福州大学至诚学院
FUZHOU UNIVERSITY ZHICHENG COLLEGE

高级语言程序设计（C语言与数据结构）

杨雄

83789047@qq.com

10.内部排序

01 概述

02 插入排序

03 快速排序

学习目的

- 熟练掌握直接插入排序的过程和算法实现
- 熟练掌握冒泡排序的过程和算法实现
- 理解两种排序的算法时间复杂度

Part.1

10.1 概述

什么是排序？

排序是计算机内经常进行的一种操作，其目的是将一组**“无序”的记录序列调整为“有序”**的记录序列。

例如：将下列关键字序列

52, 49, 80, 36, 14, 58, 61, 23, 97, 75

调整为

14, 23, 36, 49, 52, 58, 61 ,75, 80, 97

10.1 概述

- **排序定义**：假设含n个记录的序列为

$$\{R_1, R_2, \ldots, R_n\}$$

其相应的关键字序列为

$$\{K_1, K_2, \ldots, K_n\}$$

需确定1，2，…，n的一种排列p_1，p_2，…，p_n，使其相应的关键字满足如下的**非递减(或非递增)**关系

$$Kp_1 \leq Kp_2 \leq \ldots \leq Kp_n$$

即使记录序列成为一个按关键字有序的序列

$$\{Rp_1, Rp_1, \ldots, Rp_n\}$$

这样一种操作称为**排序**。

10.1 概述

- **排序的稳定性：**

假设$K_i=K_j$（$1\le i\le n, 1\le j\le n$，i!=j），且在排序前的序列中**R_i领先于R_j**（即i<j）。

若在排序后的序列中**R_i仍领先于R_j**，则称所用的排序方法是**稳定**的；

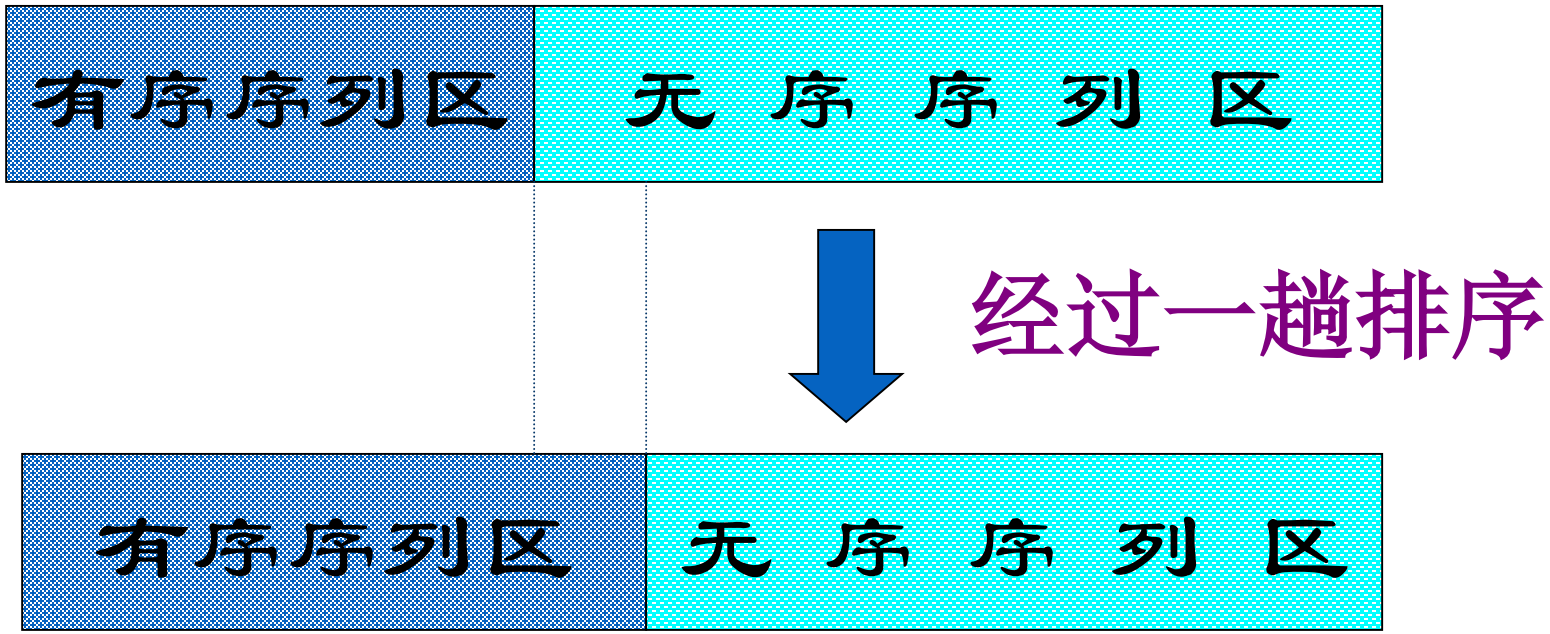
反之，若可能使排序后的序列中R_j领先于R_i，则称所用的排序方法是**不稳定**的。

10.1 概述

- **内部排序：**待排序记录存放在计算机的内存中进行排序。
- **外部排序：**待排序记录的数量很大，以致内存一次不能容纳全部记录，在排序过程中尚需对外存进行访问的排序。

内部排序方法

内部排序的过程是一个**逐步扩大**记录的**有序序列长度**的过程。



内部排序

- 排序基本操作：
  - 比较两个关键字大小
  - 将记录从一个位置移动到另一个位置

内部排序

- 待排序记录序列可有下列**三种存储方式**:
  - **记录存放在一组连续的存储单元中**;

    类似于线性表的**顺序存储结构**，记录次序由存储位置决定，实现排序需移动记录。
  - **记录存放在静态链表中;**

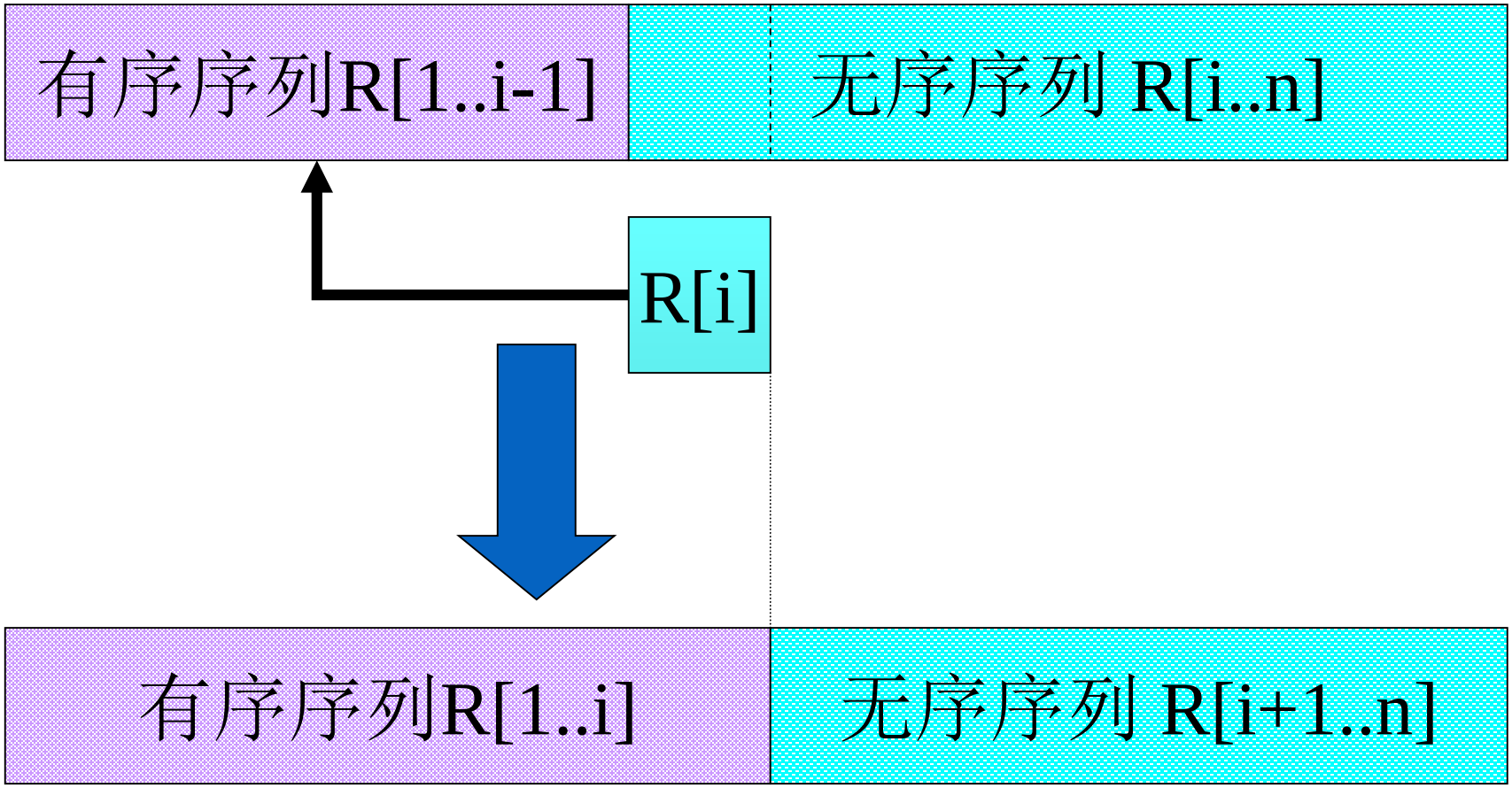
    记录次序由**指针**指示，实现排序不需移动记录，仅需修改指针(链排序)
  - **记录本身存放在一组连续的存储单元中，同时另设指示各个记录存储位置的地址向量，**

    排序过程中不移动记录本身，而移动地址向量中相应记录的地址.排序结束后再根据地址调整记录的存储位置 (地址排序)。

Part.2

10.2 插入排序

- **插入排序基本思想：**



- **排序过程（实现“一趟插入排序”可分三步进行）：**

1. 在R[1..i-1]中**查找**R[i]的插入位置，

R[1..j].key ≤ R[i].key < R[j+1..i-1].key；

2. 将R[j+1..i-1]中的所有**记录均后移**一个位置；

3. 将R[i] **插入**(复制)到R[j+1]的位置上。

● 直接插入排序

- **直接插入排序基本思想：**

每一趟将一个待排序的记录，**按其关键字值的大小插入**到**已经排好序的部分序列**的适当位置上，直到全部插入为止。

利用“顺序查找”实现

“在R[1..i-1]中**查找**R[i]的插入位置”

● 直接插入排序

例：序列为{49，38，65，97，76，13，27，49}

[初始关键字]：{(49)，38，65，97，76，13，27，49}

i=2 (38) {(38，49)，65，97，76，13，27，49}

i=3 (65) {(38，49，65)，97，76，13，27，49}

i=4 (97) {(38，49，65，97)，76，13，27，49}

i=5 (76) {(38，49，65，76，97)，13，27，49}

i=6 (13) {(13，38，49，65，76，97)，27，49}

i=7 (27) {(13，27，38，49，65，76，97)，49}

i=8 (49) {(13，27，38，49，49，65，76，97)}

● 直接插入排序

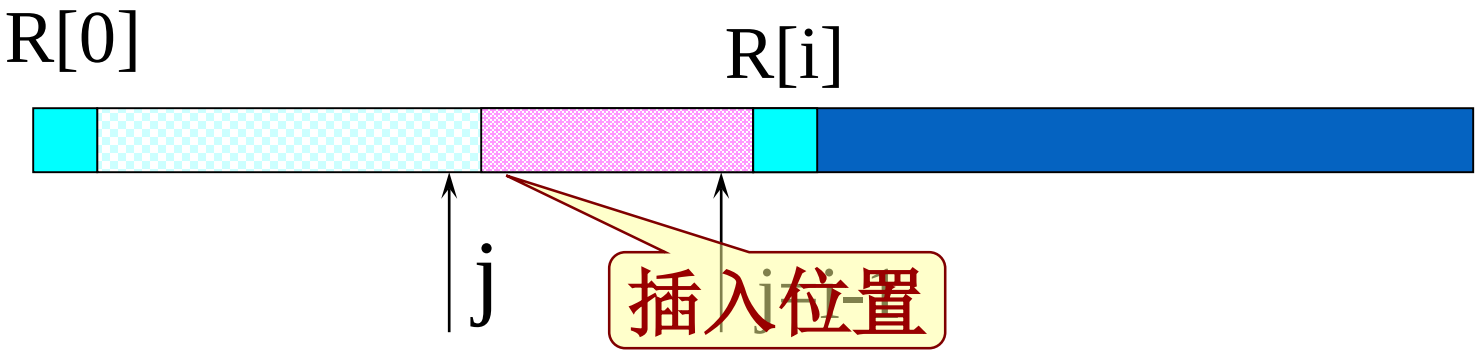
- **排序过程：**

第*i*趟直接插入排序的操作为：在含有*i-1*个记录的有序序列*r[1..i-1]*中插入一个记录*r[i]*后，变成含有*i*个记录的有序序列*r[1..i]*。

整个排序过程为**n-1**趟插入，即先将序列中第1个记录看成是一个有序子序列，然后从第2个记录开始，逐个进行插入，直至整个序列有序。

直接插入排序

- **算法实现要点：** 从R[i-1]起向前进行顺序查找，监视哨设置在R[0]；




```
R[0] = R[i];              // 设置“哨兵”
for (j=i-1; R[0].key<R[j].key; --j);
                          // 从后往前找
```

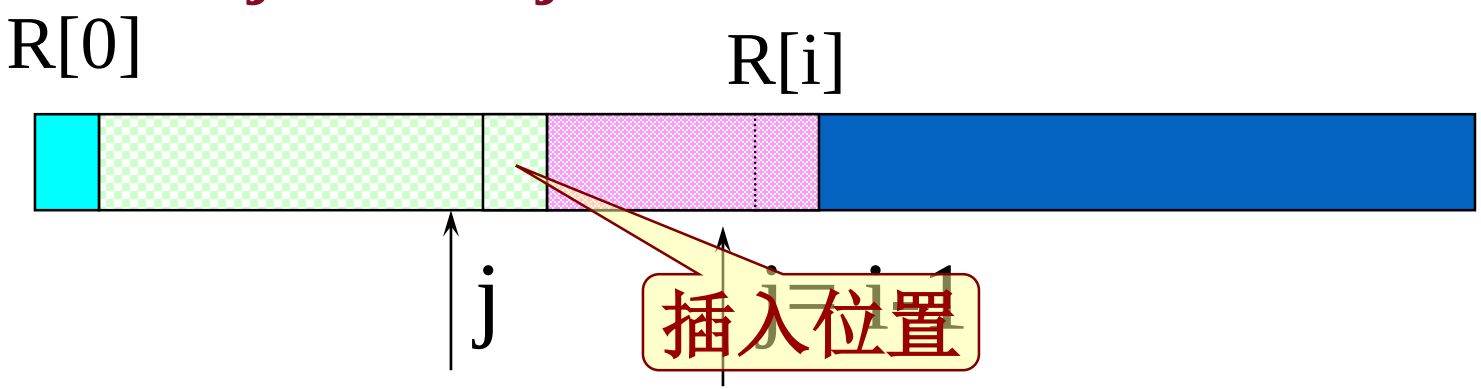
循环结束表明R[i]的插入位置为 ***j +1***

直接插入排序

- **算法实现要点：**
  - 对于在查找过程中找到的那些关键字不小于R[i].key的记录，并在查找的同时实现记录向后移动；

```
for (j=i-1; R[0].key<R[j].key; --j);
```

R[j+1] = R[j]



  - 上述循环结束后可以直接进行“插入”

直接插入排序

- **算法实现要点：**
  - 令 i = 2，3，…, n,

    实现整个序列的排序。

```
for ( i=2;  i<=n;  ++i )
  if (R[i].key<R[i-1].key)
   {  在 R[1..i-1]中查找R[i]的插入位置;
      插入R[i] ;
   }
```

- 直接插入排序

```
void  InsertSort(SqList  &L)
{
    for(i=2; i<=L.length; ++i)
      if(L.r[i].key < L.r[i-1].key ){
        L.r[0] = L.r[i]; // 设为哨兵
        for(j=i-1; L.r[0].key < L.r[j].key; --j)
           L.r[j+1] = L.r[j]; // 记录后移
        L.r[j+1] = L.r[0];
     }
}
```

- 直接插入排序
  - 空间效率：**一个辅助空间**
  - 时间效率

实现内部排序的**基本操作**有两个：

（1）**“比较”**序列中两个关键字的大小；

（2）**“移动”**记录。

- 直接插入排序

•时间效率

最好的情况（关键字在记录序列中顺序有序）：

“比较”的次数：

$$\sum_{i=2}^{n} 1 = n-1$$

“移动”的次数：

0

最坏的情况（关键字在记录序列中逆序有序）：

“比较”的次数：

$$\sum_{i=2}^{n} (i) = \frac{(n+2)(n-1)}{2}$$

“移动”的次数：

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

- 直接插入排序
  - 时间效率
    - 若待排序记录是随机的，取平均值
      - 关键字比较次数：$n^2/4$
      - 记录移动次数：$n^2/4$
    - **时间复杂度：$O(n^2)$**

Part.3

10.3 快速排序

- 冒泡排序(Bubble Sort):

将被排序的序列R**垂直排列**，每个记录R[i]看作是重量为R[i].key的气泡。

根据轻气泡不能在重气泡之下的原则，从下往上扫描序列R:**凡扫描到违反本原则的轻气泡，就使其向上"飘浮"**。

如此反复进行，直到最后任何两个气泡都是轻者在上，重者在下为止。

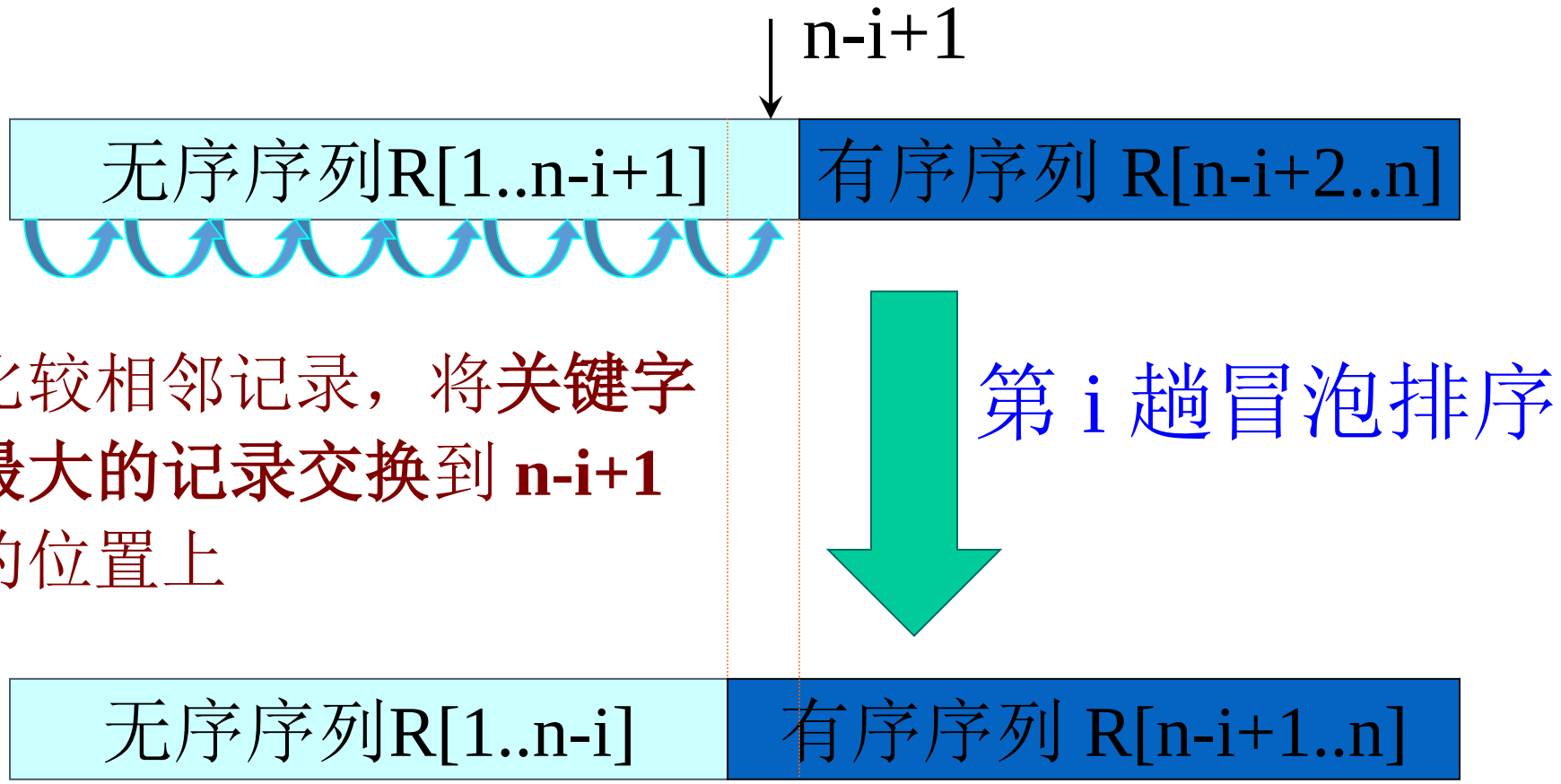
■ 冒泡排序

- 排序过程:
  - 首先将第一个记录的关键字和第二个记录的关键字进行比较,若为逆序,则将两个记录交换之,然后比较第二个记录和第三个记录的关键字。依次类推,直至第n-1个记录和第n个记录的关键字进行过比较为止。上述过程称作第一趟冒泡排序,其结果使得**关键字最大的记录被安置到最后一个记录的位置上**。
  - 然后进行第二趟冒泡排序,对前n-1个记录进行同样操作,其结果是**使关键字次大的记录被安置到第n-1个记录的位置上**

- **冒泡排序的结束条件:在某一趟排序过程中没有进行记录交换的操作”。**

- 冒泡排序(Bubble Sort):

  假设在排序过程中，记录序列R[1..n]的状态为：



例：

初始关键字	第一趟排序后	第二趟排序后	第三趟排序后	第四趟排序后	第五趟排序后	第六趟排序后
49	38	38	38	38	13	13
38	49	49	49	13	27	27
65	65	65	13	27	38	38
97	76	13	27	49	49	
76	13	27	49*	49*		
13	27	49*	65			
27	49*	76				
49*	97					

冒泡排序的一个实例

- 冒泡排序

冒泡排序.txt

冒泡排序－时间复杂度分析

最好的情况（关键字在记录序列中顺序有序）：

只需进行一趟冒泡

“比较”的次数：

n-1

“移动”的次数：

0

最坏的情况（关键字在记录序列中逆序有序）：

需进行n-1趟冒泡

“比较”的次数：

$$\sum_{i=n}^{2}(i-1)=\frac{n(n-1)}{2}$$

“移动”的次数：

$$3\sum_{i=n}^{2}(i-1)=\frac{3n(n-1)}{2}$$

T(n)=O(n²)

Part.3

总结

总 结

排序方法	平均时间	最坏情况	最好情况	辅助空间	稳定性
直接插入排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	√
冒泡排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	√