**判断线段转向问题**

1. 一条线段是在与其共享一个端点的另一条线段的顺时针方向还是逆时针方向；
2. 当沿着两条相邻的线段前进时遇到交点应该向哪个方向转；
3. **对一个由n个端点构成的序列按极角进行排序 ，要求时间复杂度为O(nlogn)，并要求用叉积来比较极角的大小；**（暂时未解决）

**判断线段是否相交**

1. 我们使用**红黑树**来存储完全全序序列T；
2. 运用红黑树来维护一个线段集合的垂直顺序，这种方法并**不是**显示的**记录红黑树关键字值**，**而是通过计算叉积来确定与同一个给定的垂直线相交的两条线段的相对位置；**
3. 当线段s1,s2**在扫除线处相交**的时候,这两条线段在完全前序关系中的位置无法判断；
4. 我们可以**将关键字的比较替换为叉积**的比较，用以确定两条线段的相对位置；
5. 时间复杂度分析： 一开始对2n个测点进行排序的时间复杂度是O(nlogn)；红黑树的基本操作时间复杂度为O(logn)，一共有2n个测点，所以总的时间复杂度为O(nlogn)；
6. 扫除提供了一种将几何物体排序的方法，通常是将其放入一个动态数据结构，从而充分利用其相互关系；
7. 当向T序列中插入元素的时候要判断插入之后的位置与相邻的元素是否相交，同理，当元素出序列时，也要检查它退出之后相邻的两个元素是否相交；

**寻找凸包**

1. 两种算法都是按**逆时针方向**的顺序输出凸包的各个顶点的；
2. Jarvis步进法运用了一种称为打包或包装礼物的技术来计算一凸包。算法时间复杂度为O(nh)，其中h为凸包的顶点个数，当h为O(nlogn)时Jarvis步进法在渐近意义上比Graham扫描更快；
3. 凸包上按**逆时针**方向的连续顶点所构成的夹角只包含**左转**；
4. 若极角趋同则选择最远的；
5. 旋转卡壳算法求凸包直径时逆时针处理每条边时，最远点也是逆时针变化，为什么是O(n)？每一次都是在前面的基础上走几步，最后他至多走到起始位置，那么他走的步数肯定少于两圈。

**寻找最近点对**

1. “最近” 指的是通常意义下的欧得里距离。最近点对问题可以应用于交通控制系统中。为检测出潜在的碰撞事故，在空中或海洋交通控制系统中，需要识别届两个距离最近的交通工具；
2. 强调一下使用分法
3. 为什么可以达到O(nlogn)的时间复杂度，因为在第一次递归调用之前对所有点进行一次排序，有两个序列：一个是按照x值的大小升序进行排列的序列X；另一个是按照y值升序排列的序列Y；