冒泡排序算法的几种优化与变形

胡伟东
（绍兴鲁迅中学，浙江 绍兴 312000）

摘　要：冒泡排序是一种经典的排序算法。根据冒泡排序的原理，总结了通过 flag 控制判断某轮排序是否有序、当有序时就提前结束循环的标志法，并通过记录每次排序最后一次交换位置，优化排序区间的区间控制法。同时根据冒泡排序的特点，总结了冒泡排序的变形、双向冒泡排序、分组冒泡排序和交替冒泡排序。

关键词：冒泡排序；优化；变形
DOI:10.16184/j.cnki.comprg.2023.04.032

1　概述

排序是数据处理中经常使用的运算方法。例如，电子邮件列表按照日期排序；购物网站上搜索到的某类商品按价格、销售等方式进行排序。冒泡排序是经典的排序算法之一。

冒泡排序是在一系列数据中对相邻两个数依次进行比较和调整，让较大的数“下沉（上冒）”、让较小的数“上冒（下沉）”的一种排序技术。以升序为例，冒泡排序算法将待排序的 n 个数中相邻两个数进行比较，将较大的数交换到后面的一个位置上。重复这一操作，直到处理完本轮数列中最后两个元素，称为一轮排序。当一轮排序完成时，最大的数就被轮换到本轮排序数列最右端位置上。重复上面的过程，经过 $n-1$ 轮后，就实现了数据升序排序。

在以下的算法中，使用 Python 语言实现排序。

2　传统冒泡排序

假定序列中有 n 个数，要进行从小到大的排序。若参与排序的数组元素共有 n 个，则需要 $n-1$ 轮排序。在第 i 轮排序中，从左端开始，相邻两数比较大小，若反序则将两者交换位置，直到比较第 $n+1-i$ 个数为止。第 1 个数与第 2 个数比较，第 2 个数和第 3 个数比较，一直到第 $n-i$ 个数与第 $n+1-i$ 个数比较，一共处理 $n-i$ 次。此时，第 $n+1-i$ 个位置上的数已经有序，后续就不需要参加以后的排序。

（1）第 1 轮冒泡排序先从第 1 个数和第 2 个数开始比较，若第 1 个数大于第 2 个数，则需要交换两者的位置；否则保持不变。重复这一过程，直到处理完本轮数列中最后两个数。

（2）第 2 轮冒泡排序与第 1 轮冒泡排序进行相同的排序，使大的数交换到 $n-2$ 的位置上。

（3）重复以上过程，共需经过 $n-1$ 轮冒泡排序后，数据实现升序排序。

例如，序列 [26，28，24，11]，排序过程如表1 所示。

表 1　4 数排序过程

轮数	a[0]	a[1]	a[2]	a[3]
0	26	28	24	11
1	26	24	11	28
2	24	11	26	28
3	11	24	26	28

具体排序算法如下。

```
for i in range(1,n):
    for j in range(0,n-i):
        if a[j]>a[j+1]:
            a[j],a[j+1]=a[j+1],a[j]
```

外循环：即主循环，其中 i 为本轮排序的左端点，需要将本轮的最大值移到右端点，需要处理 $n-1$ 轮，如果可以保证前 $n-1$ 个元素都交换到正确的位置上，那么最后一个数就是有序的，不需要进行处理。

内循环：即副循环，通过内循环将相邻元素进行比较和换位，把大的数往后移，一直循环 $n-1-i$ 轮这样可以将最大的数移到右端，而有序的数就不用再比较了。

对 n 个元素用冒泡排序进行排序时，共需比较次数如公式（1）所示：

$$(n-1)+(n-2)+\cdots+1=\frac{n(n-1)}{2} \quad (1)$$

其时间复杂度为 $O(n^2)$。

对于序列 [6，3，1，4，3]，前面的数字“3”和后面的数字“3”的顺序在整个冒泡排序过程中始终没有发生变化，即冒泡排序并不产生新的逆序对。如果两个元素相等，则相等元素的前后顺序不改变。因此，冒泡排序是一种稳定的排序算法。

3　优化 1——标志法

对于序列 [1，2，3，5，4]，在第 1 轮冒泡排序

作者简介：胡伟东，男，硕士，一级教师。

后，序列仍为 [1，2，3，4，5]，进而发现后续 3 轮中都没有发生数据交换，即其实不用再执行后面的几轮。算法只需要执行到某轮排序不需要交换数据即可。

在原来的程序中，加上一个标志变量 flag，用于记录在这一趟排序中是否发生了数交换。如果某轮结束后 flag 的值为 False，则说明本轮没有交换行为，数据已经全部有序，可以提前结束排序，具体如下。

（1）在第 1 轮冒泡排序中首先比较第 1 个数和第 2 个数，若前者大于后者，则交换两者位置，同时 flag=True；否则不换。重复这一过程，直到处理完最后两个数据。

（2）如果 flag=False，则结束循环；否则重复上述过程。具体算法如下。

```
flag=True
for i in range(1,n):
    flag=False
    for j in range(0,n-i):
        if a[j]<a[j+1]:
            a[j],a[j+1]=a[j+1],a[j]
            flag=True
    if not flag:
        break
```

4 优化 2——区间控制法

对于序列 [2，3，4，5，9，8，7]，采用“上冒”，第一轮排序结束后 [2，3，4，7，9，8]。当 7 和 9 交换位置后，后续的数字都没有互换，即在后续的排序中 [2，3，4，7] 不需要参与排序。得出结论：每轮遍历的区间边界都是由上一轮遍历最后一次发生交换的位置决定的。这样一来，可以引入标记变量 last 记录每轮排序最后一次交换的位置，并将 last 赋给下一轮遍历区间的终点。大大缩小了下一轮排序遍历区间，快速收缩待排序范围，从而提高了算法效率。

例如，序列 [2，3，4，5，9，8，7]，排序过程如表 2 所示。

表 2 7 个数排序过程

轮数	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
0	2	3	4	5	9	8	7
1	2	3	4	5	7	9	8
2	2	3	4	5	7	8	9

通过区间控制法，可以快速地结束排序。

（1）第 1 轮冒泡排序先比较第 $n-1$ 个数和第 $n-2$ 个数，若前者大于后者，则交换两者位置，同时记录最后一次交换的位置 last，重复这一过程，直到处理完数组中最后两个元素中的数据。

（2）下一轮排序从 $n-1$ 至 last，重复上面的过程，直到结束循环。

具体排序算法如下。

```
i=1
while i<n:
    last=n-1
    for j in range(n-1,i-1,-1):
        if  a[j]<a[j-1]   :
            a[j],a[j-1]=a[j-1],a[j]
             last=j
    i=last+1
```

5 变形 1——双向冒泡法

从经典的冒泡算法中可以发现，经典的冒泡排序是从一端开始比较的，经过第 1 轮排序后，一端为极值点，如此重复处理完成排序。换角度思考，在实现一端有序的同时，从另一端也进行排序，使另一端也为极值点，即实现双向冒泡排序。

（1）第 1 轮从左向右，将最大值移到右端点。

（2）第 2 轮从右向左，将最小值移到左端点。

（3）重复上述过程，数据实现交替排序。

例如，序列 [79，13，93，55，29，17]，排序过程如表 3 所示。

表 3 6 个数排序过程

轮数	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
0	79	13	93	55	29	17
1	13	17	79	55	29	93
2	13	17	29	55	79	93
3	13	17	29	55	79	93

具体算法如下：

```
left=0;right=n-1
while left<right:
for j in range(left,right):
    if a[j]>a[j+1]:
        a[j],a[j+1]=a[j+1],a[j]
        right=j
for j in range( right,left,-1 ):
    if a[j]<a[j-1]:
        a[j],a[j-1]=a[j-1],a[j]
        left=j
```

6 变形 2——分组冒泡法

经典的冒泡排序是对整个数列由小到大或由大到小进行排序，但是有时需要对数组元素进行隔位排序，如图 1 所示。

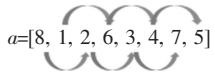

图 1　数组元素隔位排序

三轮排序后结果如图 2 所示。

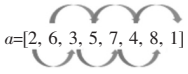

图 2　三轮排度后结果

实现了索引偶数位升序排序、奇数位降序排序。

例如，序列 [79，13，93，55，29，17]，排序过程如表 4 所示。

表 4　6 个数隔位排序过程

轮数	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
0	79	13	93	55	29	17
1	79	55	29	17	93	13
2	29	55	79	17	93	13

（1）第 1 轮冒泡排序首先比较第 1 个数和第 3 个数，若前者大于后者，则交换两者位置；否则不换位置。其次比较第 2 个数和第 4 个数，若前者小于后者，则交换两者位置，否则不需要交换。反复这一过程，直到处理完本轮最后两个数。

（2）重复上述处理过程，数据实现排序。

具体算法如下。

```
for i in range(1,n//2+1):
    k=1
    for j in range( 0,n-2-2*i ):
        if k*a[j]<k*a[j+2]:
            a[j],a[j+2]=a[j+2],a[j]
            k=-k
```

在程序中，通过变量 k 来控制升序和降序。

7　变形 3——交替冒泡法

对于数组，按最小、最大、第 2 小、第 2 大、第 3 小、第 3 大……排序。

例如，数组 a= [8，1，2，6，3，4，7，5]。

（1）第 1 轮从右向左排序，将最小值移到左端点。

（2）第 2 轮从右向左排序，将最大值移到左端第 2 个位置。

（3）重复上述过程，共经过 n–1 轮冒泡排序后，实现数据交替排序。

例如，序列 [79，13，93，55，29，17]，排序过程如表 5 所示。

具体算法如下：

```
for i in range(1,n):
    k=1
    for j in range(n-1,i-1,-1):
        if k*a[j]<k*a[j-1]:
            a[j],a[j-1]=a[j-1],a[j]
    k=-k
```

在程序中，通过变量 k 来控制极值。

表 5　6 个数交替排序过程

轮数	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
0	79	13	93	55	29	17
1	13	79	17	93	55	29
2	13	93	79	17	55	29
3	13	93	17	79	29	55
4	13	93	17	79	55	29
5	13	93	17	79	29	55

8　结语

通过对冒泡排序优化与变形的总结，提高了学生自行总结冒泡算法知识和运用冒泡算法的能力，将“死记硬背”的程序转化为属于学生的编程能力，同时训练了学生的计算思维，提高了课堂教学效率。在一系列冒泡算法优化与变形活动中，学生慢慢地理解冒泡算法的核心思想，提升了理解程序、解决问题的能力。

冒泡排序虽然有多种优化和变形，在一定程度上给教师的教学和学生的学习造成了困难，但是只要抓住冒泡排序的核心思想，就能提高学生提炼算法和运用算法的能力。将学生背下来的程序转化为真正属于他们的程序，同时锻练了学生的思维能力，提升学生的信息技术核心素养。

参考文献

[1] 张乃孝. 算法与数据结构—C 语言描述 [M]. 2 版. 北京：高等教育出版社，2008.

[2] 潘颖. 利用往返比较改进冒泡排序算法 [J]. 科技信息，2007 (29)：404–405.

[3] 王晓东. 计算机算法设计与分析 [M]. 3 版. 北京：电子工业出版社，2003.

[4] 杨朝霞. 分析和计算算法效率的便捷方法 [J]. 兰州交通大学学报，2004 (4)：78–82.

[5] 孙立会，周丹华. 国际儿童编程教育研究现状与行动路径 [J]. 开放教育研究，2019，25 (2)：23–35.

[6] 林玉慈. 高中数学课程中的逻辑推理及教学策略研究 [D]. 长春：东北师范大学，2019.

[7] 陶增乐. 普通高中课程标准实验教科书信息技术基础必修 [M]. 杭州：浙江教育出版社，2016.