

# Algorithm - P66

Young 代矩阵 对于  $n^2$  个数，利用  $n \times n$  的 Young 代矩阵进行排序

YOUNG-SORT ( $n^2$  个数) :

将  $n^2$  个数装入  $n \times n$  的矩阵 Y

for  $i := n$  down to 1 do

    for  $j := n$  down to 1 do

        MIN-SINK(Y, i, j)

    ret = []

    for  $k := 1$  to  $n^2$  do

        key := YOUNG-EXTRACT-MIN(Y)

        ret[k] := key

    return ret

注意其中有两套 for 循环，都需要执行  $n^2$  次迭代

而循环结构中仅有 MIN-SINK 和 YOUNG-EXTRACT-MIN

的时间复杂度为  $O(n+n) = O(n)$

于是 YOUNG-SORT 在时间复杂度  $O(n^3)$  内实现对  $n^2$  个数进行排序

如果令  $m = n^2$ ，则 YOUNG-SORT 的时间复杂度为  $O(m^{1.5})$  的

注意实际上是要慢于排序的下限  $\Omega(n \lg n)$  的

堆

如果数据均为 b 位整型数，且计算机内存也是由可寻址的 b 位字组成

Fredman 和 Willard 在时间复杂度  $O(1)$  时间内实现 MINIMUM

在时间复杂度  $O(\lg \lg n)$  时间内实现 INSERT, EXTRACT-MIN

Thorup 则将时间复杂度的界降低至  $O(\lg \lg n)$

性能的提升以额外的存储空间为代价

可以用随机散列方法在线性空间中实现

优先队列的重要特性和情形：EXTRACT-MIN 操作序列是单调的 (monotone)

即连续的 EXTRACT-MIN 操作的返回值随时间单调递增 (monotonically increasing overtime)

快速排序 (quicksort) 通常是实际排序应用中最好的选择

分解：将数组  $A[p..r]$  划分为两个可能为空的子数组  $A[p..q-1]$  和  $A[q+1..r]$

使得  $A[p..q-1]$  的元素均小于  $A[q]$ ,  $A[q+1..r]$  的元素均大于  $A[q]$

其中计算下标 q 也是分解过程的一部分

解决：递归地调用快速排序对子数组  $A[p..q-1]$  和  $A[q+1..r]$  进行排序

合并：由于快速排序是原址排序的，所以无需合并操作， $A[p..r]$  已经是有序的

# Algorithm - P67

## 快速排序

快速排序的关键部分是对数组的划分，实现了对子数组  $A[p..r]$  的原地重排。

**PARTITION(A, p, r)**

    pivot :=  $A[r]$

$i := p-1$

    for  $j := p$  to  $r-1$ :

        if  $A[j] \leq pivot$

$i := i+1$

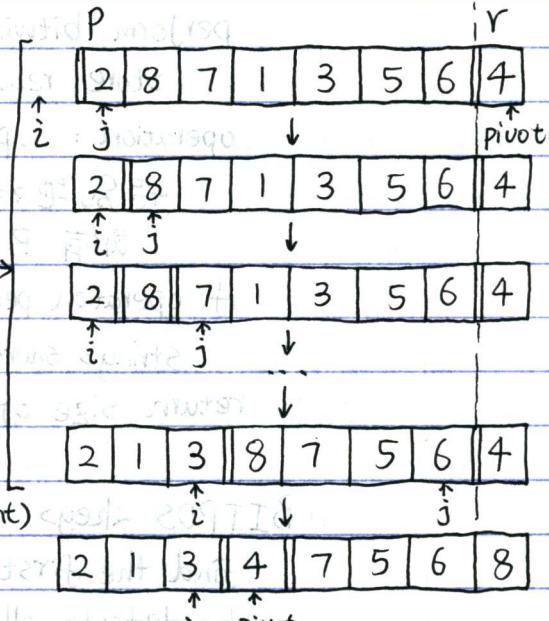
            swap  $A[i]$  with  $A[j]$

    swap  $A[i+1]$  with  $A[r]$

    return  $i+1$

PARTITION总是选择一个主元(pivot element)

并围绕主元划分子数组  $A[p..r]$



循环不变量：在循环的每一次迭代开始前，对于任意数组下标  $p \leq k \leq r$

① 如果有  $p \leq k \leq i$ , 则  $A[k] \leq pivot$

② 如果有  $i+1 \leq k \leq j-1$ , 则  $A[k] > pivot$

③ 如果有  $k=r$ , 则  $A[k] = pivot$

即 PARTITION 实际上维护着 4 个可能为空的区域

$p$	$i$	$i+1$	$j-1$	$j$	$r-1$	$r$
$\leq pivot$	$< pivot$	$> pivot$	无限制	$= pivot$		

初始化：在第 1 次迭代开始前， $i=p-1$ ,  $j=p$ ，于是子数组  $A[p..i]$  和  $A[i+1..j-1]$  为空

是 ① 和 ② 平凡地为真，而赋值保证  $A[r] = pivot$  为真

保持：假设在 循环计数器  $p \leq j \leq r-1$  的迭代开始前 循环不变量为真，则考虑下一次迭代之前

当  $A[j] > pivot$  时，唯一的操作是 循环计数器  $j++$

则  $A[j-1]$  在  $j$  增加后 对 ② 成立，① 和 ③ 保持不变

当  $A[j] \leq pivot$  时， $i++$ ，交换  $A[i]$  和  $A[j]$ ,  $j++$

则 完成后  $A[i]$  为之前的  $A[j-1] \leq pivot$ ，对 ① 成立

$A[j-1]$  为之前的  $A[i] > pivot$ ，对 ② 成立

终止：当终止时，有循环计数器  $j=r$ ，此时  $A[p..i]$ ,  $A[i+1..j-1]$ ,  $A[r]$  覆盖  $A[p..r]$

即划分为  $\leq pivot$ ,  $> pivot$  两个子数组和一个元素  $pivot$

而在交换  $A[i+1]$  和  $A[r]$  后， $A[p..i]$  中都  $\leq pivot$ ,  $A[i+1..r]$  都  $> pivot$

则返回  $pivot$  的下标  $i+1$ 。

# Algorithm - P68

## 快速排序

PARTITION 对于输入的子数组  $A[p..r]$ , 其中有  $n = r-p+1$  个元素

需要  $r-p$  次循环迭代, 且其他语句均为  $\Theta(1)$

于是 PARTITION 在时间复杂度  $\Theta(n)$  内实现对子数组的划分

在 PARTITION(A, p, r) 的基础上实现快速排序算法

QUICKSORT(A, p, r)

if  $p < r$ : ] 当子数组少于 2 个元素时, 直接返回

$q = \text{PARTITION}(A, p, r)$  ] 划分子数组,  $q$  为 pivot 下标

QUICKSORT(A, p, q-1) ] 递归地调用 QUICKSORT

QUICKSORT(A, q+1, r) ] 在子数组  $A[p..q-1]$  和  $A[q+1..r]$  上递归

初始调用为  $\text{QUICKSORT}(A, 1, A.length)$

快速排序的运行时间取决于划分子数组是否平衡

而划分平衡与否取决于用于划分的元素 pivot

最坏情形划分:

当选取的 pivot 是当前子数组中最大/最小的元素时

划分的子数组会分别包含 0 个元素与  $n-1 = r-p$  个元素

现对于当前的 QUICKSORT 调用

将有递归式  $T(n) = T(n-1) + \Theta(n)$

其中  $T(0) = \Theta(1)$ , 而  $\Theta(n)$  为 PARTITION 的时间复杂度

如果在每个步骤都选取了最大/最小元素

如输入的序列是递增/递减的, 即已排序的

则有递归式  $T(n) = T(n-1) + \Theta(n) = \sum_{i=2}^n \Theta(i)$

于是有  $T(n) = \Theta(n^2)$ , 与插入排序一致

最优情形划分:

当选取的 pivot 性质地平衡地划分子数组时

即划分的子数组 分别包含  $\lceil n/2 \rceil$  个元素 与  $\lceil n/2 \rceil - 1$  个元素

于是对于当前的 QUICKSORT 调用

将有递归关系  $T(n) = T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil - 1) + \Theta(n)$

$= 2T(n/2) + \Theta(n)$

如果在每个步骤都选取了如上的 pivot

则有递归式  $T(n) = 2T(n/2) + \Theta(n)$

于是有  $T(n) = \Theta(n \lg n)$ , 与归并排序一致

# Algorithm - P69

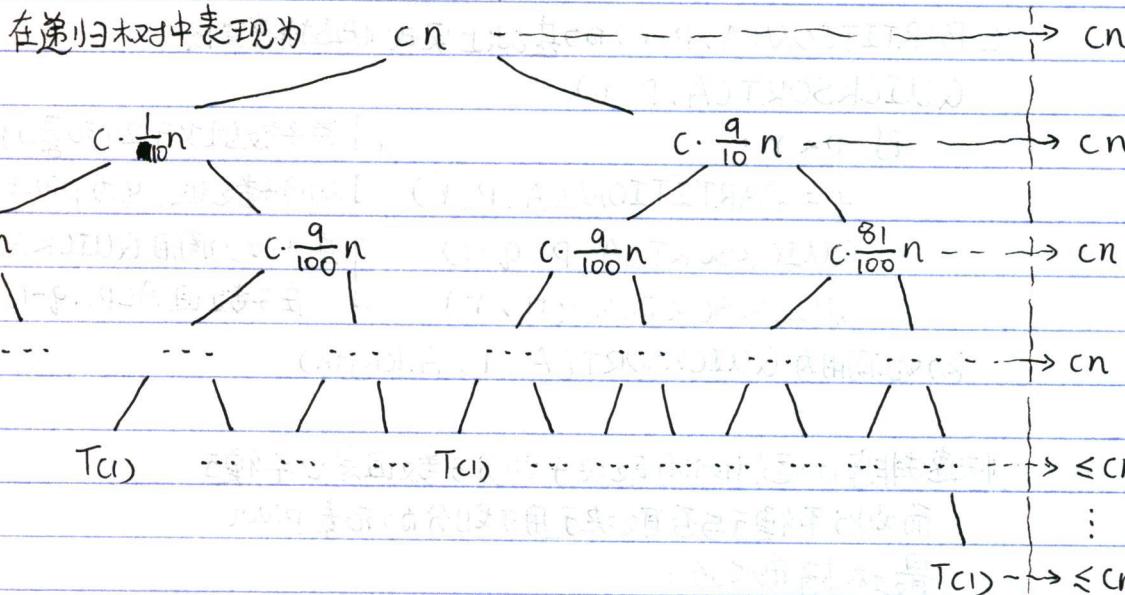
## 快速排序

平衡的划分：快速排序的平均运行时间更接近于最优划分情形

如假设划分算法总是产生 9:1 的划分

得到的快速排序时间复杂度的递归式为

$$T(n) = T(9n/10) + T(n/10) + cn$$



且是在深度为  $\log_{10} n \in \Theta(\lg n)$  之前，每层递归的代价和为  $cn$

而在深度为  $\log_{10} n \in \Theta(\lg n)$  之前，每层递归的代价和不大于  $cn$

再考虑划分算法总是产生  $a:1$  的划分

则在深度为  $\log_{a+1} n$  之前，每层递归的代价和均为  $O(cn)$

又对于任意正实数  $a \in \mathbb{R}^+$ ，都有  $\log_{a+1} n \in \Theta(\lg n)$

于是对产生任意常数比例的划分算法，快速排序的时间复杂度为  $O(n \lg n)$

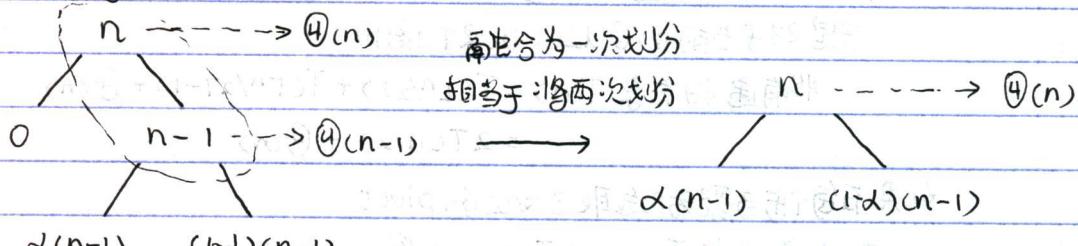
考虑平均情形下，PARTITION 产生同时混合坏与优的划分

且在平均情形对应的递归树中，好与坏的划分随机地出现

考虑在一个坏的划分  $0 : (n-1)$  之后为一个非最坏的划分  $d(n-1) : (1-d)(n-1)$

又  $\Theta(n) + \Theta(n-1) \in \Theta(n)$

于是递归树表现为



直觉上看，最坏划分的代价被吸引到好的划分的代价  $\Theta(n)$  中

于是当好/坏的划分交替出现时，快速排序的时间复杂度与最好情形一样是  $O(n \lg n)$  的

# Algorithm - P70

Shandong

839 - 8.1.1.1.1

快速排序

对于实数  $0 < \alpha \leq \frac{1}{2}$ , 考虑快速排序在每一层均产生相同比例的划分

即有划分的比例为  $1-\alpha : \alpha$ , 则有  $1-\alpha \geq \alpha$

则在相应的递归树中, 叶结点的最小深度约为  $-\lg n / \lg \alpha$

叶结点的最大深度约为  $-\lg n / \lg(1-\alpha)$

证明过程有, 由于划分的比例中  $\alpha \leq 1-\alpha$

于是在每一层划分中均为  $\alpha$  的子结点将最快抵达基础情形

即  $n \cdot \alpha^{h_{\min}} = 1$

$$h_{\min} = \log_{\alpha}(1/n) = \lg(1/n) / \lg \alpha = -\lg n / \lg \alpha$$

而在每一层划分中均为  $1-\alpha$  的子结点将最慢抵达基础情形

即  $n \cdot (1-\alpha)^{h_{\max}} = 1$

$$h_{\max} = \log_{(1-\alpha)}(1/n) = \lg(1/n) / \lg(1-\alpha) = -\lg n / \lg(1-\alpha)$$

于是有叶结点的最小深度约为  $-\lg n / \lg \alpha$ , 最大深度约为  $-\lg n / \lg(1-\alpha)$

考虑对于一个随机输入的数组进行划分, 并考虑划分比例  $1-\alpha : \alpha$

则对于任意实数  $0 < \alpha \leq \frac{1}{2}$ ,

PARTITION 产生  $1-\alpha : \alpha$  或更不平衡的划分的根无率为  $1-2\alpha$

证明过程为, 虽然 PARTITION 总是选择子数组的最后一项作为主元

但由于数组是随机输入的, 则主元在数组中的分位是随机的

假定选中任意分位点, 其元素都是可能的

而使得划分比  $1-\alpha : \alpha$  更不平衡的元素  $\rightarrow$

或者是最大的比例为  $\alpha$  的元素, 即高  $\alpha$  分位点

或者是最小的比例为  $\alpha$  的元素, 即低  $\alpha$  分位点

于是可知产生  $1-\alpha : \alpha$  或更不平衡划分的根无率为  $1-2\alpha$

考虑输入数组中存在相同元素的情形

对于极端情形下输入数组中的元素全部相同

则在 PARTITION 过程中, 根据判定条件  $A[i] \leq x$

由于所有元素都相等, 即均满足判定条件

于是除主元外的所有元素均归入一侧子数组, 而另一侧数组为空

即返回的主元的下标为  $i-1$

于是产生递归式  $T(n) = T(n-1) + \Theta(n)$

即时调用深度与最坏情况划分相同, 为  $\Theta(n^2)$

# Algorithm - P71

快速排序 考虑修改 PARTITION 过程, 以覆盖与主元相等的元素

BALANCED-PARTITION(A, p, r)

pivot := A[r]

i := p-1

for j := p to r-1

if A[j] < pivot

i := i+1

then swap A[i] with A[j]

else if A[j] == pivot && ((i+1-p)\*2 < (j-p))

    i := i+1

    swap A[i] with A[j]

Swap A[i+1] with A[r]

return i+1 + pivot - 1 - (r-p)

小于主元的元素

等于主元的元素

注意由于在出现与主元相同的元素时, 无法预测其后出现的元素大小

于是仅根据已划分的元素情况作出决定 - P-5-

当分配给左侧子数组的元素少于已分配元素的一半时

则将与主元相等放入左侧子数组, 即小于等于主元的子数组

否则将与主元相等的元素放入右侧子数组, 即大于等于主元的子数组

又对左侧子数组递归地调用 QUICKSORT

等于主元的元素均排列在子数组的最后

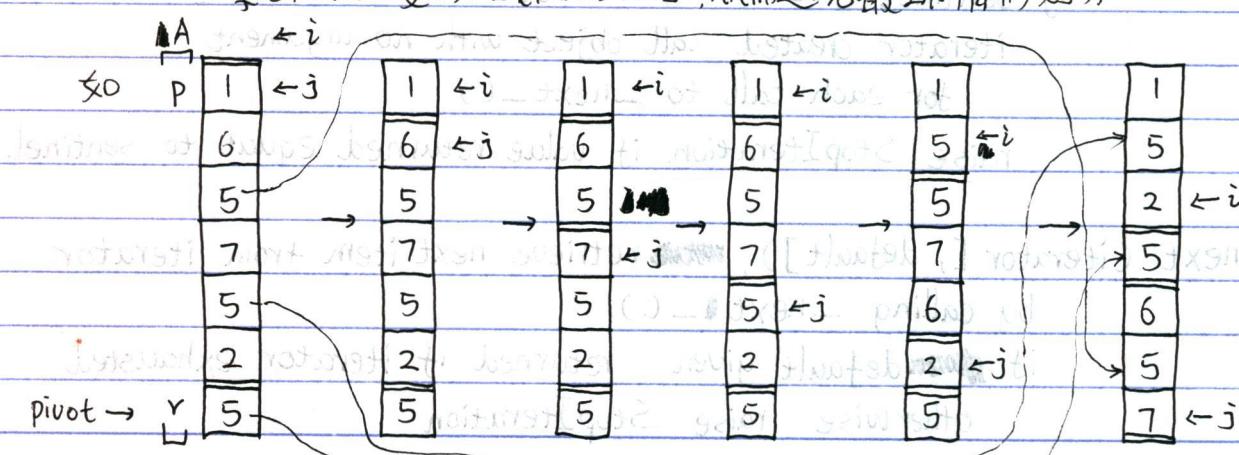
对右侧子数组递归地调用 QUICKSORT

等于主元的元素均排列在子数组的开头

于是连接左侧、主元、右侧时, 与主元相等的元素保持连续

极端情况下, 当子数组 A[p..r] 中所有元素均相等时

返回的索引由 r 变为  $\lfloor (p+r)/2 \rfloor$ , 从而避免最坏情形划分



# Algorithm - P72

INT - sendpunkt

## 快速排序

通常在讨论快速排序的平均情况性能时

前提假设输入数据的所有排列都是等概率的

但实际工程中这个假设并不总是成立

于是通过在算法中引入随机性，从而使算法对所有输入都能获得较好的期望性能

可以显式地对输入进行重新排列，从而使算法实现随机化

也可以采用随机抽样 (random sampling)，使分析变得简单

从  $A[p..r]$  中随机选择一个元素并与  $A[r]$  交换

于是主元  $\text{pivot} := A[r]$  等概率地从  $A[p..r]$  的  $r-p+1$  个元素中选取的

期望在平均情况下，对输入子数组的划分是比较均衡的

RANDOMIZED-PARTITION( $A, p, r$ )

$i := \text{RANDOM}(p, r)$

swap  $A[i]$  with  $A[r]$

return BALANCED-PARTITION( $A, p, r$ )

RANDOMIZED-QUICKSORT( $A, p, r$ )

if  $p < r$

$q := \text{RANDOMIZED-PARTITION}(A, p, r)$

RANDOMIZED-QUICKSORT( $A, p, q-1$ )

RANDOMIZED-QUICKSORT( $A, q+1, r$ )

RANDOMIZED-QUICKSORT 的最坏情况分析

假设  $T(n)$  为输入规模为  $n$  的数据集合上的最坏情况时间复杂度

则有递归式  $T(n) = \max_{0 \leq q \leq n-1} [T(q) + T(n-q-1)] + \Theta(n)$

其中  $q$  为 RANDOMIZED-PARTITION 返回的值， $\Theta(n)$  为其时间复杂度

于是猜测  $T(n) \in O(n^2)$ ，且  $T(n) \leq cn^2$ ，其中  $c$  为正实数常数

$$T(n) = \max_{0 \leq q \leq n-1} [T(q) + T(n-q-1)] + \Theta(n)$$

$$\leq \max_{0 \leq q \leq n-1} [c \cdot q^2 + c \cdot (n-q-1)^2] + \Theta(n)$$

$$= c \max_{0 \leq q \leq n-1} [q^2 + (n-q-1)^2] + \Theta(n)$$

又当  $q=0$  或  $q=n-1$  时， $q^2 + (n-q-1)^2$  取得最大值  $(n-1)^2$

于是  $T(n) \leq c \max_{0 \leq q \leq n-1} [q^2 + (n-q-1)^2] + \Theta(n)$

$$= c \cdot (n-1)^2 + \Theta(n) = cn^2 - c(2n-1) + \Theta(n)$$

又存在足够大的常数  $C$ ，使得  $c(2n-1)$  显著大于  $\Theta(n)$

于是有  $T(n) \in O(n^2)$