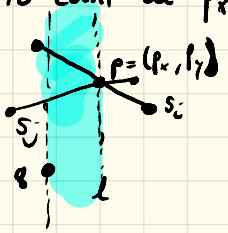
# Seg Intersection (Pt 2)



Does this find all intersections
Can we have 2 segs intersect w/o being adj on the sweepline?

**Lemma:** Let $S$ be segments (in general pos)
consider $s_i, s_j \in S$ that intersect at $p = (p_x, p_y)$
$s_i$ and $s_j$ are adj on the sweepline prior to event at $p_x$




pf:
Consider a line $\ell$ intersect left of $p$
General pos assumption
$\Rightarrow$ no 3 segs intersect at a common point
$\Rightarrow$ $s_i, s_j$ are adj on sweepline at $\ell$

Consider $q$ (eventpoint) w/ largest x-coord smaller than $p_x$
$\Rightarrow$ no events between $q_x$ and $p_x$
$\Rightarrow$ order on sweepline after $q$ is the same as before $p$
$\Rightarrow$ $s_i$ $s_j$ adj on sweepline ✓

Data structs
- ordered dict:
  segs sorted top to bottom
- $r \leftarrow$ insert$(s)$: "symbollically" insert $s$
  return ref to $r$ (location in the datastructure)
- delete$(r)$: delete entry for $r$
- $r' \leftarrow$ predecessor$(r)$: return a ref $r'$ the segment above $r$ (null or sentinal "top" segment)
- $r' \leftarrow$ successor$(r)$: return a ref $r'$ to the seg below $r$ (sentinal "below" segment)
- $r' \leftarrow$ swap$(r)$: swap $r$ w/ successor return $r$'s new location

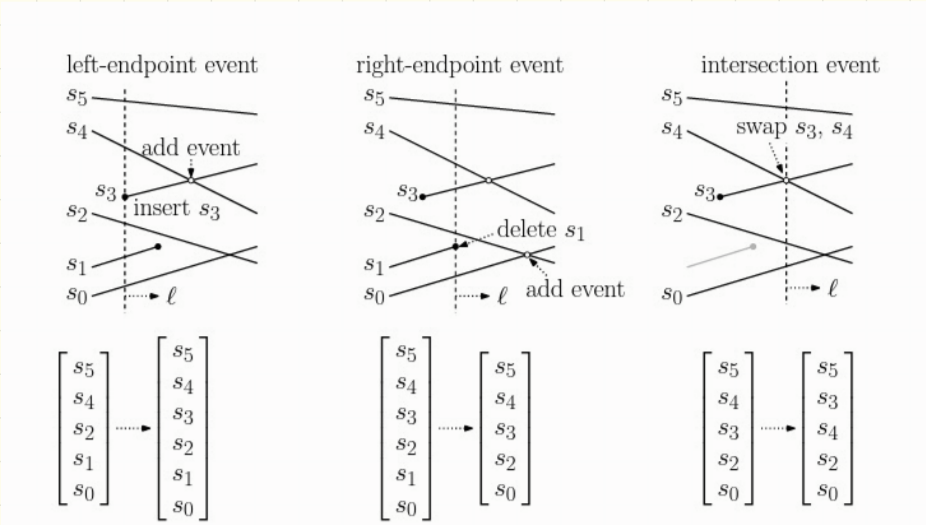
how long do these ops take?
if $m$ things can be in sweepline all ops in $O(\log m)$
any seg can only appear in status 1 $\Rightarrow O(\log n)$

Priority queue (events)
- $r \leftarrow$ insert$(e, x)$: insert event $e$ w/ priority $x$ and return ref $r$ to loc in datastructure
- delete$(r)$: delete entry $r$ in PQ
- $(e, x) \leftarrow$ extract_min() pops event $e$ w/ min priority

For $m$ events

... time:
- $\log(m)$ time

... space:
- $O(m)$






(1) Insert all of the endpoints of the line segments of $S$ into the event queue. The initial sweep-line status is empty.

(2) While the event queue is nonempty, extract the next event in the queue. There are three cases, depending on the type of event:

**Left endpoint:** (see Fig. 26(a))

(a) Insert this line segment $s$ into the sweep-line status, based on the $y$-coordinate of its left endpoint.

(b) Let $s'$ and $s''$ be the segments immediately above and below $s$ on the sweep line. If there is an event associated with this pair, remove it from the event queue.

(c) Test for intersections between $s$ and $s'$ and between $s$ and $s''$ to the right of the sweep line. If so, add the corresponding event(s) to the event queue.

**Right endpoint:** (see Fig. 26(b))

(a) Let $s'$ and $s''$ be the segments immediately above and below $s$ on the sweep line.

(b) Delete segment $s$ from the sweep-line status.

(c) Test for intersections between $s'$ and $s''$ to the right of the sweep line. If so, add the corresponding event to the event queue.

**Intersection:** (see Fig. 26(c))

(a) Report this intersection.

(b) Let $s'$ and $s''$ be the two intersecting segments. Swap these two line segments in the sweep-line status (they must be adjacent to each other).

(c) As a result, $s'$ and $s''$ have changed which segments are immediately above and below them. Remove any old events due to adjacencies that have ended and insert any new intersection events from adjacencies that have been created.

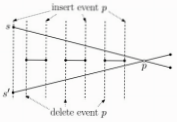



Fig. 27: An intersection event that is repeatedly inserted and deleted from the event queue

Analysis:

# of events processed: $2n + I$

- get event $O(1)$ get top elt of the PQ
- $O(1)$ accesses to the sweepline data structure
- each access to DS takes $O(\log n)$ time

$\Rightarrow$ runtime of algo is $O((2n+I)\log n) \Rightarrow O(n \log n + I \log n)$

$= O((n+I)\log n)$

Can we do better

- $O(n \log n + I)$ $\Leftarrow$ asymptotically opt