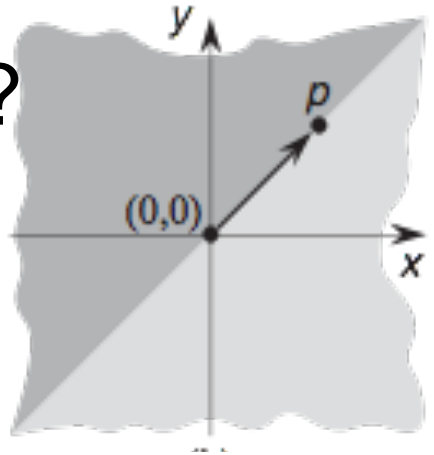
Computational Geometry

Giri Narasimhan
Programming Team
January 9, 2020

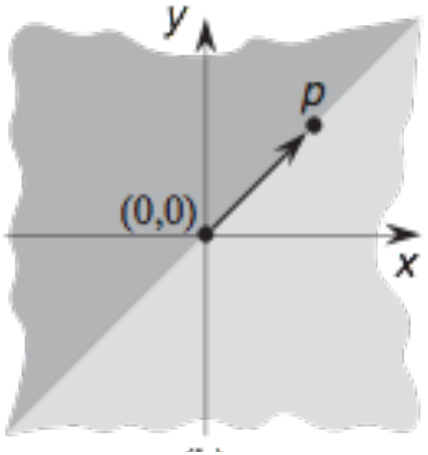
3 important tests

- Given 2 vectors ab and ac, is ab clockwise from ac with respect to a?
- If we traverse from a to b and then to c, do we make a left turn at b?
- Do segments ab and cd intersect?

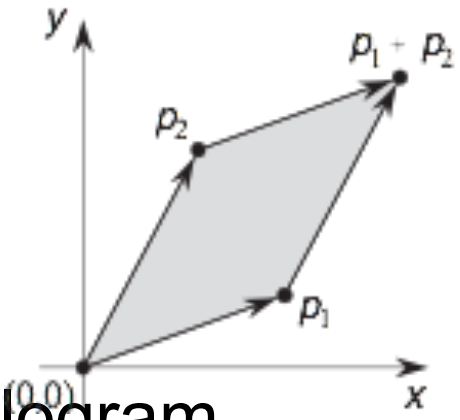


3 important tests

- Given 2 vectors ab and ac, is ab clockwise from ac with respect to a?
- If we traverse from a to b and then to c, do we make a left turn at b?
- Do segments ab and cd intersect?



Cross Products



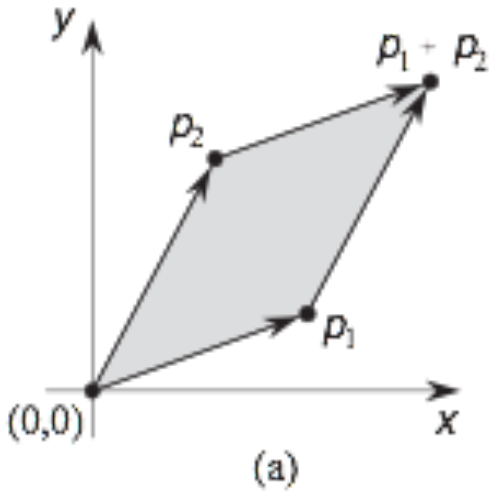
- Let a = origin (0,0)
- Let p_1 = vector from a to b
- Let p_2 = vector from a to c
- Cross product = signed area of parallelogram
- p_1 X p_2 has magnitude = $|x1\ y2 - x2\ y1|$

  $= |(b_x-a_x) * (c_y-a_y) - (c_x-a_x) * (b_y-a_y)|$
- p_1 X p_2 has direction normal to p1 and p2.
  - Use right hand rule
  - Check sign of quantity inside absolute value sign

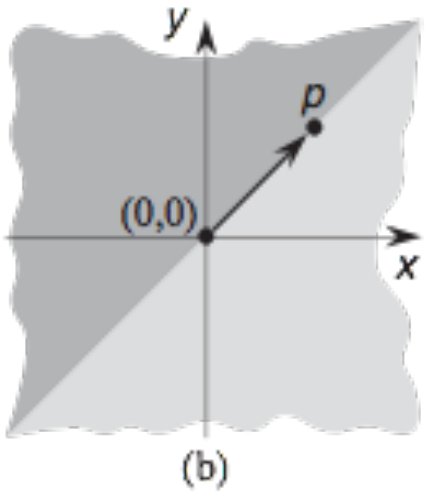
Vectors

- If O is the origin and point $p_1 = (x_1, y_1)$, then the vector $\mathbf{p_1} = (x_1, y_1)$
- If points $a = (a_x, a_y)$ and point $p_1 = (x_1, y_1)$, then the vector from a to p_1 is given by the vector subtraction of vector **a** from vector $\mathbf{p_1}$ given by $\mathbf{p_1 - a}$. This is given by vector:
  - $\mathbf{ap_1} = (x_1 - a_x, y_1 - a_y)$
-

Cross Products & “Clockwiseness”



(a)

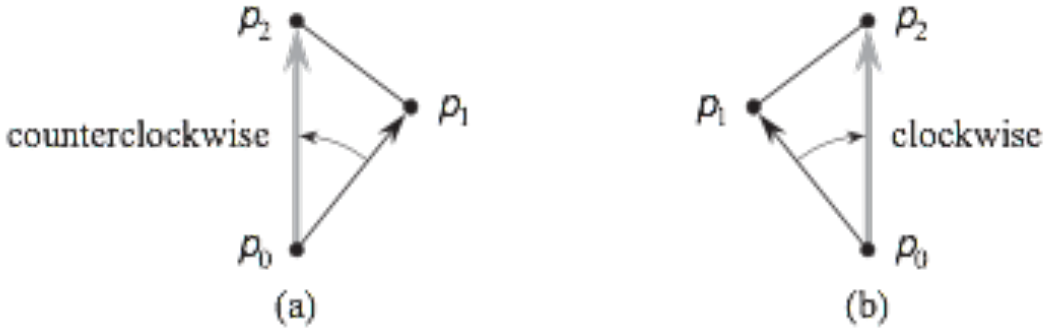


(b)

$$
\begin{aligned}
p_1 \times p_2 &= \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} \\
&= x_1 y_2 - x_2 y_1 \\
&= -p_2 \times p_1 \ .
\end{aligned}
$$

Figure 33.1 **(a)** The cross product of vectors p_1 and p_2 is the signed area of the parallelogram. **(b)** The lightly shaded region contains vectors that are clockwise from p. The darkly shaded region contains vectors that are counterclockwise from p.

“Clockwiseness”



$$(p_1 - p_0) \times (p_2 - p_0) = (x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0) .$$

If this cross product is positive, then $\overrightarrow{p_0 p_1}$ is clockwise from $\overrightarrow{p_0 p_2}$; if negative, it is counterclockwise.

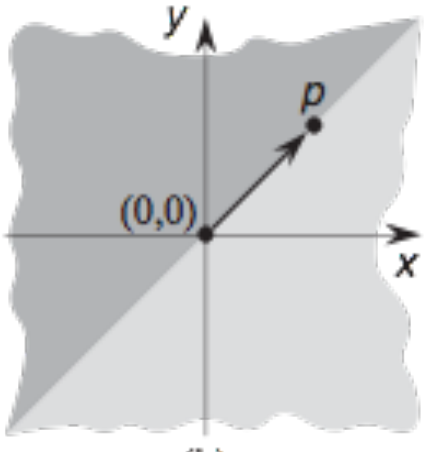
DIRECTION(p_i, p_j, p_k)

1 **return** $(p_k - p_i) \times (p_j - p_i)$

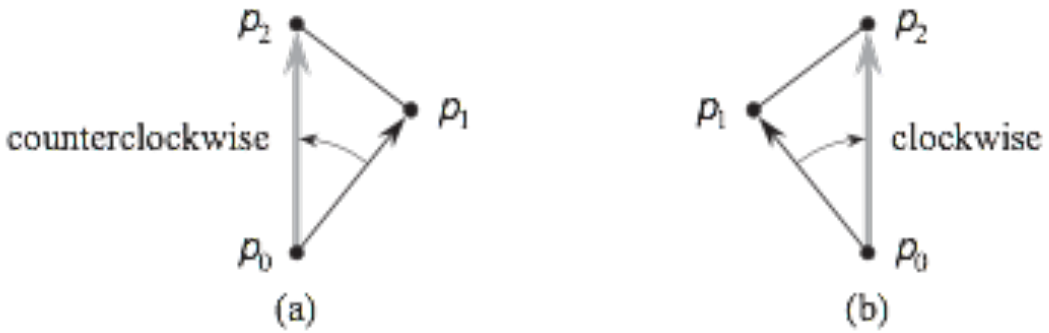
Returns true if p_i to p_k is clockwise of p_i to p_j

3 important tests

- Given 2 vectors ab and ac, is ab clockwise from ac with respect to a?
- If we traverse from a to b and then to c, do we make a left turn at b?
- Do segments ab and cd intersect?



Left Turn Test



$$(p_1 - p_0) \times (p_2 - p_0) = (x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0) .$$

If this cross product is positive, then $\overrightarrow{p_0 p_1}$ is clockwise from $\overrightarrow{p_0 p_2}$; if negative, it is counterclockwise.

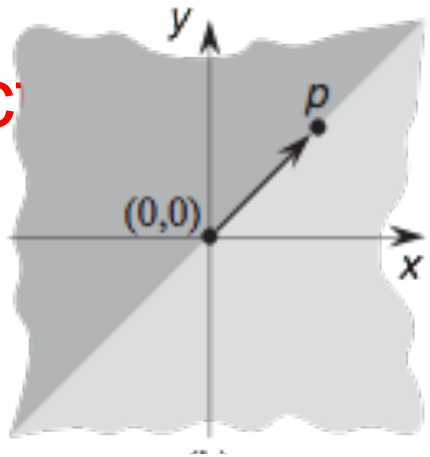
DIRECTION(p_i, p_j, p_k)

1 **return** $(p_k - p_i) \times (p_j - p_i)$

Returns true if p_i to p_k to p_j is a right turn

3 important tests

- Given 2 vectors ab and ac, is ab clockwise from ac with respect to a?
- If we traverse from a to b and then to c, do we make a left turn at b?
- Do segments ab and cd intersec



Segment Intersection Test

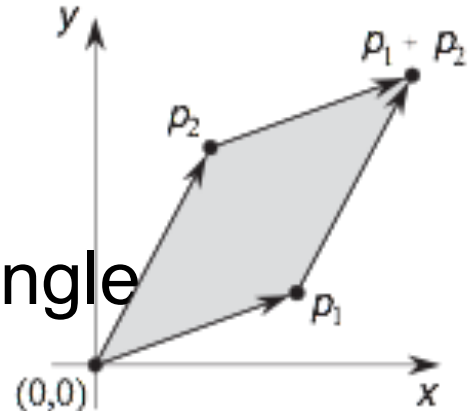
- Standard method
  - Write down equations of two lines
  - Find intersection point
  - If one is found, then the segments intersect
  - Else, they don't intersect
- How can we solve segment intersection using the LEFT-TURN test?

Segment Intersection

```
SEGMENTS-INTERSECT(p1, p2, p3, p4)
 1  d1 = DIRECTION(p3, p4, p1)
 2  d2 = DIRECTION(p3, p4, p2)
 3  d3 = DIRECTION(p1, p2, p3)
 4  d4 = DIRECTION(p1, p2, p4)
 5  if ((d1 > 0 and d2 < 0) or (d1 < 0 and d2 > 0)) and
        ((d3 > 0 and d4 < 0) or (d3 < 0 and d4 > 0))
 6      return TRUE
 7  elseif d1 == 0 and ON-SEGMENT(p3, p4, p1)
 8      return TRUE
 9  elseif d2 == 0 and ON-SEGMENT(p3, p4, p2)
10      return TRUE
11  elseif d3 == 0 and ON-SEGMENT(p1, p2, p3)
12      return TRUE
13  elseif d4 == 0 and ON-SEGMENT(p1, p2, p4)
14      return TRUE
15  else return FALSE
```

Area of a Triangle

- Area = Base X Height / 2
- Area = a X b X sin(C) / 2
  - a, b are side lengths, C is internal angle
- Area = sqrt{s (s-a) (s-b) (s-c)},
  - a,b,c are side lengths and s = half of perimeter
- Area = ½ (cross product magnitude)
  - Area = ½ |x1 y2 – x2 y1|
  - Assumes one vertex is the origin



Sorting points by polar angle

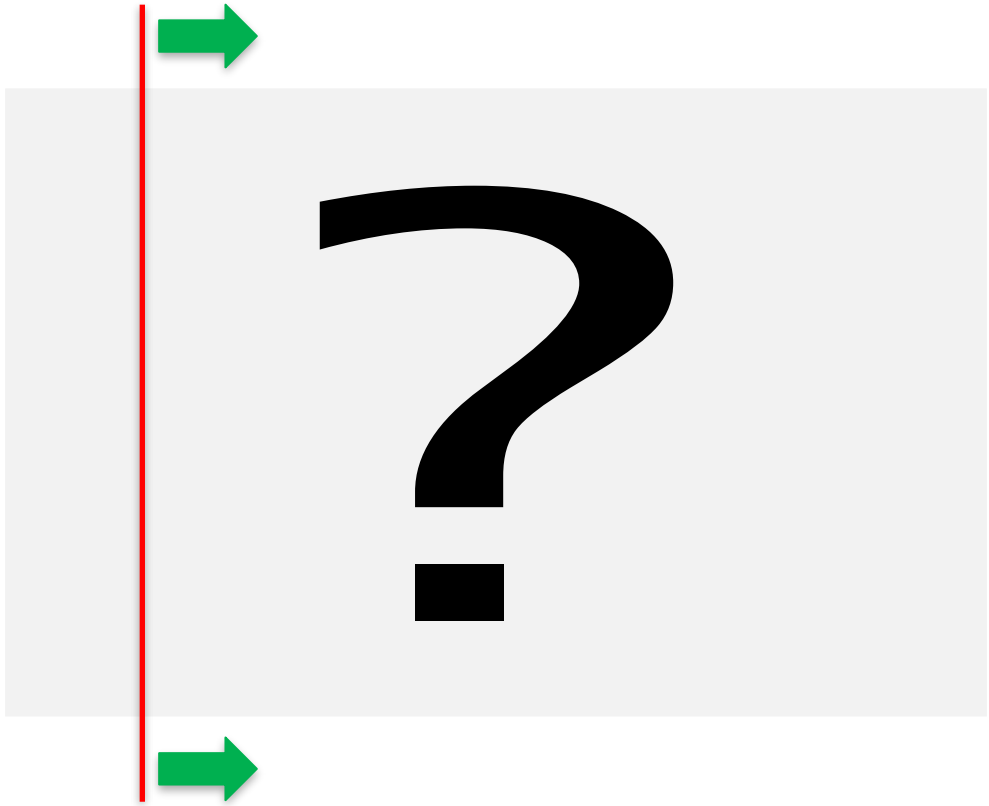
```
struct Point {int x,y;}

int operator^(Point p1, Point p2) {return p1.x*p2.y - p1.y*p2.x;}

bool operator<(Point p1, Point p2)
{
    if (p1.y == 0 && p1.x > 0) return true; //angle of p1 is 0, thus p2>p1
    if (p2.y == 0 && p2.x > 0) return false; //angle of p2 is 0 , thus p1>p2
    if (p1.y > 0 && p2.y < 0) return true; //p1 is in [0..180], p2 in [180..360]
    if (p1.y < 0 && p2.y > 0) return false;
    return (p1^p2) > 0; //return true if p1 is clockwise from p2
}
```

Rectilinear Segment Intersection

Given **n** horizontal & vertical line segments

- Do any pairs of line segments intersect?
- How many intersections occur in the set?
- Report all intersections

Line/Plane Sweep Technique



Report intersections as we sweep over vertical line segments.

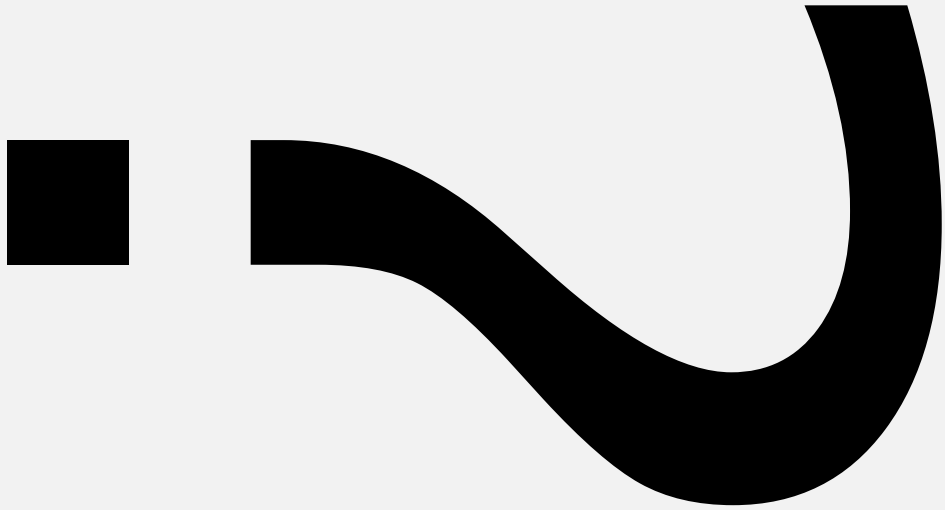
Plane Sweep Technique

- All intersections occur only at x coordinate of some vertical line.
- Maintain **Event Queue**, **Q**, with all possible "events" and report relevant events.
- Sort all endpoints by x coordinates & store in Q.
- Each x in Q is either
  - Start or end of horizontal line
  - The x coordinate of vertical line

Need a Data Structure

- Need a data structure to store all relevant horizontal line segments
- The data structure must **order** all horizontal segments from bottom to top
- This makes it convenient to search for intersections

Rectilinear Line Intersection Algorithm



Time Complexity

- $O(n \log n + R)$

General segments intersection

- Earlier, we knew that 2 horizontal lines never intersect. So there was no need to test this.
- Now any 2 line segments can intersect.
- How to test this?

Segments Intersection

```
ANY-SEGMENTS-INTERSECT(S)
 1  T = ∅
 2  sort the endpoints of the segments in S from left to right,
        breaking ties by putting left endpoints before right endpoints
        and breaking further ties by putting points with lower
        y-coordinates first
 3  for each point p in the sorted list of endpoints
 4      if p is the left endpoint of a segment s
 5          INSERT(T, s)
 6          if (ABOVE(T, s) exists and intersects s)
                or (BELOW(T, s) exists and intersects s)
 7              return TRUE
 8      if p is the right endpoint of a segment s
 9          if both ABOVE(T, s) and BELOW(T, s) exist
                and ABOVE(T, s) intersects BELOW(T, s)
10              return TRUE
11          DELETE(T, s)
12  return FALSE
```

Can be modified to report all intersections between segments

Add an event for a possible intersection between s and ABOVE(T,s) & BELOW(T,s)

Add an event for a possible intersection between ABOVE(T,s) and BELOW(T,s)