33.2 | Whether any pair of segments intersect.

Given a set of line segments of the form $s = (p_1, p_2)$

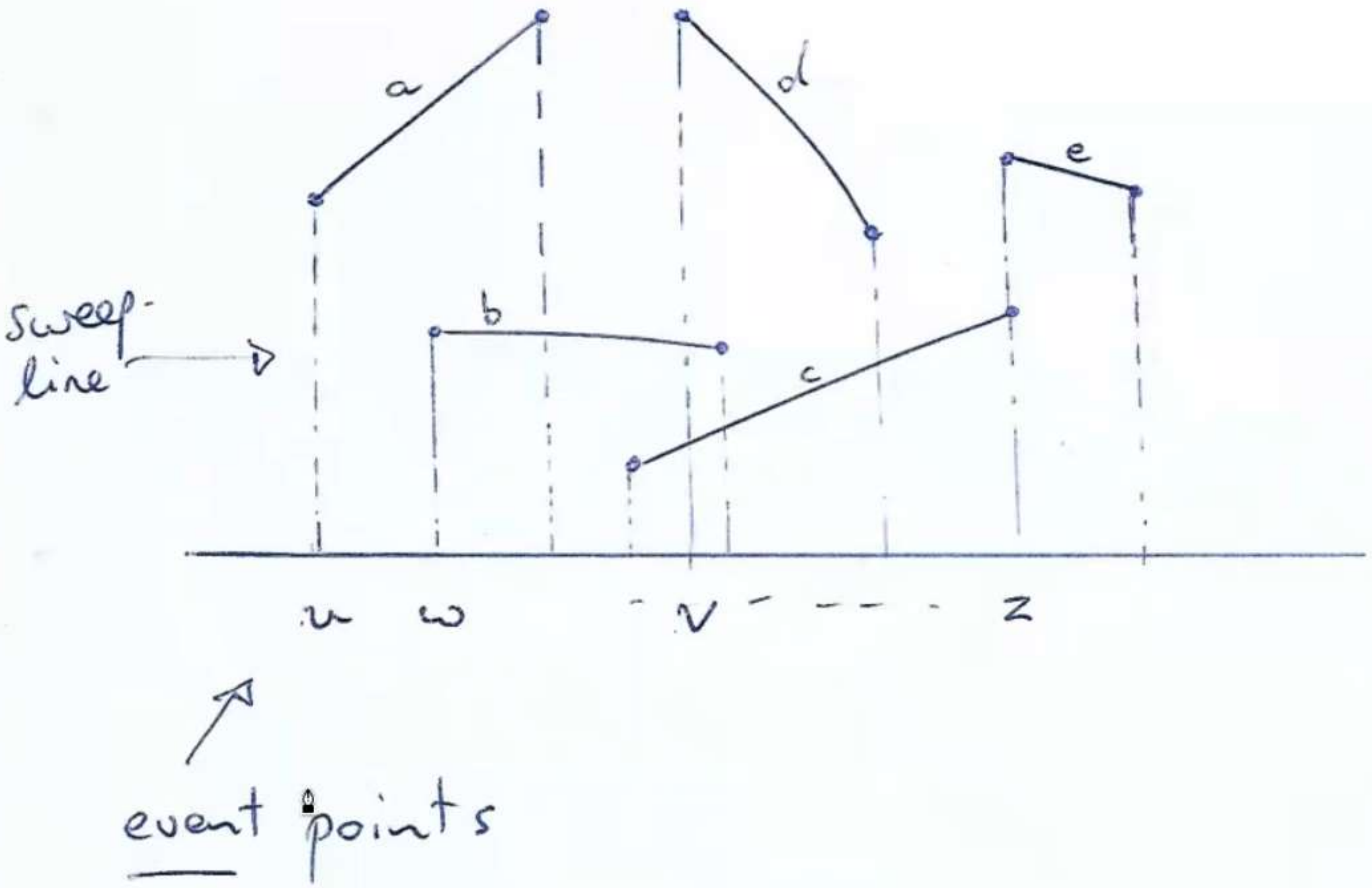
does any pair of segments intersect.

Say $s_1, s_2, \ldots, s_n$ are the segments,

a brute force method gives
an $\mathcal{O}(n^2)$ run-time algorithm -

We outline an $\mathcal{O}(n \log n)$ run-time
algorithm Any-Segments-Intersect

Note: the algorithm only determines if any pair intersects and does not find all intersections.


a
d
e
b
c
sweep-line
u
w
v
z
event points

- Each event point gives a <u>sweep-line</u> which is a vertical line at that point.

- A sweep line may cut through a number of line segments.
  Those line segments can be ordered from smallest to largest by $\preceq$

eg. At event point v

e.g. At event point r

$$c \leq_r b \leq_r d$$

The ordered sequence of event points (from left-to-right) is called the <u>event-point schedule</u>.

The algorithm works as follows:

- We maintain a data-structure T containing segments that supports Insert and Delete operations.

- T also has an order relation (a total order) and for any s in T we can obtain Above(T, s) and Below(T, s) which are the segments in T directly greater than s and directly smaller than s with respect to the order.

ANY-SEGMENTS-INTERSECT (S)

T = ∅

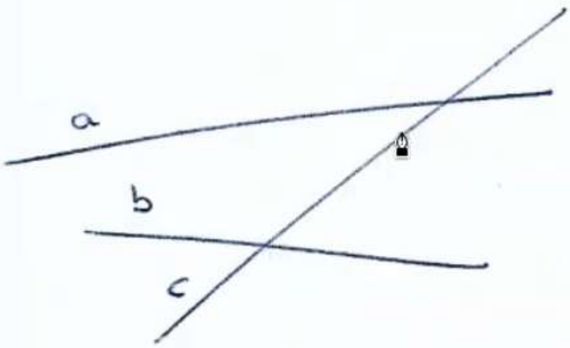
sort endpoints of segments in S from left to right
(break ties by putting left endpoints before right
and further ties by y-co-ordinate.)

for each p in sorted list of endpoints
    if p is left endpoint of segment s
        Insert(T, s)
        if Above (T, s) exists and intersects with s
        or Below(T, s) exists and intersects with s

if p is right endpoint of segment s
if both Above(T, s) and Below(T, s)
exist and intersect
return TRUE
Delete(T, s)

Return FALSE.

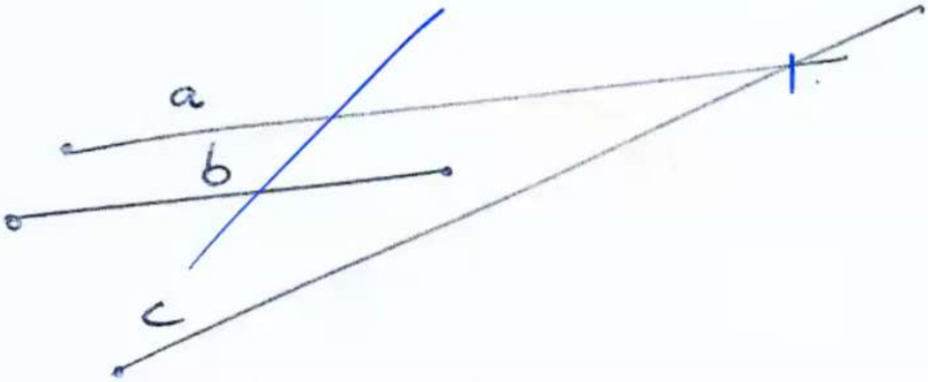
The algorithm uses the following ideas:



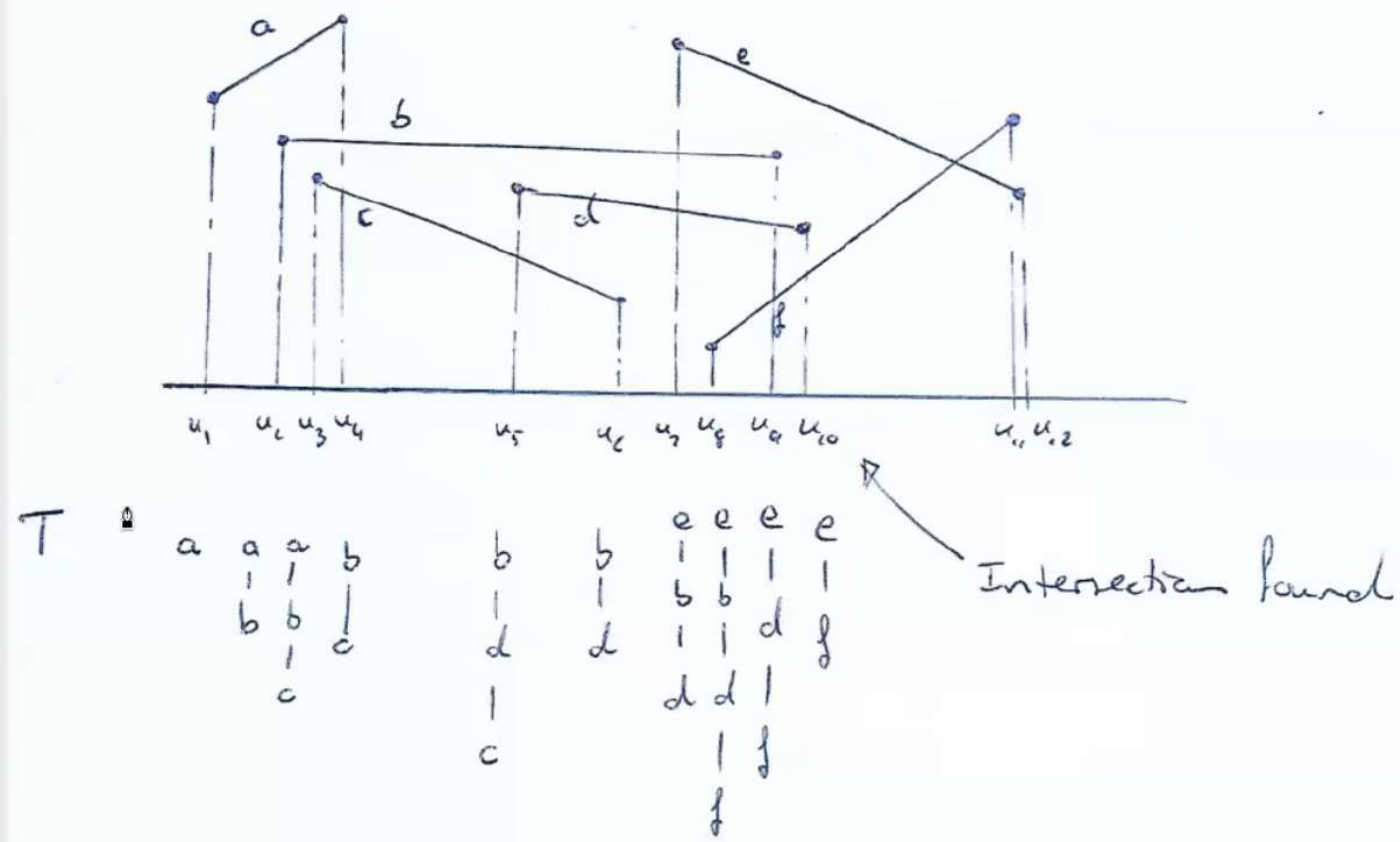
c can't intersect with a unless it intersects with b (here b = Above (T, c)).

- So it's enough to check for intersection with Above & Below.

<u>unless</u> the intersection occurs past the right endpoint of b :



- So when we get the right endpoint of a segment we check if its Above and Below endpoints intersect.


a
b
c
d
e
f
u1 u2 u3 u4 u5 u6 u7 u8 u9 u10 u11 u12
T
Intersection found

Running - time of ANY-SEGMENTS-INTERSECT.

Given a set S of n segments.

sorting the endpoints:

there are $2n$ endpoints

$\therefore O(2n \log 2n) = O(n \log n)$

For loop over all $2n$ endpoints

check for intersections - $O(1)$

For loop over all 2n endpoints

check for intersections - $O(1)$

Insert or Delete - ??

Depends on data-structure T.

Use a Red-Black tree to store segments
Instead of keys, order the nodes by
the order on segments.
I.e., keys are segments.

Recall
Insert & Delete
are $O(\log n)$.

Then the loop run-time is

$O(2n \log n)$

In total, therefore $O(n \log n)$