第10章 排 序

10.1 排序的基本概念

- 在信息处理过程中，最基本的操作是查找。从查找来说，效率最高的是折半查找，折半查找的前提是所有的数据元素(记录)是按关键字有序的。需要将一个无序的数据文件转变为一个有序的数据文件。
- 将任一文件中的记录通过某种方法整理成为按(记录)关键字有序排列的处理过程称为排序。
- 排序是数据处理中一种最常用的操作。

10.1 排序的基本概念

1. 什么是排序

- 排序是将一批(组)任意次序的记录重新排列成按关键字有序的记录序列的过程。其定义为：

  给定一组记录序列：$\{R_1, R_2, \ldots, R_n\}$，其相应的关键字序列是$\{K_1, K_2, \ldots, K_n\}$。确定一个排列，使其相应的关键字满足如下非递减(或非递增)关系：$K_{p1} \le K_{p2} \le \ldots \le K_{pn}$的序列$\{K_{p1}, K_{p2}, \ldots, K_{pn}\}$，这种操作称为排序。

  - 关键字K_i可以是记录R_i的主关键字，也可以是次关键字或若干数据项的组合。

2. 内部排序和外部排序

- 在排序过程中，若整个数据表都是存放在内存中处理，排序时不涉及数据的内、外存交换，则称之为内部排序。
- 若排序过程中要进行数据的内、外存交换，则称之为外排序。

3. 内部排序的分类

- 根据内部排序算法是否基于关键字的比较，将内部排序算法分为基于比较的排序算法和不基于比较的排序算法。
  - 像插入排序、交换排序、选择排序和归并排序都是基于比较的排序算法。
  - 而基数排序是不基于比较的排序算法。

4. 算法的稳定性

- 当待排序记录的关键字均不相同时，排序的结果是唯一，否则排序的结果不一定唯一。
- 如果待排序的表中，存在有多个关键字相同的记录，经过排序后这些具有相同关键字的记录之间的相对次序保持不变，称这种排序方法是稳定的；
- 反之，若具有相同关键字的记录之间的相对次序发生变化，则称这种排序方法是不稳定的。

{5, 3, 2, 3, 9} → {2, 3, 3, 5, 9}

- 若待排序记录的关键字顺序正好和要排序顺序相同，称此表中记录为正序。
- 若待排序记录的关键字顺序正好和要排序顺序相反，称此表中记录为反序。
- 基于比较的排序算法中有些算法是与初始序列的正序或反序相关，有些算法与初始序列的正序和反序无关。

5. 基于比较的排序算法的性能

基于比较的排序算法中，主要进行以下两种基本操作：

- 比较：关键字之间的比较
- 移动：记录从一个位置移动到另一个位置

第一种操作是必不可少的；而第二种操作却不是必须的，取决于记录的存储方式。

评价排序算法的标准有：执行时间和所需的辅助空间，其次是算法的稳定性。时间是由比较和移动的次数之和确定的，两个记录的一次交换一般需要3次移动。

6. 排序数据的组织

- 在本章中，以顺序表作为待排序数据的存储结构（除基数排序采用单链表外）。
- 为简单起见，假设关键字类型为int类型。待排序的顺序表中记录类型定义如下：

```
// ------ 待排记录的数据类型 --------
#define MAXSIZE 20     // 一个用作示例的小顺序表的最大长度
typedef int KeyType;   // 定义关键字类型为整型
typedef struct {       // 记录类型
    KeyType key;       // 关键字项
    InfoType otherinfo;   // 其它数据项，具体类型在主程中定义
} RedType;

typedef struct {                  // 顺序表类型
    RedType r[MAXSIZE + 1];       // r[0]闲置或用作哨兵单元
    int length;                   // 顺序表长度
} SqList;
```

排序算法有许多，但就全面性能而言，还没有一种公认为最好的。每种算法都有其优点和缺点，分别适合不同的数据量和硬件配置。

END