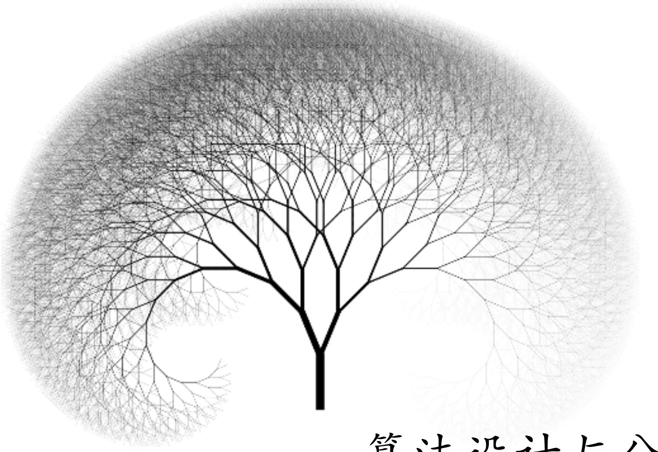


算法设计与分析

刘 安

苏州大学 计算机科学与技术学院

http://web.suda.edu.cn/anliu/

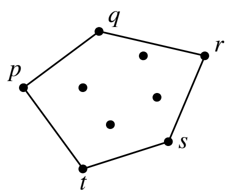
分治算法

- 分解：将原问题分解为一组子问题，子问题与原问题类似，但是规模更小
- 解决：递归求解子问题，如果子问题足够小，停止递归，直接求解
- 合并：将子问题的解组合成原问题的解

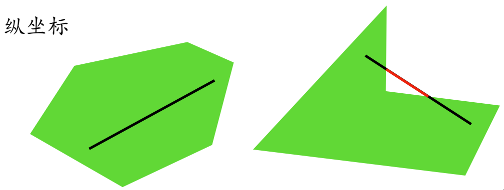
凸包

Convex Hull

基本概念

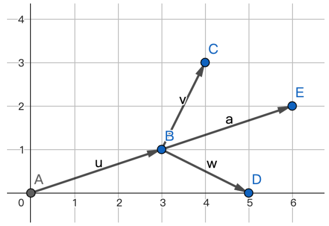


- 凸多边形P：连接P中任意两点的线段全部在P中
- 平面上某个点集的凸包
  - 包含所有点的最小凸多边形
- 如何表示凸包：最小凸多边形的顶点（或边）
  - 集合（无序）
  - 序列（有序）：按顺时针方向形成的序列：p, q, r, s, t
- 问题：给定二维平面上的n个点$S = \{(x_i, y_i) \mid i = 1,2,\cdots,n\}$，找到其凸包
  - 假设没有两点具有相同的横坐标
  - 假设没有两点具有相同的纵坐标
  - 假设没有三点共线



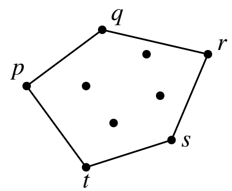
点和线段

- 向量$P_1 = (x_1, y_1)$和$P_2 = (x_2, y_2)$的叉积$P_1 \times P_2 = x_1y_2 - x_2y_1$
- 将向量P_2置于P_1之后，P_2相对于P_1是向左转还是向右转由它们的叉积决定
  - 大于0，左转：$u \times v = 3 \cdot 2 - 1 \cdot 1 = 5$
  - 等于0，同一直线上：$u \times a = 3 \cdot 1 - 1 \cdot 3 = 0$
  - 小于0，右转：$u \times w = 3 \cdot (-1) - 1 \cdot 2 = -5$
- 给定线段AB和点C，判断C与AB的位置关系
  - 计算叉积：$(x_B - x_A, y_B - y_A) \times (x_C - x_B, y_C - y_B)$
    - 大于0：C在$\overrightarrow{AB}$的左侧
    - 等于0：C在$\overrightarrow{AB}$所在的直线上
    - 小于0：C在$\overrightarrow{AB}$的右侧



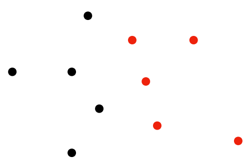
穷举法求凸包

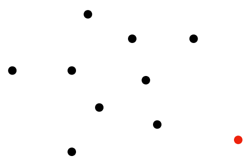
- 凸包：最小凸多边形的顶点或边
- 对点穷举
  - 对于每一个点，检查其是否在任意其它三点构成的三角形中
  - $O(n)$个点，$O(n^3)$个三角形，检查需要$O(1)$时间
  - 总的时间复杂度：$O(n^4)$
- 对边穷举
  - 对于任意两点构成的线段，检查其是否属于凸包
    - 属于：所有其它的点都在该线段的一侧
  - $O(n^2)$条线段，测试需要$O(n)$时间
  - 总的时间复杂度：$O(n^3)$



分治法求凸包

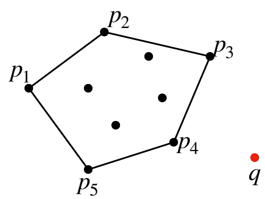
- 分解：把n个点分成A、B两部分
  - A有$n/2$个点，B有$n/2$个点
  - A有$n-1$个点，B有1个点
  - 还没有别的分解方法？
- 递归求解$<n$个点的凸包
  - 基本情况：直接求解
- 合并
  - 合并2个凸包
  - 合并一个凸包和一个点





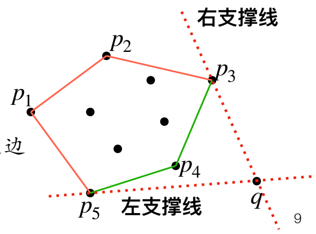
第一种分治法 - 插入凸包

- 分解：把n个点分成A、B两部分，其中A有$n-1$个点，B只有1个点
  - 选取一个特别的点q放入B：最右边的点（横坐标最大的点）
  - 该点一定属于新的凸包
- 递归求解A的凸包$CH(A)$
  - 基本情况：三个点，直接计算
- 合并$CH(A)$和q
- 算法时间复杂度：$T(n) = T(n-1) + T_{merge}$
  - $T_{merge} = O(n^2) \Rightarrow T(n) = O(n^3)$
  - $T_{merge} = O(n) \Rightarrow T(n) = O(n^2)$
  - $T_{merge} = O(\log n) \Rightarrow T(n) = O(n \log n)$



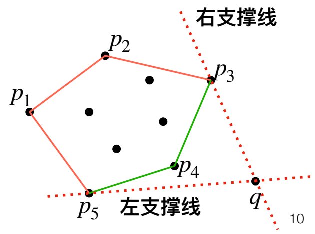
合并点和凸包

- 凸包的支撑线：与凸包仅相交于一点的直线
- 过凸包外一点有且仅有两条该凸包的支撑线（qp_3和qp_5）
- 两个交点（p_3和p_5）将凸包边界分成两个链：近链和远链
- 新的凸包由远链和点q决定
  - 从左支撑线开始，在凸包边界上顺时针前进直到右支撑线
- 如何求左右支撑线
  - $O(n)$条候选支撑线qp_i
    - 检查所有其他点是否在qp_i同一侧：$O(n)$时间
  - 共需$O(n^2)$时间
- 观察
  - 左支撑线与凸包的交点一定在所有其它qp_i的左边



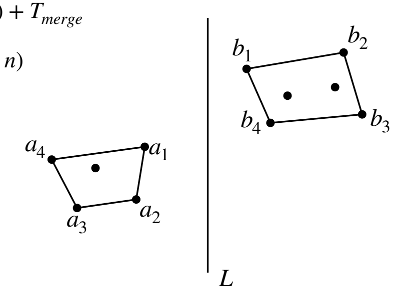
寻找左右支撑线

- 寻找左支撑线
  - 从距离q最近的p_i开始，检查$p_{(i-1)\%n}$和$p_{(i+1)\%n}$是否都在qp_i右侧
- 寻找右支撑线
  - 从距离q最近的p_i开始，检查$p_{(i-1)\%n}$和$p_{(i+1)\%n}$是否都在qp_i左侧
- 寻找左右支撑线的时间复杂度：$O(n)$
- $T(n) = T(n-1) + O(n)$
  - $\Rightarrow T(n) = O(n^2)$



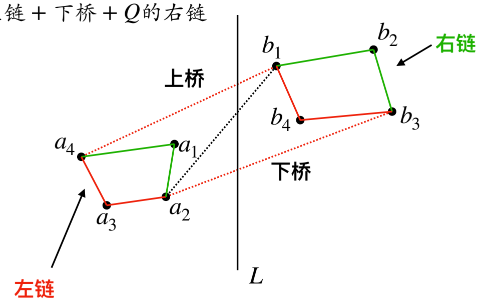
第二种分治法 - 归并凸包

- 分解：把n个点分成A、B两部分，其中A、B各有$n/2$个点
  - 预处理：将所有点按横坐标排序
- 递归求解A和B的凸包，记为P和Q
  - 基本情况：是不是三个点？
- 合并P和Q
- 算法时间复杂度：$T(n) = 2T(n/2) + T_{merge}$
  - $T_{merge} = O(n) \Rightarrow T(n) = O(n \log n)$



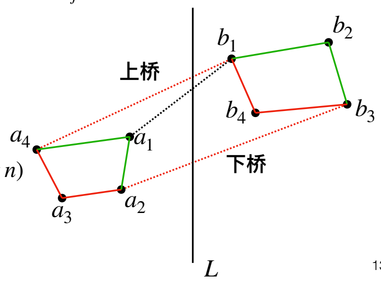
合并两个凸包

- 凸包P和Q的桥：既是P的支撑线，也是Q的支撑线（比如a_4b_1和a_2b_1）
- 上桥：如果P和Q都在桥的下方
- 下桥：如果P和Q都在桥的上方
- 找到P和Q的上桥和下桥
  - 每个凸包边界被其与桥的交点分成左链和右链
- 合并后的凸包 = 上桥 + P的左链 + 下桥 + Q的右链
- 上桥a_4b_1
  - 对b_1来说是右支撑线
  - 对a_4来说是左支撑线



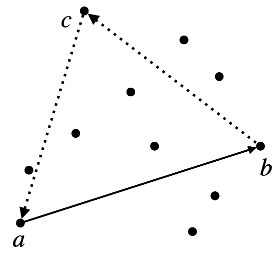
寻找上桥和下桥

- 令a_i是P中距离L最近的点，b_j是Q中距离L最近的点
- 如果a_ib_j不是a_i的左支撑线，找到a_i的左支撑线a_ib_j
  - 顺时针检查b_j的下一个点
- 如果a_ib_j不是b_j的右支撑线，找到b_j的右支撑线a_ib_j
  - 逆时针检查a_i的下一个点
- 重复上述步骤直到a_ib_j既是a_i的左支撑线，又是b_j的右支撑线，即为上桥
- 下桥类似求解
- 寻找上桥和下桥时间复杂度：$O(n)$
- 合并时间复杂度：$O(n)$
- $T(n) = 2T(n/2) + O(n) \Rightarrow T(n) = O(n \log n)$



第三种分治法 - 快速凸包

- 选择两个极端点a和b（一定属于凸包）
  - $\overrightarrow{ab}$将整个点集分成两部分：$\overrightarrow{ab}$右边的点集B、$\overrightarrow{ba}$右边的点集A
  - A中哪些点一定属于凸包？
    - 距离$\overrightarrow{ba}$最远的点c：$O(1)$时间可确定c到$\overrightarrow{ba}$的距离
    - $\triangle abc$中的点一定不属于凸包
    - $\triangle abc$外的点？
      - $\overrightarrow{bc}$右边的点：递归求解哪些点属于凸包
      - $\overrightarrow{ca}$右边的点：递归求解哪些点属于凸包
  - B中哪些点一定属于凸包？



quick_hull

quick_hull(S)

$(a, b) \leftarrow$ extreme_points(S)

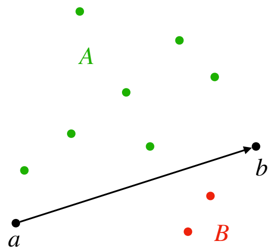
$A \leftarrow$ right_of(S, $\overrightarrow{ba}$)

$B \leftarrow$ right_of(S, $\overrightarrow{ab}$)

$Q_A \leftarrow$ quick_half_hull(A, $\overrightarrow{ba}$)

$Q_B \leftarrow$ quick_half_hull(B, $\overrightarrow{ab}$)

return $\{a\} \cup Q_A \cup \{b\} \cup Q_B$



quick_half_hull

quick_half_hull(S, $\overrightarrow{ba}$)

if ($s = \varnothing$) return $\varnothing$

$c \leftarrow$ furthest(S, $\overrightarrow{ba}$)

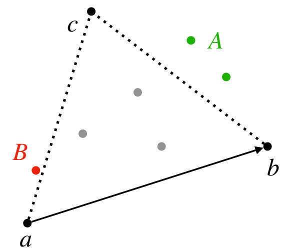
$A \leftarrow$ right_of(S, $\overrightarrow{bc}$)

$B \leftarrow$ right_of(S, $\overrightarrow{ca}$)

$Q_A \leftarrow$ quick_half_hull(A, $\overrightarrow{bc}$)

$Q_B \leftarrow$ quick_half_hull(B, $\overrightarrow{ca}$)

return $Q_A \cup \{c\} \cup Q_B$



时间复杂度分析

- 令 $|S| = n, |A| = \alpha, |B| = \beta$，$\alpha + \beta \le n - 1$
- quick_half_hull的运行时间$T(n) = T(\alpha) + T(\beta) + O(n)$
  - $\alpha = \beta = n/2 \Rightarrow T(n) = O(n \log n)$
  - $\alpha = 0, \beta = n - 1 \Rightarrow T(n) = O(n^2)$
- quick_hull的运行时间
  - 预排序计算极端点：$O(n \log n)$
  - 计算A和B：$O(n)$
  - 递归求解A和B：$< 2T(n)$
  - 最坏情况下也是$O(n^2)$