DOI: 10.13656/j.cnki.gxkx.1997.02.006
广西科学 Guangxi Sciences 1997, 4 (2): 93~96

# 二维数组的快速排序算法
# Quick Sorting Algorithm of Two-dimensional Array

彭月英
Peng Yueying

(广西师范学院计算中心 南宁市明秀东路 530001)
(Computer Center, Guangxi Teachers College, East Mingxiulu, Nanning, Guangxi, 530001)

摘要 在一维数组快速排序算法的基础上,给出了二维数组的快速排序算法。理论分析和大量的数值实验结果表明,其算法的平均计算时间仍然是 $O(n\log_2 n)$,一般所需的栈空间仍为 $O(\log_2 n)$,是有效而可靠的快速排序算法。

关键词 二维数组 一维数组 快速排序 算法

**Abstract** On the basis of the quick sorting algorithm of one-dimensional array, the quick sorting algorithm of two-dimensional array was provided. Theoretical analysis and the results from a large number of experiments dealing with data showed that the average calculation time of the algorithm was $O(n\log_2 n)$, and generally the required stack space was also $O(\log_2 n)$, which was an effective and reliable quick sorting algorithm.

**Key words** two-dimensional array, one-dimensional array, quick sorting, algorithm
中图法分类号 TP311.12

水文、气象、工程设计、石油勘探、事务管理、文献组织等各个领域中,有成千上万的信息是用矩阵表示的。有的信息虽然是用一维序列表示,但是受某些计算机语言系统的最大下标的限制,也需要将这些信息转换成用二维数组存贮。当用计算机对这些庞大的信息进行排序处理时,就需要用到二维数组的排序算法,而目前的文献资料主要是介绍一维数组的排序算法,有关二维数组的全部元素的排序算法不多。

文献 [1] 介绍了二维数组排序的 $O(n^2)$ 算法,显然它对 $n$ 很大时难以用于实际;文献 [2] 的算法采用了快速分类与直插结合法,它是以其中各行向量的前 $f$ 个分量值为关键字进行排序,不是对整个矩阵元素的排序;文献 [3] 所介绍的两段聚合矩阵排序,是按矩阵的指定行和列的元素的升序或降序重新排列矩阵,显然该法也不是本文所讨论的整个矩阵元素的排序。它们都属于多队列排序。

因此,为了满足实际应用部门和科学研究的需要,很有必要寻求一种通用且快速,而又灵活的有效算法。在分析一维数组快速排序算法[4~11]的基础上,并对该算法作了适当的改进,得到了二维数组快速排序算法和程序。通过一定的理论分析和大量的实例验证,其算法完全能满足上述要求,是一种比较理想的二维数组快速排序算法。

1997-01-27收稿, 1997-03-23修回

## 1 算法原理

### 1.1 用一维数组快速排序法对二维数组进行排序

设所需排序的矩阵为记录集合 $\{A_{ij}\}$,由 $m$ 行,$n$ 列组成,则共有 $m\times n$ 个记录 (或元素):

$$A_{11}\quad A_{12}\quad \cdots\quad A_{1n}$$
$$A_{21}\quad A_{22}\quad \cdots\quad A_{2n}$$
$$\cdots\cdots$$
$$A_{m1}\quad A_{m2}\quad \cdots\quad A_{mn}$$

把这些记录用二维数组 A (M, N) 存贮,为了叙述的方便,这里行和列 (即下标 I和 J) 的最小值都取 1。二维数组的排序就是将整个矩阵元素按从小到大 (或从大到小) 的顺序排好序,然后将排序结果以行 (或列) 的顺序存放。把每一行的尾部与下面一行的首部相连,展开后所有的记录 $A_{ij}$ 都在一条带子上,当 $m\times n$ 的值小于所用语言系统下标的上限值

时,则可把二维数组转换成一维数组存贮,这样就可用一维数组的快速排序方法[4~10]对{$A_{ij}$}进行排序处理。其中文献[4~7]介绍了非递归的算法程序,文献[7~9]介绍了递归的算法程序。

### 1.2 二维数组快速排序

记录集合{$A_{ij}$}的左上角(称始端)的记录为$A_{11}$,右下角(称终端)的记录为$A_{mn}$。快速分类法的基本思想是[4]:如果按照关键字$K$的非递减(或递减)顺序对记录集合{$A_{ij}$}进行分类,则可先任选一个记录$A_{kl}$(如可选取第一个记录$A_{11}$)的关键字$K_{kl}$作为比较标准。从集合的始、终两端依次取记录的关键字$K_{pq}$与$K_{kl}$比较,若$K_{pq} < K_{kl}$(或$K_{pq} > K_{kl}$),则置$A_{pq}$于$A_{kl}$的左上方。若$K_{pq} > K_{kl}$(或$K_{pq} < K_{kl}$),则置$A_{pq}$于$A_{kl}$的右下方。这样对记录比较定位的结果,可使原记录集合分成左上方和右下方两个子集合。这两个子集合由记录$A_{kl}$分开。称以上步骤为对记录集合进行了一次分段(或分划),然后再分别对左上方和右下方两个子集合进行分段。循此下去,就可得到原记录集合按关键字的非递减(或递减)顺序的分类排列,即从小到大(或从大到小)的排序。

然而应当注意到,由于快速分类方案是不断地对集合进行分段,又不断地分左上方和右下方两部分进行处理的,所以就需要在存贮空间上开辟一定数量的栈区,用于存放尚待处理的子集合的下标地址有关信息,以便后面处理时用。

从以上的叙述可看出,二维数组与一维数组快速排序的原理基本相同,但是也有下列不同之处:由于受存贮空间的限制,所以在进行分段时,一维数组的左指针是从集合的最左端向右移动,右指针是从集合的最右端向左移动,它们是沿着直线方向相向移动,最后两指针重合为止[10],每个指针移动的座标是一维的;而二维数组的始端指针是从集合的左上角以行列为顺序向右下方移动,终端指针是从集合的右下角以行列为顺序向左上方移动,它们是沿着直折线方向相向移动,最后两指针重合为止,每个指针移动的座标是二维的,即有行列座标,并称行的座标为行指针座标,列的座标为列指针座标。

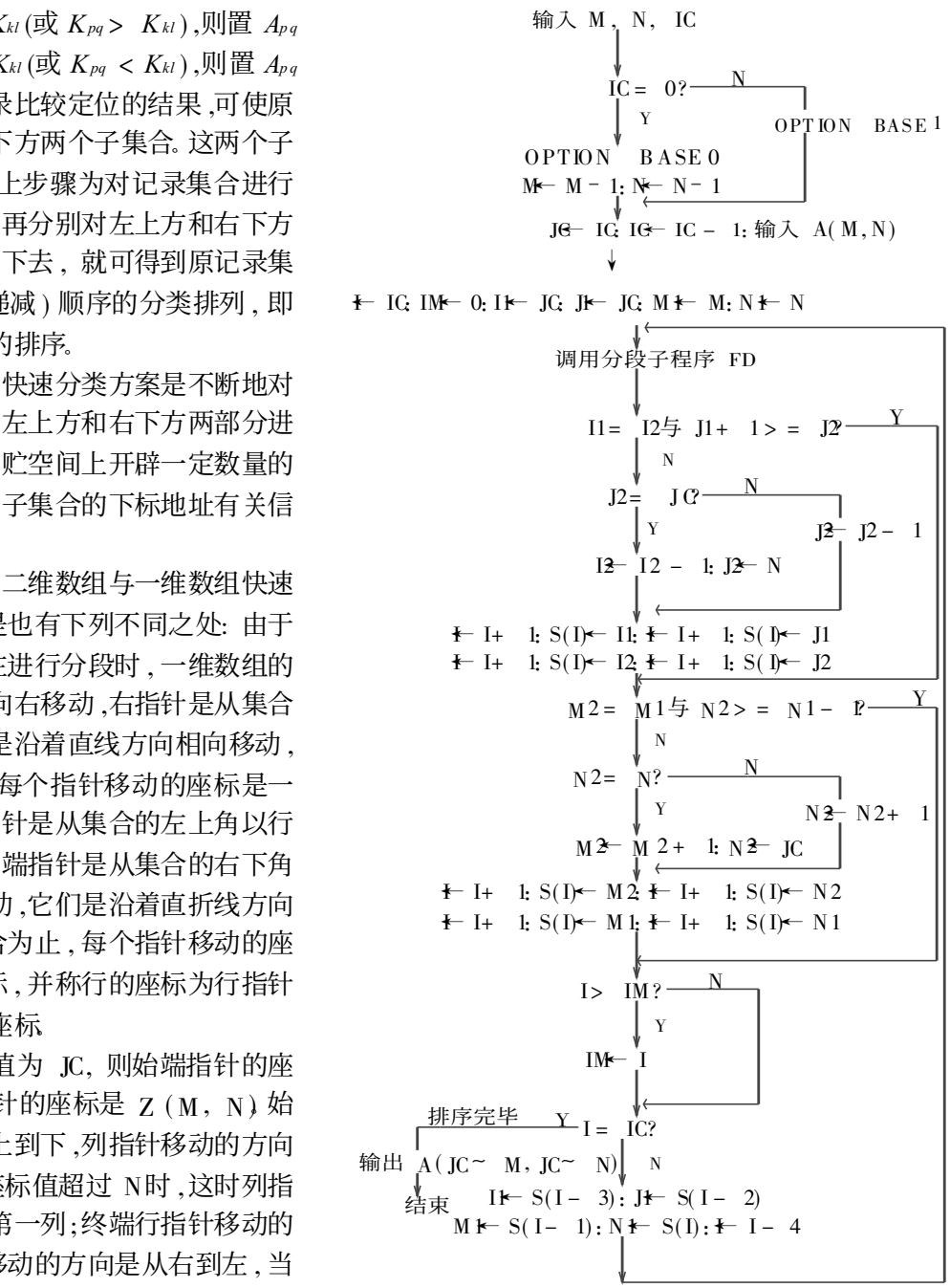
设行和列下标的最小值为JC,则始端指针的座标为S(JC,JC),终端指针的座标是Z(M,N)。始端行指针移动的方向是从上到下,列指针移动的方向是从左到右,当列指针的座标值超过N时,这时列指针的座标移到下面一行的第一列;终端行指针移动的方向是从下到上,列指针移动的方向是从右到左,当列指针的座标值小于JC时,这时列指针的座标移到上面一行的最后一列。所以不管是始(或终)端指针,走完一行以后都要换行。

上述讨论的方法其指针移动的顺序是先行后列,也可以按先列后行的顺序移动来展开讨论。

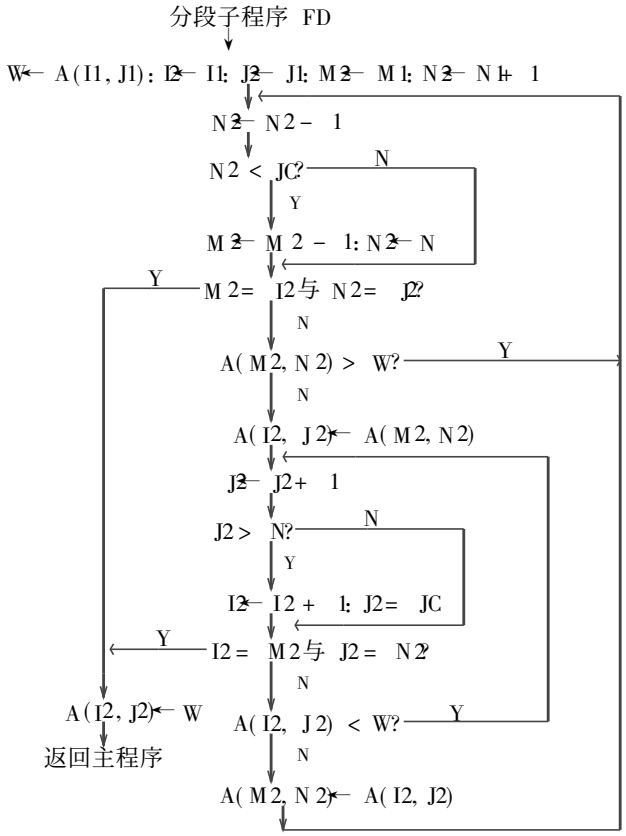
## 2 程序设计

在文献[4~6]的一维数组快速排序算法程序设计的程序流程图和BASIC语言程序基础上根据以上分析设计出二维数组快速排序算法的程序流程图如下(非递归的从小到大排序):

(1) 主程序流程图



(2) 分段子程序流程图




以上的子程序 FD用于实现对记录集合的分段,用主程序实现进栈、退栈、记下 $S$栈的最大个数和输出排序结果。根据上述程序流程图,不难应用所熟悉的算法语言写出其程序,各变量的意义和存贮说明如下:

M, N—— 待排序矩阵中的行、列数;

IC, JC—— 行和列下标的最小值,取 0或 1;

A (M, N) —— 存贮待排序矩阵中的各元素用;

S(I)—— 存放分段后子集合中第一个和最后一个记录行和列的下标所使用的栈;

I—— S栈的指针;

W—— 存放每次分段时用的界元素的变量;

I1, J1—— 集合中最上端 (第一个) 元素行和列的下标;

M1, N1—— 集合中最下端 (最后一个) 元素行和列的下标;

I2, J2—— 始端 (左上) 行和列指针单元;

M2, N2—— 终端 (右下) 行和列指针单元;

IM—— S栈的指针的最大值。

如果把与界元素 W比较的两个关系式中的关系符互换,那么就可得到二维数组从大到小的快速排序算法。

## 3 数值实验结果分析

### 3.1 算法的时间复杂性和所需的栈空间数

在所排序列最有利排列的情况下,快速排序法对 $n$个元素的排序时间复杂性为 $O(n\log_2 n)$。然而如果给定的序列几乎是已排好序的,那么,该算法效率最差,其大约需要 $n^2/2$次比较。根据计算论证[5, 8-10],快速排序法的平均比较次数为 $Cn\ln n$ ($C$为常数),它很接近于 $n\ln n$次。快速排序的平均计算时间是 $O(n\log_2 n)$。而且,实验结果证明,对平均计算时间来讲,它是各种内部排序法中最好的一种。

在插入排序中仅需存放一个记录的附加空间,而快速排序则不同,它需要用栈空间来实现递归。如果集合按相等方式分裂,则最大递归深度为 $\log_2 n$,需要的栈空间为 $O(\log_2 n)$。最坏的情况是,在递归的每一级上,集合分裂成长度 $n-1$的左子集合和长度为零的右子集合 (或者左子集合为零,右子集合长度为 $n-1$)。在这种情况下,递归深度变为 $n$,需要的栈空间为 $O(n)$。如果知道长度为 2的右子集合不必进栈,那么最坏的情况栈空间便可压缩至四分之一。如果对较小的子集合先排序,则节省一个数量级的栈空间也能够

表 1 二维与一维数组快速排序成果

**Table 1 Result of quick sorting of two-dimensional array and one-dimensional array**

| 排序数据个数 Data | 二维数组快速排序 Quick sort of two-dimensional array | | 一维数组快速排序 Quick sort of one-dimensional array | | 一维数组希尔排序 Shell sort of one-dimensional array |
|---|---|---|---|---|---|
| | 计算时间 Computing time (s) | 栈空间数 Stack space number | 计算时间 Computing time (s) | 栈空间数 Stack space number | 计算时间 Computing time (s) |
| 4000 | 1 | 56 | 1 | 28 | 1 |
| 5000 | 2 | 64 | 1 | 32 | 2 |
| 6000 | 2 | 64 | 1 | 32 | 2 |
| 8000 | 3 | 84 | 2 | 42 | 2 |
| 10000 | 4 | 64 | 3 | 32 | 3 |
| 12000 | 5 | 56 | 4 | 28 | 4 |
| 14000 | 7 | 68 | 5 | 34 | 5 |
| 14155 | 6 | 64 | 5 | 32 | 6 |

对于二维数组的快速排序,其列数固定为 1000,当排序的数据个数为 14155时,则行数定为 5,列数定为 2831。The number of column is fixed as 1000 for the quick sorting of two-dimentional array. The number of row is 5 and that of column is 2831, when the number of ordering figures is 14155.

办得到。在这种情况下,所需的附加栈空间最多为 $O(log_2 n)$。

二维数组与一维数组快速排序法相比,除了增加一些条件判断对排序速度稍有影响外,其平均计算时间仍为 $O(n\log_2 n)$,所需的附加栈空间为一维数组排序的两倍,仍为 $O(\log_2 n)$。

在快速排序中,如何选取集合中一个记录的关键字作为控制关键字,其选择方法见文献[7~9],一般的算法设计是把第一个记录的关键字作为控制关键字。

总之,快速排序算法在最坏的情况时结果不佳,但平均的性能指标是很好的,而且需要附加的存贮单元也有限,特别适合于数据量大而比较乱序的实际问题。

**3.2** 实验结果分析

根据前述的程序流程图,并应用 GW BASIC语言编制算法源程序,再用 IBM 编译 BASIC V1.0对源程序进行编译。其实验数据选取某水文站1954年4月1日至1992年12月31日逐年逐日的实测日平均流量资料,共14155个实测数据,然后用编译程序在PC386DX档的微机上运行,对不同的数据量进行从大到小的顺序排序。最后的实验结果见表1

从表1可看出:二维数组比一维数组快速排序的速度稍慢,所用的栈空间数是后者的两倍;一维数组的快速排序比一维数组的希尔排序稍快。由于受 BASIC语言系统利用最大内存空间的限制,所以无法对更大的数据量进行数值实验分析。

快速排序由于其速度快和数据结构简单而被广泛应用;二维数组快速排序与一维数组快速排序有类似的程序结构,且速度相当,因而易于为各应用部门所使用。

关于希尔排序[4~10]、超快速排序[11]和其它高效率的排序算法[12~15],如何设计出相应的二维数组排序算法,也是值得研究和解决的问题。

## 参考文献


1 徐洪亮.解决一维数组空间不够的方法.软件报. 1985,(1).

2 上海计算技术研究所. 电子计算机算法手册. 上海: 上海教育出版社, 1982. 988~992.

3 尹明万. 排序研究及其在水利水电工程中的应用. 成都科技大学学报, 1992,(2): 65~70.

4 华东师范大学计算机科学系, 郑州工学院科研所编. BASIC程序设计和算法基础. 上海: 上海科学技术文献出版社, 1983. 162~169.

5 霍义兴, 夏 炬, 汤宝骥编. 实用数据结构. 上海: 上海科学技术出版社, 1987. 103~106.

6 霍义兴,顾立尧等编.数据结构例解.上海: 上海科学技术文献出版社, 1983. 190~194.

7 [瑞士] N.沃思著. 算法+数据结构=程序.曹德和,刘椿年译,丘玉圃校. 北京: 科学出版社, 1984. 89~101.

8 Ellis Horowitz and Sartaj Sahni著.数据结构原理.本书编译组编.上海: 上海科学技术文献出版社,1988. 118~123, 132~142.

9 严蔚敏, 吴伟民编著.数据结构.北京: 清华大学出版社, 1987. 299~303, 316~318.

10 卢开澄. 组合数学算法与分析(下册).北京: 清华大学出版社, 1983. 221~227.

11 周建钦,马述杰.超快速排序算法.微计算机应用, 1995, 16(3): 25~28.

12 杨先泽.一类引人注意的排序算法.软件报, 1990,(27).

13 杨先泽.研究排序算法应该注意的一个问题.软件报, 1990,(28).

14 宋运康, 宋运国.论链式排序.软件报, 1992,(10).

15 蒋 志.排序新法—— 逆扫分量分布法.软件报, 1992,(33).


(责任编辑: 邓大玉)