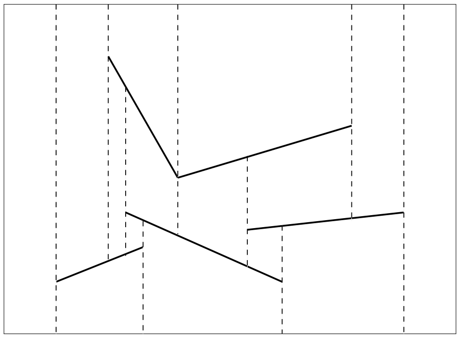
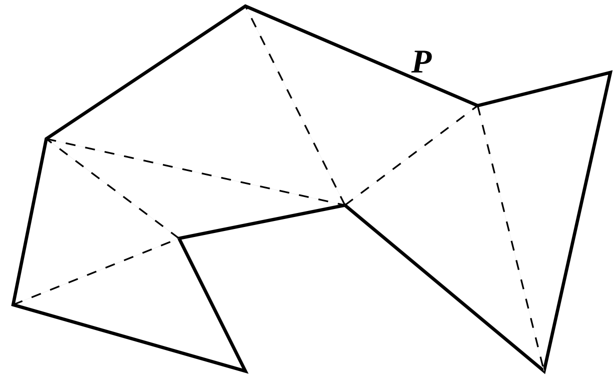
The trapezoidal map of
non-crossing line segments



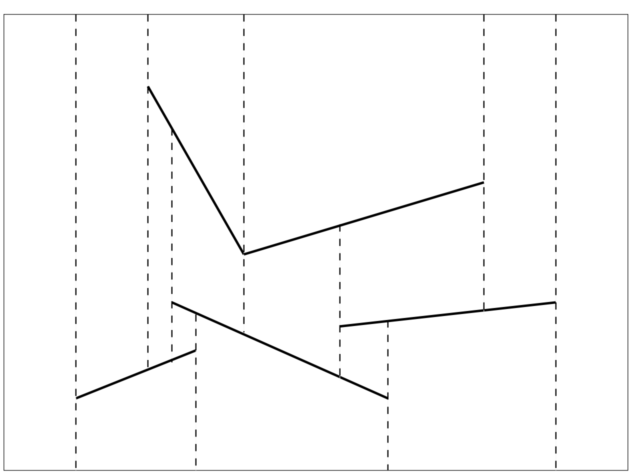
Problem: Polygon Triangulation

Given a simple polygon P with n edges, compute a triangulation of its interior.



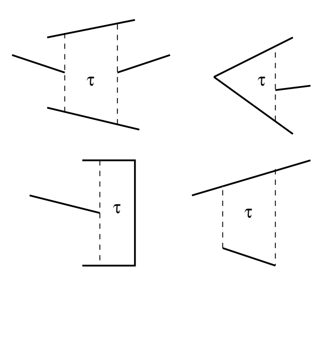
Solution via Trapezoidal Map

Given a set S of n nonintersecting segments in the plane, compute its *trapezoidal map*.



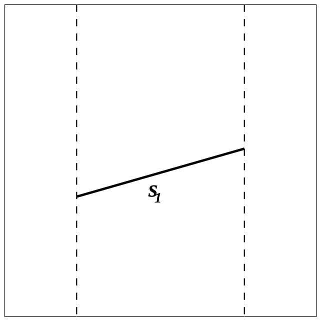
Trapezoidal Map

- planar graph, vertices V, edges E, faces F
- V: endpoints, artificial vertices
- E: pieces of segments, vertical extensions
- F: set of *trapezoids*, each one incident to at most 4 segments (assuming no two endpoints have the same x-coordinate; *not* true in triangulation application, but can be achieved even there)



Randomized Incremental Construction

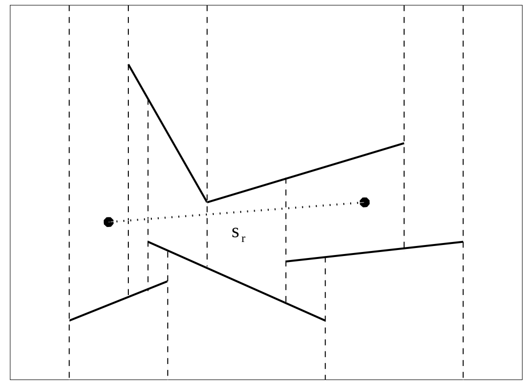
1. Compute trapezoidal map of $\{s_1\} \mapsto T_1$



2. Insert segments $s_2, \ldots, s_n$ in random order $\mapsto T_n$

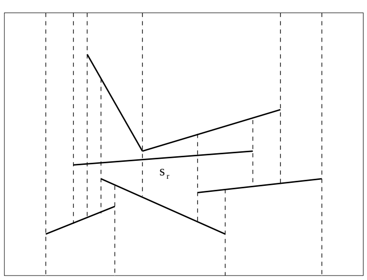
From T_{r-1} to T_r (I)

Find

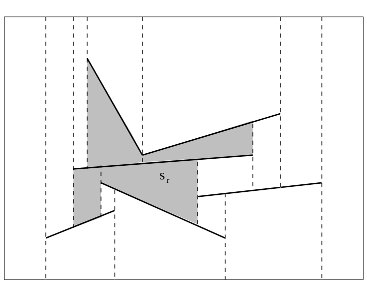


From T_{r-1} to T_r (II)

Split



Merge



From T_{r-1} to T_r (III)

1. **Find**: Find the trapezoid containing the left endpoint of s_r

2. **Split:** Trace s_r through T_{r-1} and split all the trapezoids intersected by s_r

3. **Merge:** Remove parts of vertical extensions "cut off" by s_r and merge the adjacent trapezoids

RIC – Analysis (I)

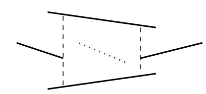
Apply configuration spaces!

- X: the set S of segments
- Π: set of all trapezoids □ defined by segments of S
- $D(\square)$: the (at most 4) segments incident to the trapezoid □
- $K(\square)$: the set of segments intersecting □

RIC – Analysis (II)

Cost of step $T_{r-1} \mapsto T_r$:

- **Find**: we'll care for that later...
- **Split**: constant time per traced □; □ is replaced by at most 4 new trapezoids.



$$\Rightarrow \; O(\text{number of removed trapezoids})$$
$$= \; O(\text{number of created trapezoids})$$

- **Merge:** O(number of trapezoids created in step **Split**)

Analysis of Update $T_{r-1} \mapsto T_r$ (I)

Observation: The number of trapezoids created by **Split** is at most twice as large as the number of new trapezoids in T_r.

Proof: For every **Merge** operation above (below) s_r, one new trapezoid below (above) s_r survives. It follows that at most half of the previously created trapezoids are not in T_r.

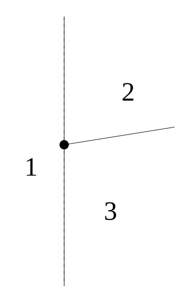
⇒ Complexity of **Split** and **Merge** is

$$O(|\{\square \mid \square \in T_r \setminus T_{r-1}\}|) = O(\deg(s_r, T_r)).$$

Analysis of Update $T_{r-1} \mapsto T_r$ (II)

Configuration Spaces ⇒ expected value of $\deg(s_r, T_r)$ is $\leq \frac{4}{r}E(|T_r|)$.

- $|T_r| \leq 6r$ (each □ is incident to a segment endpoint, and each endpoint is charged by at most three segments).



- Expected update cost $T_{r-1} \mapsto T_r$ is $O(1)$
- Overall expected update cost is $O(n)$

Realization of **Find**

- History approach: store all the trapezoids of $T_r, r = 1 \ldots n$. $\Box \in T_{r-1} \setminus T_r$ has pointers to all $\Box' \in T_r \setminus T_{r-1}$ with $\Box \cap \Box' \neq \emptyset$
- At most 4 pointers per $\Box$
- Location of segment endpoint p_r of s_r: trace p_r through the history graph

Analysis of **Find** (I)

Assume p_r runs through a trapezoid $\Box$. Then there is $j \leq r$ such that

- $\Box \in T_j \setminus T_{j-1}$
- s_r intersects $\Box$

$\Rightarrow$ length of history path to p_r

$$\leq \sum_{j=0}^{n-1} \sum_{\Box \in T_j \setminus T_{j-1}} [s_r \in K(\Box)]$$

$\Rightarrow$ expected time for history searches is proportional to the expected number $\sum_{r=0}^{n-1} K(r)$ of conflicts that appear during the algorithm.

Analysis of **Find** (II)

Configuration spaces $\Rightarrow$

$$\begin{aligned}\sum_{r=1}^{n-1} K(r) &\leq \sum_{r=1}^{n-1} (k_1(r) - k_2(r) + k_3(r)) \\ &\leq d(n-1)t_1 + \\ &\quad d(d-1)n \sum_{r=1}^{n-1} \frac{t_{r+1}}{r(r+1)} - \\ &\quad d^2 \sum_{r=1}^{n-1} \frac{t_{r+1}}{r+1} \\ &= O(n \log n),\end{aligned}$$

because

$$t_{r+1} = E(|T_r|) = O(r+1).$$

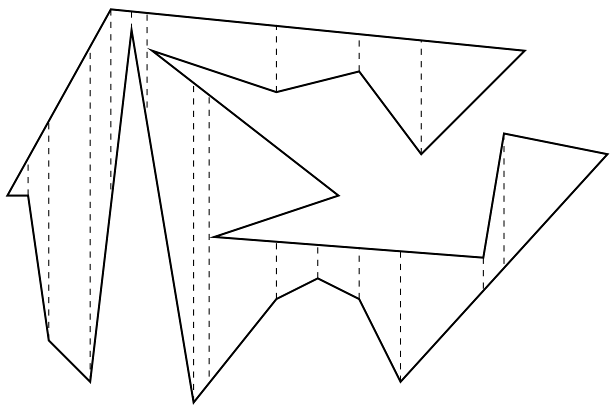
Trapezoidal Map – Conclusion

Given a set S of n nonintersecting segments in the plane, its trapezoidal map $T(S)$ can be computed in time

$$O(n \log n).$$

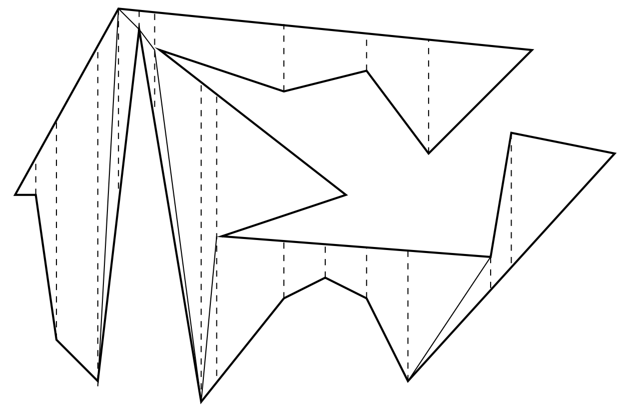
(The assumption that segment endpoints have different x-coordinates can be achieved by comparing them lexicographically.)

Special Case: S forms simple polygon P



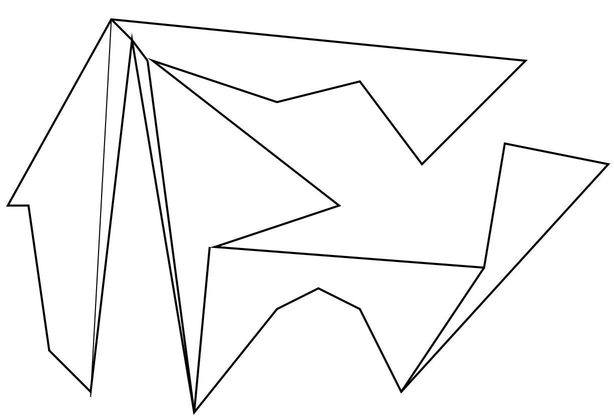
Trapezoidal Map $\rightarrow$ Triangulation (I)

Step 1: Within each trapezoid, connect the two polygon vertices



Trapezoidal Map $\rightarrow$ Triangulation (II)

Step 2: Triangulate the resulting x-monotone polygons separately, in total time $O(n)$ (Exercise)



A fast method for the special case (I)

Runtime will be $O(n \log^\star n)$.

- $\log^{(h)} n := \underbrace{\log\log\ldots\log n}_{h \text{ times}}$
- $\log^\star n := \max\{h \mid \log^{(h)} n \geq 1\}$
- Example: $\log^\star(2^{65536}) = 5 \Rightarrow \log^\star n < 5$ "for all" n.

Definition:

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

A fast method for the special case (II)

Generalized history management: keep several histories and for each $p \in P$ a pointer to the 'history in charge'.

```
compute T_1 and initialize one
history, in charge of all points
FOR h = 1 TO log* n DO
    FOR r = N(h-1)+1 TO N(h) DO
        compute T_r from T_{r-1} (* as usual *)
    END
    (* Renew histories by tracing S through T_r *)
    FOR ALL □ ∈ T_r containing an endpoint DO
        make □ the root of a history in charge
        of all the points it contains
    END
END
FOR r = N(log* n)+1 TO n DO
    compute T_r from T_{r-1} (* as usual *)
END
```


Analysis of the fast method (I)

- **Split** and **Merge** proceed as before in expected time $O(n)$
- **Find** will be faster on average, but we have
- $\log^\star n$ additional **Trace** steps

Analysis of **Find** (I)

In phase h, every trapezoid traced during the history search corresponds to a trapezoid that

- has been present in the beginning of phase h (root of a hierarchy) or was created during phase h
- is in conflict with a segment inserted in phase h

$\Rightarrow$ expected cost of history search is at most proportional to $n + K_h$,

$$K_h := \sum_{r=N(h-1)+1}^{N(h)} \sum_{\Box \in T_r \setminus T_{r-1}} |K(\Box) \cap S_{N(h)}|.$$

Analysis of **Find** (II)

For fixed $X := S_{N(h)}$, $E(K_h)$ is the expected number of conflicts appearing in steps $N(h-1)+1$ to $N(h)$ when $T(X)$ is computed.

$$i := N(h-1)+1, \quad j := N(h)-1.$$

Configuration spaces analysis $\Rightarrow$

$$\begin{aligned} E(K_h) &\le \sum_{r=i}^{j}(k_1 - k_2 + k_3) \\ &\le \frac{d(j+1-i)}{i} t_i + \\ &\quad d(d-1)(j+1)\sum_{r=i}^{j}\frac{t_{r+1}}{r(r+1)} - \\ &\quad d^2 \sum_{r=i}^{j}\frac{t_{r+1}}{r+1}. \end{aligned}$$

Analysis of **Find** (III)

Recall:

$$t_{r+1} = O(r+1).$$

Then

$$\begin{aligned} E(K_h) &= O\left(N(h) - N(h-1)\right) + \\ & \quad O\left(N(h) \sum_{r=N(h-1)+1}^{N(h)-1} \frac{1}{r}\right) \\ &= O\left(N(h) + N(h) \log \frac{N(h)}{N(h-1)}\right) \\ &= O\left(N(h) + N(h) \log^{(h)} n\right) \\ &= O(n). \end{aligned}$$

(This also holds for a random set $S_{N(h)}$ and for the last insertion phase ($i = N(\log^\star n) + 1, j = n-1$).) The total cost for **Find** over all h is then $O(n \log^\star n)$.

Analysis of **Trace** (I)

The expected cost T_h of tracing S through $T_{N(h)}$ is at most proportional to the expected number of conflicts between trapezoids in $T_{N(h)}$ and segments in S, which is

$$\frac{1}{\binom{n}{N(h)}} \sum_{R \subseteq S, |R| = N(h)} \sum_{y \in S \setminus R} |\{\square \in T(R) \mid y \in K(\square)\}|.$$

Up to a missing factor of $d/N(h)$ this is exactly the bound for the expected number $K_{N(h)}$ of new conflicts when $s_{N(h)}$ is inserted that we derived from the *configuration spaces*.

Analysis of **Trace** (II)

	configuration spaces	here
k_1	$\frac{d}{r}(n-r)t_r$	$(n-r)t_r$
k_2	$\frac{d}{r}(n-r)t_{r+1}$	$(n-r)t_{r+1}$
k_3	$\frac{d^2}{r(r+1)}(n-r)t_{r+1}$	$\frac{d}{r+1}(n-r)t_{r+1}$

Setting $r = N(h)$, we obtain $T_h = k_1 - k_2 + k_3$ as

$$\begin{aligned} T_h &\leq (n - N(h))t_{N(h)} - \\ & \quad (n - N(h))t_{N(h)+1} + \\ & \quad \frac{d}{N(h)+1}(n - N(h))t_{N(h)+1} \\ &= O\left(n(t_{N(h)} - t_{N(h)+1}) + n\right) \\ &= O(n), \end{aligned}$$

because $t_{N(h)} \leq t_{N(h)+1}$.

The total cost for **Trace** over all h is then $O(n \log^\star n)$.

Fast Trapezoidal Map – Conclusion

Given a simple polygon P with n vertices in the plane, its trapezoidal map $T(P)$ can be computed in time

$$O(n \log^\star n).$$

(This is not optimal, because Chazelle has given a (rather complicated) $O(n)$ algorithm for the problem.)