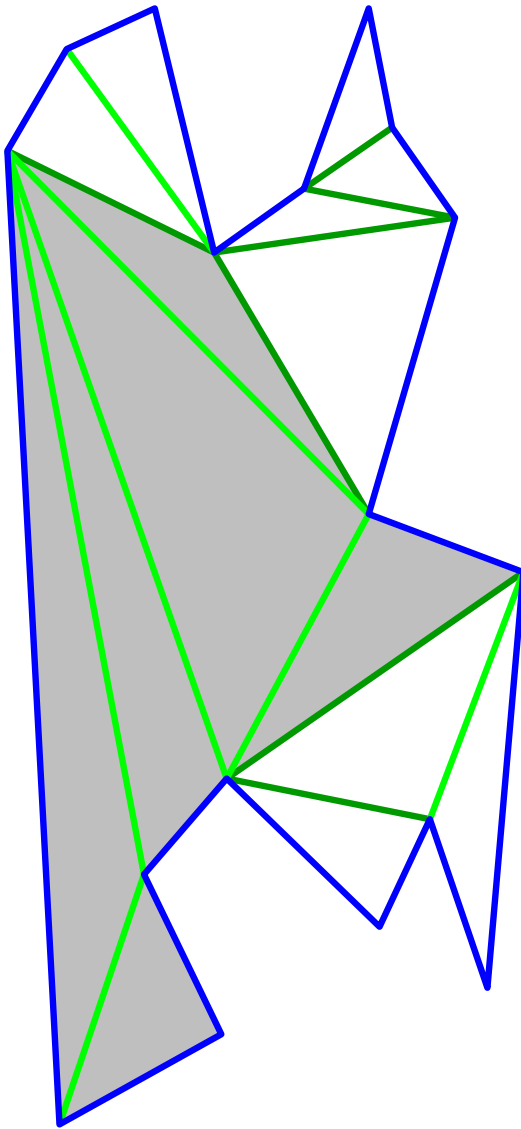
**Given:** Polygon  $P = \langle p_1, \dots, p_n \rangle$   
(list of vertices in cw order)

**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:**

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

# Triangulating a Polygon



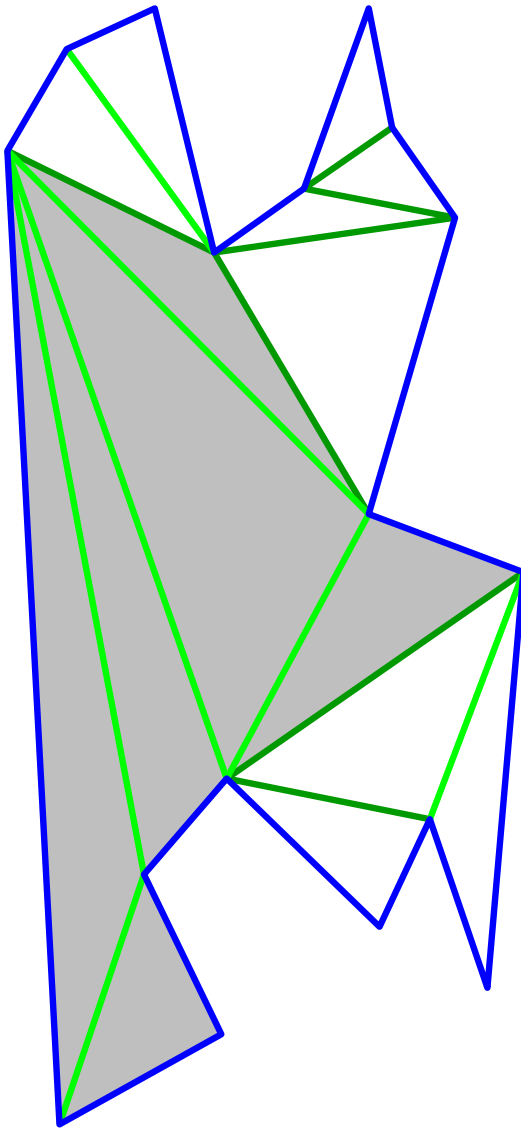
**Given:** Polygon  $P = \langle p_1, \dots, p_n \rangle$   
(list of vertices in cw order)

**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:**

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)

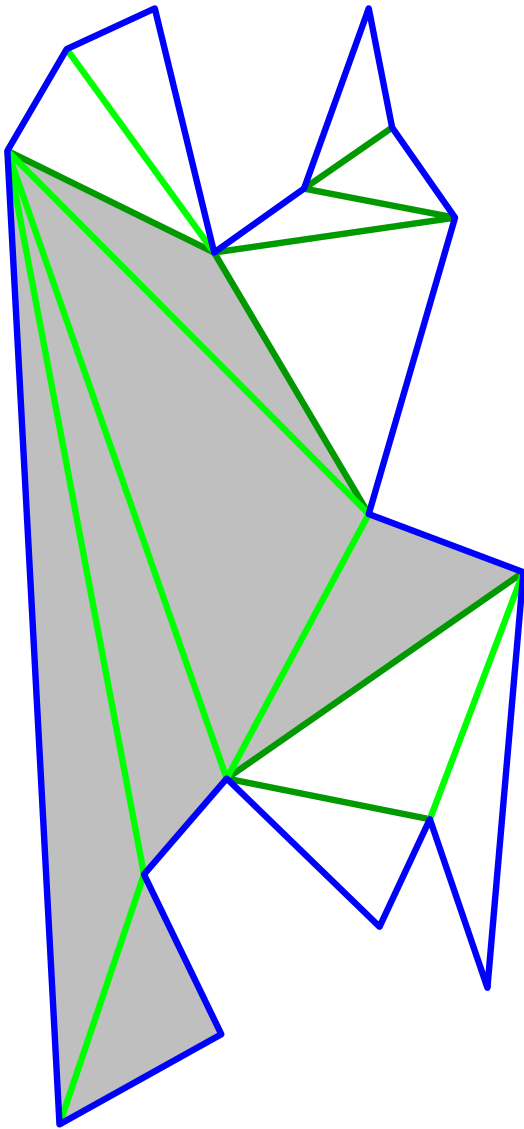
**Find:** Triangulation of  $P$   
i.e., a partition of  $P$  into triangles by  
*diagonals* (segments of type  $\overline{p_i p_j} \subset P$ )

**Approach:**

*Running time:*

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

# Triangulating a Polygon



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

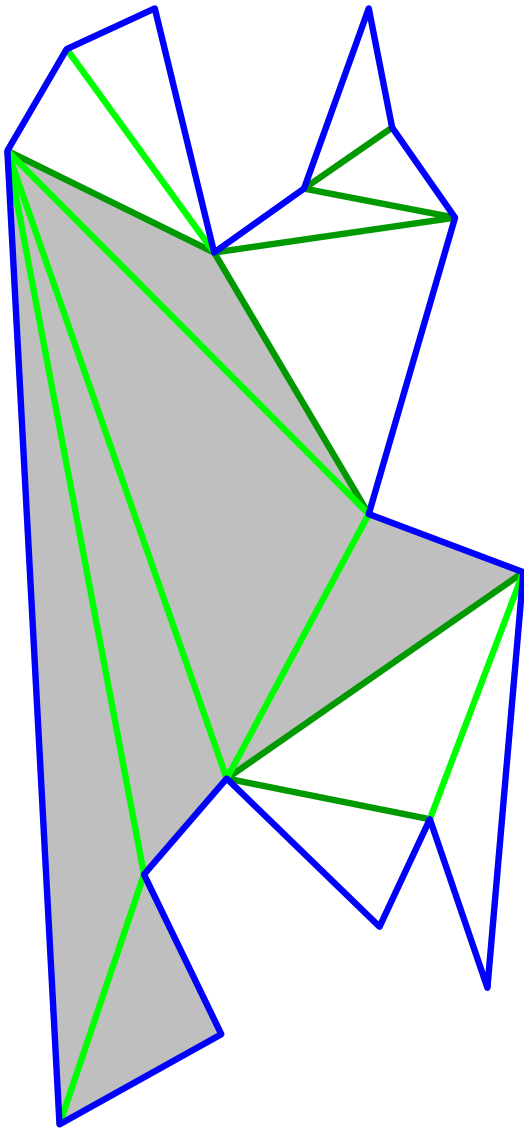
**Find:** Triangulation of  $P$   
i.e., a partition of  $P$  into triangles by  
*diagonals* (segments of type  $\overline{p_i p_j} \subset P$ )

**Approach:**

*Running time:*

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

# Triangulating a Polygon



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

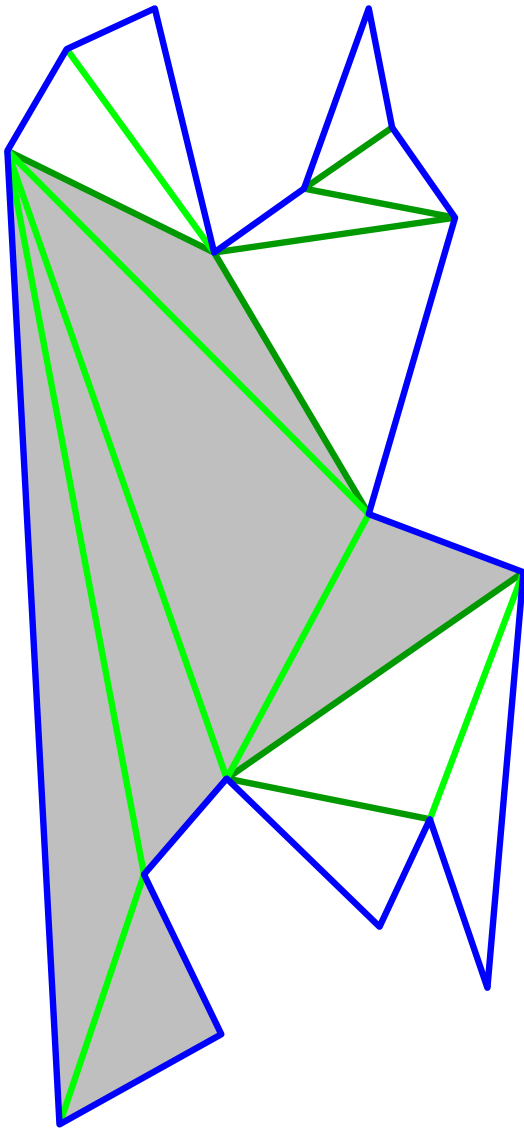
**Find:** Triangulation of  $P$   
i.e., a partition of  $P$  into triangles by  
*diagonals* (segments of type  $\overline{p_i p_j} \subset P$ )

**Approach:**

*Running time:*

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

# Triangulating a Polygon



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

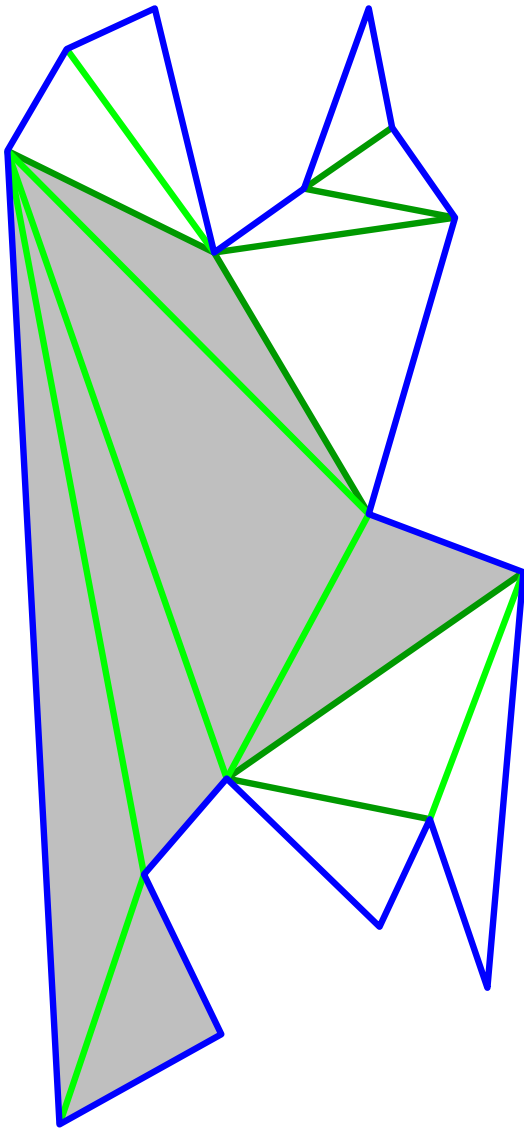
**Find:** Triangulation of  $P$   
i.e., a partition of  $P$  into triangles by  
*diagonals* (segments of type  $\overline{p_i p_j} \subset P$ )

**Approach:**

*Running time:*

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

# 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  $\overline{p_i p_j} \subset P$ )

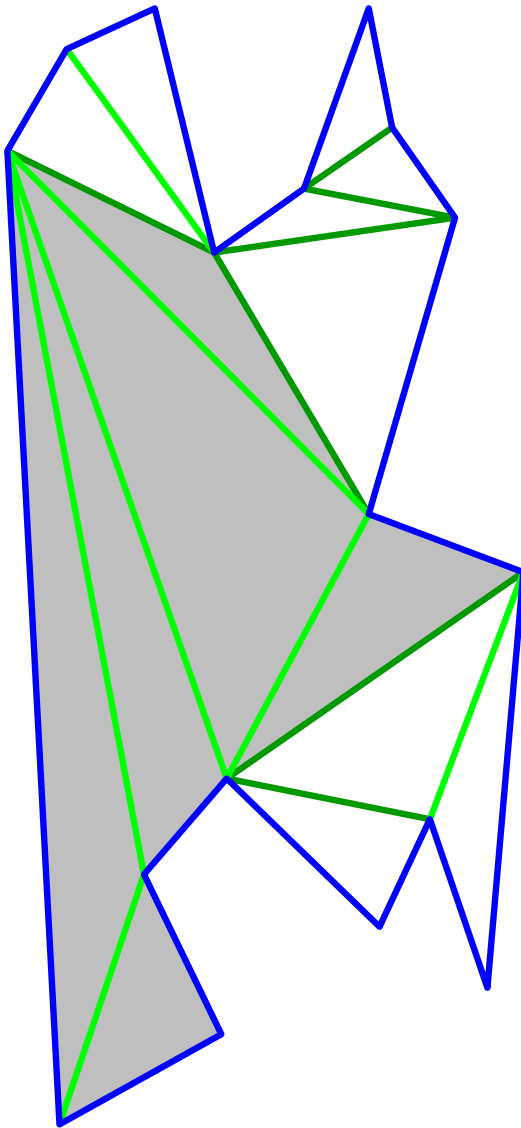
**Approach:**

*Running time:*

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



# Triangulating a Polygon



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

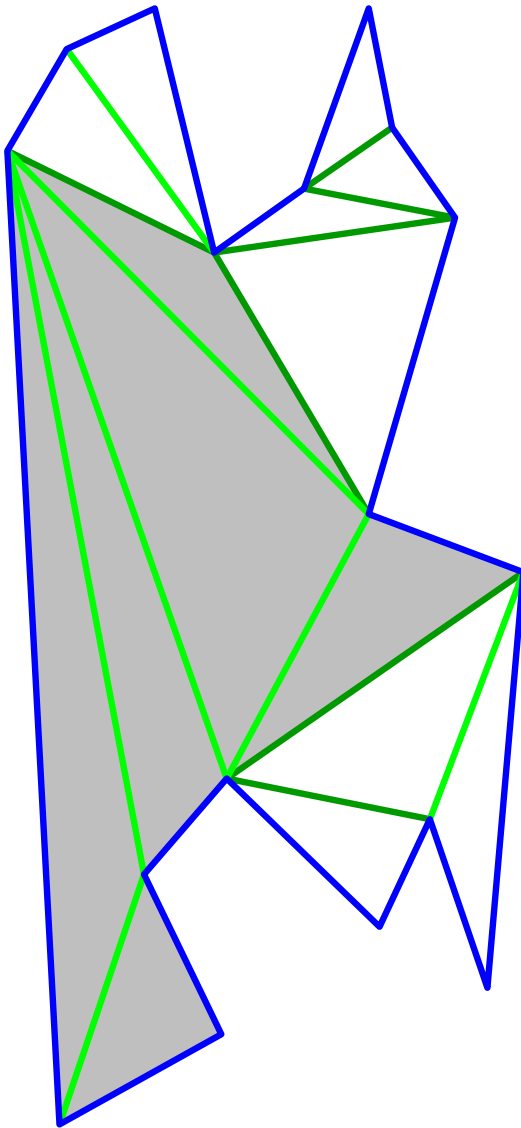
**Find:** Triangulation of  $P$   
i.e., a partition of  $P$  into triangles by  
*diagonals* (segments of type  $\overline{p_i p_j} \subset P$ )

**Approach:**

*Running time:*

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

# 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  $\overline{p_i p_j} \subset P$ )

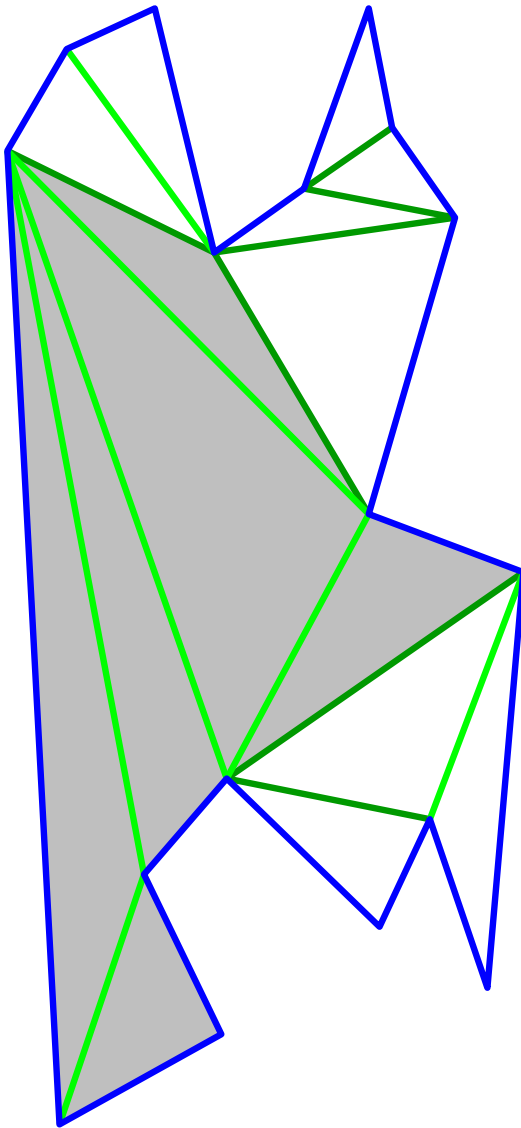
**Approach:**

*Running time:*

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

**Lemma 1.** Given a trapezoidation,  $P$  can be triangulated in linear time.

# 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  $P$ .  $O(?)$
  2. Draw diagonals inside trapezoids.  $O(n)$
  3. Triangulate  $y$ -monotone subpolygons  $O(n)$
- 
- $O(?)$

**Lemma 1.** Given a trapezoidation,  $P$  can be triangulated in linear time.

# General Idea

Let  $S$  be a set of  $n$  non-crossing segments, e.g., polygonal line

- 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, e.g., polygonal line

**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, e.g., polygonal line

**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)

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

$S_0 \leftarrow \emptyset$

**for**  $i = 1$  **to**  $n$  **do**

    let  $S_i = S_{i-1} \cup \{s_i\}$

    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, e.g., polygonal line

**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)

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

$S_0 \leftarrow \emptyset$

**for**  $i = 1$  **to**  $n$  **do**

    let  $S_i = S_{i-1} \cup \{s_i\}$

    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” time

# 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

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.

# 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.*

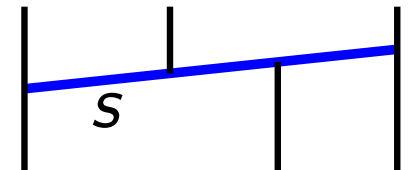
# Threading time

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  $s$ .



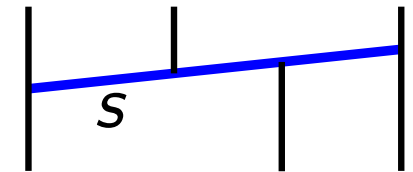
# Threading time

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  $s_i =$

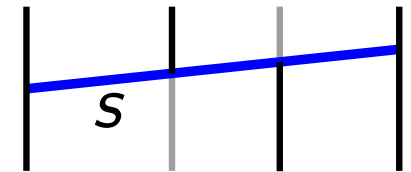
# Threading time

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

*Use lexicographic order!*

**Lemma 2.** For  $i = 1, \dots, n$ , 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  $s_i = \deg(s_i, \mathcal{T}(S_i))$

# Threading time

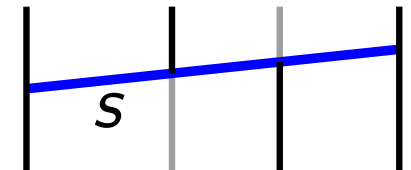
We assume general position  
(no two points have the same  $y$ -coordinate).

*Use lexicographic order!*

**Lemma 2.** For  $i = 1, \dots, n$ , 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  $s_i = \deg(s_i, \mathcal{T}(S_i))$

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

# Threading time

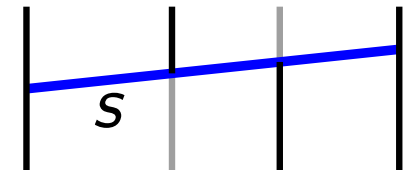
We assume general position  
(no two points have the same  $y$ -coordinate).

*Use lexicographic order!*

**Lemma 2.** For  $i = 1, \dots, n$ , 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  $s_i = \deg(s_i, \mathcal{T}(S_i))$

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



# Threading time

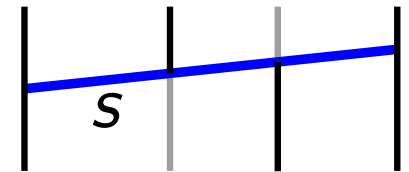
We assume general position  
(no two points have the same  $y$ -coordinate).

*Use lexicographic order!*

**Lemma 2.** For  $i = 1, \dots, n$ , 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  $s_i = \deg(s_i, \mathcal{T}(S_i))$

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

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

# Threading time

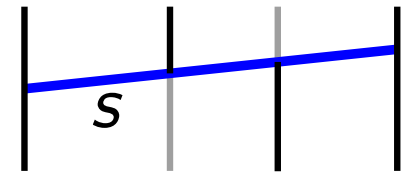
We assume general position  
(no two points have the same  $y$ -coordinate).

*Use lexicographic order!*

**Lemma 2.** For  $i = 1, \dots, n$ , 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  $s_i = \deg(s_i, \mathcal{T}(S_i))$

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

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

# Threading time

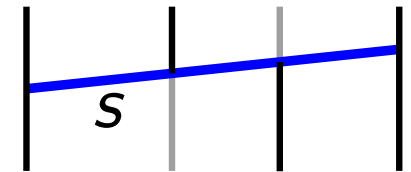
We assume general position  
(no two points have the same  $y$ -coordinate).

*Use lexicographic order!*

**Lemma 2.** For  $i = 1, \dots, n$ , 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  $s_i = \deg(s_i, \mathcal{T}(S_i))$

$\#$  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

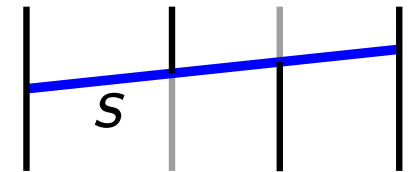
We assume general position  
(no two points have the same  $y$ -coordinate).

*Use lexicographic order!*

**Lemma 2.** For  $i = 1, \dots, n$ , 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  $s_i = \deg(s_i, \mathcal{T}(S_i))$

$\#$  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

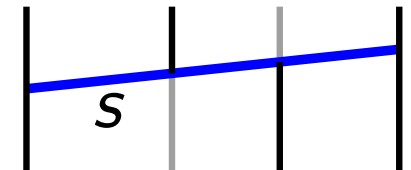
We assume general position  
(no two points have the same  $y$ -coordinate).

*Use lexicographic order!*

**Lemma 2.** For  $i = 1, \dots, n$ , 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  $s_i = \deg(s_i, \mathcal{T}(S_i))$

$\#$  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:*  $H_n := 1 + \frac{1}{2} + \dots + \frac{1}{n} \in \Theta(\quad)$

# Location Time

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

# 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  $n > 1$ .



# 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  $5H_n \in O(\log n)$ .

# 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:*  $H_n := 1 + \frac{1}{2} + \dots + \frac{1}{n} \in \Theta(\log n)$

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  $O(\log n)$ .

*Aim:* Speed-up construction for simple polygonal chains.

# New Approach

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

# New Approach

*Observe:* in  $Q(S_i)$ ,

- 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  $Q(S_i)$ ,

- 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  $Q(S_i)$ ,

- 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  $Q(S_i)$ ,

- 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  $Q(S_i)$ ,

- 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  $Q(S_i)$ ,

- 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  $Q(S_i)$ ,

- 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  $Q(S_i)$ ,

- 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 triangulation

# New Approach

*Observe:* in  $Q(S_i)$ ,

- 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 triangulation –  
*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 triangulation?



# 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 triangulation?

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

# 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 triangulation?

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

# 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

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

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

# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

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

# Logs All Over the Place

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

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

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

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

**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

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

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

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

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

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



# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

**Examples.**  $\log^{(0)} 2^{2^{2^2}} = 2^{2^{2^2}}$   
 $\log^{(1)} 2^{2^{2^2}} = \log_2 2^{2^{2^2}} =$

# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

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

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

# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

**Examples.**

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

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

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

# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

**Examples.**

$$\begin{aligned} \log^{(0)} 2^{2^{2^2}} &= 2^{2^{2^2}} \\ \log^{(1)} 2^{2^{2^2}} &= \log_2 2^{2^{2^2}} = 2^{2^2} \\ \log^{(2)} 2^{2^{2^2}} &= \log_2 \log^{(1)} 2^{2^{2^2}} = \end{aligned}$$

# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

**Examples.**

$$\begin{aligned} \log^{(0)} 2^{2^{2^2}} &= 2^{2^{2^2}} \\ \log^{(1)} 2^{2^{2^2}} &= \log_2 2^{2^{2^2}} = 2^{2^2} \\ \log^{(2)} 2^{2^{2^2}} &= \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2 \end{aligned}$$

# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

**Examples.**

$$\begin{aligned} \log^{(0)} 2^{2^{2^2}} &= 2^{2^{2^2}} \\ \log^{(1)} 2^{2^{2^2}} &= \log_2 2^{2^{2^2}} = 2^{2^2} \\ \log^{(2)} 2^{2^{2^2}} &= \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2 \\ \log^{(3)} 2^{2^{2^2}} &= \end{aligned}$$

# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

**Examples.**

$$\begin{aligned} \log^{(0)} 2^{2^{2^2}} &= 2^{2^{2^2}} \\ \log^{(1)} 2^{2^{2^2}} &= \log_2 2^{2^{2^2}} = 2^{2^2} \\ \log^{(2)} 2^{2^{2^2}} &= \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2 \\ \log^{(3)} 2^{2^{2^2}} &= 2; \end{aligned}$$

# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

**Examples.**

$$\begin{aligned} \log^{(0)} 2^{2^{2^2}} &= 2^{2^{2^2}} \\ \log^{(1)} 2^{2^{2^2}} &= \log_2 2^{2^{2^2}} = 2^{2^2} \\ \log^{(2)} 2^{2^{2^2}} &= \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2 \\ \log^{(3)} 2^{2^{2^2}} &= 2; \quad \log^{(4)} 2^{2^{2^2}} = \end{aligned}$$



# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

**Examples.**

$$\begin{aligned} \log^{(0)} 2^{2^{2^2}} &= 2^{2^{2^2}} \\ \log^{(1)} 2^{2^{2^2}} &= \log_2 2^{2^{2^2}} = 2^{2^2} \\ \log^{(2)} 2^{2^{2^2}} &= \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2 \\ \log^{(3)} 2^{2^{2^2}} &= 2; \quad \log^{(4)} 2^{2^{2^2}} = 1 \end{aligned}$$

# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

**Examples.**

$$\begin{aligned} \log^{(0)} 2^{2^{2^2}} &= 2^{2^{2^2}} \\ \log^{(1)} 2^{2^{2^2}} &= \log_2 2^{2^{2^2}} = 2^{2^2} \\ \log^{(2)} 2^{2^{2^2}} &= \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2 \\ \log^{(3)} 2^{2^{2^2}} &= 2; \quad \log^{(4)} 2^{2^{2^2}} = 1 \Rightarrow \log^* 2^{2^{2^2}} = \end{aligned}$$

# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

**Examples.**

$$\begin{aligned} \log^{(0)} 2^{2^{2^2}} &= 2^{2^{2^2}} \\ \log^{(1)} 2^{2^{2^2}} &= \log_2 2^{2^{2^2}} = 2^{2^2} \\ \log^{(2)} 2^{2^{2^2}} &= \log_2 \log^{(1)} 2^{2^{2^2}} = 2^2 \\ \log^{(3)} 2^{2^{2^2}} &= 2; \quad \log^{(4)} 2^{2^{2^2}} = 1 \Rightarrow \log^* 2^{2^{2^2}} = 4 \end{aligned}$$

# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

**Examples.**

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

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

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

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

$$N(0) = 1$$

# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

**Examples.**

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

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

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

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

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

# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

**Examples.**

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

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

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

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

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

# 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  $\log^* n := \max\{i \mid \log^{(i)} n \geq 1\}$ .

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

**Examples.**

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

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

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

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

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

# The Algorithm

PolygonTrapezoidation((edges along) simple polygonal chain  $P$ )



# The Algorithm

PolygonTrapezoidation((edges along) simple polygonal chain  $P$ )

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

# The Algorithm

PolygonTrapezoidation((edges along) simple polygonal chain  $P$ )

1.  $\langle s_1, s_2, \dots, s_n \rangle :=$  random ordering of the edges of  $P$
2. Compute  $\mathcal{T}_1$  and  $\mathcal{Q}_1$  for  $\{s_1\}$ .  
For each vtx  $v$  of  $P$ , let  $\pi(v)$  point to the leaf of  $\mathcal{Q}_1$  that contains  $v$ .

# The Algorithm

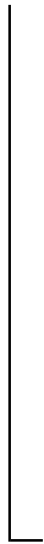
PolygonTrapezoidation((edges along) simple polygonal chain  $P$ )

1.  $\langle s_1, s_2, \dots, s_n \rangle :=$  random ordering of the edges of  $P$
2. Compute  $\mathcal{T}_1$  and  $\mathcal{Q}_1$  for  $\{s_1\}$ .  
For each vtx  $v$  of  $P$ , let  $\pi(v)$  point to the leaf of  $\mathcal{Q}_1$  that contains  $v$ .

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

3.1

3.2



# The Algorithm

PolygonTrapezoidation((edges along) simple polygonal chain  $P$ )

1.  $\langle s_1, s_2, \dots, s_n \rangle :=$  random ordering of the edges of  $P$
2. Compute  $\mathcal{T}_1$  and  $\mathcal{Q}_1$  for  $\{s_1\}$ .  
For each vtx  $v$  of  $P$ , let  $\pi(v)$  point to the leaf of  $\mathcal{Q}_1$  that contains  $v$ .

**for**  $h = 1$  **to**  $\log^* n$  **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 polygonal chain  $P$ )

1.  $\langle s_1, s_2, \dots, s_n \rangle :=$  random ordering of the edges of  $P$
2. Compute  $\mathcal{T}_1$  and  $\mathcal{Q}_1$  for  $\{s_1\}$ .  
For each vtx  $v$  of  $P$ , let  $\pi(v)$  point to the leaf of  $\mathcal{Q}_1$  that contains  $v$ .

**for**  $h = 1$  **to**  $\log^* n$  **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  $\mathcal{T}_{N(h)}$ :  
       for each vertex  $v$   
           – let  $\Delta$  be the trapezoid in  $\mathcal{T}_{N(h)}$  that contains  $v$   
           – let  $\pi(v)$  be the node in  $\mathcal{Q}_{N(h)}$  corresponding to  $\Delta$

# The Algorithm

PolygonTrapezoidation((edges along) simple polygonal chain  $P$ )

1.  $\langle s_1, s_2, \dots, s_n \rangle :=$  random ordering of the edges of  $P$
2. Compute  $\mathcal{T}_1$  and  $\mathcal{Q}_1$  for  $\{s_1\}$ .  
For each vtx  $v$  of  $P$ , let  $\pi(v)$  point to the leaf of  $\mathcal{Q}_1$  that contains  $v$ .

**for**  $h = 1$  **to**  $\log^* n$  **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  $\mathcal{T}_{N(h)}$ :

        for each vertex  $v$

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

          – let  $\pi(v)$  be the node in  $\mathcal{Q}_{N(h)}$  corresponding to  $\Delta$

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

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

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

# 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  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

# Time Complexity

Step 1: Random permutation

$$O(n)$$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$$O(n)$$

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

$$(\log^* n) \cdot$$

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

Step 3.2: Walking the polygon

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

Step 3.2: Walking the polygon

Lemma 5  $\Rightarrow$

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

Step 3.2: Walking the polygon

Lemma 5  $\Rightarrow$

$O(n)$

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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:

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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

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

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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

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

$\cdot N(h) =$



# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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

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

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

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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  $N(\log^* n) < i \leq n$ ) using  $\mathcal{Q}_{N(\log^* n)}$

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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  $N(\log^* n) < i \leq n$ ) using  $\mathcal{Q}_{N(\log^* n)}$

– threading cost:

– locating cost:

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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  $N(\log^* n) < i \leq n$ ) using  $\mathcal{Q}_{N(\log^* n)}$

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

– locating cost:

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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  $N(\log^* n) < i \leq n$ ) using  $\mathcal{Q}_{N(\log^* n)}$

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

– locating cost: Lem. 4  $\Rightarrow$

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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  $N(\log^* n) < i \leq n$ ) using  $\mathcal{Q}_{N(\log^* n)}$

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

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

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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  $N(\log^* n) < i \leq n$ ) using  $\mathcal{Q}_{N(\log^* n)}$

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

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

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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  $N(\log^* n) < i \leq n$ ) using  $\mathcal{Q}_{N(\log^* n)}$

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

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



# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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

$\left. \vphantom{\begin{array}{l} \text{– threading cost:} \\ \text{– locating cost:} \end{array}} \right\} \cdot N(h) = O(n)$

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

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

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

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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  $N(\log^* n) < i \leq n$ ) using  $\mathcal{Q}_{N(\log^* n)}$

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

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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

$\left. \vphantom{\begin{matrix} \text{– threading cost:} \\ \text{– locating cost:} \end{matrix}} \right\} \cdot N(h) = O(n)$

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

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

$\left. \vphantom{\begin{matrix} \text{– threading cost:} \\ \text{– locating cost:} \end{matrix}} \right\} \cdot O(n) =$

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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  $N(\log^* n) < i \leq n$ ) using  $\mathcal{Q}_{N(\log^* n)}$

$O(n)$

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

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

$\cdot O(n) =$

$> n/2$

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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

$\left. \vphantom{\begin{array}{l} \text{– threading cost:} \\ \text{– locating cost:} \end{array}} \right\} \cdot N(h) = O(n)$

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

$O(n)$

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

$\left. \vphantom{\begin{array}{l} \text{– threading cost:} \\ \text{– locating cost:} \end{array}} \right\} \cdot O(n) = \underline{O(n)}$

# Time Complexity

Step 1: Random permutation

$O(n)$

Step 2: Setting up  $\mathcal{T}_1$  and  $\mathcal{Q}_1$

$O(n)$

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

$(\log^* n) \cdot$

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  $N(\log^* n) < i \leq n$ ) using  $\mathcal{Q}_{N(\log^* n)}$

$O(n)$

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

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

$\cdot O(n) =$

$O(n \log^* n)$

# The Results

- Theorem.** Let  $S$  be the edge set of a **polygonal chain**,  $|S| = n$ .
- 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  $O(\log n)$ .

# The Results

**Theorem.** Let  $S$  be the edge set of a **polygonal chain**,  $|S| = n$ .

- 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  $O(\log n)$ .

**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  $O(n \log^* n + k \log n)$  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  $O(\log n)$ .