基于分布计数的基数排序方法的研究

葛 浩[1,2],杨传健[1,3]
(1.滁州学院 电子信息工程系,安徽 滁州 239012;
2.安徽大学 计算机学院,安徽 合肥 230039;
3.合肥工业大学 计算机与信息学院,安徽 合肥 230009)

摘 要:排序是计算机科学中一个非常重要的问题。提出了一种基于分布计数的基数排序方法,给出该算法定义、算法描述、算法正确性证明和算法分析;讨论了基于该排序算法几个关键问题的解决方法。算法理论分析和实验结果研究均表明该算法时间复杂度为 $O(N)$,速度优于快速排序,是一种高效的排序方法。

关键词:排序;计数;基数排序;时间复杂度
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2008)02-0122-04

Research on Radix Sort Based on Distributing Counting Sort Method

GE Hao[1,2], YANG Chuan-jian[1,3]
(1.Department of Electronic and Information Engineering, Chuzhou University, Chuzhou 239012, China;
2.School of Computer Science, Anhui University, Hefei 230039, China;
3.School of Computer and Information, Hefei University of Technology, Hefei 230009, China)

Abstract: The sorting is one of the most important problems in computer science. In this paper, radix sort using distributing counting sort method is put forward. Its algorithmic definition, algorithmic ideas, the correctness of the algorithm and algorithmic analysis are given. Solution methods of key problem for the algorithm are discussed. Its theoretical analysis and experimental result show that its time complexity is $O(N)$, and it is better than quick sort. It is efficient sorting method.

Key words: sort; count; radix sort; time complexity

0 引 言

排序(Sorting)[1,2]是计算机科学中一个非常重要的问题,它是将一个数据元素序列,按关键字(或称排序码)排列成一个有序的序列,可如下定义:

定义1:设 $R=\{R_i \mid 其中, i=1\cdots N\}$ 为含 N 个数据元素的有限集合,其对应的关键字集合为 $K=\{K_i \mid 其中, i=1\cdots N\}$,排序操作是对 R 中的元素调整,重新获得一个集合 $R'=\{R_{\pi(i)}\}$,使其满足:

(1)$\pi=\{\pi(i) \mid 1\leqslant\pi(i)\leqslant N, i=1\cdots N\}$;

(2)$K_{\pi(j)}\leqslant K_{\pi(j+1)}$ 或 $K_{\pi(j)}\geqslant K_{\pi(j+1)}$

其中,$j=1\cdots N$。

称 R' 是按关键字的一个递增或非递增序列。

目前常见的排序算法有两种分类:一类是基于比较的排序,如插入排序、归并排序、堆排序[3]、快速排序等,通过关键字比较和移动数据实现排序,这些算法的时间复杂度下界是 $\Omega(N\log_2 N)$。1962年Hoare提出的快速排序算法的平均性能是 $\Omega(N\log_2 N)$,但最坏情况下的运行时间却为 $\Theta(n^2)$;堆排序和归并排序在最坏情况时间复杂度是 $O(N\log_2 N)$。可以证明,基于比较的排序算法在最坏情况下能达到的最好时间复杂度为 $O(N\log_2 N)$[1,4]。另一类是分布式排序,为一种非比较的排序方法,根据关键字分布情况,以一定的辅助空间为代价,时间复杂度可达 $O(N)$。

文中提出一种非比较排序算法,该方法将计数方法与基数排序[5,6]相结合,称之为基于分布计数的基数排序(简称CRSORT),该算法时间复杂度为 $O(N)$,明显优于插入排序、选择排序,也优于快速排序、分段快速排序法[7,8]。

收稿日期:2007-05-10
基金项目:安徽高校省级自然科学研究项目(KJ2007B237)
作者简介:葛 浩(1976-),男,安徽滁州人,讲师,硕士研究生,研究方向为人工智能、数据挖掘。

1 算法描述

1.1 分布计数排序[9](简称计数排序)

定义2:设数据元素 R_i 的关键字 K_i 是分布在闭区

间[0,d']的整数(其中$0 \leqslant i \leqslant N, d' = d - 1$),构造一个计数表 COUNT[0],COUNT[1],…,COUNT[d'],对任意关键字 K_i,COUNT[i]用于记录比它小的数据个数;根据 COUNT 表可以把每个数据直接放到输出数组的最终位置上。

为了实现算法,要使用三个辅助数组 R、B 和 COUNT,R、B 用于存放被排序数据,其容量为 N,COUNT 作为计数表和存放被排数据排序后最终的位置,其容量为 d。以无符号整数递增排序为例,算法 1(CSORT)描述如下:

CSORT1:[给 i 设置初值]$i \leftarrow 0$

CSORT2:[将每个计数表中的元素设置初值 0]将 COUNT[i]$\leftarrow 0$

CSORT3:[修改 i]$i \leftarrow i + 1$

CSORT4:[对 i 进行循环]若 $i < d$,则转 CSORT2

CSORT5:[给 r 设置初值]$r \leftarrow 1$

CSORT6:[根据 Kr 给相应的计数表元素计数]COUNT[Kr]增 1

CSORT7:[修改 r]$r \leftarrow r + 1$

CSORT8:[对 r 进行循环]若 $r \leqslant N$,则转 CSORT6

CSORT9:[给 i 设置初值]$i \leftarrow 1$

CSORT10:[计算被排序数据的位置]COUNT[i]$\leftarrow$ COUNT[i] + COUNT[$i - 1$]

CSORT11:[修改 i]$i \leftarrow i + 1$

CSORT12:[对 i 进行循环]若 $i < d$,则转 CSORT10

CSORT13:[给 r 设置初值]$r \leftarrow N$

CSORT14:[将被排序的数组输出到输出数组中]B[COUNT[$R[r]$]]$\leftarrow R[r]$

CSORT15:[修改计数表中的元素的值]COUNT[$R[r]$]$\leftarrow$ COUNT[$R[r]$] - 1

CSORT16:[修改 r]$r \leftarrow r - 1$

CSORT17:[对 r 进行循环]若 $r \geqslant 1$,则转 CSORT14

CSORT18:[算法结束]结束。

分析算法可知 CSORT1 ~ CSORT4 循环花费的时间为 $O(d)$,CSORT5 ~ CSORT8 循环花费的时间为 $O(N)$,CSORT9 ~ CSORT12 循环花费的时间为 $O(d)$,CSORT13 ~ CSORT16 循环花费的时间为 $O(N)$,可以得到总时间的开销为 $O(d + N)$。当 $d = O(N)$ 时,算法的时间复杂度为 $O(N)$,另外算法 CSORT 需要两个辅助空间 COUNT 和 B,容量分别为 d 和 N,所以空间复杂度为 $O(d + N)$。如果 $d >> N^2$,算法的时间开销不再是 $O(N)$,而是 $\geqslant N^2$,COUNT 附加内存空间也大大增加,同时也会因 COUNT 表中有很多存储单元不被使用,而造成很大的空间浪费。为了避免时间复杂度的和辅助空间增加,以及大量的空间浪费,提出一种基于计数的基数排序,即将计数排序和基数排序(Radix Sort)结合起来。

1.2　计数的基数排序

定义 3:设待排序的 N 个数据元素存放在数组 R 中,元素 $R[i]$ 的关键字为 K_i,K_i 可以看作是基于 m 位 d 进制数(d 可称为基数),即

$$K_i = (k_i^m, \cdots, k_i^j, \cdots, k_i^2, k_i^1),$$

其中,$0 \leqslant k_i^j \leqslant d - 1, 1 \leqslant j \leqslant m$。

对关键字采用最低位优先法(LSD)[1],由最低位向最高位依次采用计数排序法,进行 m 趟排序,此时得到的序列为有序序列。算法 2(CRSORT)描述如下:

CRSORT1:[给 j 设置初值]$j \leftarrow 1$

CRSORT2:[进行分布计数排序]对第 j 位进行一趟分布计数排序 CSORT

CRSORT3:[修改 j]$j \leftarrow j + 1$

CRSORT4:[对 j 进行循环]若 $j \leqslant m$,则转 CRSORT2

CRSORT5:[算法结束]结束。

1.3　算法正确性证明

下面给出算法 CRSORT 正确性的证明。

证明:采用归纳法证明。

初始情况:如果 $m = 1$,基于基数为 d 的关键字只有一位,因此直接采用计数排序算法 CSORT,可以获得有序序列。

归纳假设:假设已完成了对低 $m - 1$ 位数据的计数基数排序,证明计数基数排序对 m 位数据的排序是正确的。

计数基数排序是分别对每一位进行计数排序,并且从最低位(第一位)开始。因此,m 位的计数基数排序,可以描述为:先对低 $m - 1$ 位排序,然后对第 m 位排序。利用归纳假设,低 $m - 1$ 位已经完成排序,所以对 m 位数据排序,是基于 $m - 1$ 位有序的情况下。因此,对 m 位数据的排序,只要对第 m 位排序。设有两个数据的关键字分别为 $K_i = (k_i^m, k_i^{m-1}, \cdots, k_i^1)$ 和 $K_j = (k_j^m, k_j^{m-1}, \cdots, k_j^1)$,第 m 位数据为 k_i^m 和 k_j^m:

(1) 如果 $k_i^m < k_j^m$,不管低位的情况如何,总有 $K_i < K_j$;

(2) 如果 $k_i^m > k_j^m$,不管低位的情况如何,总有 $K_i > K_j$;

(3) 如果 $k_i^m = k_j^m$,k_i^m 和 k_j^m 处在相同的位置上,因为该排序的稳定性,当 $k_i^m = k_j^m$,数据 K_i 和 K_j 的位置

由低 $m-1$ 位决定。

因此,可以证明基于分布计数的基数排序是正确的。

2 算法分析和讨论

2.1 时间复杂度分析

在算法 CRSORT 进行了 m 趟的 CSORT 排序,CSORT 排序的时间开销为 $O(d+N)$。因此,CRSORT 花费的总时间为 $O(m(d+N))$。如果 m 为常数并且 $d=O(N)$ 时,CRSORT 排序运行时间为 $O(N)$。也就是说,计数的基数排序具有线性运行时间,优于堆排序、归并排序和快速排序。

2.2 空间复杂度分析

算法 CRSORT 中使用了两个附加数组 COUNT 和 B,所需的空间分别为 d 和 N,所以说,辅助存储空间开销为 $O(d+N)$。由于对关键字进行了分段,m 和 d 可根据实际情况取值,以保证所取的值为常数,使基数 d 的取值不会像单一的分布计数排序算法 CSORT 时取值过大,这样 COUNT 的辅助空间明显减小,避免空间过度浪费。

2.3 算法讨论

(1) 为了验证 CRSORT 算法的时间效率,与常用的插入排序和快速排序作比较。采用 VC++ 6.0 分别编写相应排序子程序,对随机生成的无符号整数,在 Petium4 2.8GMHz,RAM512M 微机上测试,结果见表 1。

表 1 比较插入排序、快速排序和基数为 2^8 的 CRSORT 排序(单位:秒)

数据量(单位:万)	Insert Sort	Quick Sort	CRSORT Base 2^8
5	0.2775	0.0133	0.0127
10	6.3870	0.0312	0.0238
30	256.3521	0.1170	0.0708
50	716.2193	0.2730	0.1098
100	2586.4060	0.4963	0.2188

从表 1 的比较结果表明,CRSORT 算法远远快于插入排序,也优于快速排序算法。并且随着排序数据的数量级的增加,其优越性就越明显。

(2) 对于数据元素关键字 $K_i=(k_i^m,k_i^{m-1},\cdots,k_i^1)$ 的分段,可以有多种方法。也就是说,基数 d 可根据实际情况取值。

可以将关键字以十进制数来划分,基数可以取 $10^1,10^2,10^3$ 等。常用的基数为 10,即分别对关键字的个、十、百、千…等位依次采用计数排序 CSORT。如果关键字的最大取值为 65535,那么只需要进行 5 趟的计数排序。如何提取出个、十、百、千、万位?可以采用整除的方法实现。

也可以将关键字以二进制位段来划分,基数可取 $2^1,2^4,2^8,2^{16}$ 等。例如,取 $2^8=256$ 作为基数,即排序是按不同位的字节作各趟 CSORT 排序。对不同字节的分段,可以采用位操作符:<<、>>和 &,提取出不同位的字节。许多语言(如,C++、JAVA)都支持这些操作。

```
inline int digit_8_bit(unsigned int Key, int k)
{return(KEY >> 8 * (k - 1) & x0ff);}
```

用 VC++6.0 编写这两种方案的排序子程序,对随机函数生成的无符号整数测试,结果见表 2。

表 2 比较基于不同基数的 CRSORT 排序(单位:秒)

数据量(单位:万)	Base 10	Base 2^1	Base 2^8	Base 2^{16}
5	0.2732	0.0465	0.0127	0.0107
10	0.5967	0.1017	0.0239	0.0152
50	2.9448	0.7423	0.1093	0.0933
100	5.8790	1.5220	0.2188	0.2245
500	29.4258	6.7188	1.0818	1.1170
1000	58.8283	13.1290	2.0585	2.4023

试验结果表明,基于二进制位的 CRSORT 排序优于十进制。因为,二进制充分利用了数据在计算机中存储的特点,大大简化了排序过程中对数据的处理。另外,位操作花费的时间远远小于整除操作,一次位操作所需时间约为除法操作所需时间的几十分之一,这样效率便体现出来了。

另外,表 2 中的结果也表明,根据待排序数据的最大值和数据的数量级,灵活地划分位段(如,按一个二进制位分段、单字节分段或双字节分段),也可以提高排序效率。

(3) 如果排序对象为有符号整数,只需在排序之前对正负数做个判断。即,进行一次扫描,将正数和负数分别放到不同的附加数组中,然后进行 CRSORT 排序。这里只是增加了一次对整个数据元素扫描操作,时间花费为 $O(N)$,但并不影响 CSORT 排序的时间复杂度的线性运行时间特性。

要补充说明的是,对正负数的判断不要直接用比较操作,应尽量采用位操作,可有效提高判断效率。例如,

```
if ( Key&0x80000000 ) { NegativeValues ++;...}//判断关键字 Key 是否为负数。
```

(4) CRSORT 方法一般是不能直接处理浮点数的。但可以根据 IEEE-754 浮点格式(见表 3),将一个浮点数据看成一个整型数据,然后按照对整数进行 CRSORT 方法实现排序。

表3　IEEE-754浮点数标准

	符号	指数	尾数
单精度	1[31]	8[30-23]	23[22-00]
双精度	1[63]	11[62-52]	52[51-00]

例如,单精度十进制浮点数 -20.59375 采用IEEE-754浮点格式的表示形式为 11000001 10100100 11000000 00000000 ,这种形式可以将其看作一个无符号整数的二进制存储形式,因此所有的负数都大于正数;然后采用基数为 2^8 的CRSORT算法进行递增排序,则将待排序数据调整成为两个部分:前部为负数非递增序列,后部为正数递增序列;最后只需在线性时间内将正数和负数的整体位置互换(负数在调整过程中,其内部数据要倒置),便可以得到正确的有序序列。

(5) 如果对字符串排序,可以利用字符串在计算机存储器的存储特点,将每个字符作为每趟计数排序的操作的对象。由于字符是以ASCII形式存放的,取值范围在闭区间[0,255],便取 2^8 作为基数,对其进行CRSORT排序即可。

3　结束语

文中提出的基于计数的基数排序算法的时间复杂度为 $O(N)$,排序的速度优于快速排序,是一种简单高效的算法。另外,如果待排序的为负整数、字符串以及浮点型等数据,只要对这些数据转换为一种适当的描述形式进行,便可采用这种排序算法,文中对此也作了一定的讨论。

参考文献:

[1] 严蔚敏,吴伟民. 数据结构(C语言)[M]. 北京:清华大学出版社, 1997.

[2] 殷人昆,陶永雷,谢若阳,等. 数据结构(用面向对象方法与C++描述)[M]. 北京:清华大学出版社, 1999.

[3] 唐开山. 二次堆排序算法和提高排序效率的途径[J]. 计算机工程与应用,1998,34(5):45-48.

[4] Cormen T H. Introduction to Algorithms[M]. [s.l.]: The MIT Press, 1990.

[5] Sedgewick R. Algorithms in C++[M]. [s.l.]: Addison-Wesley, 1998.

[6] Baase S. Computer Algorithm: Introduction to Design and Analyse[M]. [s.l.]: Addison-Wesley, 2000.

[7] 王向阳,杨红颖. 分段快速排序法的改进[J]. 小型微型计算机系统,2001,22(11):1382-1385.

[8] 唐向阳. 分段快速排序法[J]. 软件学报,1993,4(2):53-57.

[9] Knuth D E. The Art of Computer Programming 3/Sorting and Searching[M]. 管纪文译. 北京:国防工业出版社,1984.

(上接第121页)

表2　规则提取

含水量指标(χ_1)	高	高	高	中	中	中	低	低	低
温度指标(χ_2)	高	中	低	高	中	低	高	中	低
生长状况指标(y_1)	低	中	低	中	高	中	低	中	低

通过实际数据检查,以上规则是合理的,基本与实际情况一致。实验表明,应用本算法能够获得较为满意的结果。

4　结束语

基于模糊神经网络与遗传算法的数据挖掘方法,充分考虑了数据挖掘过程中数据量大、模糊性强、提取高效准确的推理规则等特点。应用本算法进行数据挖掘,预测精度高,它比单纯的遗传算法或神经网络或模糊系统的效果要好;适合模糊信息的处理,适合对大容量数据的挖掘等。下一步主要就该算法中遗传算法部分的适应值函数、遗传算子、自适应等方面进行深入研究,以提高算法的执行效率。

参考文献:

[1] 潘　笑,万　敏. 基于模糊神经网络的数据挖掘方法研究[J]. 微电子学与计算机,2005,22(12):48-50.

[2] 张　勇,黄金才,张维明,等. 一种遗传模糊神经网络数据挖掘算法[J]. 模糊系统与数学,2006,20(5):131-135.

[3] 钟　珞,饶文碧,邹承明. 人工神经网络及其融合应用技术[M]. 北京:科学出版社,2007.

[4] 周志坚,毛宗源. 一种基于遗传算法的模糊神经网络最优控制[J]. 控制理论与应用,2000,17(5):784-788.

[5] Ishigami H, Fukuda T, Shibata T, et al. Structure optimization of Fuzzy Neural Network by Genetic Algorithm[J]. Fuzzy Sets and Systems, 1995, 71(3):257-264.

[6] Takagi T, Sugeno M. Fuzzy indentification of systems and its applications to modeling and control[J]. IEEE Transactions on Systems, Man and Cybernetics, 1985, 15(1):116-132.

[7] 王小平,曹立明. 遗传算法——理论、应用与软件实现[M]. 西安:西安交通大学出版社,2002.

[8] 李　旸,高正光,李启炎. 基于神经网络与遗传算法的数据挖掘体系结构[J]. 计算机工程,2004,30(6):155-156.