计算几何

一、点

1. 实数比较

```
typedef long double db;
const db EPS = 1e-9;
inline int sign(db a) { return a < -EPS ? -1 : a > EPS; }
inline int cmp(db a, db b) { return sign(a - b); }
```

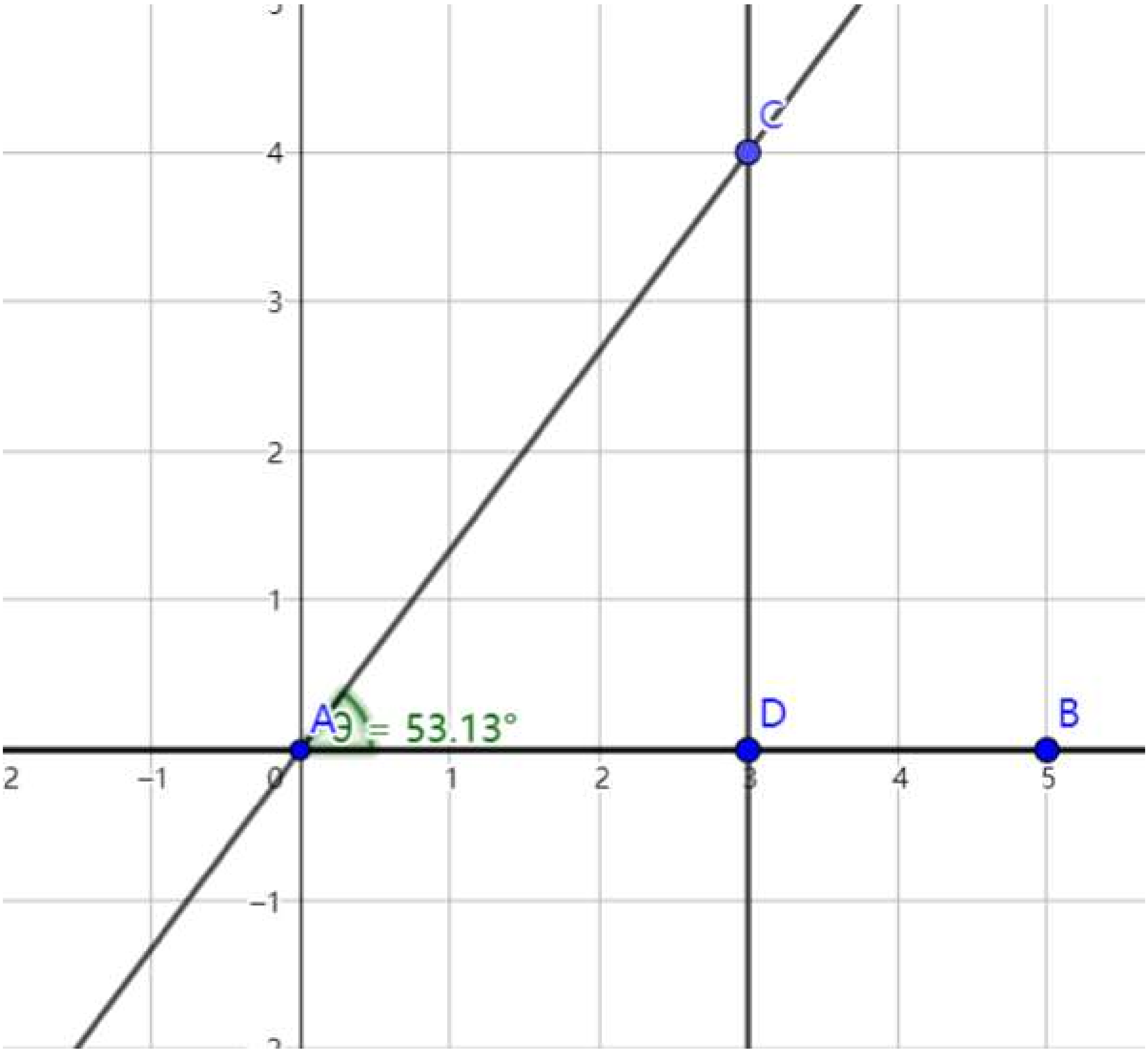
2. 点结构体

```
struct P
{
    db x, y;
    P() {}
    P(db _x, db _y) : x(_x), y(_y) {}
    P operator+(P p) { return {x + p.x, y + p.y}; }
    P operator-(P p) { return {x - p.x, y - p.y}; }
    P operator*(db d) { return {x * d, y * d}; }
    P operator/(db d) { return {x / d, y / d}; }
    bool operator<(P p) const
    {
        int c = cmp(x, p.x);
        if (c)
            return c == -1;
        return cmp(y, p.y) == -1;
    }
    bool operator==(P o) const
    {
        return cmp(x, o.x) == 0 && cmp(y, o.y) == 0;
    }
    db abs() { return sqrt(abs2()); }
    db abs2() { return x * x + y * y; }
    void read() { cin >> x >> y; }
    void write() { cout << "(" << x << "," << y << ")" << endl; }
    P rot90() { return P(-y, x); }
    P unit() { return *this / abs(); }
    db distTo(P p) { return (*this - p).abs(); }
    int quad() const { return sign(y) == 1 || (sign(y) == 0 && sign(x) >= 0); } //判断在上半区还是下半区
    db alpha() const { return atan2(y, x); }
};
```

3. 点乘 叉乘

```
db dot(const P &p, const P &q) { return p.x * q.x + p.y * q.y; }//点积
db det(const P &p, const P &q) { return p.x * q.y - p.y * q.x; }//叉积
```

点积，叉积的作用



1. 点积几何意义：

$$\vec{A}B * \vec{A}C = |AB| \cdot |AC| \cdot \cos\theta \tag{1}$$

$$\begin{array}{c|ccc} n & \text{dot>0} & \text{dot=0} & \text{dot<0} \\ \hline \cos\theta & >0 & <0 & =0 \\ \text{角度} & \text{锐角} & \text{直角} & \text{钝角} \end{array} \tag{2}$$

若以$\vec{A}B$为x轴，其垂线为y轴。若dot>0则C在`第一四象限`，dot<0则C在`二三象限`

2. 叉积几何意义：

几何意义:$|AC| * |BD|$,即三角形ABC面积的两倍

$$\vec{A}B * \vec{A}C = |AB| \cdot |AC| \cdot \sin\theta \tag{3}$$

若以$\vec{A}B$为x轴，其垂线为y轴。若det>0则C在`第一二象限`，dot<0则C在`三四象限`

4. 判断点半区

```
int quad(const P &p) { return sign(p.y) == 1 || (sign(p.y) == 0 && sign(p.x) >= 0); } //判断在上半区还是下半区
```

5. 极角排序

```
bool PolarAngleSorting(const P &p, const P &q)
{
    if (quad(p) == quad(q))
        return sign(det(p, q)) == -1;
    else
        return quad(p) < quad(q);
}
```

6. 点的旋转

```
P rot(P p, db theta) { return {p.x * cos(theta) - p.y * sin(theta), p.x * sin(theta) + p.y * cos(theta)}; }
```

二、线

1. 点的位置判断

```
#define cross(p1, p2, p3) ((p2.x - p1.x) * (p3.y - p1.y) - (p3.x - p1.x) * (p2.y - p1.y))
#define crossOp(p1, p2, p3) sign(cross(p1, p2, p3))//以p1为原点，如果p3在p2的逆时针则返回1，共线返回0，顺时针返回-1
```

2. 判断平行

```
bool chkLL(P p1, P p2, P q1, P q2)//平行为0，不平行为1
{ //判断平行，注：重合也算平行
    //若删去重合只需判断四点是否共线。
    db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
    return sign(a1 + a2) != 0;
}
```

3. 直线交点

```
P isLL(P p1, P p2, P q1, P q2)
{
    db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
    return (p1 * a2 + p2 * a1) / (a1 + a2);
}
```

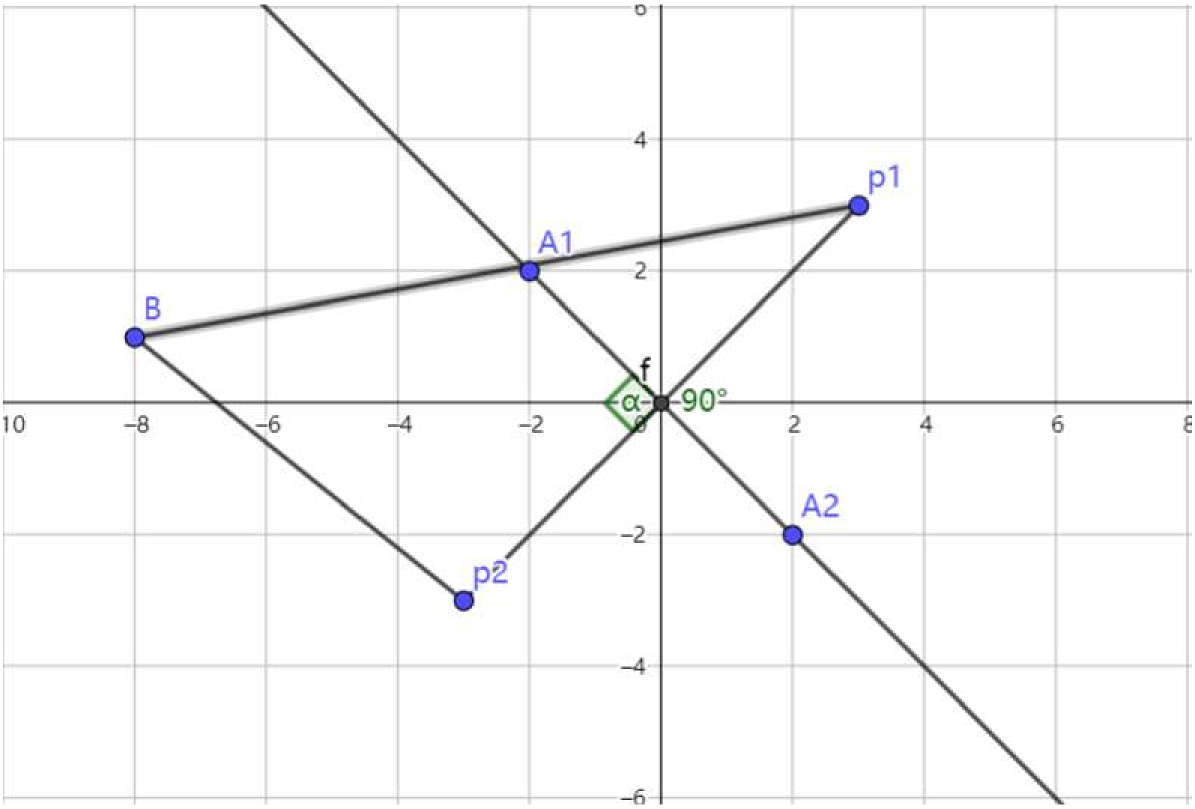
4. 线段相交

```
bool intersect(db l1, db r1, db l2, db r2)
{
    if (l1 > r1)
        swap(l1, r1);
    if (l2 > r2)
        swap(l2, r2);
    return !(cmp(r1, l2) == -1 || cmp(r2, l1) == -1);
}
bool isSS(P p1, P p2, P q1, P q2)//判断线段相交
{
    return intersect(p1.x, p2.x, q1.x, q2.x) && intersect(p1.y, p2.y, q1.y,
q2.y) &&
           crossOp(p1, p2, q1) * crossOp(p1, p2, q2) <= 0 && crossOp(q1, q2, p1)
* crossOp(q1, q2, p2) <= 0;
}
bool isSS_strict(P p1, P p2, P q1, P q2)//严格相交
{
    return crossOp(p1, p2, q1) * crossOp(p1, p2, q2) < 0 && crossOp(q1, q2, p1)
* crossOp(q1, q2, p2) < 0;
}
```

5. 点是否在直线上

```
bool isMiddle(db a, db m, db b)
{
    return sign(a - m) == 0 || sign(b - m) == 0 || (a < m != b < m);
}
bool isMiddle(P a, P m, P b)
{
    return isMiddle(a.x, m.x, b.x) && isMiddle(a.y, m.y, b.y);
}
bool onSeg(P p1, P p2, P q)
{
    return crossOp(p1, p2, q) == 0 && isMiddle(p1, q, p2);
}
bool onSeg_strict(P p1, P p2, P q)
{
    return crossOp(p1, p2, q) == 0 && sign(dot((q - p1), (p1 - p2)) *
sign(dot((q - p2), (p1 - p2))));
}
```

6. 投影 反射 最近点



- 投影：如图中A1对p1p2的投影为O
- 反射：如图中A1对p1p2的反射为A2
- 最近点：如图中A1对p1p2的最近点为O，B对p1p2的最近点为p2

```
P proj(P p1, P p2, P q)
{
    P dir = p2 - p1;
    return p1 + dir * (dot(dir, (q - p1)) / dir.abs2());
}
P reflect(P p1, P p2, P q)
{
    return proj(p1, p2, q) * 2 - q;
}
db nearest(P p1, P p2, P q)
{
    P h = proj(p1, p2, q);
    if (isMiddle(p1, h, p2))
        return q.distTo(h);
    return min(p1.distTo(q), p2.distTo(q));
}
```

7. 线段间距离

```
db disSS(P p1, P p2, P q1, P q2)
{
    if (isSS(p1, p2, q1, q2))
        return 0;
    return min(min(nearest(p1, p2, q1), nearest(p1, p2, q2)),
               min(nearest(q1, q2, p1), nearest(q1, q2, p2)));
}
```

8. 线段夹角

```
db rad(P p1, P p2)
{
    return atan2l(det(p1, p2), dot(p1, p2));
}
```

三、 多边形

1. 多边形面积

```
db area(vector<P> ps)
{
    db ret = 0;
    for (int i = 0; i < ps.size(); i++)
        ret += det(ps[i], ps[(i + 1) % ps.size()]);
    return ret / 2;
}
```

2. 点包含

```
int contain(vector<P> ps, P p) // inside-2;onSeg-1;outside-0;
{
    int n = ps.size(), ret = 0;
    for (int i = 0; i < ps.size(); i++)
    {
        P u = ps[i], v = ps[(i + 1) % n];
        if (onSeg(u, v, p))
            return 1;
        if (cmp(u.y, v.y) <= 0)
            swap(u, v);
        if (cmp(p.y, u.y) > 0 || cmp(p.y, v.y) <= 0)
            continue;
        ret ^= crossOp(p, u, v) > 0;
    }
    return ret * 2;
}
```

3. 凸包

```
vector<P> convexHull(vector<P> ps)
{
    int n = ps.size();
    if (n <= 1)
        return ps;
    sort(ps.begin(), ps.end());
    vector<P> qs(n * 2);
    int k = 0;
    for (int i = 0; i < n; qs[k++] = ps[i++])
        while (k > 1 && crossOp(qs[k - 2], qs[k - 1], ps[i]) <= 0)
            --k;
    for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
        while (k > t && crossOp(qs[k - 2], qs[k - 1], ps[i]) <= 0)
```

```
            --k;
    qs.resize(k - 1);
    return qs;
}
```

4. 点集直径

```
db convexDiameter(vector<P> ps)
{
    int n = ps.size();
    if (n <= 1)
        return 0;
    int is = 0, js = 0;
    for (int k = 1; k < n; k++)
        is = ps[k] < ps[is] ? k : is, js = ps[js] < ps[k] ? k : js;
    int i = is, j = js;
    db ret = ps[i].distTo(ps[j]);
    do
    {
        if (det((ps[(i + 1) % n] - ps[i]), ps[(j + 1) % n] - ps[j]) >= 0)
            (++j) %= n;
        else
            (++i) %= n;
        ret = max(ret, ps[i].distTo(ps[j]));
    } while (i != is || j != js);
    return ret;
}
```

齐齐哈尔大学

汪家斌