寒假的计算几何

LJZ_C

成都外国语学校

February 22th, 2020

点与矢量

我们在计算几何中一般用矢量的坐标表示，即记为 $a = (x, y)$

矢量的基本运算

```
1  inline Point operator + (const Point & a, const Point & b)
2  {
3      return Point(a.x + b.x, a.y + b.y);
4  }
5  inline Point operator - (const Point & a, const Point & b)
6  {
7      return Point(a.x - b.x, a.y - b.y);
8  }
9  inline Point operator * (const Point & a, const double & b)
10 {
11     return Point(a.x * b, a.y * b);
12 }
13 inline Point operator / (const Point & a, const double & b)
14 {
15     return Point(a.x / b, a.y / b);
16 }
```

叉积和点积

设 θ 为 $\vec{a}$ 和 $\vec{b}$ 之间的夹角, 那么
叉积 $det(a,b) = |\vec{a}| \cdot |\vec{b}| \cdot sin\theta = a.x \cdot b.y - a.y \cdot b.x$
点积 $dot(a,b) = |\vec{a}| \cdot |\vec{b}| \cdot cos\theta = a.x \cdot b.x + a.y \cdot b.y$
它们满足 $det(a,b) = -det(b,a)$, $dot(a,b) = dot(b,a)$

叉积的几何意义

叉积的几何意义为 $\vec{a}, \vec{b}$ 之间的有向面积
当 $\vec{b}$ 在 $\vec{a}$ 的逆时针方向时，$det(a, b) = 1$
当 $\vec{b}$ 在 $\vec{a}$ 的顺时针方向时，$det(a, b) = -1$

旋转矩阵

将 $\vec{v}(x,y)$ 逆时针旋转 θ 相当于

$$\begin{bmatrix} x \\ y \end{bmatrix} \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

得到 $\vec{v'} = (x \cdot cos\theta - y \cdot sin\theta, x \cdot sin\theta + y \cdot cos\theta)$

多边形的面积

对于多边形的每一条边，我们按照相同的方向 (顺时针或逆时针)，将它们和原点的有向面积加起来最后取绝对值就得到了多边形的面积. 同时这个拆分的思想可以用在解决与多边形有关的问题上.

例 1-圆与多边形的交

给出一个圆和一个 n 个点的多边形，求它们交的面积，要求复杂度 $O(n)$.

例 1-圆与多边形的交

给出一个圆和一个 n 个点的多边形，求它们交的面积，要求复杂度 $O(n)$.
将多边形拆分为 n 条边分别与原点构成的三角形，将每个三角形和圆的交的有向面积累加起来即可. 分情况讨论.

坐标轴变换

以 A 为原点 AB 为 x 轴的单位长度求 $P(x,y)$ 在新坐标轴中的坐标

$$P' = (\frac{|\vec{AP}|\cos\theta}{|AB|}, \frac{|\vec{AP}|\sin\theta}{|AB|}) = (dot(\vec{AB},\vec{AP}), det(\vec{AB},\vec{AP})) \cdot \frac{1}{|AB|^2}$$

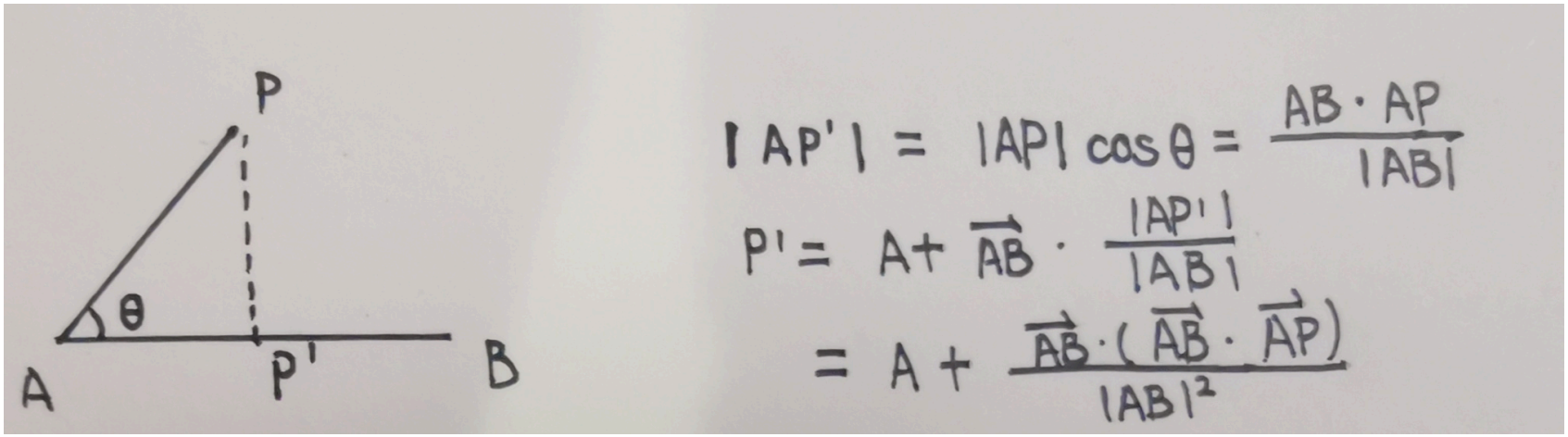
其中 θ 为 $\vec{AB}, \vec{AP}$ 之间的夹角
可以看出与三角函数有关的值可以利用叉积与点积转化

例 2-求点在直线上的投影

求点 p 在直线 AB 上的投影.

例 2-求点在直线上的投影

求点 p 在直线 AB 上的投影.

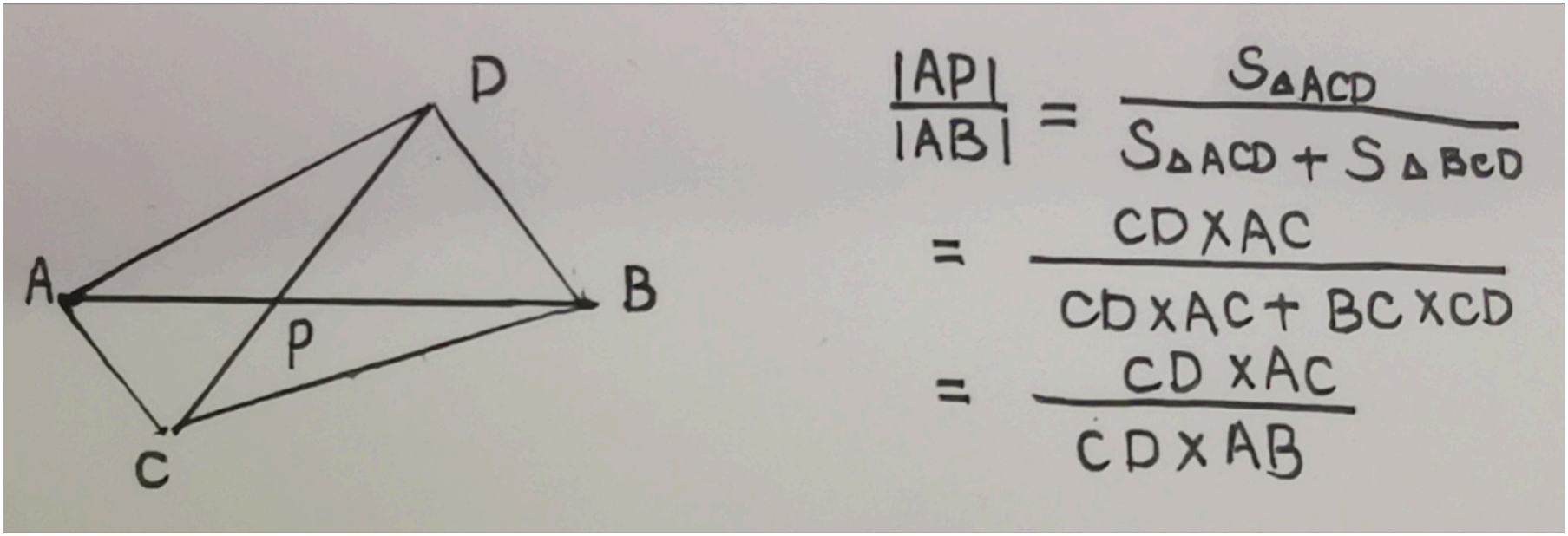


例 3-直线与直线的交点

两条不平行的直线 AB, CD，求它们的交点.

例 3-直线与直线的交点

两条不平行的直线 AB, CD，求它们的交点.



点是否在多边形内部

从这个点出发向某一个方向作一条平行于 x 轴的射线，可以根据这条射线与多边形的边相交次数的奇偶性来判断.
注意当交点为多边形顶点时会被重复统计，可以考虑每条边左闭右开的形式.
忽略多边形的水平边

点与线段的关系

```
inline int ccw(Point q, Point p0, Point p1)
{
    if(sign(cross(p1 - p0, q - p0)) == 1) return 1; // 逆时针
    if(sign(cross(p1 - p0, q - p0)) == -1) return -1; // 顺时针
    if(sign(dot(q - p0, p1 - p0)) == -1) return -2; // 后方
    if(sign(norm(q - p0) - norm(p1 - p0)) == 1) return 2; // 前方
    return 0; // 在线段上
}
```

求凸包的 Graham 方法

对于平面上的一个点集，求这样一个凸多边形

1) 所有点都在这个凸多边形的内部
2) 凸多边形的顶点为点集中的点

找到所有点中左下角的点 (y 为第一关键字,x 为第二关键字时最小).
将其他所有点以与它之间角度排序.
然后类似斜率优化一样维护一个单调栈即可.

皮克公式

对于一个格点多边形，设 I 为格点多边形内部的点，E 为其边上的点，A 为其面积，则有

$$A = \frac{E}{2} + I - 1$$

海伦公式

求三角形边长分别为 a, b, c 的三角形面积.

设 $p = \dfrac{a+b+c}{2}$，则有

$$S = \sqrt{p(p-a)(p-b)(p-c)}$$

平面图的欧拉定理

对于一张平面图，设 V 为点数，E 为边数，F 为面数，C 为联通块个数，则有

$$V - E + F = C + 1$$

例 4-[2020 省选十连测 day6] 网格

有一个无限大的网格图，每个格点四联通地连边，q 次操作，每次选择一个点，将它和与它相连的边删去，问每次操作后的联通块个数．强制在线．
$1 \leq q \leq 10^5$

例 4-[2020 省选十连测 day6] 网格

应用平面图的欧拉定理，V 和 E 很好维护，主要考虑 F.
观察一个不是 1×1 的小正方形的面，发现内部是一个已删除点的八联通块

半平面交

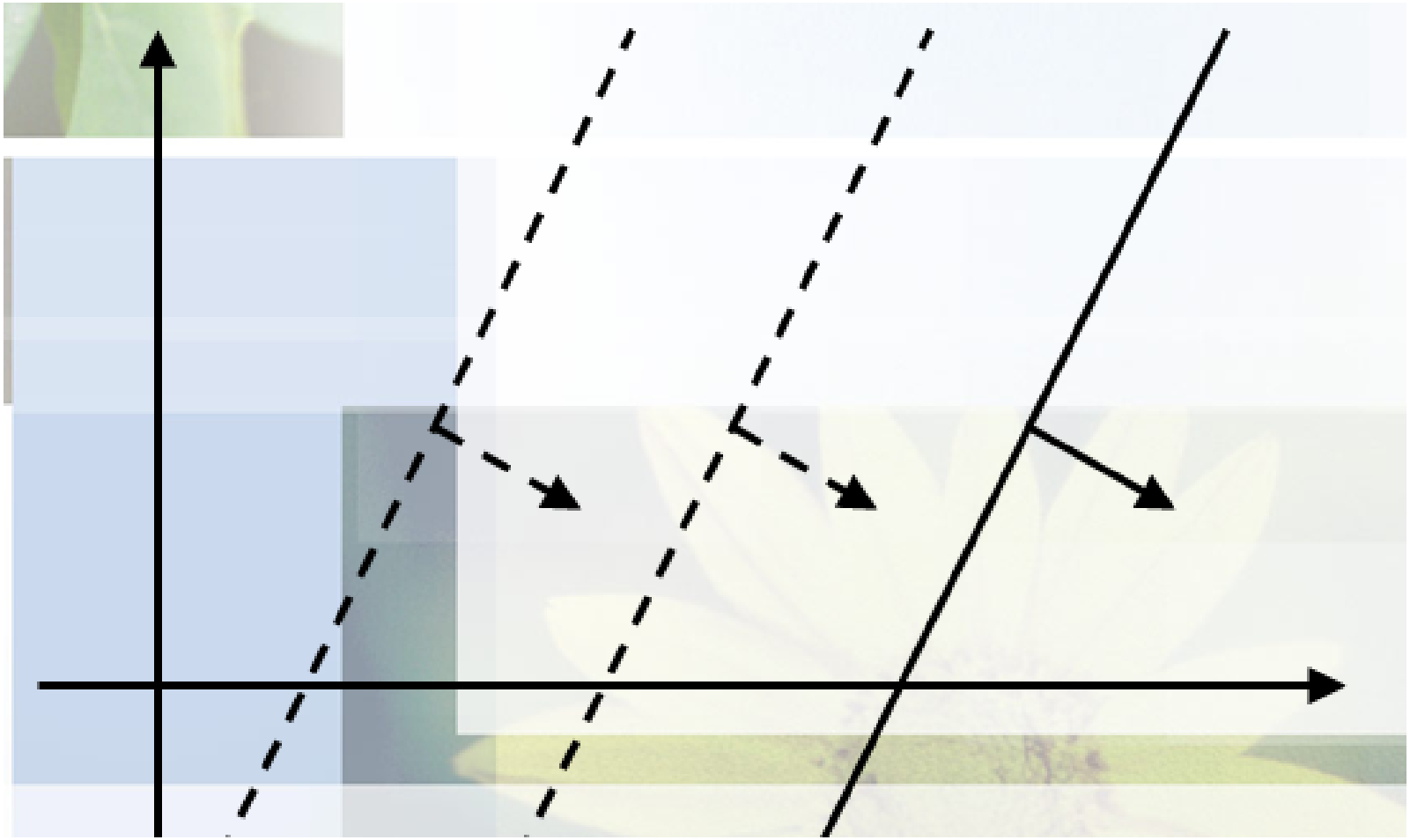
半平面，即一条直线和这条直线一侧 (接下来默认为逆时针方向) 的所有点的点集
半平面交，即多个半平面的交集，也是一个点集，加入边界半平面后，交集是一个凸多边形或空集.
也可以看做若干个形如 $Ax + By + C \geq 0$ 的方程联立后的解集.

S&I 算法

需要离线，将所有半平面排序，然后利用双端队列维护半平面交
排序复杂度 $O(n \log n)$，求解复杂度 $O(n)$.

S&I 算法 step1

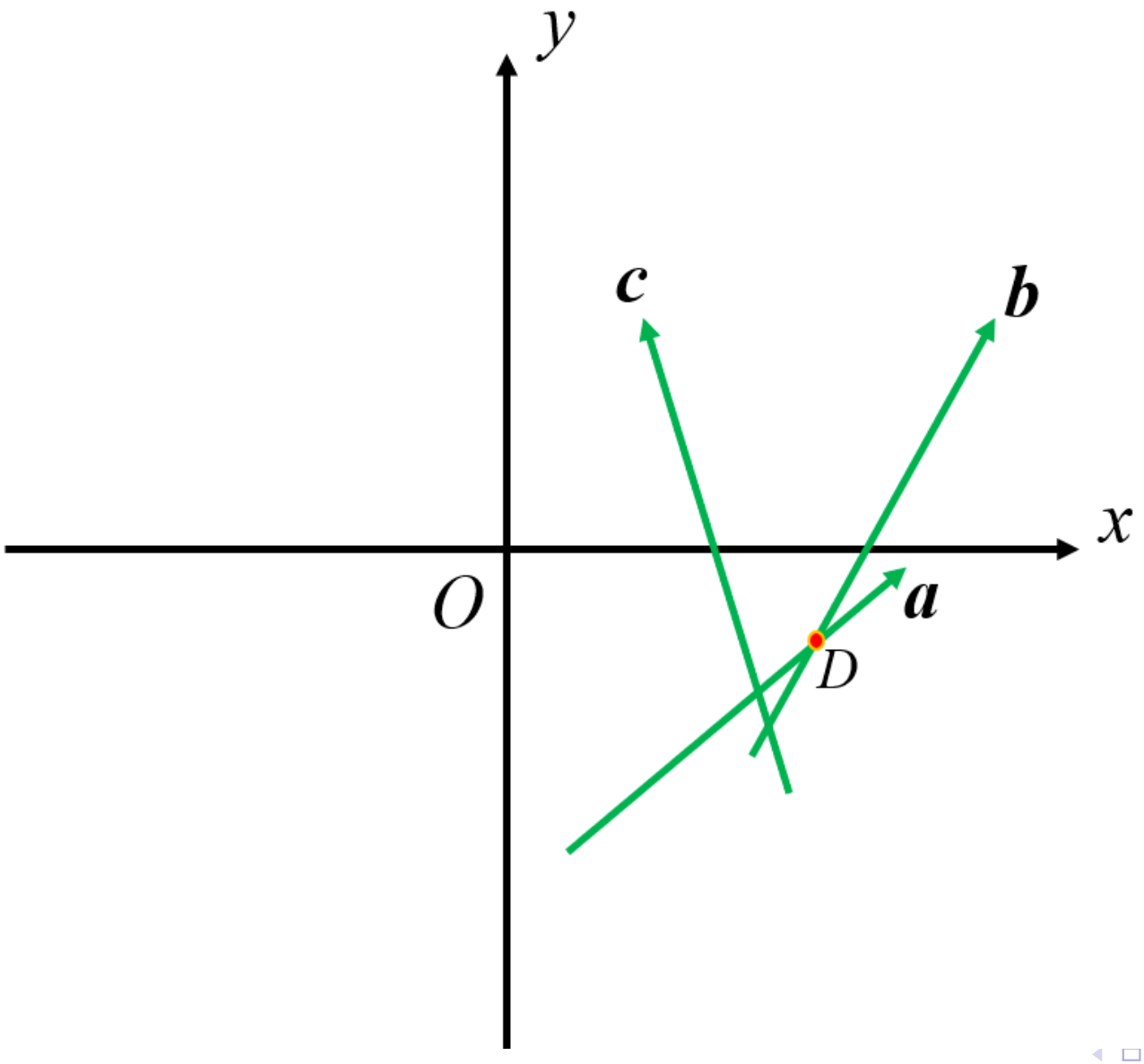
极角排序，角度相同的限制更强的排在前面.



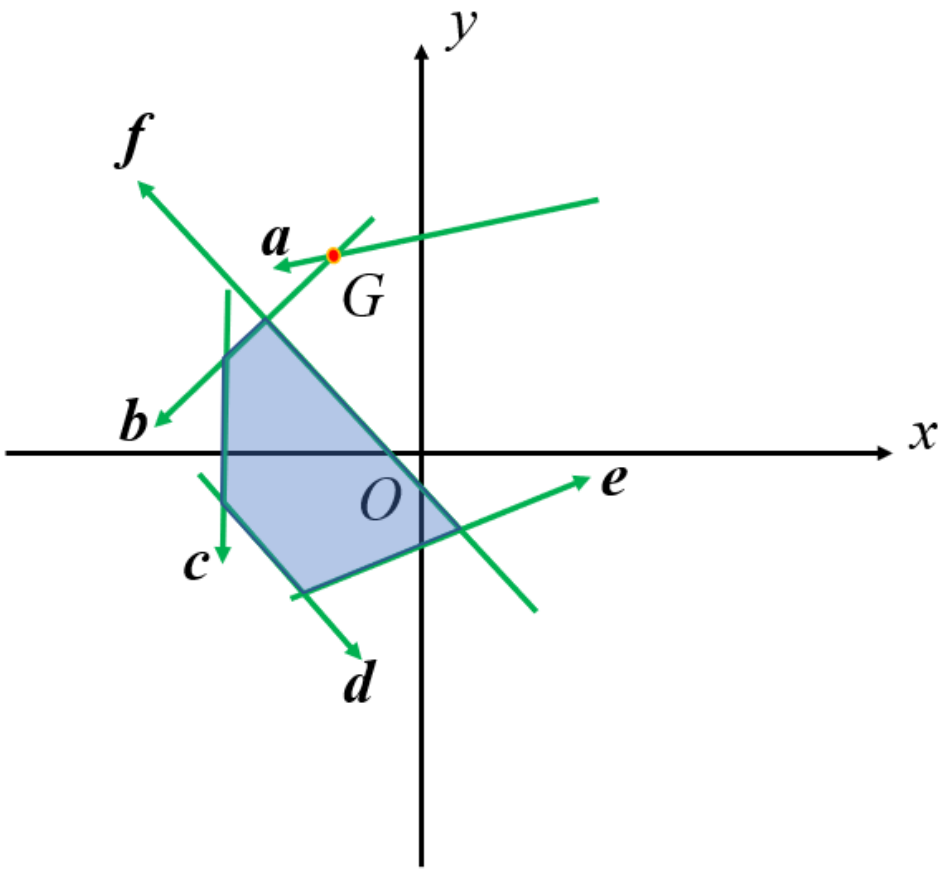
S&I 算法 step2

维护双端队列因为半平面交是一个凸多边形，所以需要维护一个凸壳. 因为后来加入的只能影响到最开始加入 (凸壳已联通) 或最后加入的边，所以需要双端队列.
我们遍历排序后的向量，并维护交点数组.
对于当前向量，如果上一个交点在这条向量表示的半平面交的异侧，那么上一条边就没有意义.

S&I 算法 step2



S&I 算法 step2

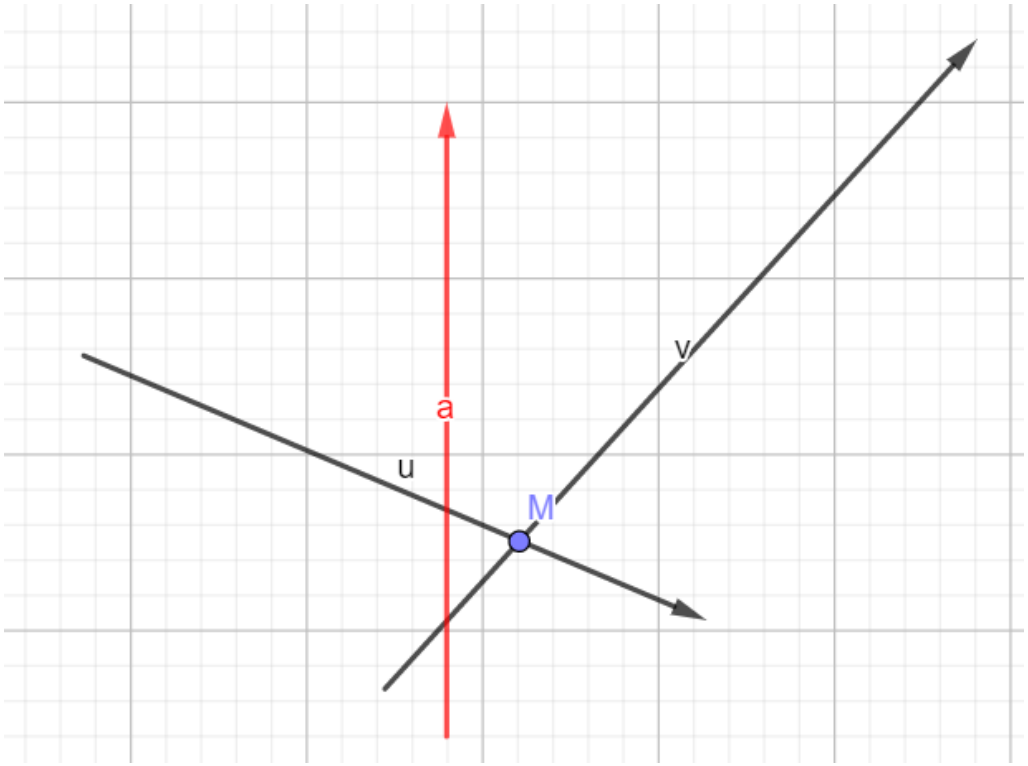


S&I 算法 step3

最后用队首的向量排除队尾多余的向量，因为队首的向量会被后面的约束，但是队尾的向量不会，但此时它们围成了一个凸多边形，所以队尾的向量应该被队首的向量约束.
最后得到的交点数组就是半平面交点集的顶点.

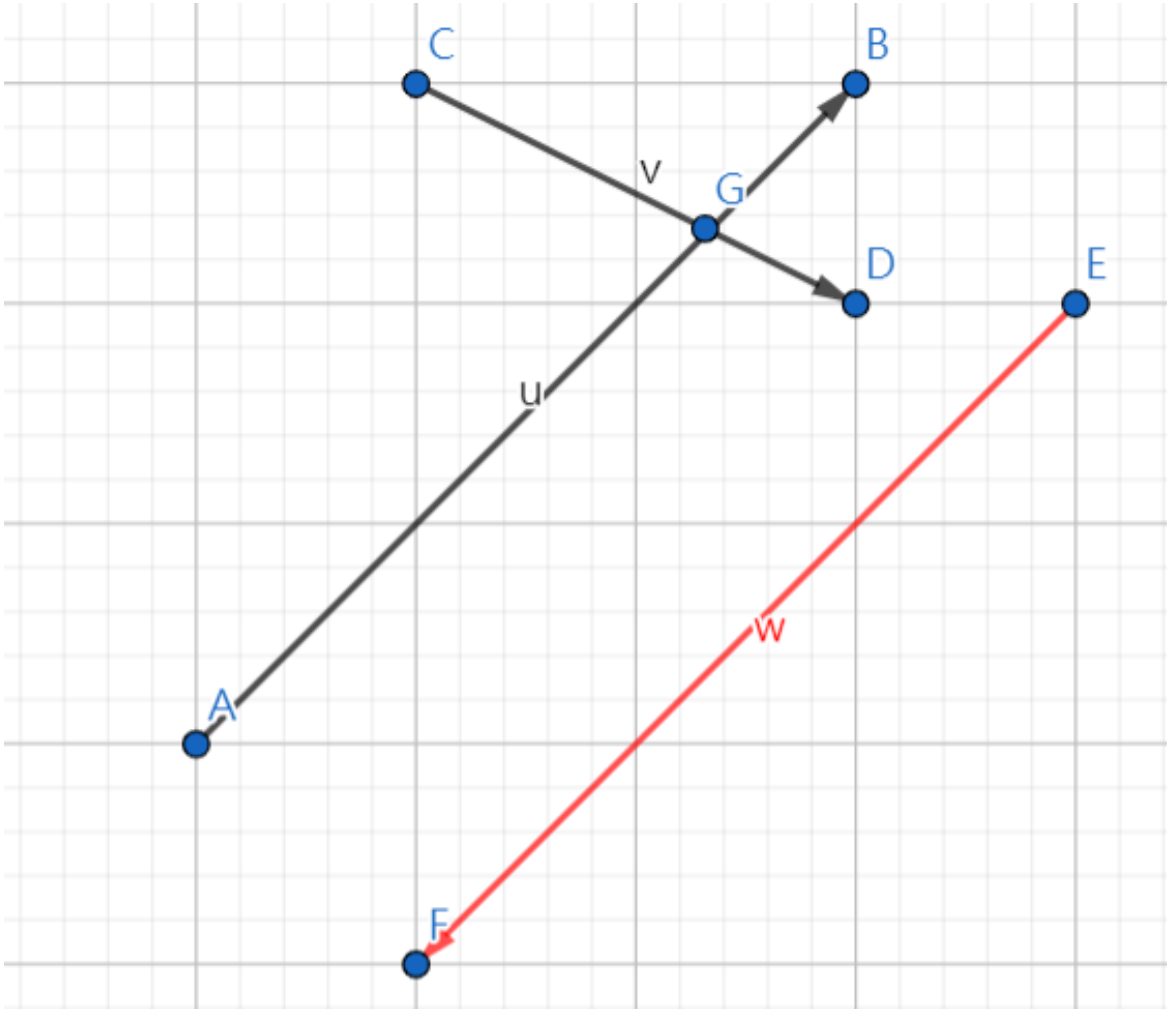
S&I 算法注意事项 1

当出现一个可以把队列中所有点弹出的向量，必须先处理队尾，再处理队首



由于是极角排序后的顺序，所以后面的 $\vec{v}$ 的影响较小，所以应该从队尾删除.

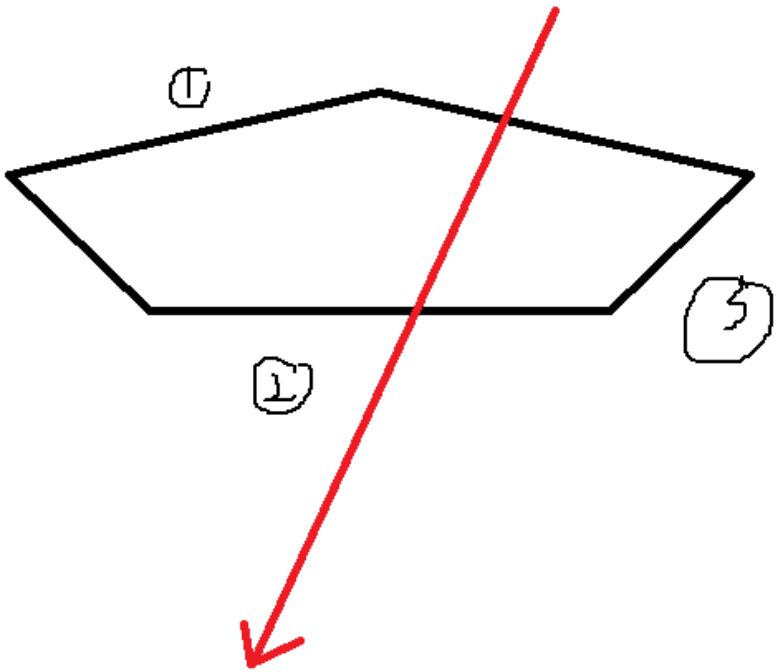
S& 算法注意事项 2



注意此时弹出队尾/队首后，现在出现队尾/队首两个向量平行的情况需特判其无解.

求半平面交的朴素算法

相交于 S&I 算法，可以支持 $O(n)$ 加入一条半平面，即可以在线.
维护当前的半平面交，即一个凸包，每次加入一条半平面时，对凸包上每一条边分情况讨论即可.



平面最近点对

平面上 n 个点，求它们两两之间欧几里得距离最近的点对的距离.
存在复杂度为 $O(n \log n)$ 的分治算法

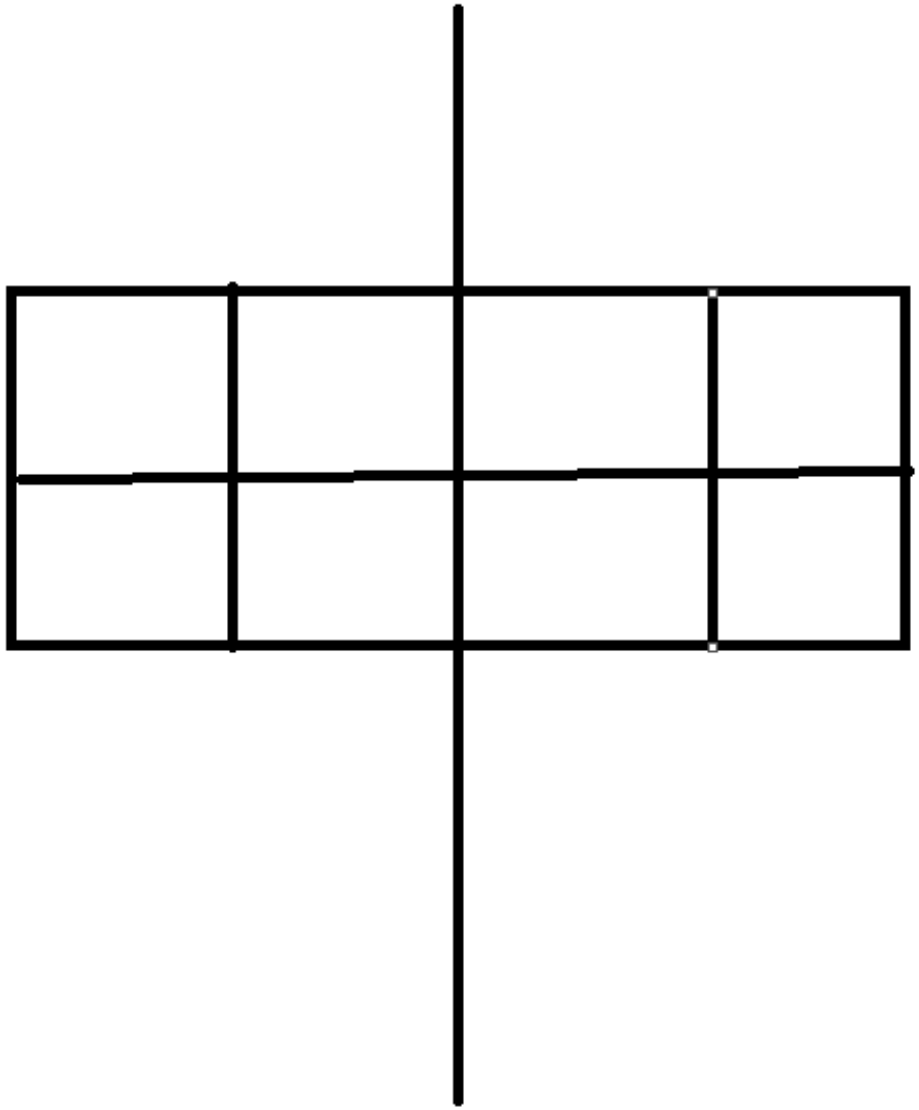
平面最近点对

考虑现在我们要求出点集 S 中的最近点对的距离.
首先处理出一个分治结构, 将它们按照 x 为第一关键字, y 为第二关键字排序, 将它们分为以 $p_m(m = \lceil \frac{n}{2} \rceil)$ 为分界线的两个点集 A, B
递归求出 A, B 中最近点对的距离, 即其中较小者为 h.
考虑如何利用 h 求出 A, B 之间的最近点对

平面最近点对

我们想要找到 A, B 之间距离 $< h$ 的点对，所以我们将考虑所有满足 $|x_i - x_m| < h$ 的点，设为集合 T.
枚举集合 T 中的点 (x_i, y_i)，由于两个距离 $< h$ 的点必然满足 $|y_i - y_j| < h$，所以如果我们将 T 按 y 排列，那么对于枚举的点有贡献的点是一个区间
而这个区间的大小是很小的，所以如此枚举即可.

平面最近点对



平面最近点对

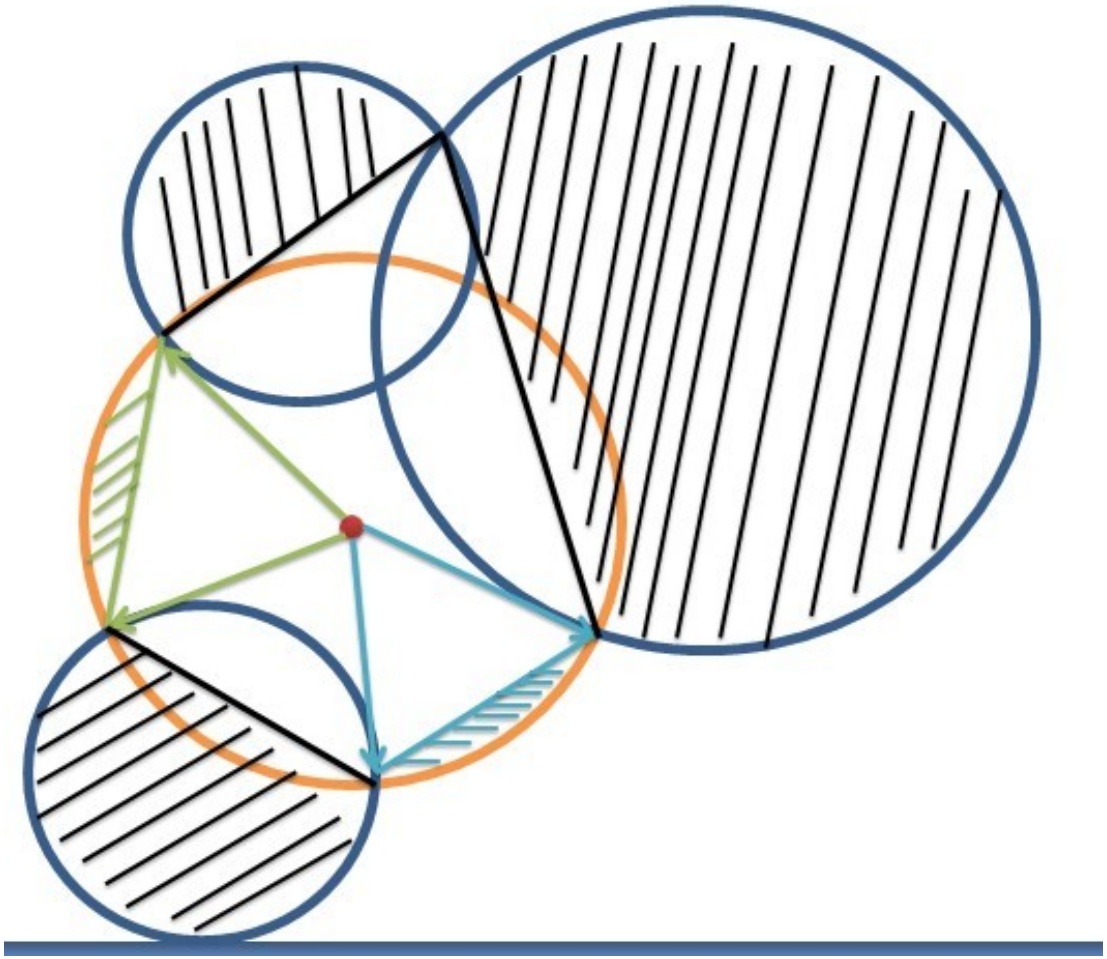
利用这种分治思想可以解决许多问题
例如平面上周长最小的三角形.

例 5-圆的并集

SPOJ CIRU
给出平面上的 n 个圆，求它们的并集的面积. $O(n^2)$

例 5-圆的并集

首先去掉被包含的圆和重合的圆.
考虑最后并集的形状.



例 5-圆的并集

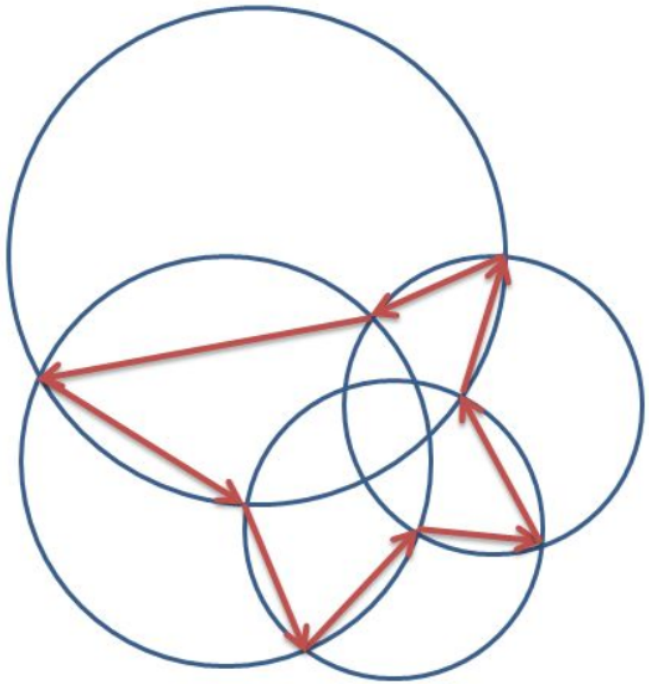
发现这是由若干弓形与简单多边形组成的一个图形
弓形相当于是没有被其他圆覆盖的部分的贡献.
发现简单多边形的边也就是弓形的边, 也就是没有被其他圆覆盖的部分
用类似算面积的方法累加起来即可

例 6-CIRUT

给出平面上 n 个圆, 对于 $k = 1, \cdots, n$ 求恰好被覆盖了 k 次的部分的面积. $O(n^2)$

例 6-CIRUT

其实和之前的算法区别不大，观察覆盖了至少 2 次的部分
发现其实是和之前类似的一个图形
需要修改的地方就是，不用去掉被覆盖的圆，对于被其他圆弧覆盖了 k 次的地方就将贡献累加到被覆盖至少 k 次的答案中即可



Delaunay Trianglation

一种三角剖分，满足每个三角形的外接圆都不包含其他点
即最小角最大的三角剖分

例 7-Sasha Circle

给出平面上两个点集 M,S, 其中 $|M| = n, |S| = m$
问是否存在这样一个圆, 满足一个集合中的点全部在圆$\mathbb{F}$, 另一个集合中的点全部在圆外
$1 \leq n, m \leq 10^4$
$-10^4 \leq |Mx|, |My| \leq 10^4$
$-10^4 \leq |Sx|, |Sy| \leq 10^4$
保证 $n + m$ 个点坐标两两不同

例 7-Sasha Circle

设在圆内部的集合为 A 集合, $|A| = n$, 圆外的集合为 B 集合, $|B| = m$
枚举 M, S 分别作为 A, B 集合

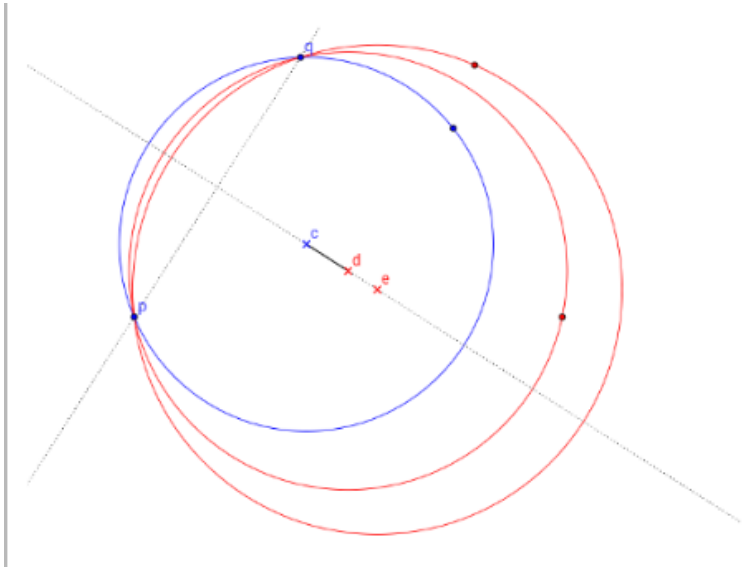
例 7-Sasha Circle

通过放缩，一定可以让圆通过 A 集合中的至少两个点，我们可以考虑枚举 A 中的每一对点 (p, q)，圆心的合法位置相当于在线段 pq 中垂线上的一段线段，设为 $l(x)$

不在 pq 直线上的每个点都会更新 $l(x)$ 定义域的上界或下界.

在 pq 直线上的点无法通过调整圆来改变位置关系

最后判断 $l(x)$ 定义域是否为空即可

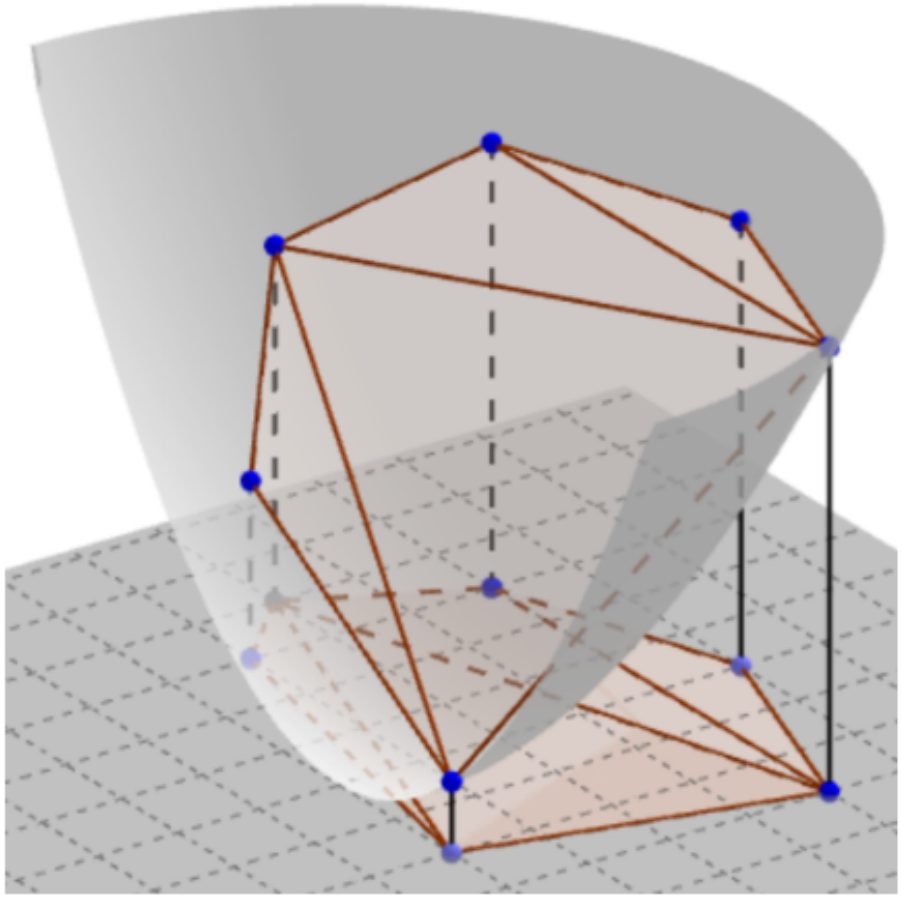


例 7-Sasha Circle

首先我们考虑一个抛物面 $z = x^2 + y^2$, 与任意一个平面 $ax + by + z = c$ 它们的交集可以表示为 $ax + by + x^2 + y^2 = c$, 发现这映射到平面上就是一个圆的形式.
进一步观察发现圆$\boxed{\mathrm{F}}$的点就相当于平面下方抛物面上的点, 圆外的点就相当于平面上方抛物面上的点
我们可以将 A, B 点集上的点的点映射到抛物面上为 A', B', 那么我们就需要找到一个平面, 使 A' 集合中的点都在平面下方, B' 集合中的点都在平面上方.

例 7-Sasha Circle

类比朴素算法，我们可以枚举两个点来判断是否存在这样的平面，思考发现，只有 A' 集合的上凸壳上的边才有可能产生合法的平面



例 7-Sasha Circle

每个三角形的外接圆都不包含其他的凸包上的点的三角剖分，即最小角最大的三角剖分，被称为 Delaunay Trianglation，我们要求的与之恰好相反，被称为 Anti-Delaunay Trianglation，可以分治算法在 $O(n^2)$ 的时间算出，具体操作为对于一条边，找到凸包上和它夹角最小的那个点.
对于三角剖分的每一条边利用刚才的朴素算法即可

例 8-Spectator Riots

场地为左下角与右上角分别为 $(0,0),(10^5,10^5)$ 的矩形 (包括边上的点)
有 n 名观众, 第 i 名观众初始位于 (x_i,y_i) , 速度为 v_i 表示他在 1s 后可以到达 $(x_i+p,y_i+q),|p|+|q|\le v_i$ 且在场地内的点
每 1s 观众会在所有他能到的点中等概率选择一个点, 要求在初始状态 1s 后可能有观众出现的点中选择不共线的三个点组成三角形, 使初始状态 1s 后在三角形的外接圆内的人数期望最大, 如果有多组方案, 选择半径最大的任意一组输出即可
$n\le 10^5$
$0\le x_i,y_i\le 10^5$
$v_i\le 1000$
保证答案半径不超过 10^{10}

例 8-Spectator Riots

statement 1

凸包点中外接圆半径最大的三角形的外接圆一定包含整个凸包

例 8-Spectator Riots

statement 1

凸包点中外接圆半径最大的三角形的外接圆一定包含整个凸包

statement 2

凸包外接圆半径最大的三角形由凸包上相邻的三个点构成

例 8-Spectator Riots

statement 1

凸包点中外接圆半径最大的三角形的外接圆一定包含整个凸包

statement 2

凸包外接圆半径最大的三角形由凸包上相邻的三个点构成

所以我们只需要求出凸包，枚举相邻的三个点即可

对于每个观众，可以将他的行动范围的凸包求出来

例 9-Geometers Anonymous Club

定义两个凸包 A, B 的和为 $C = \{a + b | a \in A, b \in B\}$
定义 p 个凸包的和为前 $p-1$ 个凸包的和与第 p 个凸包的和.
给出 n 个凸多边形, q 次询问, 每次询问 $[l, r]$ 区间内凸包的和.
凸包中不包含三点共线.
$1 \leq n \leq 100000$
总点数不超过 100000
$1 \leq q \leq 100000$

例 9-Geometers Anonymous Club

假设我们现在要计算两个凸包 a, b 的和. 首先让我们表示两个自由向量序列.

1. $u_1, \cdots, u_{k_a}$，其中 u_i 等于从 a 的第 i 个节点出发，第 $i+1$ 个节点结束的向量
2. $v_1, \cdots, v_{k_b}$，其中 v_i 等于从 a 的第 i 个节点出发，第 $i+1$ 个节点结束的向量

首先，由于同一多边形中没有三点在同一直线上，所以同一组中的不可能存在两个方向相同的向量.

考虑如何为结果的凸包构造类似的序列. 我们枚举 a 中的一个向量 u_i，分析它的贡献

例 9-Geometers Anonymous Club

当 b 中存在向量 v_j 和 u_i 方向相同时，考虑对应的两条边的凸包和. 我们可以得到一个长度为 $|u_i| + |v_j|$ 的线段. 穿过这条线段的直线将平面分为两个半平面，而 a, b 的凸包和只会在其中一个半平面内 (因为本来两个凸包在也只在对应边的半平面内). 因此结果的凸包序列中有一个向量 $u_i + v_j$

当 b 中不存在向量 v_j 和 u_i 平行时. 那么如果我们画一条与 u_i 平行的线., 那么与 b 中会存在一个恰好一个点与之相交，也就相当于一个零向量，与上面一种情况类似的分析，结果的凸包序列中会有一个向量 u_i

例 9-Geometers Aonymous Club

也就是说，凸包和的序列中向量的个数相当于 a, b 中向量方向的总数.
而这也可以扩展到多个多边形的凸包和
那么问题就是相当于查询区间中不同的元素个数.