4 简单排序(1)

Simply Sorting

郝家胜

haojiasheng@gmail.com

自动化工程学院

内容提要

- 简单插入排序
- 简单选择排序
- 冒泡排序

排序 (Sorting)

- 问题描述

  输入： 整数序列 $\langle a_1, a_2, \ldots, a_n \rangle$

  输出： 整数序列 $\langle a'_1, a'_2, \ldots, a'_n \rangle$，其中 $a'_1 \le a'_2 \le \cdots \le a'_n$.

  例：

  输入： 8 2 4 9 3 6

  输出： 2 3 4 6 8 9

基本概念

- 记录 (Record)
  - 节点，排序的对象 – 成绩单
- 关键字(Key)
  - 唯一确定记录的项 – 学号
- 排序码
  - 记录中的项，排序的依据 – 成绩
- 排序
  - 将一组记录依据排序码顺序排列起来，使得其排序码具有不减（或不增）的顺序
- 正序，逆序

1 简 单 插 入 排 序

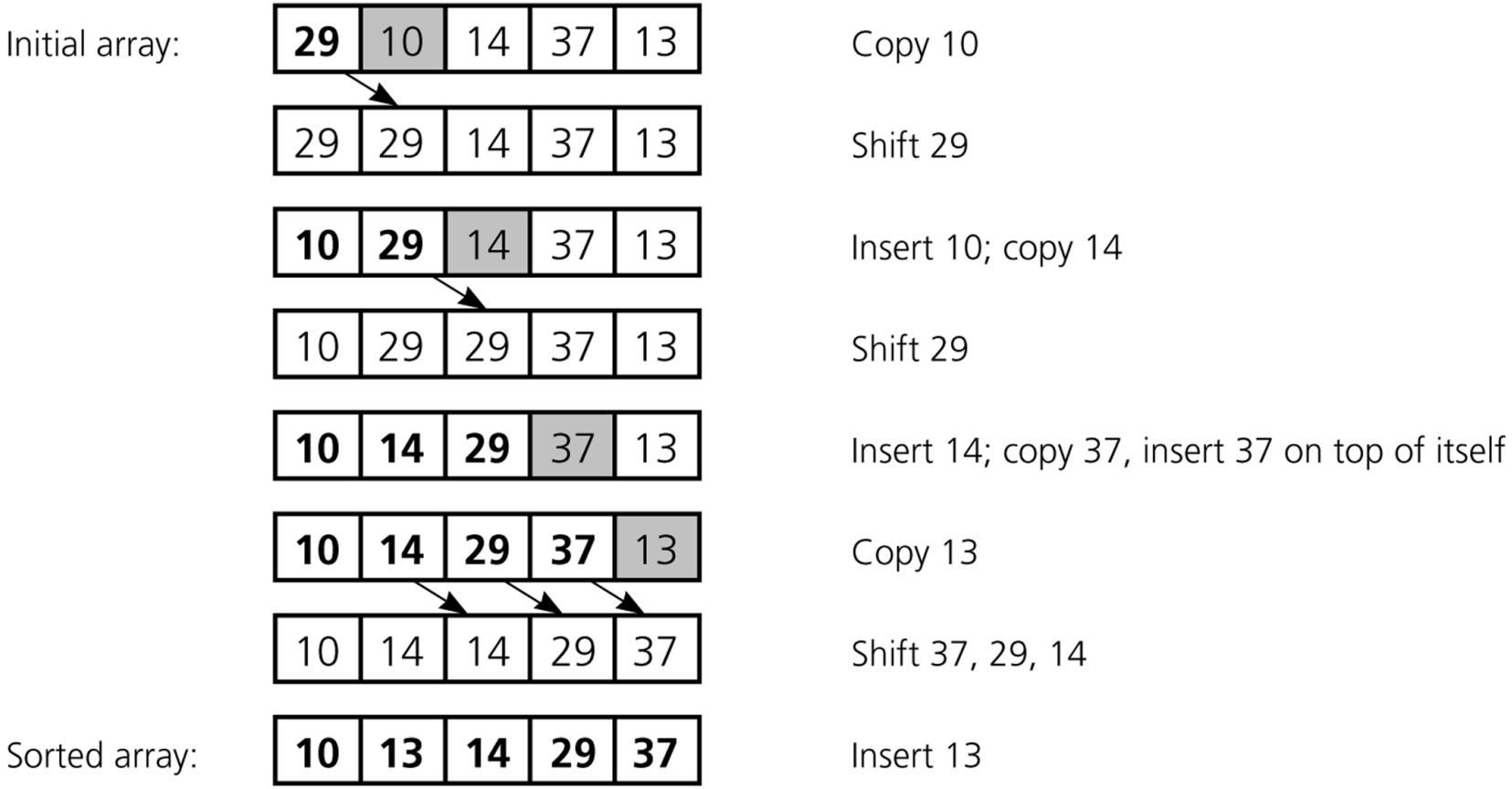
简单插入排序 (Insertion Sort)

- 每步将一个待排序的记录，按关键码值的大小插入到前面已排序的适当位置上，直到全部插完止

例11

- 待排序记录的排序码是(18, 12, 10, 12, 30, 16)，写出简单插入排序每一趟的执行结果

18	12	10	12	30	16
12	18	10	12	30	16
10	12	18	12	30	16
10	12	12	18	30	16
10	12	12	18	30	16
10	12	12	16	18	30


Initial array:
29 10 14 37 13
Copy 10
29 29 14 37 13
Shift 29
10 29 14 37 13
Insert 10; copy 14
10 29 29 37 13
Shift 29
10 14 29 37 13
Insert 14; copy 37, insert 37 on top of itself
10 14 29 37 13
Copy 13
10 14 14 29 37
Shift 37, 29, 14
Sorted array:
10 13 14 29 37
Insert 13

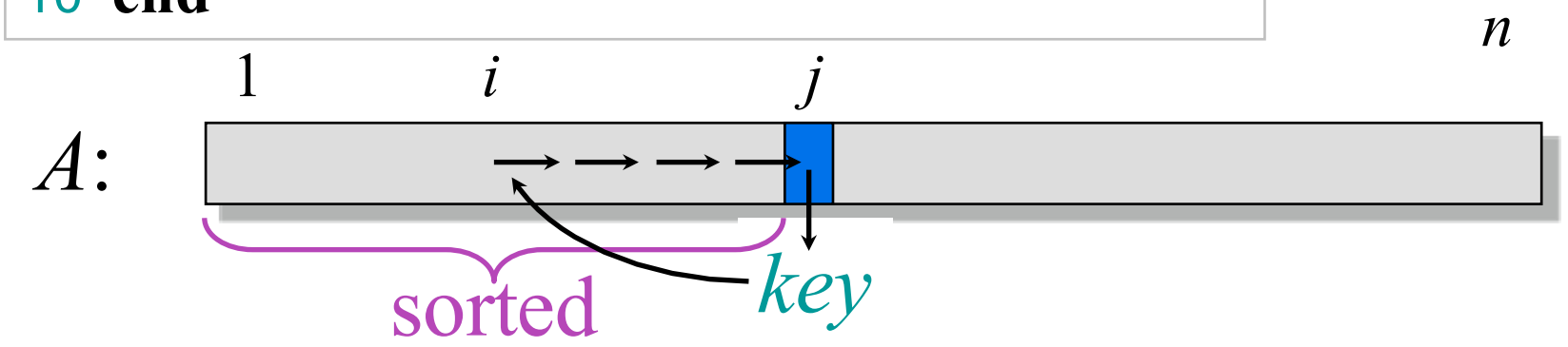
算法描述

- 设有n个记录(R_1，R_2，…，R_n)，已划分为已排序部分和未排序部分，即插入R_i时，(R_1，R_2，…，R_{i-1})是已排好序的部分，(R_i，R_{i+1}，…，R_n)属于未排序部分
- 用R_i依次与R_{i-1}，R_{i-2}，…，R_1进行比较，找出R_i在有序子文件中的插入位置，将R_i插入，原位置上的记录至R_{i-1}均顺序后移一位
- 从R_{i+1}重复上述过程，直至排序完成

伪代码描述

```
INSERTION-SORT (A, n)
1  ▷ Insert A[j] into the sorted sequence
2  for j ← 2 to n do
3        key ← A[ j]
4        i ← j – 1
5        while   i > 0 and A[i] > key   do
6              A[i+1] ← A[i]
7              i ← i – 1
8        end
9        A[i+1] = key
10 end
```

注意：伪代码惯例是编号从1开始。但采用C语言实现时数组下标从0而不是1开始



算法分析

- 算法特点
  - 原位排序
  - 稳定的(概念)
- 时间复杂度
  - 向有序表中逐个插入记录的操作，进行了n-1趟，每一趟比较和移动记录的次数取决于待排序列按关键码的初始排列状态
  - 最好情况下，即初始序列正序时，比较n-1次，移动0次
  - 最坏情况下，即初始序列逆序时，比较n(n-1)/2次，移动n(n-1)/2+n次
  - 因此，时间复杂度为O(n^2)，适合于记录个数比较少的情况，若原始数据序列在基本有序的情况下，算法效率比较高
- 空间复杂度
  - O(1)

2 选择排序

选择排序 (Selection Sort)

- 简单选择排序的方法是在所有的记录中选出关键字最小的记录，把它与第一个记录交换存储位置，然后再在余下的记录中选出次小的关键字对应的记录，把它与第二个记录交换，依此类推，直至排序完成。
- 以降低数据元素的移动次数为排序思路
- 每次从待排序的记录中选出关键字最小(或最大)的记录，顺序放在已排序的记录序列的最后，直到全部排完为止

例12

- 待排序记录的排序码是(2, 7, 2, 2, 3, 1)，写出简单选择排序每一趟的执行结果

- 不稳定

算法描述

- 第一次，选取整个序列中关键字最小的记录与第一个元素交换；
- 第二次，从剩余的n-1个记录中选出关键字最小的记录与第二个记录交换；
- 第i次，则从剩余的n-i+1个记录选出关键字最小的记录与第i个记录交换；
- 直到整个序列按关键码有序

算法伪代码

```
SELECTION-SORT (A, n)
1  ▷ Exchange A[j] with the smallest
2  for j ← 1 to n  do
3        min ← j
4        i ← j + 1
5        while i <= n  do
6              if A[i] < A[min]
7                    min ← i
8              i ← i + 1
9        end
10       A[j] ↔ A[min]
11 end
```

算法分析

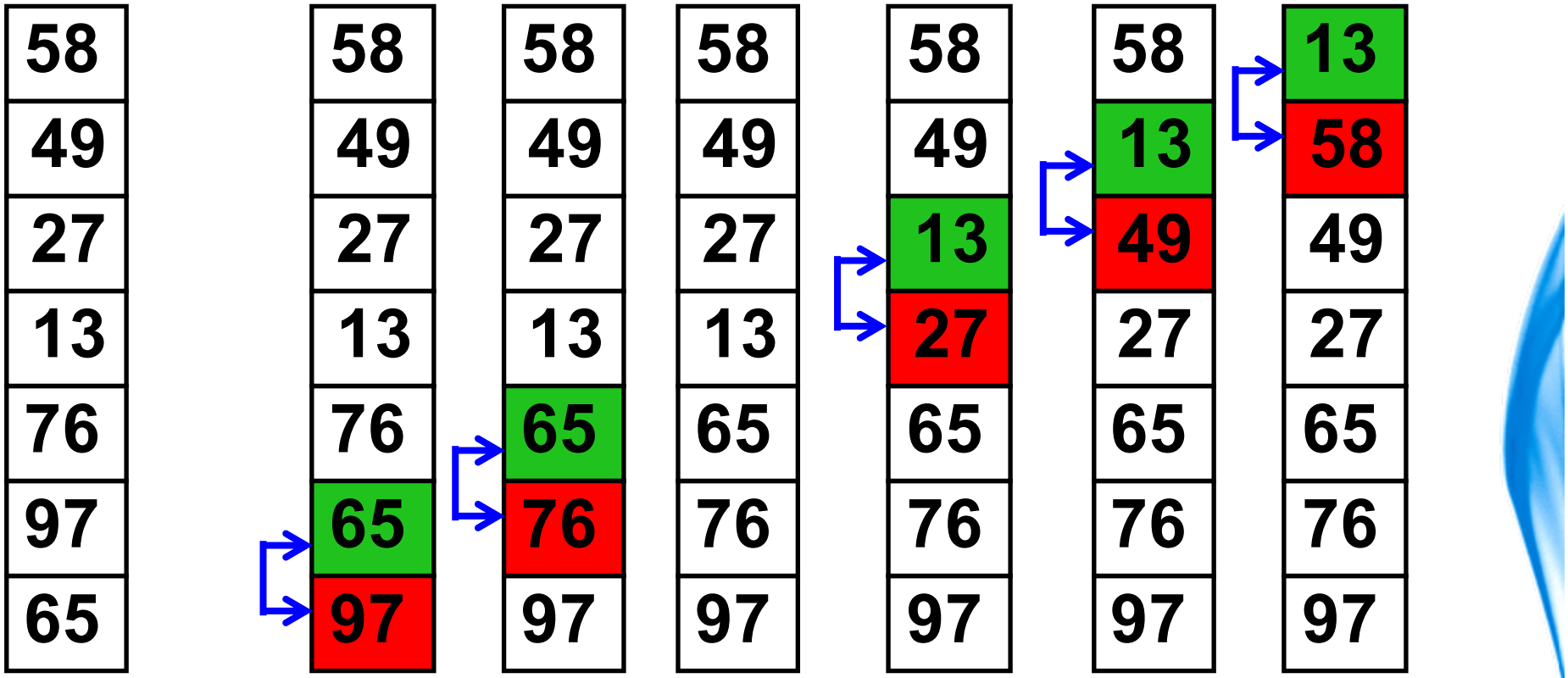
- 空间复杂度
  - 一个附加单元的存储空间，所以是O(1)。
- 时间复杂度
  - 该算法使用了循环的嵌套形式，复杂度为n(n+1)/2，所以为O(n2)。
- 算法是不稳定的。
- 直接选择排序中比较的次数比较多

3 冒泡排序

冒泡排序 (Bubble Sort)

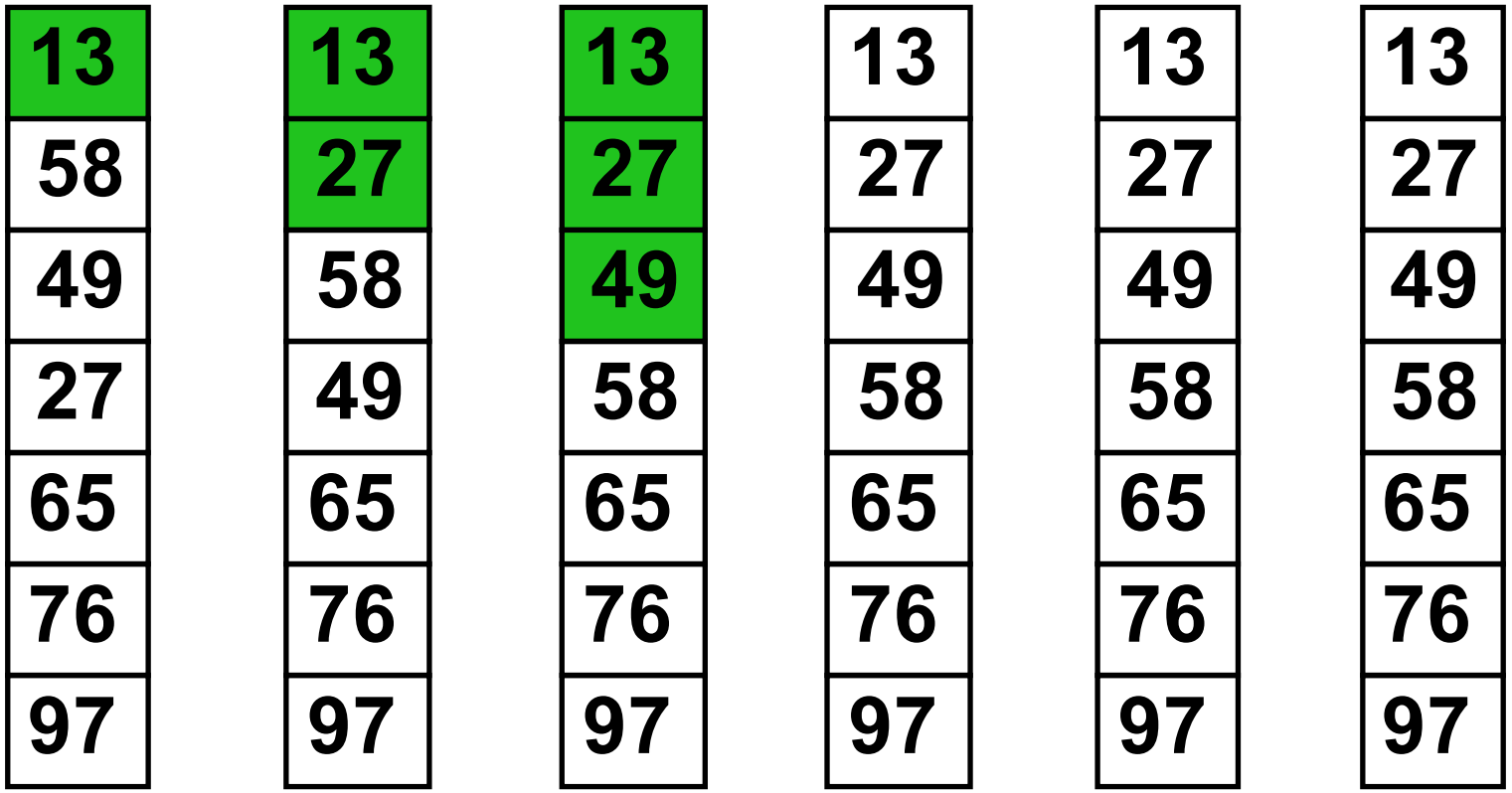
- 两两比较待排序记录的排序码，通过逐步交换相邻记录来消除逆序，直至全部正序为止
- 将待排序的记录按从后向前的顺序顺次两两比较，若为逆序则进行交换
- 将序列照此方法从尾到头处理一遍称作一趟起泡，一趟起泡的效果是将排序码值最小的记录交换到了最前位置
- 若某一趟起泡过程中没有任何交换发生，则排序过程结束

例13


58
49
27
13
76
97
65
58
49
27
13
76
65
97
58
49
27
13
65
76
97
58
49
27
13
65
76
97
58
49
13
27
65
76
97
58
13
49
27
65
76
97
13
58
49
27
65
76
97

第 1 趟全过程

各趟结果


13 58 49 27 65 76 97
13 27 58 49 65 76 97
13 27 49 58 65 76 97
13 27 49 58 65 76 97
13 27 49 58 65 76 97
13 27 49 58 65 76 97
第1趟
第2趟
第3趟
第4趟
第5趟
第6趟

算法描述

- 从R_n开始，两两比较相邻记录的关键字，即比较R_i和R_{i-1}(i=2，3，…，n)的排序码大小，若逆序(如$K_i<K_{i-1}$)，则交换R_i和R_{i-1}的位置，如此经过一趟排序，排序码最小的记录被安置在最前一个位置(R_1)上
- 然后再对后n−1个记录进行同样的操作，则具有次小排序码的记录被安置在第2个位置(R_2)上
- 如此反复，进行n−1趟冒泡排序后所有待排序的n个记录已经按排序码由小到大有序

算法伪代码

```
BUBBLE-SORT (A, n)
 1 ▷ 将最小排序码交换到前面
 2 for i ← 1 to n-1 do
 3       tag ← 0
 4       for  j ← n downto i+1 do
 5          if A[j] < A[j-1] then
 6                 A[j] ↔ A[j – 1]
 7                  tag ← 1
 8          end
 9       end
10       if  tag = 0 break
11 end
```

算法分析

- 稳定的
- 时间复杂度
  - 最好情况：正序，一趟排序，比较次数为 n-1，移动次数为 0
  - 最坏情况：逆序，n-1趟排序，比较次数和交换次数为 n(n-1)/2
  - 平均时间复杂度为O(n^2)
- 空间复杂度
  - 仅用一个辅助单元,复杂度为O(1)
- 当待排序序列是基本有序时，采用冒泡排序方法的效率较高