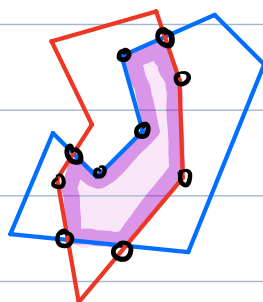
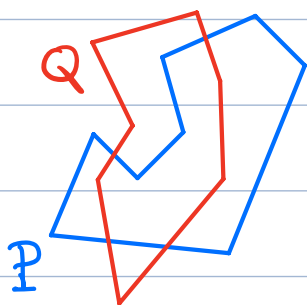
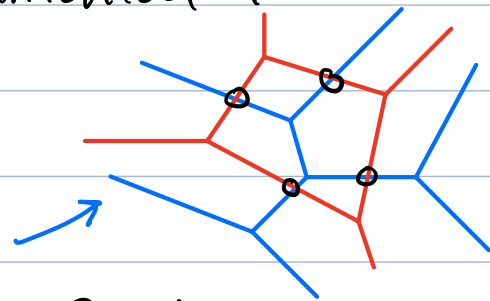


# CMSC 754 - Computational Geometry

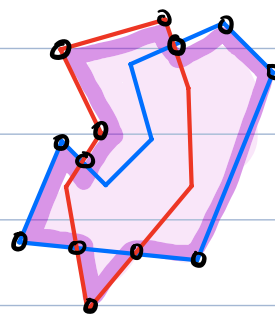
## Lecture 4: Line Segment Intersection

Computing intersections is fundamental to geometric computation

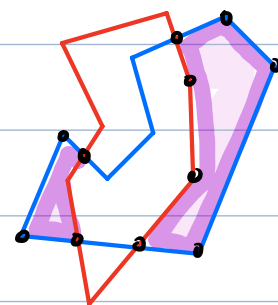
- collision detection
- subdivision overlay
- boolean operations -  $\cap, \cup, \dots$



$P \cap Q$



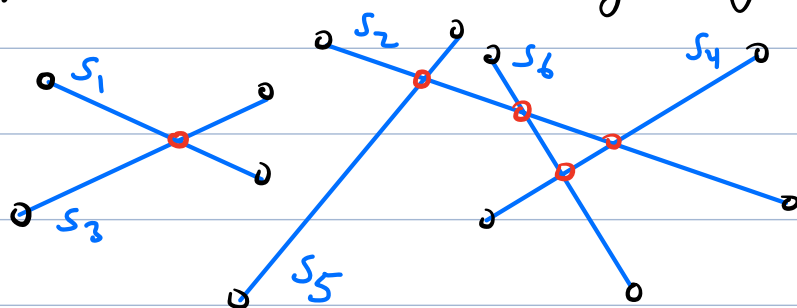
$P \cup Q$



$P \setminus Q$

### Line Segment Intersection:

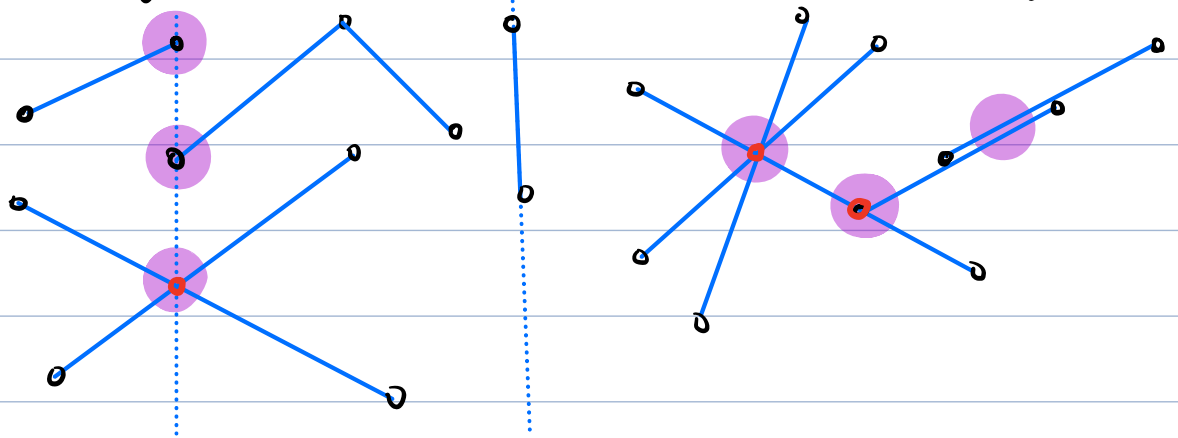
Given a set  $S = \{s_1, \dots, s_n\}$  of line segments in  $\mathbb{R}^2$  (where  $s_i = \overline{p_i q_i}$ ), report all pairs of intersecting segments.



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

## 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:  $m$

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

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

Follows from a lower bound on element uniqueness

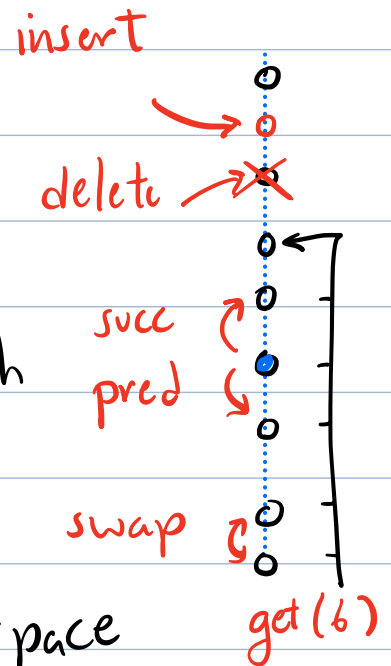
This lecture:  $O((n+m) \log n)$

↳ Plane sweep

## Utility Data Structures:

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

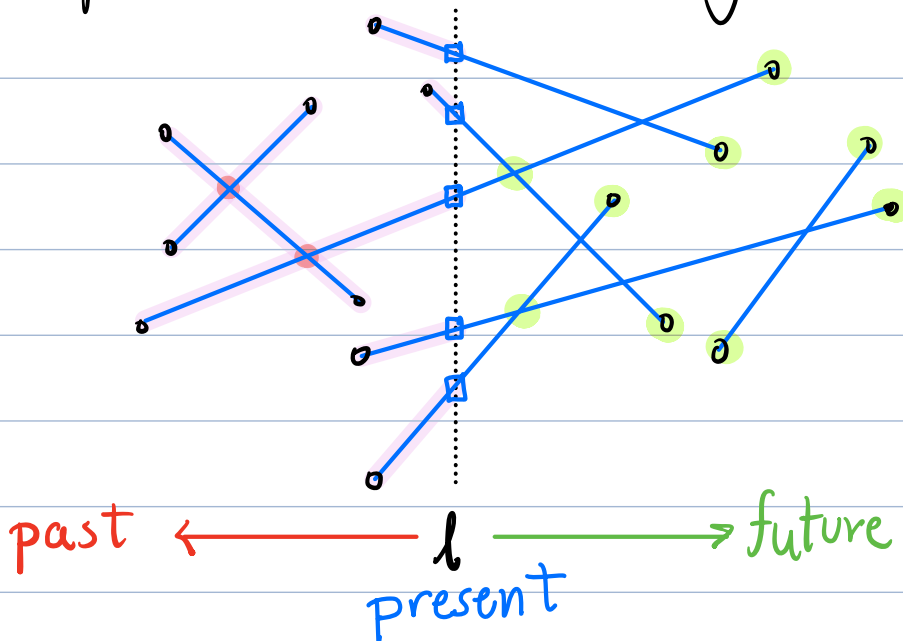
all in  $O(\log n)$  time +  $O(n)$  space



**Priority Queue:** Stores object  $\sigma$  + priority  $x$   
 $ref \leftarrow enqueue(\sigma, x)$   
 $\sigma \leftarrow extract\_min()$  - removes obj w.  
min priority  
 $delete(ref)$

## Sweep-Line Algorithm:

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

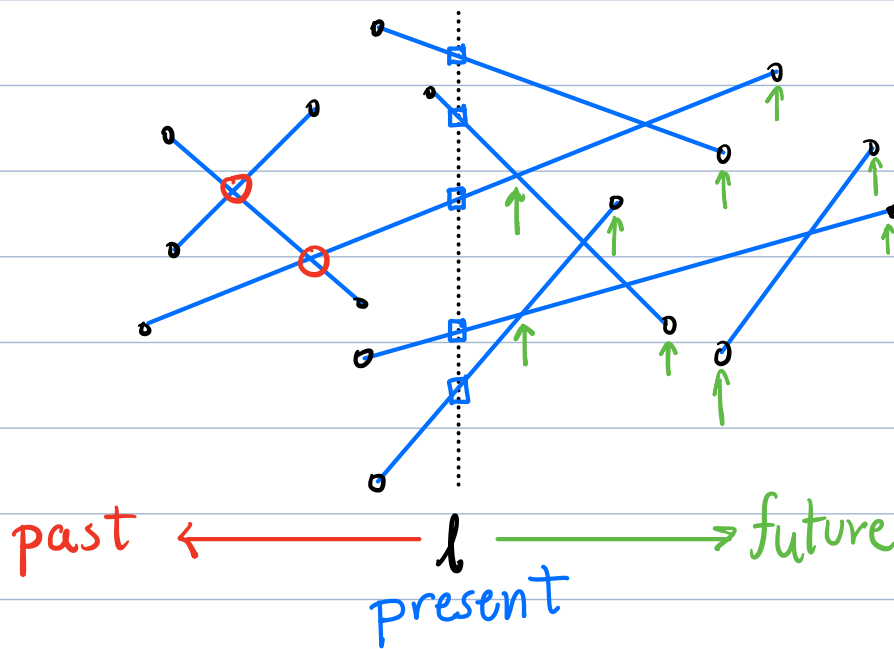


What we store: (Generic Plane Sweep)

(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)

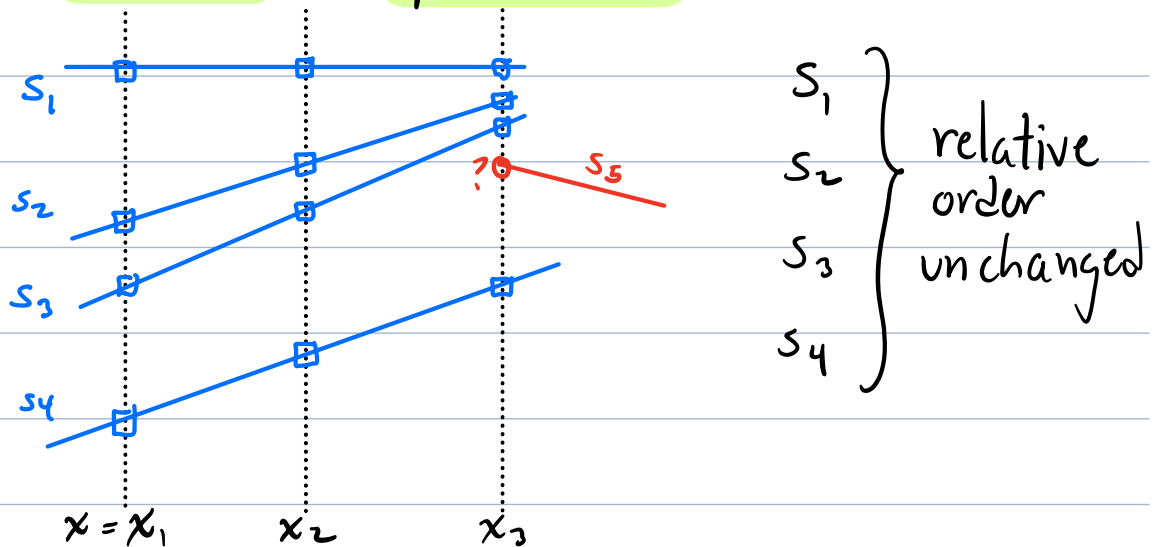
Past: List of intersecting pairs so far

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

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

## Sweep-Line Status:

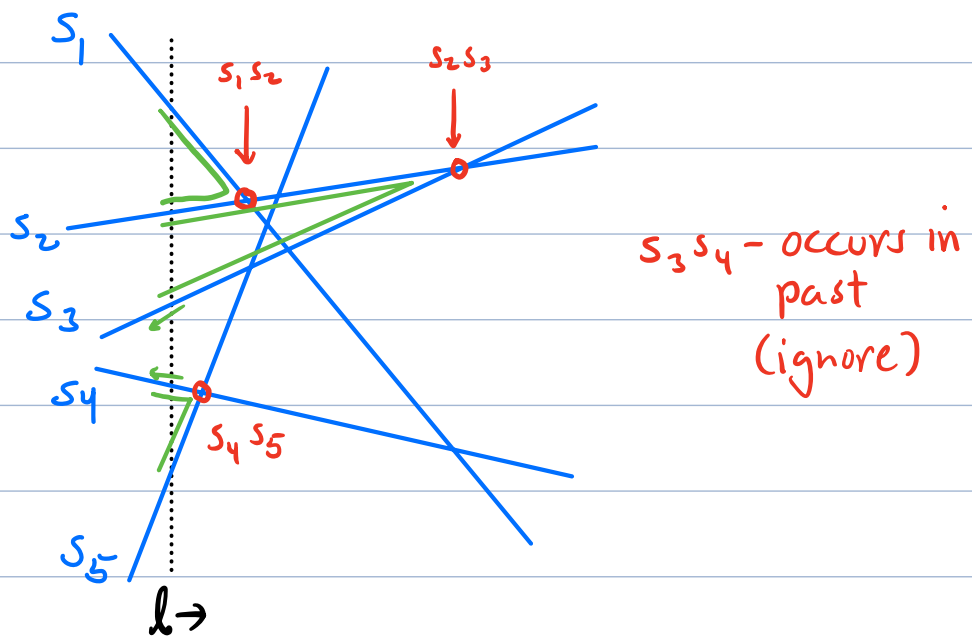
- As  $l$  moves, all  $y$ -coordinates on sweep line change
- Much too slow to update all



- **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 endpoints** to right of sweep line
- **Imminent intersections**:  
Intersections between pairs of lines that are **consecutive** on sweep line



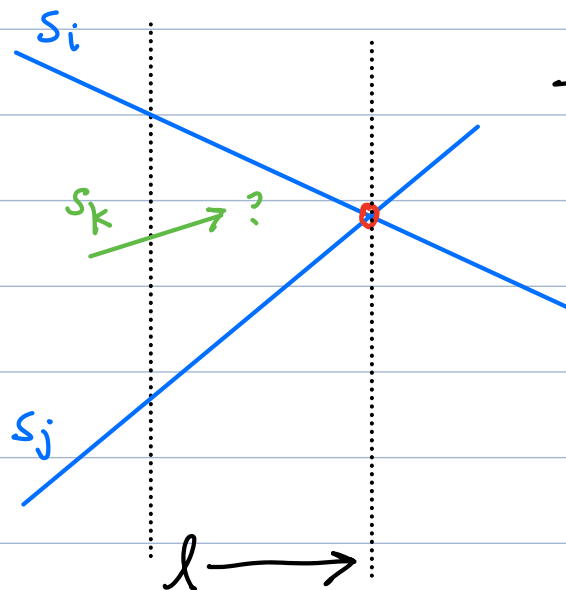
Why? - Consecutive pairs are easy to detect + update

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

Lemma: 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

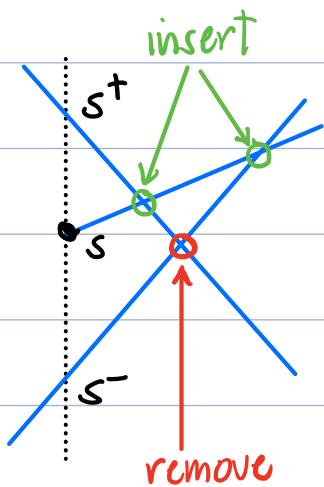


There must be an event involving  $s_k$  first

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

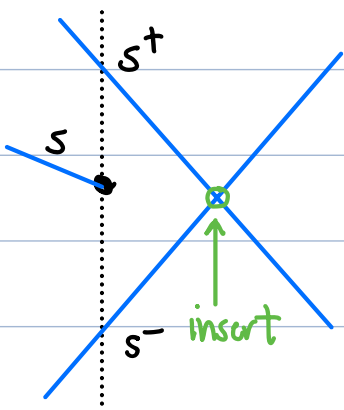
- Insert all seg. endpts into priority queue (sorted by x-coord)
- while (queue is non-empty) {
  - extract next event (min x)
  - cases:

Segment  $s$  left endpt:



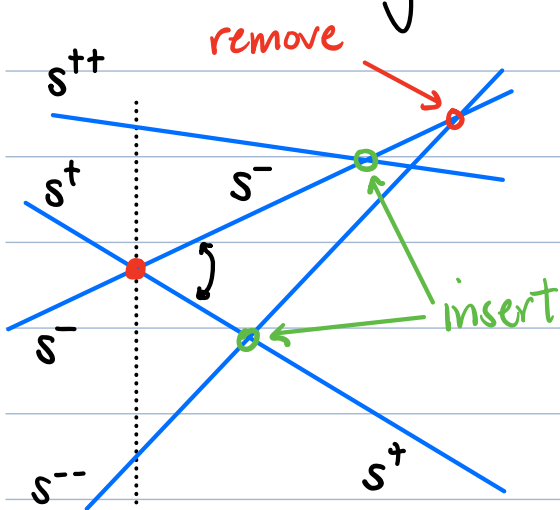
- Insert segment into sweep line (dictionary) based on y-coord
- Let  $s^+ + s^-$  be segs just above and below
- If  $s^+ s^-$  has intersection event, remove from priority queue
- Add to priority queue, intersection events for  $s s^+ + s s^-$  (if appropriate)

Segment  $s$  right endpt:



- Let  $s^+ + s^-$  be segments above + below
- Delete  $s$  from sweep-line
- 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:  $n = \text{num. of segs.}$   $m = \text{num. of intersects}$

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

Time per event: Extract min  $\left\{ \begin{array}{l} O(1) \text{ dictionary ops} \\ O(1) \text{ queue ops} \end{array} \right\} O(\log n) \text{ total}$

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

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

## Summary:

- Plane-sweep algorithm to compute line segment intersections
- $O((n+m) \log n)$  time
  - $n = \text{no. of segs}$
  - $m = \text{no. of intersections}$
- Q: Can we do better?
  - $O(n \log n + m)$  ? (yes-later)
- Plane-sweep
  - Powerful technique
  - Keys:
    - Sweep-line status
    - Queue of future events
    - Update rules