Competitive Programming

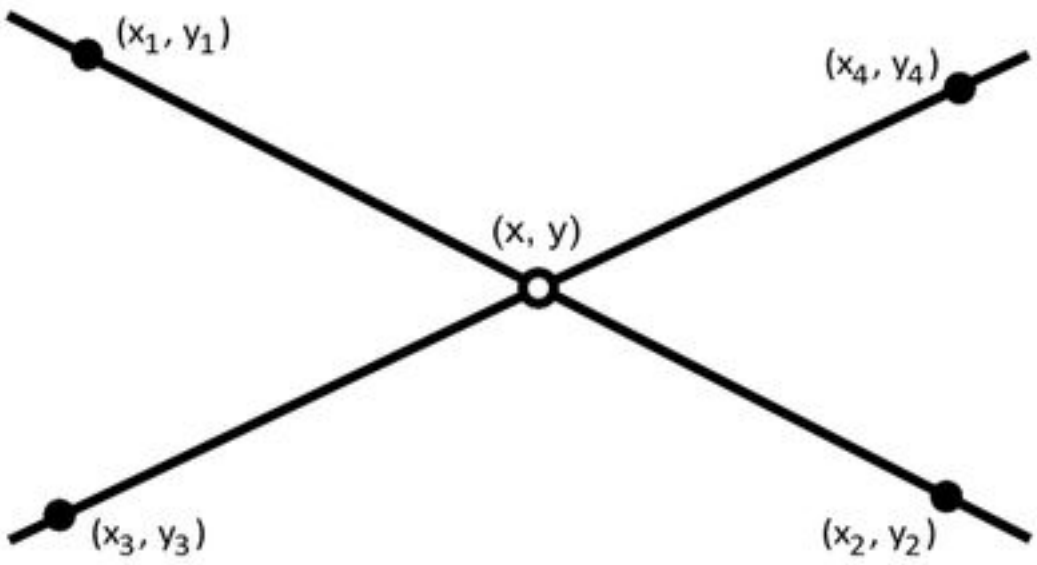
From Problem 2 Solution in O(1)

Computational Geometry

Lines Intersection

Mostafa Saad Ibrahim
PhD Student @ Simon Fraser University

Line Line Intersection



Src: https://sheridanmath.wikispaces.com/file/view/GEOMETRY_04.GIF/177066721/GEOMETRY_04.GIF

2D Lines Intersection

- In such **intersection** either no solution (parallel) or one solution only
- Assume 2D explicit line format
  - a1x+b1y=c1
  - a2x+b2y=c2
- if a1b2−a2b1=0
  - parallel (or identical) lines
- Otherwise, just solve *system of equations*
- You can do that by several ways

2D Lines Intersection

- One way is using Cramer's rule
  - Cramer is inefficient, but ok for 2x2 matrix
  - It is instable actually
- Cramer is based on determinant

$$(x, y) = \left(\frac{c_1 b_2 - b_1 c_2}{a_1 b_2 - b_1 a_2}, \frac{a_1 c_2 - c_1 a_2}{a_1 b_2 - b_1 a_2} \right)$$

Issues

- What about Segment/Segment Intersection?
  - One can simply compute the intersection point and check if it lies on the segment/ray
- What about 3D lines?
  - 2D lines are either parallel or intersecting
  - But 3D and more are rarely intersecting
  - Also, linear projections onto a 2D plane will also intersect
- Overall, we can use this method
- However, let's move to the parametric style

Parametric equations

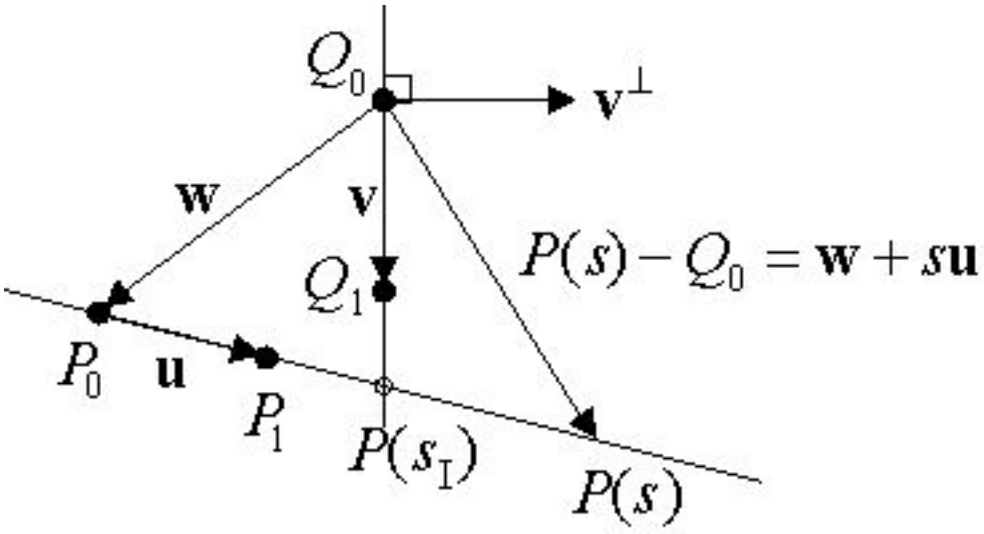
- Remember each line/segment has t parameter
  - E.g. from [0-1] express on segment
- Let first line has t1 and 2nd has t2
- Then
  - X value for first line1 at t1 = X value for first line2 at t2
  - Y value for first line1 at t1 = Y value for first line2 at t2
- using these information, we can create 2 equations and solve them to get t1 and t2
- Using them we can compute the point or validtate segment/ray things

Parametric equations

- Specifically, Assume points a, b, c, d
- Assume line 1 is ab and 2nd is cd
- $a.x + (b.x - a.x)*t1 = c.x + (d.x - c.x)*t2$ (1)
- $a.y + (b.y - a.y)*t1 = c.y + (d.y - c.y)*t2$ (2)
- One can use Cramer's to solve them
- Or even by hand computations
  - E.g. multiply first by (d.y - c.y) and
  - second by (d.x - c.x)
  - then subtract equations and rearrange

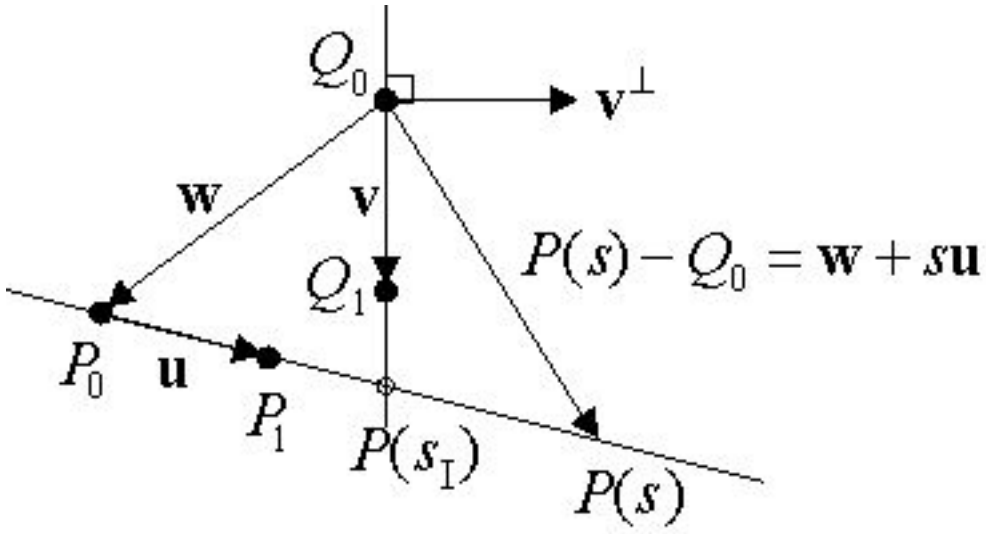
Parametric equations (2)

- We already showed how to do it
  - Let's do it using other (harder) way
- Assume lines P0P1 and Q0Q1
  - Their vectors u and v



Parametric equations (2)

- S1 is the intersecting point at line P0P1
- Can we make a solvable equation based on P(s1)?
- Notice $v.v^t = 0$ (dot product between perpendicular vecs)
- From vector addition: $v = w + s_1 u$
- Then $v^t.(w + s_1 u) = 0$

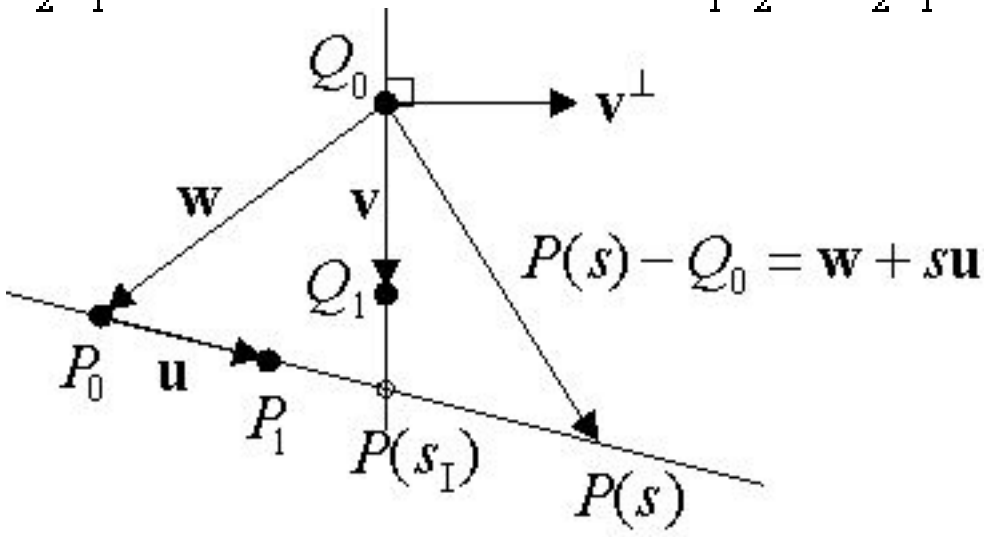


Parametric equations (2)

- Solving this equation for line 1 and (similar way to line 2)

$$\mathbf{v}^{\perp} \bullet (\mathbf{w} + s_{\mathrm{I}}\mathbf{u}) = 0$$

$$s_{\mathrm{I}} = \frac{-\mathbf{v}^{\perp} \cdot \mathbf{w}}{\mathbf{v}^{\perp} \cdot \mathbf{u}} = \frac{v_2 w_1 - v_1 w_2}{v_1 u_2 - v_2 u_1} \qquad t_{\mathrm{I}} = \frac{\mathbf{u}^{\perp} \cdot \mathbf{w}}{\mathbf{u}^{\perp} \cdot \mathbf{v}} = \frac{u_1 w_2 - u_2 w_1}{u_1 v_2 - u_2 v_1}$$



Parametric equations (2)

```
bool intersectSegments(point a, point b, point c, point d, point & intersect) {
  double d1 = cp(a - b, d - c), d2 = cp(a - c, d - c), d3 = cp(a - b, a - c);
  if (fabs(d1) < EPS)
    return false;  // Parllel || identical

  double t1 = d2 / d1, t2 = d3 / d1;
  intersect = a + (b - a) * t1;

  if (t1 < -EPS || t2 < -EPS || t2 > 1 + EPS)
    return false;  //e.g ab is ray, cd is segment ... change to whatever
  return true;
}
```

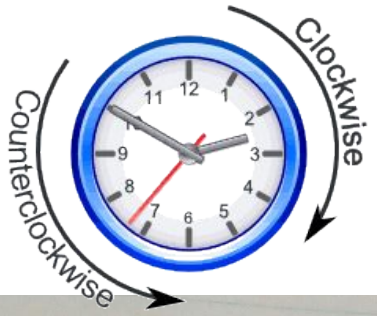
Is segment segment intersect?

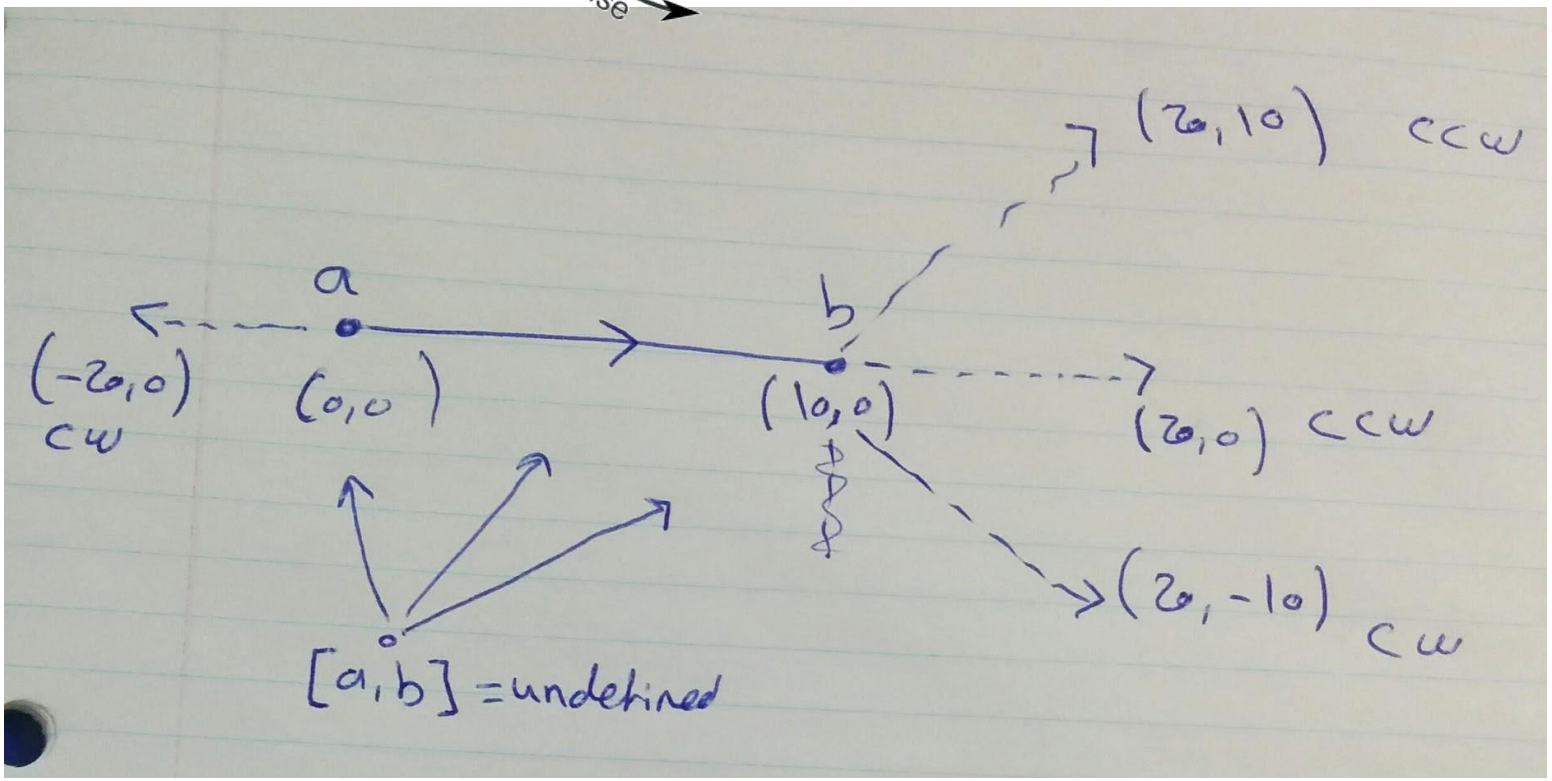
- What if all what we need to know is this boolean answer?
- Let's first know the *counter (anti) clockwise test*
  - CCW is a nice utility for some programs (e.g. test point inside a triangle)

Counterclockwise test

- Given three points a, b, c
- Think about the *path from a to b to c*
- Either it goes counterclockwise
- Or goes clockwise
- Or it doesn't in some collinear cases
- So we have **3 different responses**
  - CCW, CW, Undefined
- Let answer for any point between a-b inclusive to be undefined

Counterclockwise test

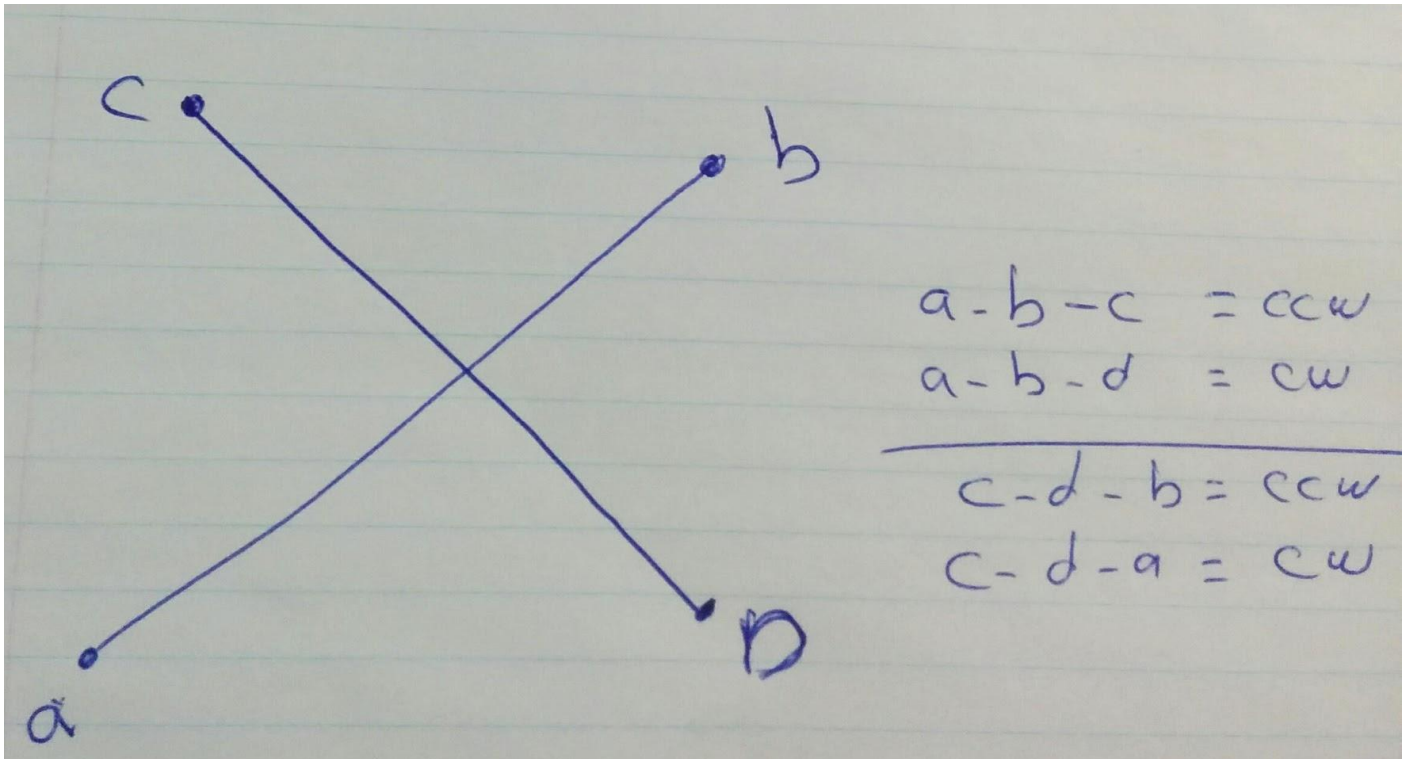




Counterclockwise test

```
// Where is P2 relative to segment p0-p1?
// ccw = +1 => angle > 0 or collinear after p1
// cw = -1 => angle < 0 or collinear after p0
// Undefined = 0 => Collinar in range [a, b]. Be careful here
int ccw( point p0, point p1, point p2 ) {
    point v1(p1-p0), v2(p2-p0);
    if ( cp(v1, v2) > +EPS)                return +1;
    if ( cp(v1, v2) < -EPS)                return -1;
    if (v1.X*v2.X < -EPS || v1.Y*v2.Y < -EPS)
        return -1;
    if ( norm(v1) < norm(v2)-EPS )  return +1;
    return  0;
}
```

Is segment segment intersect?



Is segment segment intersect?

```
bool intersect(point p1, point p2, point p3, point p4) {
  // special case handling if a segment is just a point
  bool x = (p1 == p2), y = (p3==p4);
  if(x && y)  return p1 == p3;
  if(x)   return ccw(p3, p4, p1) == 0;
  if(y)   return ccw(p1, p2, p3) == 0;

  return  ccw(p1, p2, p3) * ccw(p1, p2, p4) <= 0  &&
      ccw(p3, p4, p1) * ccw(p3, p4, p2) <= 0;
}
```

Other intersections types

- There are other types of intersections, but they are rare (if any) in competitive programming
  - Intersection of a Ray/Segment with a Plane
  - Intersection of a Ray/Segment with a Triangle
  - Intersection of a Triangle with a Plane
  - Intersection of a Triangle with a Triangle
  - Line-Plane Intersection
  - Intersection of 2 Planes
  - Intersection of 3 Planes
- For reading in these types: See1, See2

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً