# 双向选择排序算法①

袁关伟

(69027 部队, 乌鲁木齐 830000)

**摘 要**: 为丰富 $O(n^2)$ 阶排序算法的种类, 以更好地服务于教学科研和日常应用, 提出了一种新的排序算法—双向选择排序算法. 通过数学方法分析得知: 该算法的时间复杂度为 $O(n^2)$, 空间复杂度为 $O(1)$. 通过实验对比得知: 在相同条件下, 该算法的运行时间平均为冒泡排序的 27%、简单选择排序的 62%、直接插入排序的 88%.

**关键词**: 双向; 选择; 排序

## Bidirectional Selection Sort Algorithm

YUAN Guan-Wei

(69027 Troops, Urumqi 830000, China)

**Abstract**: In order to add the kinds of the sort algorithms which have the complexity of $O(n^2)$ and then serve teaching and daily application better, the paper proposes a new sort algorithm—bidirectional selection sort algorithm. Through mathematical method that the time complexity of the new algorithm is $O(n^2)$ and its space complexity is $O(1)$. It is known by empirical method that the running time of the new algorithm is averagely 27 percent of bubble sort, and 62 percent of simple selection sort and 88 percent direct insertion sort in same condition.

**Key words**: bidirectional; select; sort

排序算法是日常生活中使用最为频繁的算法之一, 2000 年曾被评为对科学和工程计算影响最大的 10 大问题之一[1]. 理论计算机科学发展至今, 计算机科学工作者发明了不少排序算法, 其中使用比较广泛、研究比较成熟、已被写入高校《数据结构与算法》教材的有直接插入排序、简单选择排序、冒泡排序、快速排序、堆排序和归并排序[2-10]. 近年来, 很多理论计算机工作者又对快速排序、堆排序和归并排序等时间复杂度为 $O(n\log n)$ 的算法进行改进, 提出了一些新算法. 如为改进快速排序的性能, 王善坤等提出了三路划分快速排序的改进算法[11], 庹清等提出了按位拆分快速排序算法[12]; 为改变快速排序的稳定性, 邵顺增提出了稳定快速排序算法[13]; 为在链式存储结构(链式存储结构的实际应用非常广泛)中使用快速排序, 白宇等提出了单向链表快速排序算法[14]. 又如为提高堆排序的性能, 吴尚至提出了改进的堆排序算法[15], 唐开山提出了 ASH 排序算法[16]; 为降低归并排序的时间复杂度, 范时平等提出了线性原地二路归并算法[17]; 为降低归并排序的空间复杂度, 白宇等提出了深度优先稳定原地归并排序算法[18]. 事实上, 对时间复杂度为 $O(n^2)$ 的排序算法进行改进也是有意义的, 因为并不是在所有情况下时间复杂度为 $O(n\log n)$ 的算法都好于时间复杂度为 $O(n^2)$ 的算法, 譬如在对数据规模不大或对时间要求不高但对空间要求很高(要求空间复杂度低)或作为排序算法的入门教学案例时, 时间复杂度为 $O(n\log n)$ 的算法较时间复杂度为 $O(n^2)$ 的算法并无优势, 这也应证了美 Knuth D.E.在其经典著作《计算机程序设计艺术》中所说的那句话: 几乎所有的算法都值得去记住, 因为在某些特定应用中, 它们是最好的[19]. 鉴于直接插入排序已有唐开山对其进行改进(循环插入排序法和 4 路插入排序法)[20,21], 冒泡排序已有淦艳等对其进行改进(带标记冒泡排序、双向冒泡排序和交替冒泡排序)[22], 本文选择对简单选择排序进行改进.

① 收稿时间:2015-05-02;收到修改稿时间:2015-06-27

## 1 算法设计

### 1.1 算法思想

双向选择排序算法的基本思想: 每趟从无序区中选出关键字最大(小)和最小(大)的记录, 并将它们分别放在当前无序区的第一个位置和最后一个位置, 自此以后当前无序区的第一个位置和最后一个位置就不再属于无序区了. 显然, 经过一趟排序后, 无序区中的记录减少两个. 重复执行上述操作, 直至无序区不存在(问题规模为偶数的情况)或无序区中只剩下一个记录(问题规模为奇数的情况)为止. 需要指出的是, 当无序区中只剩下一个记录时, 全体记录也一定按关键字有序了, 这从算法最后一次选择关键字时的情景很容易看出.

### 1.2 算法实施策略

以非递增排序为例(非递减排序同理), 实施双向选择排序算法一共需要 3 步, 具体如下:

(1)分别从当前无序区的两端开始扫描, 扫描的结果是选出关键字最大记录和最小记录.

(2)将关键字最大记录和当前无序区中的第一个记录比较, 若当前无序区中的第一个记录不是关键字最大记录, 也不是关键字最小记录, 则将它们直接交换; 若当前无序区中的第一个记录不是关键字最大记录, 但是关键字最小记录,那么在交换前, 必须将关键字最小记录的下标更改成关键字最大记录的下标, 因为交换后, 关键字最小记录的位置已不再是原先选出来的位置, 而是关键字最大记录的原位置, 而关键字最小记录的位置在它和当前无序区中最后一个记录交换时还必须用到. 此步可谓实施本算法最为关键的一步, 唯有如此, 才能确保关键字最小记录和当前无序区中的最后一个记录交换时不出错.

(3)将关键字最小记录和当前无序区中的最后一个记录比较, 若当前无序区中的最后一个记录不是关键字最小记录, 则将它们交换.

## 2 算法实现

为简单起见, 以记录的关键字类型为整型为例, 算法用 C 语言实现如下:

```
void bidirectional_select_Sort(int R[])
{ /*函数名: 双向选择排序函数 程序员: 袁关伟 时间:
   2014 年 11 月 地点: 北师大*/
    int max,min,temp,i,j,k1,k2,max_idx,min_idx;
    for(i=0,j=N-1;i<j;i++,j--)
    {// N 为宏定义标识符, 代表问题规模
        max=R[i];
        min=R[j];
        for(k1=i+1,k2=j-1;k1<N-i;k1++,k2--)
        {
            if(R[k1]>max)
            {
                max=R[k1];
                max_idx=k1;
            }
            if(R[k2]<min)
            {
                min=R[k2];
                min_idx=k2;
            }
        }
        if(R[i]!=max)
        {
            if(R[i]= =min)
                min_idx=max_idx;
            temp=R[i];
            R[i]=R[max_idx];
            R[max_idx]=temp;
        }
        if(R[j]!=min)
        {
            temp=R[j];
            R[j]=R[min_idx];
            R[min_idx]=temp;
        }
    }
}
```

## 3 算法测试及应用

现以中国主要河流为例, 在 VC++ 6.0 中用 C 语言编写测试程序对其按“长度”非递增排序, 测试结果如图 1 所示,其中测试数据来自文献[23].


图 1 双向选择排序算法测试图

## 4 算法分析

### 4.1 性能及稳定性分析

设 $n$ 是问题规模，$f(n)$ 是基本操作重复执行的次数. 由算法实现部分可知，双向选择排序算法的基本操作为第二个for循环里的比较和交换操作，此基本操作重复执行的次数即可决定算法的时间复杂度. 不难看出，比较和交换操作的次数是由第一个for循环和第二个 for 循环的执行次数决定的，第一个 for 循环最多执行 $\lfloor n/2 \rfloor$ 次，且第一个 for 循环每执行一次，第二个 for 循环就要执行 $(n-i-1)-(i+1)+1=n-2i-1$ 次. 于是可得如下关系式:

$$f(n)=\sum_{i=0}^{\lfloor n/2 \rfloor -1}(n-2i-1)$$

(1)当 $n$ 为偶数时(计算过程的原理是等差数列的求和公式):

$$f(n)=\sum_{i=0}^{\lfloor n/2 \rfloor -1}(n-2i-1)$$

$$=\lfloor n/2 \rfloor (n-1+1)/2$$

$$=\lfloor n/2 \rfloor n/2$$

$$=n^2/4=O(n^2)$$

(2)当 $n$ 为奇数时(计算过程的原理是等差数列的求和公式):

$$f(n)=\sum_{i=0}^{\lfloor n/2 \rfloor -1}(n-2i-1)$$

$$=\lfloor n/2 \rfloor (n-1+2)/2$$

$$=\lfloor n/2 \rfloor (n+1)/2$$

$$=(n-1)(n+1)/4$$

$$=(n^2-1)/4$$

$$=n^2/4-1/4=O(n^2)$$

所以,双向选择排序算法的时间复杂度为 $O(n^2)$. 因为实现双向选择排序算法需要的辅助变量不随问题规模 $n$ 的变化而变化，所以它是一个就地排序算法，其空间复杂度为 $O(1)$. 同时，经分析和实际测试，在记录交换的过程中，关键字相同的记录的相对次序可能发生变动，所以双向选择排序是一个不稳定排序.

### 4.2 算法对比

#### 4.2.1 理论对比

根据上述分析结果，现将双向选择排序算法与经典简单排序算法——直接插入排序、简单选择排序、冒泡排序作一性能对比，对比结果如表 1 所示.

表 1 双向选择排序与经典简单排序复杂度对比表

| 对比项 / 算法名 | 时间复杂度 | | | 空间复杂度 | 稳定性 |
|---|---|---|---|---|---|
| | 最坏情况 | 最好情况 | 平均情况 | | |
| 直接插入排序 | $O(n^2)$ | $O(n)$ | $O(n^2)$ | $O(1)$ | 稳定 |
| 简单选择排序 | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ | $O(1)$ | 不稳定 |
| 冒泡排序 | $O(n^2)$ | $O(n)$ | $O(n^2)$ | $O(1)$ | 稳定 |
| 双向选择排序 | $O(n^2)$ | $O(n^2)$ | $O(n^2)$ | $O(1)$ | 不稳定 |

#### 4.2.2 实验对比

根据理论对比结果，不难看出，双向选择排序算法和经典简单排序算法——直接插入排序、简单选择排序、冒泡排序的时间复杂度和空间复杂度都处于同一阶. 为了更精确地对比双向选择排序算法与经典简单排序算法的性能优劣，下面利用时间度量函数——time()函数对它们在相同条件下的实际时间耗费进行统计，其中的测试数据是通过 rand()函数随机生成的. 统计结果如表 2 所示，其中 w 和 s 分别表示数量单位“万”和时间单位“秒”. 本测试的硬件环境为: Cerleron(R) T3100 1.90GHz 双核 CPU, 3GB 内存; 软件环境为: Windows XP 操作系统, Visual C++ 6.0 集成开发环境.

表 2　双向选择排序与经典简单排序实际耗时对比表

| name / n(w) | Insertion(s) | Selection(s) | Bubble(s) | Bidirectional Selection(s) |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 |
| 2 | 1 | 1 | 3 | 1 |
| 3 | 2 | 2 | 5 | 1 |
| 4 | 3 | 5 | 9 | 2 |
| 5 | 5 | 7 | 15 | 4 |
| 6 | 6 | 9 | 21 | 6 |
| 7 | 8 | 13 | 29 | 8 |
| 8 | 11 | 17 | 38 | 11 |
| 9 | 14 | 22 | 48 | 13 |
| 10 | 18 | 27 | 58 | 16 |
| 11 | 21 | 32 | 71 | 20 |
| 12 | 25 | 38 | 84 | 23 |

以表 2 中后 11 次实验结果来分析(第 1 次存在分母为 0 的情况, 故略). 设冒泡排序、简单选择排序、直接插入排序在后 11 次实验中与双向选择排序运行时间的平均比率分别为 $q_1$、$q_2$、$q_3$，则有:

$$q_1=[(1/3+1/5+\ldots+23/84)/11]\times100\%=27\% \quad (1)$$

$$q_2=[(1/1+1/2+\ldots+23/38)/11]\times100\%=62\% \quad (2)$$

$$q_3=[(1/1+1/2+\ldots+23/25)/11]\times100\%=88\% \quad (3)$$

所以, 在相同条件下, 双向选择排序的运行时间平均为冒泡排序的 27%、简单选择排序的 62%、直接插入排序的 88%.

## 5　结语

本文提出了双向选择排序算法, 经理论分析得知, 其时间复杂度和空间复杂度与直接插入排序、简单选择排序、冒泡排序处于同一阶; 经实验分析得知, 在相同条件下, 其运行时间平均为冒泡排序的 27%、简单选择排序的 62%、直接插入排序的 88%. 排序算法历来是《数据结构与算法》课程中重点而基础的问题, 本文算法的提出, 不仅为排序算法家族增添了一个新成员, 而且可供《数据结构与算法》任课教师用作学生的课堂引申题、课后思考题和毕业设计题, 因而它具有一定的学术价值. 另外, 在现实生活中, 几乎所有可以使用简单选择排序的场合, 它都可以更好地替补上去, 这决定了它具有一定的应用价值.

## 参考文献


1 Dongarra J, Sullivan F. Gust editors introduction to the top 10 aigorithm. IEEE Computing in Science & Engineering, 2000, 2(1): 22–23

2 Horowitz E, Sahni S, Anderson-Freed S.数据结构基础.朱仲涛,译.第 2 版.北京:清华大学出版社,2009.

3 严蔚敏,吴伟民.数据结构(C 语言版).北京:清华大学出版社,1997.

4 张乃孝.算法与数据结构—C 语言描述.第 2 版.北京:高等教育出版社,2006.

5 张铭,王蛟龙,赵海燕.数据结构与算法.北京:高等教育出版社,2008.

6 李春葆,尹为民,等.数据结构教程.第 2 版.北京:清华大学出版社,2007.

7 廖明宏,郭福顺,等.数据结构与算法.第 4 版.北京:高等教育出版社,2007.

8 朱战立.数据结构—使用 C 语言.第 4 版.北京:电子工业出版社,2009.

9 陈媛,何波,卢玲,等.算法与数据结构.第 2 版.北京:清华大学出版社,2011.

10 袁关伟.面向 GIS 的计算机基础算法及其数据结构研究[硕士学位论文].昆明:西南林业大学,2012.

11 王善坤,陶祯蓉.一种三路划分快速排序的改进算法.计算机应用研究,2012,29(7):2513–2516.

12 庹清,向贵成,宋耀虎.改进的按位拆分快速排序算法.计算机应用,2011,31(z1):183–191.

13 邵顺增.稳定快速排序算法研究.计算机应用与软件,2014,31(7):263–266.

14 白宇,郭显娥.单向链表快速排序算法.计算机工程与科学,2014,36(1):115–120.

15 吴尚志.改进的堆排序算法及其复杂度分析.西北师范大学学报(自然科学版),2002,38(3):24–26.

16 唐开山.堆排序算法研究.绍兴文理学院学报,2004,10(24):16–18.

17 范时平,汪林林,何先刚.一种线性原地二路归并算法.计算机科学,2004,31(12):221–225.

18 白宇,郭显娥.深度优先稳定原地归并排序的高效算法.计算机应用,2013,33(4):1039–1042,1060.

19 Knuth DE.计算机程序设计艺术(第 3 卷 排序与查找).苏运霖,译.北京:国防工业出版社,2002.

20 唐开山.循环插入排序法.计算机工程与应用,2005,41(12):88–91.

21 唐开山.4 路插入排序法.计算机工程,2006,32(1):51–53.

22 淦艳,杨有,余平.冒泡排序算法及其改进算法的实验分析.重庆三峡学院学报,2011,27(132):53–57.

23 北京天域北斗图书有限公司.中国交通旅游图册.北京:中国旅游出版社,2008.