ORTHOgonal SEGMENT INTERSECTION

(1) $O(n \log n)$
(2) $O(n)$
(3) $O(n) \cdot O(\log n)$

$O(n) \cdot O(k + \log n)$ to report

(1) Sort all endpoints by $x$.
(2) Sweep vertical line $\rightarrow$ & insert/delete only horizontal segments.
(3) When sweeping through a vertical segment, do 1D range query.
SEARCHING FOR OVERLAPPING INTERVALS

ID:

BST w/ LEFT ENDS as KEYS

MAX RIGHT END OF SUBTREE
SEARCHING FOR OVERLAPPING INTERVALS

ID:

compare w/ root first

query segment
SEARCHING FOR OVERLAPPING INTERVALS

ID:

IF NO OVERLAP

R-subtree can't overlap query

case 1

Keep searching
LEFT

R < x < w
SEARCHING FOR OVERLAPPING INTERVALS

1D:

IF $Z > L$
- search left

IF NO OVERLAP

$L \quad \text{case 2} \quad R$

$z$

$z'$

guaranteed overlap

...else...(RIGHT SUBTREE COULD ALSO OVERLAP)
SEARCHING FOR OVERLAPPING INTERVALS

ID: ___________________________ __________________________

* To report all \( k \) intersections of query with segments
  - just search every path (left & right if necessary) \( O(k + \log n) \)

* Update BST: \( O(\log n) \)
  (could also query one, delete, and repeat)

* Find intersections among set:
  incremental search & insert
  \( O(\sum k_i) + n \cdot O(\log n) = O(K + n \log n) \)
SEARCHING FOR OVERLAPPING RECTANGLES

* n inserts & deletes
* \( \sum k_i \) to report
\[ O(k + n \log n) \]
Finding all intersections in a set of segments
Sweep

A insert

(build tree with sorted segments by y-overlap w/ sweepline)
Invariant: make sure neighbors on sweepline have been checked
Sweep

F (insert)

check F \times D
Sweep

F

B (insert)

check $F \times B$

$B \times F$ is ahead of sweep:
insert into event queue: $O(\log n)$
Sweep

A

D

C

E

B

F

G

F

C

(check CxF ✓
CxB

insert : $O(\log n)$

$O(\log [n+I])$

I = # intersections
Sweep

Swap event

$O(\log n)$

check $F \times B$

but we can see we already have it
Sweep

C
F

insert
E

B

E x F
check
E x B

A
D

C
E
F
G

B

E_L E_R B_F C_R F_R G_L B_R G_R
Sweep

Delete E

Check B x F (again)
Recap:

- Event queue: binary tree, size $O(n+I)$
- Sweep structure: binary tree, size $O(n)$
- Inserting, deleting, swapping on sweepline: $O(\log n)$
  
  Total time: $O(n \log n)$

- Adding events (intersections): necessary to detect swaps.
  
  - Time: $O(\log [n+I])$ per event
  
  $I: O(n^2) \Rightarrow$ time: $O(\log n)$
  
  Total time: $(n+I) \log n$

$O(I + n \log n)$ algo exists