第9卷 第1期 华侨大学学报(自然科学版) Vol.9 No.1
1988年2月 JOURNAL OF HUACHIAO UNIVERSITY (NATURAL SCIENCE) Feb. 1988

微机外排序EMSD SORT方法及其实现

张 银 明

〔计算机科学(电脑)系〕

摘 要

本文介绍一种在微机上进行随机数据文件EMSD外排序方法的基本思想及其实现.这是一种有效的、较为快速的和少占用存贮空间的排序方法，尤其适用于大容量数据文件的外排序.

一、前 言

SORT有的称为排序，有的则叫做分类. 居于分类中的“类”含有类型之意,而实际上，SORT所担负的任务则是按关键字的码值进行递增或递减的顺序排列，且所排序的对象应是同类型的. 故把SORT称为排序似乎更贴切一些. 这是本文叫做排序的因由.

D. E. Knuth曾断言：“程序设计的每一个重要方面，实际上都离不开排序和查找”. 也就是说，无论对系统软件还是对应用软件，排序和查找都是极为重要的课题. 实际情况表明，排序在计算机应用中占有很大的份量. 因而它的效率已成为一个应用软件系统性能的关键因素之一. 为提高排序的效率，至今已有不少成功的方法. 然而，明显的事实是内排序比外排序的方法多且高效，尤其微机中的外排序更不理想. 如dBASEⅡ中的SORT语句，速度之慢令人难于忍受. 所以，有必要研制一种较为有效的外排序方法，它具有对大容量的磁盘随机数据文件进行排序的能力，速度较快. 使用辅助空间较少，且能在数据文件的存贮区内进行排序.

按照上述目标，根据微机特点，本人研制了一种EMSD (External Most Significant Digit) SORT的外排序方法，使用按值定位的方式进行排序. 在IBM微机上，模拟配件库数据文件，对1350个记录，以机件类型为关键字进行排序，花了一分半钟. 如对3500个具有四位数为关键字的数据文件进行排序，费时31分钟. 当记录缩为1750个时需11分钟. 虽速度尚不算很快，但比dBASE的SORT则要快得多，且对大容量的数据文件进行外排序是很有效的. 使用该方法，用APPLEⅡ的Soft BASIC编程，对DISK上的数据文件进行排序也获成功. 若使用汇编语言或目标语言编程，其速度可望有较大的提高.

下面着重介绍EMSD SORT方法的基本思想及其实现方法，我们约定排序按递增方式.

本文1986年6月26日收到.

二、一种特殊情况的外排序方法

我们所述的配件库存数据文件含有六千多个记录，其机件类型仅分八种。因而在数据文件的属性域中，仅占一位，其值范围为1至8。所以，要以机件类型作为关键字进行排序，完全不必进行属性域取值比较。实际上是把所有记录分成八块，只要能确定各块的长度和始点，便可依照按值定位方式进行排序。

假设数据文件为F$，含有 n 个记录，L(I)表示关键字为I的记录个数，RG(I)表示关键字为I的记录位置指针。那么，其排序的过程可由图1的流程图表示。其中所指的特殊关键字系指仅含一位数字且取值为1至8的数据文件。从流程图中可以看到，这种排序具有下面的特点：

(1)排序在原来的数据文件的存贮区中进行，使用的辅助工作单元极少。因而，只要外存磁盘容得下，便可对它进行排序。

(2)由于设计了按值定位的排序方式较为巧妙，因而对任一个记录皆可一次定位。

(3)排序时间为O(n)，速度快。

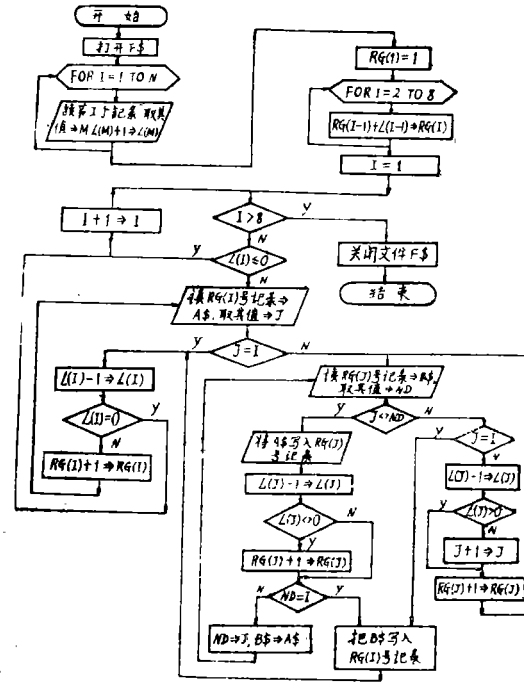


图1　特殊关键字的外排序流程图

按照流程图，使用BASIC语言编程，对配件库存数据文件的1350个记录，以机件类型为关键字进行排序，只需90s。

为使按值定位的排序方法说明的更为具体，下面以具体的数据记录来阐明它的排序过程。

假设有下列数据：

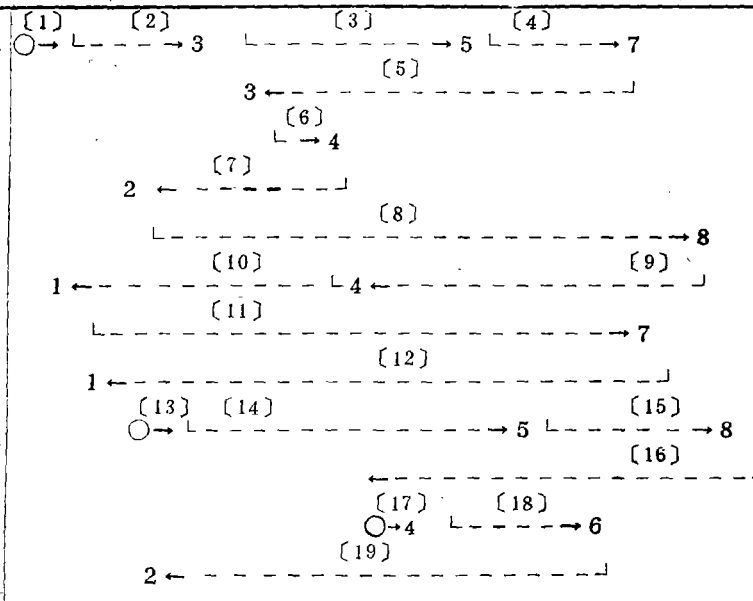
1，3，7，8，5，5，4，3，2，1，4，6，7，8，2，3，1，4，4，8。

那么，对此进行的排序过程如表1所示。

表中的方括号所标数字表示排序过程的顺序步骤，箭头表示记录传送方向，小圆圈表示该记录不必移动。

排序从I=1开始，因L(1)=3，所以含1的记录应占三个记录，而它是第一个分块存贮区，则其记录位置的指针起始点为RG(1)=1。当读第一个记录时，其值为1，同I相等，说明该记录不必移动，但此时存贮区长度L(1)应减1，亦即L(1)=2，而指针RG(1)要加1，则RG(1)=2。从而前进到第二个记录，其值为3，不等于I，所以应转向RG(3)所指的记录，也就是第6个记录，读出5并与RG(3)中的3比较，因不等，它应让位，并写入3，同时，L(3)减1，RG(3)加1。该记录6所读出的值为5，又按其值去寻找

表1 按值定位排序过程

KEY \ L(J)	3	2	3	4	2	1	2	3
0	1 3 7	8 5	5 4 3	2 1 4 6	7 8	2	3 1	4 4 8
1								
2								
⋮								
结 果	1 1 1	2 2	3 3 3	4 4 4 4	5 5	6	7 7	8 8 8

它对应的存贮块位置。表1给出了详细的按值定位的过程，而表2则列出了排序各步骤所对应的L(I)与RG(I)的变化情况。排序进行到所有的L(I) = 0 (I = 1, 2, …, 8)，便告结束。

从上例可见，记录在排序过程中的传送次数只有16次，反而比记录数少，至多同记录数相同。这是按值定位排序的特点，也正是它的巧妙之处。

该例虽属特殊，然而其排序的思想和方式则具有普遍意义，也就是说，可把它用到一般数据文件的排序上去。这正是本文的中心内容。

三、EMSD SORT 方法及其实现

MSD SORT是内排序的一种基数排序方法，它采用堆栈分配、收集及链接的方法进行排序。而这里是借用最高位优先的排序思想，但使用的是按值定位的外排序方法，称为EMSD SORT排序方法。

1. EMSD SORT 的基本思想

假如记录的关键字由 $x^1x^2\cdots x^k$ 组成，x^1 是关键字的最高位，x^k 是最低位。当且仅当每两个记录 i, $j(i<j)$，有 $x_i^1x_i^2\cdots x_i^k \leqslant x_j^1x_j^2\cdots x_j^k$ 时称记录 $R_1R_2\cdots R_n$ 是按关键字 $x^1x^2\cdots x^k$ 排好序。

如果关键字是数字，则可把每个十进制位作为一个关键字位，其值必在0至9之间，因

表2 记录存区长度L(I)及指针RG(I)的变化情况

L(1)	RG(1)	L(2)	RG(2)	L(3)	RG(3)	L(4)	RG(4)	L(5)	RG(5)	L(6)	RG(6)	L(7)	RG(7)	L(8)	RG(8)
3	1	2	4	3	6	4	9	2	13	1	15	2	16	3	18
(1)2	2			(2)2	7			(3)1	14			(4)1	17		
				(5)1	8	(6)3	10								
		(7)1	5											(8)2	19
						(9)2	11								
(10)1	3														
(12)0	3											(11)0	17		
		(13)1	5					(14)0	14					(15)1	20
						(16)1	12								
						(17)0	12			(18)0	15				
		(19)0	5												
				(20)0	8	(21)0	12	(22)0	14	(23)0	15	(24)0	17	(25)0	20

而可以从高位开始，对每位使用二、所述的按值定位的方法．这样，当关键字的所有位都排好序后，整个文件的记录也就完全排序好了．

2. EMSD SORT 的实现

假设记录的关键字由 K 元组 $x^1x^2\cdots x^k$ 组成，其中 $x^i\epsilon[0, 9]$．排序从 x^1 开始，直至 x^k 为止．由于每位 x^i 可能有 0—9 十种取值，所以对 x^1 进行排序时，实际上是将 n 个记录按 x^1 取 0，1，…，9 的十种情况分成十块．因而，只要能确定每块的长度及其起点位置，也就可以采用按值定位的方法进行排序．为此，必须分别计算取值为 0，1，…9 的记录个数，也就是每块的长度 $L_1(I)$ 以及相应的记录起始位置 $N_1(I)(I=0, 1, \cdots, 9)$．尔后按二、的方法排序．接着，对 x^2 的排序则是在 x^1 排好序的十块中，又对每一块分别按 0，1，…，9 的顺序排成十块．此时，须以 $N_1(I)$ 为起点、以 $L_1(I)(I=0, 1, \cdots, 9)$ 为记录数计算 $L_2(J)$ 和 $N_2(J)(J=0, 1, 2, \cdots, 99)$．並对每一块进行排序．依此类推，直至 x^k 排好序，整个文件的排序也就结束．

这里的问题是要考虑记录数 n 同关键字位数 d 的关系．当 $\log n\geqslant d$ 时，此时只要对关键字中的每一位采用上述的按值定位方法进行排序便可完成整个文件的排序，这种情况不妨称 n 同 d 相匹配．而当 $\log n<d$ 时，比如 $n=1000$，$d=6$，$\log 1000=3<d$，此时称 n 同d不匹配．这种文件的排序不能对关键字的每一位采用按值定位的方法，而可对前 INT ($\log n$) 位进行按值定位的排序．最后，在各个小块中使用折半插入的排序方法或其他较好的排序技术．

由于按 x^i 的排序过程要反复计算各块长度及其记录起点，所以可把它作为子程序，称为CLBB (Calculate length and Beginning of Block) 子程序．而把按 0，1，…，9 的按值定位排序作为子程序 LPAV (Locate the Point Acccording to the Value)．同样地，定义折半插入的排序方法为子程序 BIS (Binary Insertion Subroutine)．

为使算法更直观和书写方便而使用类PASCAL，並约定下面的语句：

WRITE (R(I)，A) ——表示把 A 写入第 I 个记录；

READ (A，R(I)) ——表示将第 I 个记录读入 A；

J：=A↑．KEY (K) ——表示关键字的第 K 位赋给 J．这样，子程序 CLBB 的算法可表示成：

```
algorithm CLBB(L, V, IB, IE, K)
for i=0 to 9
    L(i):=0;
for i=IB to IE
    read(A, R(i))
    I1:=A↑.KEY(K)
    L(I1):=L(I1)+1;
    RG(0):=NRB;
for i=0 to 9
    RG(i):=RG(i-1)+L(i-1);
end(CLBB)
```

该子程序的功能是计算从记录起点 IB 开始到终点记录号IE为止的记录块中，关键字第K

位含 0，1，…，9 的记录数及相应的起始位置，並分别存在 L(i)及 RG(i)(i = 0, 1, …, 9) 之中。

LPAV 子程序的算法如下：

```
algorithm LPAV(R, K)
i : =1
while i≤9 do
    begin
    while L(i)>0 do
      begin
      read(A, R(i))
      J : =A↑. KEY(K);
      while J≠i do
        begin
        read(B. R(RG(i)))
        ND : =B↑. KEY(K);
        if J≠ND
          then begin
           while(R(RG(i)), B)
            L(j) : =L(j)-1;
            if L(j)≠0
             then RG(j) : =RG(j)+1;
               if ND≠i
                then begin
                 j : =ND
                 A : =B
                    end;
                end;
            if j≠i
             then begin L(j) : =L(j)-1;
                   if L(j)≠0
                     then j : =j+1;
                   RG(j) : =RG(j)+1
                   end;
      end;
      write(R(RG(i)), B);
    end;
  L(i) : =L(i)-1;
   if L(i)≠0
    then RG(i) : =RG(i)+1;
    end;
i : =i+1;
end(LPAV)
```

LPAV 子程序的功能是根据关键字第 K 位的取值进行按值定位的排序。

子程序 BIS 的算法可写成：

```
algorithm BIS(R, IB, IE)
for i=IB+1  to IE do
  begin
    low : =IB
    high : =i
    while low≤high do
      begin P : =L(low+high)/2 1
            if R,↑. KEY>R,↑. KEY
              then high : =P-1
              else low : =P+1
      end
    p : =low
    if p<i
      then
        begin
          temp : =R,
          R,+1…R, : =R,…R,-1
          R, : =temp
        end
  end
end(BIS)
```

EMSD SORT 是通过适当地调用上述三个子程序来实现的。在调用过程中，要根据关键字排序的位数给出各记录块的起点和长度。为此，必须定义相应的数组。这同文件的记录数 n 与关键字的位数 d 有关。设 $d_1=[\log n]$，则可定义：

$$m=\begin{cases} d & \text{当 } \log n>d \text{ 时} \\ d_1 & \text{当 } \log n\leqslant d \text{ 时} \end{cases}$$

相应地，为简化算法而定义以下的数组：

$L_0(0)$，$L_1(9)$，$L_2(99)$，…，$L_{m-1}(10^{m-1}-1)$

$N_0(0)$，$N_1(9)$，$N_2(99)$，…，$N_{m-1}(10^{m-1}-1)$

那么，EMSD SORT 的示意简化算法可表示为：

```
algorithm EMSD SORT(R, n, d)
  d1 : =〔log n〕;
  if d1>d
    them m : =d
    else m : =d1;
  N0(0)=1
  L0(0)=n;
for K=1 to m do
```

```
begin
for IK=0 to IE(k-1)-1
  begin IB:=N_{k-1}(IK)
        IE:=IB+L_{k-1}(IK)-1
        NRB:=IB
      CALL CLBB(L, R, IB, IE);
      if k≠m
        then begin
             J:=IK*10+9
             for IJ=0 to J
               IN:=IJ-INT(IJ/10)*10
               LK(IJ):=L(IN)
               N_k(IJ):=RG(IN)
               end;
       CALL CPAV(R, K)
  end;
end;
  if d1>d
    then begin for IK=0 to IE(m-1)-1
                IB:=N_{m-1}(IK)
                IE:=IB+L_{m-1}(IK)-1
                CALL BIS(R, IB, IE)
         end;
end(EMSD SORT)
```

依照该排序算法，使用BASIC语言编程，分别在IBM和APPLE Ⅱ微机上，对硬盘和软盘数据文件（记录数为3500）进行排序都获成功。因此可以说这种外排序方法是有效的，且相对速度较快，适用于大容量的数据文件。

四、EMSD SORT 算法分析

EMSD SORT算法已经实践检验，无疑是正确的。

1. 时间复杂性

按值定位是对关键字的每一位进行的。

当 n 与 d 相匹配时，在每位排序过程中要对记录关键字作两次比较访问。第一次为的是计算各块长度，第二次便是实现按值定位。所以，一个具有 n 个记录的数据文件，当排序完成时，总的比较次数为 2nd。故其时间复杂性为 O(n)。

而当 n 与 d 不相匹配时，则对关键字的前 d_1 位，其比转次数为 $2d_1n$。其后的 $(d-d_1)$ 位是在 $10^{m-1}-1$ 个小块中进行排序，可选取一种最佳的排序方法与按值定位混合使用。比如上述算法中选用的是折半排序。不妨假定各小块长度分别为 n_1，n_2，…，$n_{10^{m-1}-1}$，而其最长的一块为 N，那么，必存在一个常数 M，使平均的排序时间为

$$T_v(n) \leqslant \sum_{i=1}^{10^{m-1}-1} M' \cdot n_i \cdot \log_2 n_i \leqslant M' \cdot N \cdot \log_2 N.$$

因而，在这种情况下，其排序时间可看成两部分组成，为说明方便而表示为

$$O(n) + O(M' \cdot N \cdot \log_2 N)$$

这后一项表明时间复杂性主要取决于N，但N值是很小的。故使通常外排序速度随记录数增加而急剧下降的问题得以解决。

EMSD SORT方法在排序时，记录的交换次数是很少的。当按某位排序时，一个记录的交换次数可能为0，最多为1。所以，排序的交换次数其下界为0，上界为d·n。如d与n不匹配，对$(d-d_1)$部分的排序交换已计入T_v之中。显然，关键字中任一位数的值同块号相等的数目越多，其排序的交换次数也就越少，速度越快。

2. **算法的空间复杂性**

由于排序是在数据文件的原存贮区中进行。因而可以不占用其它外存空间。在排序过程中所使用的数组$L_i(10^{i-1}-1)$及$N_i(10^{i-1}-1)(i=1, 2, \cdots, m)$，其元素总和为

$$2[1+9+99+999+\cdots+(10^{m-1}-1)] = 2\sum_{i=1}^{m-1} 10^i - 2(m-1) \approx \frac{1}{5} \cdot 10^m$$

亦即数组元素个数约占记录数n的五分之一。由于数组元素皆为整型，所占存贮空间很少，至于其它工作单元仅有少数几个，因而其空间复杂性已降至极低的程度。

从上述讨论中可以知道，EMSD SORT方法的按值定位排序方式是对关键字逐位进行的。第一位是把整个文件作为一块进行排序，而从第二位开始以后便依次分为$10、10^2、\cdots、10^{m-1}$块，此时每位排序的各块之间已无须进行记录的交换和传送，亦即各分块可以独立进行排序。因而，若得到并行处理的支持，便可对各分块同时使用按值定位的方法亦即进行并行处理。显然，由于分块随着关键字位数的推进而以10的倍数增多，所以并行处理的程序也将随之提高。

而且排序过程的按值定位可在各块内进行，只要各块排序好了，整个文件的排序也就顺利结束，这样，排好序的各块也无需进行合并。

该排序虽有其本身的特色，但由于排序全在外存进行，其速度深受影响，这是因为磁盘上的记录访问及交换要付出记录定位和读写的寻找时间、等待时间和数据记录的传输时间，这些时间占了整个排序时间的大部分。这就是说，如能更好地发挥内存资源的优点来克服这个不足之处，排序的效率将得到改善。

五、改进的EMSD SORT方法

要减少寻找、等待与交换记录的传送等时间，就要考虑如何达到输入/输出与CPU处理之间的最优化平衡。这当然受到CPU处理的牵制，亦即随其任务的条件而变化。而主要的因素有记录的大小、关键字的长度、组块因子、中央处理机的速度和输入/输出的速度。

为利用内存资源加快排序速度，通常采用的方法是把数据文件分成若干块。尔后，分别把各块输入内存进行排序，再进行合并，以完成对数据文件的排序，其中每一步都要选择最优的策略。本文並不想涉及这方面的讨论，而要介绍一种较为新颖的处理方法。

目前用于数据处理的微机，一般其内存是可以扩充的。如IBM-PC便可扩充至512k，最大可达640k，供用户使用的内存空间也有500多k。通常要进行处理的数据文件，其记录数最多达16000多个，若仅取其关键字加上记录号，每个记录约长20 byte左右，总共存量还不到315 k。这就是说，可把整个数据文件的关键字加上记录号存入内存。据此，可把EMSD SORT方法的改进由下述过程来实现。

（1）假设要进行排序的数据文件为ESFILE，共有n个记录。由它的关键字和相应的记录号组成的新文件为NEWFILE，其数据结构形如：

记录号	关键字	原记录号
RN	NKEY	OLDRN

（2）对NEWFILE文件的关键字NKEY，在内存中使用按值定位的EMSD SORT方法进行排序。

（3）按排好序的NEWFILE文件中的记录顺序，对OLDRN值存入数组A(i)，$i=1, 2, \cdots, n$。

（4）如果文件ESFILE以NKEY为关键字进行排序后生成的文件叫为SORTFILE，那么，根据A(i)元素的值为记录号读取ESFILE文件的相应记录，並依次存入SORTFILE文件便是所要求的按指定关键字排好序的数据文件。

根据上述过程，其相应的算法简述如下：

（1）打开ESFILE及NEWFILE文件；

（2）置初值：I←1，IN←n；

（3）读取ESFILE第I个记录的关键字KEY(I)，NKEY(I)←KEY(I)，OLDRN←I，RN←I；

（4）I←I+1，若I≤IN转（3）；

（5）对NEWFILE文件，按NKEY调用按值定位的EMSD SORT程序；

（6）按记录顺序，将NEWFILE文件的OLDRN(i)值存入数组A(i)，$i=1, 2, \cdots, n$；

（7）读取ESFILE文件的第A(i)号记录，並把它依次存入SORTFILE文件，$i=1, 2, \cdots, n$；

（8）关闭ESFILE及SORTFILE文件，删除NEWFILE文件。

这个算法是容易实现的。

由于排序的过程移至内存进行，又是应用EMSD SORT方法，因而在形式上用于比较的次数CN也同原来一样，亦即

$$CN=\begin{cases}2dn & \text{当 } n \text{ 与 } d \text{ 相匹配时}\\ (2d_1+M'\log_2 N)\cdot n & \text{当 } n \text{ 与 } d \text{ 不匹配时}\end{cases}$$

用于记录交换的次数有：（1）在形成NEWFILE文件时，由外存读入内存一次；（2）当把ESFILE文件的记录按A(i)的记录号顺序进行调整而成为SORTFILE文件时，在外存交换一次，其总交换次数的上界≤2n。

改进后的方法比原算法多了$2dn$或$2d_1n$次的内存交换，但少了$2(d-1)n$或$2(d_1-1)n$

次的外存交换，这将显著地提高排序的速度。同样，在使用混合方式排序时，其时间复杂性也有相应的降低。

这个方法要使用较多的辅助存贮空间。一方面内存要开辟存贮 NEWFILE 文件的较大存贮空间以及数组 A(*i*)的存贮开销。这正是使内存得以充分利用以便更好发挥其特点加快排序的目的。另一方面外存空间的增加则可根据排序的要求而定，一般要求排好序的数据文件另外命名，且与被排序的数据文件並存，以便于记录的调整。当然，如果无须保留原有文件，则可把它删除，假如外存甚至不能提供同原数据文件相同的存贮空间，那么，完全可在原数据文件存贮区中进行调整，这並不难实现。

在内存对关键字进行排序，而在外存进行记录调整，以实现整个文件的排序，这是一种可行而新颖的外排序方法。

参 考 文 献

〔1〕D.E.克努特著，营纪文等译，计算机程序设计技巧，第三卷（排序和查找），国防工业出版社，（1984）.
〔2〕王广芳等编著，电子计算机软件（数据结构），湖南科学技术出版社，（1984）.
〔3〕A.S.菲利帕基斯等著，朱致远等译，COBOL 信息系统，上海科学技术出版社，（1983）.
〔4〕张学平，dBASE Ⅱ 分析报告，计算机工程与应用，4（1984）.
〔5〕N.沃思等著，俞嘉惠译，程序设计语言 PASCAL，科学出版社，（1983）.
〔6〕Peter Grogono 著，蒋国南译，PASCAL 程序设计，清华大学出版社，（1983）.

An External EMSD Sorting Method and Its Performance on Microcomputer

Zhang Yinming

Abstract

This paper describes the basic idea of an external FMSD sorting method for random files and its performance on microcomputer. It is an ingenious and rapid method of sorting, suitable especially for mass data files in less storage space.