

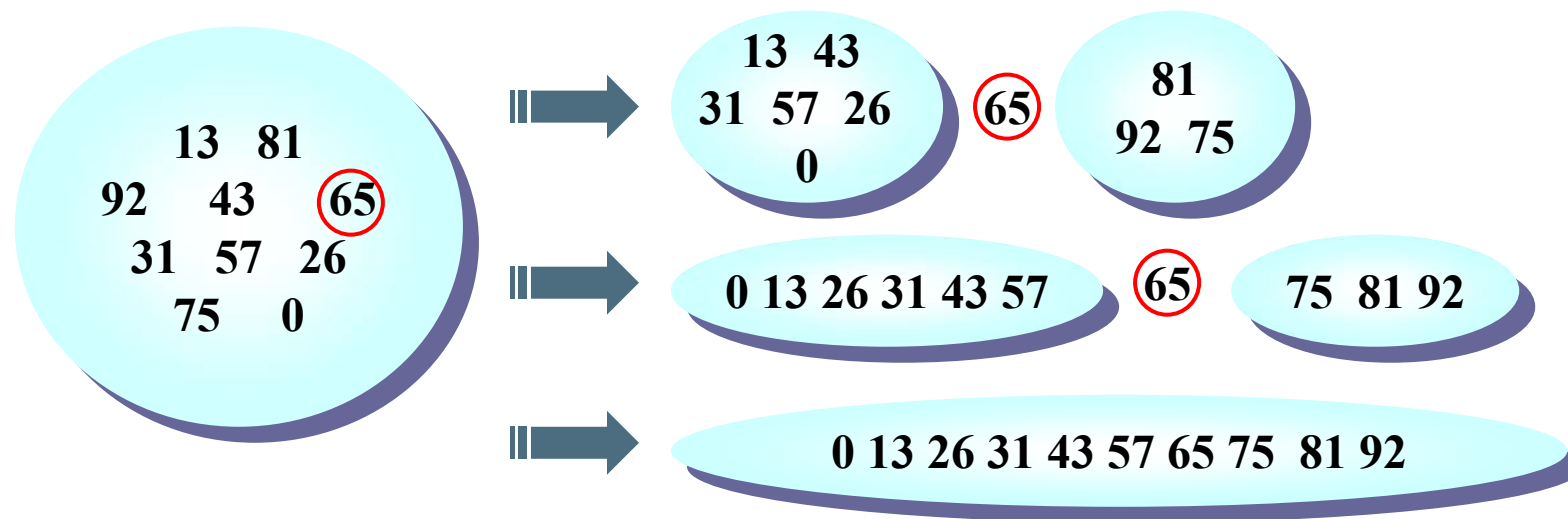
第十讲 排序（下）

浙江大学 陈 越

10.1 快速排序

算法概述

■ 分而治之



算法概述

什么是快速排序算法的最好情况？

每次正好中分 $\rightarrow T(N) = O(M \log N)$

■ 分而治之

```
void Quicksort( ElementType A[], int N )
{
    ? pivot = 从A[]中选一个主元;
    ? 将S = { A[] \ pivot } 分成2个独立子集:
        A1={ a∈S | a ≤ pivot } 和
        A2={ a∈S | a ≥ pivot };
    A[] = Quicksort(A1,N1) ∪
        {pivot} ∪
        Quicksort(A2,N2);
}
```

选主元

- 令 $\text{pivot} = A[0]$?

① 2 3 4 5 6 N-1 N
② 3 4 5 6 N-1 N
3 4 5 6 N-1 N



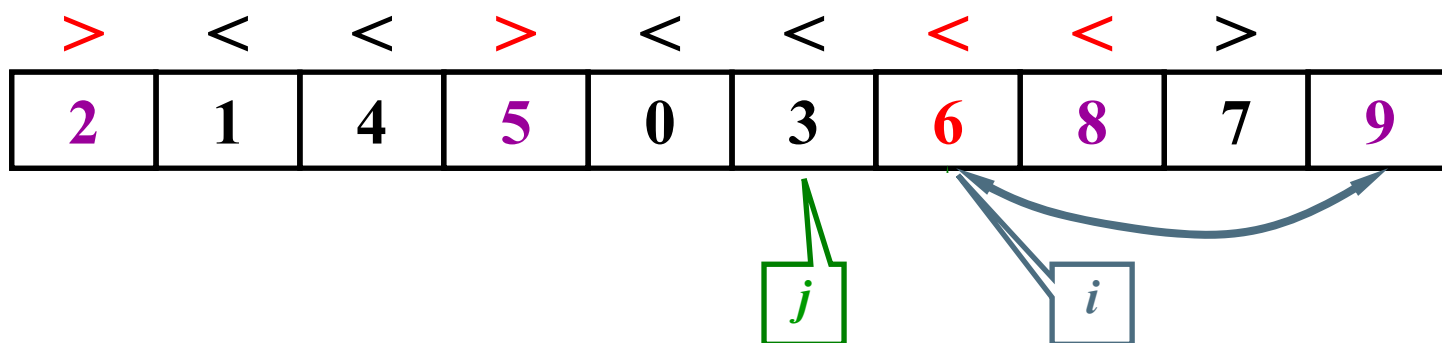
$$\begin{aligned} T(N) &= O(N) + T(N-1) \\ &= O(N) + O(N-1) + T(N-2) \\ &= O(N) + O(N-1) + \dots + O(1) \\ &= O(N^2) \end{aligned}$$

选主元

- 随机取 **pivot**? **rand()** 函数不便宜啊!
- 取头、中、尾的中位数
 - 例如 8、12、3的中位数就是8
 - 测试一下**pivot**不同的取法对运行速度有多大影响?

```
ElementType Median3( ElementType A[], int Left, int Right )
{
    int Center = ( Left + Right ) / 2;
    if ( A[ Left ] > A[ Center ] )
        Swap( &A[ Left ], &A[ Center ] );
    if ( A[ Left ] > A[ Right ] )
        Swap( &A[ Left ], &A[ Right ] );
    if ( A[ Center ] > A[ Right ] )
        Swap( &A[ Center ], &A[ Right ] );
    /* A[ Left ] <= A[ Center ] <= A[ Right ] */
    Swap( &A[ Center ], &A[ Right-1 ] ); /* 将pivot藏到右边 */
    /* 只需要考虑 A[ Left+1 ] ... A[ Right-2 ] */
    return A[ Right-1 ]; /* 返回 pivot */
}
```

子集划分



- 如果有元素正好等于 **pivot** 怎么办?
 - 停下来交换? ✓
 - 不理它, 继续移动指针?

小规模数据的处理

■ 快速排序的问题

- 用递归.....
- 对小规模的数据（例如 N 不到100）可能还不如插入排序快

■ 解决方案

- 当递归的数据规模充分小，则停止递归，直接调用简单排序（例如插入排序）
- 在程序中定义一个**Cutoff**的阈值 —— 课后去实践一下，比较不同的**Cutoff**对效率的影响

算法实现

```
void Quicksort( ElementType A[], int Left, int Right )
{
    if ( Cutoff <= Right-Left ) {
        Pivot = Median3( A, Left, Right );
        i = Left;  j = Right - 1;
        for( ; ; ) {
            while ( A[ ++i ] < Pivot ) { }
            while ( A[ --j ] > Pivot ) { }
            if ( i < j )
                Swap( &A[i], &A[j] );
            else break;
        }
        Swap( &A[i], &A[ Right-1 ] );
        Quicksort( A, Left, i-1 );
        Quicksort( A, i+1, Right );
    }
    else
        Insertion_Sort( A+Left, Right-Left+1 );
}
```

```
void Quick_Sort(ElementType A[],int N)
{
    Quicksort( A, 0, N-1 );
}
```

第十讲 排序（下）


浙江大学 陈 越

10.2 表排序

算法概述

■ 间接排序

- 定义一个指针数组作为“表”（table）




A	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
key	f	d	c	a	g	b	h	e
table	3	5	2	1	7	0	4	6

如果仅要求按顺序输出，则输出：

$A[\text{table}[0]]$, $A[\text{table}[1]]$,, $A[\text{table}[N-1]]$

物理排序

- N个数字的排列由若干个独立的环组成



A	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
key	f	d	c	a	g	b	h	e
table	3	5	2	1	7	0	4	6

Temp = f

如何判断一个环的结束？

`if (table[i] == i)`

复杂度分析

- 最好情况：初始即有序
- 最坏情况：
 - 有 $\lfloor N/2 \rfloor$ 个环，每个环包含2个元素
 - 需要 $\lfloor 3N/2 \rfloor$ 次元素移动

$T = O(mN)$ ， m 是每个A元素的复制时间。

第十讲 排序（下）

浙江大学 陈 越

10.3 基数排序

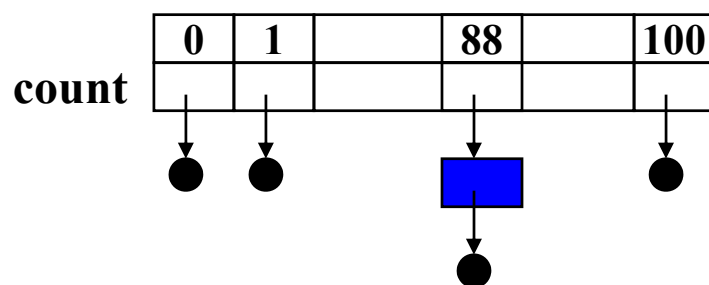
举个例子



桶排序



假设我们有 N 个学生，他们的成绩是0到100之间的整数（于是有 $M = 101$ 个不同的成绩值）。如何在线性时间内将学生按成绩排序？



如果 $M \gg N$
该怎么办？

```
void Bucket_Sort(ElementType A[], int N)
{
    count[] 初始化;
    while (读入1个学生成绩grade)
        将该生插入count[grade]链表;
    for ( i=0; i<M; i++ ) {
        if ( count[i] )
            输出整个count[i]链表;
    }
}
```

$$T(N, M) = O(M + N)$$

基数排序



假设我们有 $N = 10$ 个整数，每个整数的值在0到999之间（于是有 $M = 1000$ 个不同的值）。还有可能在线性时间内排序吗？

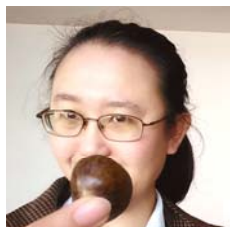
输入序列: 64, 8, 216, 512, 27, 729, 0, 1, 343, 125

$$T = O(P(N+B))$$

用“次位优先” (Least Significant Digit)

Bucket	0	1	2	3	4	5	6	7	8	9
Pass 1	0	1	512	343	64	125	216	27	8	729
Pass 2	0	512	125		343		64			
	1	216	27							
	8		729							
Pass 3	0	125	216	343		512		729		
	1									
	8									
	27									
	64									

多关键字的排序



一副扑克牌是按2种关键字排序的

K^0 [花色]

♣ < ♦ < ♥ < ♠

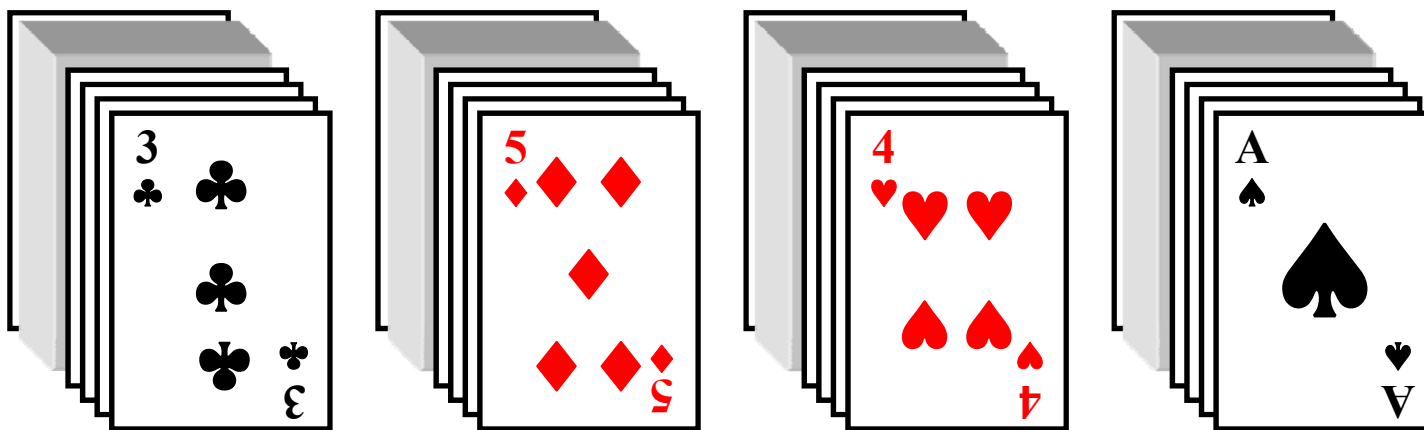
K^1 [面值]

2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < 10 < J < Q < K < A

有序结果:

2♣ ... A♣ 2♦ ... A♦ 2♥ ... A♥ 2♠ ... A♠

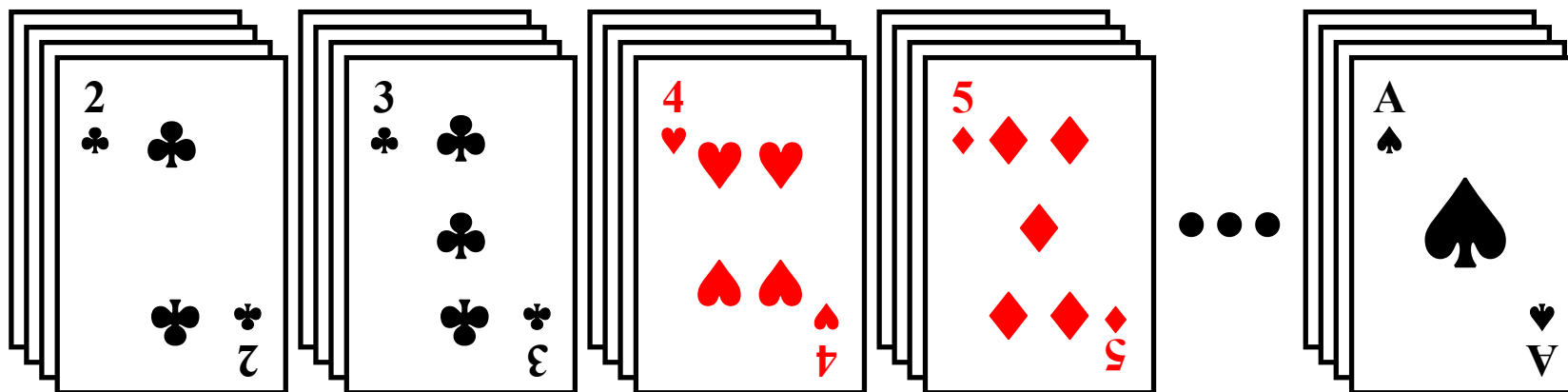
☞ 用“主位优先” (**M**ost **S**ignificant **D**igit) 排序: 为花色建4个桶



在每个桶内分别排序, 最后合并结果。

多关键字的排序

- 用“次位优先” (**Least Significant Digit**) 排序：为面值建13个桶



- 将结果合并，然后再为花色建4个桶
- 问题：LSD任何时候都比MSD快吗？

第十讲 排序（下）

浙江大学 陈 越

10.4 排序算法的比较

排序方法	平均时间复杂度	最坏情况下时间复杂度	额外空间复杂度	稳定性
简单选择排序	$O(N^2)$	$O(N^2)$	$O(1)$	不稳定
冒泡排序	$O(N^2)$	$O(N^2)$	$O(1)$	稳定
直接插入排序	$O(N^2)$	$O(N^2)$	$O(1)$	稳定
希尔排序	$O(N^d)$	$O(N^2)$	$O(1)$	不稳定
堆排序	$O(N\log N)$	$O(N\log N)$	$O(1)$	不稳定
快速排序	$O(N\log N)$	$O(N^2)$	$O(\log N)$	不稳定
归并排序	$O(N\log N)$	$O(N\log N)$	$O(N)$	稳定
基数排序	$O(P(N+B))$	$O(P(N+B))$	$O(N+B)$	稳定