中国科学 (E 辑)
第 26 卷 第 6 期 SCIENCE IN CHINA (Series E) 1996 年 12 月

纵、横、斜——倾斜与振荡法多路归并排序*

高庆狮 (北京科技大学计算机系，北京 100083)
刘志勇 (中国科学院计算技术研究所，北京 100080)

摘要 提出一种新的归并技术，称为倾斜与振荡法多路归并，且提出建立在这种多路归并基础上的排序技术(Sloping-and-Shaking K-way Merging and Sorting)，并且讨论该类算法的时间复杂度. 所提出的归并算法的主要特色是：它不是建立在 2-Way 归并的基础上，而 2-Way 归并可以做为它 $K=2$ 时的 1 种特例. 指出 Sloping-and-Shaking K-Way 归并及排序速度可以高于 2-Way 归并及排序.

关键词 并行算法 归并排序 归并与排序网络 复杂度

归并与排序操作在计算技术领域有着广泛的应用，它们往往构成许多计算过程的核心操作，对计算速度有重要的影响. 数十年来，人们对归并与排序技术进行了广泛的研究[1~14]，自从 Batcher 提出归并算法[1]，大量串行和并行归并算法相继被提出. 这些归并算法及相应的归并排序算法建立在两路(2-Way)归并的基础上. 2-Way 归并又分为“奇偶归并”与“双调归并”，均为 Batcher 提出[1,2]. Stone 介绍了如何用混洗-交换(Shuffle-Exchange)互联网实现双调归并[3]，Orcutt 介绍如何在并行计算机 ILLIAC IV 上实现双调归并. Thompson 和 Kung 在网格联接(Mesh-Connected)的处理机上实现双调归并，改进了其实现复杂度[4]. Ja′Ja′ 和 Owens 研究了如何以减少硬件的方式实现双调归并[5]. Nakatani 等[6]和 Batcher[2]分别介绍了建立在 2-Way 归并的基础上的 K-Way 双调排序方法，Alnuweiri 和 Kumar 提出了优化实现排序算法的 VLSI 结构[7]. Guan 和 Langston 提出了“时-空”最优化的并行归并和排序技术[8]. 文献[9~19]分别介绍了多处理机系统及分布处理系统上不同的归并与排序方法.

2-Way 归并方法是指把两个已有序的关键字序列归并成 1 个有序的序列(当然从物理上讲，其目的是排列关键字所表示的那些记录或文件，本文一概称为排列这些“关键字”或更简单地称排列这些“数字”). 而上面提到的并行多路归并算法，依然是建立在 2-Way 归并的基础上. 这就是说，算法所处理的输入源是 2 个有序的序列，其输出是 1 个有序序列. 如果有多个有序序列需要处理，比如 N 个，则算法采用递归的方法，以 2-Way 归并方法把 N 个序列归并为 $[N/2]$ 个，再以 2-Way 归并方法归并为 $[N/4]$ 个，…，直至最后再以 2-Way 归并方法把 2 个序列归并为 1 个. 需要注意的是，文献中有的地方也称“多路归并”、或“K-Way”，这些算法与本文将探讨的 K-Way 归并思路完全不同，它们的思路是对 2-Way 的递归应用，而本文所提的 K-Way 归并则完全不是建立在对 2-Way 归并递归调用的基础上的. 比如，Batcher 在文献[2]中提出的并行归并 $n \times k$ 方法，如果把输入看成是 $n \times k$ 的矩阵，则该方法先用 n 个 K-

1996-05-24 收稿，1996-07-10 收修改稿
* 国家“八六三”高技术计划资助项目

Way 排序器以纵向方式排序该矩阵的每 1 列(k 个关键字,各列可并行进行),再用 k 个 n-Way 排序器以横向的方式排序矩阵的每 1 行(n 个关键字,各行可并行进行). 结果便得到 1 个有序序列. 而算法的输入($n\times k$ 个关键字)须是双调的或递升-递降的.

那么,可否找到 1 种归并方法,当多个有序序列,比如 $K(K\geqslant 2)$个,欲归并成 1 个有序序列时,我们不是基于 2-Way 归并方法而逐次进行两两归并,而是"直接"把这 k 个有序序列归并成 1 个有序序列? 本文将提出一种倾斜与振荡法多路归并算法(Sloping-and-Shaking K-Way Merging)对上述问题给出当 K 为任何素数时的肯定的答案.

1 倾斜与振荡法多路归并算法(SS-M$_k$)

倾斜与振荡法 K-Way 归并算法把 K 个有序序列归并为 1 个有序序列.

我们将先给出倾斜与振荡法 K-Way 算法的形式描述. 为了易于理解,我们还将结合 1 个实例,给出算法的较为直观的解释.

设待归并的 K 个有序序列中每个序列含 m 个数,我们用 1 个 $m\times k$ 的矩阵 A 表示待排序的 $m\times k$ 个数. 令 $0\leqslant i\leqslant m-1, 0\leqslant j\leqslant k-1$, A 的第 j 列表示第 j 个有序序列,为不降序列,即任取 $a_{x,j}, a_{y,j}$,如果 $0\leqslant x\leqslant y\leqslant m-1$,则有 $a_{x,j}\leqslant a_{y,j}$. 算法的目标是,以归并的方式把数组 A 的各元素排列成以行为主排列而非降有序序列,即对任意 $i, j, 1\leqslant i\leqslant m-1, 1\leqslant j\leqslant k-1$,均有 $a_{i,j-1}\leqslant a_{i,j}$,并且 $a_{i-1,k-1}\leqslant a_{i,0}$.

下面我们再引入算法描述中将用到的几个记号.

该算法每对矩阵 A 做过 1 遍变换,A 的各元素仍放在 A 中,只是各元素的相对位置做了变化. 我们用 $A^{(t)}$ 代表矩阵 A 在第 t 遍结束时的状态. 从而,$A^{(0)}$ 即是待变换的矩阵.

该算法每步对 A 中以某种规律组织起来的元素做排序. 我们以 $A(i, j, u, v, c)$ 表示 A 中如下元素中的集合:

$$A(i,j,u,v,c)=\{a_{i,j}, a_{i-u,j+v}, \cdots, a_{i-cu,j+cv}\}.$$

即 i, j 为该集合起始元素的行、列号,u, v 为行、列跳变的值,$0\leqslant i-cu, j+cv\leqslant k-1$, c 表示集合元素个数减 1,我们可称之为元素个数计数器.

我们用 SORT 表示 1 个排序操作,$A^{(t+1)}(i, j, u, v, c):=\mathrm{SORT}(A^{(t)}(i, j, u, v, c))$表示对 $\{a_{i,j}^{(t)}, a_{i-u,j+v}^{(t)}, \cdots, a_{i-cu,j+cv}^{(t)}\}$ 进行排序,使得 $(a_{i,j}^{(t+1)}, a_{i-u,j-v}^{(t+1)}, \cdots, a_{i-cu,j+cv}^{(t+1)})$ 成为 1 个非升序列. 算法要用到的 1 个常数(当 m 与 k 确定时),我们定义如下:

$$r=1+\lceil \log_2(m/k)\rceil.$$

在下面算法描述中,"FOR i IN [X, Y] DO"是指对于所有 $X\leqslant i\leqslant Y$ 的 i 并行执行;而"FOR $i:=X$ TO Y DO"是对 i 串行执行. 下面是倾斜与振荡法多路归并算法 SS-M$_k$(Sloping-and-Shaking Multiway Merging)的形式描述.

Algorithm SS-M$_k$;/ * Sloping-and-Shaking K-Way 归并算法 M$_k$ * /

Input: k 个有序序列,每个序列长度为 m,即数组 $A[0:m-1, 0:k-1]$,其每列为 1 个长度为 m 的非降序列.

Output: 1 个长度为 $m\times k$ 的有序序列,即整个数组 $A[0:m-1, 0:k-1]$被组织成了 1 个其元素以行为主排列而非降的序列.

Merging Procedure:

```
BEGIN
1. FOR i IN [0, m-1] DO
     A^(1)(i,0,0,1,k-1):=SORT(A^(0)(i,0,0,1,k-1));
     /*并行对每1行进行排序*/.
2. FOR t:=1 TO r DO /*"斜线"排序*/
3. BEGIN
4.     FOR i IN [0, m-1] DO
5.         {c:=min([i*2^(r-t)], k-1);
6.         A^(t+1)(i,0,2^(r-t),1,c):=SORT(A^(t)(i,0,2^(r-t),1,c));}
7.     FOR i IN [m-2^(r-t), m-1]) ∧ (j IN [1, k-2]) DO
8.         {c:min(⌊i/2^(r-t)⌋, k-1-j)
9.         A^(t+1)(i,j,2^(r-t),1,c):=SORT(A^(t)(i,j,2^(r-t),1,c));}
10. END;
11. FOR v:=2 TO k-1 DO
12. BEGIN
13.     FOR (j IN [0,min(v-1,k-v-1)] ∧ (i IN [0, m-1]) DO
14.     {c:=min(i,⌊(k-1-j)/v⌋);
15.     A^(r+v)(i,j,1,v,c):=SORT(A^(r+v-1)(i,j,1,v,c));}
16. IF v≤k-v-1
17. THEN
18.     FOR j IN[v, k-v-1] DO
19.         {c:=min(m-1,[(k-j-1)/v])
20.         A^(r+v)(m-1,j,1,v,c):=SORT(A^(r+v-1)(m-1,j,1,v,c));}
21. END;
END;
```

由上述算法描述我们可以看出，算法须串行执行的部分主要由 2 个循环构成，(1)由第 2 行 FOR 语句表达的循环，(2)是由第 11 行 FOR 语句表达的循环. 循环中的操作是可并行执行的对 A 的某些元素集合的排序. 由元素个数计数器 c 的表达式可以看出，c 永远小于 k. 从而算法中的主要计算可由 K-Key 排序器实现. 令每个 K-Key 排序器所需的时间为 t_k. 这里，K-Key 排序器的功能是对 K 个数并行排序.

该算法的时间复杂度为

$$(1 + r + k - 2) \times t_k = (k + r - 1)t_k = (k + \lceil \log_2(m/k) \rceil)t_k,$$

这样，算法 SS-M$_k$ 对 N 个数(即 $N = mk$)的 K-Way 归并时间即为

$$(k + \lceil \log_2(N/k^2) \rceil)t_k.$$

$k = 2$ 时，2-Way 归并时间为 $\lceil \log_2 N \rceil t_2$，这就是 Batcher 的奇偶归并和 BITONIC[1] 归并的时间，即，2-Way 最小的并行归并时间.

下节我们将证明，算法的输出 A^{r+k-1} 确实是 1 个以行为主排列而非降的有序序列.

为了易于理解，我们先对算法 SS-M$_k$ 的思路做一个直观的解释.

算法的输入相当于对矩阵 A 已经在各列的范围内"纵向"地排好次序. 而算法中第1行相应于在行的范围内"横向"地 c 个元素进行斜向排序. 算法的主体部分(第2～21行)是以不同斜度的各斜线所"串起来" c 个元素进行"斜向"的排序: 首先是"斜度"很大, $2^{r-1}/1$, 逐步放平, $2^{r-2}/1, 2^{r-3}/1, \cdots, 1/1, 1/2, \cdots, 1/(k-1)$. 这一过程如图1的(a)～(d)所示. 这相当于以不同的倾斜度"振荡"待排序的数组, 以使"重量小"的元素向 A 的左上部漂, 而"重量大"的元素 A 的右下部沉. 最轻的最先漂到左上角, 最重的最先沉到右下角.

我们以1个例子说明上述过程. 本例中 $k=3, m=6$, 从而 $r=1+\lceil \log_2 6/3 \rceil = 2$. 在算法执行过程中 A 的变化过程如图1的(a)～(e)所示. 图中所示各斜线即表示 $A^{(t)}(i,j,u,v,c)$. 比如(b)中, $A^{(1)}(2,0,2^1,1,1)$含2个元素, $\{a^{(1)}_{2,0}, a^{(1)}_{0,1}\}$ 而 $A^{(1)}(5,0,2^1,1,2)$含3个元素 $\{a^{(1)}_{5,0}, a^{(1)}_{3,1}, a^{(1)}_{1,2}\}$. 我们可以看到, 算法结束后 $A^{(r+k-1)}(=A^{(4)})$确实变成1个以行为主排列的有序序列.

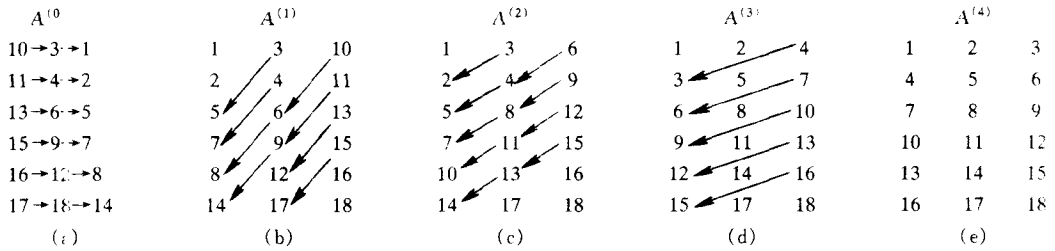

图1 3-Way归并过程

2 算法 SS-M$_k$ 的正确性

设 $X=(X_1, X_2, \cdots, X_m)$, $Y=(Y_1, Y_2, \cdots, Y_n)$为2个非降序列; 设 $X'=(X'_1, X'_2, \cdots, X'_m)$及 $Y'=(Y'_1, Y'_2, \cdots, Y'_n)$分别为 X 和 Y 的任意排列. 对于这两组数, 我们证明下述定理, 我们称之为"关系稳定性定理"(定理1,2及推理1,2). 下面定理中, "θ"表示"大于或等于"或"小于或等于"关系.

定理1 当 $m=n$ 时, 如 $x'_p \theta y'_p, 1 \leqslant p \leqslant n$, 则 $x_p \theta y_p, 1 \leqslant p \leqslant n$.

证 若 θ 为"小于或等于"关系, 由于 $X=(X_1, X_2, \cdots, X_n)$是非降的, 故知 X' 集合中至多有 p 个元素小于或等于 x_p, 因此 Y' 集合至少有 $n-p+1$ 个元素大于或等于 x_p. 又因为 Y 也是非降的, 故知必有 $x_p \leqslant y_p$. 对于 θ 为"大于或等于"情况可类似证得. 证毕.

推论1 当 $m=n$ 时, 如$(x'_p \theta y'_p), 1 \leqslant p \leqslant n$ 及$(x'_p \theta y'_{p-1}), 2 \leqslant p \leqslant n$, 则$(x_p \theta y_p), 1 \leqslant p \leqslant n$ 及$(x_p \theta y_{p-1}), 2 \leqslant p \leqslant n$.

推论2 当 $m=n$ 时, 如$(x'_p \theta y'_{p-1}) \wedge (x'_p \theta y'_p) \wedge (x'_p \theta y'_{p+1}), 2 \leqslant p \leqslant n-1$, $(x'_n \theta y'_n) \wedge (x'_n \theta y'_{n-1}) \wedge (x'_1 \theta y'_1) \wedge (x'_1 \theta y'_2)$, 则有 $x_p \theta y_p, 1 \leqslant p \leqslant n$ 及 $x_p \theta y_{p-1}, 2 \leqslant p \leqslant n$.

定理2 当 $m=n-1$ 时, 如$(x'_p \theta y'_p) \wedge (x'_p \theta y'_{p+1}), 1 \leqslant p \leqslant m$, 则$(x_p \theta y_p) \wedge (x_p \theta y_{p+1}), 1 \leqslant p \leqslant m$.

定理3 算法 SS-M$_k$ 执行结束后, 矩阵 A 确实变成了以行为主排列的非降的有序序列, 即矩阵中的元素 $A^{(r+k-1)}(i,j)$满足下列条件:

(1)任取 $i,j, 0 \leqslant i \leqslant m-1, 1 \leqslant j \leqslant k-1$, 均有 $A^{(r+k-1)}(i,j-1) \leqslant A^{(r+k-1)}(i,j)$;

(2)任取 $i, 1\leqslant i\leqslant m-1$，均有 $A^{(r+k-1)}(i-1,k-1)\leqslant A^{(r+k-1)}(i,0)$.

证 (i)我们先证明算法第 2～10 行循环结束后矩阵的每行是非降的有序序列.

算法的第 1 行语句是对矩阵 A 的各行独立地进行排序，显然 $A(1)$各行均为非降序列. 算法的第 2 行到第 10 行的循环是每次对倾斜率为 2^{r-t}的各直线上的各点并行排序(即同一条直线上的两相邻点间行号差为 2^{r-t}，而列号差为 1). 我们证明该循环结束后矩阵 $A^{(r-1)}$的各行仍为非降序列. 考虑每次循环中 $A(i,j-1)$及 $A(i,j)$所属的两相邻斜线，它们可能的关系如图 2(a)～(d)所示. 图 2(e)给出 $m=10, k=5$ 的矩阵，在 $t=1$ 时的各斜线.

比如过 $A(3,1)$及 $A(3,2)$的两斜线相应于图 2(a)表示的关系，过 $A(7,1)$及 $A(7,2)$的两斜线相应于图 2(c)表示的关系.

当 $t=1$ 时，可以证明，经过 $A(i,j-1)$和 $A(i,j)$的两相邻斜线不会有图 2(d)所示的关系(见附录中引理 A1).

对于如图 2(a)所示的两斜线，把 $A(i,j-1)$所在的直线上各点视作 X'，而把 $A(i,j)$所在的斜线视作 Y'，则 $A(i,j-1)$及 $A(i,j)$相应于 x'_p 和 y'_p(斜线排序前)，x_p 和 y_p(斜线排序后). 由于斜线排序前各行已是非降的，即 $A^{(1)}(i,j-1)\leqslant A^{(1)}(i,j)$故这里 X', Y'满足定理 2 条件，从而知该遍斜线排序后有 $A^{(2)}(i,j-1)\leqslant A^{(2)}(i,j)$.

类似于上述论证，可知处于图 2(b)和(c)所示的两相邻斜线上的 $A(i,j)$与 $A(i,j-1)$也

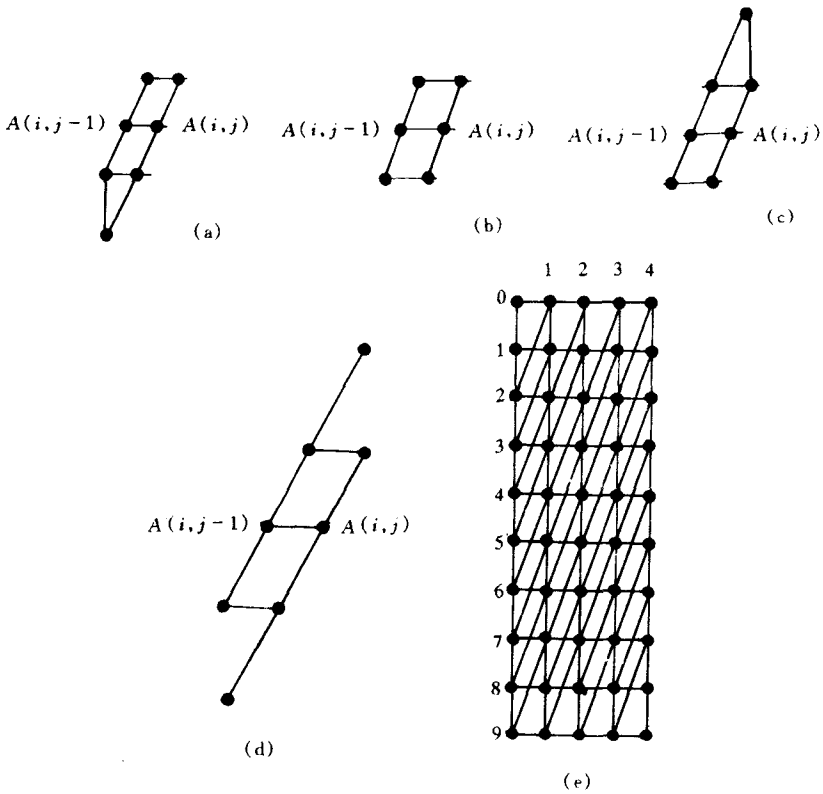

图 2

将形成 $A^{(2)}(i,j-1)\leqslant A^{(2)}(i,j)$的关系.

相似地,可以证明1遍斜线排序后各列仍保持为非降有序序列,即 $A^{(2)}(i-1,j)\leqslant A^{(2)}(i,j)$.

自然地,我们有 $A^{(2)}(i-2^{r-t},j-1)\leqslant A^{(2)}(i,j)$,这是因为这样的2点是同一斜线上的相邻点.

设 $t=c$ 循环后,我们有关系式(1):

$$\left.\begin{aligned}&A^{(1+c)}(i,j-1)\leqslant A^{(1+c)}(i,j)\ (0\leqslant i\leqslant m-1,1\leqslant j\leqslant k-1),\\&A^{(1+c)}(i-1,j)\leqslant A^{(1+c)}(i,j)\ (1\leqslant i\leqslant m-1,0\leqslant j\leqslant k-1),\\&A^{(1+c)}(i-2^{r-c},j+1)\leqslant A^{(1+c)}(i,j)\ (i\geqslant 2^{r-c},j\leqslant k-2).\end{aligned}\right\}\quad(1)$$

我们证明 $t=c+1\leqslant r$ 循环后(2)式成立,

$$\left.\begin{aligned}&A^{(2+c)}(i,j-1)\leqslant A^{(2+c)}(i,j)\ (0\leqslant i\leqslant m-1,1\leqslant j\leqslant k-1),\\&A^{(2+c)}(i-1,j)\leqslant A^{(2+c)}(i,j)\ (1\leqslant i\leqslant m-1,0\leqslant j\leqslant k-1),\\&A^{(2+c)}(i-2^{r-(c+1)},j+1)\leqslant A^{(2+c)}(i,j)\ (i\geqslant 2^{r-c},j\leqslant k-2).\end{aligned}\right\}\quad(2)$$

式中 $A^{(2+c)}(i-2^{r-(c+1)},j+1)\leqslant A^{(2+c)}(i,j)$的成立是自然的,因为它们是该步进行斜线排序的同一斜线上的两个相邻的点.

任取一行 i,考虑通过行上两相邻点 $A(i,j-1)$和 $A(i,j)$的两条斜线,它们仍可用图2所示的关系表示(只是斜率比前一步"平缓"了一半).

对于处于图2(a),(b)和(c)上两斜线上的 $A(i,j-1)$和 $A(i,j)$类似于 $t=1$ 时可论证将得到 $A^{(2+c)}(i,j-1)\leqslant A^{(2+c)}(i,j)$. 下面考虑图(2)(d)的情况.

由关系式(1)知,在图3中 $A^{(1+c)}(i,j-1)\leqslant A^{(1+c)}(i,j)$, $A^{(1+c)}(i,j-1)\leqslant A^{(1+c)}(i+2^{r-(c+1)},j-1)$,及 $A^{(1+c)}(i,j-1)\leqslant A^{(1+c)}(i+2^{r-c},j-2)$.

可见此时把 $A(i,j-1)$所在的斜线及 $A(i,j)$所在的斜线视作 X'及 Y',其上各点的关系满足推论2的条件,(若 $A(i,j-1)$为 x'_p,则 $A(i,j)$为 y'_{p-1}). 从而知该步斜线排序后有 $A^{(2+c)}(i,j-1)\leqslant A^{(2+c)}(i,j)$.

对于列, $A^{(2+c)}(i-1,j)\leqslant A^{(2+c)}(i,j)$可类似证得.

可见算法第2行到第10行的循环结束后矩阵 $A^{(r+1)}$的各行(及各列)为非降的有序序列.

(i)下面证明算法第11行到第21行的循环结束后定理3所给的结果得到满足.

根据前面的论证算法第11行的循环开始前我们有 $A^{(r+1)}(i-1,k-1)\leqslant A^{(r+1)}(i,k-2)$,及 $A^{(r+1)}(i,k-3)\leqslant A^{(r+1)}(i,k-2)$,如图4所示. 该循环开始时 $v=2$,此时对 $A(i-$

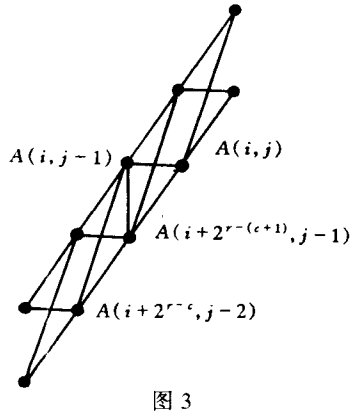

图3

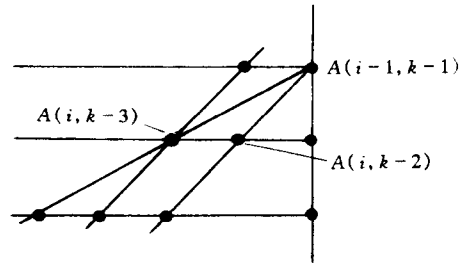

图4

$1, k-1)$及 $A(i, k-3)$连接的斜线上各点进行排序. 可以证明(见附录引理 2),此时过同一行上任意两相邻点的 2 条相邻的斜线不存在图 2(d)所示的关系. 不难推得,这样的 2 条斜线相应于图 2(a)、(b)、(c)的关系,而满足定理 1、推论 1 及定理 2 的条件,从而得到:任取 i,均有

$$\left.\begin{aligned} &A^{(r+2)}(i, j-1) \leqslant A^{(r+2)}(i, j), \quad 0 \leqslant i \leqslant m-1, 1 \leqslant j \leqslant k-1, \\ &A^{(r+2)}(i-1, j) \leqslant A^{(r+2)}(i, j), \quad 1 \leqslant i \leqslant m-1, \\ &A^{(r+2)}(i-1, k-1) \leqslant A^{(r+2)}(i, k-3), \quad 1 \leqslant i \leqslant m-1. \end{aligned}\right\} \tag{3}$$

类似地,可知列的非降特性得到保持. 以此类推(用归纳法),可知该算法结束时($v = k-1$),任取 $i, 1 \leqslant i \leqslant m-1$,有 $A^{(r+k-1)}(i-1, k-1) \leqslant A^{(r+k-1)}(i, 0)$,及 $A^{(r+k-1)}(i, j-1) \leqslant A^{(r+k-1)}(i, j)$.

这意味着矩阵 A 已变成了一个以行为主排列的非降的有序序列. 证毕.

3　基于 SS-M_k 多路归并算法的多路排序算法

基于上节描述的倾斜与振荡法 K-Way 归并算法,我们给出 1 种 K-Way 排序算法.

设给出 1 组任意次序的 N 个数,其中 $N = K^p$, k 是任意 1 个素数,而 p 是任意正整数. 该算法重复进行 p 次,每次由 k^{p-i} 个并行的 SS-M_k 多路归并过程组成,而每个多路归并过程把 k 个各由 k^i 个数组成的序列归并为 1 个非降的有序序列. 下面是该方法,倾斜与振荡法多路排序的一种形式描述.

Algorithm SS-S_k

```
BEGIN
  FOR i:=1 TO p DO
    FOR j IN [0, k^(p-i)-1] DO
      a_j^(i) := SS-M_k{a_(j·k^i)^(i-1), a_(j·k^i+1)^(i-1), ···, a_(j·k^i+k-1)^(i-1)};
END;
```

上述描述中 $a_j^{(i)}$ 代表 1 个长度为 k^i 的非降有序序列, SS-$M_k\{a_{j\cdot k^i}^{(i-1)}, a_{j\cdot k^i+1}^{(i-1)}, \cdots, a_{j\cdot k^i+k-1}^{(i-1)}\}$ 是指对花括中的各序列进行 K-Way 归并所得的结果.

下面分析上述 K-Way 排序算法的时间复杂度.

算法 SS-S_k 由 $\log_k N$ 步 K-Way 归并过程组成,而第 i 步每一个 K-Way 归并过程要把 k 个长度为 k^{i-1} 的序列归并为一个长度为 k^i 的序列.

从而由第 2 节中分析 SS-M_k K-Way 归并过程的时间复杂度知,第 i 步($i \geqslant 2$) K-Way 归并过程所需的时间为 $(k + \lceil \log_2 k^{i-2} \rceil) t_k$,从而得到,算法 SS-$S_k$ 的时间复杂度 $T(S_k)$ 为

$$T(S_k) = (1 + (p-1)k + ((p-2)(p-1)/2)(\lceil \log_2 k \rceil) t_k, \text{其中 } p = \log_k N.$$

由第 1 节算法 SS-M_k 的描述我们知道,算法的主要操作是对 k 个数的排序,它既可能以专用硬件以网络形式实现,也可以在向量或并行处理机上实现,或者以串行的方式采用不同的排序方法(比如快速排序)实现. 不同的实现方式将决定 t_k 的值. 本文不再对此进行分析.

4　结论

本文提出了 1 个多路归并算法:倾斜与振荡法 SS-M_k. 该算法可把任意素数路有序序列

归并为1个有序序列. 它不是递归地应用两路归并过程而使有序序列每次成倍递减,从而它打破了对两路递归过程的依赖. 这是它区别于已有的多路递归算法的最主要特点,是1种完全新型的归并方式. 由于2同样是个素数,从而两路奇偶归并可以看成是本算法的1个特例.

本文所提出的归并算法以及基于 SS-M_k K-Way 归并的 SS-S_k K-Way 排序算法既可用于多处理机或并行处理机系统,也可以用专用硬件设计 K-Way 归并网络实现,此种网络中的主要部件仅为1种有 K 个输入/输出端的 K-Way 比较器. 与两路归并相比本文所提出的多路归并的优越性在于选择适当的 K,多路归并网络可以比两路归并网络速度快. 比如虽然上面给出的 $T(S_k)$ 不是 K 的简单的函数,仍不难得知采用专门设计的3-比较器及3-Way 归并排序网络的速度高于采用2-比较器及2-Way 归并排序网络. 对本算法具体实现方法的分析和网络设计方法及其与传统的两路归并算法的比较,将在另一专题文章中论述,由于篇幅所限,本文未包含该项内容.

参 考 文 献

1 Batcher K E. Sorting networks and their applications. In: 1968 Spring Joint Comput Conf AFIPS Proc, Washington D C, 1968, 32: 307~314

2 Batcher K E. On bitonic sorting networks. Proc of 1990 International Conf On Parallel Processing, 1990. I-376~379

3 Stone H S. Parallel processing with the perfect shuffle. IEEE Trans on Computer, 1971, C-20(2): 153~161

4 Thompson C D, Kung H T. Sorting on a mesh-connected parallel computer. Communications of ACM, 1977, 20(4): 263~271

5 Ja' Ja' J, Owens M. VLSI sorting with reduced hardware. IEEE Trans on Computer, 1984, C-33(7): 668~671

6 Nakatani T, Huang S T, Arden B W *et al*. K-way bitonic sort. IEEE Trans on Computer, 1989, C-38(2): 283~288

7 Alnuweiri H M, Kumar V K. Optimal VLSI sorting with reduced number of processors. IEEE Trans on Computer, 1991, C-40(1): 105~110

8 Guan X, Langston M A. Time-space optimal parallel merging and sorting. IEEE Trans on Computer, 1991, C-40(5): 596~602

9 Knuth D E. The Art of Computer Programming. Vol 3. Sorting and Searching. Addison-Wesley Publishing Company, 1973

10 Beck M, Bitton D, Wilkinsin W K. Sorting large files on a backend multiprocessor. IEEE Trans on Computers, 1988, C-37(7): 769~778

11 Rotem D, Santoro N, Sidney J B. Distributed sorting. IEEE Trans on Computer, 1985, C-34(4): 372~375

12 Iyer B, Ricard G, Varman P. Percentile finding algorithm for multiple sorted runs. In: Proc 15th International Conference on Very Large Databases, 1989. 135~144

13 Varman P J, Iyer B R, Scheufler S D. A multiprocessor algorithm for merging multiple sorted lists. In: Proc of 1990 International Conference on Parallel Processing, 1990. III-22~26

14 Leighton T. Tight bound on the complexity of parallel sorting. IEEE Trans on Computer, 1985, C-34(4): 344~354

15 Lee D, Batcher K E. Multiway merge sorting network. IEEE Trans on Parallel and Distributed Systems, 1995, 6(2): 211~215.

16 Parker B, Parberry Ian. Constructing sorting networks from k-sorters. Information Processing Letters, 1989, 33: 157~162

17 Liszka K L, Batcher K E. A modular merge sorting network. In: Proc Frontiers' 92, The fourth Symposium on the Frontiers of Massively Parallel Computation, 1992. 164~169

18 Liszka K L, Batcher K E. A generalized bitonic sorting network. In: Proc International conference on Parallel Processing, 1993. I-105~I-108

19　陈国良．并行算法：排序和选择．合肥：中国科技大学出版社，1990

附录 A

引理 A1　算法 SS-M_k 中，当 $t=1$ 时，

过矩阵 A 任意一行 i 的两相邻点 $A(i,j-1)$ 及 $A(i,j)$ 的两斜线至少有一端的两元素是处于同一行上的两相邻元素．

证　如图 A-1 所示假设 f，$f=A(f_1,f_2)$ 点存在，而在通过 $A(i,j)$ 的斜线上另一端点 e 的元素下标为 (e_1,e_2)（即 $e=A(e_1,e_2)$）．

(1) 若 e 是第 0 列上的点，即 $e_2=0$ 时，这时如 f 是第 $k-1$ 列，即矩阵最后 1 列上的点，则 $e_1=f_1+2^{r-1}\times(k-1)$　如果此时通过 $A(i,j)$ 的斜线上还要有一点 g，$A(g_1,0)$，则 g 点的行号 g_1 应为：

$g_1=e_1+2^{r-1}=f_1+2^{r-1}\times k=f_1+\lceil m/k\rceil\times k\geqslant m$；我们知道，$A$ 矩阵是 $m\times k$ 的矩阵，最大行号仅为 $m-1$，故上述情况不可能存在．

若 f 不是第 $k-1$ 列上的点，易知 h 的斜上方必然还有一点 $h'=A(f_1,f_2+1)$ 处在通过 $A(i,j)$ 的斜线上

(2) 若 e 不是第 0 列上的点，与上述方法类似，可以证明或者 g 不存在，或者 e 的斜下方还有一点 $e'=A(g_1,g_2-1)$ 处在通过 $A(i,j-1)$ 的斜线上．证毕．

引理 A2　算法 SS-M_k 第 11 行的循环中，过矩阵 A 任意 1 行 i 的两相邻点 $A(i,j-1)$ 及 $A(i,j)$ 的两斜线，至少有一端的两元素是处于同 1 行上的两相邻元素．

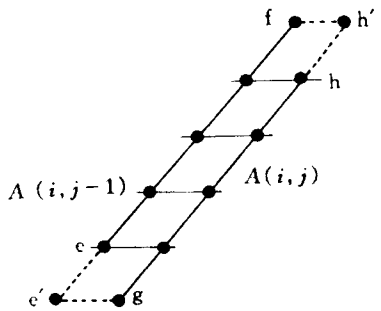

图 A-1

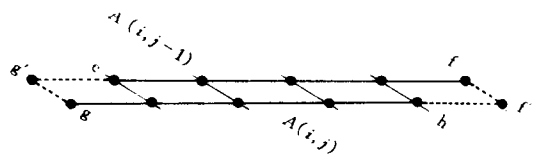

图 A-2

证　如图 A-2 所示，

(1) 如果 $g=A(g_1,g_2)$ 是第 0 列上的点，即 $g=A(g_1,0)$，若此时 $f=A(f_1,f_2)$ 为第 $k-1$ 列上的点，设斜线 ef（或 gh）上包含 x 个点，则点 g 和 f 的列号差为 $(x-1)\times v+(v-1)=k-1$，从而 $x\times v=k$ 这是不可能的，因为 k 是素数．若 f 为是第 $k-1$ 列上的点，则知 f 点右方必有一点 $f'=A(f_1,f_2+1)$ 处在通过 $A(i,j)$ 的斜线上．

(2) 若 g 不是经 0 列上的点，则知 g 左方必有一点 $g'=A(g_1,g_2-1)$ 处在通过 $A(i,j-1)$ 的斜线上．证毕．