Line Segment Intersection

Given n line segments S in \( \mathbb{R}^2 \) (coordinates of endpoints).

Want to report all m intersections between pairs of segments.

Bentley & Ottmann ('79): \( O((n+m) \log n) \) time.

Plane sweep.
Will track event points as a vertical sweep line $l$ goes from left to right.

Maintains:
1) partial solution of intersections left of $l$
2) sweep-line status describing how segments of $S$ intersect $l$
3) event queue holding some
Future events including the next one

Events:
1) Left endpoints
2) Right endpoints
3) Intersections between segments

Can store only type 3 events between segments adjacent along sweep line.
Data Structures:
Event queue as priority queue
with operations:

\[ r \leftarrow \text{insert}(e, x) \text{: insert event } e \text{ with } x \text{-coordinate/'priority'} x + \]

return reference \( r \) to its entry

\[ \text{delete}(r) \text{: delete } r \text{'s entry} \]

\( (e, x) \leftarrow \text{extract-min}() \text{: extract } e \text{ with priority } x \)
Sweep-line status as an ordered dictionary:

- \( r \leftarrow \text{insert}(s) \): insert segment \( s \)
- \( r \leftarrow \text{delete}(r) \)
- \( r' \leftarrow \text{predecessor}(r) \): return a ref \( r' \) to segment immediately above entry for \( r \) (or null if \( r \)'s segment is topmost)
- \( r' \leftarrow \text{successor}(r) \): as before but segment below
- \( r' \leftarrow \text{swap}(r) \): swap order of \( r \) and its successor. Return new ref for the segment of \( r \)
If one holds \( n \) entries, can use \( O(n) \) space \& \( O(\log n) \) time per operation, using e.g. min-heap \& balanced binary search tree.
The Algorithm

Line Segment Intersection Reporting

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. 25(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 an intersection between \( s \) and \( s^+ \), and if so, add it to the event queue. Do the same for \( s \) and \( s^- \).

   **Right endpoint:** (see Fig. 25(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 an intersection between \( s^+ \) and \( s^- \) to the right of the sweep line, and if so, add the corresponding event to the event queue.

   **Intersection:** (see Fig. 25(c))
   
   (a) Let \( s^+ \) and \( s^- \) be the two segments involved (with \( s^+ \) above just prior to the intersection). Report this intersection.
   
   (b) Let \( s^{++} \) and \( s^{--} \) be the segments immediately above and below the intersection. Remove any event involving the pair \((s^+, s^{++})\) and the pair \((s^-, s^{--})\).
   
   (c) Swap \( s^+ \) and \( s^- \) in the sweep-line status (they must be adjacent to each other).
   
   (d) Test for an intersection between \( s^- \) and \( s^{++} \) to the right of the sweep line, and if so, add it to the event queue. Do the same for \( s^+ \) and \( s^{--} \).

---

![Fig. 25: Plane-sweep algorithm event processing.](image-url)
Analysis

Sweep line status has $\leq n$ segments so event queue has size $O(n)$

$\Rightarrow$ events take $O(\log n)$ time each

$2n + m$ events processed, so

$O((n + m) \log n)$ time

There is an $O(n \log n + m)$ time alg. + an $\Omega(n \log n + m)$ lower bound.
Planar Subdivisions (simple)

A planar graph is a graph where you can map vertices to points in $\mathbb{R}^2$ and edges to line segments that are disjoint except at their endpoints, vertices.

![Diagram](image)
Embedding separates plane into regions called faces.

Planar subdivision is the combo of vertices, edges, and faces.

Represent with a doubly-connected edge list (DCEL).
Each edge is represented as a pair of twin directed half-edges. Each has an origin (tail) and destination (head).

Origin of one is destination of other.

Half-edges point to face on their left.
Formally, each vertex $v$ has coordinates $(v)$. Incidental Edge $(v)$: an arbitrary half edge $e$ with $v$ as its origin.

For each face $f$:

- Outer Component $(f)$: an arbitrary half-edge $e$ s.t. $f$ is on left of $e$.

- Inner Components $(f)$: a list of half-edges, one per inner component $st. f$ is on left of $e$. 
For each half-edge $e$:

- **Origin** $(e)$
- **Twin** $(e)$
- **Incident Face** $(e)$: the one on the left
- **Next** $(e)$: next half-edge along the incident face $(e)$
- **Prev** $(e)$

Can do pretty much any "local" thing in constant time per op.
such as edge next vertex leaving $v$ in cow order is Twin (Prev(c)) can even subdivide a face on vertex in constant given next to prev. edges at the subdivision.

Can even subdivide a face on vertex in constant given next to prev. edges at the subdivision.