



目录

CONTENTS

01

【排序的概念】

02

【直接插入排序的算法思想】

03

【算法描述】

04

【算法分析】



01

排序的概念

排序的概念

假设含 n 个记录的序列 $\{R_1, R_2, \dots, R_n\}$

其相应的关键字序列为 $\{K_1, K_2, \dots, K_n\}$

需确定 $1, 2, \dots, n$ 的一种排列 $P_1, P_2, \dots, P_n$, 使其相应的关键字满足如下的非递减(或非递增)关系 $K_{P_1} \leq K_{P_2} \leq \dots \leq K_{P_n}$

即: 使记录的序列成为一个按关键字有序的序列

$$\{R_{P_1}, R_{P_2}, \dots, R_{P_n}\}$$

这样一种操作称为排序。

排序的概念

排序算法的稳定性：

假设 K_i 等于 K_j ($1 \leq i \leq n, 1 \leq j \leq n, i \neq j$), 且在排序前的序列中, R_i 领先于 R_j ($i < j$)。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。

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

待排记录的数据类型描述如下：

```
#define MAXSIZE 20;    //顺序表的最大长度 为20
typedef int KeyType;    //定义关键字类型为整型
typedef struct {
    KeyType    key;      //关键字项
    InfoType   otherinfo;
} RedType;             //记录类型
typedef struct {
    RedType r[MAXSIZE +1]; //r[0]闲置或者用作哨兵单元
    int length;            //顺序表长度
} SqList;               //顺序表类型
```

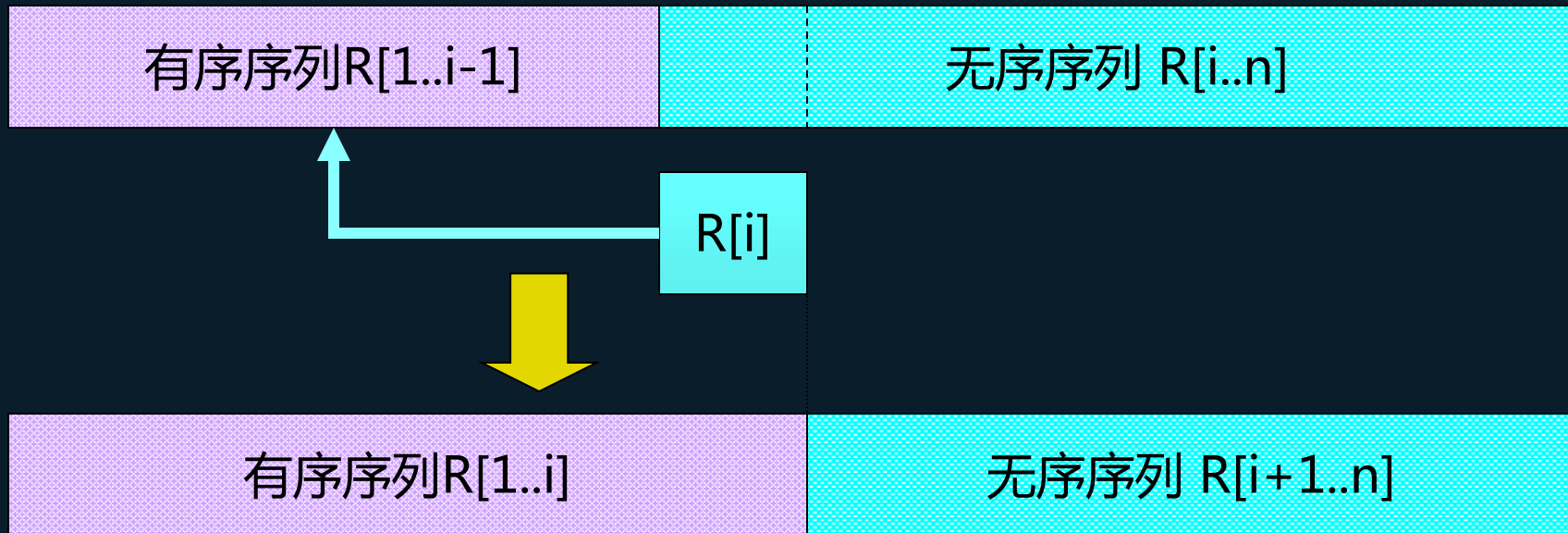


02

直接插入排序的算法思想

直接插入排序的算法思想

一趟直接插入排序的基本思想：是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。



一趟直接插入排序的过程为：

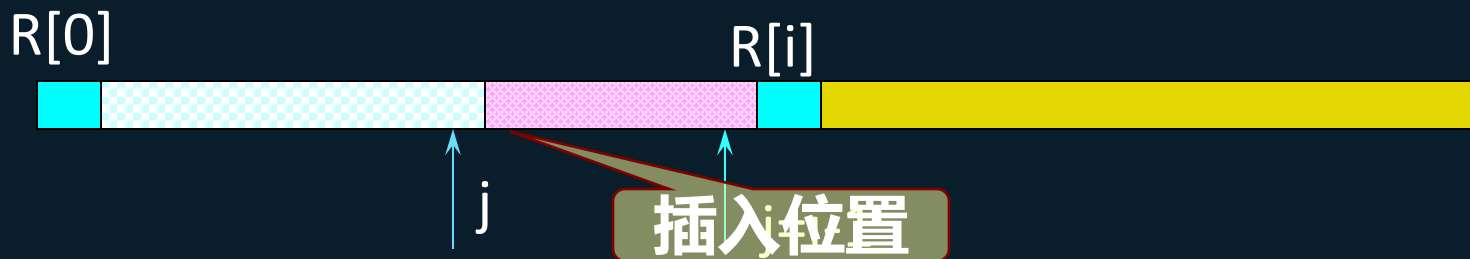
1 . 在 $R[1..i-1]$ 中查找 $R[i]$ 的插入位置，

$R[1..j].key \leq R[i].key < R[j+1..i-1].key$ ；

2 . 将 $R[j+1..i-1]$ 中的所有记录均后移一个位置；

3 . 将 $R[i]$ 插入到 $R[j+1]$ 的位置上。

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”

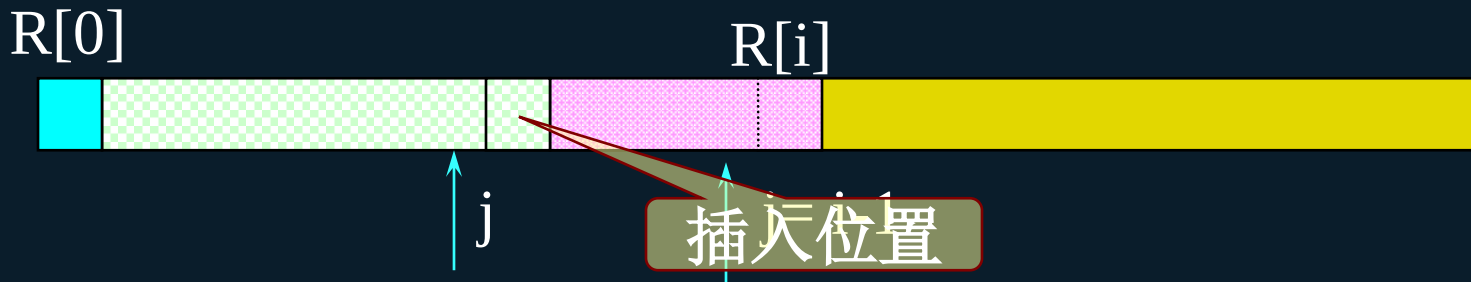
for ($j=i-1$; $R[0].key < R[j].key$; $--j$); // 从后往前找

循环结束， $R[i]$ 的插入位置为 $j+1$

在查找的过程中关键字大于 $R[i].key$ 的记录，可以同时向后移动；

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”

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/138052014121006044>