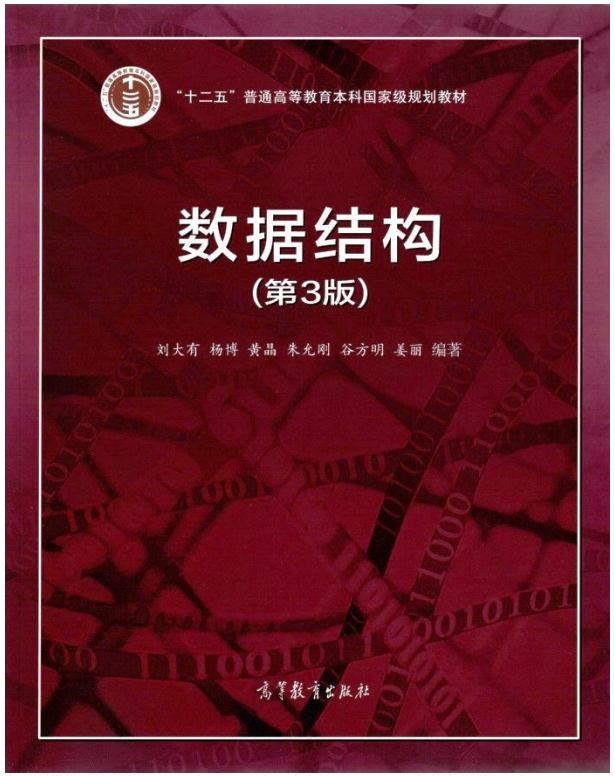
think.create.solve



平方阶排序算法

- 排序概念
- 直接插入排序
- 希尔排序
- 冒泡排序
- 直接选择排序

数据之结构之美算法之道

zhuyungang@jlu.edu.cn

Last updated on 2022.11


淘宝网
书包
搜 索
所有分类
找到相关宝贝 647157件
所有宝贝
人气
天猫
二手
1/100
常用：海外商品 货到付款 消费者保障 7天退换 正品保障 旺旺在线
默认排序
销量
信用
价格
总价
所在地
列表
大图
销量从高到低
信用从高到低
价格从低到高
价格从高到低
总价从低到高
总价从高到低
owbone双肩包 韩版背包男休闲旅行
电脑双肩背包运动包女新款书包
天猫 TMALL.COM
owbone唯然专卖店 和我联系
¥ 108.77
运费：0.01
信用卡
折扣
广东 广州
货到付款
月售5058笔
13735条评价
消费者保障
七天退换
正品保障
木村耀司 双肩包学生包书包旅行包双
肩背包电脑包男女韩版潮包
天猫 TMALL.COM
木村耀司旗舰店 和我联系
¥ 54.00
运费：0.00
信用卡
折扣
上海
代购
月售3704笔
8481条评价
消费者保障
七天退换
正品保障
韩版包袋日韩新款糖果色背包笔记本包
海绵隔层双肩糖果包书包男女
特价折扣包 和我联系
¥ 56.00
运费：12.00
信用卡
折扣
上海
月售140笔
368条评价
消费者保障
七天退换
24小时发货
2012迷彩耐克双肩包 nike背包 男女学
生书包 情侣包 日韩流行背包
¥ 75.60
运费：0.01
北京
月售81笔
21条评价
消费者保障

排序问题的基本概念

```
struct student {
    char Name[10];
    int Gender;
    int ID;
}
```

- 文件：给定待排序的 n 个数据对象，R_1，R_2，…，R_n，这些数据对象为记录，并称这 n 个记录的集合为一个文件；
- 关键词：通常数据对象包括多个属性域，可将其中的一个属性域作为排序的依据，称其为关键词域。上述 n个记录的关键词分别是 K_1，K_2，…，K_n；
- 排序：按规定的顺序，以关键词为依据，对一个文件中的诸记录进行排列的过程。

排序算法的度量指标:

- 时间开销(时间复杂性)：是衡量算法好坏的最重要标志。可用算法执行中关键词的比较次数与数据的移动次数来衡量。
- 空间复杂性：主要考察排序过程占用存储空间的大小。
- 排序算法的稳定性：如果两个对象R_i和R_j，其关键词相同，且在排序前R_i在R_j的前面，若排序后R_i仍在R_j的前面，则称该排序算法是稳定的，否则称这个排序算法是不稳定的。

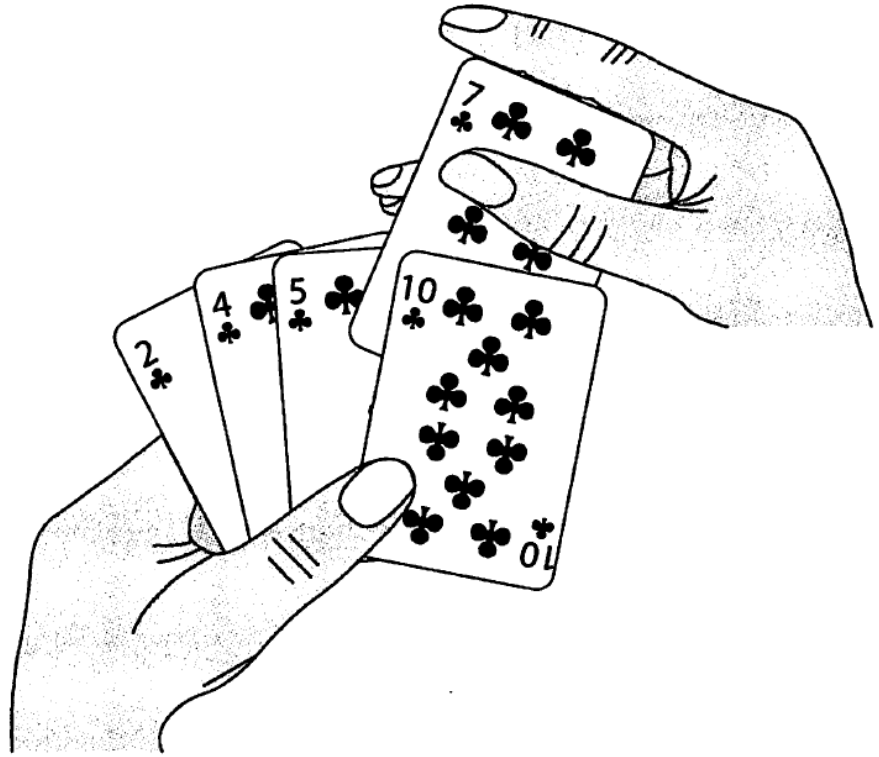
排序算法的分类

- 从存储设备角度：内排序和外排序。在排序过程中，所有待排序记录都在内存中，称为内排序；当输入文件所占存储空间大到计算机内存不能容纳，排序过程必需借助外存来完成，这时的排序就称为外排序。
- 按对关键词的操作
  可分为基于关键词比较的排序和分布排序。
- 按时间代价
  平方阶排序算法，它们一般都较简单，易于实现，但时间复杂性较高，平均时间复杂度为 $O(n^2)$；
  线性对数阶算法，以分治策略算法为主，平均时间复杂度为 $O(n\log n)$。

直接插入排序

直接插入排序的思想：

将一个记录插入到已排好序的有序表中，从而得到一个新的、记录数增1的有序表。



例：

原有序表：(9　15　23　28　37) 20

找插入位置：(9　15 ↑ 23　28　37) 20

新有序表：(9　15　20　23　28　37)

排序方法:

第一步：R_1 //使第1个元素有序

第二步：R_1 , R_2 //使前2个元素有序

第三步：R_1 , R_2, R_3 //使前3个元素有序

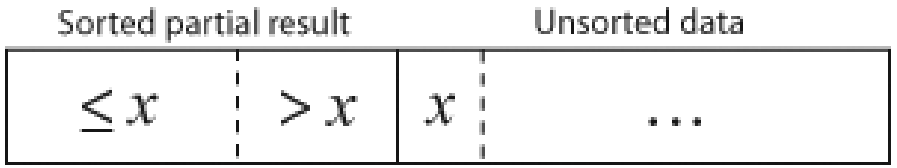
…… ……

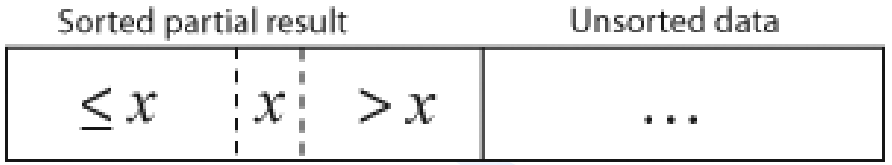
第j步：R_1，R_2，…，R_{j-1}, R_j

…… ……

第n步：R_1，R_2，…，R_{n-1} , R_n

直接插入排序算法





算法InsertSort (R, n)

FOR j=2 **TO** n **DO**

// 每次循环将R_j插入到R_1，…，R_{j-1}中

(　　i←j-1.　// 指针i扫描R_1，…，R_{j-1}

　　$K \leftarrow K_j$. $R \leftarrow R_j$.

　　WHILE i>0 **AND** $K_i > K$ **DO**

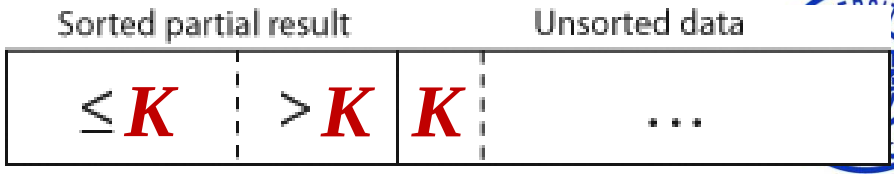
　　// 从右往左找第1个≤R_j的元素

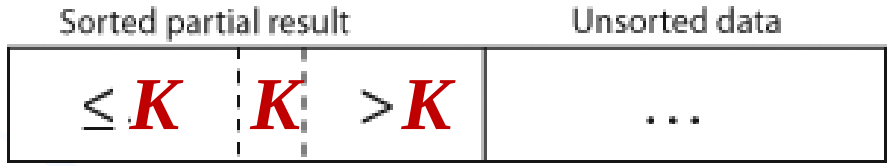
　　(　$R_{i+1} \leftarrow R_i$.　i ← i-1　) .

　　$R_{i+1} \leftarrow R$.

)

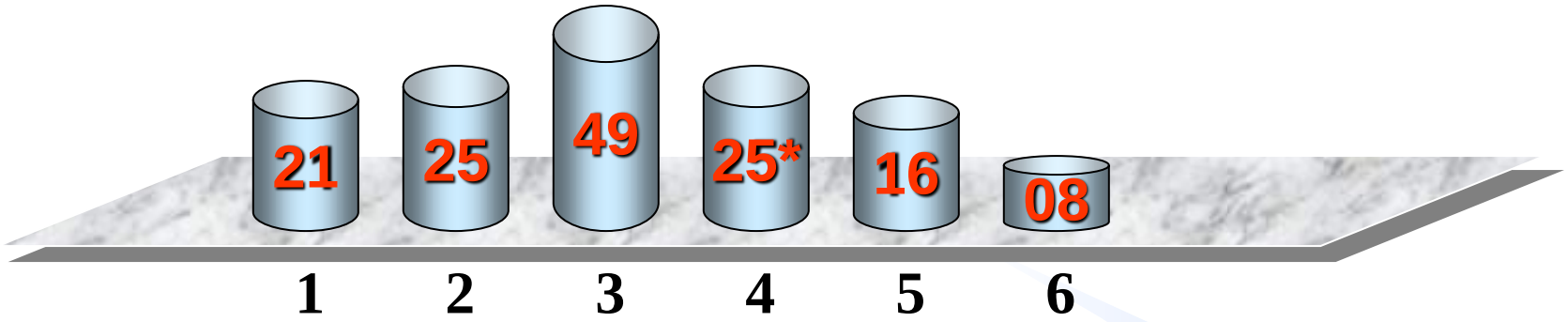
直接插入排序



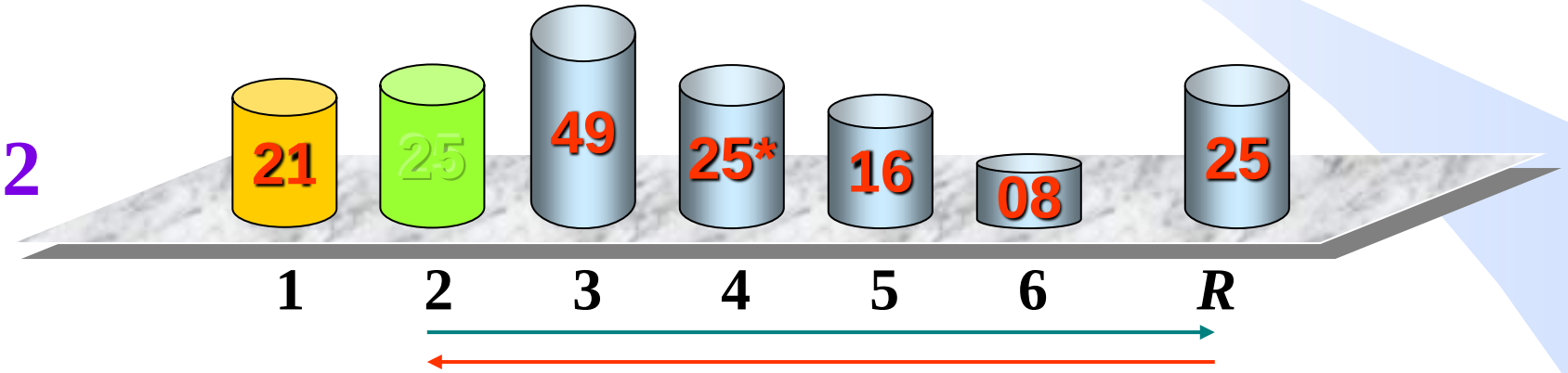



```
void InsertSort(int R[], int n){
     for(int j=2;j<=n;j++){
          //R[1]..... R[j-1] ← R[j]
          int i=j-1; //指针i扫描R[1]..... R[j-1]
          int K=R[j];
          while (i>0  &&  R[i]>K){
               //从右往左找第1个≤ R[j]的元素
               R[i+1]=R[i];
               i--;
          }
          R[i+1]=K;
     }
}
```

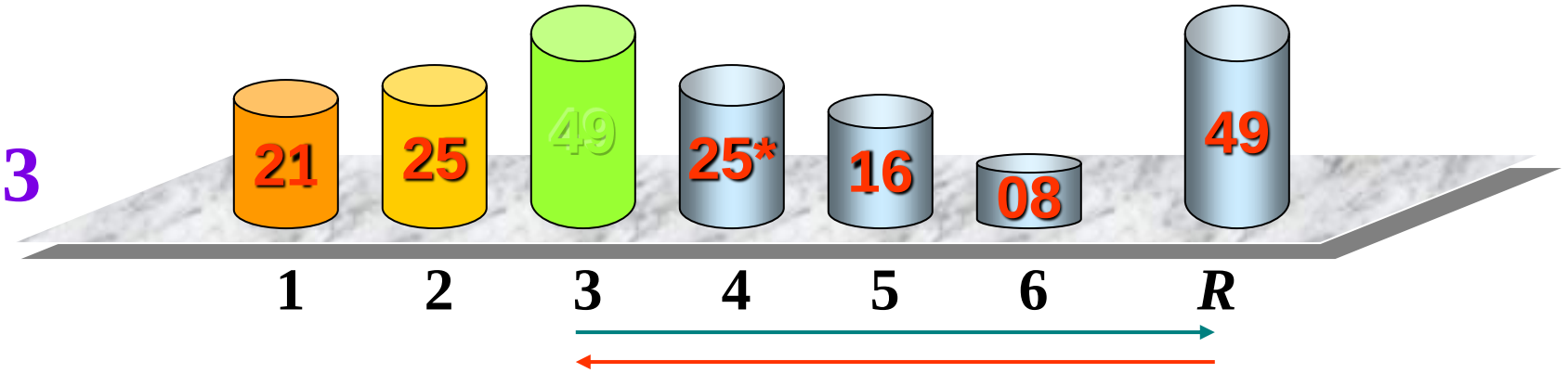
各趟排序结果

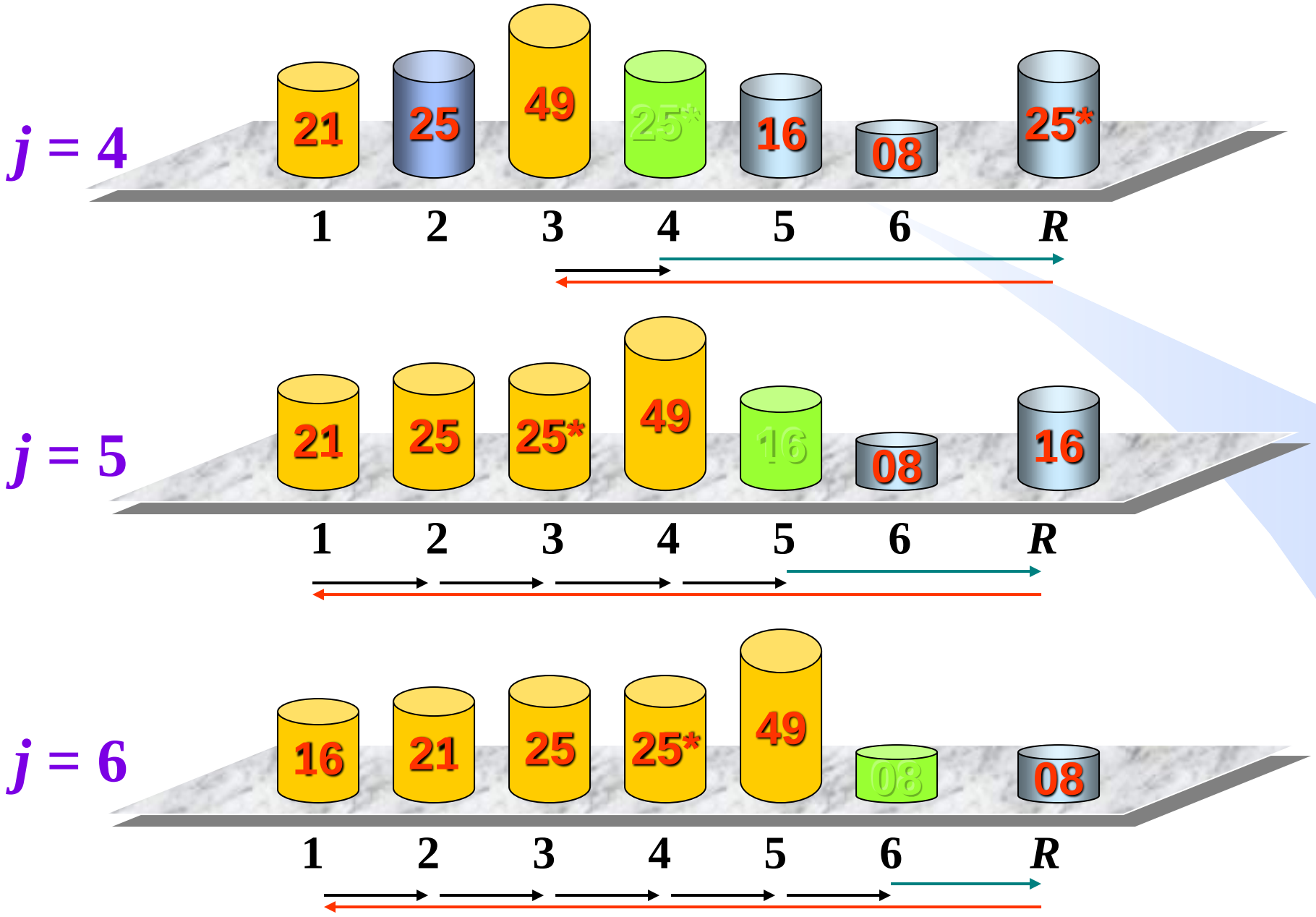

21
25
49
25*
16
08
1
2
3
4
5
6

$j = 2$

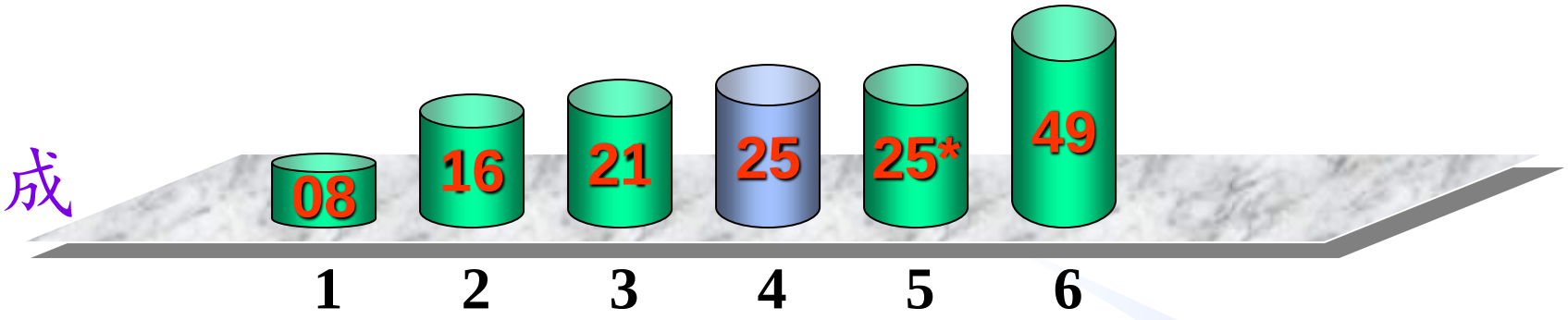

21
25
49
25*
16
08
25
1
2
3
4
5
6
R

$j = 3$


21
25
49
25*
16
08
49
1
2
3
4
5
6
R

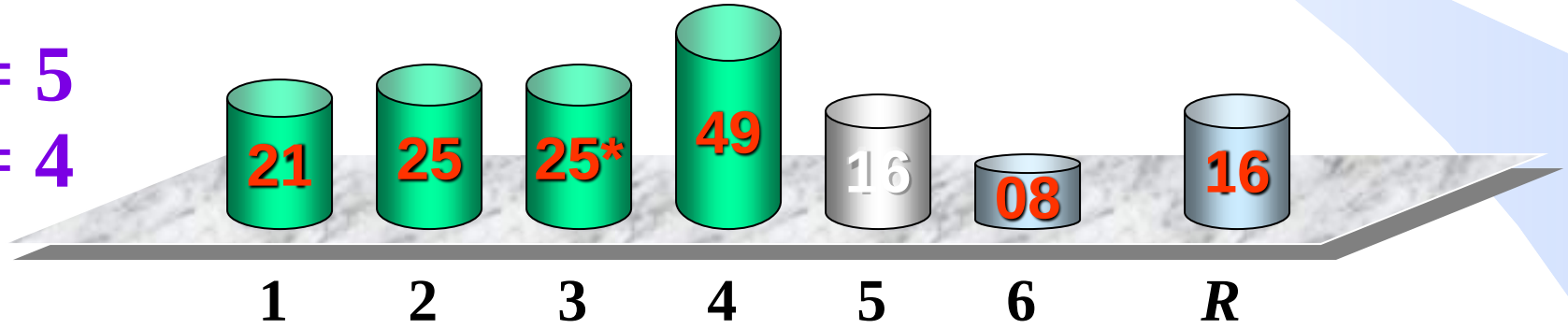

j = 4
21
25
49
25*
16
08
25*
1
2
3
4
5
6
R
j = 5
21
25
25*
49
16
08
16
1
2
3
4
5
6
R
j = 6
16
21
25
25*
49
08
08
1
2
3
4
5
6
R

完成

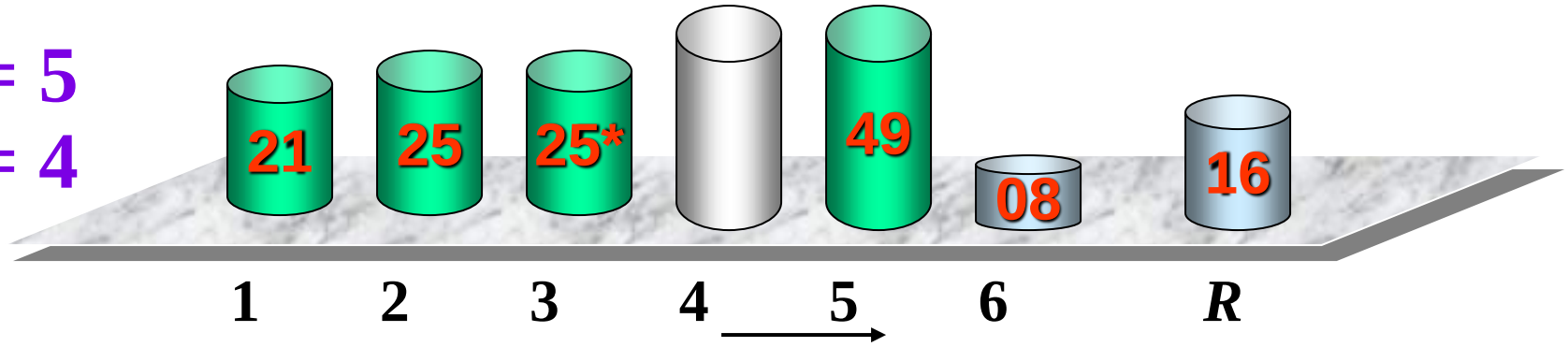


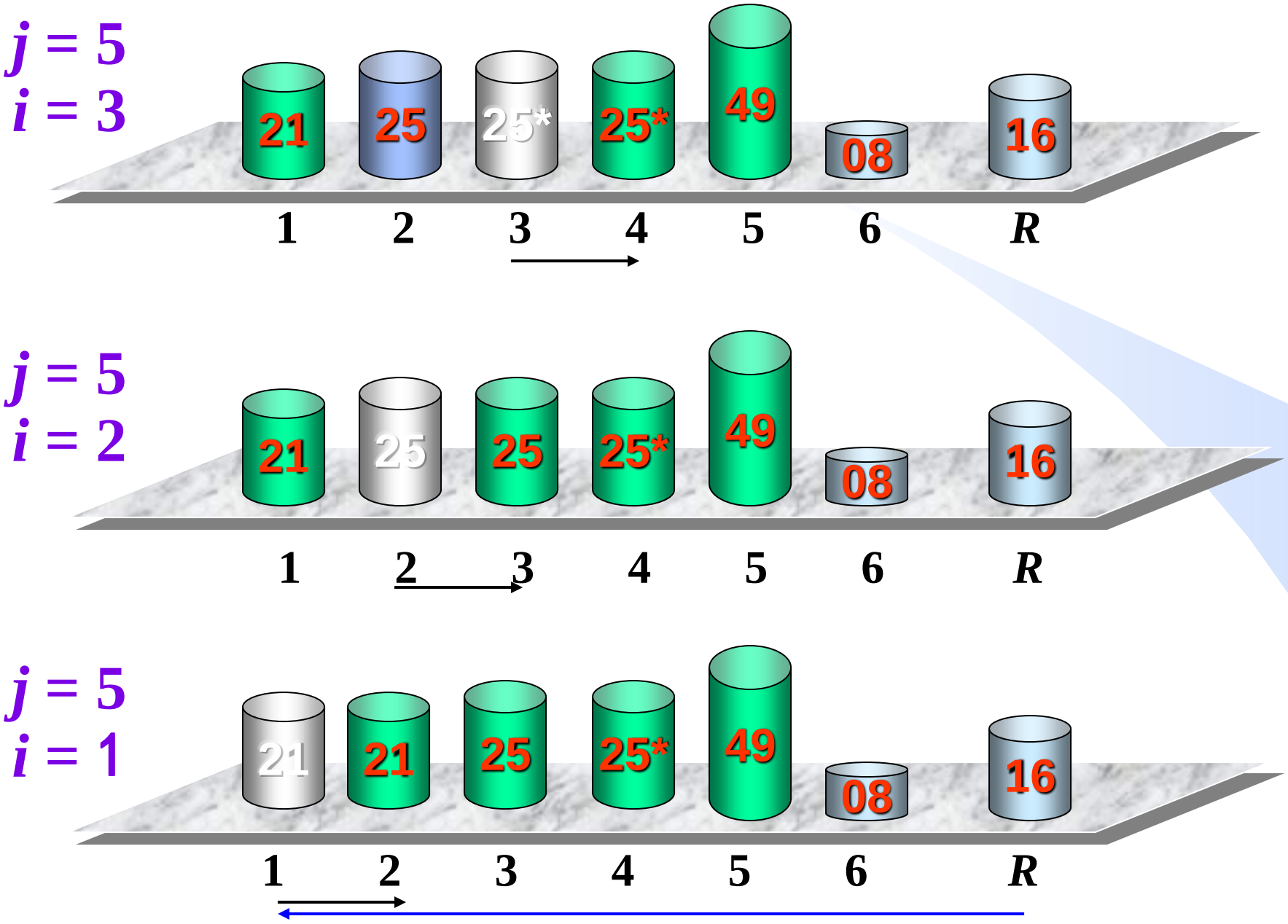
$j = 5$ 时的排序过程

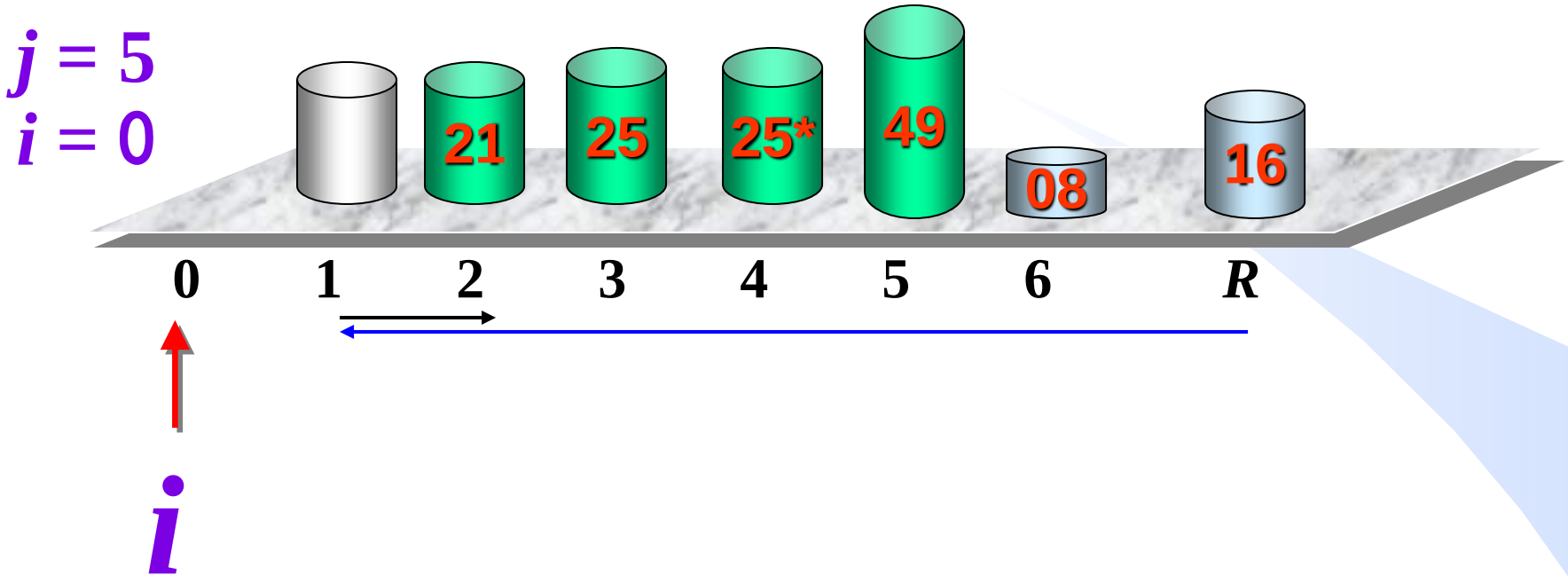
$j = 5$
$i = 4$



$j = 5$
$i = 4$




j = 5
i = 3
21
25
25*
25*
49
08
16
1
2
3
4
5
6
R
j = 5
i = 2
21
25
25
25*
49
08
16
1
2
3
4
5
6
R
j = 5
i = 1
21
21
25
25*
49
08
16
1
2
3
4
5
6
R


j = 5
i = 0
21
25
25*
49
08
16
0
1
2
3
4
5
6
R
i

改进的插入排序算法

```
算法InsertSortA( R, s, e )
/* s , e 分别为文件记录序列的左端和右端下标，引入虚拟记录R_{s-1}，K_{s-1} ≤ min{ K_i | s ≤ i ≤ e }*/
FOR j=s+1 TO e DO
(    i←j-1 .   K←K_j .   R←R_j .
     WHILE K_i > K DO
     (    R_{i+1}←R_i .   i←i-1.    ) .
     R_{i+1}←R.
) ▌
```

往左扫描一定能找到比R_j小的元素

```
WHILE i>0 AND K_i > K DO
```

最好时间复杂度

- 在算法InsertSortA中，设$s=1$，$e=n$．分析该算法时间复杂性的关键在于计算关键词的比较次数。
- 最好情况下，排序前记录已按关键词递增排序。
- 每趟只需与前面的有序子数组的最后一个元素的关键词比较1次，因此总的关键词比较次数为$n-1$，即最好情况下算法的时间复杂度为O(n)。

最坏时间复杂度

- 最坏情况下，待排序文件在排序前已按逆序排列。
- 使得每趟的FOR循环，记录R_j前面所有记录的关键词都比R_j的关键词大，需做j次关键词比较，使R_j移动到最前端，故此时总的关键词比较次数为

$$\sum_{j=2}^{n} j = (n-1)(n+2)/2$$

- R_j需要与其前面的j个元素都进行比较。

平均时间复杂度

- 假定关键词的分布是随机的。在每次FOR循环中，R_j被插入到R_1，R_2，…，R_{j-1}之中，具体来说，R_j可能被插入到

  “[1] R_1 [2] R_2 [3] R_3 [4]…R_{j-2}[j–1] R_{j-1} [j]”

  中的[1]到[j]的某个位置。

- 由于关键词是随机分布的，R_j被插入到这j个位置的概率相等，均为$1/j$.

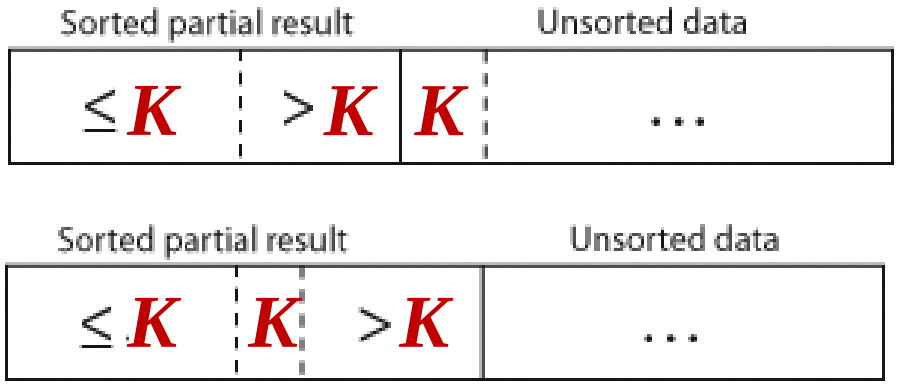
- R_j被插入到位置[j]到[1]所对应的关键词的比较次数分别为1，2，3，…，j.故关键词的总比较次数为：

$$\sum_{j=2}^{n}\frac{1}{j}(1+2+\ldots+j)=\sum_{j=2}^{n}\frac{1}{j}\frac{j(j+1)}{2}=\sum_{j=2}^{n}\frac{(j+1)}{2}=\frac{(n-1)(n+4)}{4}$$

	时间复杂性
最好	O(n)
平均	$O(n^2)$
最坏	$O(n^2)$
稳定性	稳定

进一步改进

- 查找R_j插入位置时，改用折半查找，可减少元素比较次数。
- 但元素移动次数仍为$O(n)$，故时间复杂度的阶不会降低。



练习

对同一序列分别进行折半插入排序和直接插入排序，两者之间可能的不同之处是_____。【考研题全国卷】

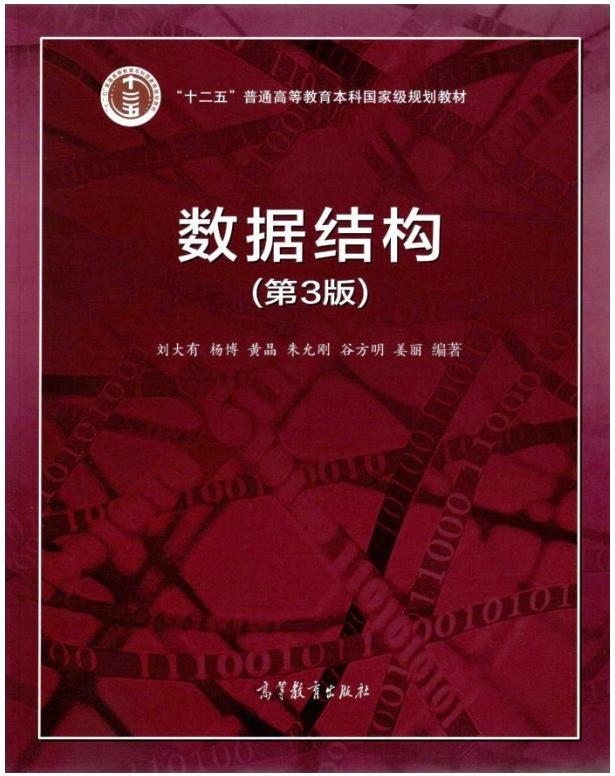
A.排序的总趟数　　B.元素的移动次数

C.使用辅助空间的数量　　D.元素的比较次数

直接插入排序算法

- 优点：算法的执行过程相当清晰，并且书写容易。
- 缺点：平均时间复杂性为$O(n^2)$。
- 稳定性：直接插入排序是稳定的排序方法。
- 最好情况：被排序文件已排好序时。
- 在“短文件”、“基本有序”时速度快。
- 适合增量数据环境。
- 空间复杂度：$O(1)$。

think. create. solve



平方阶排序算法

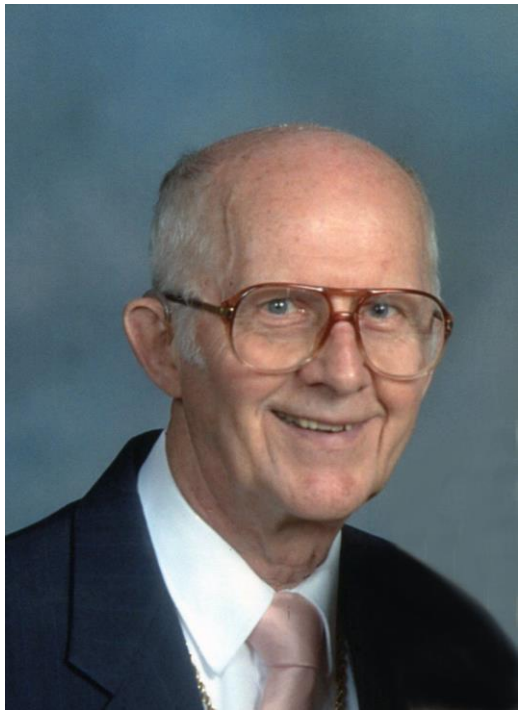
- 排序概念
- 直接插入排序
- **希尔排序**
- 冒泡排序
- 直接选择排序

数据之法 结构之美 算法之道

zhuyungang@jlu.edu.cn

Last updated on 2022.11

希尔排序



Donald Shell
（1924-2015）
美国辛辛那提大学数学博士
通用电气公司

希尔排序

希尔排序（渐减增量排序）思想：

- 1959年Donald Shell提出希尔排序算法，是对直接插入算法的改进。
- 把记录按下标的一定增量分组。
- 直接插入排序算法在“短文件”、“基本有序”时速度快，希尔排序充分利用了这两个特性。

假设对 n 个记录排序，增量为d，将待排序记录分成如下d组：

{ R[1]，R[1+d]，R[1+2d]，R[1+3d]，R[1+4d]，… }

{ R[2]，R[2+d]，R[2+2d]，R[2+3d]，R[2+4d]，…}

…

{ R[d]，R[d+d]，R[d+2d]，R[d+3d]，R[d+4d]，… }

✓每组相邻元素的下标间隔d

✓一共d组

- 对每组使用直接插入法排序；
- 把增量值d减小，再将文件分成多个组，每组使用直接插入法排序。以此往复。
- 随着增量值d逐渐减小，每组包含的元素越来越多，当增量值减少到 1 时，整个文件恰被分成一组，算法终止。

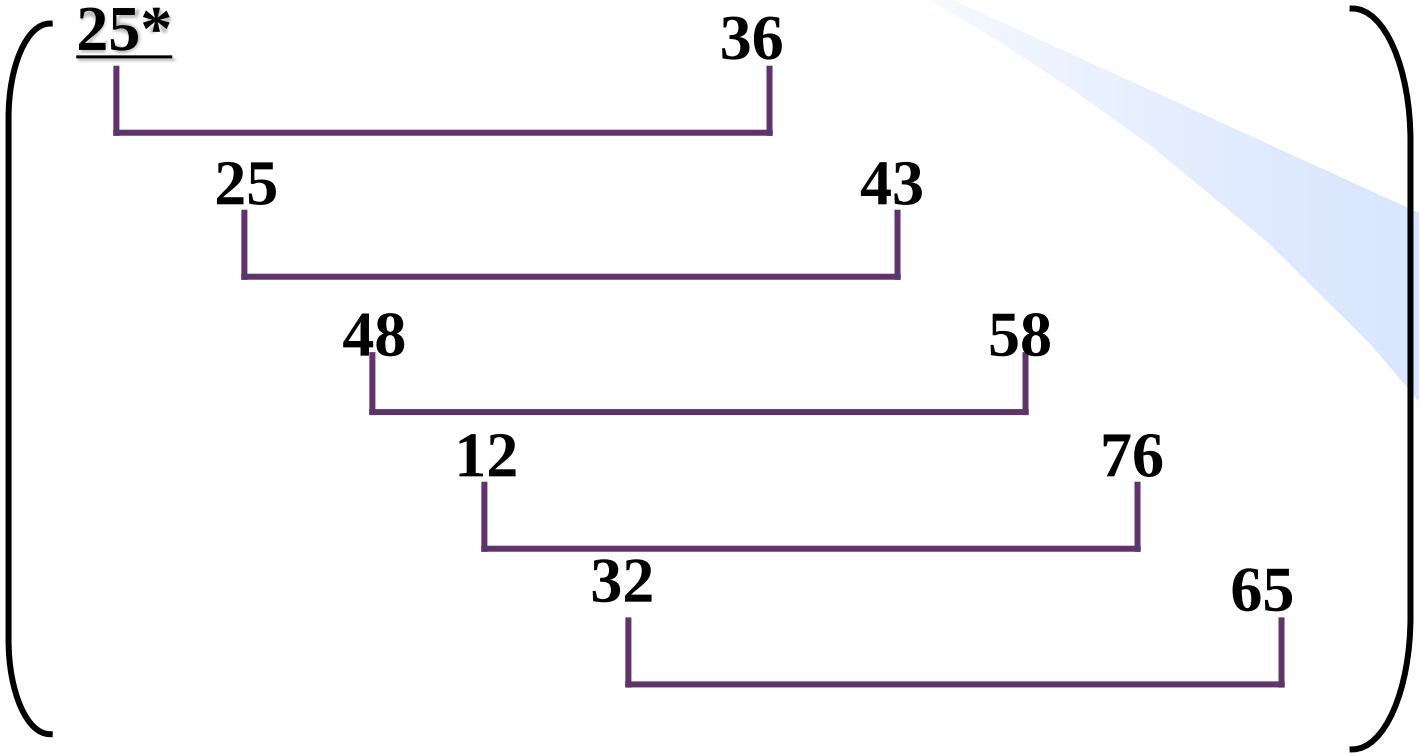
希尔排序增量d的一种取法（以n=10为例）：

$$d_1=\lfloor n/2\rfloor=\lfloor 10/2\rfloor=5$$
$$d_2=\lfloor d_1/2\rfloor=\lfloor 5/2\rfloor=2$$
$$d_3=\lfloor d_2/2\rfloor=\lfloor 2/2\rfloor=1$$

[例] 将十个数进行希尔排序的示例。

下标	1	2	3	4	5	6	7	8	9	10
	36	25	48	12	65	25*	43	58	76	32

$d_1=5$



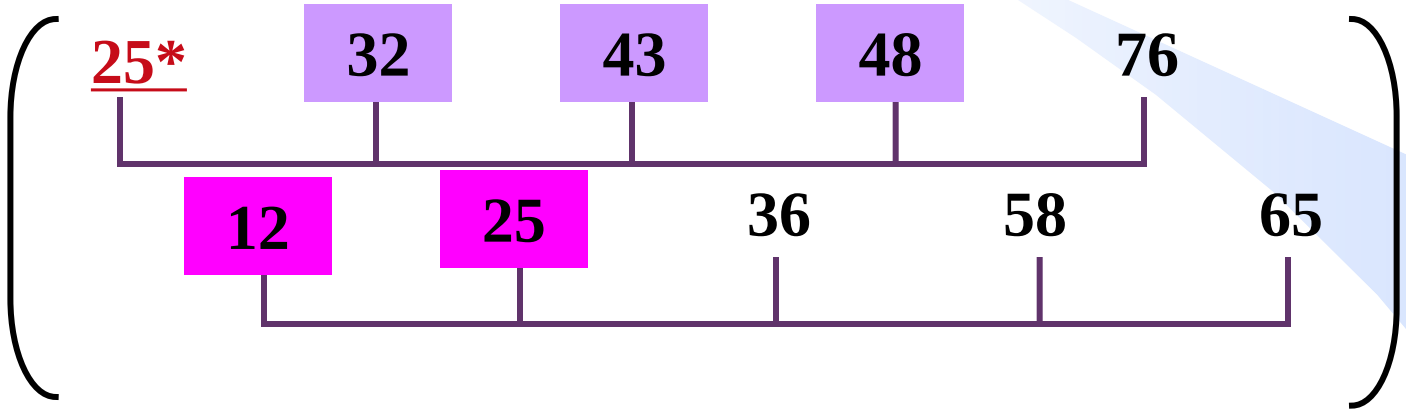
第一趟排序结果

25*	25	48	12	32	36	43	58	76	65

[例] 将十个数进行希尔排序的示例。

下标	1	2	3	4	5	6	7	8	9	10
	25*	25	48	12	32	36	43	58	76	65

$d_2=2$



第二趟排序结果

25*	12	32	25	43	36	48	58	76	65

第三趟排序结果

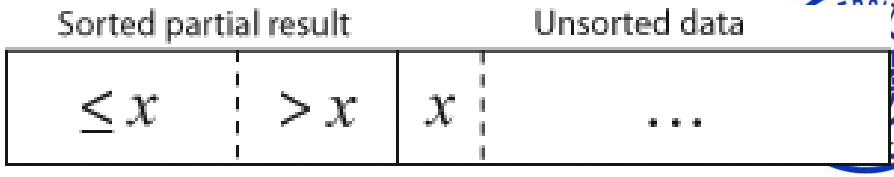
12	25*	25	32	36	43	48	58	65	76

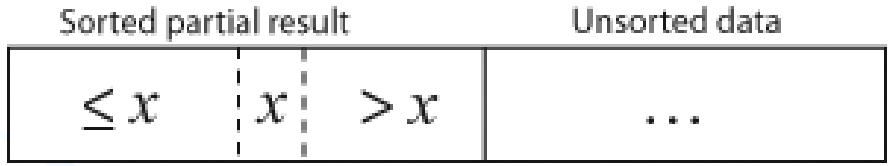
$d_3=1$

希尔排序的原理

- 开始时增量值较大，每组中的记录较少，插入排序速度较快；
- 算法执行过程中，随着增量值逐渐变小，每组中记录个数逐渐变多，由于前面工作的基础，大多数记录已基本有序，所以插入排序速度仍然很快。

回顾：直接插入排序






```
void InsertSort(int R[], int n){
    for(int j=2;j<=n;j++){
        //R[1]..... R[j-1] ← R[j]
        int i=j-1; //指针i扫描R[1]..... R[j-1]
        int K=R[j];
        while (i>0  &&  R[i]>K){
            //从右往左找第1个≤ R[j]的元素
            R[i+1]=R[i];
            i--;
        }
        R[i+1]=K;
    }
}
```

希尔排序

```
void ShellSort(int R[], int n){ //对R[1]…R[n]递增排序
    for(int d=n/2; d>0; d/=2)   //d为增量值
        for(int j=d+1; j<=n; j++){
            //.....R[j-3d], R[j-2d], R[j-d] ← R[j]
            int i=j-d; //指针i扫描R[j-d],R[j-2d],R[j-3d]..
            int K=R[j];
            while (i>0 && R[i]>K){
                //在本组中从右往左找第1个≤ R[j]的元素
                R[i+d]=R[i];
                i-=d;
            }
            R[i+d]=K;
        }
}
```

{ R[1], R[1+*d*], R[1+2*d*], R[1+3*d*], R[1+4*d*], … }
{ R[2], R[2+*d*], R[2+2*d*], R[2+3*d*], R[2+4*d*], …}
…
{ R[*d*], R[*d*+*d*], R[*d*+2*d*], R[*d*+3*d*], R[*d*+4*d*], … }

增量的取法

- 增量可选如下的任意正整数序列：

$$n > d_1 > d_2 > \ldots > d_t = 1$$

- Shell 提出取$d_1 = \lfloor n/2 \rfloor$, $d_2 = \lfloor d_1/2 \rfloor$, 直到$d_t = 1$.
- Knuth 提出取$d_{j+1} = \lfloor d_j/3 \rfloor + 1$.
- 取奇数
- 各增量互质

练习

用希尔排序对一个数据序列进行排序时，若第一趟排序结果为9，1，4，13，7，8，20，23，15，则该趟排序采用的增量可能是_____。【考研题全国卷】

A. 2　　B. 3　　C. 4　　D. 5

希尔排序的稳定性

- Shell排序算法是不稳定的排序算法。

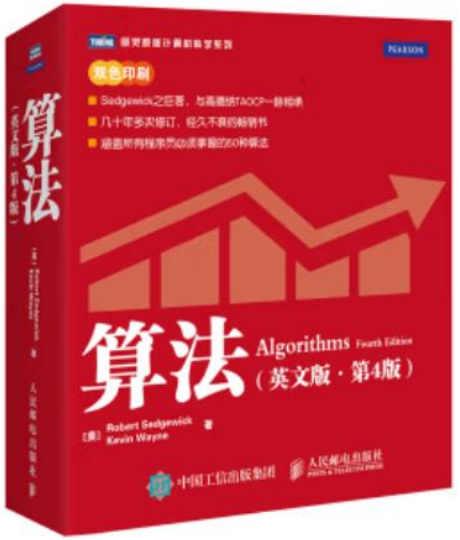
希尔排序的时间复杂度

- 算法的时间复杂度与所选取的增量有很大关系。
- 增量是“每次除以2递减”时，最坏情况时间复杂度为$O(n^2)$.
- Papernov和Stasevich提出$\{2^k-1, 2^{k-1}-1,\ldots,15, 7, 3, 1\}$作为增量序列，时间复杂度可达$O(n^{1.5})$.
- Pratt提出增量选取形如2^p3^q的数，如…12, 6, 3, 2, 1，则时间复杂度为 $O(n(logn)^2)$.
- Sedgewick提出增量序列$h_s = \begin{cases} 9\cdot 2^s - 9\cdot 2^{\frac{s}{2}} + 1, & x\text{是偶数} \\ 8\cdot 2^s - 6\cdot 2^{\frac{s+1}{2}} + 1, & x\text{是奇数} \end{cases}$，

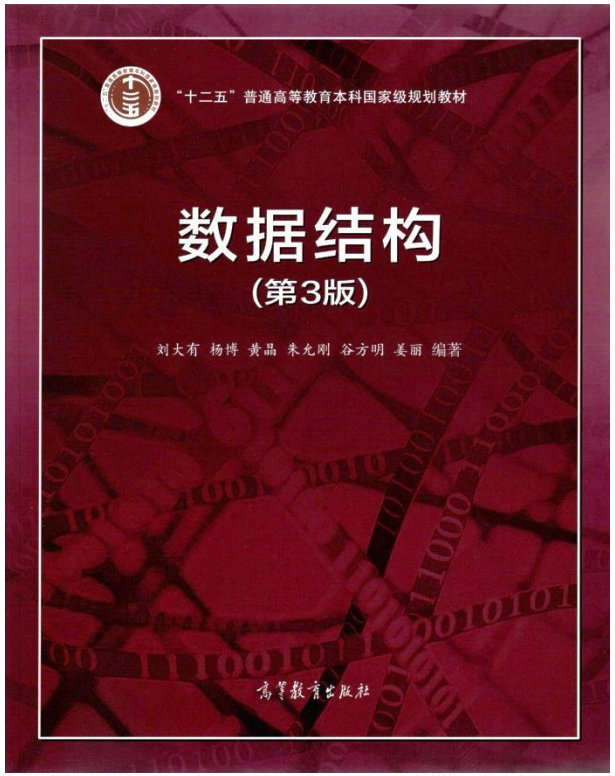
  即$\{\ldots,109, 41, 19, 5, 1\}$，时间复杂度可达$O(n^{4/3})$.

希尔排序的时间复杂度

- Sedgewick提出{$4^{j+1}+3\cdot 2^{j}+1,\ldots,77, 23, 8, 1$}作为增量序列，时间复杂度可达$O(n^{4/3})$.
- 一般认为希尔排序的平均时间复杂度介于$O(n^2)$和$O(nlogn)$之间。
- 目前还未被解决的问题：是否存在一个增量序列使希尔排序的平均时间复杂度达到 $O(nlogn)$.



think.create.solve



平方阶排序算法

- 排序概念
- 直接插入排序
- 希尔排序
- **冒泡排序**
- 直接选择排序

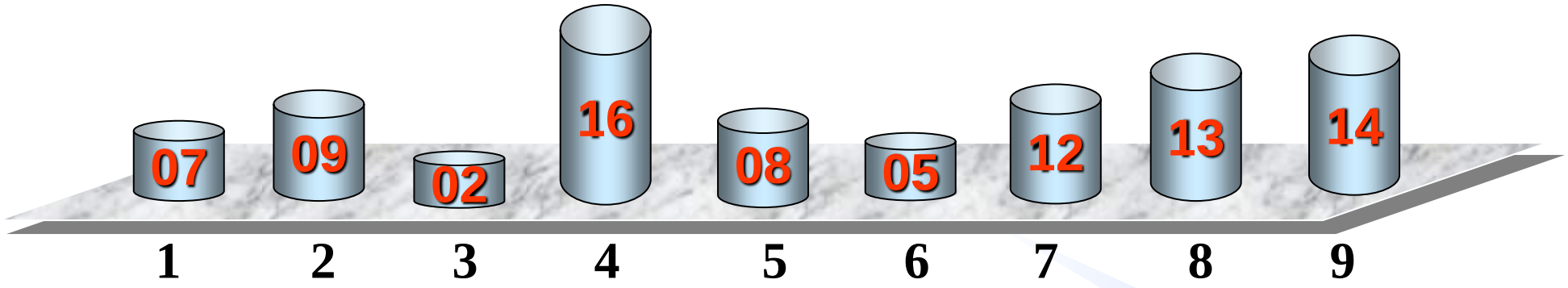
数据之法 结构之美 算法之道

zhuyungang@jlu.edu.cn

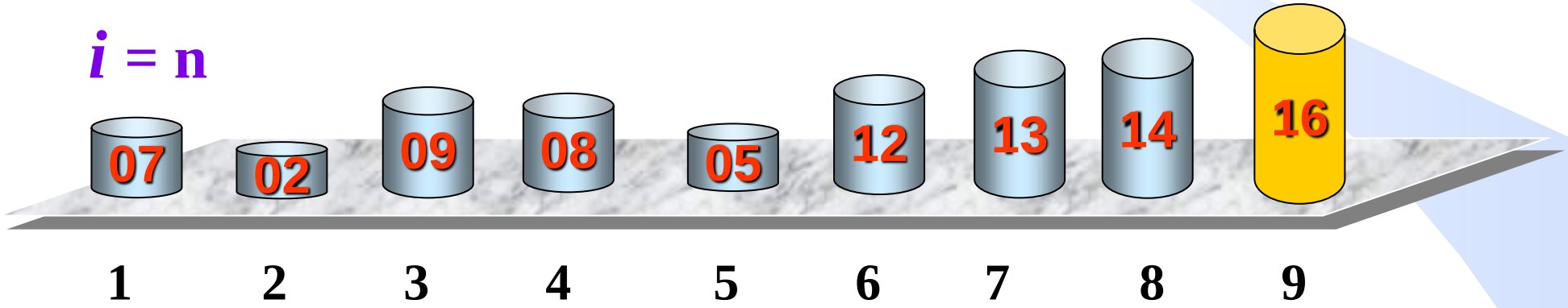
Last updated on 2022.11

冒泡排序

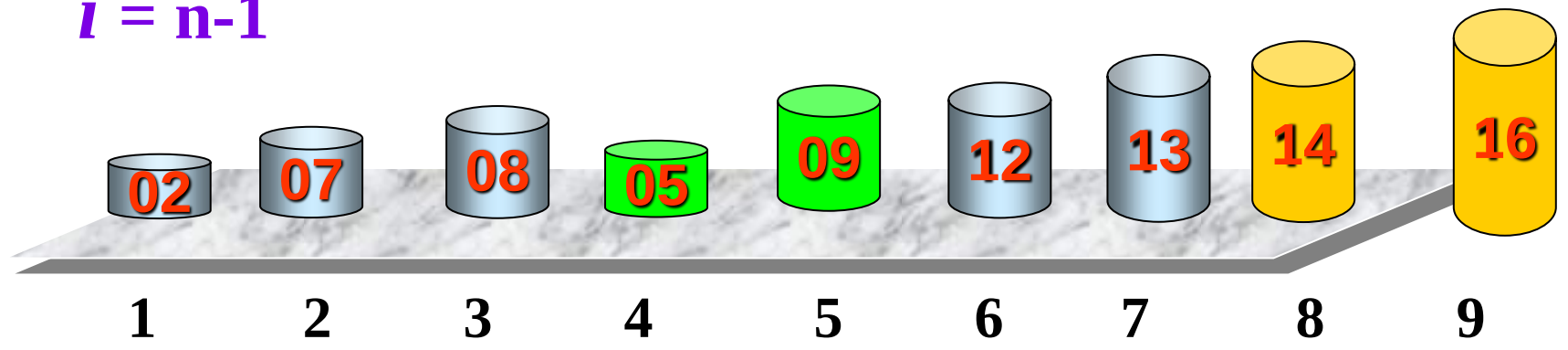
思想：自下而上（或从左到右）比较相邻记录的关键词，交换存在逆序的记录（若$K_j > K_{j+1}$，则互换R_j和R_{j+1}）；使关键词较大的记录如气泡一般逐渐往上“飘移”直至“水面”。


07
09
02
16
08
05
12
13
14
1
2
3
4
5
6
7
8
9

i = n


07
02
09
08
05
12
13
14
16
1
2
3
4
5
6
7
8
9

i = n-1


02
07
08
05
09
12
13
14
16
1
2
3
4
5
6
7
8
9

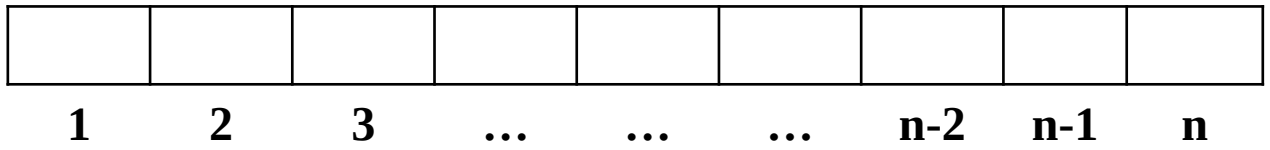
冒泡排序算法

算法BSort (R, n)

FOR $i = n$ **TO** 2 **STEP** –1 **DO**

  FOR j =1 **TO** i-1 **DO**

    IF $K_j > K_{j+1}$ **THEN** $R_j \leftrightarrow R_{j+1}$. ■



- 关键词的比较次数:

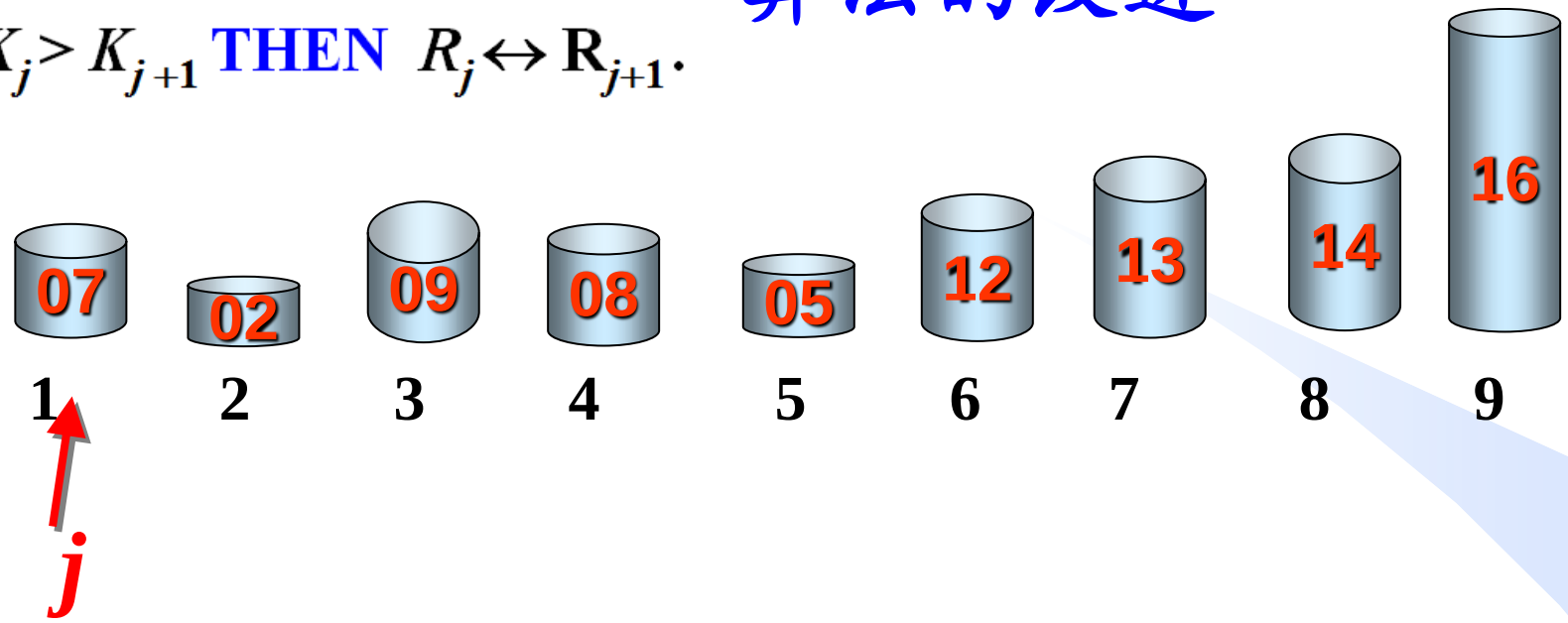
  $(n-1)+(n-2)+\ldots+1= (n-1)n/2$

- 经过第一趟冒泡，可把具有最大关键词的记录移至最终位置（第 n 个位置）.
- 通过$n-1$ 趟冒泡,就可完成对所有记录的排序.
- 发生一次记录交换，反序对就减少一个.
- 排序中可能出现如下情况：

  每趟比较中，在某位置后，没有记录交换，就意味着该位置后已经排好序了。

算法的改进

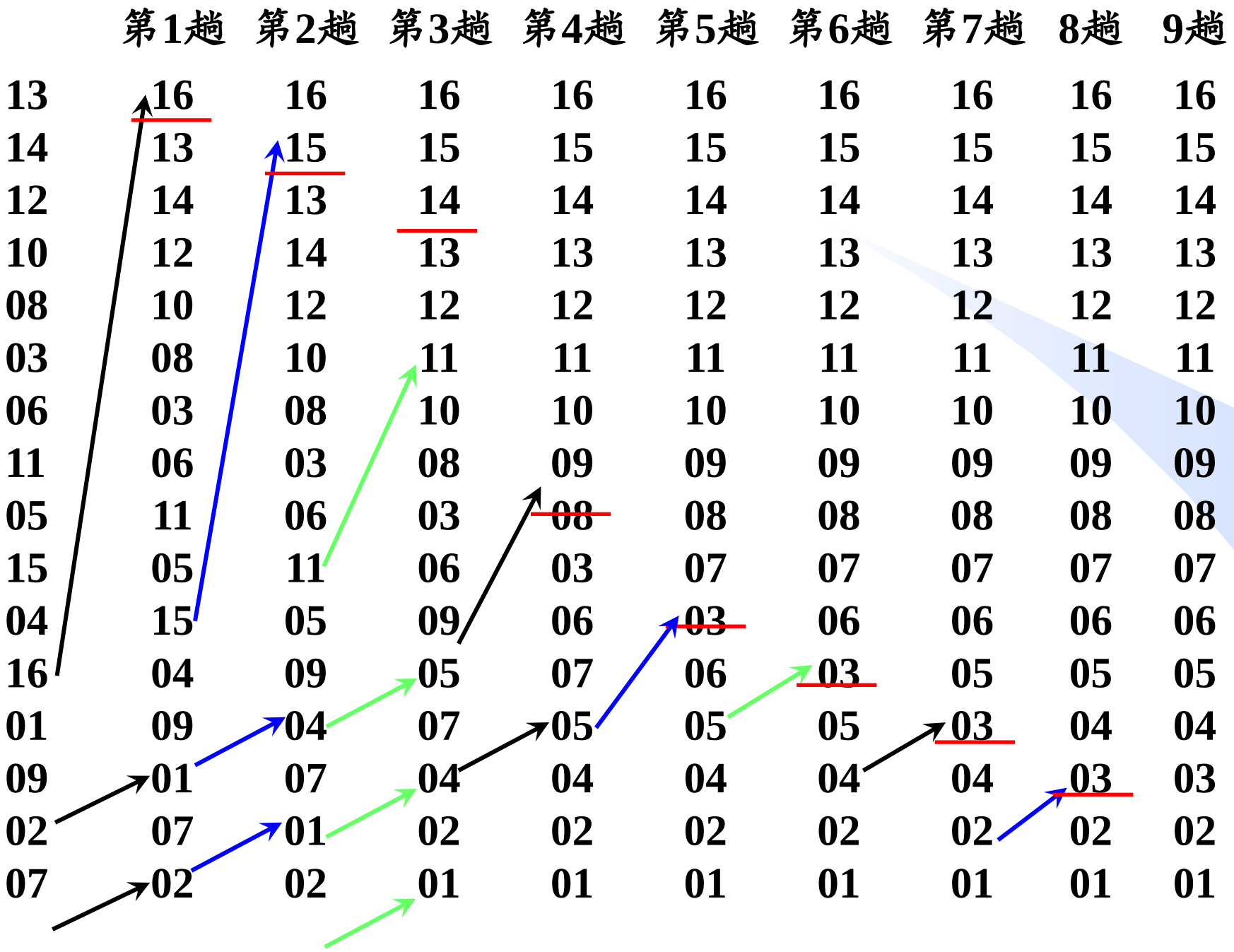
IF $K_j > K_{j+1}$ THEN $R_j \leftrightarrow R_{j+1}$.



- 在每一趟比较中，当比较结束后，如果发现位置t是最后一次记录交换，即说明从$R_{t+1} \sim R_n$已经排序。
- 从而下一趟比较只要进行到位置t即可。这样可以减少算法的关键词比较次数。
- 据此可以给出一种改进的冒泡排序算法。

改进的冒泡排序算法

```
算法Bubble ( R, n )
    BOUND ← n.  //每趟冒泡关键词比较的终止位置
    WHILE  BOUND ≠ 0  DO // BOUND=0, 终止算法
    (       t ← 0.   //本趟冒泡元素交换的最后位置
            FOR  j =1 TO BOUND-1 DO
                IF K_j > K_{j+1}  THEN (R_j ↔ R_{j+1}.  t ← j. )
            BOUND ← t.
    ) .
```


第1趟 第2趟 第3趟 第4趟 第5趟 第6趟 第7趟 8趟 9趟
13 16 16 16 16 16 16 16 16 16
14 13 15 15 15 15 15 15 15 15
12 14 13 14 14 14 14 14 14 14
10 12 14 13 13 13 13 13 13 13
08 10 12 12 12 12 12 12 12 12
03 08 10 11 11 11 11 11 11 11
06 03 08 10 10 10 10 10 10 10
11 06 03 08 09 09 09 09 09 09
05 11 06 03 08 08 08 08 08 08
15 05 11 06 03 07 07 07 07 07
04 15 05 09 06 03 06 06 06 06
16 04 09 05 07 06 03 05 05 05
01 09 04 07 05 05 05 03 04 04
09 01 07 04 04 04 04 04 03 03
02 07 01 02 02 02 02 02 02 02
07 02 02 01 01 01 01 01 01 01

算法分析

- **最好情形**

  若在待排序文件中，诸记录对应之关键词满足$K_1 \leq K_2 \leq \ldots \leq K_n$，则算法只需进行一趟起泡，做 $n-1$ 次关键词比较，无记录交换；

- **最坏情形**

  算法执行$n-1$趟起泡，第 i 趟 $(1 \leq i < n)$ 做了 $n-i$ 次关键词比较，执行了$n-i$ 次记录交换，此时，总的关键词比较次数和记录交换次数均为：

  $(n-1)n/2$

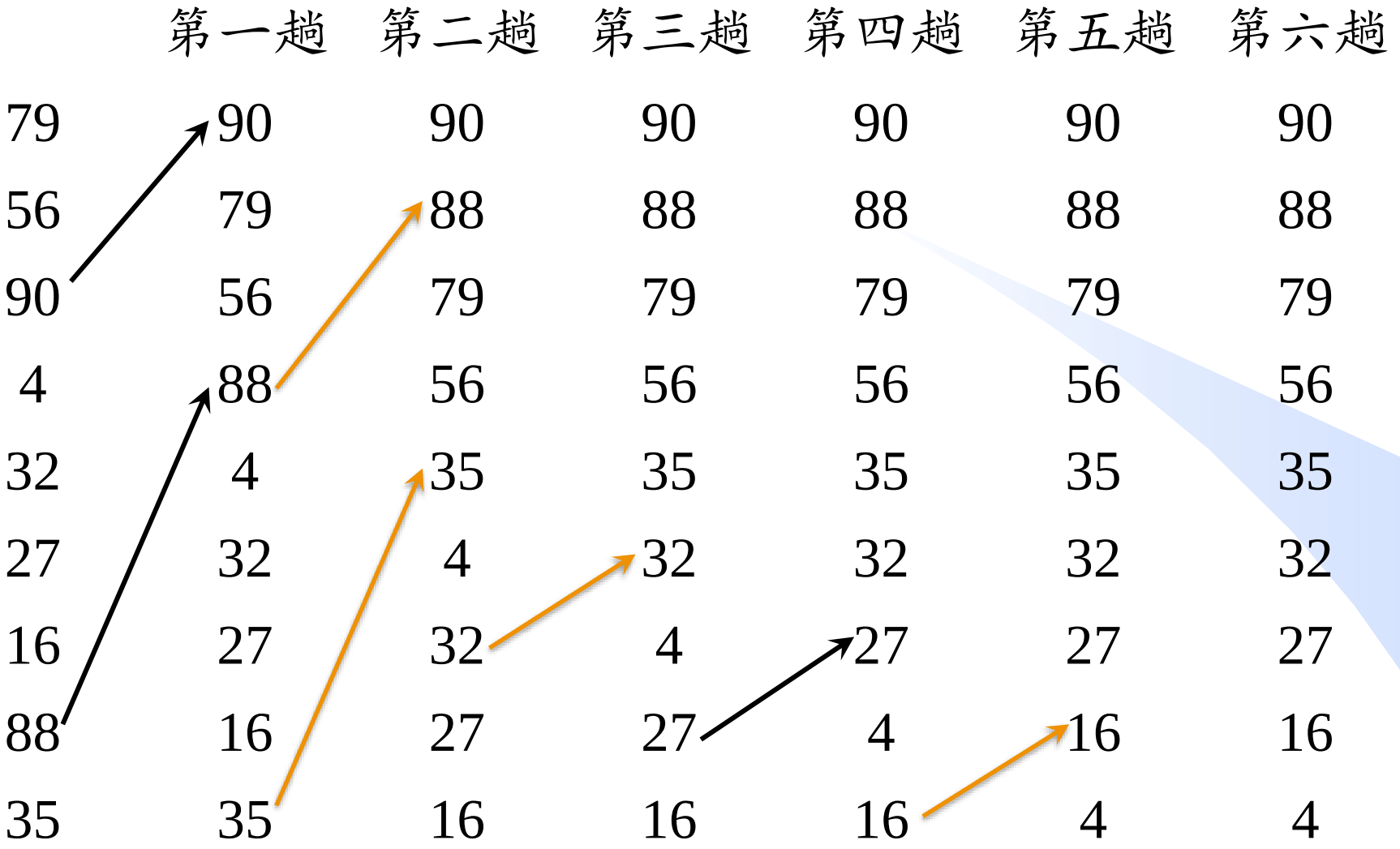
改进的冒泡排序算法的时间复杂性

	时间复杂性
最好情况	$O(n)$
平均情况	$O(n^2)$
最坏情况	$O(n^2)$

冒泡排序算法

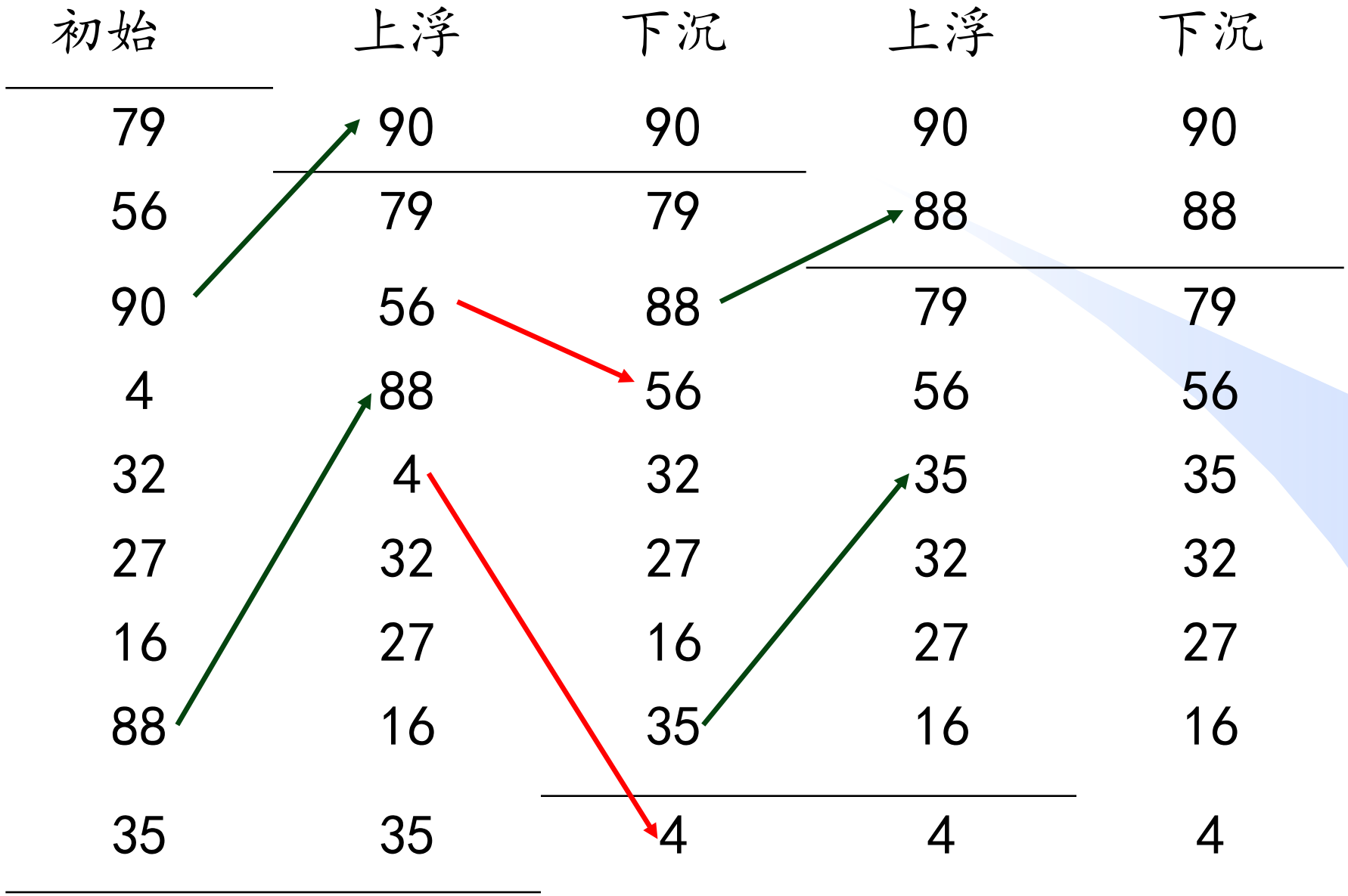
$$\text{IF } K_j > K_{j+1} \text{ THEN } R_j \leftrightarrow R_{j+1}.$$

- 时间复杂度：$O(n^2)$（包括最坏和平均）。
- 稳定性：冒泡排序是稳定的排序方法。
- 最好情况是：当被排序文件初态为正序时，算法的时间复杂度为$O(n)$.
- 空间复杂度：$O(1)$.
- 可以采用气泡上浮和下沉交替的方法排序。
- 下沉：把小的元素向左移或向下移，一趟冒泡，至少可以把关键词最小的记录送到最前左的位置（或最下边）。



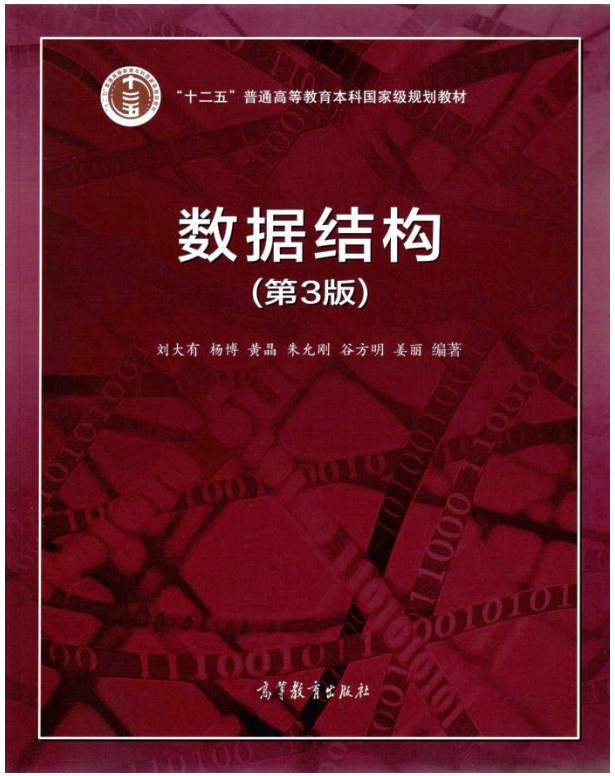
单纯上浮排序算法的运行过程

该例需要扫描 6 趟


初始
上浮
下沉
上浮
下沉
79
56
90
4
32
27
16
88
35
90
79
88
56
4
32
27
16
35
90
79
88
56
35
32
27
16
4
90
88
56
35
32
27
16
4

6：4

think.create.solve



平方阶排序算法

- 排序概念
- 直接插入排序
- 希尔排序
- 冒泡排序
- **直接选择排序**

数据之法 结构之美 算法之道

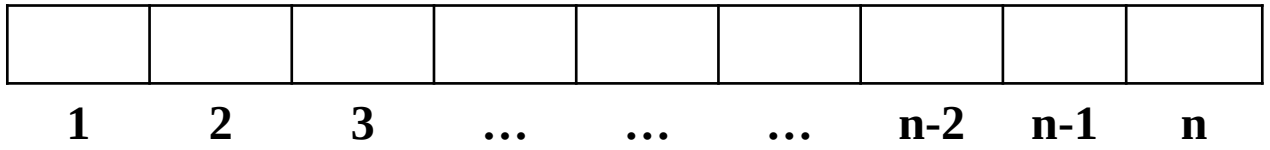
zhuyungang@jlu.edu.cn

Last updated on 2022.11

直接选择排序

直接选择排序思想：

- 在未排序的序列中选最小的元素，并和首元素交换。
- 在剩下的未排序的序列中再选出最小的元素与序列中第二个元素交换。
- 以此类推，最后形成递增的已排序序列。



算法 SelectionSort (*R*, *n*) // 直接选择排序算法

FOR *i*=1 **TO** *n*-1 **DO** //*i*表示要处理的子数组的起点

/*在R_i … R_n的子数组里找最小元素，和R_i交换表示要处理的子数组的起点*/

(　*min* ← *i* . //扫描以*i*为起点的子数组，找最小元素

　FOR *j* = *i*+1 **TO** *n* **DO**

　　IF $K_j < K_{min}$ **THEN** *min* ← *j*.

　$R_i \leftrightarrow R_{min}$.　　//将第*i*个元素与最小元素交换

)

算法分析

直接选择排序的关键词比较次数与记录的初始排列无关。假定整个待排序文件有 n 个记录，第 i 趟选择具有最小关键词的记录所需的比较总次数是 $n-i$ 次。因此，总的关键词比较次数为

$$(n-1)+(n-2)+\ldots+1=\frac{1}{2}n(n-1)$$

直接选择排序算法——总结

- 时间复杂度：$O(n^2)$（最好、最坏和平均）。
- 稳定性：不稳定的排序方法。
- 空间复杂度： $O(1)$ 。
- 优点：简单、书写容易。