CMPUT 403: Geometry

Zachary Friggstad

February 16, 2016

Points

```
typedef complex<double> point; //#include <complex>
point p(1.0, 5.7);
p.real(); // x component
p.imag(); // y component
```

Helpful operations

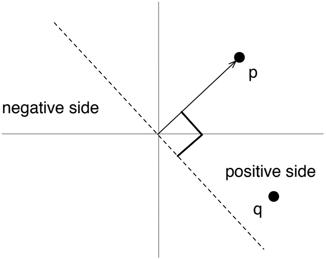
```
point p, q;
p+q; // exactly what you expect
p *= polar(1, theta); //rotate p by theta radians
abs(p-q); //distance between p and q
arg(p); //angle from positive x-axis, measured in (-pi, pi]
```

One Annoyance: we can't update the x and y components without creating a new point.

Dot Product

$p \circ q = p_x \cdot q_x + p_y \cdot q_y = ||p|| \cdot ||q|| \cdot \cos\theta$

```
double dot(point p, point q) { return real(p*conj(q)); }
```

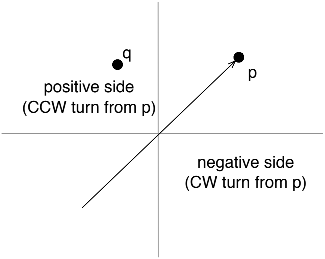


- $p \circ q = 0$ if $p \perp q$ (or one of them is 0)
- $p \circ p = ||p||^2$
- $p \circ q > c$: q lies in the *same* general direction as p
- $p \circ q < 0$: q lies in the *opposite* general direction as p

Cross Product

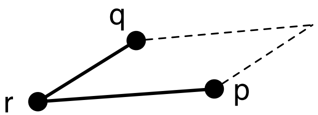
$$p \times q = p_x \cdot q_y - p_y \cdot q_x = ||p|| \cdot ||q|| \cdot \sin\theta$$

```
double cross(point p, point q) { return imag(p*conj(q)); }
```

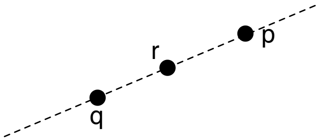


- $p \times q = 0$ if p and q are collinear with 0 (or one of them is 0)
- $p \times q > 0$: $\angle(p, q)$ is *counterclockwise*
- $p \times q < 0$: $\angle(p, q)$ is *clockwise*

$\frac{1}{2}|(p-r)\times(q-r)|$ is the area of the triangle with corners p, q, r.



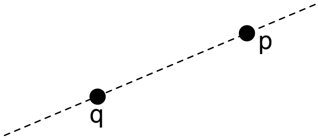
Tip: p lies on the line passing through points q and r **if and only if** $(q-p)\times(r-p)=0$.



Additionally: if p lies on this line then it lies between q and r **if and only** if $(q-p)\circ(r-p)\leq 0$.

Parametric representation of a line passing through p and q:

$$r(t) = (1-t) \cdot p + t \cdot q, \quad t \in \mathbb{R}$$



Note $r(0) = p$ and $r(1) = q$.

The line segment connecting p and q is parameterized this way, with the restriction $0 \leq t \leq 1$.

Given two lines passing through p, q and p', q' respectively, compute their intersection point (if any).

Solve $(1-t)\cdot p + t\cdot q = (1-t')p' + t'\cdot q'$ for $t, t' \in \mathbb{R}$.

$$\begin{bmatrix} q_x - p_x & p'_x - q'_x \\ q_y - p_y & p'_y - q'_y \end{bmatrix} \cdot \begin{bmatrix} t \\ t' \end{bmatrix} = \begin{bmatrix} p'_x - p_x \\ p'_y - p_y \end{bmatrix}$$

Use Cramer's rule: for $A \cdot x = b$ we have

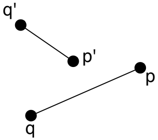
$$x_i = \frac{\det A_i}{\det A}$$

Where A_i is the matrix obtained by replacing column i of A with b.

Tip: $\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc$.

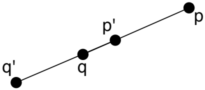
$\det A = 0$ means the lines are parallel.

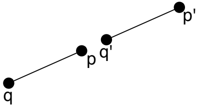
Computing the intersection of two line segments.



- If the infinite lines intersect (i.e. not parallel), get values t, t'.
- Ensure $0 \leq t \leq 1$ and $0 \leq t' \leq 1$.

If lines are colinear:

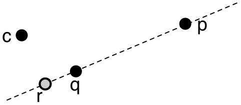




- If $p = p'$ and $q = q'$ or vice versa, they overlap.
- Else if a point is in the interior of the other segment, they overlap.
- Else if the segments share a point, they touch only at that point.
- Else they do not touch.

Closest point on a line.

Given a line passing through p, q and another point c, find the point r on the line $\overline{pq}$ nearest to c.



Minimize $||(1-t)p + t \cdot q - c||^2$ over $t \in \mathbb{R}$ (avoids the square root).

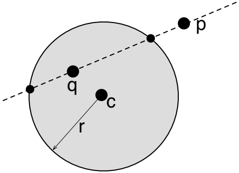
Expanding, this is just minimizing some quadratic:

$$a \cdot t^2 + b \cdot t + c$$

Tip: the minimum is at $-\frac{b}{2a}$. If $a = 0$, then $p = q$.

Line-Circle Intersection

A line $\overline{pq}$ and a circle (r, c) (radius/center point) does the line puncture the circle?



Solve $||(1-t) \cdot p + t \cdot q - c||^2 = r^2$ for t.

Rearrange: $a \cdot t^2 + b \cdot t + c = 0$. $a = 0$ means $p = q$.

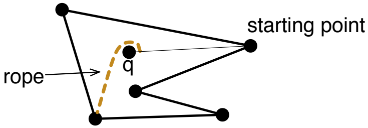
Tip: Use the quadratic formula

$$\frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{where} \quad \Delta = b^2 - 4ac.$$

$\Delta < 0$: line misses, $\Delta = 0$: line tangent, $\Delta > 0$: line punctures.

Point q in Polygon $P : p_1, \ldots, p_n$?

Intuition: walk about the polygon with a rope taut with “post” q.

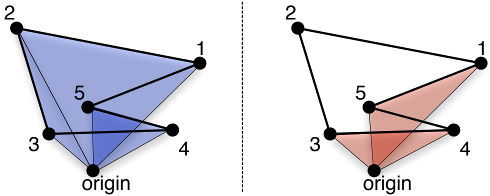


If q is inside, the rope wraps around once. If q is outside, the rope doesn't wrap around.

```
double delta = 0;
for (int i = 0, j = n-1; i < n; j = i++)
   delta += arg((p[i]-q) / (p[j]-q)); //change in angle
return fabs(delta) > 1; //|delta| be 0 or 2*pi
```

Have to add a bit more to check if q lies on the boundary.

Area of a Polygon $P : p_1, \ldots, p_n$



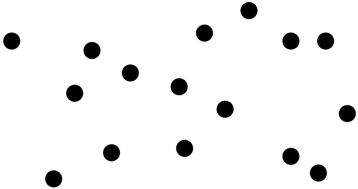
blue triangle: count the area **positively** (CCW turn about origin)
red triangle: count the area **negatively** (CW turn about origin)

```
double area = 0;
for (int i = 0, j = n-1; i < n; j = i++)
   //signed area of triangle [origin, p[i-1], p[i]]
   area += cross(p[i], p[j])*0.5;
return fabs(area);
```

Points outside are "cancelled", points inside are counted +1 times (after cancelling). The net sum is the area of P.

Convex Hull

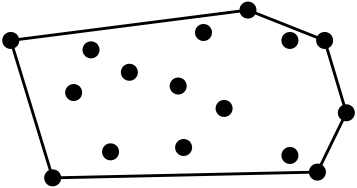
Given points $p_1, \ldots, p_n$, what is their *convex hull*?



Output: The points on the convex hull in CCW order.

Convex Hull

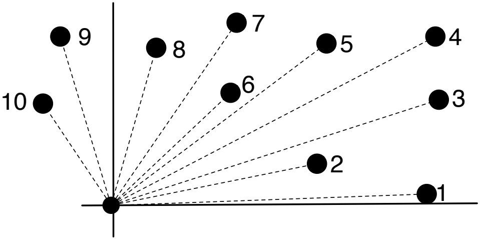
Given points $p_1, \ldots, p_n$, what is their *convex hull*?



Output: The points on the convex hull in CCW order.

Convex Hull

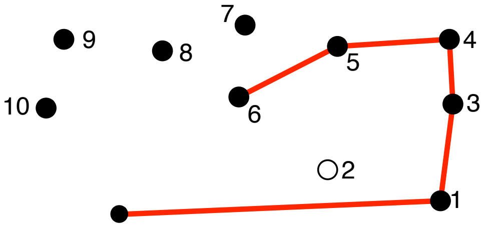
The bottom-most point must be on the convex hull. If there are many, choose the leftmost. Suppose this point is $(0, 0)$ by shifting all points if needed.



Sort all other points by their angle with the positive x-axis. To check if $p_i < p_j$ according to this order, just check $p_i \times p_j > 0$.

The origin and the first point after sorting are on the hull for sure.

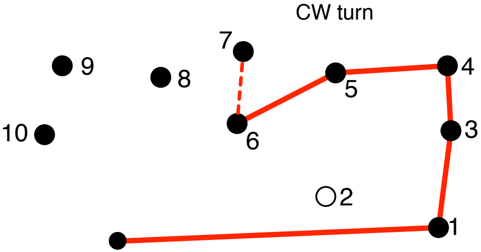
Build up the hull one point at a time: add the points in the sorted order.



If one creates a clockwise turn with the previous two on the hull, then pop the end of the current hull.

The origin and the first point after sorting are on the hull for sure.

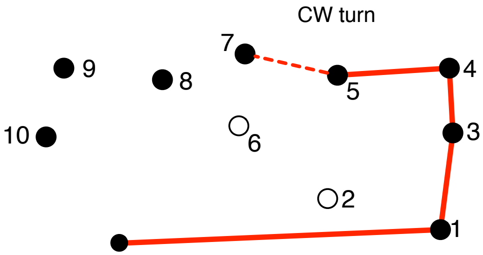
Build up the hull one point at a time: add the points in the sorted order.



If one creates a clockwise turn with the previous two on the hull, then pop the end of the current hull.

The origin and the first point after sorting are on the hull for sure.

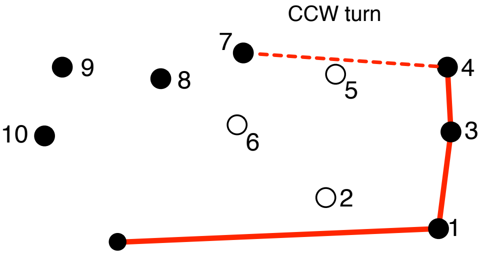
Build up the hull one point at a time: add the points in the sorted order.



If one creates a clockwise turn with the previous two on the hull, then pop the end of the current hull.

The origin and the first point after sorting are on the hull for sure.

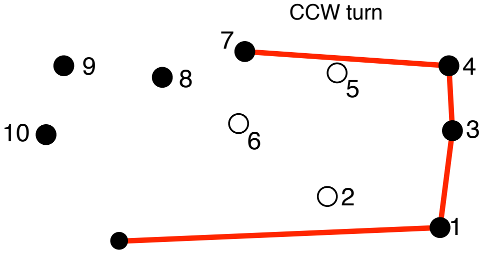
Build up the hull one point at a time: add the points in the sorted order.



If one creates a clockwise turn with the previous two on the hull, then pop the end of the current hull.

The origin and the first point after sorting are on the hull for sure.

Build up the hull one point at a time: add the points in the sorted order.



If one creates a clockwise turn with the previous two on the hull, then pop the end of the current hull.

```
bool cmp(point a, point b) { return cross(a, b) > 0; }

point p[MAXN], hull[MAXN];
//Assume p[n-1] is the lowest point (breaking ties by
//taking leftmost) and no 3 points are collinear

for (int i = 0; i < n-1; ++i) p[i] -= p[n-1];
sort(p, p+n-1, cmp); //sort by angle from x-axis

hull[0] = point(0,0);
hull[1] = p[0];
int hs = 2;

for (int i = 1; i < n-1; ++i) {
  while (cross(hull[hs-1]-hull[hs-2],p[i]-hull[hs-2]) < 0)
     --hs;
  hull[hs++] = p[i];
}
for (int i = 0; i < hs; ++i) hull[i] += p[n-1];
```

Running time: $O(n \log n)$

Tips

Never use `float`, not enough precision. Use `double`.

Sometimes values that should be equal aren't quite due to floating point errors.

Use the following for safer comparisons.

```
#define EPS 1e-8
double a, b;
if (fabs(a-b) < EPS) ...   //check if a == b
if (a + EPS < b) ...       //check if a < b
if (a < b + EPS) ...       //check if a <= b
```

Can there be 3 collinear points? Can points be equal? How can you deal with this?

Missing Topics

Voronoi diagrams/Delaunay triangulations, 3D convex hull, closest pair of points, furthest pair of points, triangulating a polygon.

Next Time

Number Theory

To Come

- Combinatorics and Arithmetic
- String Processing
- Matchings and Network Flow

Possible Misc. Topics

Probability, counting MSTs, edge colourings, voronoi diagrams, simplex for linear programming, Nim + extensions, matroids.