Line algorithms
Numerical error

Leaving discrete graphs/sequences in continuous space (Euclidean)

The algorithms in this chapter are not prone to numerical error

They simply use +, - and * (fast) (See: numericalError.py)
Notation

If we have two points $p_0 = (x_0, y_0)$, $p_1 = (x_1, y_1)$ then a line segment from $p_0$ to $p_1$ is: $p_0 \rightarrow p_1$

A cross product between two points: $p_0 \times p_1 = x_0 \times y_1 - y_0 \times x_1$

(grumble... grumble... )
Can also define:
\[ p_0 \times p_1 = \det( \begin{bmatrix} x_0 & x_1 \end{bmatrix} ) \]
\[ \begin{bmatrix} y_0 & y_1 \end{bmatrix} \]
gives area (signed)
Turn left or right?

Right-hand-rule (from physics)
For $p_0 \times p_1$, 
Pointing finger is $p_0$...
Index finger is $p_1$...

Thumbs up => $p_1$ is to the left of $p_0$
Thumbs down => $p_1$ to the right of $p_0$
Turn left or right?

Math version:
\[ p_0 \times p_1 == \text{positive:} \]
\[ p_1 \text{ is to the left of } p_0 \text{(smallest angle)} \]
\[ p_0 \times p_1 == \text{negative:} \]
\[ p_1 \text{ is to the right of } p_0 \]

Tell crossing line segments?
Turn left or right?

Both on left?
Will not go between!
Crossing line segments

One left, one right...

Will go between if we get that far
Crossing line segments

Segments intersect\((p_1, p_2, p_3, p_4)\)
\[d_1 = p_3 \rightarrow p_4 \times p_3 \rightarrow p_1 \quad \text{//} \quad p_1 \quad \text{left of line?}\]
\[d_2 = p_3 \rightarrow p_4 \times p_3 \rightarrow p_2\]
\[d_3 = p_1 \rightarrow p_2 \times p_1 \rightarrow p_3\]
\[d_4 = p_1 \rightarrow p_2 \times p_1 \rightarrow p_4\]
if\(((d_1 \times d_2 < 0) \text{ and } (d_3 \times d_4 < 0))\)
return true
Crossing line segments

Previous algorithm misses if \( p_1, p_2, p_3 \) and \( p_4 \) are all on the same line (cross product is zero) so add to previous:

\[
m = \text{midpoint}(p_1, p_2, p_3, p_4)
\]

if \( d_1 == 0 \) and \( d_2 == 0 \)

if exists \( p \) in \{p_1,p_2\} s.t.

\[
\min(p_3.x , p_4.x) < p.x < \max(p_3.x, p4.x) \text{ and } \min(p_3.y , p_4.y) < p.y < \max(p_3.y,p4.y)
\]

return true

return false
Crossing line segments

Run time?
Crossing line segments

Run time?

$O(1)$
Any crossing line segments

How would we check to see if any line segments from a set cross?
Any crossing line segments

How would we check to see if any line segments from a set cross?

Brute force! Try all possible pairs...
Each pair takes O(1), so runtime is O(n^2) if n pairs

Can we do better?
Any crossing line segments

We can do better, but first we assume there are no vertical lines ($\infty$ slope)

To do this we keep a set of “active” lines and who is above/below

Order the points by $x$, and set them “active” on the start (“off” at end)
Any crossing line segments

There are two cases for crossing:
1. When I first become active, I will cross either the above/below line
2. If I turn “off”, the above/below line can cross in the future
Any crossing line segments

Any-segments-intersect(S)

sort p in S by x, T = empty
for each p (smallest to largest x)
    if p is start point of line segment s
        Insert(T,s)
        if Above(T,s) intersect s OR Below(T,s) intersect s
            return true
    if p is end point of line segment s
        if Above(T,s) intersect Below(T,s)
            return true
        Delete(T,s)
return false
Any crossing line segments

Using red/black trees (Ch 13.) we can do Insert(), Delete(), Above(), and Below() in O(lg n)

Runtime?
Using red/black trees (Ch 13.) we can do Insert(), Delete(), Above(), and Below() in $O(lg \ n)$

Runtime? Overall = $O(n \ lg \ n) < O(n^2)$

sorting: $O(n \ lg \ n)$
loops: $O(n)$ times
inside loop work: $O(lg \ n)$
Any crossing line segments

Using red/black trees (Ch 13.) we can do Insert(), Delete(), Above(), and Below() in $O(\lg n)$

Runtime? Overall = $O(n \lg n) < O(n^2)$

sorting: $O(n \lg n)$

loops: $O(n)$ times inside loop work: $O(\lg n)$
Any crossing line segments

Correctness:
alg. returns true => intersection
Trivial

intersection => alg. returns true

Suppose $p_0 \rightarrow p_1$ intersects $p_2 \rightarrow p_3$, with $p_0.x \leq p_2.x \leq p_3.x \leq p_4.x$

(first line this happens for)
There are 2 cases:
No line segments “activate” between $p_0$ and $p_2$

In this case, when $p_2$ activates, $p_0$ is below/above and the alg. will check if $p_0 \rightarrow p_1$ crosses $p_2 \rightarrow p_3$ (true)
Any crossing line segments

If another segment activates between, it must end (as this does not cross)

When the last line segment before \( p_0 \rightarrow p_1 \) crosses \( p_2 \rightarrow p_3 \) ends, will have \( p_0 \rightarrow p_1 \) above and \( p_2 \rightarrow p_3 \) below (or vica versa), thus alg. will return true
Any crossing line segments

Correctness:
alg. returns true => intersection
Trivial

intersection => alg. returns true

Suppose $p_0 \rightarrow p_1$ intersects $p_2 \rightarrow p_3$, with $p_0.x \leq p_2.x \leq p_3.x \leq p_4.x$