Asymptotics:

(Q1) WHAT (do I want to study)

- * worst-case time/space
- expected/average case time/space
- best-case (not very helpful) time/space

(Q2) HOW?

- upper-bound : big-O $O(f(n))$ ← input size
- lower-bound: Ω
- tight-bound: Θ = both $O(f(n))$ and $\Omega(f(n))$

Complexities of:

- Algorithm: line-by-line
  loop-by-loop
- Problem (e.g., searching in a sorted array)
  ↳ over all ways to solve the prob., give the complexity of the best sol'n.

e.g., Prob: searching in sorted array

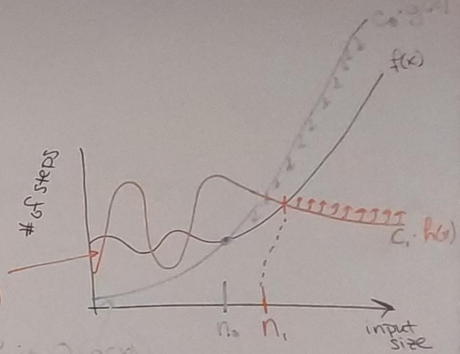
- Alg Sol'n 1: linear search ← worst-case: $\Theta(n)$

⇓

The prob is $O(n)$.

But, we know, the prob is $\Theta(\log n)$

f is $O(h(x))$



f(x) represents the answer to Q1 for various size inputs

e.g., worst-case:

f(x) is the max # of steps taken in computing the ans when the input size is x.

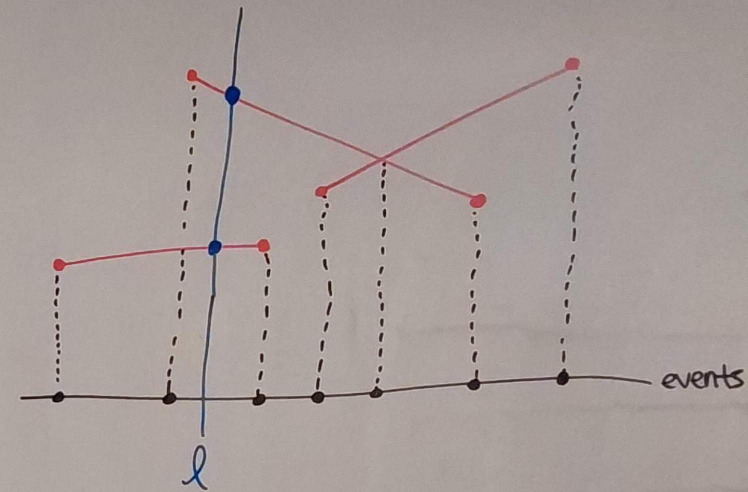
Line Segment Intersection

Problem:

Given a set of line segments

$S = \{ s_i = \overrightarrow{p_i\, q_i} \}$ — s_i ← segment, p_i ← left endpt, q_i ← right endpoint

Find: all pairwise intersections of segments



Naive Sol'n: test all pairs

$\Rightarrow \Theta(n^2)$ is the worst-case RT of naive sol'n

$\therefore$ since $\exists\, \Theta(n^2)$ sol'n, the prob is $O(n^2)$.

★ our sol'n will be output sensitive

$m :=$ the number of intersections

DS for events:

priority queue

- priority = x-coord (lower is higher priority)
- event: endpoint of segment
  intersection of 2 segments

Operations

$$\overset{\text{reference}}{r} = \text{insert}(\overset{\text{event}}{e}, \overset{\text{priority}}{x})$$

- delete (r)
- e = extract-min

} $\Theta(\log n')$ time, $\Theta(n')$ space, n' = # elts in the priority queue

e.g., binary heap

note: throughout execution, $n' \leq 2n-1$

Sweepline:

ordered dictionary

- ordered by y-coord on ℓ
- only contains $S \cap \ell$

Operations

- r = insert (s)
- delete (s)
- swap (s) ← swap s w/ below (s)
- below(s) / successor (s)
- above (s) / predecessor (s)
- find (s)

} if n' are in DS, then $\Theta(\log n)$ time, $\Theta(n)$ space is needed.

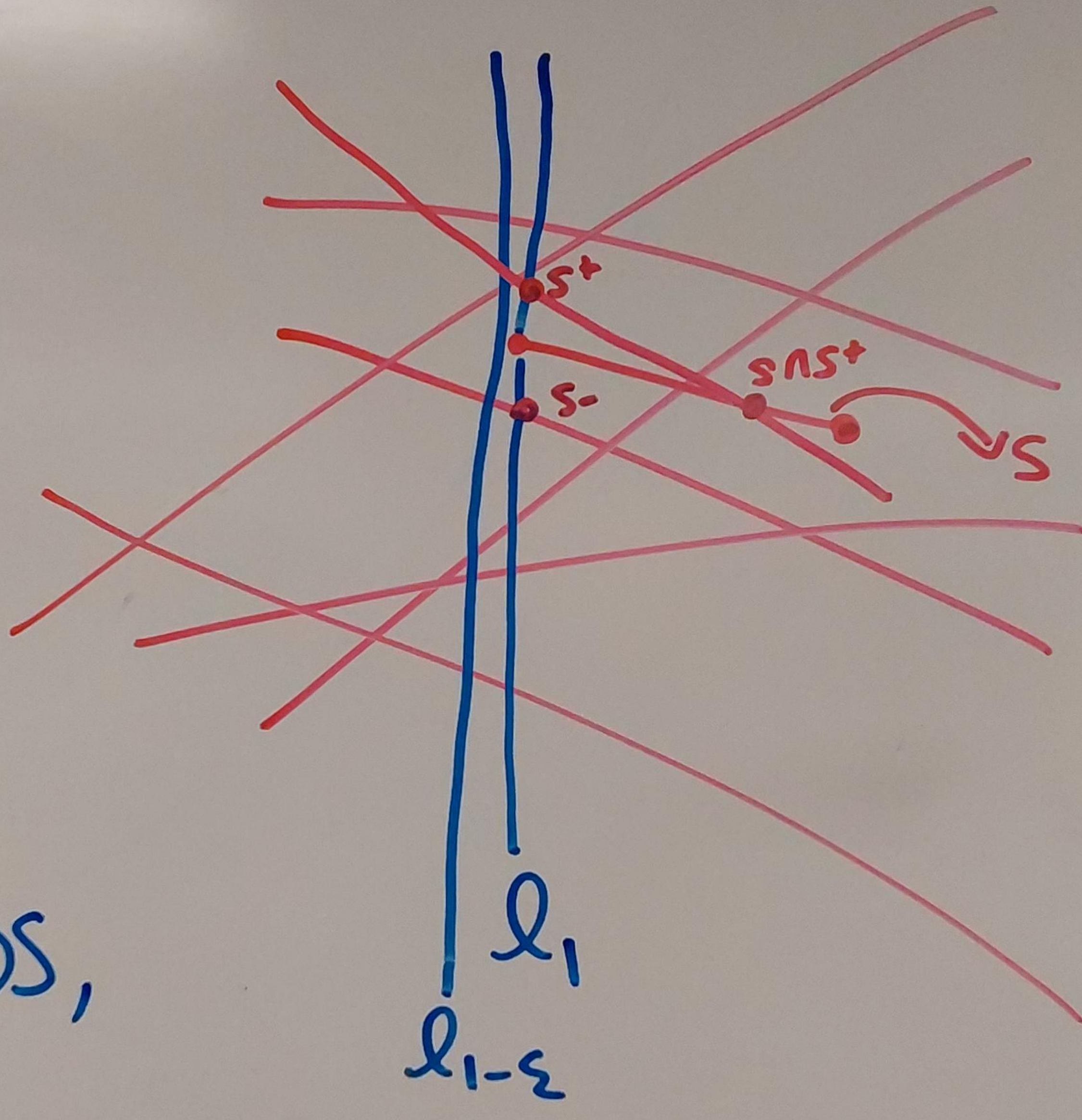
e.g., bal BST

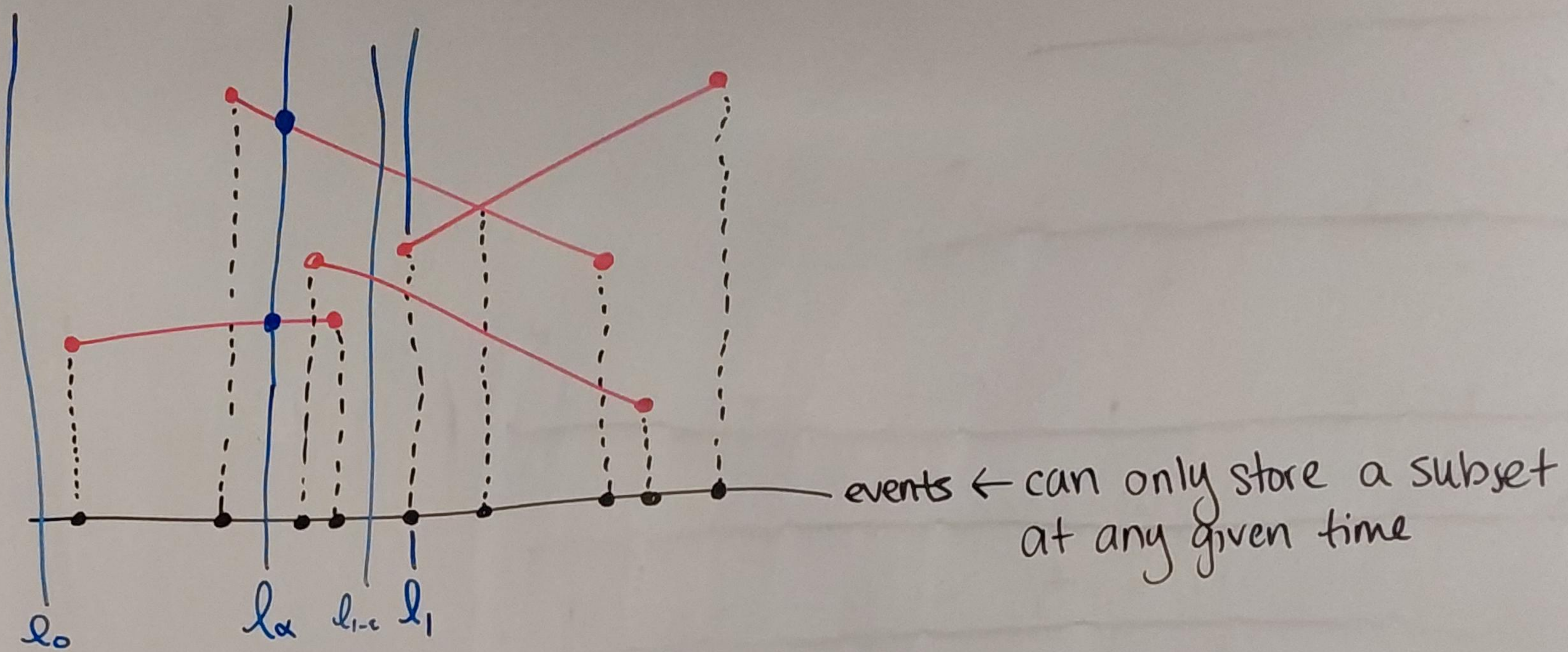
note: might need to use references instead of segments themselves.

n' are in DS,

en

)(logn) time





n segments $\Rightarrow$ $2n$ endpoints

s_i $\{p_i\}$ $\{q_i\}$

m intersections, $m < n^2$

Algorithm

$\ell \leftarrow$ initialized (empty)

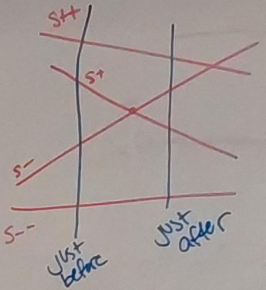
$Q \leftarrow$ priority queue, initialized to $\{p_i\} \cup \{q_i\}$

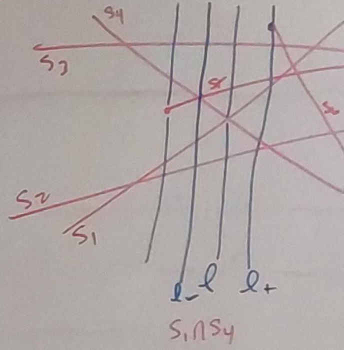
while $(r,e) \leftarrow Q$.extract-min

- if $e \in \{p_i\}$
  - $r \leftarrow \ell$.insert(e)
  - $s^+ \leftarrow \ell$.above(r)
  - $s^- \leftarrow \ell$.below(r)
  - check $r.s \cap s^+$ and $r.s \cap s^-$
    - if intersects to right of line, add it to the queue
  - if $s^- \cap s^+ \in Q$, remove it
- if $e \in \{q_i\}$
  - $s^+ \leftarrow \ell$.above(r)
  - $s^- \leftarrow \ell$.below(r)
  - add $s^+ \cap s^-$ to Q if $\cap$ to right of ℓ
- if e is an intersection
  - report the intersection
  - $s^+ \leftarrow$ above(r)
  - $s^- \leftarrow$ below(r)
  - $s^{++} \leftarrow$ above(s^+)
  - $s^{--} \leftarrow$ below(s^-)
  - Check $s^{++} \cap s^-$ and $s^{--} \cap s^+$ to add to Q
    - if $\cap$'s are to right of ℓ
  - swap(r)
  - if $s^{++} \cap s^+$ or $s^{--} \cap s^- \in Q$, remove them

end while

return





$\Theta((n+m)\log n)$ time

$\Theta(n)$ space