判断线段相交

向量叉乘： a = (x1, y1), b = (x2, y2) a × b = x1*y2 - y1*x2 a×b,若结果小于0，表示向量b在向量a的顺时针方向；若结果大于0，表示向量b在向量a的逆时针方向；若等于0，表示向量a与向量b平行。(顺逆时针是指两向量平移至起点相连，从某个方向旋转到另一个向量小于180度)

两线段AB, CD相交的充要条件： 1.线段AB与CD所在的直线相交，即点A和点B分别在直线CD的两边；2.线段CD与AB所在的直线相交，即点C和点D分别在直线AB的两边；

一般情况：如果线段CD的两个端点C和D，与另一条线段的一个端点（A或B，只能是其中一个）连成的向量，与向量AB做叉乘，若结果异号，表示C和D分别在直线AB的两边，若结果同号，则表示CD两点都在AB的一边，则肯定不相交。 特殊情况：1.若有一点相交，则可能有一个值为0； 2.若两个线段在同一直线而相交，则一定有一个值为0； 但是当两条线段所在直线重合而没有交点的情况也是有一个值为0 改进：判断时加入0，同时剔除等于0的不合法情况，即两条线段所在直线重合而没有交点的情况(用矩形来判断)

```
struct Point
{
    int x, y;
};

struct Segment
{
    Point p, q;
};

int det(Point k1, Point k2, Point k3)        //向量k1k2叉乘k1k3
{
    return (k2.x - k1.x) * (k3.y - k1.y) - (k3.x - k1.x) * (k2.y - k1.y);
}

bool isIntersect(Segment k1, Segment k2)
{
    if(max(k1.p.x, k1.q.x) < min(k2.p.x, k2.q.x) || max(k2.p.x, k2.q.x) <
min(k1.p.x, k1.q.x) ||
       max(k1.p.y, k1.q.y) < min(k2.p.y, k2.q.y) || max(k2.p.y, k2.q.y) <
min(k1.p.y, k1.q.y))                                    //首先判断矩形
        return false;
    if(det(k1.p, k2.p, k1.q) * det(k1.p, k1.q, k2.q) >= 0 && det(k2.p, k1.p,
k2.q) * det(k2.p, k2.q, k1.q) >= 0)                          //判断叉乘结果
        return true;
    return false;
}
```