10.1 概 述

一、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

一般情况下，假设含*n*个记录的序列为{ R1, R2, …， Rn }， 其相应的关键字序列为 { K1, K2, …， Kn }

这些关键字相互之间可以进行比较，即在它们之间存在着这样一个关系 ：

$$Kp1 \leq Kp2 \leq \ldots \leq Kpn$$

按此固有关系将上式记录序列重新排列为

{ Rp1, Rp2, …， Rpn }

的操作称作排序。

二、内部排序和外部排序

若整个排序过程**不需要访问外存**便能完成，则称此类排序问题为**内部排序**；

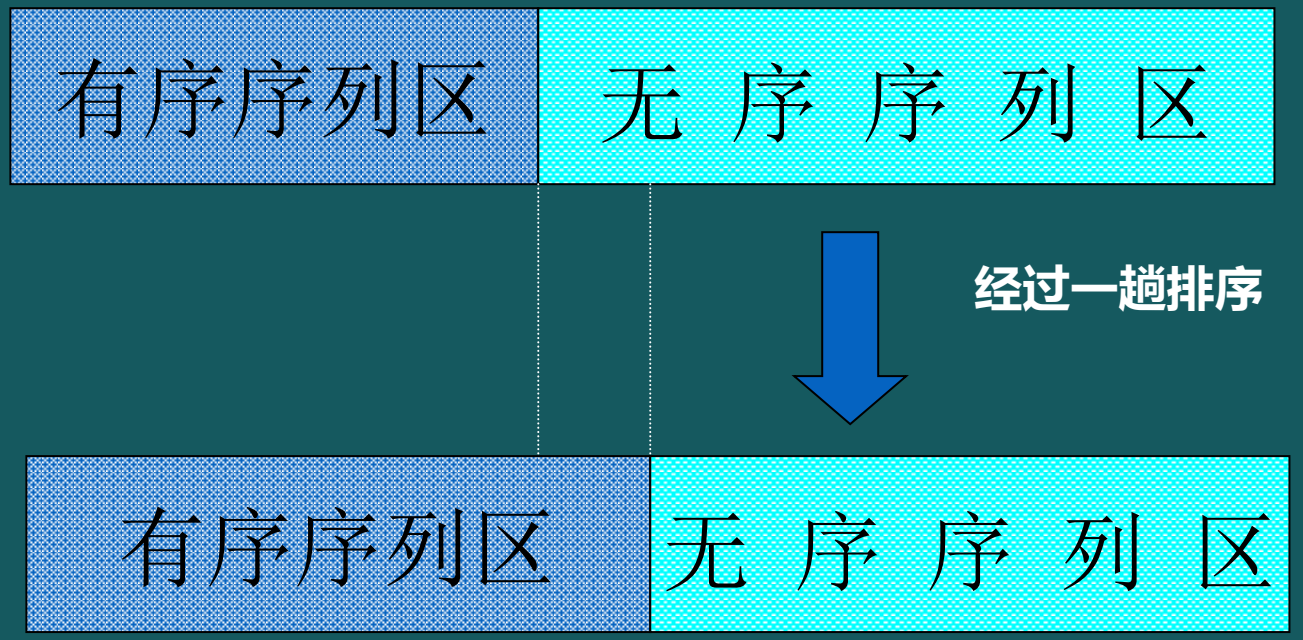
反之，若参加排序的记录数量很大，整个序列的排序过程不可能在内存中完成，则称此类排序问题为**外部排序**。

稳定和非稳定

假设$K_i=K_j$（$i\neq j$），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，若可能使排序后的的序列中R_j领先于R_i，则称所用的排序方法**是非稳定的**。

三、内部排序的方法

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。



基于不同的“**扩大**”有序序列长度的**方法，内部排序方法**大致可分下列几种类型：

插入排序类

交换排序类

选择排序类

归并排序类

其它方法

待排记录的数据类型定义如下:

```
#define MAXSIZE  1000 // 待排顺序表最大长度

typedef  int  KeyType;  // 关键字类型为整数类型

typedef  struct {
   KeyType   key;              // 关键字项
    InfoType  otherinfo;  // 其它数据项
} RcdType;                      // 记录类型

typedef  struct {
    RcdType    r[MAXSIZE+1]; // r[0]闲置
    int              length;          // 顺序表长度
} SqList;                             // 顺序表类型
```

1. 插入排序类

将无序子序列中的一个或几个记录“**插入**”到有序序列中，从而增加记录的有序子序列的长度。

2. 交换排序类

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。

3. 选择排序类

从记录的无序子序列中“选择”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。

4. 归并排序类

通过**“归并”**两个或两个以上的记录有序子序列，逐步增加记录有序序列的长度。

5. 其它方法

通常，在排序的过程中需进行下列两种基本操作：（1）比较两个关键字的大小；（2）将记录从一个位置移动至另一个位置。

待排序的记录序列可有下列3种存储方式：

（1）待排序的记录存放在地址连续的一组存储单元上。它类似于线性表的顺序存储；（2）待排序的记录存放在静态链表中，记录之间的次序关系由指针指示；（3）待排序的记录本身存放在一组地址连续的存储单元上，同时另设一个指示各个记录存储位置的地址向量。

10.1 概 述

一、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

一般情况下，假设含*n*个记录的序列为{ R1, R2, …， Rn }， 其相应的关键字序列为 { K1, K2, …， Kn }

这些关键字相互之间可以进行比较，即在它们之间存在着这样一个关系 ：

$$Kp1 \leq Kp2 \leq \ldots \leq Kpn$$

按此固有关系将上式记录序列重新排列为

{ Rp1, Rp2, …， Rpn }

的操作称作排序。

二、内部排序和外部排序

若整个排序过程**不需要访问外存**便能完成，则称此类排序问题为**内部排序**；

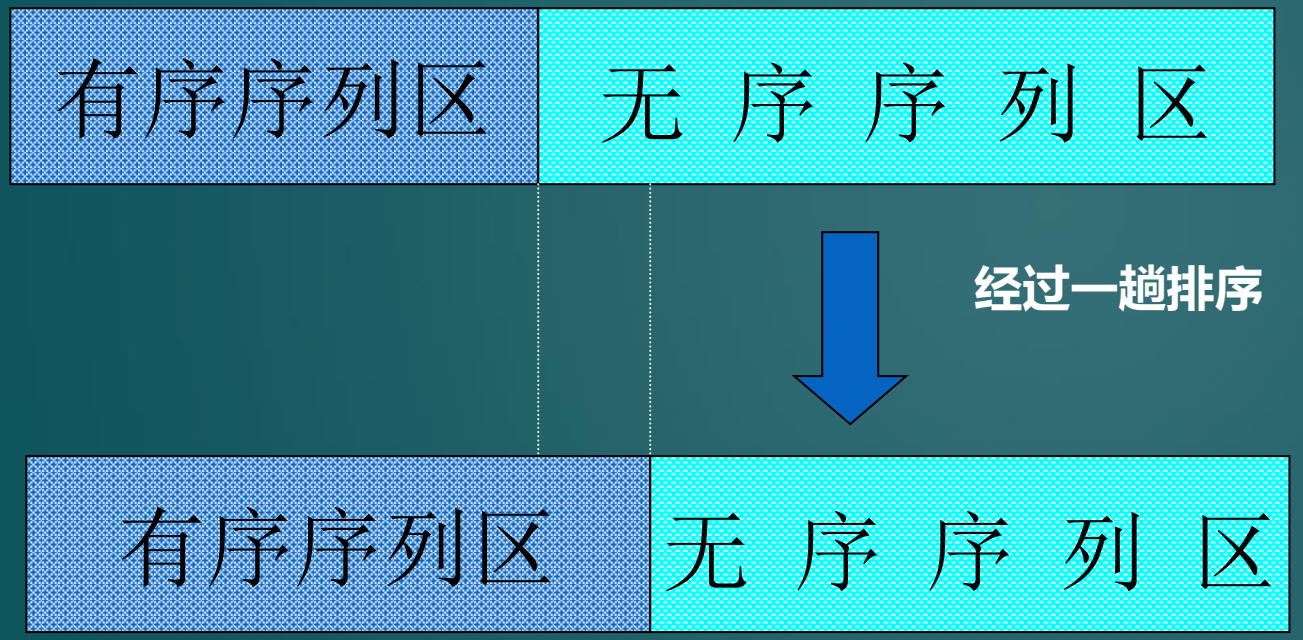
反之，若参加排序的记录数量很大，整个序列的排序过程不可能在内存中完成，则称此类排序问题为**外部排序**。

稳定和非稳定

假设$K_i=K_j$（$i\neq j$），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，若可能使排序后的的序列中R_j领先于R_i，则称所用的排序方法**是非稳定的**。

三、内部排序的方法

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。



基于不同的“**扩大**”有序序列长度的**方法，内部排序方法**大致可分下列几种类型：

插入排序类

交换排序类

选择排序类

归并排序类

其它方法

待排记录的数据类型定义如下:

```
#define MAXSIZE  1000 // 待排顺序表最大长度

typedef  int  KeyType;  // 关键字类型为整数类型

typedef  struct {
   KeyType   key;             // 关键字项
    InfoType  otherinfo;  // 其它数据项
} RcdType;                      // 记录类型

typedef  struct {
    RcdType    r[MAXSIZE+1]; // r[0]闲置
    int              length;          // 顺序表长度
} SqList;                             // 顺序表类型
```

1. 插入排序类

将无序子序列中的一个或几个记录“**插入**”到有序序列中，从而增加记录的有序子序列的长度。

2. 交换排序类

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。

3. 选择排序类

从记录的无序子序列中“选择”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。

4. 归并排序类

通过**“归并”**两个或两个以上的记录有序子序列，逐步增加记录有序序列的长度。

5. 其它方法

通常，在排序的过程中需进行下列两种基本操作：（1）比较两个关键字的大小；（2）将记录从一个位置移动至另一个位置。

待排序的记录序列可有下列3种存储方式：

（1）待排序的记录存放在地址连续的一组存储单元上。它类似于线性表的顺序存储；（2）待排序的记录存放在静态链表中，记录之间的次序关系由指针指示；（3）待排序的记录本身存放在一组地址连续的存储单元上，同时另设一个指示各个记录存储位置的地址向量。