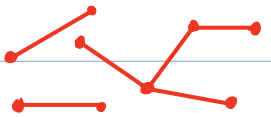
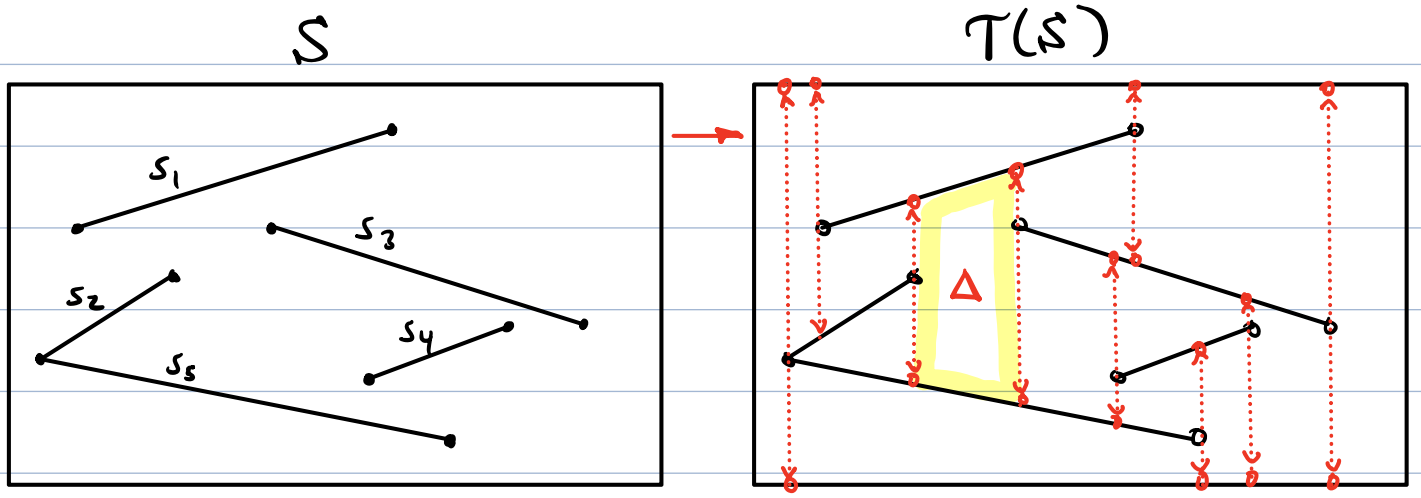
CMSC 754 - Computational Geometry
Lecture 8: - Trapezoidal Maps

Trapezoidal Maps:

- Given a set $S = \{s_1, \ldots, s_n\}$ of line segments in $\mathbb{R}^2$ which we assume do not intersect (except at their endpoints)



- General Position: No duplicate coordinate values (excluding common endpoints)
- Enclose in a large bounding rectangle
- Shoot a bullet path up + down from each endpoint
  - Stop at first intersection



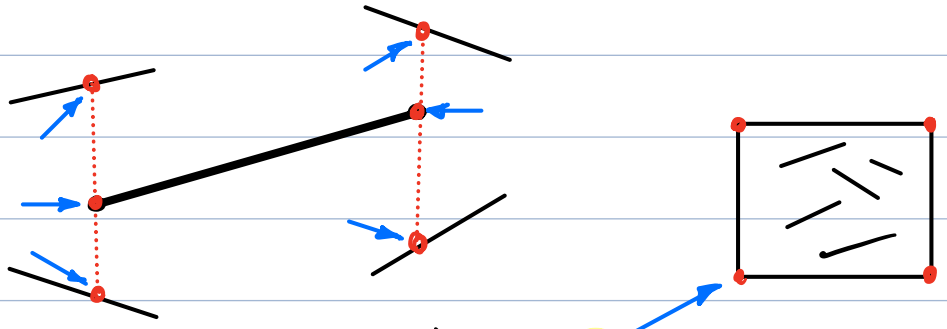
The resulting subdivision (into trapezoids + triangles) is called the trapezoidal map of S, denoted $T(S)$

- Represented as DCEL data structure

Lemma: Given a set S of n segments, $\Upsilon(S)$ has $\leq 6n+4$ vertices + $\leq 3n+1$ cells (trapezoids)

Proof:

- Each segment contributes ≤ 6 vertices to $\Upsilon(S)$

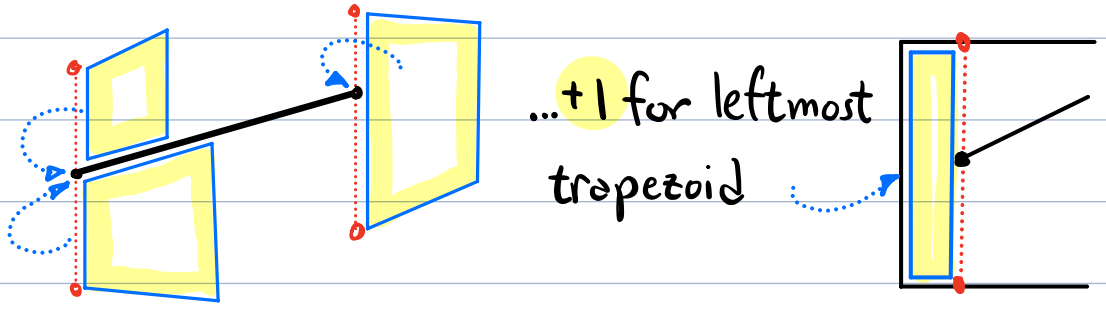


... and the bounding box contributes 4

For a total of $\leq 6n+4$ vertices

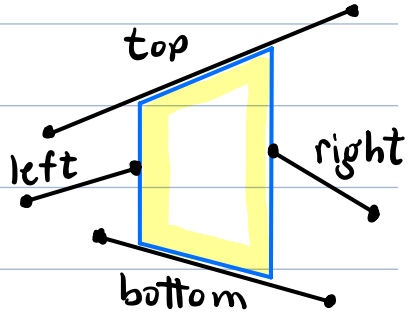
- Charge each trapezoid to its left-side seg. endpoint

  ⇒ Each segment charged at most 3 times



⇒ Total of $3n+1$ trapezoids □

Observation: Each trapezoid owes its existence (i.e., depends) on ≤ 4 segments.



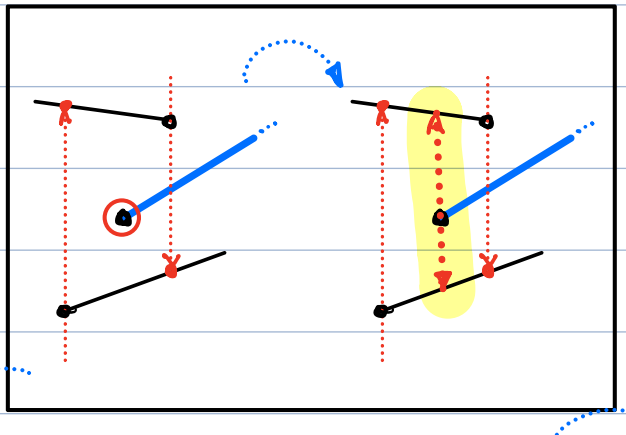
Construction:

- Plane sweep - $O(n \log n)$ [exercise]
- Randomized incremental - $O(n \log n)$ expected

Randomized Incremental Construction (RIC)

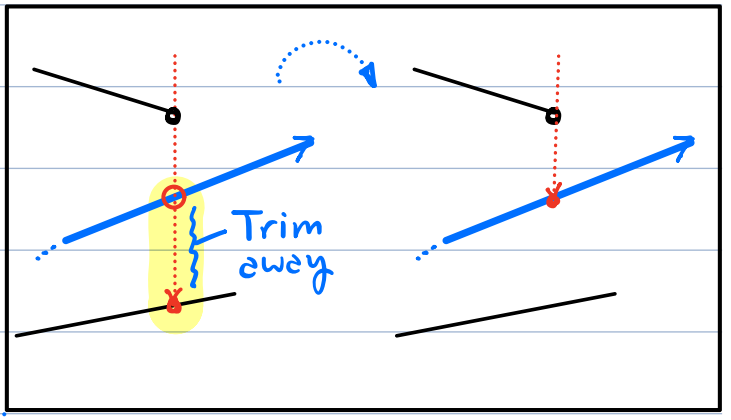
- Add segments one-by-one in random order
- Updates:
  - Endpoint: Shoot 2 bullet paths
  - Cut existing wall:
    - Trim the wall back

Segment endpoint



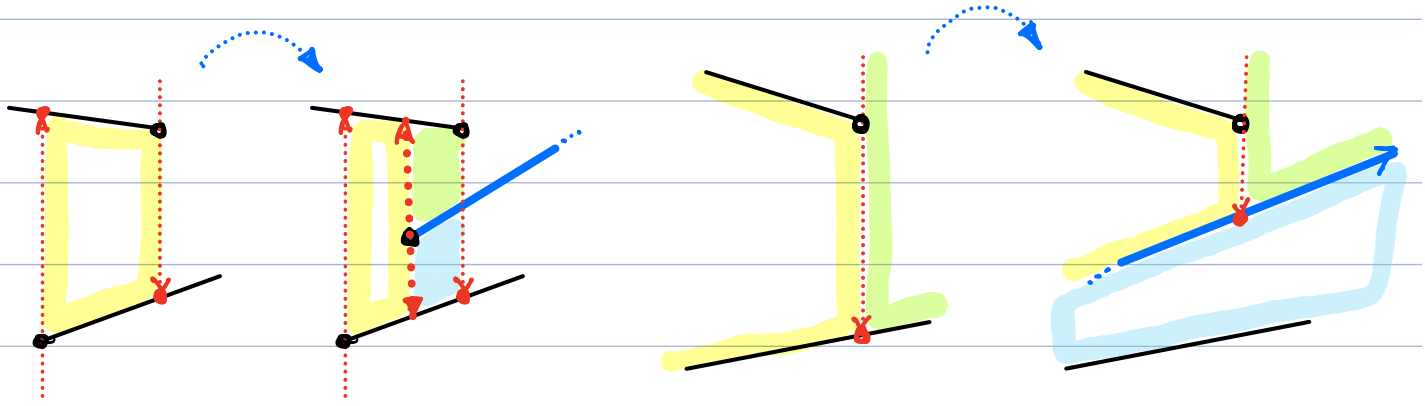
- Find cell containing endpoint
- Add bullet path to cell's floor & ceiling
- Create new cells

Cut an existing wall

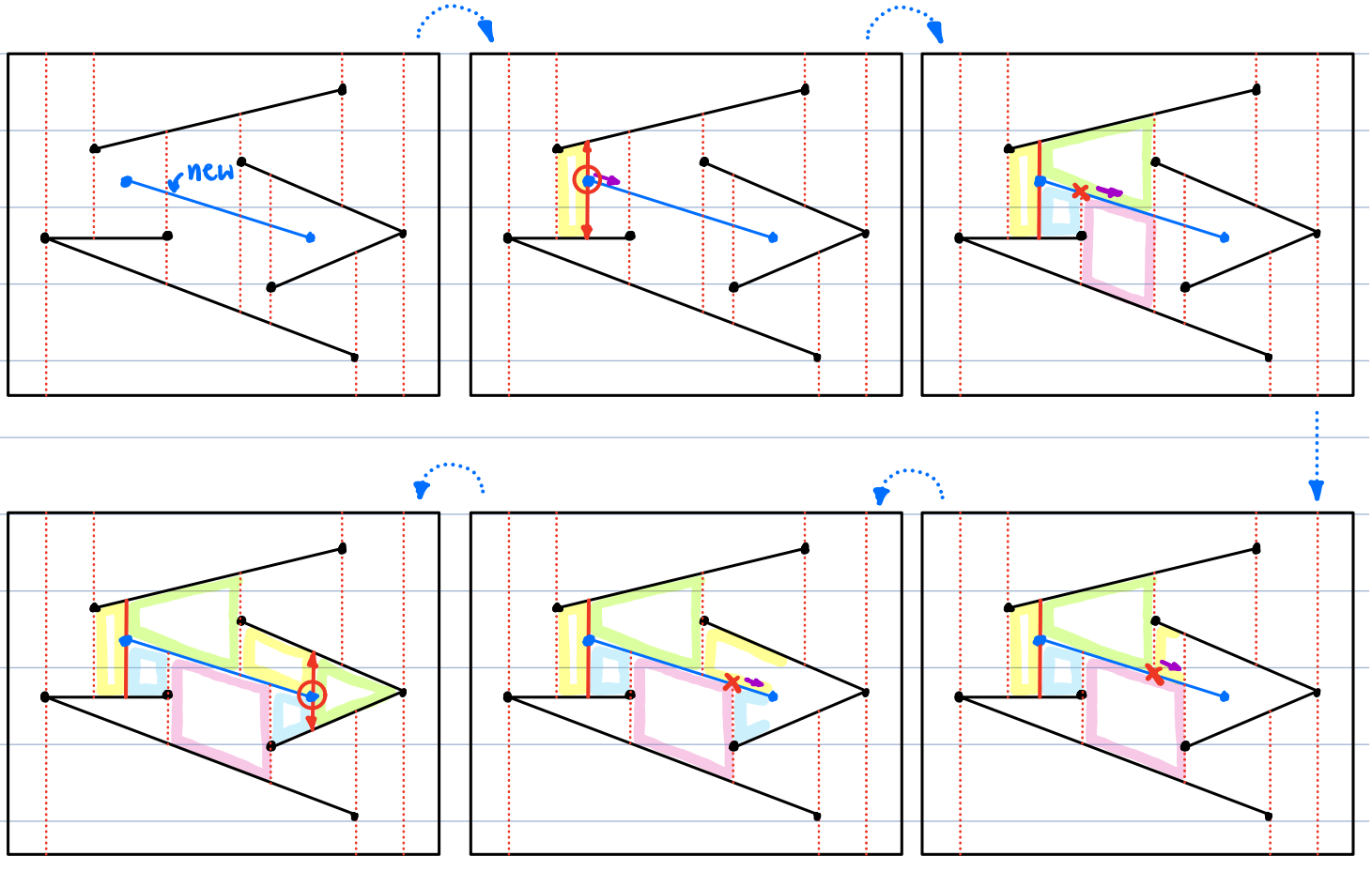


- Trim wall back to the intersection point (keep the side with seg. endpoint)
- Split/Merge cells

These updates create new cells and destroy old ones



Example:



Note: All updates (add vertex, trim wall, create/destroy trapezoid) can be done in O(1) time with DCEL

Running Time: (to insert segment s_i)

called point location

- Find cell containg s_i's left endpoint
  - → We'll show (next lecture) this takes $O(\log n)$
- Trace segment, trim walls, create cells
  - → Proportional to 1+(no. of walls hit)
  - ≈ No. of new cells created

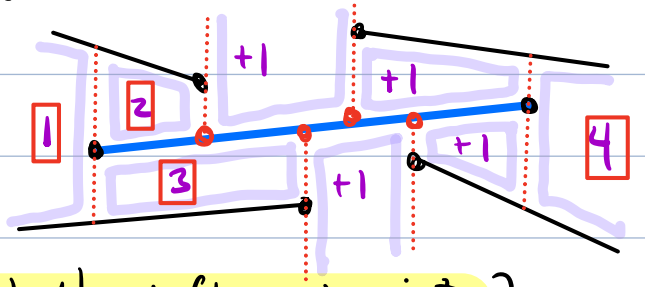
Lemma: For $1 \le i \le n$, let k_i denote the number of new trapezoids created (including trimmed + merged) when s_i is inserted. The time to update the subdivision (DCEL) is $O(k_i)$.

Note: k_i is a random variable (dep. on insert order)

Proof: Let w_i denote the number of walls hit by s_i. We claim that the number of new cells is $\le 4 + w_i$

Eg. $w_i = 4$

$k_i = 8$



$k_i =$

3 new cells around the left endpoint

+ 1 new cell for each wall

+ 1 new cell at right endpoint

$= 4 + w_i$

Since DCEL supports updates in $O(1)$ time, total update time is:

$$2 + 2 + w_i + k_i = 4 + (k_i - 4) + k_i$$
$$= 2k_i = O(k_i)$$

- 2: Left bullet paths
- 2: Right bullet paths
- w_i: No. of trimmed walls
- k_i: No. of new cells

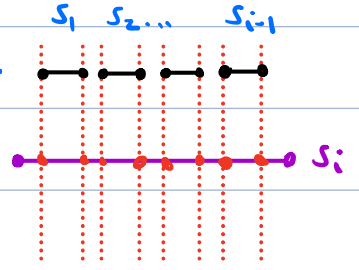
□

Overall Running Time:

Point-location time (next lecture shows $O(n \log n)$)

+ Structure update time

$$= \sum_{i=1}^{n} k_i$$



Obs: Segment i can hit up to $2(i-1)$ walls

$\Rightarrow \sum k_i = \sum 2(i-1) = O(n^2)$



Too slow!

Randomization to the Rescue!

Expected value

- We'll show that if segs. are inserted in random order, then $E(k_i) = O(1)$

$\Rightarrow$ Total expected update time is

$$E\left(\sum k_i\right) = \sum E(k_i) = \sum_{i=1}^{n} O(1) = O(n)$$



Lemma: Assuming segments are inserted in random order, $E(k_i) \leq 12 + 4/i \leq 16 = O(1)$

Proof: (by backwards analysis)

Let $S_i = \{s_1, \ldots, s_i\}$ (fix i, $1 \leq i \leq n$)

Let τ_i = trapezoidal map after inserting S_i

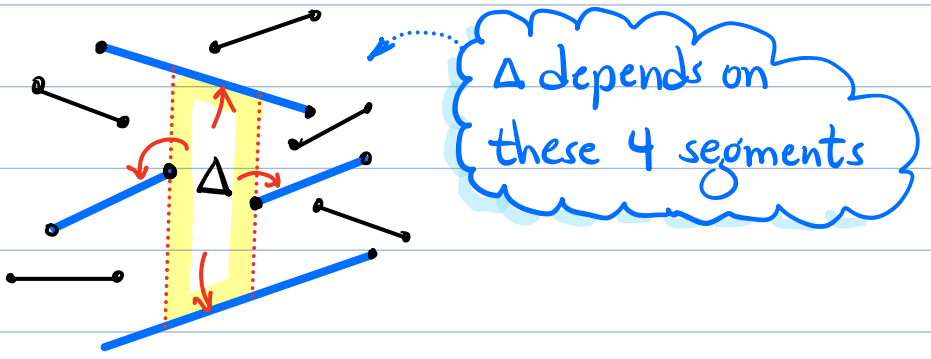
Obs: τ_i does not depend on insert order.

For any $s \in S_i$, define $p(s)$ = probability s inserted last in S_i

Since order is random, each seg. of S_i is equally likely to be last inserted.

$\Rightarrow p(s) = 1/i$

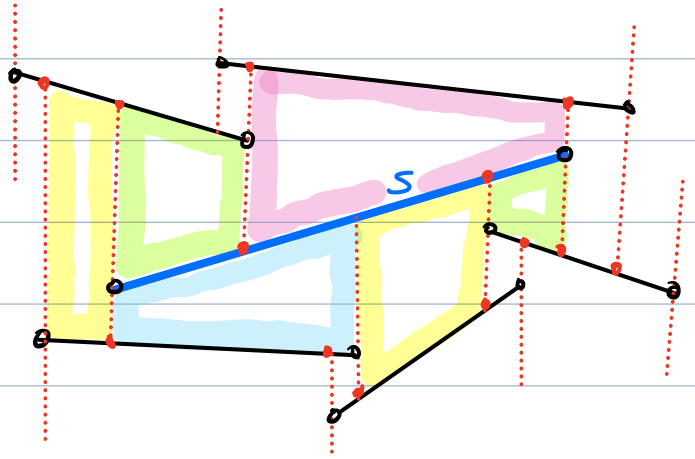
Given any cell $\Delta \in \tau_i$ & any $s \in S_i$, we say Δ depends on s if Δ would have been created if s was inserted last.



Notation: $d(\Delta, s) = \begin{cases} 1 & \text{if } \Delta \text{ depends on } s \\ 0 & \text{o.w.} \end{cases}$

Obs:

A given cell depends on at most 4 segments,
but many cells may depend on a given segment.



$\{\Delta \mid \delta(\Delta, s) = 1\}$

Note: "depends" ≡ "incident to"

Also $\delta(\Delta, s)$ is independent of insert. order

We can express $E(k_i)$ as:

$$E(k_i) = \sum_{s \in S_i} \left(\text{prob } s \text{ inserted last}\right) \cdot \left(\text{no. of cells that depend on } s\right)$$

$$= \sum_{s \in S_i} p(s) \cdot \sum_{\Delta \in \tau_i} \delta(\Delta, s)$$

$$= \sum_{s \in S_i} (1/i) \cdot \sum_{\Delta \in \tau_i} \delta(\Delta, s) \quad (\text{since } p(s) = 1/i)$$

$$= (1/i) \sum_{s \in S_i} \sum_{\Delta \in \tau_i} \delta(\Delta, s)$$

$$= (1/i) \sum_{\Delta \in \tau_i} \sum_{s \in S_i} \delta(\Delta, s) \quad \} \text{ (rearrange sum order)}$$

Recall: A given cell Δ depends on ≤ 4 segs

$$\Rightarrow \sum_{s \in S_i} d(\Delta, s) \leq 4$$

$$\Rightarrow E(k_i) \leq (1/i) \sum_{\Delta \in \tau_i} 4 = (4/i) \sum_{\Delta \in \tau_i} 1$$

$$= (4/i)(\text{No. of cells in } \tau_i)$$

By previous lemma, no. cells in $\tau_i \leq 3 \cdot i + 1$

$$\Rightarrow E(k_i) \leq (4/i)(3 \cdot i + 1)$$

$$= 12 + 4/i$$

□

Whew!

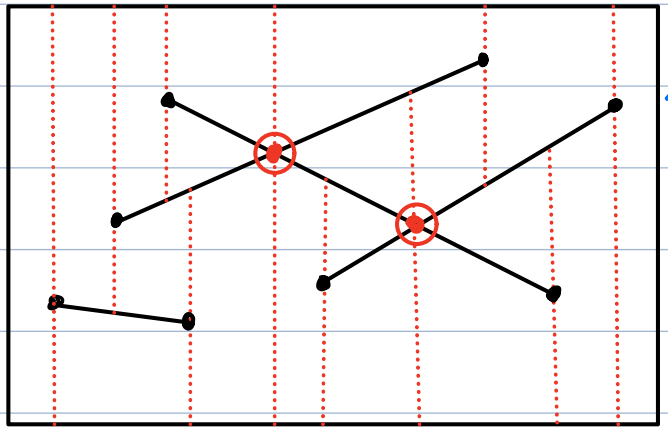
Takeaway - Although each segment may have high dep. count (i.e., it touches many cells) the total number of cells is $O(i) \Rightarrow$ Each of the i segs. touch $O(i)/i = O(1)$ cells on average.

Line Segment Intersection (Revisited)

- We showed that we could compute the m intersections of n lines in $O((n+m) \log n)$ time.
- We also showed lower bound of $\Omega(n \log n + m)$

- We left as an open problem to improve the upper bound to match the lower bound.

- Solution: Modify the RIC algorithm to handle segment intersections.



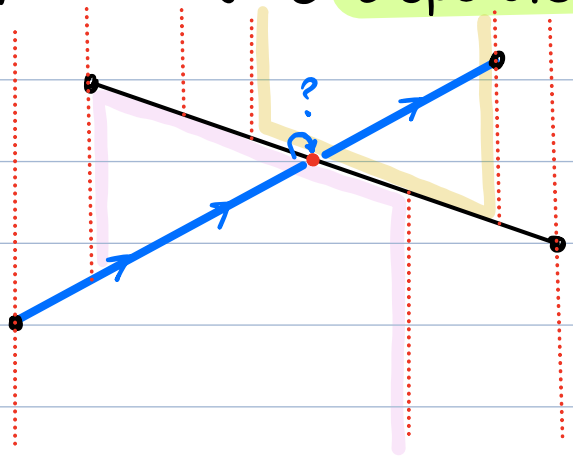
Modifications:

- Think of intersections as splitting each segment into subsegments
- There are O(n+m) subsegments

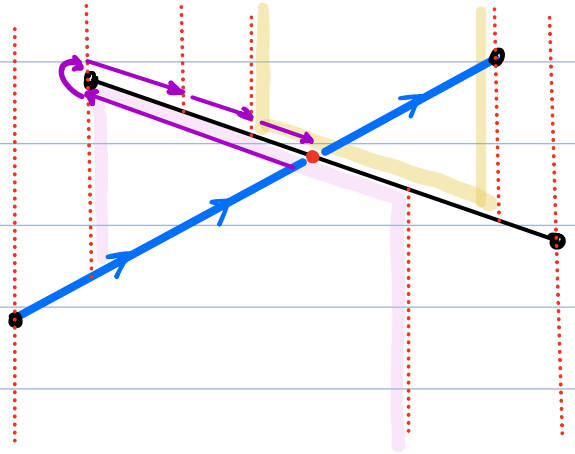


- The above analysis implies the expected number of structural updates is O(no. of subsegs) = O(n+m)

- What changes?
  - Whenever intersection detected, need to find trapezoid to continue with on other side of segment



  - Do this by walking along the segment from one side to the other



  - Can show that if segs. inserted in random order - expected time is $O(n+m)$

Summary:
- Can build trap. map in expected linear time (ignoring time to locate starting trapezoid.)
- Next - Finding start traps. in $O(n \log n)$ time.