改进的快速排序算法

Improved Fast Sorting Algorithm

杨锋英[1,2] 刘会超[2]
Yang Fengying Liu Huichao
(1.武汉大学软件工程国家重点实验室，湖北 武汉 430072；2.黄淮学院计算机科学系，河南 驻马店 463000)
(1.Wuhan University State Key Lab of Software Engineering, Hubei Wuhan 430072; 2.Huanghuai University Department of Computer Science, Henan Zhumadian 463000)

摘 要 本文通过分析快速排序算法中固有的不足之处，提出了改进的快速排序算法，并对算法的时间复杂度进行分析，通过编写程序上机实验，将原算法与改进的算法运行所需时间进行比较，证明了改进算法的有效性。

关键词 排序;算法;快速排序;插入排序;时间复杂度
中图分类号 :TP301 文献标识码 :A 文章编号 :1671-4792-(2010)1-0012-03

Abstract:This paper analyzes the inherent shortcomings of the quick sort algorithm, puts forward a improved quick sort algorithm, and analysis the algorithm's time complexity, through the experiments, compared the time that the original algorithm and the improved algorithm required to run, proved the effectiveness of improved algorithm.

Keywords:Sort; Algorithm; Fast Sorting Algorithm; Insert Sorting Algorithm;Time Complexity

0 引言

快速排序算法不仅是计算机数据处理中经常被采用且行之有效的算法之一，而且在其它算法研究中也常常用来作为参考和比较的对象。其原因是快速排序算法的二分有效性作为定理经过了严格的证明。即使这样，快速排序算法也有致命的缺点，对其缺点，文中提出了相应对策。

1 问题的提出

首先，快速排序算法是基于划分—递归求解—合并（也就是 Divide-Conquer-Combine）方法的算法之一。对于该方法中的划分部分，主要是采用重排数列的形式实现的，即将数列中的数，随机地选取任一元素作为支点元素，将数列中其它小于等于支点的元素放于支点的右边，大于支点的元素置于支点的左边，然后将支点位置返回给主调程序[7]。而对于划分的程序实现部分，大部分文献采用双重循环的形式实现的[1][4][6]，这样，完全掩盖问题的线性本质，使算法“快”的理念受到影响。

其次，在对部分有序或者区域相同数据处理方面。从对现实生活中员工工资、企业利润等实际数据观察可知，数据都在一定范围内进行变化，但变化有一定的限度，并围绕该限度上下波动，因此，通常在一个数列中会有部分数据是有序或者一样的。对于一般数据库中的数据而言，小区域内相同数据的存在是一种必然现象。快速排序在经过多次分解后，形成多个小规模数据区，这是最有可能产生这种小规模数据区域数据相同或者数据已经有序的情况，但快速排序在处理相同数据或者有序数据排序时，每一次划分都是基于“单枝”树的无效划分，从而充分显示了“慢”的一面[2][5]，这也是快速排序算法固有的、致命的缺点。

2 改进算法

对于上面提出的双重循环掩盖算法的线性本质问题，解决的办法是：对一组数据，首先假设首元素为支点元素，记下支点元素后的元素位置为新的首元素位置，同时记下给定的尾元素位置，接下来采用 While 语句通过从新的首元素“走一趟”的方法，边“走”边判断，如果新的首元素恰好小于等于支点元素，则新的首位置向后“走一步”，否则对尾元素进行判断，如果尾元素正好大于支点元素，则尾位置向前“走一步”，如果这两个条件都不成立，则交换这两个位置的元素，

然后首尾位置分别向后和向前“走一步”，重复 while，直到首位置在尾位置后面结束。最后将支点由假设位置调整到划分位置。通过这样一个过程，最终形成三分序列的格局，即小于等于支点部分、支点、大于支点部分。采用的单重循环改进划分算法如图一所示。

```
partition(a,p,r)
{
    x=a[p];
    i=p+1;
    j=r;
    while(i<=j)
    {
        if(a[i]<=x)  i++;
        else if(a[j]>x) j--;
        else
        {
            t=a[i];a[i]=a[j];a[j]=t;
            i++; j--;
        }
    }
    a[p]=a[j];
    a[j]=x;
    return j;
}
```

①

图一　单重循环改进划分算法

本算法中主要是利用 if 语句替换第二个 while 语句，仍采用两端向中间走的方式。当一端走不通时，改为另一端继续向中间走，只有两端都走不通时，才进行交换。通过 if 语句的替换，使得算法的线性本质一目了然。

对于快速排序在处理小批量相同数据排序的“单枝”树问题，可以采用与其它算法相结合的办法。在排序算法中，插入排序在处理小批量数据排序上，时间复杂度接近于 n，因此，改进的算法是将快速排序与插入排序结合起来。

改进的快速排序算法思想是：对于给定的 n (n>=10000) 个数据序列，先调用单重循不改进划分算法中划分策略进行若干区域的划分，若某个区域数据个数小于等于 100 个时，则停止划分，对该区域进行插入排序。在插入排序中，又采用设立标志的方法对已经有序或者是相同数据排序进行判断，若在一遍扫描过程中，没有一次数据交换发生，则表明该数列已经有序，排序结束。

改进后的算法如图二所示。

3 时间复杂度

设存在 n 个元素的数据序列，每个元素被选中作为支点

```
quicksort(a,p,q)
{
    if(p-q>100)
    {
        r=partition(a,p,q);
        quicksort(a,p,r-1);
        quicksort(a,r+1,q);
    }
    else
        insersort(a,p,q);
}
insertsort(a,u,v)
{
    for(i=u;i<v;i++)
    {
        if(a[i+1]<a[i])
        {
            t=a[i+1];   s=i
            while (t < a[s]) &&(s >u-1)
            {
                a[s+1]=a[s];
                s:=s-1;
            }
            a[s+1]:=t;
        }
    }
}
```

②

图二　改进后的算法

元素的概率相等，即每个元素被选中作为支点元素的概率均为 $\frac{1}{n}$，则改进后快速排序算法的时间复杂度为执行 n 个语句一次的时间，设为 n，m 为划分的块数，k 为每一块进行插入排序所需时间（注：此处每一块中必存在大量相同数据）。那么，时间复杂度的计算如公式(1)所示。

$$T(n)=n+\frac{1}{n}\sum_{i=101}^{n}[T(i-1)+T(n-i)]+mk\le n+\frac{1}{n}\sum_{i=1}^{n}[T(i-1)+T(n-i)] \quad (1)$$

由式 (1) 分析可知，在一定区域内存在批量重复数据时，改进算法的时间复杂度明显优于其它算法，最好时能够接近 O(n)。

4 实验结果

表一是对快速排序和改进后的快速排序在排序时间进行的比较。所对比的快速排序算法是采用文献[1]中算法。表中的数据产生方式若为随机产生表示该数列是由 C++ 语言的 rand()函数产生，若为受限则表示该数据是人为加以控制而产生的区域性 50 个相同数据的数列。实验是在 P41.8G、512M 内存的机器上运行的，从多次的试验结果可以看出，若数据是小区域性分散相同，则算法明显优于原快速排序算法。

5 结束语

多次不同区域相同数据的实验表明，改进的快速排序算

表一 快速排序算法改进前后的时间比较

算法	数据个数	数据产生方式	排序时间（ms）
快速排序	10000	随机	11
	10000	受限	16
	100000	随机	27
	100000	受限	93
改进的快速排序	10000	随机	19
	10000	受限	6
	100000	随机	54
	100000	受限	41

法不仅具备原有算法在处理数据二分法的优越性，并且采用单重循环来缩短二分处理内时间花费，展示出算法的线性本质，同时充分运用插入排序在处理小规模相同数据排序的优越性。因此，改进的快速排序算法是正确、有效的。

参考文献

[1]严蔚民，吴伟民.数据结构[M].北京：清华大学出版社，2002：263-292.

[2]Thomas H.Cormen,Charles E.Leiserson,Clifford Stein.Introduction to Algorithms [M].The MIT Press, 2001：145-182.

[3]官章全.标准C++库大全[M].北京：电子工业出版社，2002.

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

[5]王红梅，应红霞，季绍红.递归函数时间复杂度的分析[J].东北师范大学学报(自然科学版)，2001,(4)：111-113.

[6]肖奎，吴天吉.快速排序算法的改进[J].福建电脑，2008,(8)：98-113.

[7]Anany Levitin.The Design & Analysis of Algorithms[M].TSINGHUA UNIVERSITY PRESS,2007,(11)：129-135.