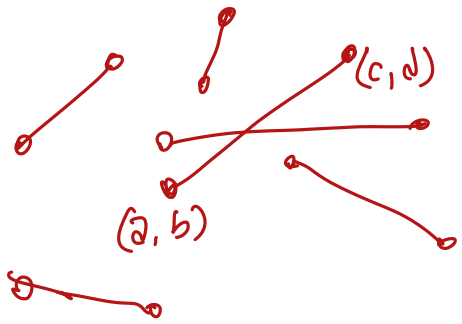
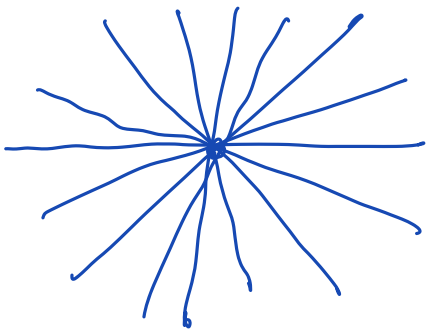
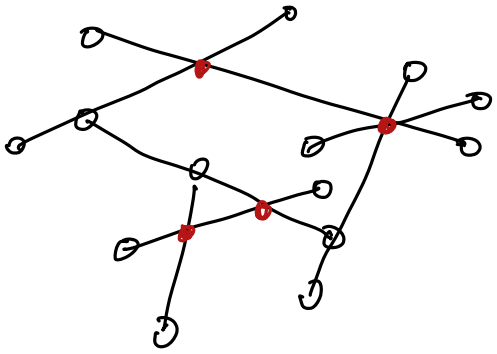
HW1 out later today
HW every 2ish weeks 5 problems #problems = #credits

Line Segment Intersection



Given a set S of n line segments
- Do any two intersect?
- Which pairs intersect?
- How many pairs intersect
- Find/count intersection points
- Subdivide S at all int. points ↳ Arrangement of S





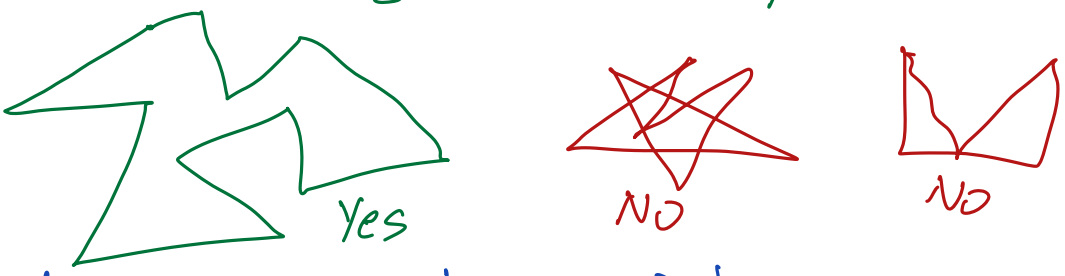
$O(n^2)$ time brute force

Arrangement is planar straight-line graph (PSLG)
Vertices = endpoints & intersection points
Edges = maximal subsegments

Examples:

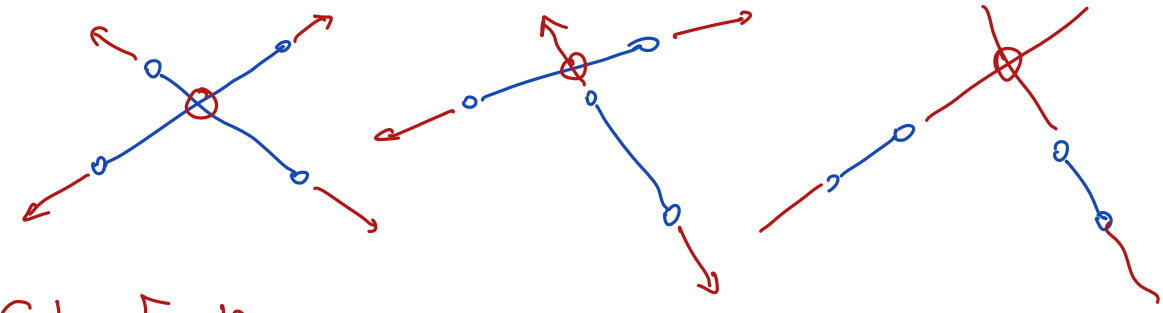
Given 2 polygons, compute union/intersection/convex hull

Given a single polygon, is it simple?

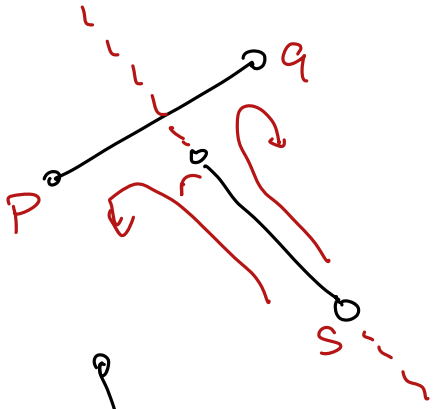


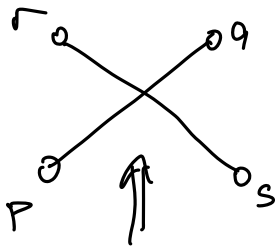
Generalizes to lines, boxes, circles, curves, Δs in 3D

Two segments



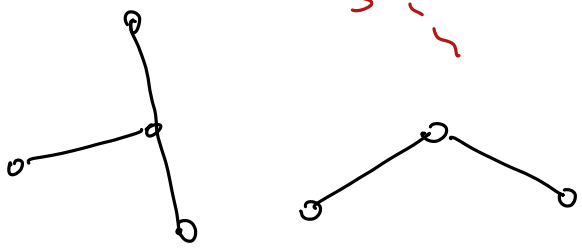
Solve for lines
Solve for point
Check if pt is on segments

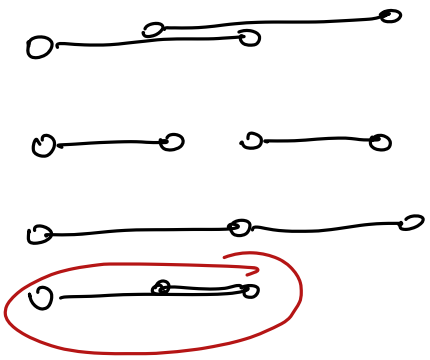




"crossing"

$pq \cap rs \neq \emptyset$
iff
$\text{orient}(pqr) \neq \text{orient}(pqs)$
and
$\text{orient}(prs) \neq \text{orient}(qrs)$





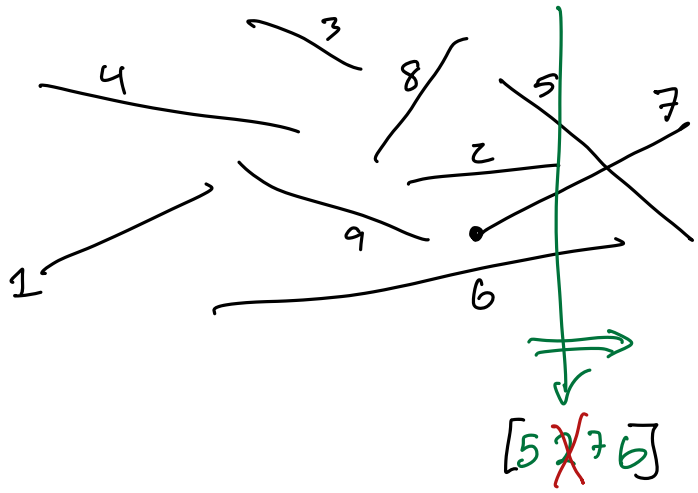
$$\text{orient}(pqr) = \text{sgn det}\begin{bmatrix} 1 & p_1 & p_2 \\ 1 & q_1 & q_2 \\ 1 & r_1 & r_2 \end{bmatrix} \begin{matrix} >0 \\ =0 \\ <0 \end{matrix} \quad \begin{matrix} \text{ccw} \\ \text{flat} \\ \text{cw} \end{matrix}$$

For now assume general position

- no equal coordinates
- no 3 endpoints collinear

Detecting intersections

Use sweep line algorithm



Sweep line maintains sorted seq. of segments it intersects

Left endpoint: insertion

Right endpoint: deletion

Crossings involve adjacent indices

"Sweep line structure" is balanced BST

Implicity use y-coords of intersection with ℓ as "keys" but computed on the fly

Left end: insert $O(\log n)$ time

check new seg and successor
check new seg and predecessor } $O(\log n)$ time

Right end: delete $O(\log n)$ time

check pred + succ of old segment $O(\log n)$ time

```
Sort endpoints ---->
For i <- 1 to 2n
    if P[i] is left
        {
    else
        }
```

$O(n \log n)$

$O(n)$

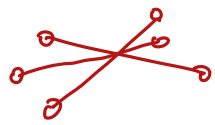
$\times\ O(\log n)$

$\boxed{O(n \log n) \text{ time}}$

Count/list all intersections

swap segments when they intersect

At most 2 segs intersect at one pt



Maintain a priority queue of events "event queue"

Init: insert all 2n endpts into PQ

When we detect intersection

Add x-coord of that intersection to PQ

Init PQ with endpoints

while PQ is not empty

x ← next event from PQ

if x is left end

O(logn) { if new seg + succ intersect
add x-coord of int to PQ
if new seg + pred intersect
add ——————— PQ
update BST }

if x right

O(logn) { }

if x intersection

O(logn) { swap in BST
check 2 new pairs }

[2 5 8 6 4]
↓
[2 8 5 6 4]

Overall running time = O(logn) × # events

$= \boxed{O((n+k)\log n)}$

2n + k

k = # crossings

Later: $\boxed{O(n \log n + k) \text{ time}}$ — different method

Counting intersections: $O(n^{4/3+\varepsilon})$

VERY different method.