

凸包問題

趙耀

凸包问题描述

- ▶ 凸包（Convex Hull）是一个计算几何（图形学）中的概念。
- ▶ 用直线段顺序连接平面上的点 P_1, P_2, \dots, P_n 构成的封闭曲线称为一个多边形， P_i 称为多边形的顶点，线段 P_iP_{i+1} 称为多边形的边。多边形中不相邻顶点之间的连线称为多边形的弦。如果多边形的任意一条弦均位于多边形内部，则称该多边形是凸多边形。平面上 n 个点 P_1, P_2, \dots, P_n 的凸包（convex hull）是指一个最小凸多边形，满足这 n 个点或者在多边形边上或者在其内。
- ▶ 在二维欧几里得空间中，凸包可想象为一条刚好包著所有点的橡皮圈。给定二维平面上的点集，凸包就是将最外层的点连接起来构成的凸多边形，它能包含点集中所有的点。

暴力算法1

- ▶ 定理1：给定平面点集 S ，如果 $P, P_i, P_j, P_k \in S$ 是4个不同的点且 P 位于 $\triangle P_i P_j P_k$ 的内部或边界上，则 P 不是 S 的凸包顶点。

如何判断点 P 是否在三角形 $\triangle P_i P_j P_k$ 的内部或边界上？

假设 $P_j P_k$ 确定的直线方程为 $L_{jk} = ax + by + c = 0$

假设 P 的坐标为 (x_p, y_p) ,

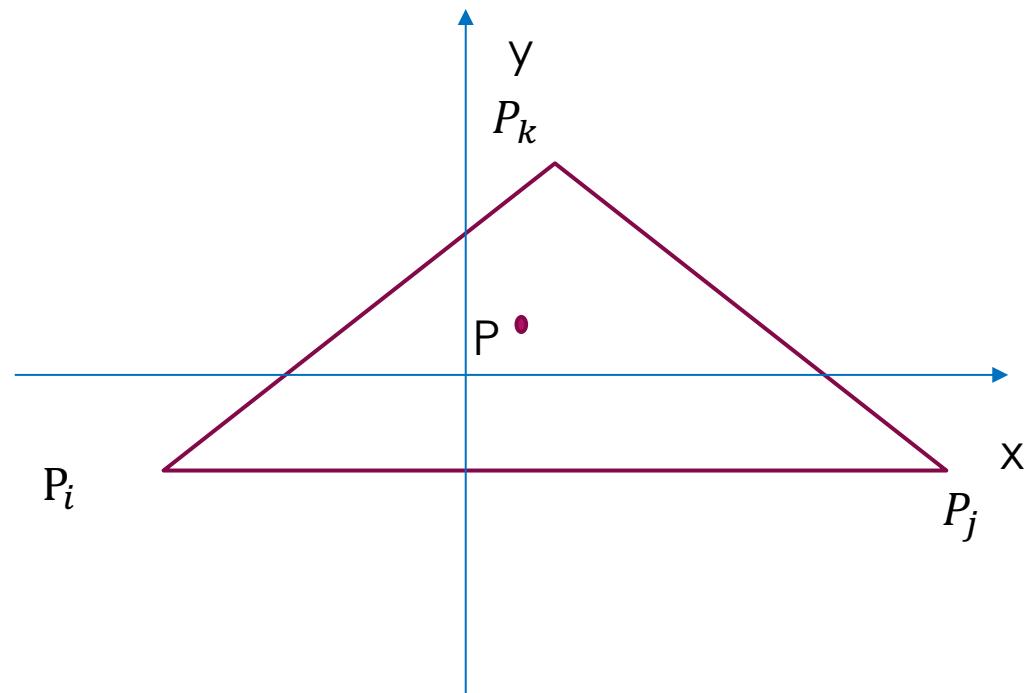
假设 P_i 的坐标为 (x_i, y_i)

如果 P 在 L_{jk} 的边界上，则有 $ax_p + by_p + c = 0$ ；

如果 P 在 $\triangle P_i P_j P_k$ 的内部，则 P 和 P_i 都在 L_{jk} 的同一边，则有

$$L_{jk}(P)L_{jk}(P_i) = (ax_p + by_p + c)(ax_i + by_i + c) > 0$$

如果 P 在 $\triangle P_i P_j P_k$ 的内部，对于每条边，上述的条件都是要满足的，所以要同时满足： $L_{jk}(P)L_{jk}(P_i) \geq 0$, $(L_{ij}P)L_{ij}(P_k) \geq 0$, $(L_{ik}P)L_{ik}(P_j) \geq 0$,



暴力算法1

If ($n = 3$) then 输出3

For (S 中任意的4个不同的点 P, P_i, P_j, P_k)

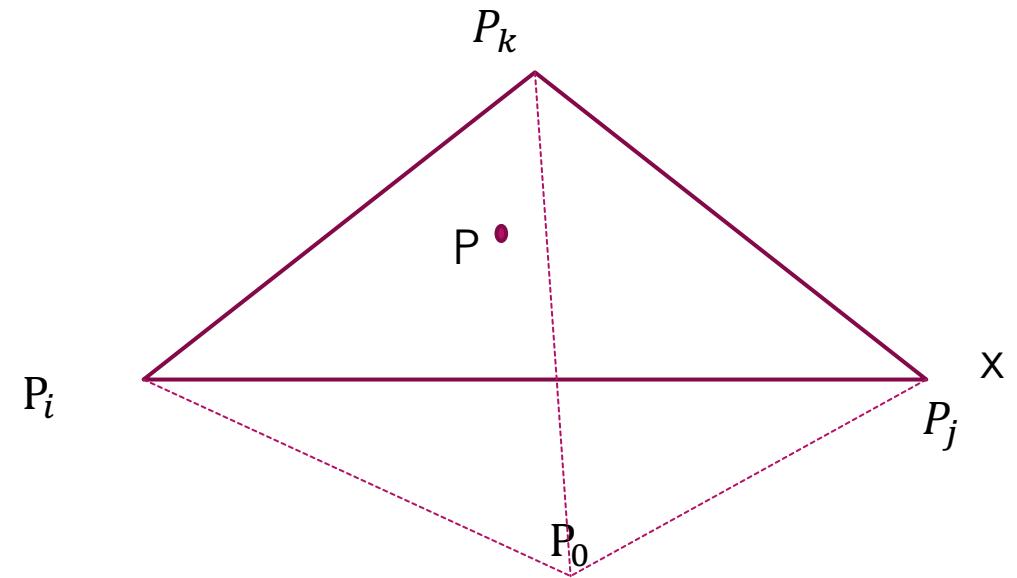
 if (P, P_i, P_j, P_k 中一个点位于其他3个点构成的三角形内)

 删除该点

输出 S 中剩下的点的个数

暴力算法2

- ▶ 定理2：给定平面点集 S ，设 $P_0 \in S$ 是一个凸包顶点。 S 中任意顶点不位于其中某3个顶点构成的三角形内或边界上，当且仅当 S 中任意顶点不位于 P_0 与 S 中其他两点构成的三角形或边界上。



暴力算法2

If ($n = 3$) then 输出3

找出纵坐标最小的点P0

For (S 中任意的3个不同的点 P_i, P_j, P_k)

 if (P_i, P_j, P_k 中一个点位于其他2个点与P0构成的三角形内)

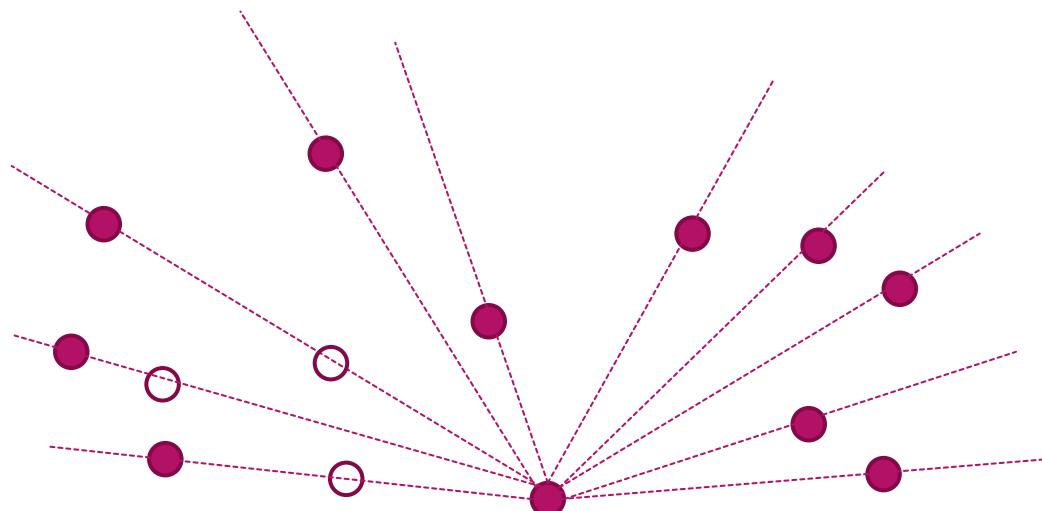
 删除该点

输出 S 中剩下的点的个数

GrahamScan算法

- ▶ Ronald Graham发现，确定P₀后，不需要枚举其他点的组合，通过极角定位找到最右侧的三角形 $\triangle P_0P_1P_2$ ，然后围绕P₀逆时钟旋转，按与P₀极角的从小到大的顺序依次找后续的点。

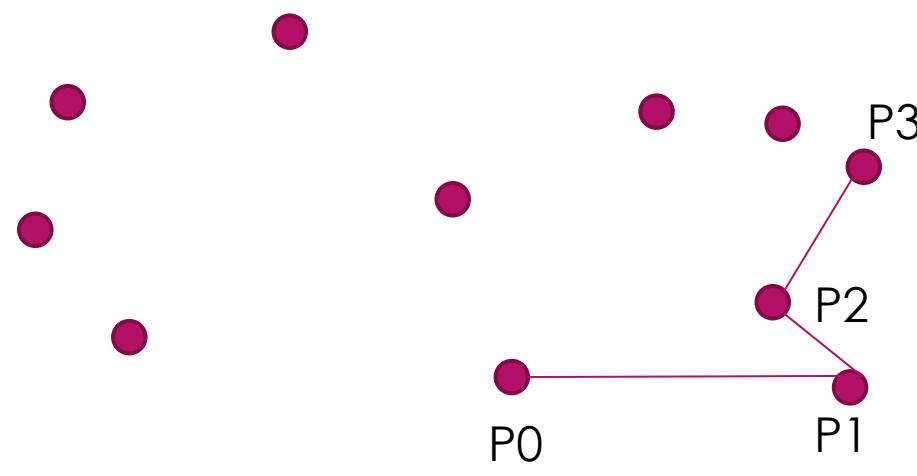
GrahamScan算法对点进行预处理



选定P0为纵坐标最小的点，其余点在以P0为极点、水平方向为极轴的极坐标系下按照极角从小到大排列。对于极角相同的点，只保留距离P0最远的点。
经过预处理，就可以排除一些点，如图中处理为空心的点。

算法用一个栈，假设为stack，来记录目前发现的可能为凸包点的点，每次拓展新的点都要维持该stack中的已有的所有点满足定理2。

算法执行



在引入P3之前，先判断P3P1P0形成的三角形，有没有把P2包含在内。

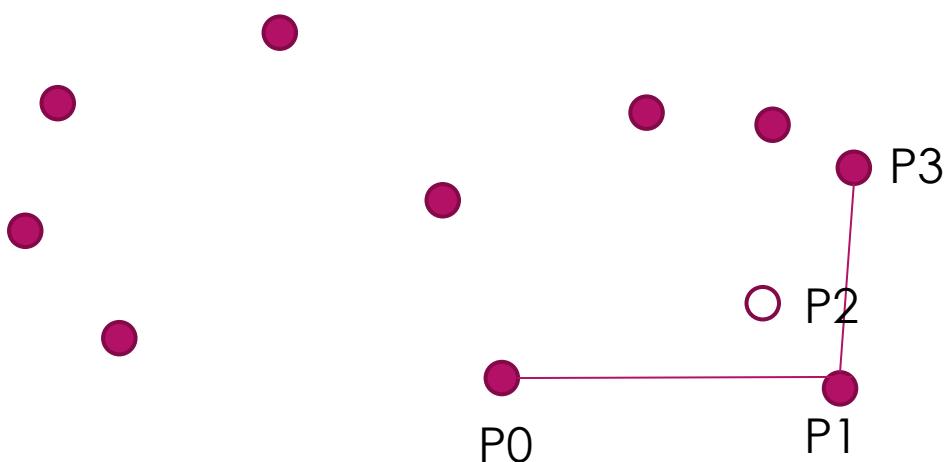
首先P2点和P3点肯定在P0P1这条线的同侧；
P1点和P2点也在P0P3这条线的同侧；（因为
P1P2P3是按照与P0的极角排序的）

最后只剩下一个判断，如果P0 P2 在P1P3的同侧，则表明P2被包含在P3P1P0这个三角形内部，不是凸包顶点，删除，把P3放进来
如果P0 P2 在P1P3的异侧，则表明P2仍然有可能是凸包顶点，保留并把P3放进来

P0 | P1 | P2 |

P3

算法执行



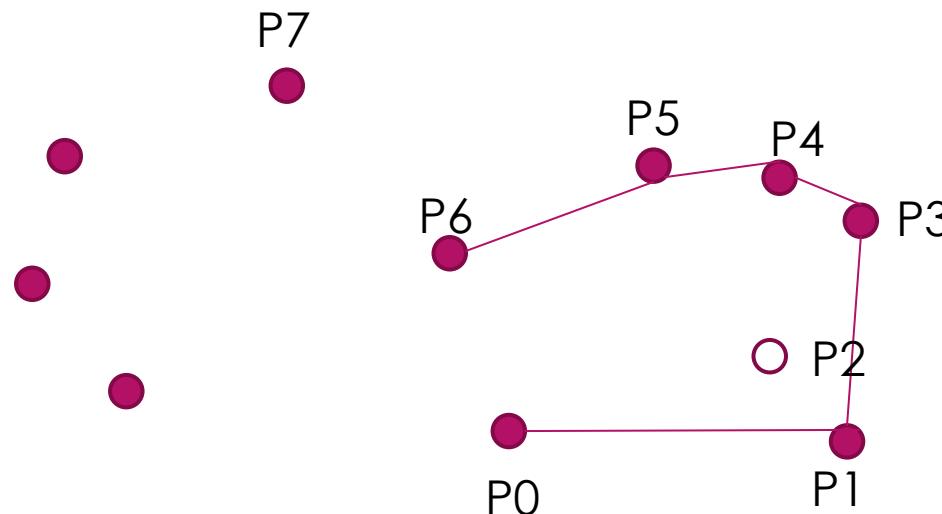
P0 | P1 | P3 |

P4

此例，异侧，删除P2之后放入P3

算法执行

P0 | P1 | P3 | P4 | P5 | P6 |

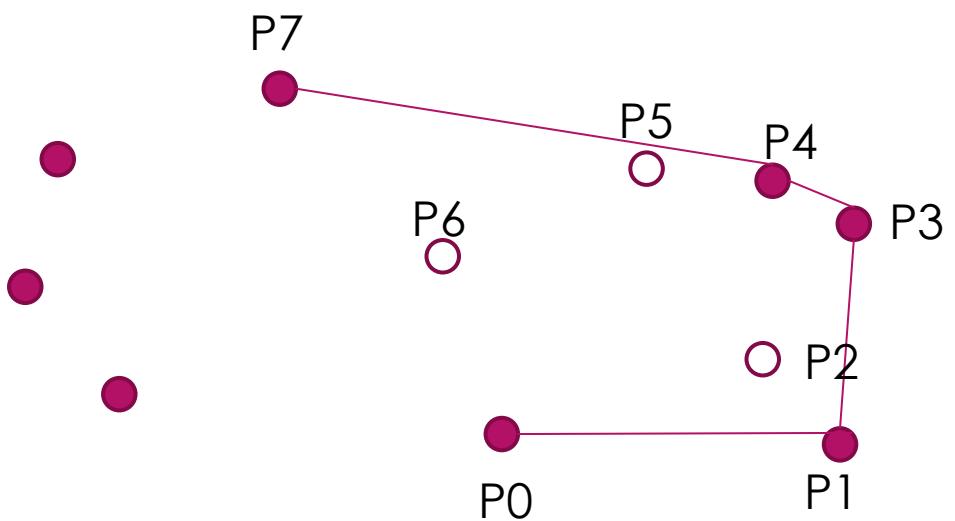
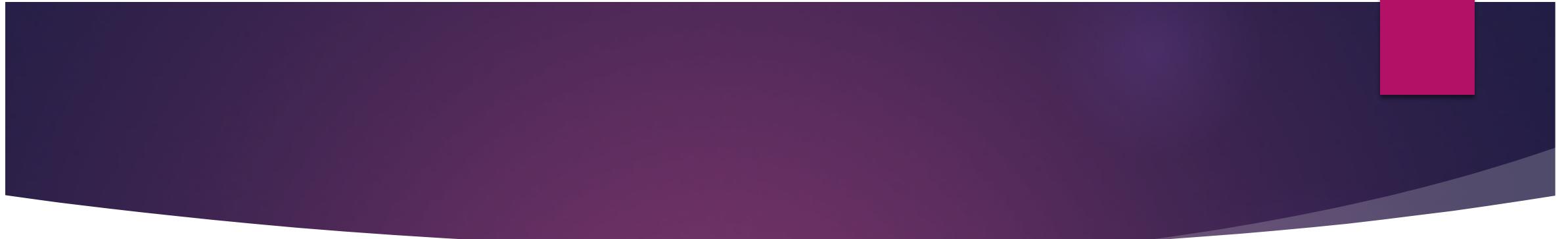


在引入P7之前，先判断P7P5P0形成的三角形，有没有把P6包含在内。

判断，如果P0 P6在P7P5的同侧，则表明P6被包含在P7P5P0这个三角形内部，不是凸包顶点，删除P6

删除P6之后，继续判断P5，如果P0 P5在P7P4的同侧，则表明P5被包含在P7P4P0这个三角形内部，不是凸包顶点，删除P5

删除P5之后，继续判断P4,直到top点仍然为当前可能的凸包点为止



P0	P1	P3	P4	P7		
-----------	-----------	-----------	-----------	-----------	--	--