# 交换排序

### 1. 冒泡排序

起泡排序,别名"冒泡排序",该算法的核心思想是将无序表中的所有记录,通过两两比较关键字, 得出升序序列或者降序序列。

例如,对无序表{49,38,65,97,76,13,27,49}进行升序排序的具体实现过程如图 所示:



上图所示是对无序表的第一次起泡排序,最终将无序表中的最大值 97 找到并存储在表的最后一个位置。具体实现过程为:

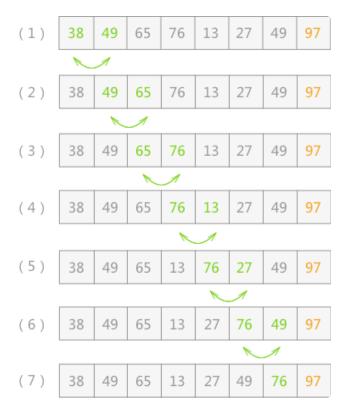
首先 49 和 38 比较,由于 38<49,所以两者交换位置,即从(1)到(2)的转变;

然后继续下标为 1 的同下标为 2 的进行比较,由于 49<65,所以不移动位置,(3)中 65 同 97 比较得知,两者也不需要移动位置;

直至(4),97 同76进行比较,76<97,两者交换位置,如(5)所示;

同样 97>13(5)、97>27(6)、97>49(7),所以经过一次冒泡排序,最终在无序表中找到一个最大值 97,第一次冒泡结束;

由于 97 已经判断为最大值,所以第二次冒泡排序时就需要找出除 97 之外的无序表中的最大值,比较过程和第一次完全相同。



通过一趟趟的比较,一个个的"最大值"被找到并移动到相应位置,直到检测到表中数据已经有序, 或者比较次数等同于表中含有记录的个数,排序结束,这就是起泡排序。

## 2. 关于冒泡的优化

基本的冒泡排序的实现方式,就是两个for循环,持续比较和交换。这种实现方式有一个明显的弊端,就是不论数组是否有序,两层 for 循环都要执行一遍,而我们是希望数组有序的时候,仅进行一轮判断,或者一轮都不进行(当然不判断,排序算法是不能知道数组是否有序的)。

#### 一次优化:

这里我们增加了一个标识数组是否有序,当冒泡排序过程中没有交换操作时, swapped = false,也意味着数组有序; 否则数组无序继续进行冒泡排序。不要小看这个变量奥,因为这个变量, 当数组有序的时候,冒泡排序的时间复杂度将降至 (因为其只需要执行一遍内层的 for 循环就可以结束冒 泡排序),没有这个变量,数组有序也需要 的时间复杂度。

#### 二次优化:

是否能够确定出已经有序部分和无序部分的边界呢?

#### 3. 快速排序

快速排序算法是在起泡排序的基础上进行改进的一种算法,其实现的基本思想是:通过一次排序将整个无序表分成相互独立的两部分,其中一部分中的数据都比另一部分中包含的数据的值小,然后继续沿用此方法分别对两部分进行同样的操作,直到每一个小部分不可再分,所得到的整个序列就成为了有序序列。