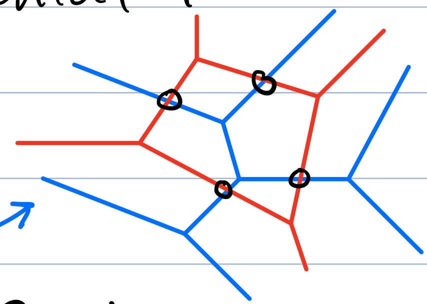
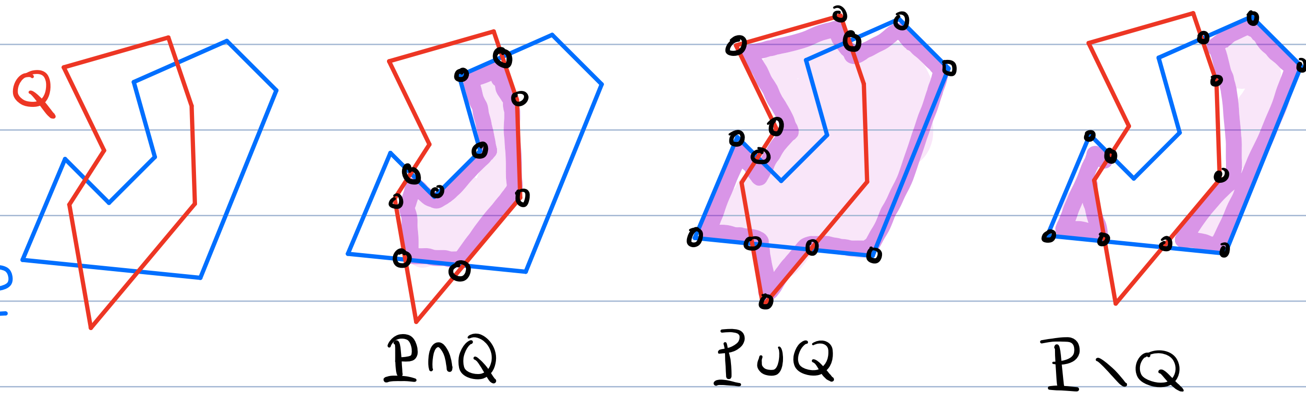
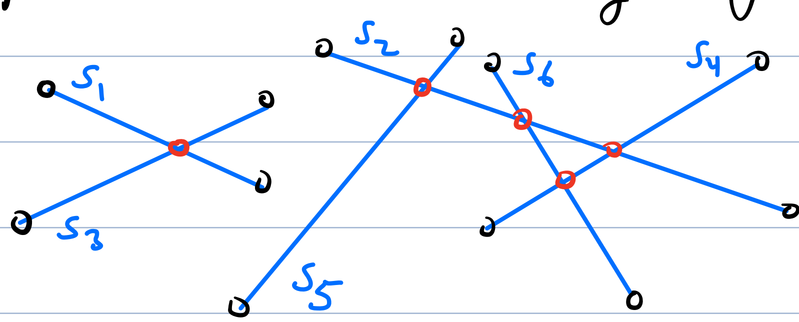
CMSC 754 - Computational Geometry
Lecture 4: Line Segment Intersection

Computing intersections is fundamental to geometric computation
- collision detection
- subdivision overlay
- boolean operations – $\cap, \cup, \ldots$





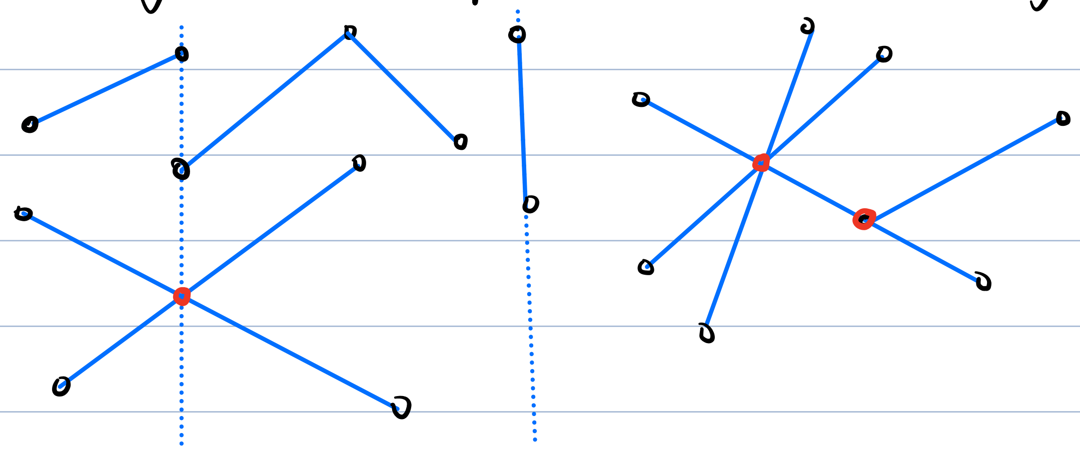
Line Segment Intersection:
Given a set $S = \{s_1, \ldots, s_n\}$ of line segments in $\mathbb{R}^2$ (where $s_i = \overline{p_i q_i}$), report all pairs of intersecting segments.



(s_1, s_3)
(s_2, s_5)
(s_2, s_6)
$\vdots$

General Position Assumptions:

- No duplicate x-coords (for both endpoints + intersections)
- No segment endpt on another segment



Output Sensitivity:

Input size: n (2n endpts, 4n coords)

Output size: m

$$0 \le m \le \binom{n}{2} = O(n^2)$$

Best possible: $O(m + n\log n)$

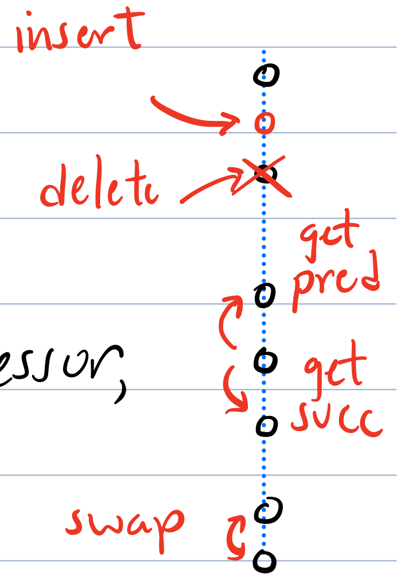
Follows from a lower bound on element uniqueness

This lecture: $O((n+m)\log n)$

Plane sweep

Utility Data Structures:

Ordered Dictionary: Supports:
insert, delete, find,
get-predecessor, get-successor,
swap adjacent

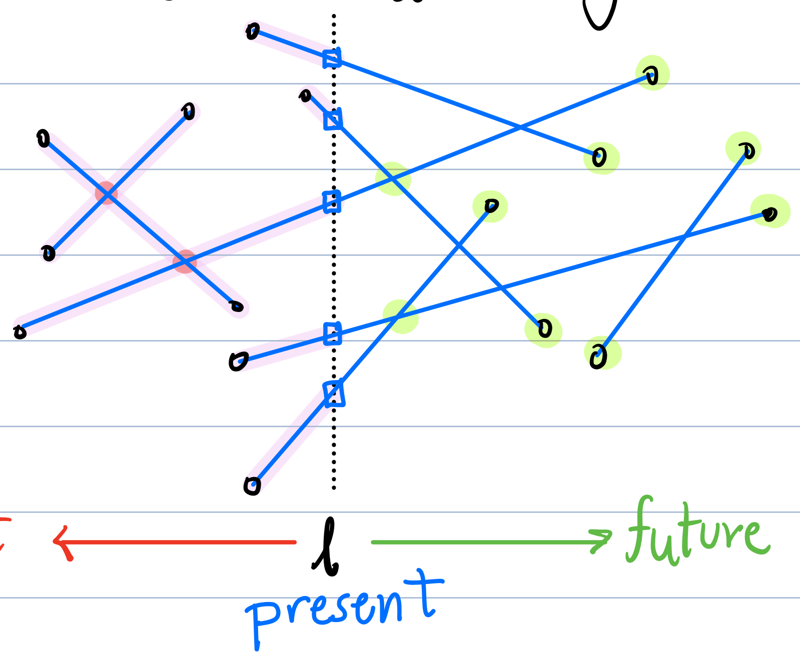


all in $O(\log n)$ time + $O(n)$ space

Priority Queue: Stores object o + priority x
ref $\leftarrow$ enqueue(o, x)
$o \leftarrow$ extract_min() - removes obj w. min priority
delete(ref)

Sweep-Line Algorithm:

Sweep a vertical line ℓ from **left to right** + update solution as we go.

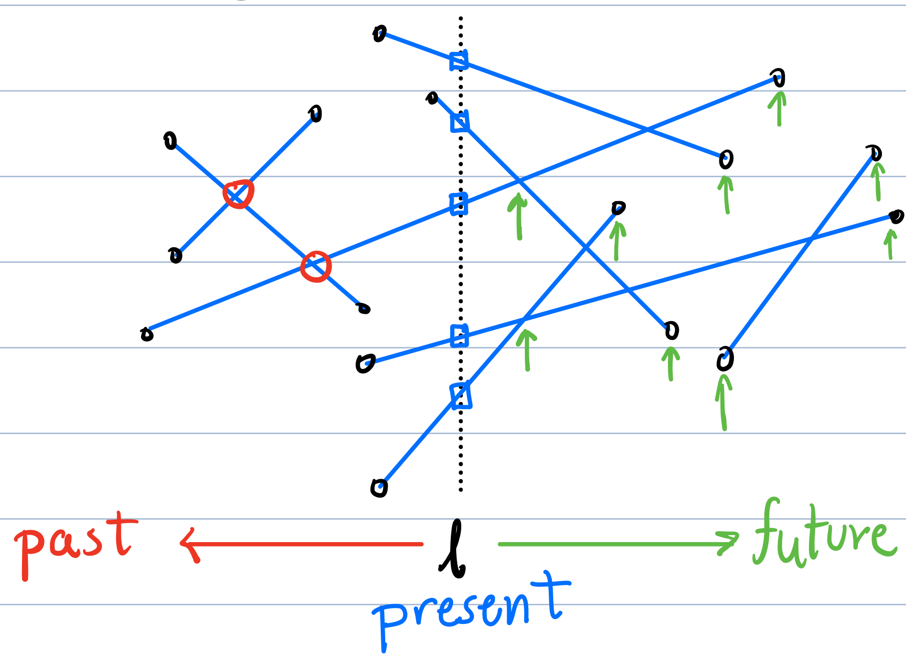


What we store: (Generic Plane Sweep)

(Past) Partial solution to left of ℓ

(Present) Current status along ℓ

(Future) (Known) Events to right of ℓ



What we store: (For segment intersection)

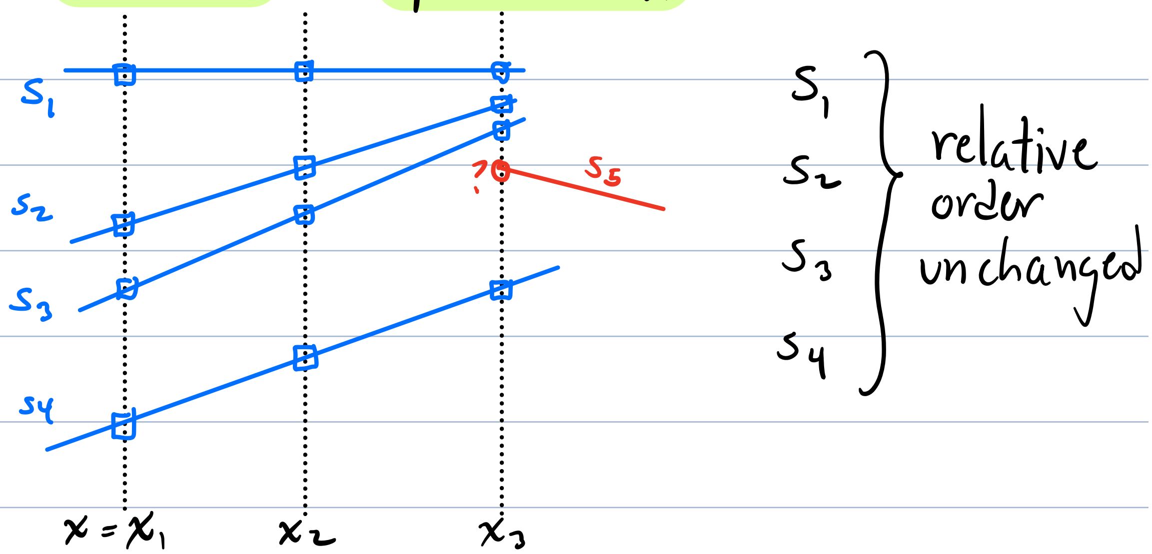
Past: List of intersecting pairs so far

Present: Ordered dictionary (top to bottom, say) of segments intersecting ℓ
— sweep-line status

Future: Priority queue with future events:
- segment endpts to right of ℓ
- "imminent" intersections right of ℓ

Sweep-Line Status:

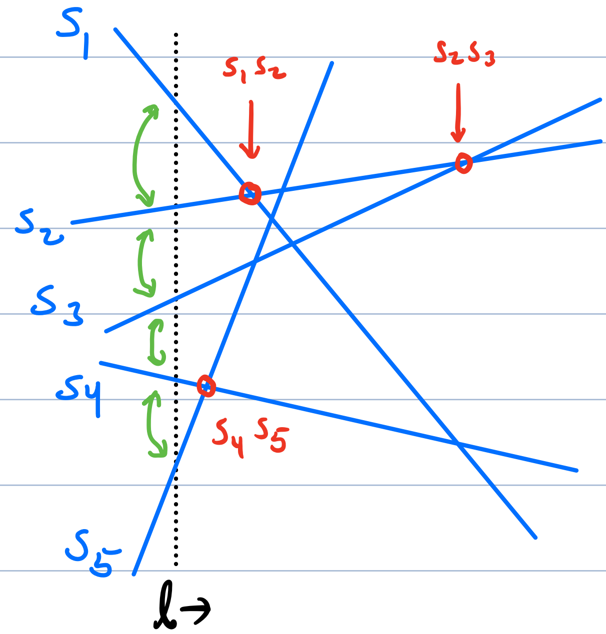
- As ℓ moves, all y-coordinates on sweep line change
- Much too slow to update all



- Dynamic comparator: Rather than storing y coords in dictionary, store line equation: $y = ax + b$
  - As x changes, reevaluate to compare y based on current x value

Future Events: (Stored in priority queue)

- All segment endpts to right of sweep line
- "Imminent" intersections: Intersections between pairs of lines that are consecutive on sweep line



$s_3 s_4$ - occurs in past (ignore)

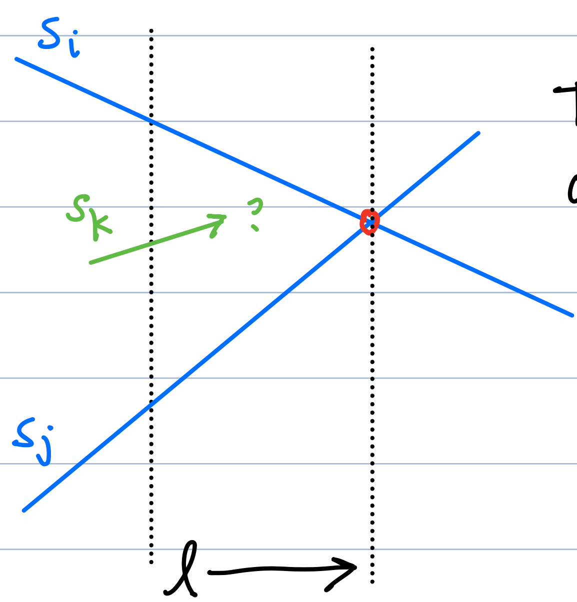
Why? - Consecutive pairs are easy to detect + update

- At most $n-1 = O(n)$ intersection events in priority queue ($+ \leq 2n$ end pt events)

$$\boxed{\textbf{Lemma:}\ \text{If the next event is an intersection, these segments will be consecutive on the current sweep line.}}$$

Proof:

- Suppose not
- $s_i s_j$ is next event, but not consecutive

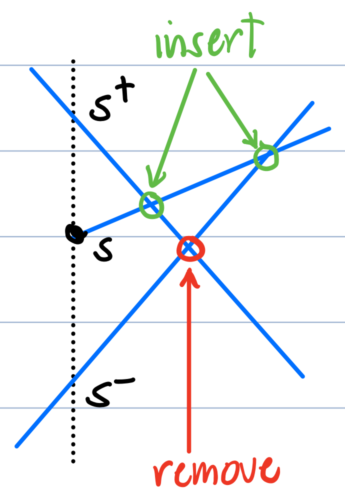


There must be an event involving s_k first

Final Sweep-Line Algorithm: $S = \{s_1, \ldots, s_n\}$ $s_i = \overline{p_i q_i}$

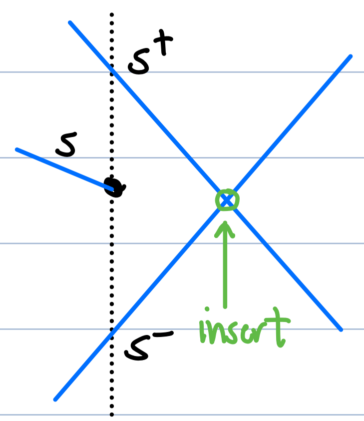
- Insert all seg. endpts into priority queue (sorted by x-coord)
- while(queue is non-empty) {
  - extract next event (min x)
  - cases:

Segment s left endpt:



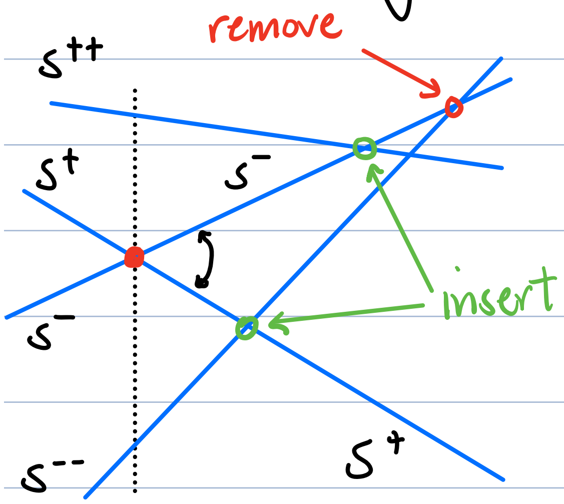
- Insert segment into sweep line (dictionary) based on y-coord
- Let s^+ + s^- be segs just above and below
- If s^+s^- has intersection event, remove from priority queue
- Add to priority queue, intersection events for ss^+ + ss^- (if appropriate)

Segment s right endpt:



- Let s^+ + s^- be segments above + below
- Add to priority queue, intersect event for s^+s^- (if appropriate)

Segment s^+s^- intersection:



- Let s^{++} + s^{--} be segs above and below intersection
- Remove intersection events s^+s^{++} + s^-s^{--} (if exist)
- Swap s^+ + s^- on sweep line
- Add to prior. queue, intersect events for s^+s^{--} + s^-s^{++} (if appropriate)

Correctness: Easy, but be sure not to forget anything

Running Time: n = num. of segs. m = num. of intersects

Total events: $2n + m = O(n+m)$

Time per event: Extract min, $O(1)$ dictionary ops, $O(1)$ queue ops $\Big\}$ $O(\log n)$ total

Total time: $O((n+m)\log n)$

Space: $O(n)$ for data structures, $O(m)$ for output