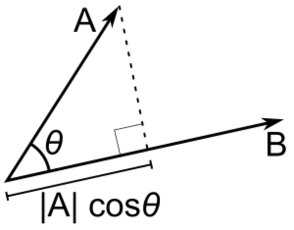
Geometry

Dot product



$$a \cdot b = |a|\ |b| \cos(\theta)$$

$$a \cdot b = a_x b_x + a_y b_y$$

$$\theta = \arccos\left(\frac{a_x b_x + a_y b_y}{|a|\ |b|}\right)$$

Projection of a onto b: $\frac{a \cdot b}{|b|}$

Cross product

$$a \times b = |a|\ |b| \sin(\theta) = a_x b_y - a_y b_x$$

θ is positive if a is clockwise from b

Line-point distance

Line given by $ax + by + c = 0$ and point (x_0, y_0).

$$\text{distance} = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

The coordinates of this point are:

$$x = \frac{b(bx_0 - ay_0) - ac}{a^2 + b^2} \quad y = \frac{a(-bx_0 + ay_0) - bc}{a^2 + b^2}$$

Shoelace formula

$$2A = \sum_{i=1}^{n} \begin{vmatrix} x_i & y_i \\ x_{i+1} & y_{i+1} \end{vmatrix}, \text{where} \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

Circumradius

Let a, b, c be the sides of a triangle and A the area of the triangle. Then the circumradius $R = abc/(4A)$. Alternatively, using the Law of Sines:

$$R = \frac{a}{2\sin(\alpha)} = \frac{b}{2\sin(\beta)} = \frac{c}{2\sin(\gamma)}$$

where α, β, and γ are the angles opposite sides a, b, and c respectively.

Law of Sines

In any triangle with sides a, b, c and opposite angles α, β, γ respectively:

$$\frac{a}{\sin(\alpha)} = \frac{b}{\sin(\beta)} = \frac{c}{\sin(\gamma)} = 2R$$

where R is the circumradius of the triangle. This can be rearranged to find any side or angle: $a = 2R\sin(\alpha)$ and $\sin(\alpha) = \frac{a}{2R}$

Law of Cosines

In any triangle with sides a, b, c and opposite angles α, β, γ respectively:

$$c^2 = a^2 + b^2 - 2ab\cos(\gamma)$$

Median Length Formulas

In any triangle with sides a, b, c, the lengths of the medians m_a, m_b, m_c from the respective vertices are given by:

$$m_a = \frac{1}{2}\sqrt{2b^2 + 2c^2 - a^2}$$

These formulas can be derived using the Apollonius's theorem.

Segment to line linear equation

Converting segment $((P_x, P_y), (Q_x, Q_y))$ to $Ax + By + C = 0$:

$$(P_y - Q_y)x + (Q_x - P_x)y + (P_xQ_y - P_yQ_x) = 0$$

Three point orientation

```
int orientation(Point p1, Point p2, Point p3){
    int val = (p2.y-p1.y)*(p3.x-p2.x)-(p2.x-p1.x)*(p3.y-p2.y);
    if (val == 0) return 0; // collinear
    return (val > 0) ? 1 : 2; // clock or counterclock
}
```

Line-line intersection

From system of linear equations derived Cramer's rule:

$$\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases} \Rightarrow \begin{cases} x = (c_1b_2 - c_2b_1)/(a_1b_2 - a_2b_1) \\ y = (a_1c_2 - a_2c_1)/(a_1b_2 - a_2b_1) \end{cases}$$

If the denominator equals zero, the lines are parallel or coincident.

Check if two segments intersect

```
bool on_segment(Point p, Point q, Point r) {
    return (q.x <= max(p.x, r.x) && q.x >= min(p.x, r.x) &&
        q.y <= max(p.y, r.y) && q.y >= min(p.y, r.y))
}

bool do_intersect(Point p1, Point q1, Point p2, Point q2){
    int o1 = orientation(p1, q1, p2);
    int o2 = orientation(p1, q1, q2);
    int o3 = orientation(p2, q2, p1);
    int o4 = orientation(p2, q2, q1);
    if (o1 != o2 && o3 != o4) return true;
    if (o1 == 0 && on_segment(p1, p2, q1)) return true;
    if (o2 == 0 && on_segment(p1, q2, q1)) return true;
    if (o3 == 0 && on_segment(p2, p1, q2)) return true;
    if (o4 == 0 && on_segment(p2, q1, q2)) return true;
    return false;
}
```

Heron’s formula

Let a, b, c - sides of a triangle. Then the area A is:

$$A = \frac{1}{4}\sqrt{(a+b+c)(-a+b+c)(a-b+c)(a+b-c)}$$

Numerically stable version:

$$a \geq b \geq c, A = \frac{1}{4}\sqrt{(a+(b+c))(c-(a-b))(c+(a-b))(a+(b-c))}$$

Graham’s scan

```
struct pt {double x, y;};
int orientation(pt a, pt b, pt c) {
    double v = a.x*(b.y-c.y)+b.x*(c.y-a.y)+c.x*(a.y-b.y);
    if (v < 0) return -1; // clockwise
    if (v > 0) return +1; // counter-clockwise
    return 0;
}

bool cw(pt a, pt b, pt c, bool include_collinear) {
    int o = orientation(a, b, c);
    return o < 0 || (include_collinear && o == 0);
}
bool collinear(pt a, pt b, pt c) { return orientation(a, b, c) == 0; }

void convex_hull(vector<pt>& a, bool include_collinear = false) {
    pt p0 = *min_element(a.begin(), a.end(), [](pt a, pt b) {
        return make_pair(a.y, a.x) < make_pair(b.y, b.x);
    });
    sort(a.begin(), a.end(), [&p0](const pt& a, const pt& b) {
        int o = orientation(p0, a, b);
        if (o == 0)
            return (p0.x-a.x)*(p0.x-a.x) + (p0.y-a.y)*(p0.y-a.y)
                < (p0.x-b.x)*(p0.x-b.x) + (p0.y-b.y)*(p0.y-b.y);
        return o < 0;
    });
    if (include_collinear) {
        int i = (int)a.size()-1;
        while (i >= 0 && collinear(p0, a[i], a.back())) i--;
        reverse(a.begin()+i+1, a.end());
    }

    vector<pt> st;
    for (int i = 0; i < (int)a.size(); i++) {
        while (st.size() > 1 && !cw(st[st.size()-2], st.back(), a[i],
include_collinear))
            st.pop_back();
        st.push_back(a[i]);
    }

    a = st;
}
```

Circumradius

Let a, b, c - sides of a triangle. A - area of the triangle. Then the circumradius is:

$$R = \frac{abc}{4A}$$

Closest pair of points

```
double mindist;
pair<int, int> best_pair;

void upd_ans(const pt & a, const pt & b) {
    double dist = sqrt((a.x - b.x)*(a.x - b.x) + (a.y - b.y)*(a.y - b.y));
    if (dist < mindist) {
        mindist = dist;
        best_pair = {a.id, b.id};
    }
}
vector<pt> t;

void rec(int l, int r) {
    if (r - l <= 3) {
        for (int i = l; i < r; ++i) {
            for (int j = i + 1; j < r; ++j) {
                upd_ans(a[i], a[j]);
            }
        }
        sort(a.begin() + l, a.begin() + r, cmp_y());
        return;
    }

    int m = (l + r) >> 1;
    int midx = a[m].x;
    rec(l, m);
    rec(m, r);

    merge(a.begin() + l, a.begin() + m, a.begin() + m, a.begin() + r, t.begin(),
cmp_y());
    copy(t.begin(), t.begin() + r - l, a.begin() + l);

    int tsz = 0;
    for (int i = l; i < r; ++i) {
        if (abs(a[i].x - midx) < mindist) {
            for (int j = tsz - 1; j >= 0 && a[i].y - t[j].y < mindist; --j)
                upd_ans(a[i], t[j]);
            t[tsz++] = a[i];
        }
    }
}
```