交换排序

交换排序

通过“交换”无序序列中的记录从而得到其中关键字最小或最大的记录，并将它加入到有序子序列中，以此方法增加记录的有序子序列的长度。

- 冒泡排序
- 快速排序

冒泡排序

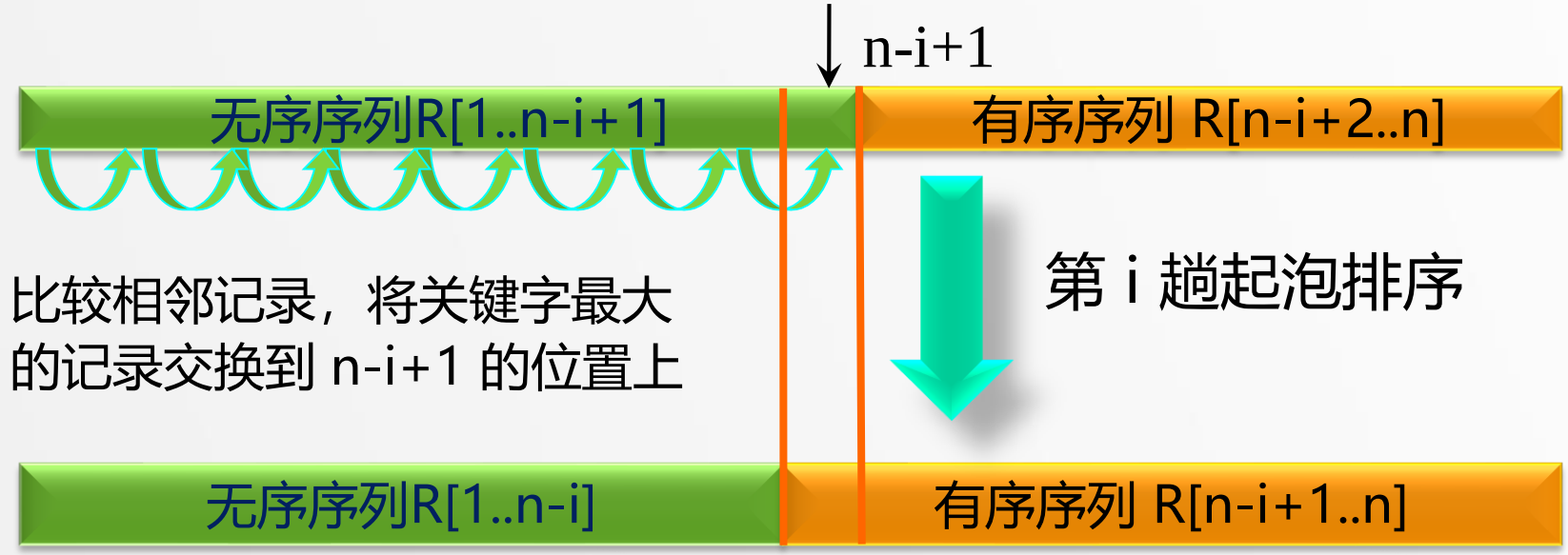
交换排序

冒泡排序

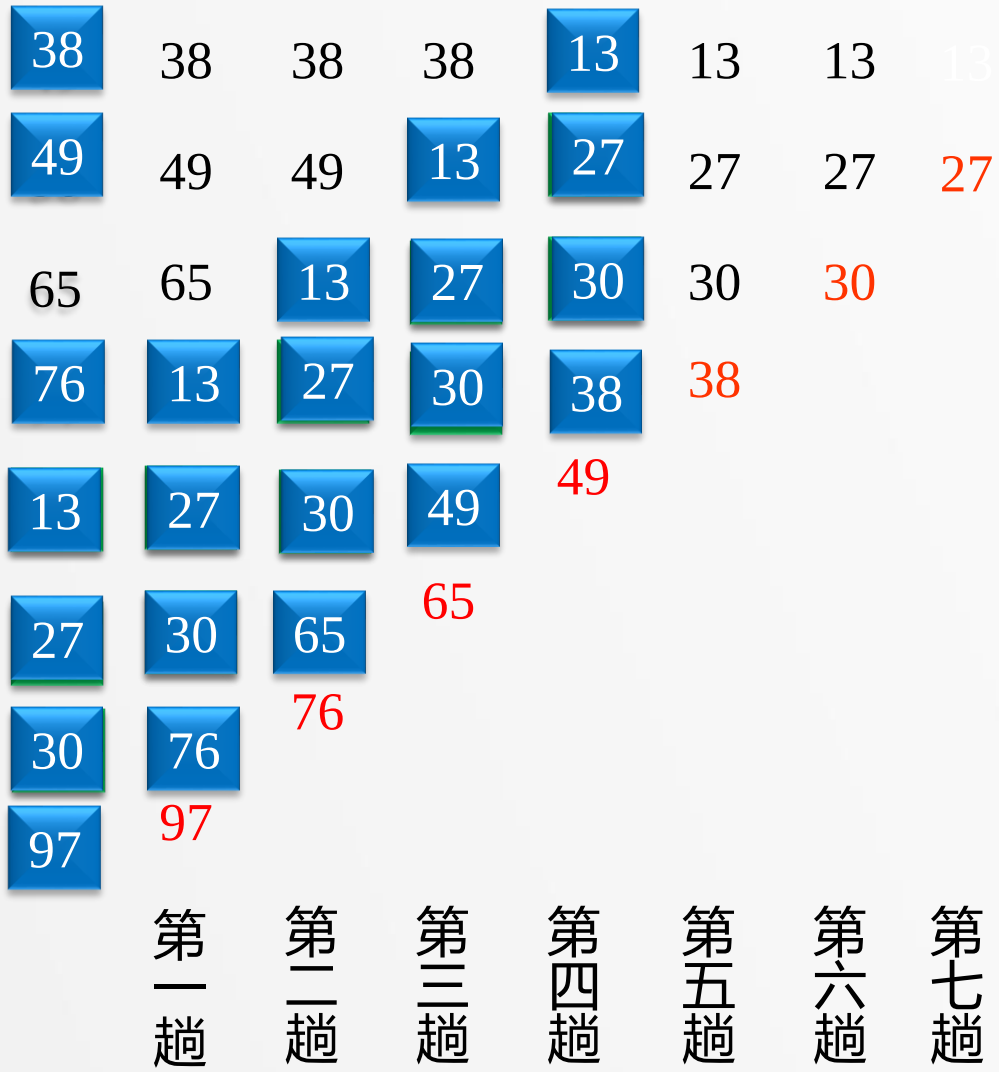
排序过程

1. 将第一个记录的关键字与第二个记录的关键字进行比较，若为逆序r[1].key>r[2].key，则交换；然后比较第二个记录与第三个记录；依次类推，直至第n-1个记录和第n个记录比较为止——第一趟冒泡排序，结果关键字最大的记录被安置在最后一个记录上
2. 对前n-1个记录进行第二趟冒泡排序，结果使关键字次大的记录被安置在第n-1个记录位置
3. 重复上述过程，直到“在一趟排序过程中没有进行过交换记录的操作”为止

假设在排序过程中，记录序列R[1..n]的状态为：



例


38 49 65 76 13 27 30 97
38 49 65 13 27 30 76 97
38 49 13 27 30 65 76
38 13 27 30 49 65
13 27 30 38 49
13 27 30 38
13 27 30
13 27
13
第一趟
第二趟
第三趟
第四趟
第五趟
第六趟
第七趟

```
void bubble_sort(JD r[], int n)
{
    int m, i, j, flag = 1;
    JD x;
    m = n ;
    while ((m>1) && (flag == 1))/*趟数*/
    {
        flag = 0;/*本趟是否有交换操作标识初始化*/
        for (j = 1;j < m;j++)//*本趟将最大元素放到为排序序列的最后*/
        if (r[j].key>r[j + 1].key)
        {
            flag = 1;
            x = r[j];
            r[j] = r[j + 1];
            r[j + 1] = x;
        }
        m--;
    }
}
```

算法评价

时间复杂度

- 最好情况（正序）
  - 比较次数：n-1
  - 移动次数：0
- 最坏情况（逆序）
  - 比较次数：$\sum_{i=1}^{n-1}(n-i)=\frac{1}{2}(n^2-n)$
  - 移动次数：$3\sum_{i=1}^{n}(n-i)=\frac{3}{2}(n^2-n)$

$T(n)=O(n^2)$

空间复杂度：$S(n)=O(1)$

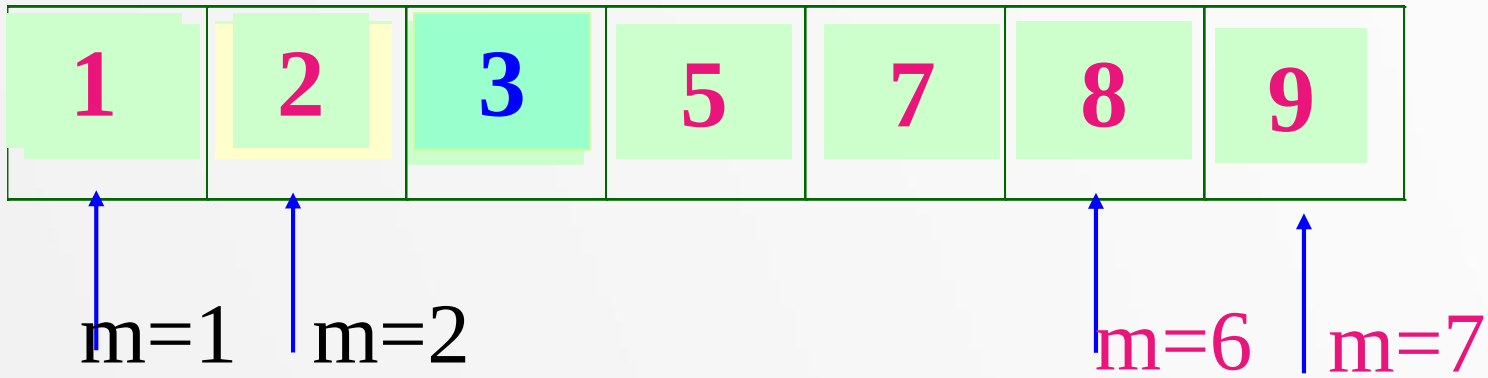
Ch8_4.c

冒泡排序的改进:

1. 冒泡排序的结束条件为，最后一趟没有进行“交换记录”。

2. 一般情况下，每经过一趟“冒泡”，“m减1”，但并不是每趟都如此。

例如:




```
for (j = 1; j < m; j++) if (R[j+1].key < R[j].key) ...
```

```
void BubbleSort(Elem R[ ], int n) {
   m = n;
  while (m >1) {
      lastExchangeIndex = 1;
       for (j = 1;  j < m;  j++)
      if (R[j].key > R[j+1].key)
       { Swap(R[j], R[j+1]);
         lastExchangeIndex = j;  /*记下进行交换的记录位置*/
        }
      m = lastExchangeIndex; /*本趟最后一次交换的位置*/
  }
}
```