

选最大与最小

选择问题

输入：集合 L (含 n 个不等的实数)

输出： L 中第 i 小元素

$i=1$, 称为最小元素

$i=n$, 称为最大元素

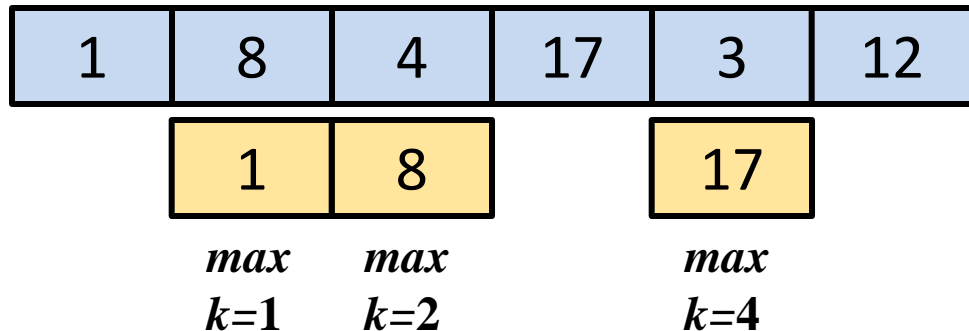
位置处在中间的元素，称为中位元素

n 为奇数，中位数唯一， $i = (n+1)/2$

n 为偶数，可指定 $i = n/2+1$

选最大

算法：顺序比较



输出： $max = 17$, $k=4$

算法最坏情况下的时间 $W(n)=n-1$

伪码

算法 Findmax

输入: n 个数的数组 L

输出: max, k

1. $max \leftarrow L[1]$

2. for $i \leftarrow 2$ to n do

3. if $max < L[i]$

4. then $max \leftarrow L[i]$

5. $k \leftarrow i$

6. return max, k

选最大最小

通常算法:

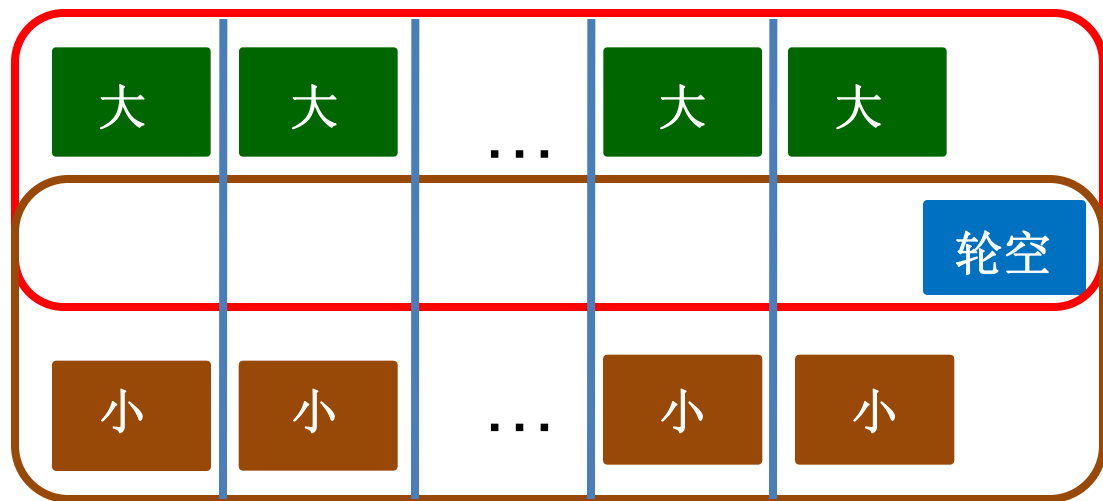
1. 顺序比较, 先选最大 max
2. 顺序比较, 在剩余数组中选最小 min , 类似于选最大算法, 但比较时保留较小的数

时间复杂性:

$$W(n) = n-1 + n-2 = 2n-3$$

分组算法

组1 组2 ... 组 $\lfloor n/2 \rfloor$



伪码

算法 FindMaxMin

输入： n 个数的数组 L

输出： max , min

1. 将 n 个元素两两一组分成 $\lfloor n/2 \rfloor$ 组
2. 每组比较，得到 $\lfloor n/2 \rfloor$ 个较小和 $\lfloor n/2 \rfloor$ 个较大
3. 在 $\lceil n/2 \rceil$ 个较大（含轮空元素）中找最大 max
4. 在 $\lceil n/2 \rceil$ 个较小（含轮空元素）中找最小 min

最坏情况时间复杂度

行2 的组内比较: $\lfloor n/2 \rfloor$ 次

行3--4 求 max 和 min 比较:

至多 $2\lceil n/2 \rceil - 2$ 次

$$W(n) = \lfloor n/2 \rfloor + 2\lceil n/2 \rceil - 2$$

$$= n + \lceil n/2 \rceil - 2$$

$$= \lceil 3n/2 \rceil - 2$$

分治算法

1. 将数组 L 从中间划分为两个子数组 L_1 和 L_2
2. 递归地在 L_1 中求最大 max_1 和 min_1
3. 递归地在 L_2 中求最大 max_2 和 min_2
4. $max \leftarrow \max\{ max_1, max_2 \}$
5. $min \leftarrow \min\{ min_1, min_2 \}$

最坏情况时间复杂度

假设 $n = 2^k$,

$$W(n) = 2W(n/2) + 2$$

$$W(2) = 1$$

解
$$\begin{aligned} W(2^k) &= 2W(2^{k-1}) + 2 \\ &= 2[2W(2^{k-2}) + 2] + 2 \\ &= 2^2W(2^{k-2}) + 2^2 + 2 = \dots \\ &= 2^{k-1} + 2^{k-1} + \dots + 2^2 + 2 \\ &= 3 \cdot 2^{k-1} - 2 = 3n/2 - 2 \end{aligned}$$

选择算法小结

选最大：顺序比较, 比较次数 $n-1$

选最大最小

- 选最大+ 选最小, 比较次数 $2n-3$
- 分组： 比较次数 $\lceil 3n/2 \rceil - 2$
- 分治： $n=2^k$, 比较次数 $3n/2-2$

选第二大

选第二大

输入： n 个数的数组 L

输出： 第二大的数 $second$

通常算法： 顺序比较

1. 顺序比较找到最大 max
2. 从剩下 $n - 1$ 个数中找最大，就是第二大 $second$

时间复杂度：

$$W(n) = n - 1 + n - 2 = 2n - 3$$

提高效率的途径

- 成为第二大数的条件：仅在与最大数的比较中被淘汰.
- 要确定第二大数，必须知道最大数.
- 在确定最大数的过程中记录下被最大数直接淘汰的元素.
- 在上述范围（被最大数直接淘汰的数）内的最大数就是第二大数.
- 设计思想： 用空间换时间.

锦标赛算法

1. 两两分组比较，大者进入下一轮，直到剩下 1 个元素 *max* 为止
2. 在每次比较中淘汰较小元素，将被淘汰元素记录在淘汰它的元素的链表上
3. 检查 *max* 的链表，从中找到最大元，即 *second*

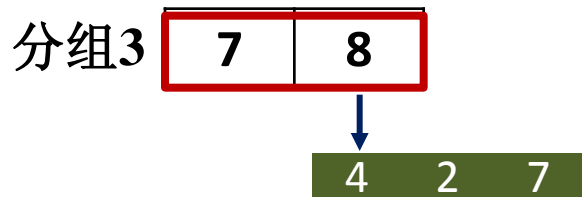
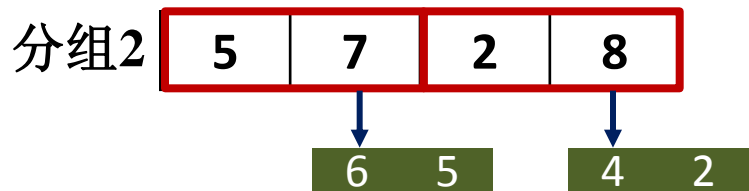
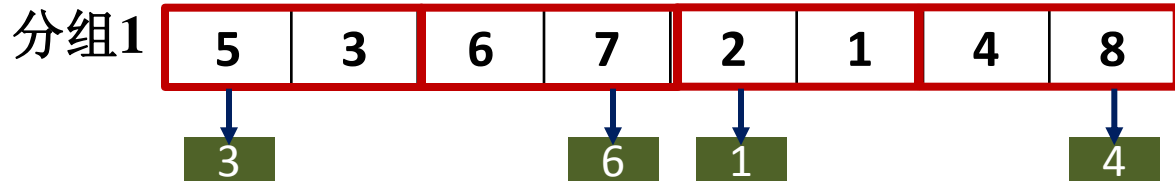
伪码

算法 FindSecond

输入: n 个数的数组 L , 输出: $second$

1. $k \leftarrow n$ // 参与淘汰的元素数
 2. 将 k 个元素两两1组, 分成 $\lfloor k/2 \rfloor$ 组
 3. 每组的2个数比较, 找到较大数
 4. 将被淘汰数记入较大数的链表
 5. if k 为奇数 then $k \leftarrow \lfloor k/2 \rfloor + 1$
 6. else $k \leftarrow \lfloor k/2 \rfloor$
 7. if $k > 1$ then goto 2
 8. $max \leftarrow$ 最大数
 9. $second \leftarrow max$ 的链表中的最大
- 一轮淘汰
- 继续分组淘汰

实例



时间复杂度分析

命题1 设参与比较的有 t 个元素，经过 i 轮淘汰后元素数至多为 $\lceil t/2^i \rceil$.

证 对 i 归纳. $i=1$, 分 $\lfloor t/2 \rfloor$ 组，淘汰 $\lfloor t/2 \rfloor$ 个元素，进入下一轮元素数是 $t - \lfloor t/2 \rfloor = \lceil t/2 \rceil$

假设 i 轮分组淘汰后元素数至多为 $\lceil t/2^i \rceil$ ，那么 $i+1$ 轮分组淘汰后元素数为

$$\lceil \lceil t/2^i \rceil / 2 \rceil = \lceil t/2^{i+1} \rceil$$

时间复杂度分析（续）

命题2 max 在第一阶段分组比较中总计进行了 $\lceil \log n \rceil$ 次比较.

证 假设到产生 max 时总计进行 k 轮淘汰, 根据命题 1 有 $\lceil n/2^k \rceil = 1$.

若 $n=2^d$, 那么有

$$k = d = \log n = \lceil \log n \rceil$$

若 $2^d < n < 2^{d+1}$, 那么

$$k = d + 1 = \lceil \log n \rceil$$

时间复杂度分析（续）

第一阶段元素数： n

比较次数： $n-1$

淘汰了 $n-1$ 个元素

第二阶段：元素数 $\lceil \log n \rceil$

比较次数： $\lceil \log n \rceil - 1$

淘汰元素数为 $\lceil \log n \rceil - 1$

时间复杂度是

$$\begin{aligned} W(n) &= n - 1 + \lceil \log n \rceil - 1 \\ &= n + \lceil \log n \rceil - 2. \end{aligned}$$

小结

求第二大算法

- 调用2次找最大: $2n-3$
 - 锦标赛算法: $n + \lceil \log n \rceil - 2$
- 主要的技术: 用空间换时间

一般选择问题 的算法设计

一般性选择问题

问题：选第 k 小.

输入：数组 S , S 的长度 n , 正整数 k ,
 $1 \leq k \leq n$.

输出：第 k 小的数

实例 1

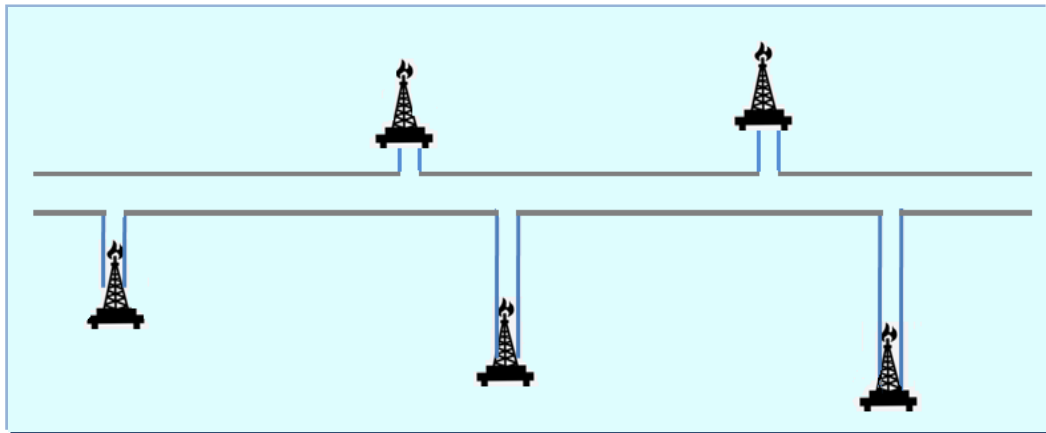
$S = \{ 3, 4, 8, 2, 5, 9, 18 \}$, $k = 4$, 解: 5

实例 2

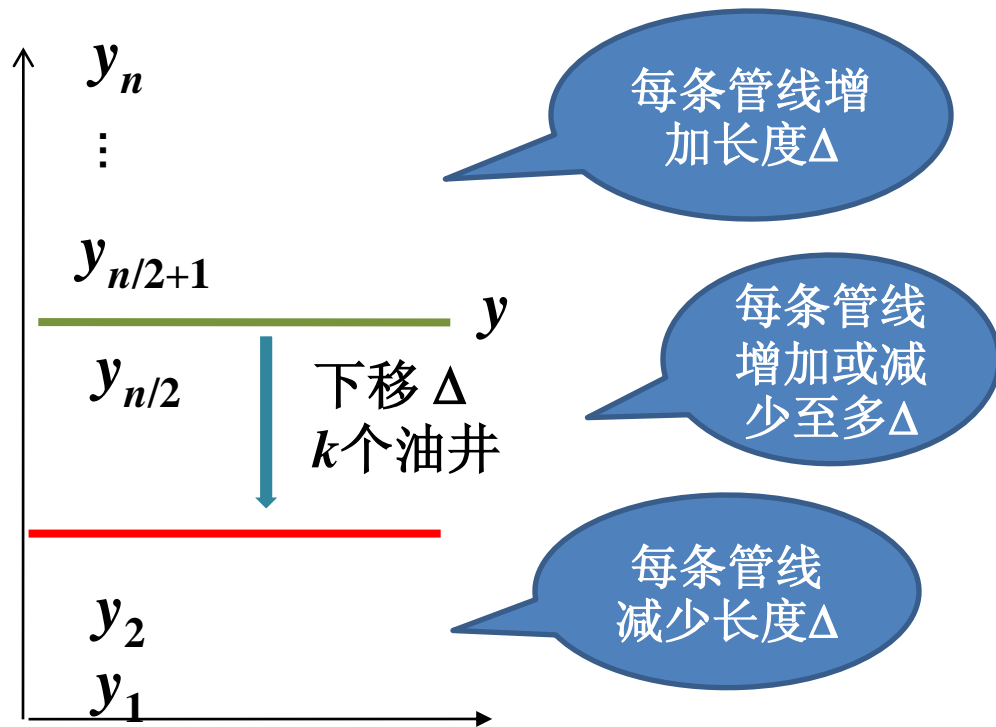
统计数据的集合 S , $|S|=n$,
选中位数, $k = \lceil n/2 \rceil$

一个应用：管道位置

问题：某区域有 n 口油井，需要修建输油管道. 根据设计要求，水平方向有一条主管道，每口油井修一条垂直方向的支管道通向主管道. 如何选择主管道的位置，以使得支管道长度的总和最小？



最优解: Y 坐标的中位数



下移后支管线总长度增加

简单的算法

算法一：

调用 k 次选最小算法

时间复杂度为 $O(kn)$

算法二：

先排序，然后输出第 k 小的数

时间复杂度为 $O(n \log n)$

分治算法

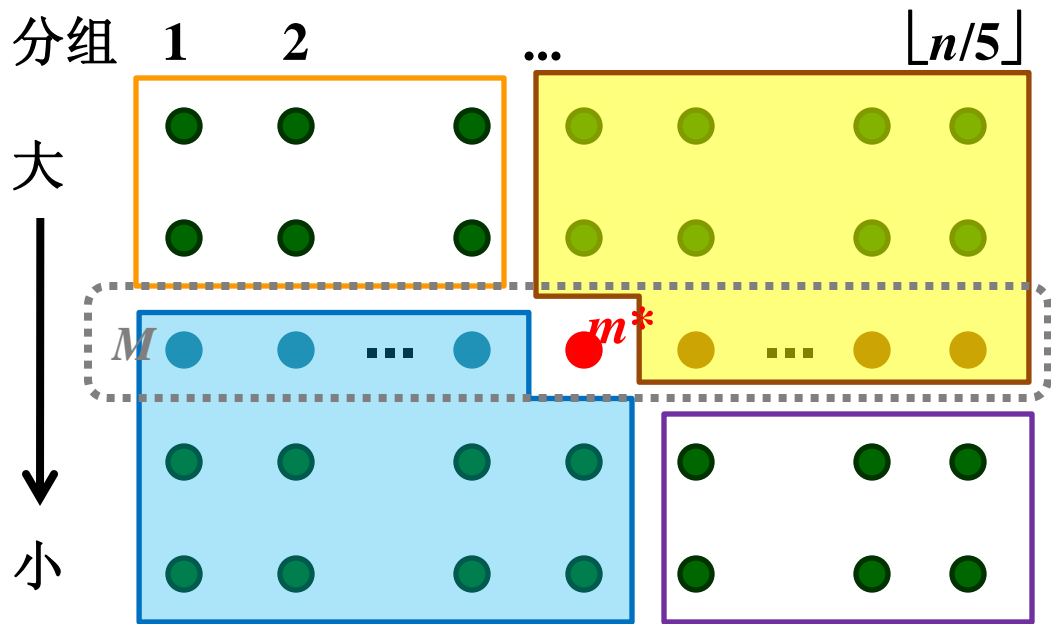
假设元素彼此不等，设计思想：

1. 用**某个元素 m^*** 作为标准将 S 划分成 S_1 与 S_2 ，其中 S_1 的元素小于 m^* ， S_2 的元素大于等于 m^* 。
2. 如果 $k \leq |S_1|$ ，则在 S_1 中找第 k 小。
如果 $k = |S_1| + 1$ ，则 m^* 是第 k 小
如果 $k > |S_1| + 1$ ，则在 S_2 中找第 $k - |S_1| - 1$ 小



算法效率取决于子问题规模，
如何通过 m^* 控制子问题规模？

m^* 的选择与划分过程



A: 数需要与 m^* 比大小, **B**: 数大于 m^*

C: 数小于 m^* , **D**: 数需要与 m^* 比大小

实例: $n=15, k=6$

8	2	3	5	7	6	11	14	1	9	13	10	4	12	15
---	---	---	---	---	---	----	----	---	---	----	----	---	----	----

	8	14	15	
	7	11	13	
M	5	9	12	$m^* = 9$
	3	6	10	
	2	1	4	

	8	14	15	
A	7	11	13	B
	5	9	12	
C	3	6	10	D
	2	1	4	

8, 7, 10, 4 需要与9比较

归约为子问题

S_1	8	14	15	S_2
	7	11	13	
	5	9	12	
	3	6	10	
	2	1	4	

子问题

8 7 5 3 2 6 1 4

子问题规模 = 8, $k=6$

伪码

算法 Select (S, k)

输入：数组 S ，正整数 k ，

输出： S 中的第 k 小元素

1. 将 S 分 5 个一组，共 $n_M = \lceil n/5 \rceil$ 组
2. 每组排序，中位数放到集合 M
3. $m^* \leftarrow \text{Select}(M, \lceil |M|/2 \rceil)$ // S 分 A, B, C, D
4. A, D 元素小于 m^* 放 S_1 , 大于 m^* 放 S_2
5. $S_1 \leftarrow S_1 \cup C; S_2 \leftarrow S_2 \cup B$ 划分
6. if $k = |S_1| + 1$ then 输出 m^*
7. else if $k \leq |S_1|$ \Leftarrow 递归计算子问题
8. then Select (S_1, k)
9. else Select ($S_2, k - |S_1| - 1$)

小结

选第 k 小的算法:

- 分治策略
- 确定 m^*
- 用 m^* 划分数组归约为子问题
- 递归实现

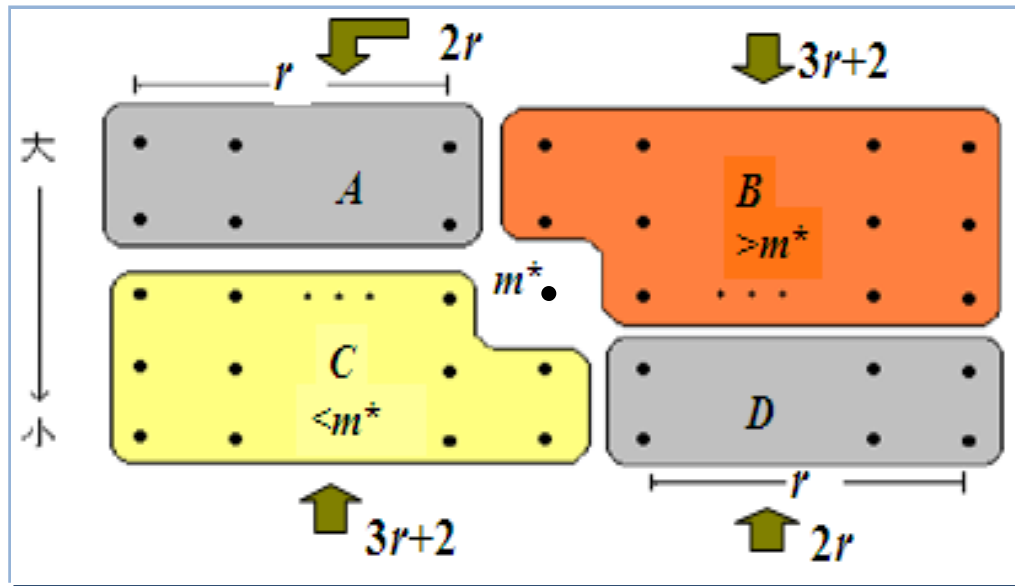
选择问题的 算法分析

伪码

算法 Select (S, k)

1. 将 S 分5个一组, 共 $n_M = \lceil n/5 \rceil$ 组
2. 每组排序, 中位数放到集合 M
3. $m^* \leftarrow \text{Select}(M, \lceil |M|/2 \rceil)$ // S 分 A, B, C, D
4. A, D 元素小于 m^* 放 S_1 , 大于 m^* 放 S_2
5. $S_1 \leftarrow S_1 \cup C; S_2 \leftarrow S_2 \cup B$
6. if $k = |S_1| + 1$ then 输出 m^*
7. else if $k \leq |S_1|$
8. then Select (S_1, k)
9. else Select ($S_2, k - |S_1| - 1$)

用 m^* 划分



$$n = 5(2r + 1), \quad |A| = |D| = 2r$$

子问题规模至多: $2r + 2r + 3r + 2 = 7r + 2$

子问题规模估计

不妨设 $n = 5(2r + 1)$, $|A|=|D|=2r$,

$$r = \frac{n/5 - 1}{2} = \frac{n}{10} - \frac{1}{2}$$

划分后子问题规模至多为

$$\begin{aligned} \underline{7r + 2} &= 7\left(\frac{n}{10} - \frac{1}{2}\right) + 2 \\ &= \frac{7n}{10} - \frac{3}{2} < \frac{7n}{10} \end{aligned}$$

时间复杂度递推方程

算法工作量 $W(n)$

行2: $O(n)$ //每5个数找中位数,构成 M

行3: $W(n/5)$ // M 中找中位数 m^*

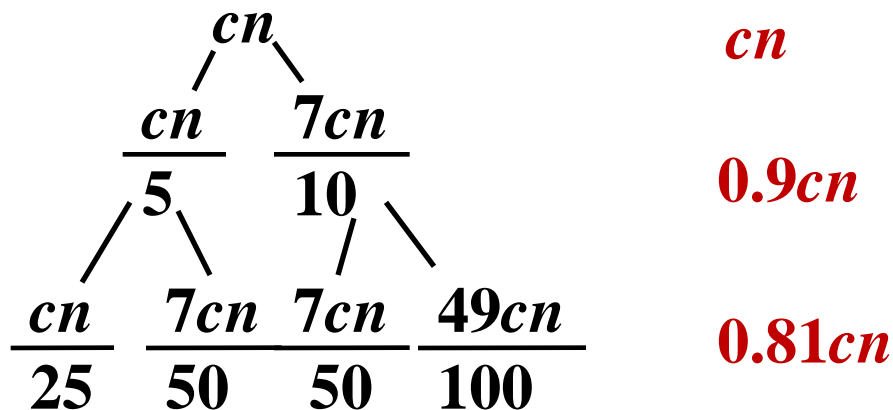
行4: $O(n)$ // 用 m^* 划分集合 S

行8-9: $W(7n/10)$ //递归

$$W(n) \leq W(n/5) + W(7n/10) + O(n)$$

递归树

$$W(n) = W(n/5) + W(7n/10) + cn$$



.....

$$W(n) \leq cn (1 + 0.9 + 0.9^2 + \dots) = O(n)$$

讨论



分组时为什么5个元素一组？

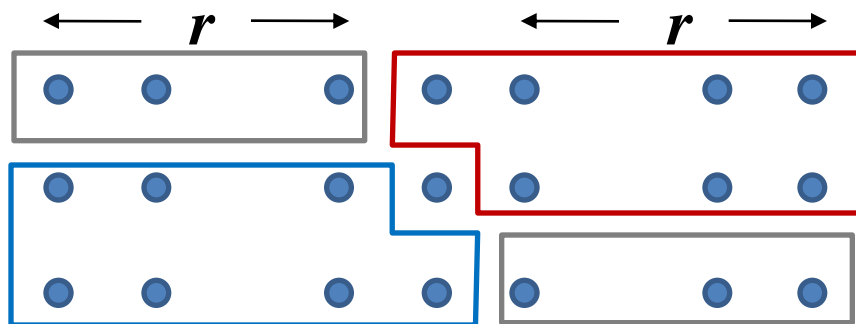
3个一组或 7个一组行不行？

分析：递归调用

1. 求 m^* 的工作量与 $|M| = n/t$ 相关, t 为每组元素数. t 大, $|M|$ 小
2. 归约后子问题大小与分组元素数 t 有关. t 大, 子问题规模大

3分组时的子问题规模

假设 $t=3$, 3个一组:



$$n = 3(2r + 1)$$

$$r = (n/3 - 1)/2 = n/6 - 1/2$$

子问题规模最多为 $4r+1 = 4n/6 - 1$

算法的时间复杂度

算法的时间复杂度满足方程

$$W(n) = W(n/3) + W(4n/6) + cn$$

由递归树得 $W(n) = \Theta(n \log n)$

关键：

$|M|$ 与归约后子问题规模之和小于 n ，
递归树每行的工作量构成公比小于 1
的等比级数， 算法复杂度才是 $O(n)$ 。

小结

选第 k 小算法的时间分析

- 递推方程
- 分组时每组元素数的多少对时间复杂度的影响

卷积及其应用

向量计算

给定向量 $a = (a_0, a_1, \dots, a_{n-1})$
 $b = (b_0, b_1, \dots, b_{n-1})$

向量和 $a+b = (a_0+b_0, a_1+b_1, \dots, a_{n-1}+b_{n-1})$

内积 $a \cdot b = a_0b_0 + a_1b_1 + \dots + a_{n-1}b_{n-1}$

卷积 $a*b = (c_0, c_1, \dots, c_{2n-2})$, 其中

$$c_k = \sum_{\substack{i+j=k \\ i,j < n}} a_i b_j, \quad k = 0, 1, \dots, 2n-2$$

卷积的含义

在下述矩阵中，每个斜线的项之和恰好是卷积中的各个分量

$$\begin{array}{ccccccc}
 & & ab_0 & ab_1 & \cdots & ab_{n-2} & ab_{n-1} \\
 a_0b & \cancel{a_0b_0} & a_0b_1 & \cdots & a_0b_{n-2} & \cancel{a_0b_{n-1}} & \\
 a_1b & \cancel{a_1b_0} & a_1b_1 & \cdots & \cancel{a_1b_{n-2}} & a_1b_{n-1} & \\
 \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\
 a_{n-2}b & \cancel{a_{n-2}b_0} & \cancel{a_{n-2}b_1} & \cdots & a_{n-2}b_{n-2} & \cancel{a_{n-2}b_{n-1}} & \\
 a_{n-1}b & \cancel{a_{n-1}b_0} & \cancel{a_{n-1}b_1} & \cdots & \cancel{a_{n-1}b_{n-2}} & \cancel{a_{n-1}b_{n-1}} &
 \end{array}$$

c_0 c_1 c_{n-1} c_{2n-3} c_{2n-2}

计算实例

向量 $a = (1, 2, 4, 3), \quad b = (4, 2, 8, 0)$

则 $a+b = (5, 4, 12, 3)$

$a \cdot b = (4, 4, 32, 0)$

$a * b = (4, 10, 28, 36, 38, 24, 0)$

	ab_0	ab_1	ab_2	ab_3
a_0b	1×4	1×2	1×8	1×0
a_1b	2×4	2×2	2×8	2×0
a_2b	4×4	4×2	4×8	4×0
a_3b	3×4	3×2	3×8	3×0

$$c_2 = 4 \times 4 + 2 \times 2 + 1 \times 8 = 28$$

卷积与多项式乘法

多项式乘法: $C(x) = A(x) B(x)$

$$A(x) = a_0 + a_1x + a_2x^2 + \dots + a_{m-1}x^{m-1}$$

$$B(x) = b_0 + b_1x + b_2x^2 + \dots + b_{n-1}x^{n-1}$$

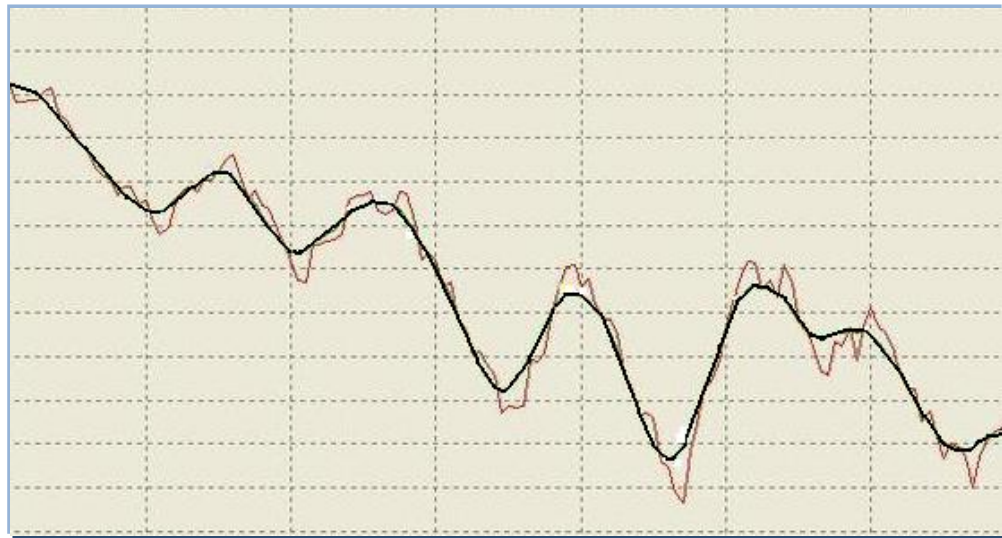
$$C(x) = \underline{a_0b_0 + (a_0b_1 + a_1b_0)x + \dots}$$
$$\underline{+ a_{m-1}b_{n-1}x^{m+n-2}}$$

其中 x^k 的系数

$$c_k = \sum_{\substack{i+j=k \\ i \in \{0,1,\dots,m-1\} \\ j \in \{0,1,\dots,n-1\}}} a_i b_j, \quad k = 0, 1, \dots, m+n-2$$

卷积应用：信号平滑处理

由于噪音干扰，对信号需要平滑处理

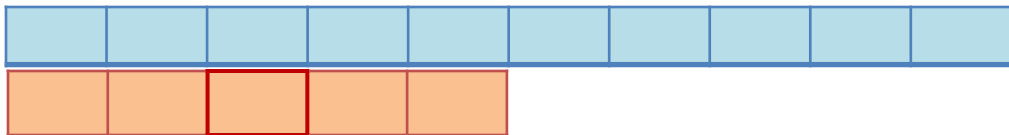


平滑处理

信号向量: $a=(a_0, a_1, \dots, a_{m-1})$

$b=(b_{2k}, b_{2k-1}, \dots, b_0) = (w_{-k}, \dots, w_k)$

$$a_i' = \sum_{s=-k}^k a_{i+s} b_{k-s} = \sum_{s=-k}^k a_{i+s} w_s$$



把向量 b 看作 $2k+1$ 长度窗口在 a 上移动计算 $a*b$, 得到 $(a_0', a_1', \dots, a_{m-1}')$. 有少数项有误差.

实例

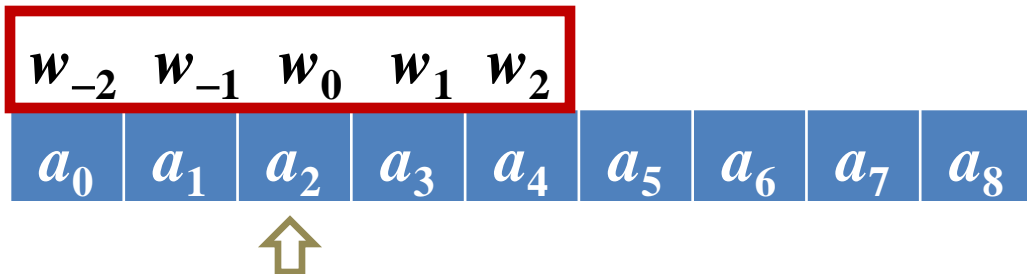
信号向量: $a = (a_0, a_1, \dots, a_8)$

步长: $k = 2$

权值: $w = (w_{-2}, w_{-1}, w_0, w_1, w_2)$
 $= (b_4, b_3, b_2, b_1, b_0)$

$$a_i' = a_{i-2}b_4 + a_{i-1}b_3 + a_ib_2 + a_{i+1}b_1 + a_{i+2}b_0$$

下标之和为 $i + k$



$$a_i' = a_{i-2}b_4 + a_{i-1}b_3 + a_ib_2 + a_{i+1}b_1 + a_{i+2}b_0$$

a_0b_0	a_0b_1	a_0b_2	a_0b_3	a_0b_4	a_2'
a_1b_0	a_1b_1	a_1b_2	a_1b_3	a_1b_4	a_3'
a_2b_0	a_2b_1	a_2b_2	a_2b_3	a_2b_4	a_4'
a_3b_0	a_3b_1	a_3b_2	a_3b_3	a_3b_4	a_5'
a_4b_0	a_4b_1	a_4b_2	a_4b_3	a_4b_4	a_6'
a_5b_0	a_5b_1	a_5b_2	a_5b_3	a_5b_4	
a_6b_0	a_6b_1	a_6b_2	a_6b_3	a_6b_4	
a_7b_0	a_7b_1	a_7b_2	a_7b_3	a_7b_4	
a_8b_0	a_8b_1	a_8b_2	a_8b_3	a_8b_4	

小结

- 卷积的定义
- 卷积与多项式乘法的关系
- 卷积的应用——信号平滑处理

卷积计算

卷积计算：蛮力算法

向量 $a=(a_0,a_1,\dots,a_{n-1})$ 和 $b=(b_0,b_1,\dots,b_{n-1})$

$$A(x)=a_0+a_1x+a_2x^2+\dots+a_{n-1}x^{n-1}$$

$$B(x)=b_0+b_1x+b_2x^2+\dots+b_{n-1}x^{n-1}$$

$$C(x) = A(x)B(x)$$

$$=a_0b_0 + (a_0b_1+a_1b_0) x + \dots + a_{n-1}b_{n-1} x^{2n-2}$$

$C(x)$ 的系数向量就是 $a*b$.

卷积 $a*b$ 计算等价于多项式相乘

蛮力算法的时间: $O(n^2)$

计算 $2n-1$ 次多项式 $C(x)$

1. 选择值 $x_0, x_1, \dots, x_{2n-1}$,

求出 $A(x_j)$ 和 $B(x_j)$,
 $j = 0, 1, \dots, 2n-1$

主要步骤:
多项式求值

2. 对每个 j , 计算 $C(x_j) = A(x_j)B(x_j)$

3. 利用多项式插值方法, 由 $C(x)$ 在

$x = x_0, x_1, \dots, x_{2n-1}$

的值求出多项式 $C(x)$ 的系数



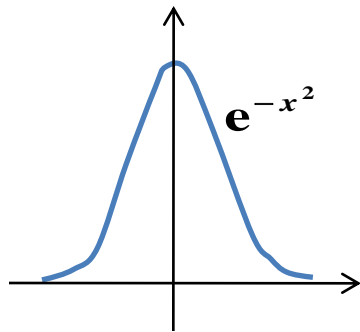
如何选择 $x_0, x_1, \dots, x_{2n-1}$ 的值?
高效多项式求值算法

高斯滤波的权值函数

高斯滤波的权值函数为

$$w_s = \frac{1}{z} e^{-s^2}, \quad s = 0, \pm 1, \dots, \pm k$$

$$w = (w_{-k}, \dots, w_{-1}, w_0, w_1, \dots, w_k)$$



其中 z 用于归一化处理，使所有的权值之和为1. 处理结果

$$a_i' = \sum_{s=-k}^k a_{i+s} w_s$$

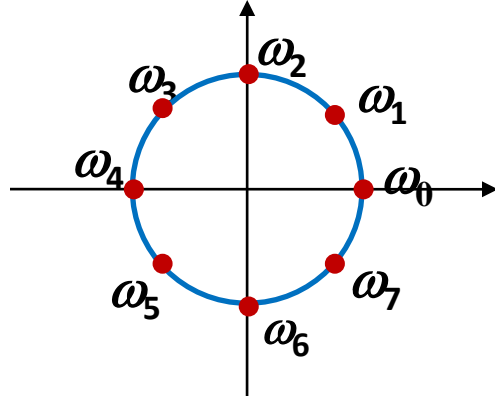
$2n$ 个数的选择:1的 $2n$ 次根

$$\omega_j = e^{\frac{2\pi j}{2n}i} = e^{\frac{\pi j}{n}i}$$

$$= \cos \frac{\pi j}{n} + i \sin \frac{\pi j}{n}$$

$$j=0,1,\dots,2n-1, \quad i=\sqrt{-1}$$

$n=4$ 的实例



$$\begin{aligned}\omega_0 &= 1, & \omega_1 &= e^{\frac{\pi}{4}i} = \sqrt{2}/2 + \sqrt{2}/2 \cdot i, \\ \omega_2 &= e^{\frac{\pi}{2}i} = i, & \omega_3 &= e^{\frac{3\pi}{4}i} = -\sqrt{2}/2 + \sqrt{2}/2 \cdot i, \\ \omega_4 &= e^{\pi i} = -1, & \omega_5 &= e^{\frac{5\pi}{4}i} = -\sqrt{2}/2 - \sqrt{2}/2 \cdot i, \\ \omega_6 &= e^{\frac{3\pi}{2}i} = -i, & \omega_7 &= e^{\frac{7\pi}{4}i} = \sqrt{2}/2 - \sqrt{2}/2 \cdot i\end{aligned}$$

快速傅立叶变换FFT

1. 对 $x=1, \omega_1, \omega_2, \dots, \omega_{2n-1}$, 分别计算 $A(x), B(x)$
2. 利用步1的结果对每个 $x = 1, \omega_1, \omega_2, \dots, \omega_{2n-1}$, 计算 $C(x)$, 得到
 $C(1)=d_0, C(\omega_1)=d_1, \dots, C(\omega_{2n-1})=d_{2n-1}$
3. 构造多项式
$$D(x) = d_0 + d_1x + d_2x^2 + \dots + d_{2n-1}x^{2n-1}$$
4. 对 $x=1, \omega_1, \omega_2, \dots, \omega_{2n-1}$, 计算 $D(x)$,
 $D(1), D(\omega_1), \dots, D(\omega_{2n-1})$

快速傅立叶变换FFT (续)

可以证明:

$$D(1) = 2n c_0$$

$$D(\omega_1) = 2n c_{2n-1}$$

...

$$D(\omega_{2n-1}) = 2n c_1$$



$$c_0 = D(1)/2n$$

$$c_{2n-1} = D(\omega_1)/2n$$

...

$$c_1 = D(\omega_{2n-1})/2n$$

知道了 $D(x)$ 的值, 就能求 $C(x)$ 的系数

算法的关键

令 $x = 1, \omega_1, \omega_2, \dots, \omega_{2n-1}$,

- 步 1 对 $2n$ 个 x 值分别求值多项式 $A(x), B(x)$
- 步 2 做 $2n$ 次乘法
- 步 3 对 $2n$ 个 x 值求值多项式 $D(x)$

关键：一个对所有的 x 快速多项式求
值算法

小结

卷积计算

- 蛮力算法 $O(n^2)$
- 快速傅立叶变换FFT算法
确定 x 的取值: 1 的 $2n$ 次根
关键步骤: 多项式对 x 求值



如何设计多项式求值的快速算法?

快速傅立叶 变换:FFT算法

多项式求值算法

给定多项式:

$$A(x)=a_0+a_1x+\dots+a_{n-1}x^{n-1}$$

设 x 为 1 的 $2n$ 次方根, 对所有的 x 计算 $A(x)$ 的值.

算法1: 对每个 x 做下述运算:

依次计算每个项 $a_i x^i$, $i=1, \dots, n-1$

对 $a_i x^i$ ($i=0,1,\dots,n-1$) 求和.

蛮力算法的时间复杂度

$$T_1(n) = O(n^3)$$

改进的求值算法

算法2: 依次对 每个 x 做下述计算

$$\underline{A_1(x)} = a_{n-1}$$

$$A_2(x) = a_{n-2} + x \underline{A_1(x)} = a_{n-2} + a_{n-1}x$$

$$A_3(x) = a_{n-3} + x A_2(x) = a_{n-3} + a_{n-2}x + a_{n-1}x^2$$

...

$$A_n(x) = a_0 + x A_{n-1}(x) = A(x)$$

时间复杂度

$$T_2(n) = O(n^2)$$

偶系数与奇系数多项式

$$n=4$$

$$A(x)=a_0+a_1x+a_2x^2+a_3x^3$$

$$A_{\text{even}}(x) = a_0+a_2x$$

$$A_{\text{odd}}(x) = a_1+a_3x$$

$$A(x) = A_{\text{even}}(x^2) + xA_{\text{odd}}(x^2)$$

$$= a_0+a_2x^2 + x(a_1+a_3x^2)$$

分治多项式求值算法

一般公式 (n 为偶数)

$$A(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}$$

$$A_{\text{even}}(x) = a_0 + a_2x + a_4x^2 + \dots + a_{n-2}x^{(n-2)/2}$$

$$A_{\text{odd}}(x) = a_1 + a_3x + a_5x^2 + \dots + a_{n-1}x^{(n-2)/2}$$

$$A(x) = A_{\text{even}}(x^2) + xA_{\text{odd}}(x^2)$$

- x^2 也是1 的 $2n$ 次根
- 偶次数与奇次数多项式计算作为 $n/2$ 规模的子问题, 然后利用子问题的解 $A_{\text{even}}(x^2)$ 与 $A_{\text{odd}}(x^2)$ 得到 $A(x)$

分治求值算法设计

算法 3:

1. 计算 1 的所有的 $2n$ 次根

$$1, \omega_1, \omega_2, \dots, \omega_{2n-1}$$

2. 分别计算 $A_{\text{even}}(x^2)$ 与 $A_{\text{odd}}(x^2)$

3. 利用步2 的结果计算 $A(x)$

$$A(x) = A_{\text{even}}(x^2) + x A_{\text{odd}}(x^2)$$

注意: x^2 不需要重新计算, 所有根在单位圆间隔分布, 隔一取一即可.

分治求值算法分析

$$T(n) = T_1(n) + f(n)$$

$f(n)=O(n)$ 是步 1 计算 $2n$ 次根的时间

递归过程 $T_1(n) = 2T_1(n/2) + g(n)$

$$T_1(1) = O(1),$$

$g(n) = O(n)$ 是对所有 $2n$ 次根在步3组合解的时间

$$T_1(n)=O(n\log n)$$

$$T(n)=O(n\log n)+O(n)=O(n\log n)$$

FFT算法伪码

1. 求值 $A(\omega_j)$ 和 $B(\omega_j)$, $j=0,1,\dots,2n-1$
2. 计算 $C(\omega_j)$, $j=0, 1, \dots, 2n-1$
3. 构造多项式

$$D(x)=C(\omega_0)+C(\omega_1)x+\dots+C(\omega_{2n-1})x^{2n-1}$$

4. 计算所有的 $D(\omega_j)$, $j=0,1,\dots,2n-1$
5. 利用下式计算 $C(x)$ 的系数 c_j ,

$$D(\omega_j) = 2nc_{2n-j}$$
$$j = 0, 1, \dots, 2n-1$$

FFT算法分析

步1: 求值 $A(\omega_j)$ 和 $B(\omega_j)$ $O(n \log n)$

步2: 计算所有的 $C(\omega_j)$ $O(n)$

步3:

步4: 计算所有的 $D(\omega_j)$ $O(n \log n)$

步5: 计算所有的 c_j $O(n)$

算法时间为 $O(n \log n)$

小结

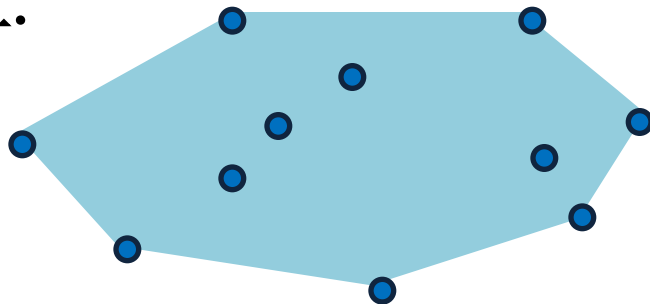
- 多项式求值算法
蛮力算法: $O(n^3)$
改进的求值算法: $O(n^2)$
FFT算法: $O(n\log n)$
- FFT算法的设计与分析

平面点集的凸包

平面点集的凸包

问题（平面点集的凸包）

给定大量离散点的集合 Q ，求一个最小的凸多边形，使得 Q 中的点在该多边形内或者边上。

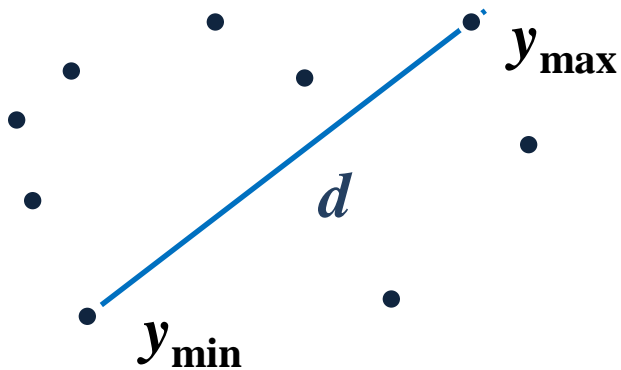


应用背景

图形处理中用于形状识别：字形识别、碰撞检测等

分治算法

1. 以 连接最大纵坐标点 y_{\max} 和最小纵坐标点 y_{\min} 的线段 $d = \{y_{\max}, y_{\min}\}$ 划分 L 为左点集 L_{left} 和右点集 L_{right}



2. Deal (L_{left}); Deal (L_{right})

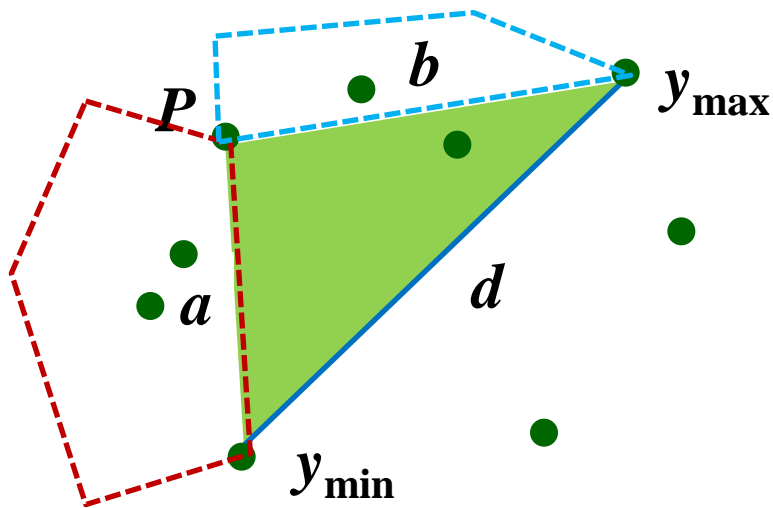
Deal (L_{left})

考虑 L_{left} : 确定距 d 最远的点 P

在三角形内的点, 删除;

a 外的点与 a 构成 L_{left} 的子问题;

b 外的点与 b 构成 L_{left} 的子问题.



伪码

Deal (L_{left})

1. 以 d 和距离 d 最远点 P 构成三角形, P 加入凸包, 另外两条边分别记作 a 和 b
2. 检查 L_{left} 中其他点是否在三角形内; 在则从 L 中删除; 否则根据在 a 或 b 边的外侧划分在两个子问题中
3. Deal (a)
4. Deal (b)

算法分析

- 初始用 d 划分 $O(n)$
- Deal 递归调用 $W(n)$
 - 找凸包顶点 P $O(n)$
 - 根据点的位置划分子问题 $O(n)$

- $$W(n) = W(n-1) + O(n)$$
$$W(3) = O(1)$$

最坏情况为 $O(n^2)$

$$T(n) = O(n) + W(n) = O(n^2)$$

- Graham扫描算法 $O(n \log n)$

小结：分治算法设计

- 将原问题归约为子问题
 - 直接划分注意尽量均衡
 - 通过计算归约为特殊的子问题
 - 子问题与原问题具有相同的性质
 - 子问题之间独立计算
- 算法实现：
 - 递归或迭代实现
 - 注意递归执行的边界

小结：分治算法的 分析及改进

- 时间复杂度分析
 - 给出关于时间复杂度函数的
递推方程和初值
 - 求解方程
- 提高效率的途径
 - 减少子问题个数
 - 预处理

重要的分治算法

- 检索算法：二分检索
- 排序算法：快速排序、二分归并排序
- 选择算法
- 快速傅立叶变换 FFT 算法
- 平面点集的凸包