**作业二：实现二位点集凸包算法并可视化**

1. 作业要求
2. 实现凸包算法（二维点集）
3. 进行算法的可视化
4. 对不同规模的点数评估算法的实际复杂度（运算时间）并进行图示
5. 完成报告
6. Graham算法

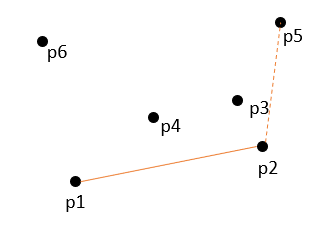
本次作业采用的凸包算法是Graham扫描算法。算法基本原理如下：

1、找出点集p[]中最左下的点p1，把p1同点集中其他各点用线段连接，并计算这些线段与水平线的夹角，然后按夹角从小到大和按到p1的距离从近到远排序,夹角范围为 [0, 180]度，而且可以删除相同夹角且距离p1较近的点，保留最远点，这样可减少计算量。因为直线上的非端点不是凸包的极点，即如果p1,p2,p3在一条直线上，则只取凸点p1,p3。p2不在端点，故可以去掉)，得到新的节点序列p1,p2,...pn。依次连接这些点，得到一个多边形(已经逆时针，有所进展，但还需去掉不在凸包上的点)。此时p1是凸包的边界起点，p2和pn也是最终凸包的顶点，p[n+1]=p1(看成循环的)

2、删除p3,p4,...p[n-1]中不在凸包上的点：

先把p1,p2,p3入栈S中，再依次循环(i=3->n-1)，若栈顶的两个点和当前的点p[i]这三点连线的方向向顺时针方向偏转，表明是凹的，应删除，则栈顶元素出栈(要循环判断，即可能前面的仍是凹的，还需再出栈，举例如下图)，直到向逆时针方向偏转或者栈内只有2个元素了(p1p2)，就把当前点p[i]入栈。

3、最后栈中的元素就是最终凸包上的点。



1. 作业成果
2. 将凸包算法过程进行可视化处理，如图3-1所示（gif见mht文件）。

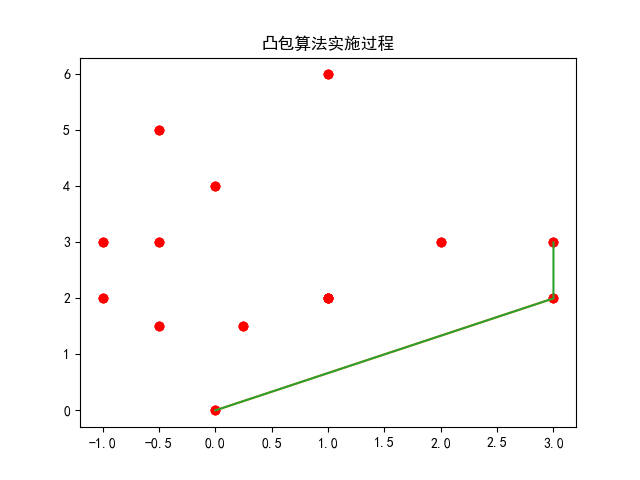
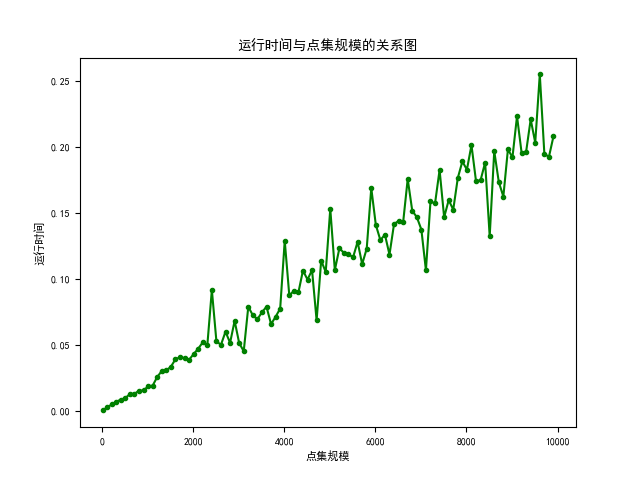


图3-1 凸包算法实施过程

1. 运行时间与点集规模关系如图3-2所示。



可以看出，点集规模越大，运行时间越长，呈现近线性的曲折上升。