



# 福昕高级PDF编辑器

高效 · 安全 · 专业

立即下载

点击购买



OFFICE格式互转



OCR文字识别



文本图像编辑



加密和签署



交互式动态表单



互联PDF文档



# 福昕高级PDF编辑器

高效 · 安全 · 专业

立即下载

购物车 点击购买



OFFICE格式互转



OCR文字识别



文本图像编辑



加密和签署



交互式动态表单



互联PDF文档



中国科学技术大学 计算机科学与技术系  
University of Science and Technology of China  
DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

# 算法设计与分析

## Design and Analysis of Algorithms

主讲人 徐云

Fall 2018, USTC



Part 1 Foundation

Part 2 Sorting and Order Statistics

chap 6 Heapsort

chap 7 Quicksort

**chap 8 Sorting in Linear Time**

chap 9 Medians and Order Statistics

Part 3 Data Structure

Part 4 Advanced Design and Analysis Techniques

Part 5 Advanced Data Structures

Part 6 Graph Algorithms

Part 7 Selected Topics

Part 8 Supplement



## 第8章 线性时间的排序

### 8.1 排序的下界

### 8.2 计数排序

### 8.3 基数排序

### 8.4 桶排序



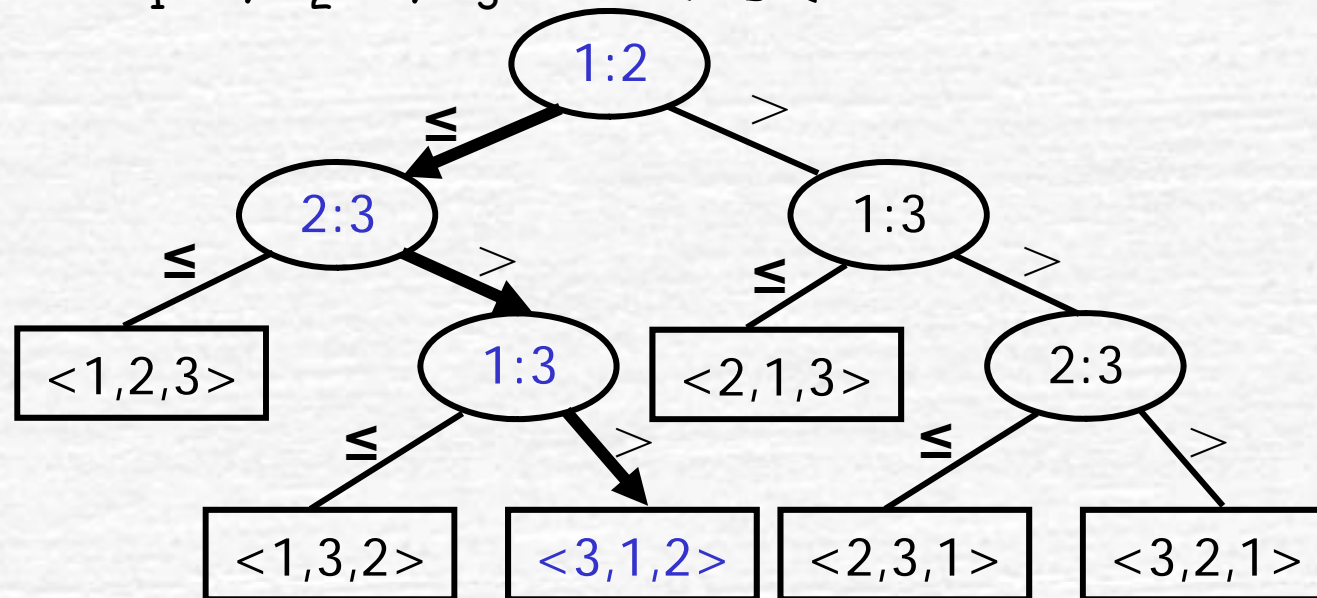
# 8.1 排序的下界

- 判定树模型
- 最坏情形的下界



# 判定树模型

- 任何 $n$ 个元素 $a_1, a_2, \dots, a_n$ 的基于比较的排序可以用一棵二叉排序树表示，其叶子节点代表一类输入实例的排序结果。
- 示例： $a_1=6, a_2=8, a_3=5$ 其判定树如下



该实例的比较路径由箭头所示，叶子节点代表的是排序结果。

# 最坏情形的下界

- 定理8.1 基于比较排序算法在最坏情形下，需要  $\Omega(n \log n)$ 。
- 证明：设  $h, l$  分别代表判定树的高度和叶子数
  - $\therefore$  判定树是一棵二叉树
  - $\therefore l \leq 2^h$  // 叶子数不超过  $2^h$
  - $\therefore n! \leq l \leq 2^h$  //  $n!$  为排序的结果数  
两边取对数于是有， $h \geq \log n! \geq \Omega(n \log n)$  证毕！

归并排序和堆排序是渐近最优的比较排序算法





## 第8章 线性时间的排序

### 8.1 排序的下界

### 8.2 计数排序

### 8.3 基数排序

### 8.4 桶排序



## 8.2 计数排序

- 基本思想
- 一种特殊情形的计数排序
- 一般情形的计数排序

# 基本思想

- 统计小于或等于 $A[i]$ 的元素数目，将 $A[i]$ 置入相应的位置，即

$A[i] \rightarrow B[\text{小于或等于 } A[i] \text{ 的元素数目}]$

- 主要要解决的问题：
  - 计数：统计小于或等于 $A[i]$ 的元素数目
  - 值相同元素的处理

# 一种特殊情形的计数排序

- 问题：n个互不相同的整数 $A[1..n]$ ,  $1 \leq A[i] \leq n$   
 $i=1 \sim n$
- 排序算法：  
SpecialCountingSort(A,B)  
{//B[1..n]为排序结果  
  for  $i \leftarrow 1$  to  $n$  do  
     $B[A[i]] \leftarrow A[i]$ ; //如 $A[i]=5$ , 就放到 $B[5]$ 中  
}
- 时间： $O(n)$ ，无比较



# 一般情形的计数排序 (1)

- 问题:  $n$ 个可以相同的整数 $A[1..n]$ ,  $1 \leq A[i] \leq k$ ,  $i=1 \sim n$ , 这里 $k$ 是 $A[i]$ 的取值范围, 不一定为 $n$ 。
- 基本思想:  
步骤:  $A[1..n] \rightarrow$  计数器 $C[1..k] \rightarrow B[1..n]$ 
  - Step 1(值相同元素的计数): 将 $A$ 中的值为 $i$ 的元素个数记入 $C[i]$ 中;
  - Step 2(累计计数): 对 $C[1..k]$ 进行修改, 使得 $C[i]$ 的值表示为 $\leq i$ 的元素个数;
  - Step 3(放置): 将 $A[j]$ 依据 $C[A[j]]$ , 放入正确位置 $B[C[A[j]]]$ 上, 并修改 $C[A[j]] \leftarrow C[A[j]] - 1$ ;



# 一般情形的计数排序 (2)

- 算法:

CountingSort(A,B,k)

{//B[1..n]为排序结果, C[1..k]为计数数组

for  $i \leftarrow 1$  to  $k$  do  $C[i] \leftarrow 0$ ;

for  $j \leftarrow 1$  to  $\text{length}[A]$  do //扫描A, 值相同元素计数

$C[A[j]]++$ ;

for  $i \leftarrow 2$  to  $k$  do //C[i]修改, 累计计数

$C[i] \leftarrow C[i] + C[i-1]$ ;

for  $j \leftarrow \text{length}[A]$  downto 1 do

{  $B[C[A[j]]] \leftarrow A[j]$ ;

$C[A[j]]--$ ;

}

}

时间:  $T(n,k) = \theta(n+k) = \theta(n)$ , 如果  $k = \theta(n)$  时



## 第8章 线性时间的排序

8.1 排序的下界

8.2 计数排序

8.3 基数排序

8.4 桶排序



## 8.3 基数排序

- 算法
- 一些讨论

# 基数排序算法

- 假定  $A[1..n]$  是非负整数，用  $k$  进制表示为不超过  $d$  位数。
- 算法：  
RadixSort( $A, d$ )  
{ for  $i \leftarrow 1$  to  $d$  do  
    使用稳定的排序算法对  $A$  的第  $i$  位排序； // 如计数排序  
}
- 时间： $T(n) = \theta(d(n+k))$  //  $k$  为基， $d$  为位数  
     $= \theta(n)$  // 如果  $k = \theta(n)$  且  $d$  是常数
- 示例：Fig. 8.3



# 一些讨论

- $d$ 不为常数，基数排序算法还是线性时间吗？  
设 $n$ 个整数的取值范围是 $0 \sim n^c$ ， $c$ 是整常数， $c \geq 1$   
对于十进制整数， $n^c$ 需要的位数 $d = \lfloor \log_{10} n^c \rfloor + 1 \approx \log_{10} n$   
 $\therefore T(n) = \theta(d(n+k)) = \theta(n \log n)$  //  $k$ 为10  
因此，不是线性时间排序
- 算法何时为线性时间？
  - Idea: 只要使 $d$ 变为常数， $k$ 变大到与 $n$ 同阶
  - How to do:  
选基 $k=n$ ，则 $n^c$ 的位数为 $\log_n n^c = c = d$   
 $\therefore d=c, k=n$   
 $\therefore T(n) = \theta(n)$





## 第8章 线性时间的排序

8.1 排序的下界

8.2 计数排序

8.3 基数排序

8.4 桶排序



## 8.4 桶排序

- 基本思想
- 示例
- 算法描述

# 基本思想和示例

- 假定：输入是均匀分布在 $[0,1)$ 上的实数。
- 基本思想：
  - ①  $[0,1)$ 划分为 $[0,1/n), [1/n, 2/n), \dots, [k/n, (k+1)/n), \dots, [(n-1)/n, 1)$   $n$ 个大小相等的子区间，每个子区间看作一个桶；
  - ② 将 $n$ 个元素分配到桶中；
  - ③ 对每个桶里的元素进行排序，依次连接桶；
- 示例：Fig. 8.4

# 桶排序算法 (1)

- 输入:  $0 \leq A[1..n] < 1$
- 辅助数组:  $B[0..n-1]$  是一个指针数组, 指向每个桶(链表)
- 关键字映射:  
由于  $0 \leq A[i] < 1$ , 必须将  $A[i]$  映射到  $0, 1, \dots, n-1$  上  
 $\therefore [0, 1) \rightarrow [0, n)$  // 通过函数  $nA[i]$   
即  $k \leq nA[i] < k+1$  // 存在  $k$   
 $\therefore$  桶号  $k = \lfloor nA[i] \rfloor$  // 映射函数



# 桶排序算法 (2)

- 算法描述:

BucketSort(A)

{  $n \leftarrow \text{length}[A]$ ;

  for  $i \leftarrow 1$  to  $n$  do   //扫描A

    将  $A[i]$  插入到链表  $B[\lfloor nA[i] \rfloor]$  中;

  for  $i \leftarrow 0$  to  $n-1$  do

    用插入排序将  $B[i]$  排序;

  将  $B[0], B[1], \dots, B[n-1]$  连接起来;

}

Time

$\theta(n)$

$\theta(n)^*$

$\theta(n)$

\*注:  $\because n$  个数是均匀分布在  $[0,1)$  中,

$\therefore$  每个桶中大约只有一个数,  $\therefore$  时间为  $O(1)$

- 期望时间分析: 略

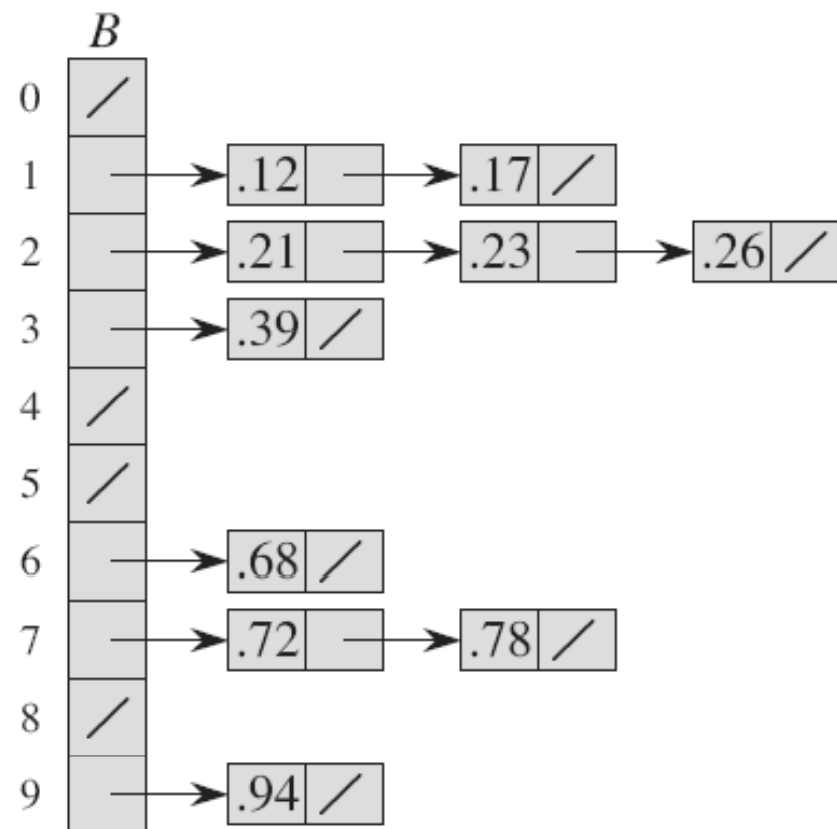


# 桶排序算法 (3)

- 示例：

A	
1	.78
2	.17
3	.39
4	.26
5	.72
6	.94
7	.21
8	.12
9	.23
10	.68

(a)



(b)



# End of Chap8

