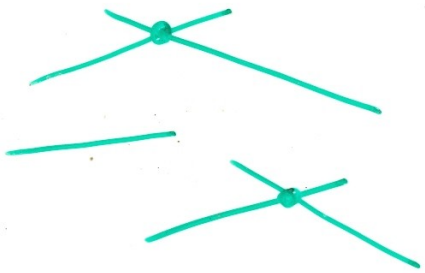
Segment intersection Problem



Given a set of line segments in plane determine (i) if any pair intersects (ii) compute all intersections.

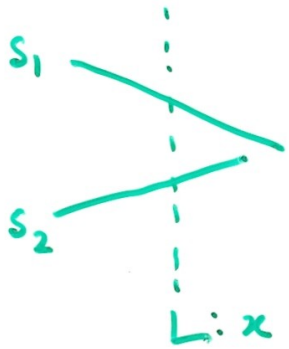
We can solve both (i) & (ii) using the sweeping technique. In this technique we move a vertical line from left to right and record information as we continue seeing more and more segments and possibly their intersections.

(i) Any pair intersection: if there are n segments, ~~this~~ it can be reported if any pair intersects in time $O(n \log n)$.

(ii) All intersection points can be computed in $O((n+s)\log n)$ where s is the number of intersections.

Reporting any pair intersection

Ordering segments. We say two segments s_1 and s_2 are comparable at x if they intersect the vertical line with x-coordinate x. We say s_1 is above s_2 at x, written $s_1 \geq_x s_2$ if s_1 & s_2 are comparable at x and the intersection of s_1 with the sweep line L lies above that of s_2.



$s_1 \geq_x s_2$ } s_1 "above" s_2
$s_2 \leq_x s_1$ } s_2 "below" s_1.

Assumptions. (i) No segment is vertical
(ii) No three segments intersect in a common point.

These are assumed for simplicity. But, they can be removed with some additional steps in the algorithms.

We maintain two data structures.

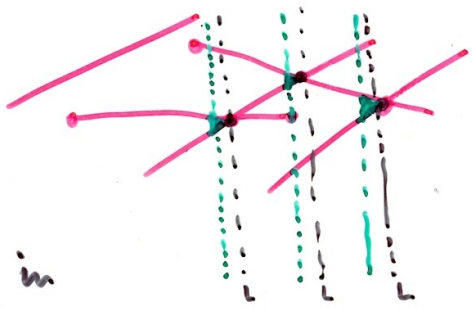
A. Sorted list for sweep order

B. Sorted list of intersection order on L.

- The sweep order is determined by the endpoints of the segments. So, A is implemented with a simple list of endpoints of the segments ordered from left to right.

- The list of intersection points with L change as we sweep. So, B is implemented with a dynamic data structure that allows insertions and deletions. A balanced binary search tree ... red-black tree for example.

- **An observation:**
  Immediately before any intersection the two segments become consecutive in intersection order.

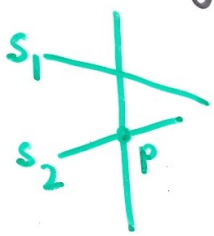


We can exploit previous observation.
The sweep line stops at endpoints, called event points. We modify the order T of intersections as a result of events.
T can be maintained as ordering of segments.

- Insert (T, s): insert segment s into T
- Delete (T, s): delete s from T.
- Above (T, s): return the segment immediately above s in T.
- Below (T, s): return the segment immediately below s in T.

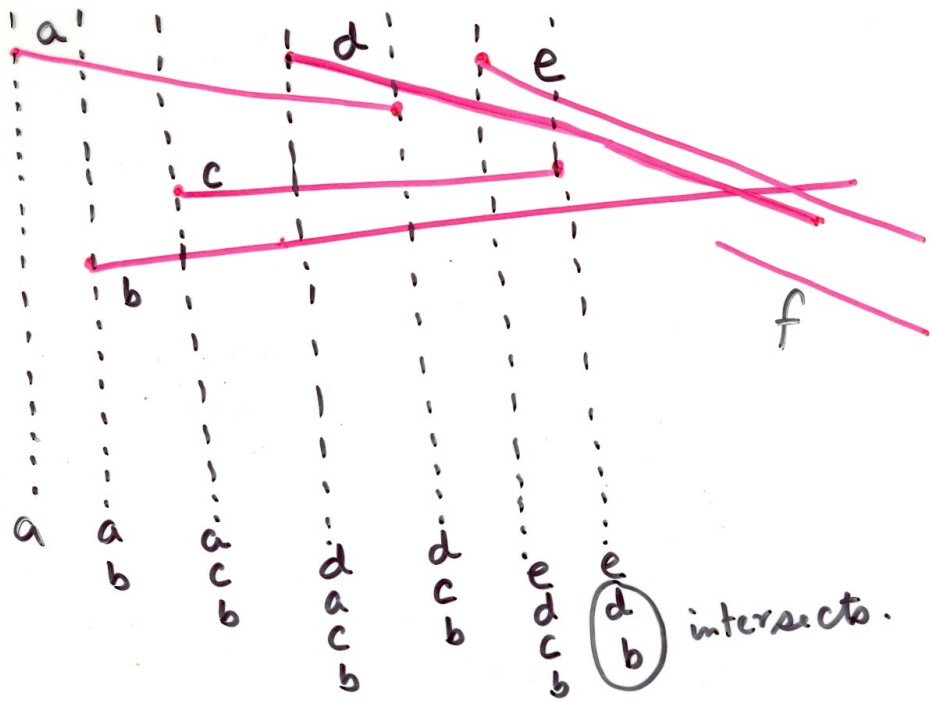
All the above operations can be implemented in $O(\log n)$ time by say red-black tree (Dynamic balanced search tree).

The relative ordering between segments can be determined by cross-product computation.



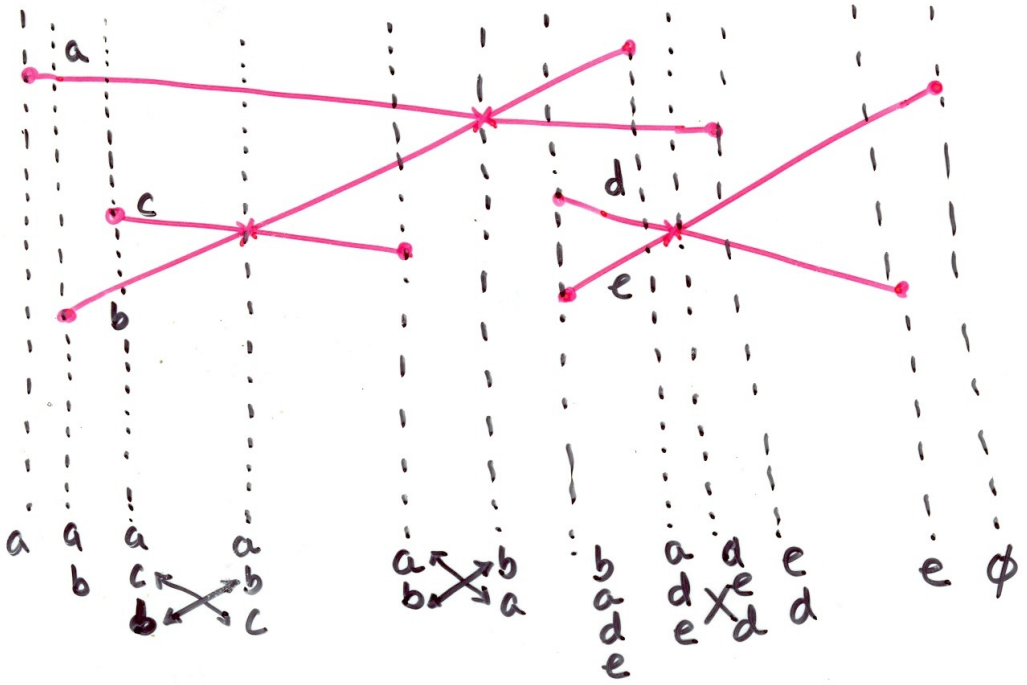
Determine if p below s_1 to determine the order between s_1 & s_2.

```
Any-Segment-intersect(S)
T := φ;
Sort endpoints in S from left to right
for each p in the sorted list of endpoints
    if p is the left endpoint of s
        then Insert(T, s)
             if (Above(T, s) exists &
                 intersects s)
                or (Below(T, s) exists &
                    intersects s)
                    then return True;
    endif
    if p is the right endpoint of s
        if (Above(T, s) & Below(T, s) exist)
           and (Above(T, s) intersects Below(T, s))
               then return True
        endif Delete(T, s)
endfor
return False.
```



Next, we consider the (ii) all intersections detection.

- Now, the events happen also at the intersection points. So, the sweep line stops at the endpoints as well as ~~at~~ the intersection points.
- The event queue is now dynamic as new intersections are detected, they need to be inserted.



- The event queue can be maintained as a priority queue (heap).

- At each stop, check if the new line segments that become consecutive in order T intersect on right. If they do, insert the intersection point in the event queue.

```
All-Segment-intersection(S)
T := φ; Q := φ;
Insert all endpoints of S into Q in the sorted order.
While Q ≠ φ do
    p := dequeue(Q);
    if p is the left endpoint of s then
        Insert(T, s);
        if (Above(T, s) exists and intersects s) then
            insert the intersectction point into Q;
        if (Below(T, s) exists and intersects s) then
            insert the intersection point into Q;
    endif
    if p is the right endpoint of S then
        if (Above(T, s) and Below(T, s) exist)
            and (Above(T, s) intersects Below(T, s))
            then insert the intersection point into Q
        Delete(T, s);
    endif
    if p is an intersection point of s1 & s2
        exchange the order of segments s1 & s2 in T
        report p;
    endif
endwhile
```

* and Compute and insert intersections with new neighbors.

Time = $O((n+s)\log n)$. Space = $O(n+s)$. make it $O(n)$.

Can you make it $O(n\log n + s)$? (Difficult)