Line Segment Intersection

Given a set $S$ of $n$ line segments in the plane:

- Do any two segments in $S$ intersect?
- Which pairs of segments intersect?
- Find all points of intersection
- Subdivide $S$ into subsegments that intersect only at endpoints.

Arrangement of $S$ has complexity $O(n \log n)$.

Examples:

- Given a polygon, is it simple?
- Given two polygons, compute their union/intersection.
- Given two planar maps, compute their overlap.

Variants:
- Replace segments with boxes, circles, curves, triangles in $\mathbb{R}^3$, ...

Elementary Cases:

Two segments $pq$ and $rs$ cross if they intersect at a single interior point.

$Pq$ crosses $rs$ and $pq$ crosses $rs$.

We can compute $Pq \cap rs$ by solving linear equations:

\[
\begin{align*}
\text{orient}(p,q,r) &= \text{orient}(p,q,s) \quad \text{and} \\
\text{orient}(p,r,s) &= \text{orient}(q,r,s) \\
\text{orient}(p,q,z) &= 0 \\
\text{orient}(r,s,z) &= 0
\end{align*}
\]

Four 3x3 determinants:

\[
\begin{align*}
(b-d)x + (c-a)y &= ad-bc \\
f-h)x + (g-e)y &= ch-fg
\end{align*}
\]
Not all intersections are crossings!

Still detectable in $O(n \log n)$ time but more cases to consider.

So we can find all intersections in $O(n^2)$ time by brute force.

**Intersection Detection**

Given $n$ line segments, do any two intersect?

[Shamos Hoey 1976] $O(n \log n)$—time sweep algorithm

Imagine a vertical line $L$ sweeping continuously from left to right. Maintain sequence of intersections between $L$ and $S$. Watch for crossings between segments.

Continuous $\rightarrow$ discrete:

Maintain sequence of segments (indices) crossing $L$, not actual intersection points.

Sequence changes only when $L$ reaches:
- segment endpoint
- segment crossing

We know these in advance. Always between adjacent pairs.
Maintain sweep-line status in balanced binary search tree (AVL, red-black, treap, splay, skip list, ...) 
Insert/Delete/Pred/Succ in $O(\log n)$ time

Comparisons implemented using orientation tests:
- $r$ is above $pq_i$ $\Rightarrow$ $pq_i$ is CCW
- $r$ is below $pq_i$ $\Rightarrow$ $pq_i$ is CW

Consider endpoints in order from left to right

$O(\log n)$ time to sort

At left endpt $p_j$:
- $i \leftarrow \text{pred}(p_j)$
- If $p_iq_i$ and $pq_j$ cross, return True
- $k \leftarrow \text{succ}(p_j)$
- If $p_iq_i$ and $pq_k$ cross, return True
- Insert $j$ into BST (using $p_j$ for comparisons)

$O(\log n)$ time per endpoint $\Rightarrow O(n \log n)$ time

At right endpt $q_j$:
- Delete $j$ from BST
- $i \leftarrow \text{pred}(q_j)$
- $k \leftarrow \text{succ}(q_j)$
- If $p_iq_i$ and $pq_k$ cross, return True

If we reach last endpt with no crossings
return False
Reporting all intersecting pairs \cite{BentleyOttmann}

Keep a priority queue of potential crossings.

For each adjacent pair $ij$ in sweep line sequence, store $\Pi_i \cap \Pi_j$.

Priority $= x$-coord but ignore if left of $l$.

Use the priority queue to find next event (seed with endpoints of $S$).

At left endpoint $p_j$:

- $i \leftarrow \text{pred}(j)$
- $k \leftarrow \text{succ}(j)$
- Insert $j$ into BST
- Delete $(ik)$ from PQ
- Insert $(ij)$ into PQ
- Insert $(jk)$ into PQ

At right endpoint $q_j$:

- $i \leftarrow \text{pred}(j)$
- $k \leftarrow \text{succ}(j)$
- Delete $j$ from BST
- Insert $(ik)$ into PQ
- Delete $(ij)$ from PQ
- Delete $(jk)$ from PQ

At crossing $S_j \cap S_k$:

- Report crossing
- $i \leftarrow \text{pred}(j)$
- $l \leftarrow \text{succ}(k)$
- Swap $j, k$ in BST
- Delete $(ij)$ and $(k, l)$ from PQ
- Insert $(i, k)$ and $(j, l)$ into PQ
$O(\log n)$ time per event

$\#\text{events} = \#\text{endpts} + \#\text{crossing pairs}$

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

$k=\text{output size}$