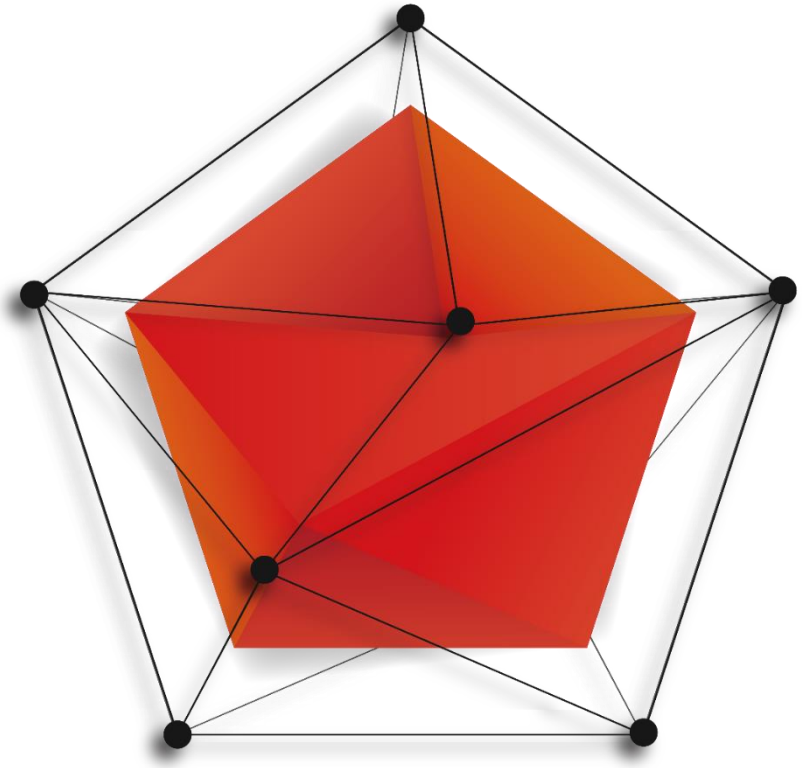
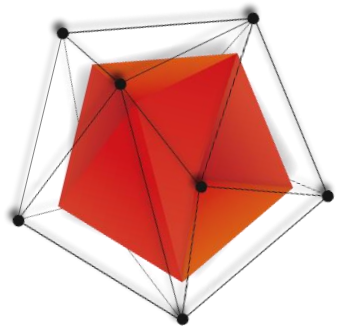


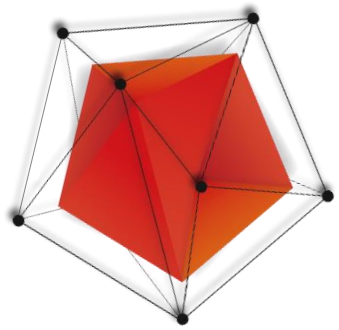
计算几何

颜苏卿



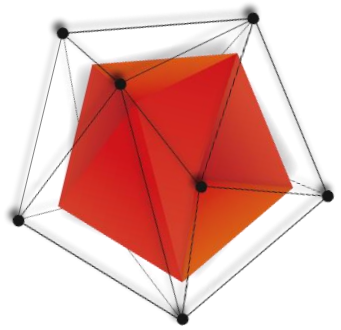
0.前言

- 计算几何算是算法的一个相对比较独立的分支
- 运用非常广泛，例如图形学，CAD，etc
- 计算几何往往不是水题，但也一般不会是防AK题（hdu4130级别的除外）
- 计算几何的题目，往往代码量大、特殊情况多、精度难以估计，简称玄学（误）



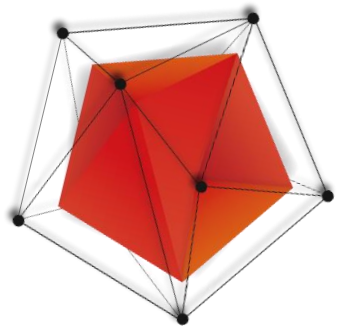
1.前置知识

- 头文件 math.h
- 浮点数 float, double
- 常用常量
  - const double PI=acos(-1.0);
  - const double INF=1e100;
  - const double eps=1e-6;



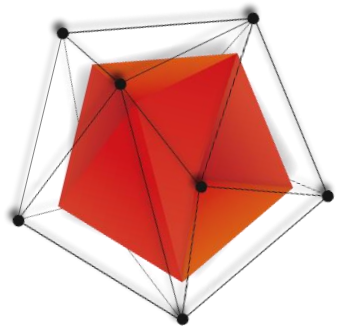
1.前置知识

- 精度控制——玄学
  - （其实不是玄学，如果你看过数值分析
- 判断等于 fabs(a - b) < eps;
- 尽量少用三角函数、除法、开方、求幂、取对数
  - 只是少用，该用的地方还是要用的
- 因此尽量先写出公式，然后化简之，最好是仅限于加、减、乘



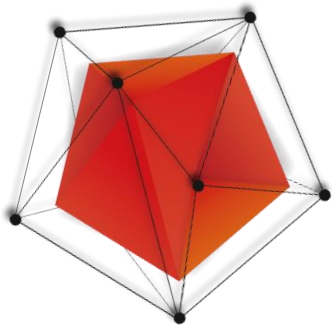
1.前置知识

- 符号函数
  - int sign(double x)
  - {
  - if (fabs(x) < eps) return 0;
  - else return x > 0 ? 1 : -1;
  - }
- 浮点→整数
  - floor(), ceil(), (int)x+0.5



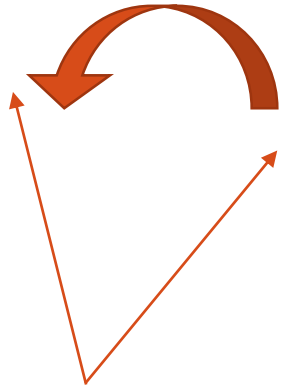
2.向量

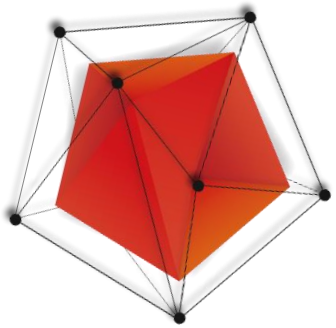
- 来来来，帮大家复习一下线性代数~~~
- a=(x1,y1)，b=(x2,y2)
- a+b=(x1+x2,y1+y2)
- a-b=(x1-x2,y1-y2)
- a·b=|a||b|cosθ
- |a×b|=|a||b|sinθ



2.向量

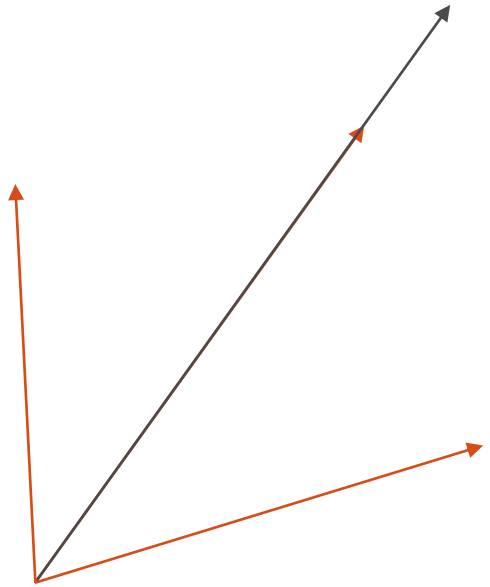
- 点(x,y)关于点(x0,y0)逆时针旋转θ后为(x1,y1)
  - x1=(x-x0)*cosθ+(y-y0)*sinθ+x0
  - y1=(x-x0)*sinθ+(y-y0)*cosθ+y0
- 向量间夹角——acos, asin
- 判断向量方向、位置
- 求三角形面积(两边叉积)
- 判断三点共线(叉积为0)

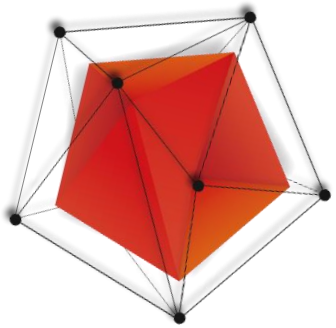




2.向量

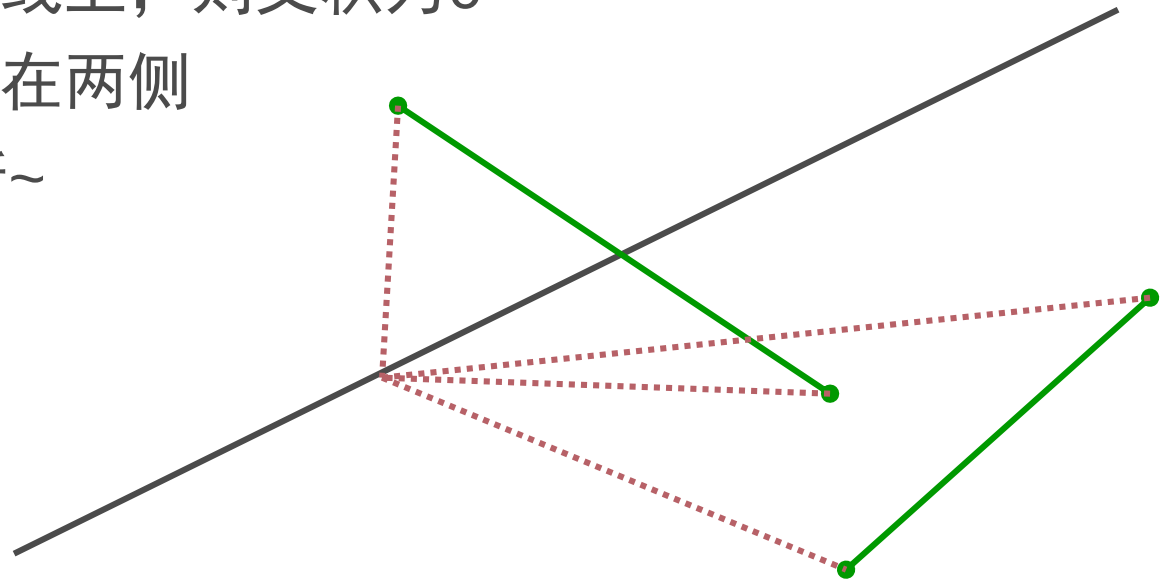
- 点在线段(以及其所在直线)上(叉乘 = 0)
- 点在线段顺时针方向(叉乘 > 0)
- 点在线段逆时针方向(叉乘 < 0)

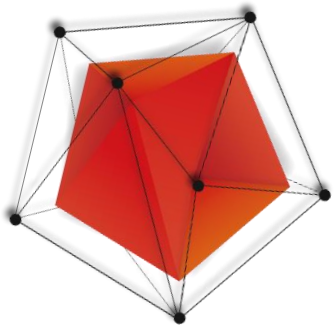




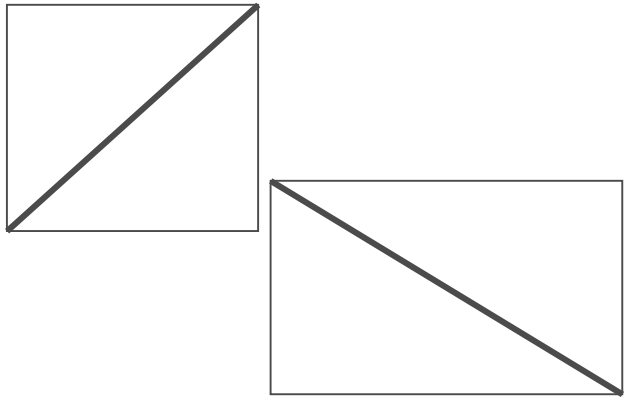
2.向量

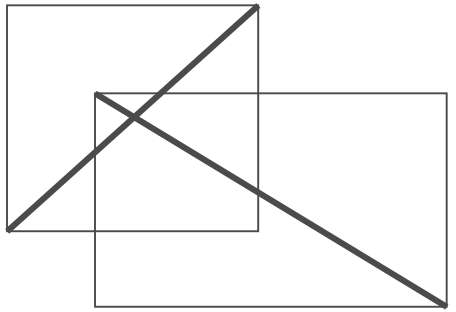
- 判断直线与线段(s,e)相交
- 考虑直线上取一点，若端点在直线上，则叉积为0
- 若线段与直线相交，则交点必定在两侧
- 因此利用叉积的符号积<1可判断~



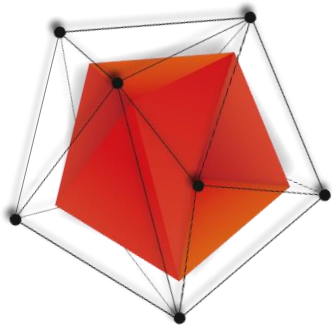


2.向量

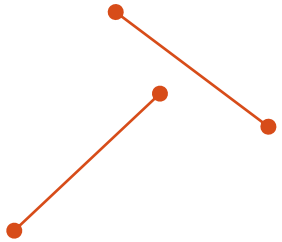


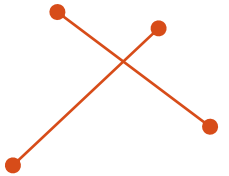


- 线段的相交——规范相交、非规范相交
- 规范相交，则两个端点在另一条线段的异侧
- 如何判断两条线段相交？
- 第一步、快速排斥试验
  - 设以线段 P1P2 为对角线的矩形为R，设以线段 Q1Q2 为对角线的矩形为T，如果R和T不相交，显然两线段不会相交。

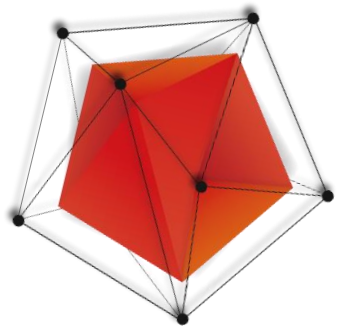


2.向量



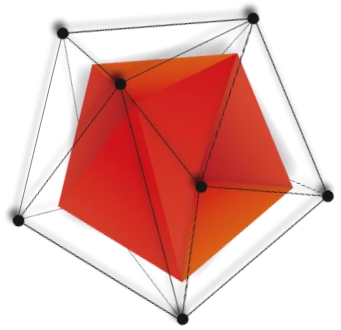


- 第二步、跨立试验
- 如果两线段相交，则两线段必然相互跨立对方。
- 若P1P2跨立Q1Q2，则矢量 (P1 - Q1) 和(P2 - Q1)位于矢量 (Q2 - Q1) 的两侧。
- 同理判断Q1Q2跨立P1P2。
- 为什么要分别测试？



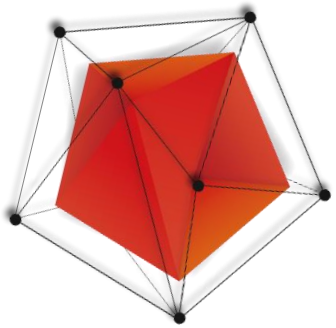
2.向量的其他运用

- 顺便复习了一下自己的模版——
- 点到直线¥线段的距离
- 关于直线对称的点
- 两点中垂线
- etc



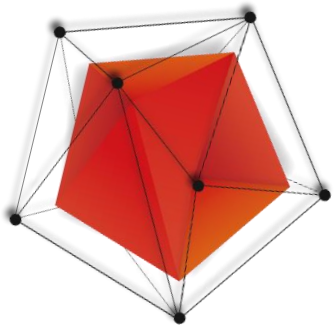
3.三角形

- 三角形面积S
- S = a * h / 2
- = a * b * sin(C)/2
- = a * b * c / (4R)　　//外接圆半径
- = a * b * c * r / 2　　//内切圆半径
- = sqrt(p * (p – a) * (p – b) * (p – c))　//海伦公式
- 当年年轻的我曾经暴力套海伦QAQ



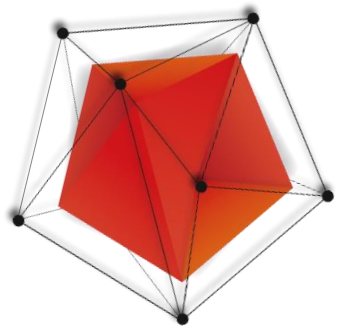
3.三角形

- 外心：三边中垂线交点，到三角形三个顶点的距离相等
- 内心：角平分线的交点，到三角形三边的距离相等
- 垂心：三条高线的交点
- 重心：三条中线的交点，到三角形三顶点距离的平方和最小的点，三角形内到三边距离之积最大的点
- 费马点：到三个顶点距离之和最短的点
- 以上均可编程求解~
- 拓展：四边形费马点？



4.多边形

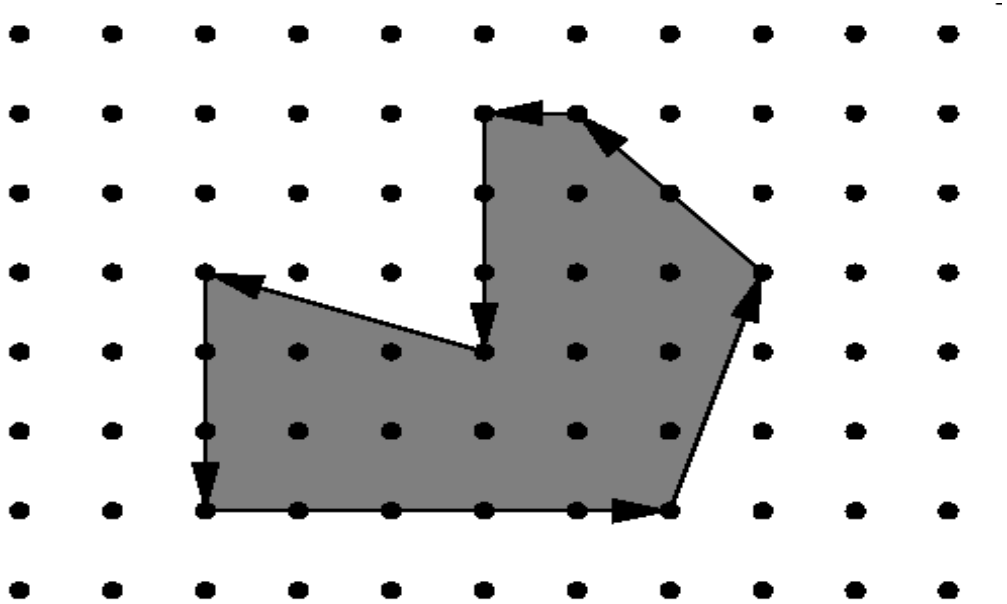
- 我们一般讨论的都是简单多边形，也就是除了邻边之外没有任意两边相交~
- 定理：仅使用对角线，则任意一个n多边形可以被拆成n-2个不重叠三角形

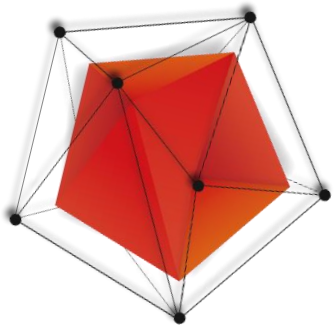


4.多边形

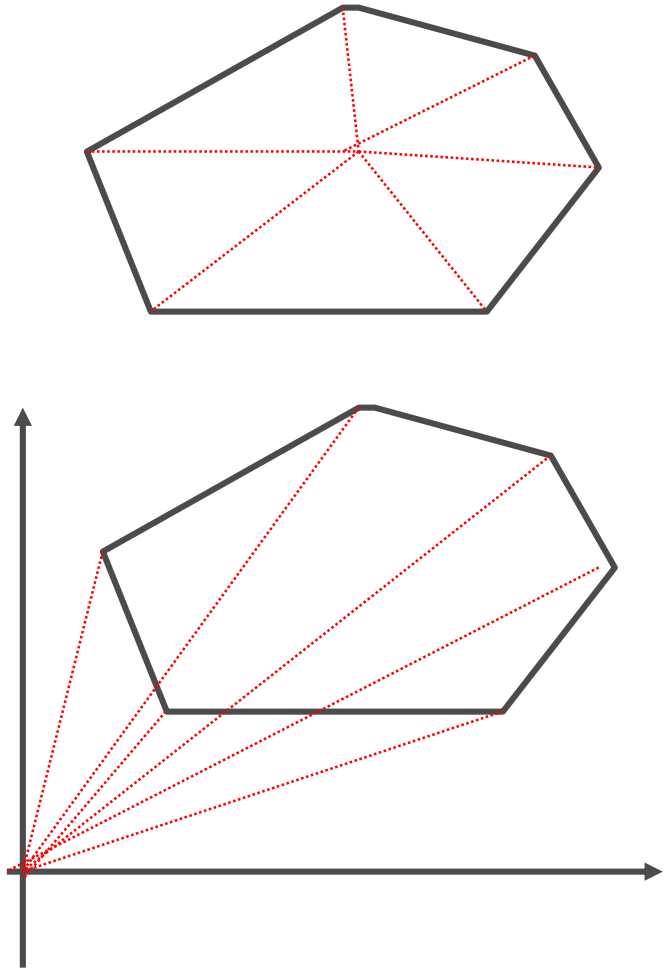
$$Area = \frac{1}{2}EdgeDot + InnerDot - 1$$

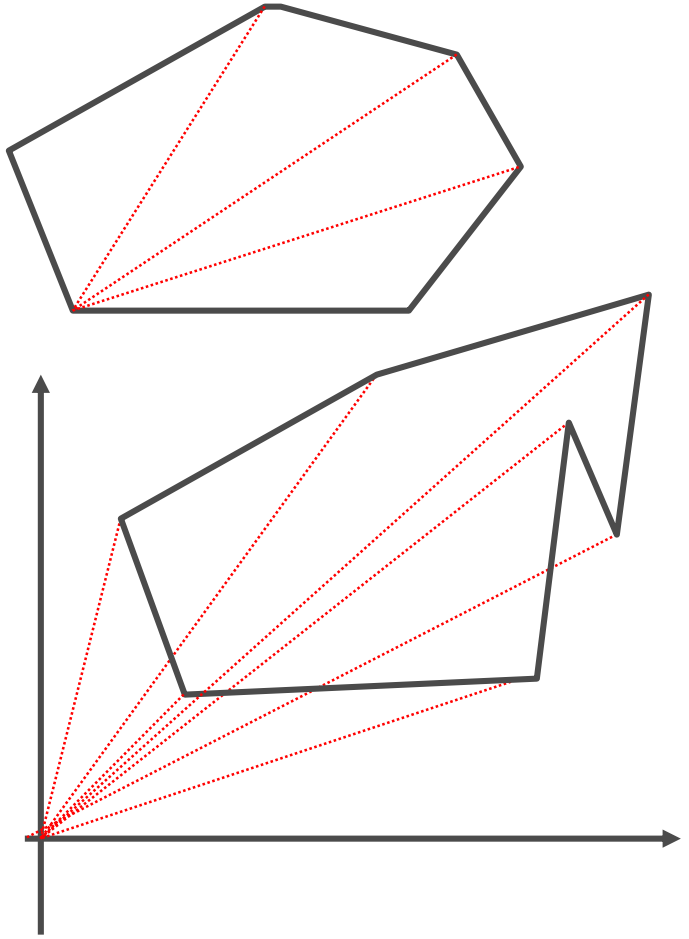
- 从最简单的开始——如何求一个多边形的面积？
- 若在网格下——

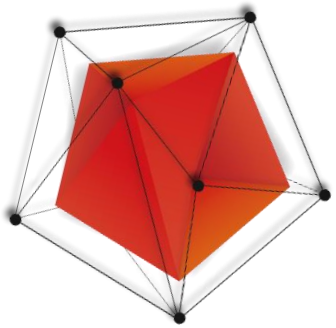




4.多边形

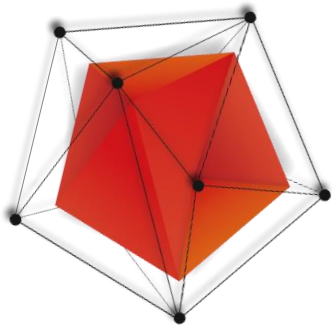




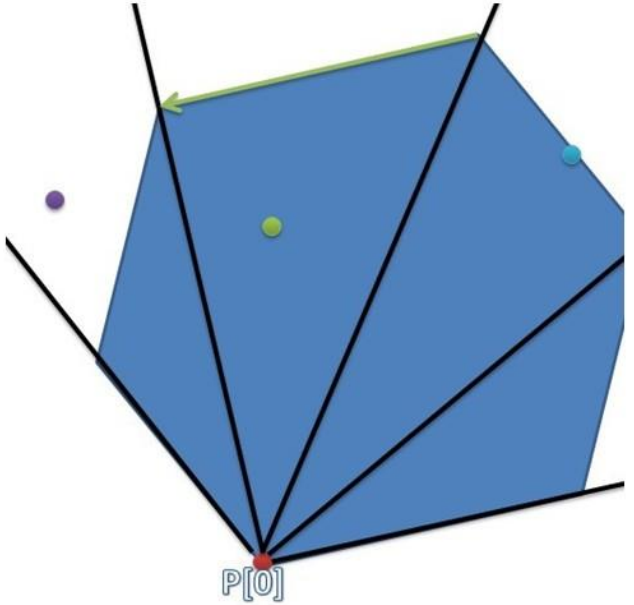


4.多边形

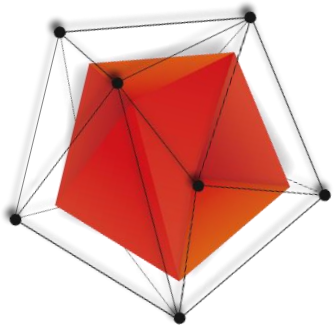
- “有向面积”能够更本质地算出面积！
- 再来一个问题——多边形的重心？
- 利用有向面积求出！求出每个三角形的重心，利用有向面积求加权平均~



4.多边形

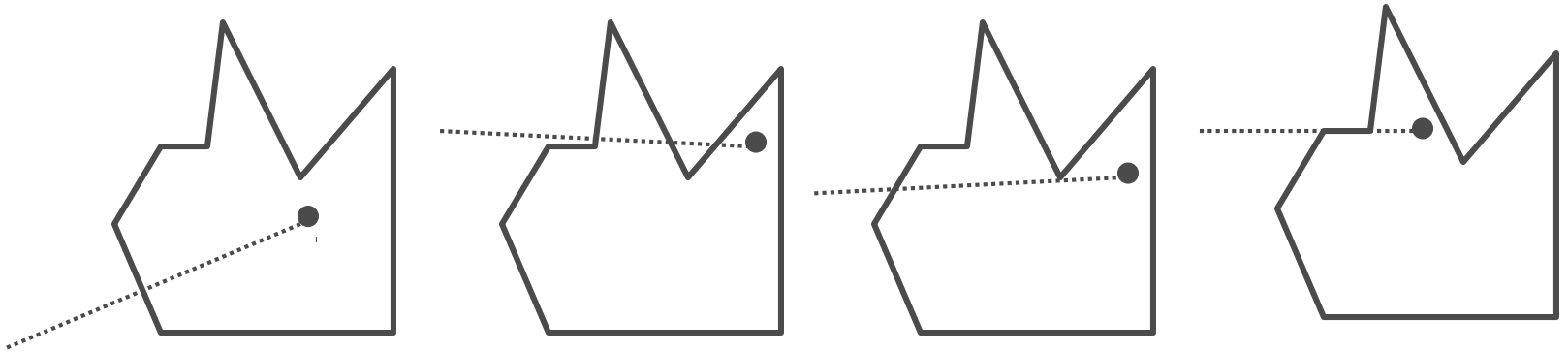


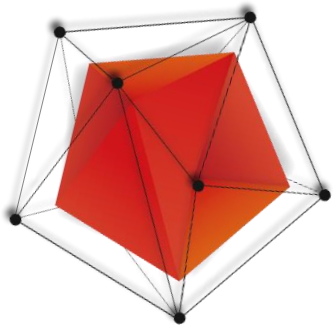
- 点与多边形的关系
- 如果是凸多边形——则根据三角剖分，将凸多边形分为n-2个三角形，然后二分判断是否在某一个三角形中；之后判断是否在对于的边的左侧（逆时针）
- 总时间 O(logn)



4.多边形

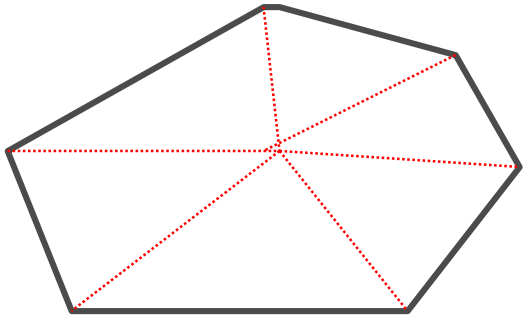
- 对于一般多边形，介绍两种方法：射线法（常用）、面积法
- 射线法：从这个点往外作一条射线，判断与多边形的交点个数
  - 注意：复杂情况相当多！

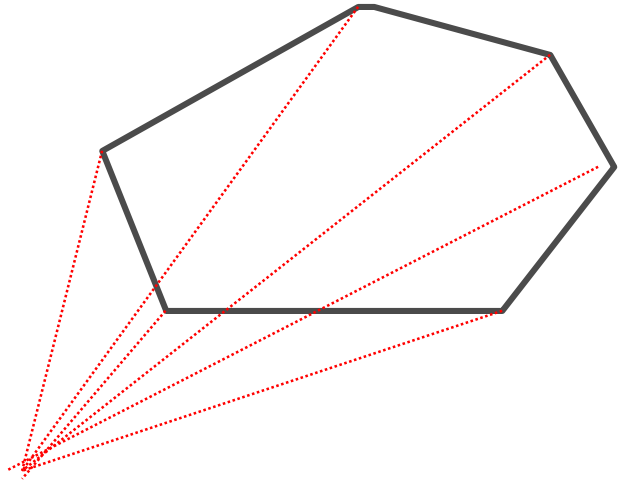


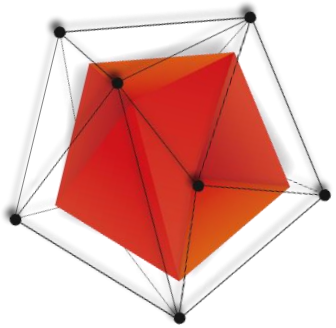


4.多边形

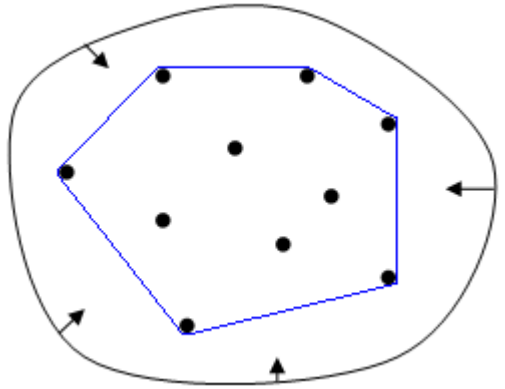
- 面积法：计算以这个点到多边形所有边的面积之和，若与多边形面积相等，则在多边形内，否则在多边形外。（容易有精度问题）



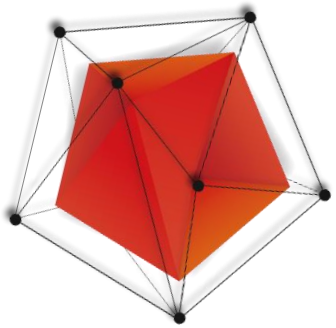




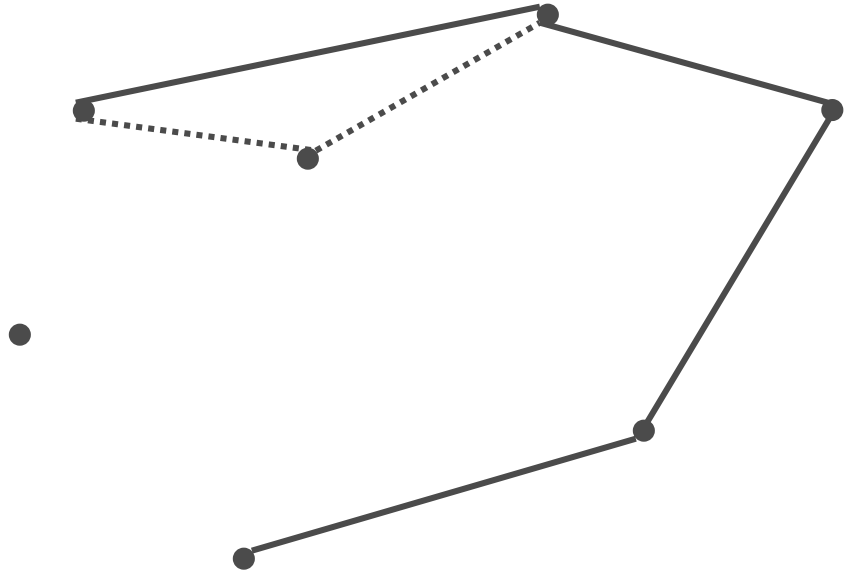
5.凸包



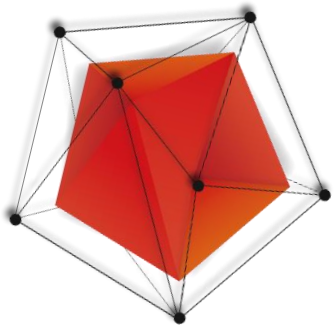
- 定义：凸包是包含所有给定点的凸多边形的交集
- 通俗一点来说，最外面的所有点
- 再通俗一点来说，在每个点的位置上钉一个钉子，拉开一根橡皮筋环，让它能够包住所有点，然后让它收缩，最后形成的就是凸包



5.凸包

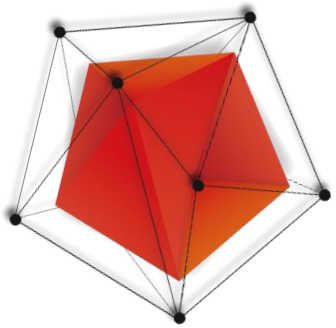


- 常用的凸包求法：
- Graham扫描法，最常用，时间复杂度O(nlogn)
  - 按照先y后x的优先度排序，找出p[0]
  - 相对于p[0]极角排序，然后依次加入每个点，看是否在上一次匹配的边的左侧，若在左侧，则加入该点，否则退回上上一条边，再次检查。



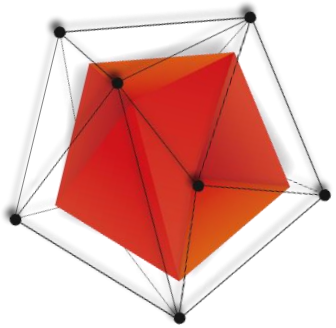
5.凸包

- Jarvis步进法（卷包裹法），偶尔使用，复杂度O(nm)，m为凸包顶点数
  - 先找出最左侧点p[0]
  - 然后找出p[1]，使得剩余所有点都在(p[0],p[1])左侧；
  - 再找出p[2]，使得剩余所有点都在(p[1],p[2])左侧；
  - 以此类推~

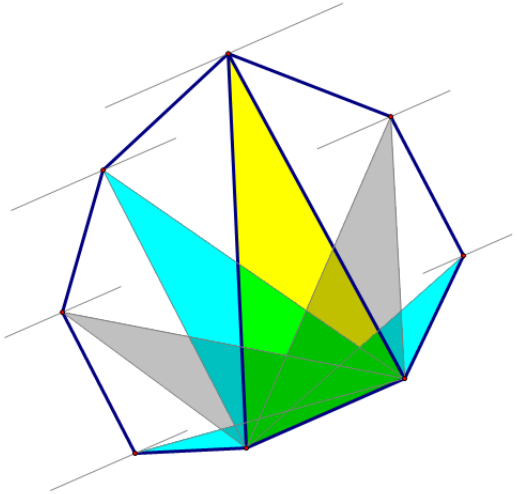


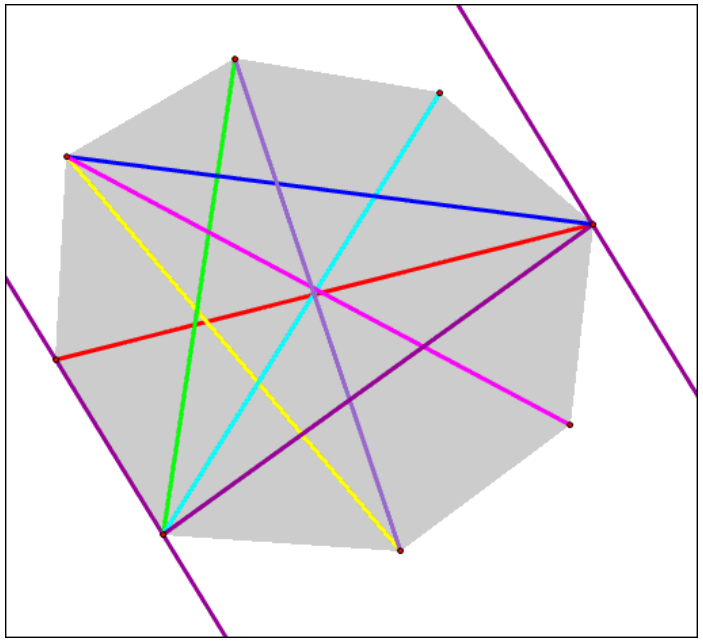
5.凸包

- Melkman算法，利用双端队列代替栈，在线，不需要极角排序（听说最快？还没遇到必须用这个的题目……
  - 要求比较多，要求输入的点已经是有序的，时间复杂度O(n)
  - 时间复杂度不是重点，重点是这是个在线算法，可以加入点
- 具体实现步骤——
  - 先找p[i]，p[i]必须不与(p[0],p[1])共线
  - 先向双端队列d中压入p[i],p[0],p[1],p[i]，设左指针为l，右指针为r
  - 让i++，直到(d[l],d[l-1],p[i])不左转或者(d[r-1],d[r],p[i])不左转
  - 退右指针，直到左转，加入该点；
  - 退左指针，直到左转，加入该点；

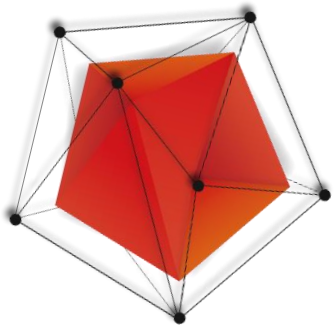


5.凸包



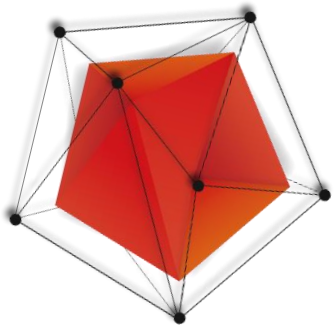


- 欧拉距离最远点对——旋转卡壳
- 首先可以肯定，最远点对必定在凸包上
- 对踵点——过这两个点作平行直线，则所有点都在这两个平行直线之间，那么这两个点被称为一对对跖点
- 最远点对必定是对踵点对
- 双指针寻找对踵点对，并更新最大值



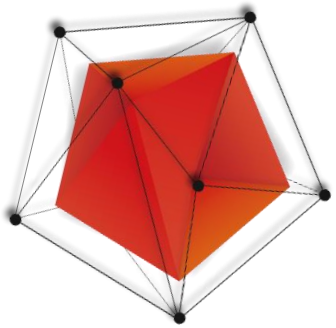
5.凸包

- 额外拓展——最近点对，分治法
- 凸包之间最大距离、最小距离同样可以利用旋转卡壳来求，这时两个点分别在两个凸包上；
- 这里不赘述，有兴趣的同学可以自学哟~

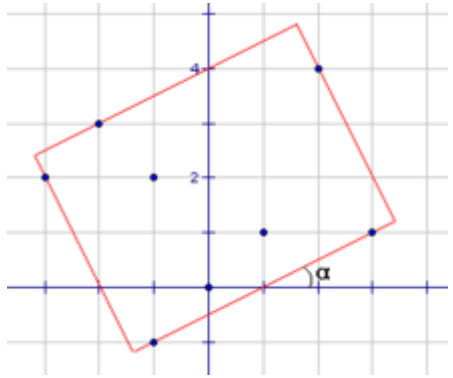


5.凸包

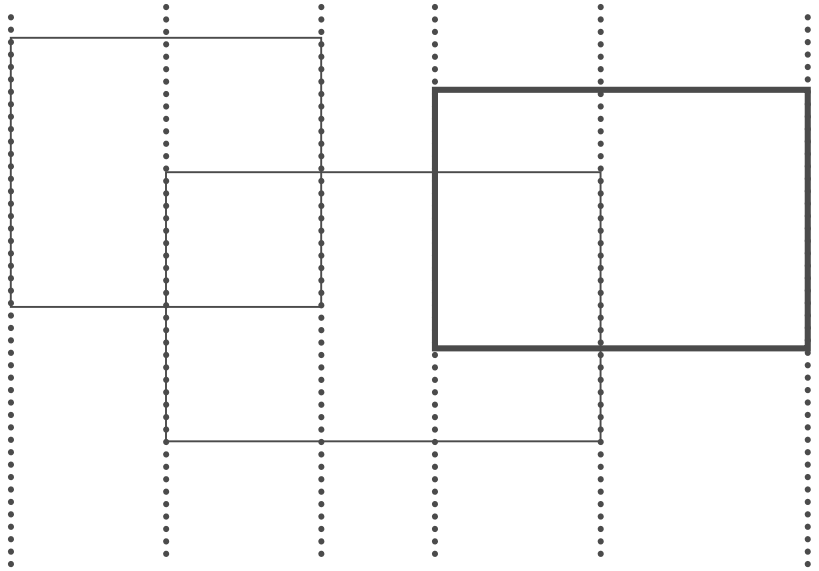
- 半平面交
- 顾名思义——半平面是平面的一部分
- 半平面交即求多个半平面的交，最终往往是一个多边形
- 重要应用：
- 求多边形的面积交
- 求多边形的核，即可以看到多边形中所有点的点集

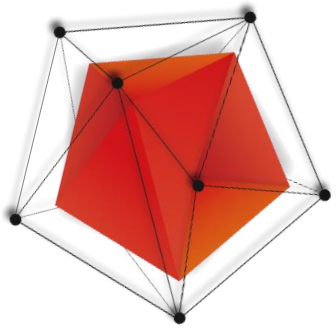


6.其他



- 离散化





没了