计算几何基础

邵俊儒

点、向量

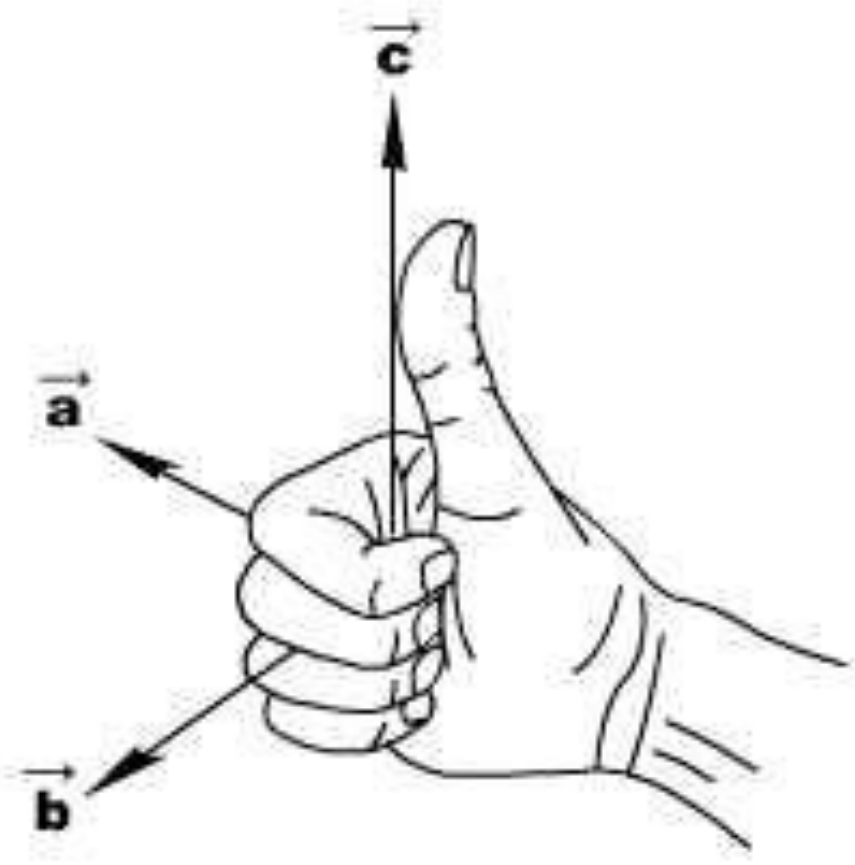
```
class Point {  // also represents a vector
        double x, y;
        …
};
```

向量的点积

- $(x_1, y_1) \cdot (x_2, y_2) = x_1 y_1 + x_2 y_2$
- $v_1 \cdot v_2 = |v_1||v_2| \cos\langle v_1, v_2 \rangle$

向量的叉积

- $(x_1, y_1) \times (x_2, y_2) = x_1 y_2 - x_2 y_1$
- $v_1 \cdot v_2 = |v_1||v_2| \sin\langle v_1, v_2 \rangle$
- 四边形的有向面积
- 正负性：右手定则



直线 / 线段

```
struct Line {
	double a, b, c;
};

struct Line {
	Point a, b;
};
```

点与线

- 点 P 在直线 AB 上
  - $\overrightarrow{AP} \times \overrightarrow{BP} = 0$
- 点 P 在线段 AB 上
  - $\overrightarrow{AP} \times \overrightarrow{BP} = 0$
  - $\overrightarrow{AP} \cdot \overrightarrow{BP} \le 0$

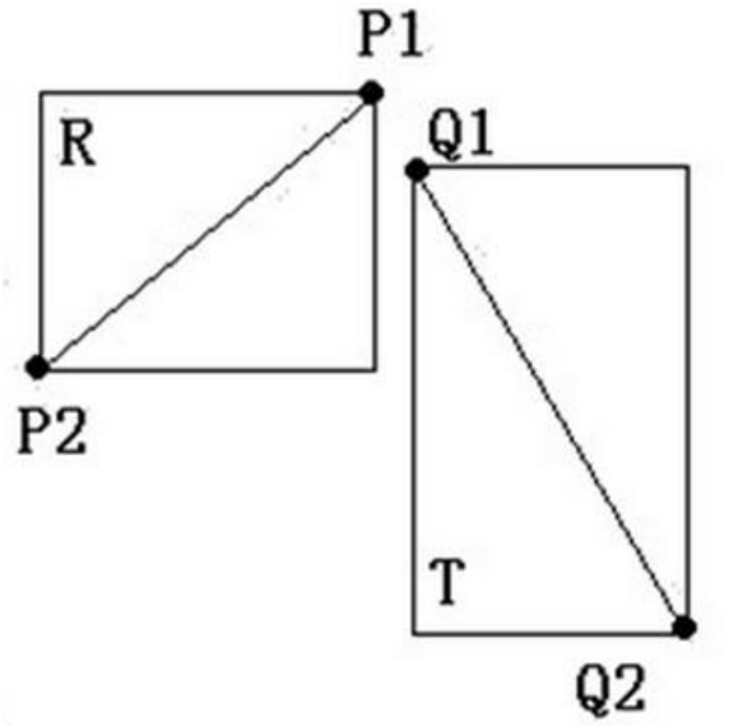
Hint: 浮点数比较大小

```
int sign(double x, double eps = EPS) {
    if (x > eps)
        return 1;
    if (x < -eps)
        return -1;
    return 0;
}
```

线段判交

Step 1: 快速排斥

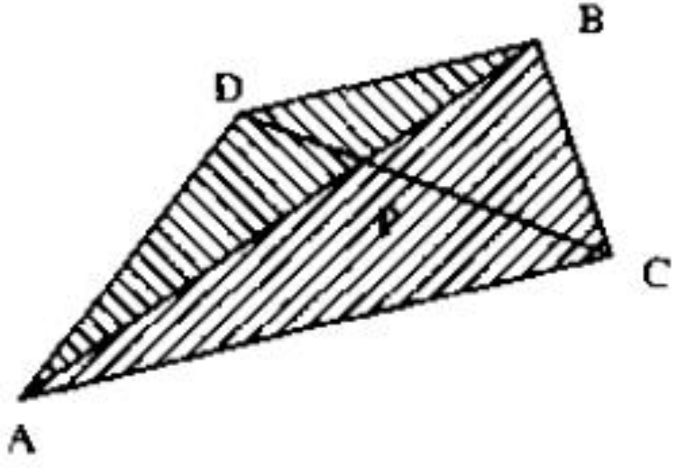
- 线段在任意直线投影不交
  - 必然无交点
- 分别选取两个坐标轴



线段判交

Step 2: 跨立实验

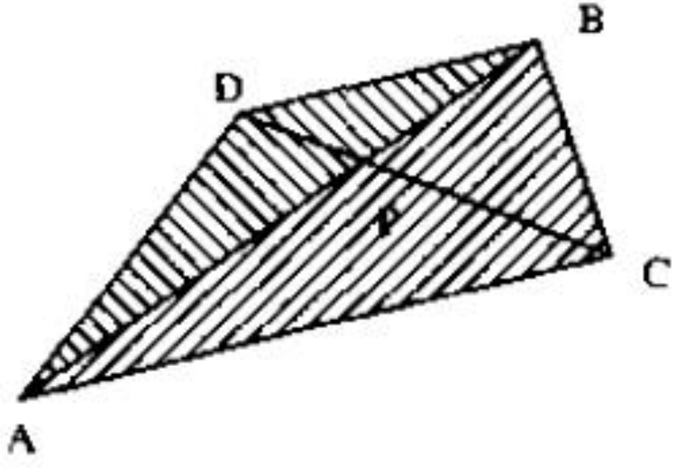
- 点 A 与点 B 必须在直线 CD 两侧
- 点 C 与点 D 必须在直线 AB 两侧



线段判交

Step 3: 求交点

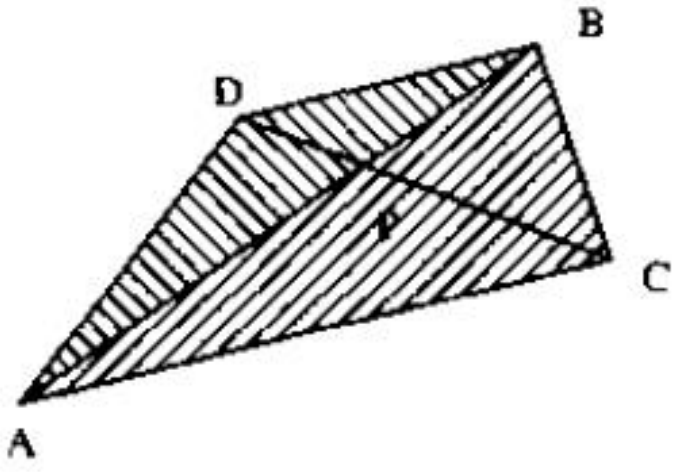
- 定比分点
- $x_p = \frac{S_{\Delta ABD} \cdot x_C + S_{\Delta ABC} \cdot x_D}{S_{\Delta ABD} + S_{\Delta ABC}}$



线段判交

Alternate

- 直线求交得到 P
- 判定 P 是否在线段 AB 和 CD 上

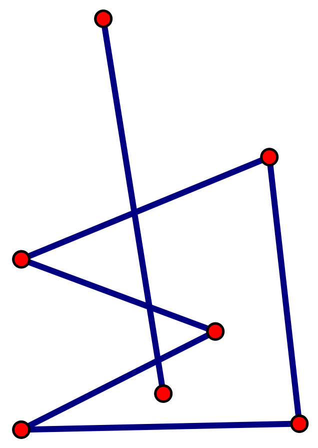


线段判交

```
int segIntersect(const Point &a, const Point &b, const Point &c, const Point &d, Point &o) {
    double s1, s2, s3, s4; int d1, d2, d3, d4, iCnt = 0;
    d1 = sign(s1 = det(b - a, c - a)); d2 = sign(s2 = det(b - a, d - a));
    d3 = sign(s3 = det(d - c, a - c)); d4 = sign(s4 = det(d - c, b - c));
    if ((d1 ^ d2) == -2 && (d3 ^ d4) == -2) {
        o = (c * s2 - d * s1) / (s2 - s1); return true;
    }
    if (d1 == 0 && c.onSeg(a, b)) o = c, ++iCnt;
    if (d2 == 0 && d.onSeg(a, b)) o = d, ++iCnt;
    if (d3 == 0 && a.onSeg(c, d)) o = a, ++iCnt;
    if (d4 == 0 && b.onSeg(c, d)) o = b, ++iCnt;
    return iCnt ? 2 : 0; // 不相交返回0, 严格相交返回1, 非严格相交返回2
}
```

判定点在多边形内: 射线法

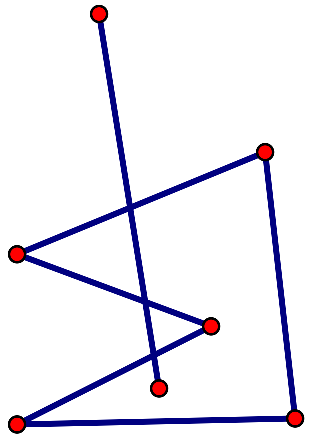
- 从该点往某个方向画射线
  - 与边界交奇数次
    - 在多边形中
  - 与边界交偶数次
    - 不在多边形中
- 特殊情况
  - 射线与多边形某条边重合
  - 射线经过了某点多次



判定点在多边形内: 射线法

方向的选取 I

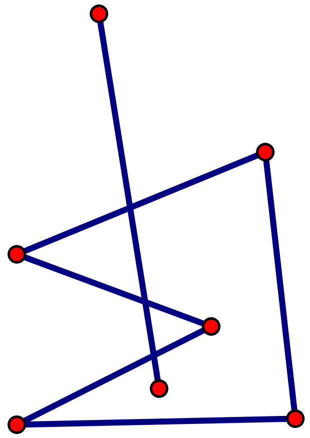
- 随机选取



判定点在多边形内: 射线法

方向的选取 II

- 避免随机
- 选择斜率为 0 的直线
- 如何避免特殊情况？
  - 把点 P 上移无穷小



判定点在多边形内: 射线法

```
bool contain(Point ps[], int n, const point &p) {
    Point A, B;
    int res = 0;
    for (int i = 0; i < n; ++i) {
        Point A = ps[i], B = ps[(i + 1) % n];
        if (p.onSeg(A, B)) return 1;
        if (sign(A.y - B.y) <= 0) swap(A, B);
        if (sign(p.y - A.y) > 0) continue;
        if (sign(p.y - B.y) <= 0) continue;
        res += (int)(sign(det(B - p, A - p)) > 0);
    } return res & 1;
}
```

多边形面积

- 凸多边形？
  - 剖分成三角形
  - 求每个三角形面积
- 一般情形？
  - 结论：做法依然成立

凸包

- 给定平面上 n 个不重合的点构成的点集 S
- 求面积最小的凸多边形，包含这 n 个点

- Observation
  - 这个多边形的每个顶点，必然是 n 个点中的某些点

凸包

- 考虑只有下半凸壳的情况
- 把点集按照 x 坐标为第一关键字，y 坐标为第二关键字升序排序

```
Stack ← empty
for each P in S
    while Stack 非空 and (Stack[-2], Stack[-1], P) 不下凸
        Stack.pop()
    Stack.push(P)
```

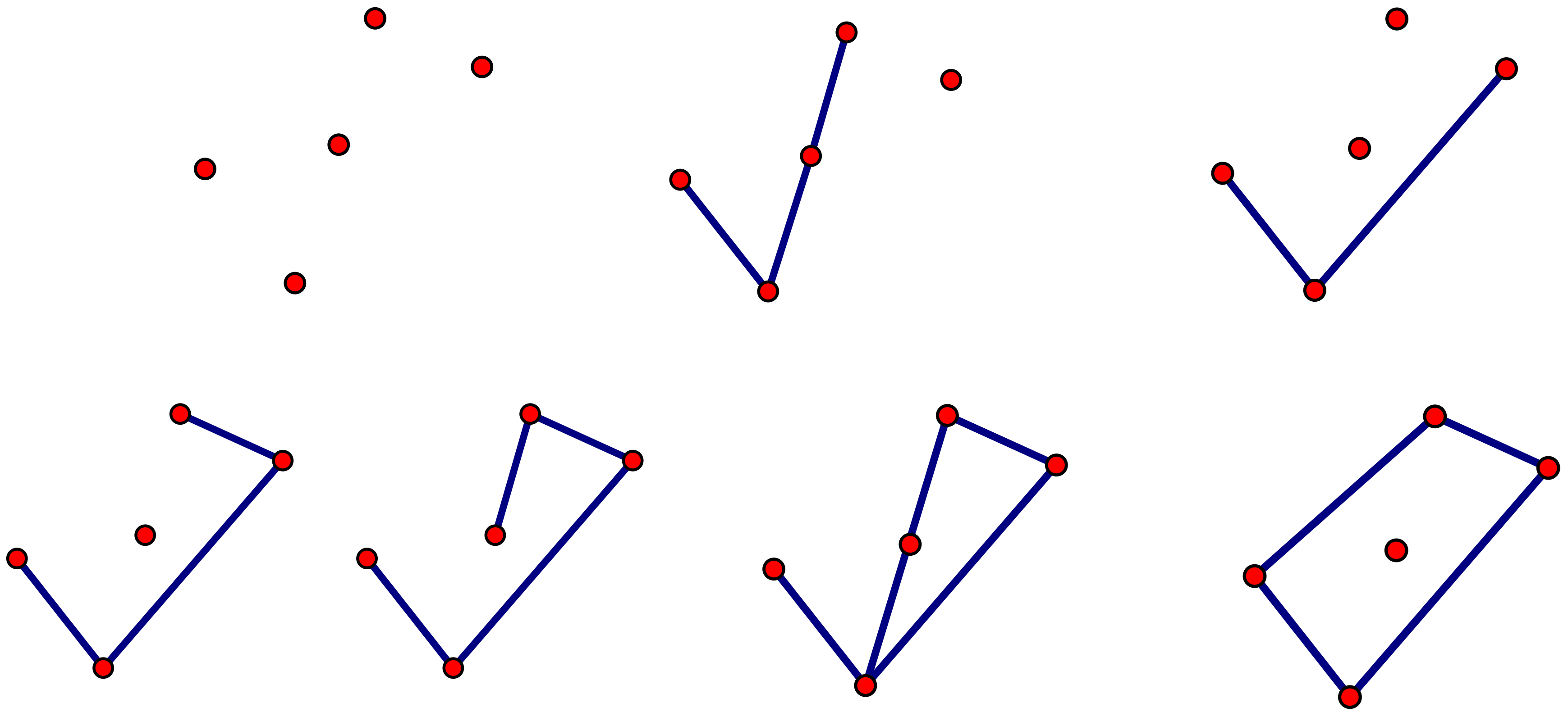
凸包

- 继续维护上半凸壳

凸包

```
vector<point> convexHull(int n, point ps[]) {
    static point qs[MAXN * 2];
    sort(ps, ps + n, cmpByXY);
    if (n <= 2)
        return vector(ps, ps + n);
    int k = 0;
    for (int i = 0; i < n; qs[k++] = ps[i++])
        while (k > 1 && det(qs[k - 1] - qs[k - 2], ps[i] - qs[k - 1]) < EPS)
            --k;
    for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
        while (k > t && det(qs[k - 1] - qs[k - 2], ps[i] - qs[k - 1]) < EPS)
            --k;
    return vector<point>(qs, qs + k);
}
```

凸包



复化 Simpson 公式

- 求函数 f 在区间 $[a, b]$ 上与 x 轴围成的面积
- 把区间切成小块 $[a_i, b_i]$
- 每段 2 次曲线拟合
- 一个 practical 的做法
  - 求出 $f(a_i)$、$f(b_i)$、$f\left(\frac{a_i+b_i}{2}\right)$
  - 认为 $S_i = \frac{f(a_i)+4f\left(\frac{a_i+b_i}{2}\right)+f(b_i)}{6}$

自适应 Simpson 法

```
simpson(a, b):
    double mid = (a + b) / 2.0
    if |simpson(a, mid) + simpson(mid, b)-simpson(a,b))| < threshold
        return simpson(a,b)
    else
        return simpson(a, mid) + simpson(mid, b)
```

补充作业

- SGU 253
- POJ 1151
- POJ 1269
- POJ 1696
- POJ 2318
- POJ 3348
- NOI2005 月下柠檬树