DOI:10.15998/j.cnki.issn1673-8012.2010.03.021
2010年6月 重庆文理学院学报（自然科学版） Jun., 2010
第29卷 第3期 Journal of Chongqing University of Arts and Sciences (Natural Science Edition) Vol.29 No.3

五种排序算法的性能分析

涂 艳，杨 有
（重庆师范大学 信息科学与工程学院，重庆 沙坪坝 400047）

[摘 要]排序是计算机科学中基本的研究课题之一，其目的是方便记录的查找、插入和删除.通过描述冒泡、选择、插入、归并和快速5种排序算法，总结了它们的时间复杂性和空间复杂性，指出5种排序算法可分为平方阶排序和线性对数阶排序两类.通过实验验证了5种排序算法在随机、正序和逆序3种情况下的性能，指出排序算法的适用原则：当记录较小时，可采用插入或选择排序；当记录基本有序时，可选用插入或冒泡排序；当记录较大时，则应选择快速排序或归并排序.

[关键词]排序算法；冒泡排序；选择排序；插入排序；归并排序；快速排序
[中图分类号] TP301 [文献标志码] A [文章编号] 1673－8012(2010)03－0045－06

排序是计算机程序设计中的一种重要操作，它的功能是将任意序列的数据元素或记录重新按关键字顺序排列成有序的序列.有序序列为记录的查找、插入和删除提供了方便，可以有效提高搜索效率.因此，研究各类排序方法是计算机研究中的重要课题之一.

根据待排序记录数量及其在排序过程中涉及的存储器，可将排序方法分为两大类[1]：一类是内部排序，指的是待排序记录存放在计算机存储器中进行的排序过程；另一类是外部排序，指的是待排序记录的数量很大，以致于内存一次不能容纳全部记录，在排序过程中尚需对外存进行访问的排序过程.

考虑到外部排序涉及的待排序记录数量大，可以采取分治的思想（即先分解，再递归求解，然后合并）[2]，将其划分成几段合适的待排序记录，然后对每一小段采用内部排序方法.换句话说，就是将外部排序转化为内部排序，所以为了进一步研究外部排序，需先对内部排序进行深入的讨论.如果在排序过程中依据不同原则对内部排序方法进行分类，则大致可分为插入排序、冒泡排序、选择排序、归并排序和快速排序等5类.

通常，排序记录存储具有如下3种形式：

①待排序的一组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结构，在序列中相邻的2个记录 R_j 和 R_{j+1}（$j=1,2,\cdots,n-1$）其存储位置也相邻.在这种存储方式中，记录之间的次序关系由其存储的位置决定，排序通过移动记录来实现.

②一组待排序记录存放在静态链表中，记录之间的次序关系由指针指示，则实现排序不需要移动记录，仅需移动指针即可.

③待排序记录本身存储在一组地址连续的存储单元内，同时另设一个指示各个记录存储位置的地址向量，在排序过程中不移动记录本身，而移动地址向量中这些记录的“地址”，在排序结束之后再按照地址向量中的值调整记录的存储位置.

为便于后续讨论，进行如下假设：

考虑到易于理解、随机访问和尽量添加更少的附加信息来实现记录的存储（形式②、③需要另外记录指针和地址信息，附加信息比较多），将待排序记录的一组记录以形式①进行存储，而且记录的关键字为整数；没有特殊说明时，排序均

[收稿日期] 2010－04－15
[基金项目] 云南省2009年社会发展科技计划项目（2009ZC128M）.
[作者简介] 涂艳（1988－），男，重庆南岸区人，主要从事软件理论与技术方面的研究.
杨有（1965－），男，重庆梁平人，博士，副教授，主要从事数字图像处理方面的研究.

认为按升序排序.

1 算法与特性

1.1 冒泡排序

冒泡排序的基本思想是[1,3]:首先将第 1个记录的关键字和第 2个记录的关键字进行比较,若为逆序,则将 2个记录交换,然后比较第 2个和第 3个记录的关键字,依次类推,直至 $n-1$个记录和第 n个记录的关键字进行过比较为止.上述过程称为第 1次排序,其结果是让最大的记录被安置到最后一个记录的位置上,然后再进行下一次排序,直至整个序列有序为止.

冒泡排序算法用 C语言描述如下:

```
template< class Type>
void BubbleSort( Type * R, int n)
{
int i, j;
Type temp;
for( i=0; i< n-1; i++)
    for( j=0; j< n- i-1; j++)
    if( R[ j]> R[ j+1] )    //满足条件,则交换
    {
        temp= R[ j];
        R[ j]= R[ j+1];
        R[ j+1]= temp;
    }
}
```

由此算法可以分析出它的效率,在最好情况下,只需通过 $n-1$次比较,不需要移动关键字,即时间复杂度为 $O(n)$(即正序);在最坏情况下是初始序列为逆序,则需要进行 $n-1$次排序,需进行 $\sum_{i=n}^{2}(i-1)=\frac{n(n-1)}{2}$次比较,因此在最坏情况下时间复杂度为 $O(n^2)$,附加存储空间为 $O(1)$.

1.2 选择排序

选择排序的基本思想是[1,3]:每一次从待排序的记录中选出关键字最小的记录,顺序放在已排好序的文件的最后,直到全部记录排序完毕.常用的选择排序方法有直接选择排序和堆排序,考虑到简单和易理解,这里讨论直接选择排序.直接选择排序的基本思想是 n个记录的文件的直接排序可经过 $n-1$次直接选择排序得到有序结果.具体算法分为 3个步骤:①初始状态,无序区 $R[0,\cdots,n-1]$,有序区为空;②第 1次排序.在无序区 $R[0,\cdots,n-1]$中选出关键字最小的记录 $R[k]$,将它与无序区的第 1个记录 $R[0]$交换,使有序区记录增加 1个,无序区记录减少 1个;③第 i次排序.在开始时,当前有序区和无序区分别为 $R[0,\cdots,i]$和 $R[i+1,\cdots,n-1]$($0\leqslant i\leqslant n-2$),该次排序从当前无序区中选出关键字最小的记录 $R[k]$,将它与无序区的第 1个记录 $R[i]$交换,使有序区记录增加 1个,无序区记录减少 1个.

选择排序算法用 C语言描述如下:

```
template< class Type>
void SelectSort( Type R[], int n)
{
int i, j, k;
for( i=1; i< n; i++)
{
    k= i;    //记下标位置
    for( j= i+1; j<= n; j++)
        if( R[ j]< R[ k] )
            k= j;
    if( k!= i)
    {
        R[ 0]= R[ j]; R[ j]= R[ k]; R[ k]= R[ 0];    //R[ 0]作为暂存单元
    }
}
}
```

容易推出,在直接选择排序过程中所需进行记录移动的操作次数最少为 0,最大值为 $3(n-1)$.然而,无论记录的初始排序如何,所需进行的关键字间的比较次数相同,均为 $n(n-1)\backslash 2$,时间复杂度为 $O(n^2)$,附加存储空间为 $O(1)$.

1.3 插入排序

插入排序的基本思想是[1,3]:每次将 1个待排序的记录按其关键字大小插入到前面已经排好序的子文件中适当的位置,直到全部记录插入完成为止.

插入排序分直接插入排序和希尔排序 2类.考虑到简单、易理解的因素,这里讨论直接插入排序,其基本思想与打扑克牌时整理手上的牌非常相似:初始时 $R[0]$自成一个有序区,无序区为 $R[1,\cdots,n-1]$,从 $i=1$至 $i=n-1$为止,依次将 $R[i]$插入到当前的有序区中,生成含 n个记录的有序区.

插入排序算法用 C语言描述如下:

```
template< class Type>
```

```
void InsertSort(Type R[], int left,int right)
{
Type temp;
int i,j;
for(i=left+1; i<=right;i++)  //逐步扩大有序表
    if(R[i]<R[i-1])  //逆序才找插入的位置
    {
        temp=R[i]; j=i-1;
        do{
            R[j+1]=R[j]; j--;
        }while(j>=left&&temp<R[j]);
R[j+1]=temp;
    }
}
```

当待排序文件为正序时,所需进行的关键字间比较的次数达最小值 $n-1$,记录不需要移动;反之,逆序时比较次数达最大值 $(n+2)(n-1)\backslash 2$,移动次数为 $(n+4)(n-1)\backslash 2$,因此,其时间复杂度为 $O(n^2)$,附加存储空间为 $O(1)$.

1.4 归并排序

归并排序的基本思想是[2,4]:采用分治策略,将待排序文件分成大小大致相同的 2个子集合,分别对 2个子集合进行排序,最终将排好序的子集合合并成所要求的排好序的集合.假设初始序列含有 n个记录,则可以将每个记录看成是长度为 1的子序列,然后两两归并,得到对 $\lceil n\backslash 2\rceil$个长度为 2或 1的有序子序列;再两两归并,如此重复,直至得到 1个长度为 n的有序序列为止.附加存储空间为 $O(n)$.

归并排序算法用 C语言描述如下:

```
template<class Type>
void merge(Type R1[], Type R2[], int,l int m, int r)
{ //将 R1[l:m]和 R1[m+1:r]归并到 R2[l:r]
int i=l, j=m+1, k=l;
while((i<=m)&&(j<=r))
    if(R1[i]<=R1[j])
        R2[k++]=R1[i++];
    else R2[k++]=R1[j++];
    if(j>m)
    for(int q=j;q<=r;q++)
        R2[k++]=R1[q];
    else for(int q=i;q<=r;q++)
        R2[k++]=R1[q];
}
void mergesort(Type R[], int left,int right)
{
if(left<right)
{
    int i=(left+right)/2;
    mergesort(R, left, i);
    mergesort(R, i+1, right);
    int b[M]; //M是通过 JHJdefineM宏定义的
    merge(R, b, left, i, right); //合并到数组 b
    copy(R, b, left, right); //复制回数组 R
    cout<<endl;
}
}
```

在上述算法中,merge合并 2个排好序的数组段到 1个新的数组 b中,然后由 copy将合并后的数组段再复制回数组 a中.merge和 copy所做的工作,可以在 $O(n)$的时间内完成,因此归并排序算法对 n个待排序记录进行排序,在最坏的情况下所需的计算时间 $T(n)$满足:

$$T(n)=\begin{cases}O(1), & n\leqslant 1\\ 2T(n/2)+O(n), & \text{otherwise}\end{cases} \tag{1}$$

可以将等式(1)等价转化成:

$$T(n)=\begin{cases}c, & n\leqslant 1\\ 2T(n\backslash 2)+cn, & \text{otherwise}\end{cases} \tag{2}$$

其中 c为一常数.由等式(2)可以解出

$$T(n)=O(n\log n). \tag{3}$$

1.5 快速排序

快速排序的基本思想是[2,5]:它是对冒泡排序的一种改进,通过 1次排序将待排序记录分割成独立的 2部分,其中一部分记录的关键字均比另一部分记录的关键字小,则可分别对这 2部分记录继续进行排序,以达到整个序列有序.假设当前待排序序列为 R[low,…, high],排序过程分为分解、求解和合并 3个步骤:①分解.在 R[low,…, high]中任选一个记录作为基准(base),以此基准将当前待排序序列分为左、右 2个子区间,前者为 R[low,…, base-1],均小于基准,后者为 R[base+1,…, high],均大于基准,基准则处于正确的位置上;②求解.通过递归调用快速排序对左、右 2个子区间进行快速排序;③合并.当求解中的 2个递归调用结束时,左、右 2个子区间已有序,即完成排序.附加存储空间为 $O(\log n)$.

快速排序用 C语言描述如下:

```
template<class Type>
int paritition(Type R[], int p, int r)
```

```
{
int i = p, j = r + 1;
Type x = R[p];
while(1)
{
    while(R[++i] < x && i < r);
    while(R[--j] > x);
    if(i >= j)
        break;
    Type temp;
    temp = R[i];
    R[i] = R[j];
    R[j] = temp;
}
R[p] = R[j];
R[j] = x;
return j;
}
void quicksort(Type R[], int p, int r)
{
if(p < r)
{
    int q = partition(R, p, r);    //划分两部分
    quicksort(R, p, q-1);    //递归调用
    quicksort(R, q+1, r);
}
}
```

根据上述算法的描述,可以分 2种情况讨论:

1)当待排序序列有序(序列按正序排列)时,每次划分只得到 1个比上一次少 1个记录的子序列.这样,必须经过 $n-1$次才能把所有记录定位,而且第 次需要 $n-$ 次比较才能找到第 i 个记录的正确位置,故总的比较次数达到

$$\sum_{i=1}^{n-1}(n-i)=\frac{1}{2}n(n-1)=O(n^2).$$

2)当排序序列是随机序列时,且 $T(n)$是对 n个记录的序列进行排序所需的时间,每次对 1个记录正确定位后,正好把序列划分为长度相等的 2个子序列,此时 $T(n)$满足

$$T(n)=\begin{cases}O(1), & n\leqslant 1\\ 2T(n/2)+O(n), & \text{otherwise}\end{cases}\quad(4)$$

等式(4)可以推出其时间复杂度为 $O(n\log n)$,其推理过程与等式(1)~(3)相同.

2 性能评价

2.1 实验与结果

为了比较各种排序算法的性能,我们使用一台桌面电脑(AMD Sempron(tm) Dual Core Processer2100, 1.81 GHz,2.87 GB的内存, Windows XP系统),在 VS6.0环境下,用 C语言编写程序,调用随机函数、时间函数来统计输入规模不同和不同的排序算法的用时情况.该程序的主要功能为:产生的序列为 1到 100之间的随机序列;采用数组的方式存储随机序列;根据不同的输入规模,实现 5种排序算法;能够根据不同的输入规模,计算出各种算法的用时.

当输入由随机函数产生,规模为 1 000、2 000、4 000、8 000、10 000、20 000、40 000、80 000时,5种排序算法的用时情况如下表 1所示,其对应的耗时曲线如图 1所示.

表 1 不同规模下 5种排序算法的耗时表

规模\个	排序算法耗时\s				
	冒泡	选择	插入	归并	快速
1 000	0.015	0.000	0.016	0.000	0.000
2 000	0.016	0.032	0.015	0.000	0.000
4 000	0.063	0.109	0.062	0.015	0.000
8 000	0.203	0.469	0.219	0.031	0.000
10 000	0.328	0.734	0.266	0.046	0.000
20 000	1.281	2.922	1.265	0.265	0.015
40 000	4.765	11.734	5.843	1.515	0.016
80 000	18.188	47.203	19.407	6.859	0.031

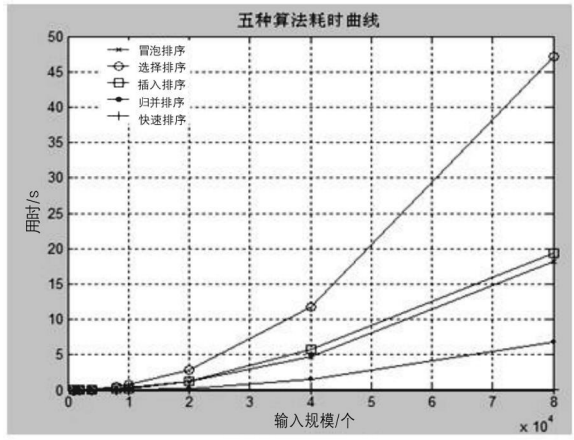

图 1 不同规模下 5种排序算法的耗时曲线

由表 1和图 1可知:当输入规模为 10 000时,5种排序算法的用时情况差不多,但随着输入规模的增大,快速排序算法的优势就体现出来,其次是归并排序,最差的是选择排序,插入排序略优于冒泡排序.

当输入是正序，且规模依次为 1 000、2 000、4 000、8 000、10 000时，5种排序算法的用时情况如表 2所示，其对应的耗时曲线如图 2所示.

表 2 正序输入时 5种排序算法的耗时表

规模\个	排序算法耗时\s				
	冒泡	选择	插入	归并	快速
1 000	0.000	0.000	0.000	0.000	0.000
2 000	0.016	0.032	0.000	0.000	0.015
4 000	0.047	0.110	0.000	0.015	0.063
8 000	0.284	0.469	0.000	0.031	0.219
10 000	0.328	0.734	0.000	0.046	0.328

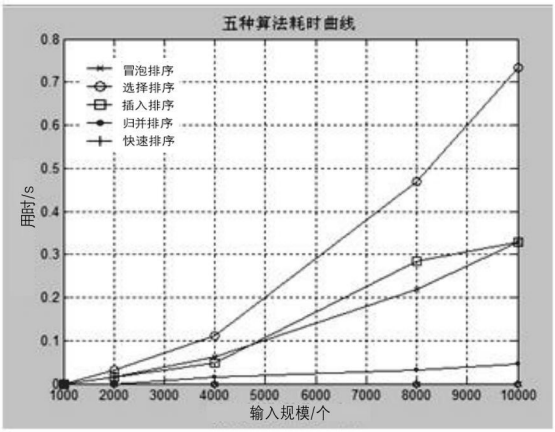

图 2 正序输入时 5种排序算法的耗时曲线

由表 2和图 2可知：当输入序列是正序时，插入排序算法最佳，其次是归并排序，快速排序略优于冒泡排序，最差的是选择排序.

当输入是逆序，且规模依次为 1 000、2 000、4 000、8 000、10 000时，5种排序算法的用时情况如表 3，其对应的耗时曲线如图 3所示.

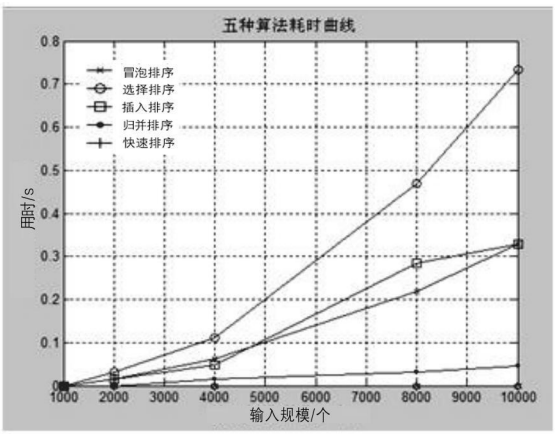

图 3 逆序输入时 5种排序算法的耗时曲线

由表 3和图 8可知：当输入序列是逆序时，归并排序算法是最理想的，最坏的是选择排序；输入规模在 8 000个记录范围内时，插入排序和快速排序差不多，随着输入规模的增大，插入排序比快速排序耗时更少；输入规模在 4 000个记录范围内时，冒泡排序和选择排序几乎一样，差别极其微小，随着规模增大，冒泡排序优于选择排序.

表 3 逆序输入时 5种排序算法的耗时表

规模\个	排序算法耗时\s				
	冒泡	选择	插入	归并	快速
1 000	0.000	0.016	0.000	0.000	0.000
2 000	0.032	0.031	0.031	0.000	0.016
4 000	0.125	0.125	0.062	0.015	0.047
8 000	0.437	0.485	0.281	0.015	0.283
10 000	0.703	0.766	0.438	0.031	0.328

2.2 算法性能评价

评价排序算法好坏的标准主要有 2条[6]：执行时间和所需的辅助空间. 算法本身的复杂程度，即空间复杂度和时间复杂度. 如果所需的辅助空间并不依赖于问题的规模，即辅助空间是 $O(1)$，称之为就地排序；非就地排序一般要求的辅助空间为 $O(n)$. 然而，时间复杂度取决于算法本身涉及的记录之间的比较次数和交换次数.

假设输入规模为 N，其中有序（即元素在序列正确的位置上）元素个数为 n，有序因子为 K，K规定了一个随机序列有序的程度，定义为 $K=\frac{n}{N}$，$K\in[0,1]$. 比较次数用 KCN表示，移动的次数用 RCN表示，时间复杂度用 $T(n)$表示，空间复杂度用 $S(n)$表示. 由此可以认为 K的值越接近 1，移动的次数 RCN就越少，进而所需 $T(n)$就会越少.

综合比较上述讨论的几种内部排序算法，可得到如表 4所示的结论.

根据表 4，可以将 5种排序算法按照平均时间分为 2类：冒泡排序、选择排序、插入排序其时间复杂度为 $O(n^2)$，即平方阶排序；归并排序、快速排序其时间复杂度为 $O(n\log n)$，即线性对数阶排序. 从平均时间性能而言，快速排序和归并排序优越于其他 3种排序，所需时间最省，但在最坏情况下，快速排序则不如归并排序. 在输

入规模较大时，快速排序比较有优势，但当输入规模不是很大时，5种排序算法的耗时相差无几.就空间复杂度而言，快速和归并排序的辅助空间开销比较大，冒泡、选择、插入排序辅助空间开销比较少.

表 4　5种排序算法的性能比较

排序方法	平均时间	最坏情况	辅助空间
冒泡	$O(n^2)$	$O(n^2)$	$O(1)$
选择	$O(n^2)$	$O(n^2)$	$O(1)$
插入	$O(n^2)$	$O(n^2)$	$O(1)$
归并	$O(n\log n)$	$O(n\log n)$	$O(n)$
快速	$O(n\log n)$	$O(n^2)$	$O(\log n)$

3　结语

在选择合适的排序算法时，应该考虑到算法本身的时间复杂度、空间复杂度等.其具体的用法要根据应用环境而定，侧重点不同，则选择的排序方法也各异.根据以上实验结果，得出结论：若记录较小，可以选择插入排序或者选择排序；若记录基本有序，则应考虑插入排序或冒泡排序；若记录较大，则宜采用快速排序或者归并排序；从辅助空间来看，归并排序需要的辅助空间最大，快速排序次之，其它 3种只需要 $O(1)$.

本文仅从随机序列、正序序列、逆序序列 3种情况进行了讨论，往往排序算法在具体应用中考虑的因素会更多、更复杂，因此需要更进一步地去研究排序算法的性能，这有利于更好地将其运用于实际的环境.

[参考文献]

[1] 严蔚敏，吴伟民. 数据结构[M]. 北京：清华大学出版社，1997：263－289.

[2] 王晓东. 计算机算法设计与分析[M]. 北京：电子工业出版社，2007：21－25.

[3] 曹衍龙，林瑞仲，徐慧. C语言实例解析[M]. 北京：人民邮电出版社，2007：94－117.

[4] 王颖，李肯立，李浪，等. 纵横多路并行归并算法[J]. 计算机研究与发展，2006，43(12)：2180－2186.

[5] 周建钦. 超快速排序算法[J]. 计算机工程与应用，2006，29(86)：41－42.

[6] 张建伟，张保威，郭云飞. 一类分批排序问题的复杂性分析及近似算法[J]. 计算机工程与应用，2007，43(3)：175－178.

Analysis on the performances of five sort algorithms

GAN Yan，YANG You

(School of Information Science and Engineering，Chongqing Normal University，Shapingba Chongqing 400047，China)

Abstract：Sorting algorithm is one of the most basic research fields in computer science. It's goal is to make record easier to search，insert and delete. Through the description of five sort algorithms：bubble，select，insert，merger and quick，the time and space complexity was summarized. Furthermore，two categories of $O(n^2)$ and $O(n\log n)$ could be divided. On the record sequence of random，positive and reverse，the application rules was pointed out based on the experiments. When the size of records is small，insertion sort or selection sort performs well. When the sequence is ordered，insertion sort or bubble sort performs well. When the size of records is large，quick sort or merge sort performs well.

Key words：sort algorithm；bubble sort；select sort；insert sort；merger sort；quick sort

(责任编辑　吴朝平)