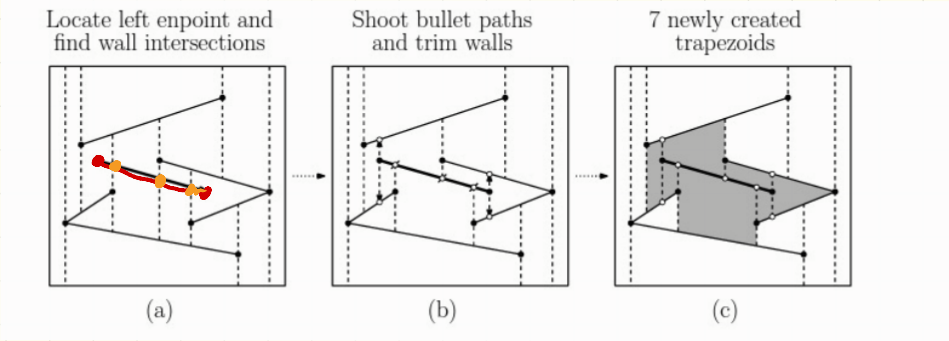
Trap Map (pt 2)

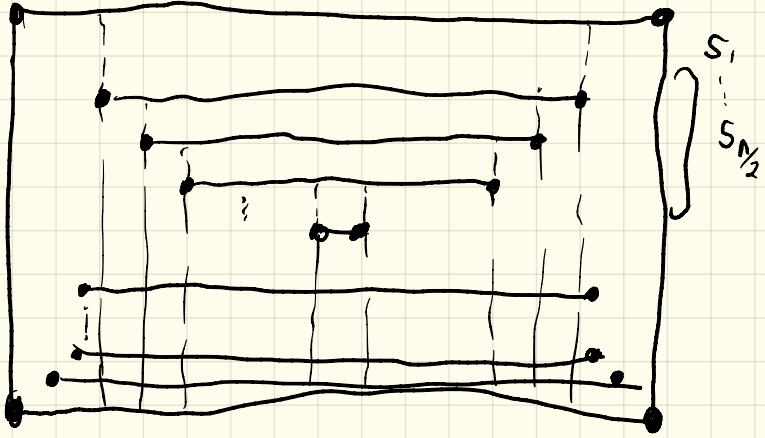
2/25



Alg:
- start w/ 1 trap (bounding box of all segments
- add segments in random order
- for each segment
  insert segment & upate trap map

Update:
let $S_i = \{s_1, \ldots, s_i\}$
let T_i be the trap map of S_i
add s_{i+1}

1. find trap w/ left endpoint of s_{i+1}
2. walk through the map to find the right endpoint of s_{i+1} → all the traps that need to be updated
3. fix up the traps!
   a. left/right endpoints of s_{i+1} add rays
   b. trim back walls



$\frac{n}{2}$ segs cutting through $\frac{n}{2}$ traps $\Rightarrow \Omega(n^2)$ - time

★ Structure of the decomp does not depend on the segment input order

Claim: ignore the time spent for location of the left endpoint
the time to add the i^{th} segment
and update the map is $O(k_i)$
w/ k_i is the # of new traps

Pf consider the i^{th} segment
let K be the # of walls that s_i intersects
- shoot 4 rays ℓ
- trim K walls $\Rightarrow K+4$ ops

if a segment crosses no walls 
we get 4 traps
and 1 new trap for every wall crossed $\rightarrow K+4$ traps
Let $k_i = K+4$
# of traps updated at insertion of s_i
and each op performed in $O(1)$ time
(if we use a data struct like a PCEL)

2 steps for insertion
- left point location steps → (we will see later) expected $O(n \lg n)$ for all locations
- trimming walls from new segments

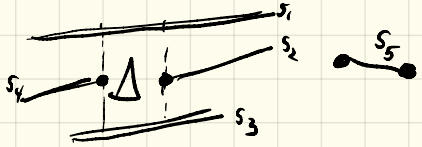
Lemma: Consider a RIC for the trap map
let k_i denote the # of new traps added by i^{th} segment
$E(k_i) = O(1)$
↖ expected value of k_i take over all permutations of insertion order

Pf Backwards analysis
idea: assume a random order of segments
how much stuff is created from removing

T_i be the trap map after inserting the i^{th} segment
(over all permutations)
⇒ any segment is S_i - has $1/i$ prob of being i^{th} segment

for any segment S count # of traps created if S was inserted last

def trap Δ as depending on S
S would cause Δ to be created if S was added last



$\delta(\Delta, S_1) = 1$
$\delta(\Delta, S_5) = 0$

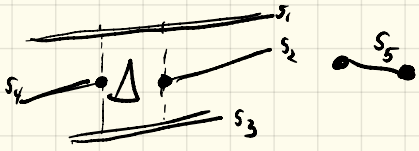
define an indicator random variable
$$\delta(\Delta, S) = \begin{cases} 1 & \text{if } \Delta \text{ depends on } S \\ 0 & \text{otherwise} \end{cases}$$

$$E(k_i) = \frac{1}{i} \sum_{s \in S_i} \#\text{ of traps that depend on } s = \frac{1}{i} \sum_{s \in S_i} \sum_{\Delta \in T_i} \delta(\Delta, s)$$

what about the # of segs each trap depends on

$$E(k_i) = \frac{1}{i} \sum_{\Delta \in T_i} \underbrace{\sum_{s \in S_i} \delta(\Delta, s)}_{\text{must be 4 or smaller}} \leq \frac{1}{i} \left(\sum_{\Delta \in T_i} 4 \right) = \frac{1}{i} 4 |T_i| = \frac{1}{i} 4\alpha(i) = \alpha(1)$$

($|T_i|$: # of traps in trap map from the learn) ✓



$\delta(\Delta, s_1) = 1$

$\delta(\Delta, s_5) = 0$

$\Rightarrow$ expected total # of traps created in the life of the algo is $\alpha(n)$

Datastructure:
for planar point location

- Construct DS: expected $O(n \lg n)$ time
- size: expected $O(n)$ space
- locations: expected $O(\lg n)$ - time

Data structure
- Directed acyclic graph (DAG)
- each node has 0 or 2 outgoing edges
  - 0 → leaves (1-1 correspondence w/ traps in the map)
  - 2 → internal nodes

2 types of internal nodes
- x-node: defined by segment endpoint p is the query point to the left or right of p
- y-node: ptr to a segment s: is query point above or below s