

# Computational Geometry

Winter term 2016/17

## Line-Segment Intersection

or

## Map Overlay

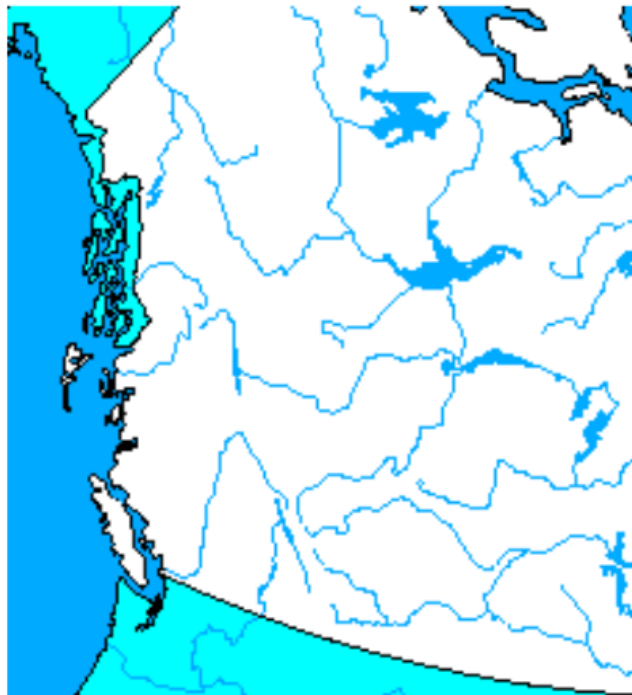
Lecture #2

# Chapter 2

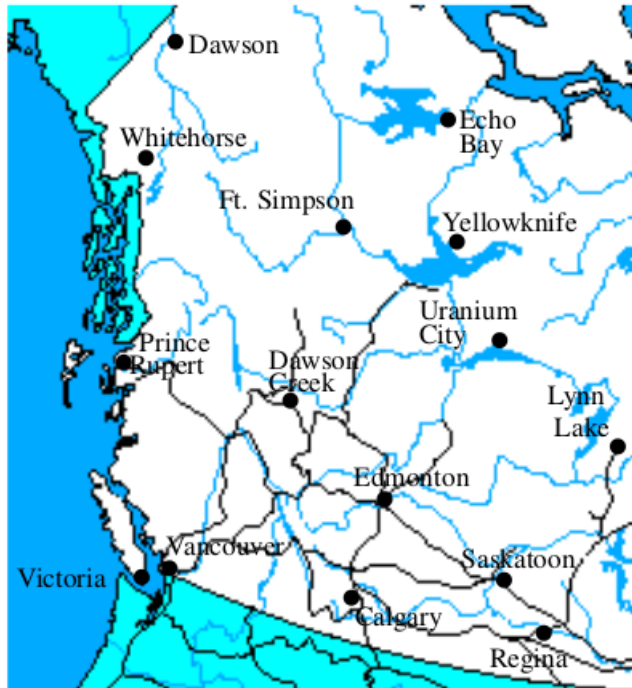
## Line-Segment Intersection

or

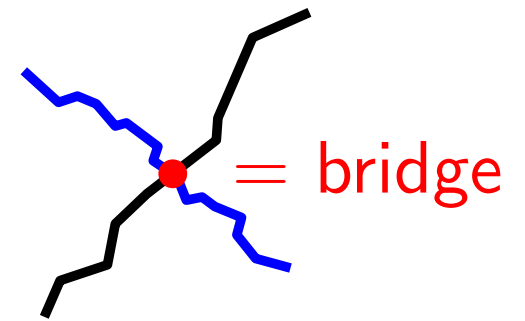
## Map Overlay



# Map Overlay in Geographic Information Systems (GIS)

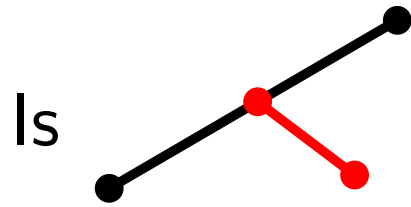


*Here:*



# Line-Segment Intersection

**Definition:**



Is an intersection?

**Answer:**

Depends...

yes!

**Problem:**

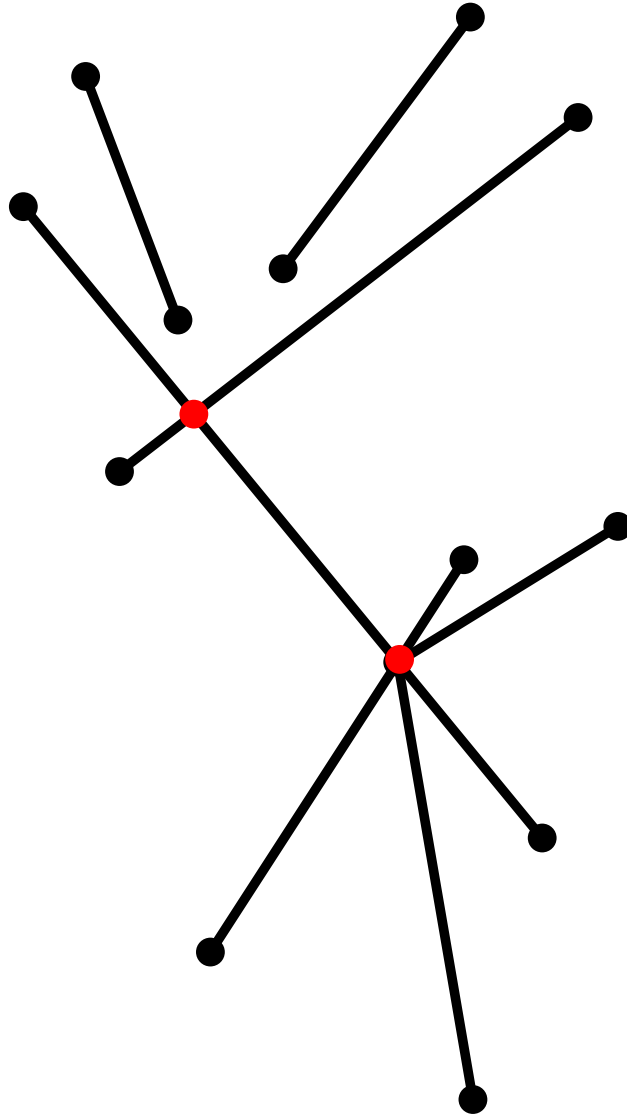
Given a set  $S$  of  $n$  *closed* non-overlapping line segments in the plane, compute...

- all points where at least two segments intersect and
- for each such point report all segments that contain it.

**Task:**

Discuss with your neighbor: how would *you* do it?

# Example

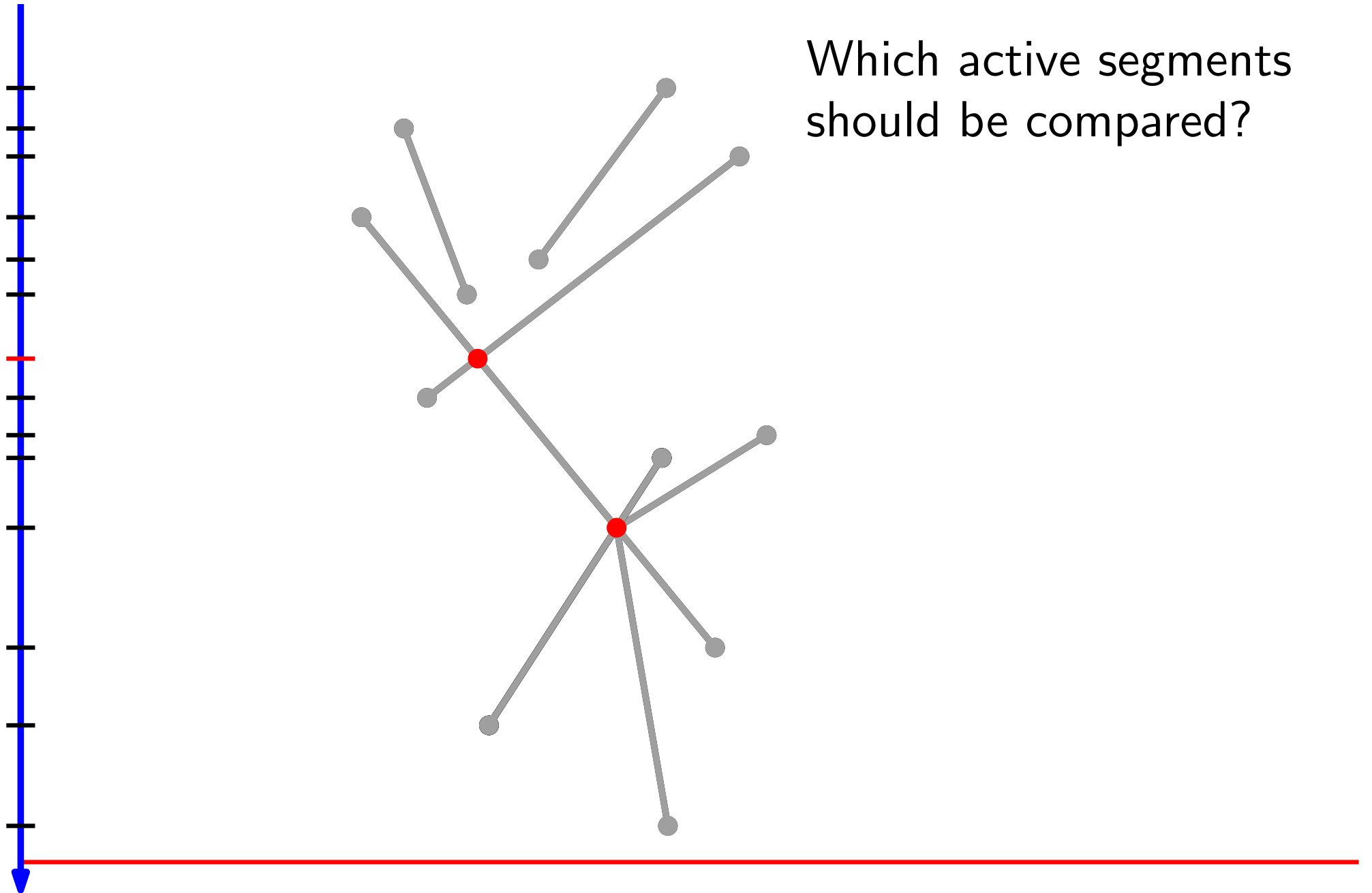


Brute Force?

$O(n^2)$  ... can we do better?

Idea: Process Segments top-to-bottom using a "sweep-line"

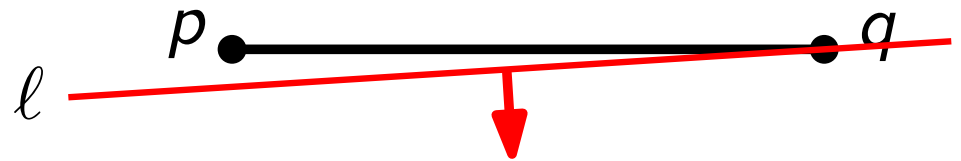
# Sweep-Line Algorithm



# Data Structures

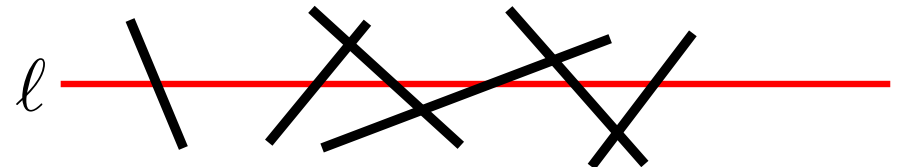
## 1) event (-point) queue $\mathcal{Q}$

$$p \prec q \iff_{\text{def.}} y_p > y_q \quad \text{or} \quad (y_p = y_q \text{ and } x_p < x_q)$$



Store event pts in *balanced binary search tree* acc. to  $\prec$   
 $\Rightarrow$  nextEvent() and del/insEvent() take  $O(\log |\mathcal{Q}|)$  time

## 2) (sweep-line) status $\mathcal{T}$



Store the segments intersected by  $\ell$  in left-to-right order.

How? In a balanced binary search tree!

# Pseudo-code

## findIntersections( $S$ )

**Input:** set  $S$  of  $n$  non-overlapping closed line segments

**Output:** – set  $I$  of intersection pts  
– for each  $p \in I$  every  $s \in S$  with  $p \in s$

$Q \leftarrow \emptyset$ ;  $\mathcal{T} \leftarrow \langle \text{vertical lines at } -\infty \text{ and } +\infty \rangle$  // sentinels

**foreach**  $s \in S$  **do**

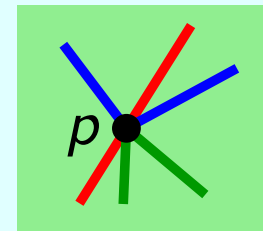
**foreach** endpoint  $p$  of  $s$  **do**

**if**  $p \notin Q$  **then**  $Q.\text{insert}(p)$ ;  $L(p) = U(p) = \emptyset$

**if**  $p$  lower endpt of  $s$  **then**  $L(p).\text{append}(s)$

**if**  $p$  upper endpt of  $s$  **then**  $U(p).\text{append}(s)$

$C(p)$



**while**  $Q \neq \emptyset$  **do**

$p \leftarrow Q.\text{nextEvent}()$

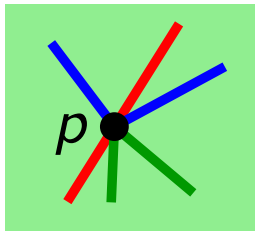
$Q.\text{deleteEvent}(p)$

$\text{handleEvent}(p)$

This subroutine does the real work –  
how would you implement it?



# Handling an Event



$C(p), L(p), U(p)$

**handleEvent**(event  $p$ )

**if**  $|U(p) \cup L(p) \cup C(p)| > 1$  **then**

┌ report intersection in  $p$ , report segments in  $U(p) \cup L(p) \cup C(p)$

delete  $L(p) \cup C(p)$  from  $\mathcal{T}$  // consecutive in  $\mathcal{T}$ !

insert  $U(p) \cup C(p)$  into  $\mathcal{T}$  in their order slightly below  $\ell$

**if**  $U(p) \cup C(p) = \emptyset$  **then**

┌  $b_{\text{left}}/b_{\text{right}}$  = left/right neighbor of  $p$  in  $\mathcal{T}$

└ findNewEvent( $b_{\text{left}}, b_{\text{right}}, p$ )

**else**

┌  $s_{\text{left}}/s_{\text{right}}$  = leftmost/rightmost segment in  $U(p) \cup C(p)$

└  $b_{\text{left}}$  = left neighbor of  $s_{\text{left}}$  in  $\mathcal{T}$

└  $b_{\text{right}}$  = right neighbor of  $s_{\text{right}}$  in  $\mathcal{T}$

└ findNewEvent( $b_{\text{left}}, s_{\text{left}}, p$ )

└ findNewEvent( $b_{\text{right}}, s_{\text{right}}, p$ )

**findNewEvent**( $s, s', p$ )

**if**  $s \cap s' = \emptyset$  **then return**

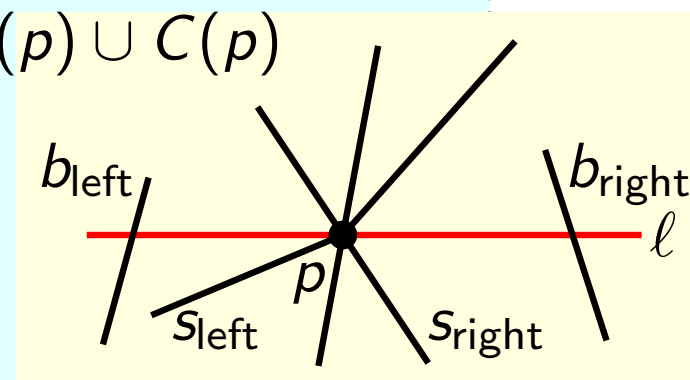
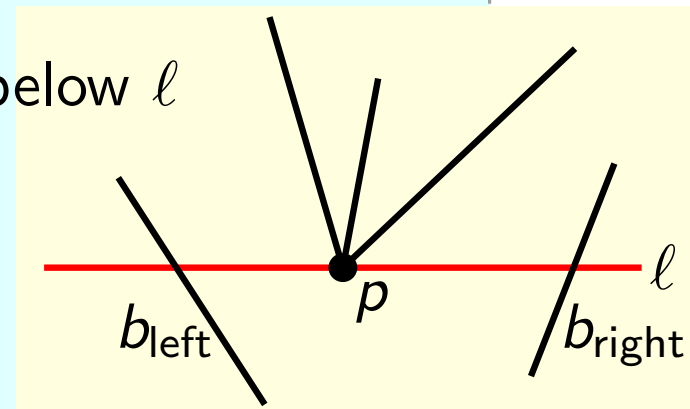
$\{x\} = s \cap s'$

**if**  $x$  below  $\ell$  or to the right of  $p$  **then**

┌ **if**  $x \notin Q$  **then**  $Q.\text{add}(x)$

└ **if**  $x \in \text{rel-int}(s)$  **then**  $C(x) \leftarrow C(x) \cup \{s\}$

└ **if**  $x \in \text{rel-int}(s')$  **then**  $C(x) \leftarrow C(x) \cup \{s'\}$



# Correctness

**Lemma.** `findIntersections()` correctly computes all intersection points & the segments that contain them.

*Proof.* Let  $p$  be an intersection pt. Assume (by induction):

- Every int. pt  $q \prec p$  has been computed correctly.
- $\mathcal{T}$  contains all segments intersecting  $\ell$  in left-to-right order.

**Case I:**  $p$  is not an interior pt of a segment.

$\Rightarrow p$  has been inserted in  $\mathcal{Q}$  in the beginning.

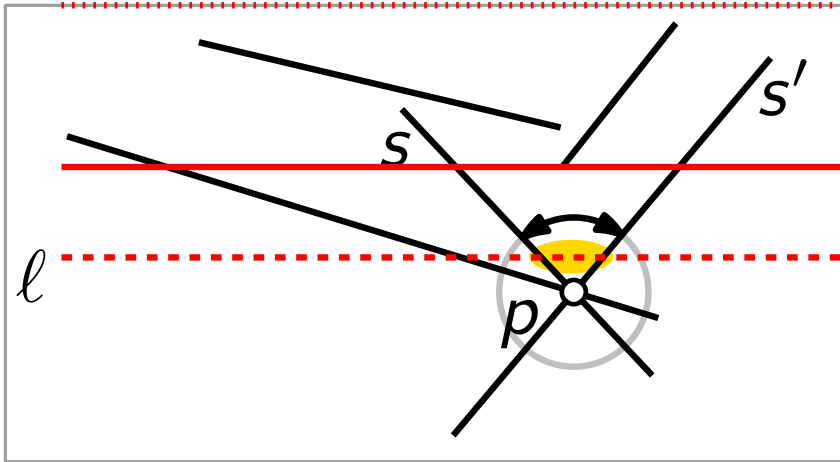
Segm. in  $U(p)$  and  $L(p)$  are stored with  $p$  in the beginning.

When  $p$  is processed,  $\mathcal{T}$  contains all segm. in  $U(p) \cup L(p)$ .

$\Rightarrow$  All segments that contain  $p$  are reported.

# Correctness (Case II)

**Case II:**  $p$  is an interior point of some segment., i.e.,  $C(p) \neq \emptyset$ .  
If  $p$  is not an endpt, need that  $p$  is inserted into  $\mathcal{Q}$  “above”  $p$ .



We also need that every segment with  $p$  as an interior point is added to  $C(p)$ .

Let  $s, s' \in C(p)$  be neighbors in the circular ordering of  $C(p) \cup \{\ell\}$  around  $p$ . Imagine moving  $\ell$  slightly back in time. Then  $s, s'$  were neighbors in the left-to-right order on  $\ell$  (in  $\mathcal{T}$ ). At the beginning of the algorithm, they weren't neighbors in  $\mathcal{T}$ .  
 $\Rightarrow$  There was some moment when they became neighbors!  
This is when  $\{p\} = s \cap s'$  was inserted into  $\mathcal{Q}$ . □

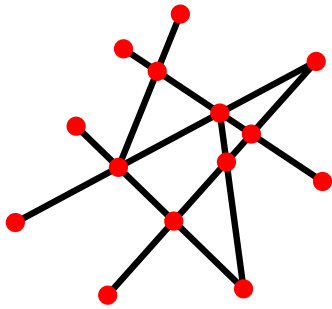
# Running Time

*Check your knowledge about planar graphs!*

**Lemma:** findIntersections() finds  $I$  intersection points among  $n$  non-overlapping line segments in  $O((n + I) \log n)$  time.

*Proof.*

Let  $p$  be an event pt,  
 $m(p) = |L(p) \cup U(p) \cup C(p)|$  and  $m = \sum_p m(p)$ .  
Then it's clear that the runtime is  $O((m + n) \log n)$ .



We show that  $m \in O(n + I)$ . ( $\Rightarrow$  lemma)

Define (geometric) graph  $G = (V, E)$  with  
 $V = \{ \text{endpts, intersection pts} \} \Rightarrow |V| \leq 2n + I$ .

For any  $p \in V$ :  $m(p) \leq \text{deg}(p)$ .

$$\Rightarrow m \leq \sum_p \text{deg}(p) = 2|E| \leq 2 \cdot (3|V| - 6)$$

Euler ( $G$  is planar!!)  $\in O(n + I)$   $\square$

# Today's Main Result

**Theorem:** We can report all  $I$  intersection points among  $n$  non-overlapping line segments in the plane and report the segments involved in the intersections in  $O((n + I) \log n)$  time and  $O(n)$  space.

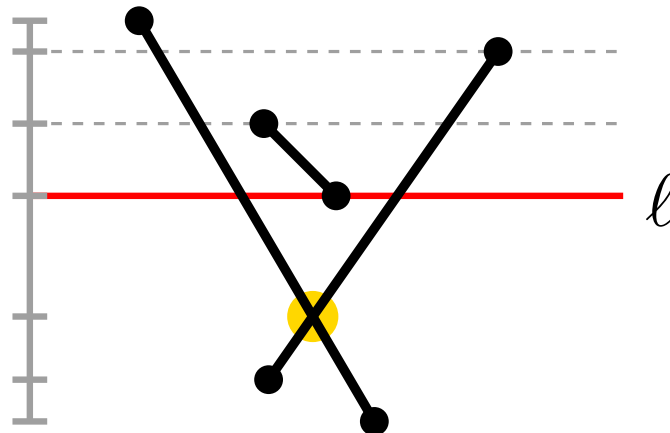
Sure?

The event-point queue  $Q$  contains

- all segment end pts below the sweep line
- all intersection pts below the sweep line

$\Rightarrow$  (worst-case) space consumption  $\in \Theta(n + I) :-$

Can we do better?



- insert  $s \cap s'$  into  $Q$
- remove  $s \cap s'$  from  $Q$
- re-insert  $s \cap s'$  into  $Q$

$\Rightarrow$  need just  $O(n)$  space,  
(asymptotic) running  
time doesn't change  $\square$