用计算法预测某金矿带金总量结果表（电算） 表4

储量基准（t）	取秩	已知储量（t）	预测储量（t）	资源潜力（t）	资源潜力（%）
>1	142	90.8	786.191	695.391	88.45
>5	28	67.1	557.658	490.558	87.9
>20	7	48.7	368.18	319.48	86.77

2.对某银矿带资源总量的预测

用同法，其预测结果列于表5。

笔者曾征询过在该区工作过的专家，他对该区的银资源潜力百分比主观概率估计与预测结果是十分相近的（估计为30～40%）。

某银矿带银资源总量预测结果表（电算） 表5

储量基准（t）	取秩	已知储量（t）	预测储量（t）	资源潜力（t）	资源潜力（%）
50	24	2588.95	4565.13	1976.18	43.28

应用齐律进行资源总量预测的关键问题之一，是求解最佳齐律拟合线。求解最佳齐律拟合线数组的数学方法很多，相信本文提供的数学方法并非唯一的，但应遵循齐波夫指出的人类行为的最省力原理，来探求出既能求解又最省力的数学方法。

Computational Criterion for Best Fitting Line Data by Zipf Law and Its Application in Resources Prediction

Xie Qiaoxun Wang Xiaojun

(*Zhengzhou Geological School*)

Abstract

In this paper the author advances the first rank successive increasing method for best fitting line data by zipf law. A BASIC computor program is also provided. This method has been used to resources prediction for a Au-Ag ore belt and good result was yielded.

快速排序算法的微机应用程序

张 益 民

（中南冶金地质研究所）

介绍了一种改进的快速排序算法，给出一个可直接在IBM--PC/XT机上，用BASIC语言编写的快速排序程序。本程序具有非递归调用、快速排序、更趋稳定的时间复杂性等特点。

在物化探数据处理中，常利用大量离散的、非规则网分布的场观测数据，进行网格化处理。为提高此项工作的效率，通常先要对观测数据按测点的某一坐标值进行排序。当观测点数量较大时，用一般的排序算法相当费时，普通微型机难以实现这一过程。当前，在利用微型机进行各种事务管理及数据和表格处理过程中，排序和检索是最基本的操作。检索速度是衡量一个管理程序系统优劣的重要标志，而检索总是在排序的基础上来实现的。这样，排序方法的改进就十分必要。目前常用的排序方法有：选择法、冒泡法、堆排法、归并法等。尽管这些方法各有特色，但其共同缺点是排序速度慢，尤其是当排序的序列较长、计算机运算速度不高时，问题更为突出。因此，又研究出一种快速排序新方法，该法可在微机上实现较大的排序过程，在微机管理应用中，可提高管理程序运行效率。

快速排序原理与方法

设有一N个元素的杂乱序列，存放在数组A（N）的后N个数组元素中。如果运行如下一段BASIC程序：

```
140 A (0) =A (1)
150 FOR I = 2 TO N
160 FOR J = I - 2 TO 0 STEP - 1
170 IF A(I) ) =A (J) THEN A (J + 1)
                =A (I) : GOTO 210
180 A (J + 1) =A (J) : IF J = 0 THEN A (J) =A (I)
190 NEXT J
200 NEXT I
```

就可在数组A（N）的前N个元素中，顺次得到按数值由小到大排列的有序序列。该程序很简单，原理清楚，称为“冒泡法”。

但是，为完成N个数的排序，需作N^2次数值大小的比较，还要作不多于N^2次的数据交换，计算时间将正比于N^2，或其时间复杂性为N^2。笔者曾在IBM—PC/XT机上用解释BASIC试验性运行过该程序，当$N=100$时，所用机时$CPU=34.66$秒；$N=200$时，$CPU=117.59$秒；而当$N=1000$时，$CPU=2931.66$秒。

从试验看出，当N更大时，该排序算法在一般微机上效率很低，已失去实用意义。因此，人们又从减小时间复杂性入手，找到一类快速排序的新方法，以提高排序速度。

若有N个元素的序列为S，按上述方法排序，其时间复杂性为N^2。那么，只要将S序列分成元素个数大致相等的两个子序列S_1和S_2，令S_1集中原序列中较小的序列元素，S_2集中剩下的较大的序列元素，这时，由于S_1和S_2的排序时间复杂性均接近于$(N/2)^2$即$N^2/4$，则两个子序列总的时间复杂性为$N^2/2$。这就是说，一个较长的序列，在将其划分为两部分排序后（不计附加工作量所耗时间），则可节省一半的排序时间。若再分别对S_1和S_2继续上述过程，则各序列总的排序时间复杂性为$4\times(N/4)^2=N^2/4$。依此，直至所有的子序列最多包含两个元素，才停止这一分割，此时为完成所有子序列总的排序时间复杂性为$N\log_2 N$。当N稍大时，用$N\log_2 N$与N^2比较，可看出通过这一处理能提高排序的效果。例如，当$N=1024$时，处理的时间复杂性仅为原来的1%。实算时，其排序效果要比理论值低得多，因为在运行BASIC程序时，程序解释需要时间。应用此法虽大大缩短了纯排序时间，但也增加了一些附加工作量。

如何将S分成S_1和S_2两部分序列呢？其方法是在序列S的两端设立两个指针I和J，它们分别向序列的另一端移动，指针I每指向序列的一个元素时，将该元素与序列的平均值E相比较，若$S_i>E$，则将指针I停下。指针J指向的序列元素$S_j<E$时，令指针J也停下来，交换S_i和S_j两个元素在序列S中的位置，然后指针I和J继续各向序列的另一端移动。这样，比较→指针停→交换两个元素在序列中的位置→继续移动指针，不断进行，直到J所指的序列的序号I_E，大于J所指的序列元素的序号I_E（即$I_E>J_E$时），上述过程终止。此时，序列的前J_E个元素划归序列的前半部分，从序号为I_E起到序列末尾的序列元素划归序列的后半部分。

用该法划分出的前半部分子序列包含了原序列中数值较小的J_E个元素，后半部分序列则包含剩下的较大元素。这样，可对划分出来的两个子序列分别排序来等效于对原序列排序。因I和J指针移动过程中其所指元素的值是与序列的平均值比较的，故其所分的两个子序列均各有近一半个数的元素（图1）。

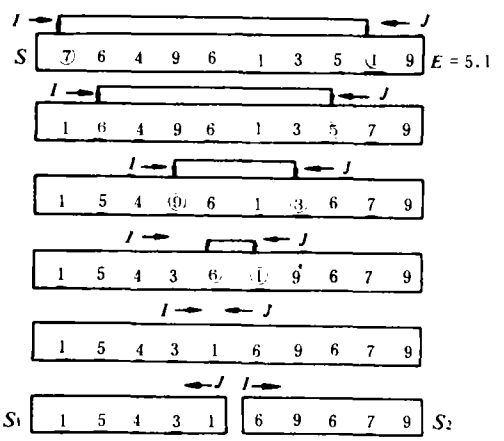

图1 分序示意图

通过上述过程处理，N个元素的序列的排序时间复杂性就从N^2下降到$N^2/2$，加之其间要增加N次比较，进行不大于$N/2$次数据交换和N次加法运算，其实际时间复杂性接近于$N^2/2+N$，故N较大时，可减少约一半的计算工作量。

快速排序程序

```
10 REM This is sorting program
20 CLEAR:DEFINT I-N:WIDTH "LPT1:",130
30 INPUT "N=";N:LPRINT " SORTING LONG :     N=";USING "#####";N
40 DIM A(N),K(N)
50 FOR I=1 TO N:A(I)=INT(N*RND(I)):NEXT I
60 LPRINT:LPRINT "******** SOURCE DATA "
70 FOR I=1 TO N:LPRINT USING "########.####";A(I);:NEXT I:LPRINT
80 CPU=TIMER:K(1)=N+1
90 FOR I=2 TO N:K(I)=0:NEXT I
100 FOR I=1 TO N:IF K(I)<=I+1 GOTO 120
110 IF K(I)<>I+2 THEN IL=I:IR=K(I)-1:IS=K(I)-I:GOSUB 200:I=IR  ELSE  K(I)=0:
    K(I+1)=0:I=I+1:IF A(I-1)>A(I) THEN A(0)=A(I-1):A(I-1)=A(I):A(I)=A(0)
120 NEXT I
130 FOR I=1 TO N:IF K(I)<>0 GOTO 100
140 NEXT I
150 CPU=TIMER-CPU:LPRINT "******** SORT RESOURLT "
160 FOR I=1 TO N:LPRINT USING "########.####";A(I);:NEXT I:LPRINT
170 LPRINT TAB(100)"CPU:  ";:LPRINT USING "######.##";CPU;:LPRINT "  (SEC)"
180 END
200 REM This is sorting subroutine
210 E=0
220 FOR J=I TO IR:E=E+A(J):NEXT J
230 E=E/IS
240 ML=0:MR=0:JL=IL-1:JR=IR+1
250 JL=JL+1:JR=JR-1:IF JR<JL GOTO 360
260 IF A(JL)>=E THEN ML=JL
270 IF A(JR)<=E THEN MR=JR:IF ML=0 GOTO 320 ELSE 350
280 IF ML=0 GOTO 250
290 JR=JR-1:IF JR<JL GOTO 360
300 IF A(JR)<=E THEN MR=JR:GOTO 350
310 GOTO 290
320 JL=JL+1:IF JR<JL GOTO 360
330 IF A(JL)>=E THEN ML=JL:GOTO 350
340 GOTO 320
350 A(0)=A(JL):A(JL)=A(JR):A(JR)=A(0):ML=0:MR=0:GOTO 250
360 IF JR=I THEN K(I)=0 ELSE K(I)=JR+1
370 IF JL=IR THEN K(JL)=0 ELSE K(JL)=IR+1
380 RETURN
```

快速排序程序

依据上述原理，笔者在IBM—PC/XT机上，运用AdvancecIBASIC语言编制了一个快速排序的微机应用程序（图2、3）。

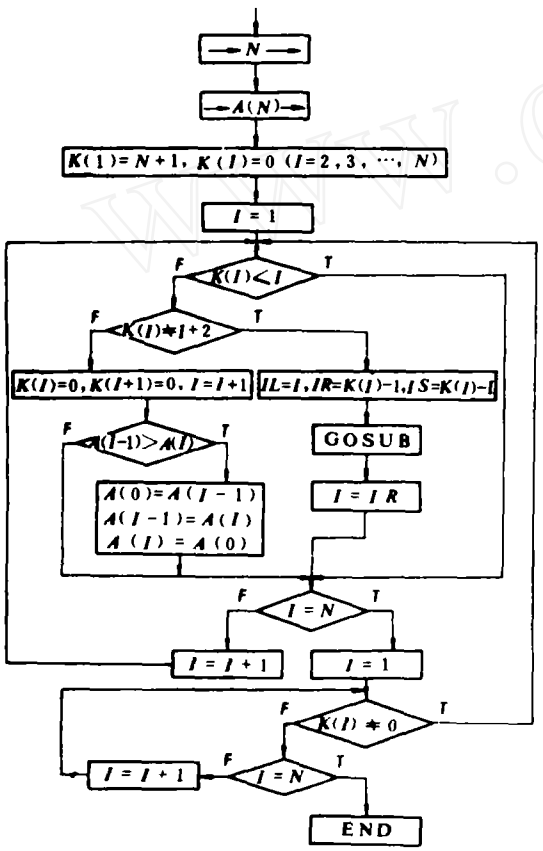

图2 主程序框图

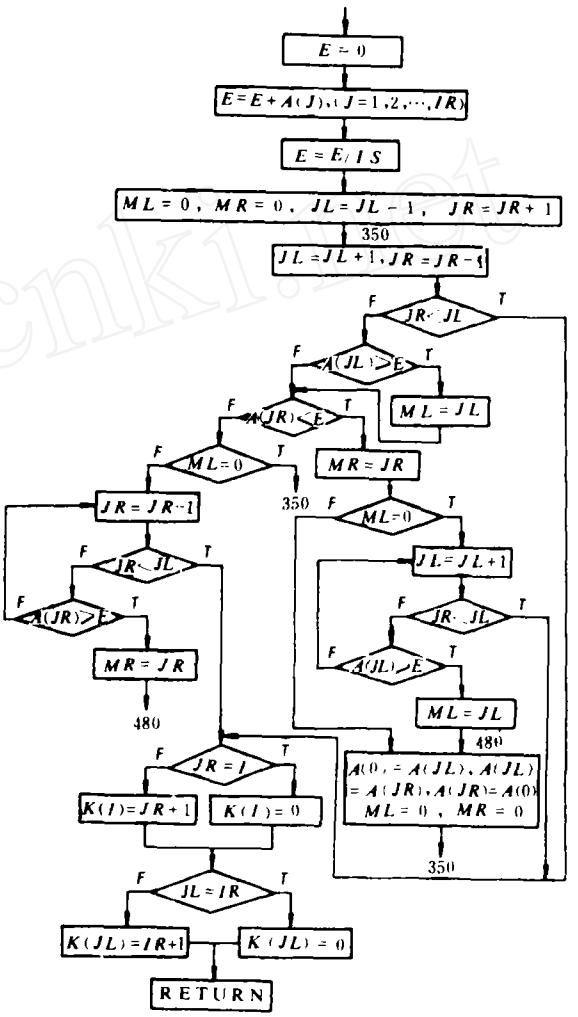

图3 子程序框图

从快速排序原理可看出，这是一个递归调用过程。为避免递归调用，本程序将其分为主、辅程序两部分，即10～180语句的主程序和200～380语句的子程序。子程序只用来完成图1所示范围内的运算，即将一个长的序列，分成序列元素数值较小的前半部分和较大的后半部分。在主程序的40语句内，定义了两个数组A（N）和K（N），其中K（N）为整型工作数组，依据其N个数组元素的值，提供序列分段状况，记载下调用子程序之后的序列分段的变化。

整个快速排序过程为：依据K（N）数组提供的信息调用子程序，对大于两个元素的子序列进行分段。而当子序列只剩下一个或两个序列元素时，不再调用子程序，而对两个元素的序列直接进行排序。从序列的第1个元素到序列最后的第N个元素，反复进行扫描处理。直到A（N）数组提供的N个元素的序列，其中包含的子序列只有一个或两个元素，并对两个元素的子序列已排好序为止。程序中100～200语句即执行这一使使命，其中130～140语句则判定是否进行上述扫描过程。

这里给出一个可在IBM—PC机上直接运行的完整程序。其中包括排序数据的提供和排序前后序列元素的行式打印机（132列）输出，作为一个试验性实用程序，程序中50语句表明是由随机函数提供随机数序列的。为将其应用于实际，

可改换50语句及改变排序序列的提供方式。当单独运行本程序时，只要通过键盘回答序列长度N即可。若将其作为子程序调用应去掉其中的30、50等语句，稍加改变即可。

应用举例

为检验本程序的排序效果，笔者在IBM—PC/XT机上，用本程序与冒泡法及其他排序方法作了计算对比：

方法 \ 序列	100	200	500	1000	5000	7500	9000
冒泡法	34.66	117.59	700.85	2931.60			
快速排序法（解释）	16.59	37.13	106.39	224.65	1342.71	2119.13	2560.03
快速分类合并法（解释）					1582.90	2392.17	
快速排序法（编译）				15.00	87.00	135.00	157.00

对比表明，本程序具有如下特点：

1．本程序避免了递归调用，结构简单，条理清晰，阅读容易；

2．本程序达到了快速排序的目的，当序列较长时，其效果尤为突出；

3．由于本程序是用序列或子序列的平均值代替中位数值（序列中间位置的那个序列值），并作为分序比较标准E，这样虽增加一些计算工作量，但却避免了同类程序因取用中位数值作比较标准而可能导致的运算失败。如果不考虑分列时的附加工作量，本程序的排序时间复杂性更加稳定地趋于$N\log_2 N$。

4.实用中，多是对实数序列排序，此时通常要开辟一个与序列长相等的整型数组做工作单元，这对于IBM—PC/XT类微机，意味着节省一半的工作单元，可供更长的序列进行排序。表中快速排序法和快速分类合并法两栏中，给出了IBM—PC/XT机两种快速排序算法程序能进行排序的最大序列长，前者为9000左右，后者为7500左右。本方法仍需开辟一定量的工作单元，以供排序时应用。

参考文献

［1］ 张汉亭：微计算机应用，1985，第4期，第9～11页

［2］ 李国强等：计算机应用与软件，1985，第2卷，第4期，第63～64页

An Application Program of Quick Sorting Algorithm

Zhang Yimin

(*Central South Geological Institute, Ministry of Metallurgical Industry*)

Abstract

In this paper an improved quick sorting algorithm is presented and a corresponding program, written in BASIC language and capable to run on a IBM-PC/XT computer, is also given. Comparing the results calculated by using this program with those computed by other sorting algorithm, it shows that our sorting algorithm is characterized by its non-recursive procedure, high speed sorting and stable time complexity.