Computational Geometry Session

material prepared by Coach Mohamed Mahmoud AbdelWahab

References:

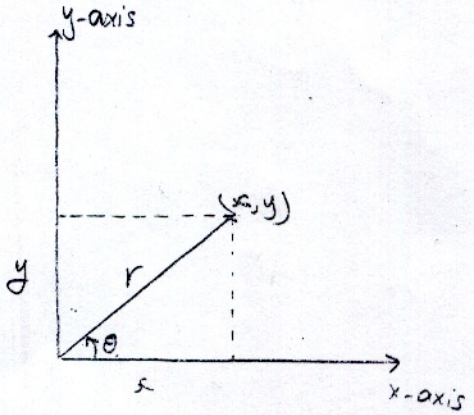
- "Geometry using Complex Numbers", Topcoder Featured Articles by Bruce Merry
- Competitive Programming 2 by Steven Halim
- Introduction to Algorithms, Computational Geometry chapter, by Thomas Cormen.

Point:

- Point is the basic object used in Geometry.
- Point in 2D plan can be represented in 2 ways:
  1) (x, y): pair of (x, y) coordinates
  2) (r, θ): magnitude & angle
- Transferring from 2 representations

$x = r\cos\theta$, $y = r\sin\theta$

while, $r = \sqrt{x^2 + y^2}$, $\theta = \tan^{-1}\left(\frac{y}{x}\right)$



Coding wise:

```
r = sqrt(x*x + y*y),   theta = atan(y/x)
```

Using sqrt() is not recommended as squaring x & y values may lead to overflow & precision errors, so it is preferred to use hypot(x, y) in C++ & Math.hypot(x, y) in Java

Also using atan(y/x) is not recommended too. As the Range of atan() return value is $[-\frac{\pi}{2}, \frac{\pi}{2}]$, so it will be hard to determine the angle so it is required to check the quart from the signs of y & x to determine the actual value of θ.

So it is recommended to use atan2(y, x) as the range of the return value is $[-\pi, \pi]$

→ slow (but no overflow)

∴ `r = hypot(x, y)`, `theta = atan2(y, x)`

Note:

* Don't use hypot(x, y) except when you need it, you can use squared distance if comparison is needed.

CMP NOTE:

All angles used are in the radian form

It is clear that there is a big relation between point's representations and Complex numbers representation, then This tutorial will target C++ Coders so as to use Complex Library in C++

∴ point = $x + yi = R(\cos\theta + i\sin\theta) = Re^{\theta i}$

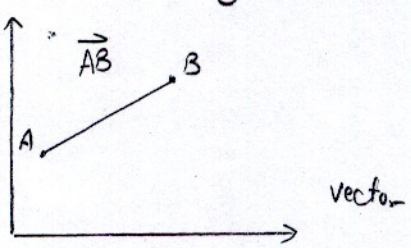
C++ Code

```
#include<complex>
#include<cstdlib>
#include<algorithm>
#include<vector>

typedef complex<double> point; //ToDo choosing type depending on the problem
#define X real()
#define Y imag()
#define polar(r,t) ((r)*exp(point(0,(t))))
#define length(a) hypot((a).X,(a).Y)
#define angle(a) atan2((a).Y,(a).X)
```

Vector:

A vector is an equivalence class of all directed segments of the same length and direction



Dot Product: given two vectors $V_1(x_1, y_1)$, $V_2(x_2, y_2)$

dot product $(V_1, V_2) = x_1 x_2 + y_1 y_2$

Generally, given two vectors $a = [a_1, a_2, a_3, \cdots, a_n]$ and $b = [b_1, b_2, b_3, \cdots, b_n]$

dot product $(a, b) = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$

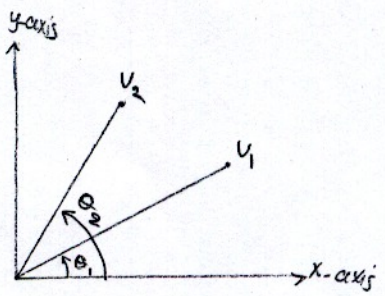
and it is called Scalar product too.

* dot product in the other representation

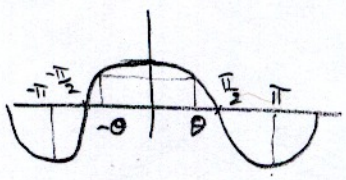
dot product $(V_1, V_2) = r_1 \cos\theta_1 \; r_2 \cos\theta_2 + r_1 \sin\theta_1 \; r_2 \sin\theta_2$

$= r_1 r_2 (\cos\theta_1 \cos\theta_2 + \sin\theta_1 \sin\theta_2)$

$= r_1 r_2 \cos(\theta_1 - \theta_2)$



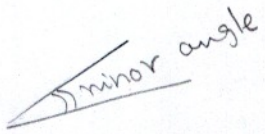
∵ Cos function is symmetric about y-axis so it doesn't matter if $(\theta_1 - \theta_2)$ or $(\theta_2 - \theta_1)$ is used, as $\cos(\theta) = \cos(-\theta)$ as shown in the figure



$\therefore$ dot product $(V_1, V_2) = r_1 r_2 \cos(\theta_1 - \theta_2)$ magnitude always +ve

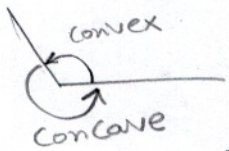
and $r_1 r_2$ is a product of magnitudes & can't be -ve value.

$\therefore$ if dot product = $\begin{cases} +ve & \cos(\theta_1 - \theta_2) > 0 \quad \text{acute angle} \\ 0 & \cos(\theta_1 - \theta_2) = 0 \quad \text{Right angle} \\ -ve & \cos(\theta_1 - \theta_2) < 0 \quad \text{obtuse angle} \end{cases}$



Test: Type of minor angle between two vectors (acute, Right, Obtuse)

* use dot product sign check



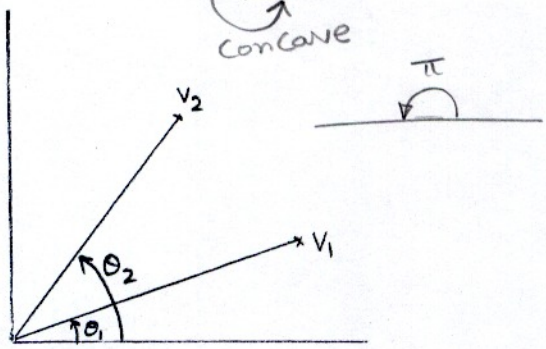
Cross product:

cross product $(V_1, V_2) = x_1 y_2 - x_2 y_1$

$= r_1 \cos\theta_1 \, r_2 \sin\theta_2 - r_2 \cos\theta_2 \, r_1 \sin\theta_1$

$= r_1 r_2 (\cos\theta_1 \sin\theta_2 - \cos\theta_2 \sin\theta_1)$

$= r_1 r_2 \sin(\theta_2 - \theta_1)$



Cross product is strict about the sign of the angle as sine function is not symmetric about y-axis

ccw

if cross product = $\begin{cases} +ve & \sin(\theta_2 - \theta_1) > 0, \text{ angle between two vectors } V_1 V_2 \text{ is convex} \\ 0 & \sin(\theta_2 - \theta_1) = 0, \sim \sim \sim \sim \text{ is } 0 \text{ or } \pi \text{ (two vectors colinear)} \\ -ve & \sin(\theta_2 - \theta_1) < 0, \sim \sim \sim \sim V_1 V_2 \text{ is Concave} \end{cases}$

cw

Dot and cross Products in Complex numbers:

Conjugate in the for $Re^{\theta i} \rightarrow Re^{-\theta i}$

$V_1 = x_1 + y_1 i \quad , \quad V_2 = x_2 + y_2 i$

$\therefore \text{Conj}(V_1) * V_2 = (x_1 - y_1 i) * (x_2 + y_2 i)$

$= \underbrace{(x_1 x_2 + y_1 y_2)}_{\text{dot product}} + \underbrace{(x_1 y_2 - x_2 y_1)}_{\text{cross product}} i$

C++ Code

```
#define vec(a, b)      ((b) - (a))
#define dot(a, b)      ((conj(a) * (b)).real())
#define cross(a, b)    ((conj(a) * (b)).imag())
#define lengthSqr(a)   dot(a, a)
```

Vector Rotation:

Given a vector $v = Re^{\alpha i}$ and required to rotate this vector by angle θ to get $v' = Re^{(\alpha+\theta)i}$

C++ Code

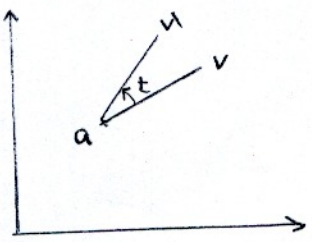
```
#define rotate(v,t)  ((v) * exp(point(0,t)))
#define rotateabout(v,t,a) (rotate(vec(a,v),t)+(a))
```

$v = re^{\alpha i}$

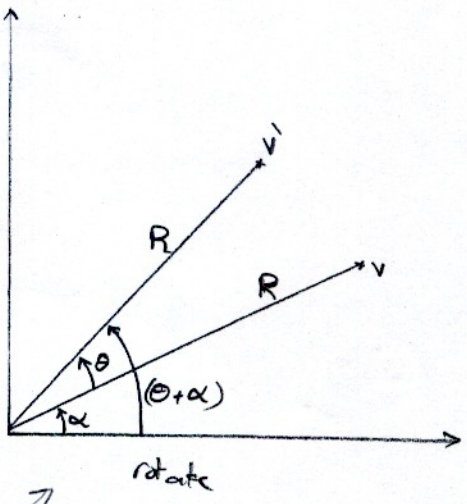
$v' = re^{\alpha i} * e^{\theta i}$

$= re^{(\alpha+\theta)i}$

$v' = r + polar(1,t)$



rotate about



rotate

Vector Rotation using matrices:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned}$$

↓ Rotation Matrix

$$\Downarrow$$

$$\begin{aligned} x' &= R\cos\alpha\cos\theta - R\sin\alpha\sin\theta = R\cos(\alpha+\theta) \\ y' &= R\cos\alpha\sin\theta + R\sin\alpha\cos\theta = R\sin(\alpha+\theta) \end{aligned}$$

$$\therefore v' = R(\cos(\alpha+\theta) + i\sin(\alpha+\theta)) = Re^{(\alpha+\theta)i}$$

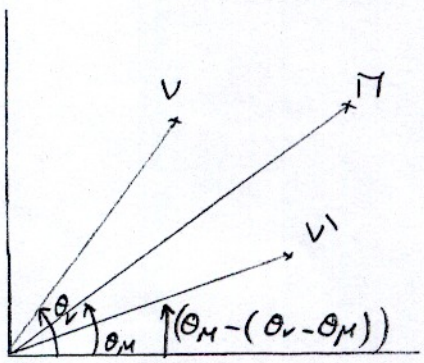
Vector Reflection:

$\text{Conj}\left(\frac{v}{M}\right) * M$

$= \text{Conj}\left(\frac{R_v e^{\theta_v i}}{R_M e^{\theta_M i}}\right) * R_M e^{\theta_M i}$

$\frac{R_v}{R_M} e^{(\theta_M - \theta_v)i} * R_M e^{\theta_M i}$

$R_v e^{(\theta_M - (\theta_v - \theta_M))i}$

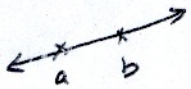


C++ Code

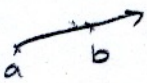
```
#define reflect(v,M)  (conj((v)/(M)) * (M))
```

Line:

* St. line: defined by any two different points on the line



* Ray: defined by starting point a & another point defining the direction of the Ray

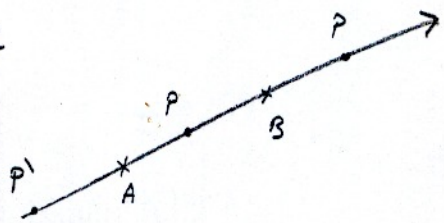


* Line segment: defined by starting & ending points of the segment.

Test: Point on a Ray

① check if P is on the same line of AB using
cross (B−A, P−A) == 0

② check if angle between AB & AP is acute (= 0) not obtuse (= π) using
dot (B−A, P−A) < 0



C++ Code:

```
bool pointOnRay(const point& a, const point& b, const point& p){
    return fabs(cross(vec(a,b), vec(a,p))) < EPS &&
           dot(vec(a,b), vec(a,p)) > -EPS;
}
```

Test: Point on line:

* only check #1 in pointOnRay test is enough

C++ Code

```
bool pointOnLine(const point& a, const point& b, const point& p){
    return fabs(cross(vec(a,b), vec(a,p))) < EPS;
}
```

Test: Point on Segment:

C++ Code

```
bool pointOnSegment(const point& a, const point& b, const point& p){
    if (lengthSqr(vec(a,b)) < EPS) return lengthSqr(vec(a,p)) < EPS;
    return pointOnRay(a,b,p) && pointOnRay(b,a,p);
}
```

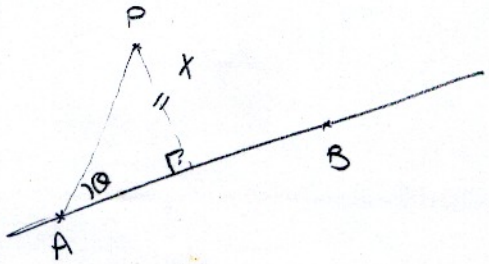
OR use pointonline check & every coordinate of P is in the range $[\min(x_1,x_2), \max(x_1,x_2)]$ & $[\min(y_1,y_2), \max(y_1,y_2)]$

↑ case of vertical line

Point-Line perpendicular distance:

$$X = |AP \sin\theta|$$
$$= \frac{|AP * AB \sin\theta|}{|AB|}$$

$$\boxed{X = \frac{|\text{cross}(B-A, P-A)|}{|AB|}}$$

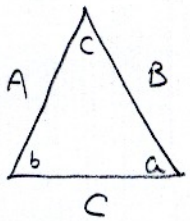


C++ Code

```
double pointLineDistance (const point& a, const point& b, const point& p){
    return fabs(cross(vec(a,b), vec(a,p)) / dist(a,b));
}
```

Triangle:

* one of the most important Triangle properties is that sum of any two sides of triangle is greater than the 3rd line.



Sine and Cosine rule:

Given any triple of a, b, c, A, B, C in a triangle except (a, b, c), it is possible to get the other data using sine & cosine rules

$$\frac{\sin a}{A} = \frac{\sin b}{B} = \frac{\sin C}{C}$$

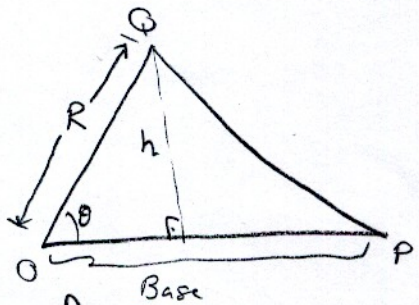
$$\cos C = \frac{A^2 + B^2 - C^2}{2AB} \quad \text{or} \quad C^2 = A^2 + B^2 - 2AB\cos C$$

before calling acos(div)
check if (div > 1)
div = 1;
else if (div < -1)
div = -1;

Area of a triangle:

$$\text{cross}(\overline{OP}, \overline{OQ}) = \overline{OP} \times \overline{OQ}$$
$$= RB\sin\theta$$
$$= B(R\sin\theta)$$
$$= B * h$$

$$\boxed{\text{area} = \frac{1}{2}|\text{cross}(\overline{OP}, \overline{OQ})|}$$ used if coordinates of the triangle are given.



* If side lengths of triangle is only given used Heron's Formula

$S = (A+B+C)/2$ (S is called semi-perimeter)

$$\text{Area} = \sqrt{S(S-A)(S-B)(S-C)}$$

Lines :-

$y = mx + c \Rightarrow y = \frac{\Delta y}{\Delta x} x + c$

∴ $\Delta x\, y - \Delta y\, x + c = 0$

> 0 : P

$= 0$: P

< 0 : P

∴ Line general form $Ax + By = C$ given two points on the line (x_1, y_1) & (x_2, y_2)

∴ $A = y_2 - y_1$

$B = x_1 - x_2$

$C = Ax_1 + By_1$

∴ Given two lines find the point of intersection :

$A_1 x + B_1 y = C_1$

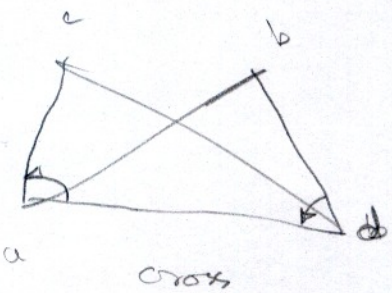
$A_2 x + B_2 y = C_2$

using cramer's Rule

$$\Delta = \begin{vmatrix} A_1 & B_1 \\ A_2 & B_2 \end{vmatrix} = A_1 B_2 - A_2 B_1$$

if ($\Delta == 0$)

then two lines are parallel & no solution.



cross AB, AC

cross AD, AB

} same sign ⇒ intersect

else

$$x = \frac{\begin{vmatrix} C_1 & B_1 \\ C_2 & B_2 \end{vmatrix}}{\Delta} = (C_1 * B_2 - B_1 * C_2) / \Delta$$

$$y = \frac{\begin{vmatrix} A_1 & C_1 \\ A_2 & C_2 \end{vmatrix}}{\Delta} = (A_1 * C_2 - A_2 * C_1) / \Delta$$

What is the difference between Line-Line intersection & segment-segment intersection?

By reformulating the previous eqns, it is possible to use cross product to find point of intersections as follows

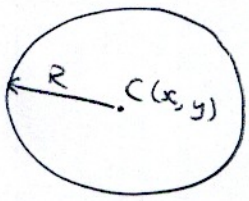
```
bool intersect(const point& a, const point& b, const point& p, const point& q, point& ret){
    double d1 = cross(p-a, b-a);
    double d2 = cross(q-a, b-a);
    ret = (d1 * q - d2 * p) / (d1 - d2);
    return fabs(d1-d2) > EPS;
}
```

⇒ typedef mytype double, - - -

mytype

Circle:

* Circle is defined by center and a Radius

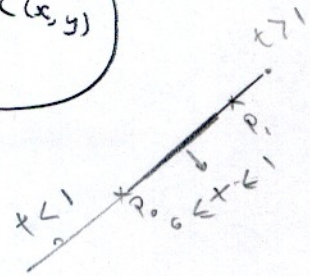


eqn:

$$(x - x_c)^2 + (y - y_c)^2 = R^2$$

OR

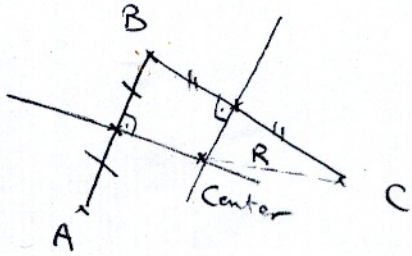
$$(P - C) \cdot (P - C) = R^2$$



Note:
Line parametric eqn is: $P = P_0 + t(P_1 - P_0)$, $0 \le t \le 1$ to get any point P on the line segment.

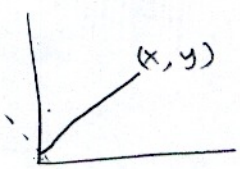
Find Circle by 3 points:

get perpendicular bisector of AB & BC, then the center of the circle is the intersection point of the two bisectors.
And the Radius is the distance between the center & any point of the three points.

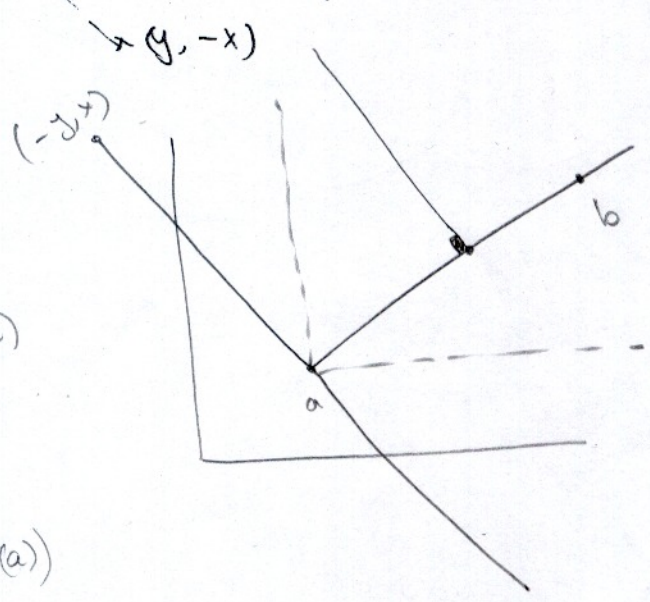



```
bc = (b+c)/2
rec(bc) = rec(b,c)
point perp(vec bc * y, vec bc * x)
    perp += bc
```

How to get perpendicular?
two perpendicular on vector(x,y)
(-y, x) & (y, -x)



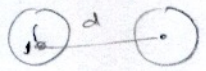
∵ $m_1 * m_2 = -1$
if line of slope m_1 is ⊥ to line of slope m_2



① shift to a
② get ⊥
③ shift back (not to a) (to c)

define normalize(a) (a)/length(a))

circle - circle intersection

#1  $r_1 + r_2 < d$

#2  $r_1 + r_2 = d$

$t = \frac{r_1}{r_1 + r_2}$ $p = a + t * (b - a)$ } intersection point

using parametric eqn

$t > 1$

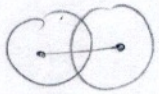
#3  $|r_1 - r_2| = d$ $r_1 > r_2$

#4  $r_1 - r_2 \geq d$

#5  $r_1 = r_2$ & $c_1 = c_2$: ∞ intersection points

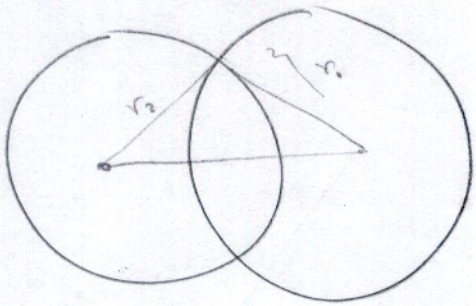
vector.size > 2

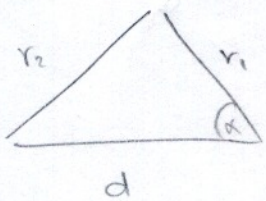
vector.size = 2

#6  $r_1 + r_2 \not\geq d$

#7  $r_{small} = 0$ && $d = r_1$

Case 6





normalize(rec(b,a)) * r1

alpha = cosRule(r2, r1, d);

res.pushback(rotate(x, alpha) + b);

res.push_back(rotate(x, -alpha) + b);