SIAM J. COMPUT.
Vol. 1, No. 1, March 1972

A SIMPLE ALGORITHM FOR MERGING TWO DISJOINT LINEARLY ORDERED SETS*

F. K. HWANG AND S. LIN†

Abstract. In this paper we present a new algorithm for merging two linearly ordered sets which requires substantially fewer comparisons than the commonly used tape merge or binary insertion algorithms. Bounds on the difference between the number of comparisons required by this algorithm and the information theory lower bounds are derived. Results from a computer implementation of the new algorithm are given and compared with a similar implementation of the tape merge algorithm.

Key words. algorithms, merging

1. Introduction. Suppose we are given two disjoint linearly ordered subsets A and B of a linearly ordered set S, say

$$A = \{a_1 < a_2 < \cdots < a_m\},$$
$$B = \{b_1 < b_2 < \cdots < b_n\}.$$

The problem is to determine the linear ordering of their union (i.e., to merge A and B) by means of a sequence of pairwise comparisons between an element of A and an element of B. Given any algorithm s to solve this problem, we are interested in the maximum number of comparisons, $K_s(m, n)$, required under all possible orderings of $A \cup B$. An algorithm s is said to be M-optimal if $K_s(m, n) = K(m, n)$, where $K(m, n) = \min_x K_x(m, n)$. In this paper, we give a simple algorithm for solving this problem, called the generalized binary algorithm g, and derive some bounds for $K_g(m, n) - K(m, n)$ which are substantially better than two other known algorithms.

2. Some preliminary discussions and results. Let the cardinality of A and B be m and n respectively. We assume $m \leqq n$. Let $\mathscr{D}_0$ be the set of all possible orderings of $A \cup B$ and $\mathscr{D}_k$ be the subset of $\mathscr{D}_0$ consistent with the results of the first k comparisons we have made thus far. It is clear that, after making the ith comparison, $i = 1, 2, \cdots, k$, one of the two possible outcomes must have $|\mathscr{D}_i| \geqq \frac{1}{2}|\mathscr{D}_{i-1}|$ and that merging is achieved if and only if $\mathscr{D}_k$ contains exactly one element. Since $\mathscr{D}_0$ has $\binom{m+n}{n}$ elements, or as we say, data points, we must have, for any algorithm s,[1]

$$K_s(m, n) \geqq \left\lceil \log_2 \binom{m+n}{m} \right\rceil \equiv I(m, n).$$

$I(m, n)$ is usually called the information theory bound.

For $m = 1$, the binary insertion algorithm is optimal and hence

$$K(1, n) = I(1, n) = \lceil \log_2 (n + 1) \rceil. \tag{1}$$

* Received by the editors August 30, 1971.

† Bell Laboratories, Murray Hill, New Jersey 07974.

[1] As usual, we let $\lceil x \rceil$ denote the smallest integer $\geqq x$ and $\lfloor x \rfloor$ the largest integer $\leqq x$.

In a recent paper [1], the authors constructed an M-optimal algorithm for $m = 2$ and thereby determined the values of $K(2, n)$. It can also be shown that [2]

$$K(m, n) = m + n - 1 \quad \text{for } 3 < m \leqq n \leqq m + 3 \tag{2}$$

and

$$K(m, 2m) \leqq 3m - 2 \quad \text{for } m \geqq 3. \tag{3}$$

The determination of $K(m, n)$ for $m \geqq 3$ appears to be a very difficult problem.

3. Two existing algorithms. For the purpose of comparing with the generalized binary algorithm to be presented in the next section, we mention two existing algorithms.

I. *The "tape merge" algorithm t.* The "tape merge" algorithm is the commonly used procedure to merge two tapes or lists of sorted items. It can be described by the following steps (details of storing and stop conditions are omitted):

TM1. Compare a_m with b_n.

TM2. If $a_m < b_n$, set $n = n - 1$ and go to TM1.

TM3. If $a_m > b_n$, set $m = m - 1$ and go to TM1.

It can be easily shown that

$$K_t(m, n) = m + n - 1$$

and hence the "tape merge" algorithm is M-optimal for $n \leqq m + 3$ [2].

II. *The "simple binary" algorithm s.* The "simple binary" algorithm can be described by the following steps:

SB1. Merge a_m into B by the binary search procedure.

SB2. Pull out a_m and elements of $B > a_m$. (These are already in order and larger than the rest of the elements of $A \cup B$.) Set $m = m - 1$ and redefine m and n. (The new $n \geqq$ new m.) Go back to SB1.

It is clear that under the worst possible outcome, a_m is always larger than b_n and hence no element of B is discarded. Therefore,

$$K_s(m, n) = m\lceil \log_2 (n + 1)\rceil.$$

For $m = 1$, we have

$$K_s(m, n) = K(m, n).$$

However, we shall show in the next section that

$$K_s(m, n) > K(m, n) \quad \text{for } m > 2.$$

The distinctive feature of these two algorithms is their simplicity, although in general, they are quite inefficient in the sense that both $K_t(m, n) - K(m, n)$ or $K_s(m, n) - K(m, n)$ can be very large. In the next section, we shall present an algorithm which matches the two abovementioned algorithms in simplicity and yet improves a great deal on their efficiency.

4. The generalized binary algorithm g. For the sake of simplicity, we shall assume that whenever we are required to merge two disjoint linearly ordered sets with cardinalities x and y respectively, n will always refer to $\max(x, y)$ and m, to $\min(x, y)$, so that $n \geqq m$.

The generalized binary algorithm may now be described as follows (again, details of storage and stop criteria are omitted):

GB1. Let $\alpha = \lfloor \log_2 (n/m) \rfloor$ and $x = n - 2^\alpha + 1$.

GB2. Compare a_m with b_x. If $a_m < b_x$, pull out the set of all elements in $B \geqq b_x$, say C. We are then left with the problem of merging two disjoint sets A and $B - C$. Redefine m and n and go back to GB1. (Note that $B - C$ has $n - 2^\alpha$ elements and we need to interchange the role of m and n if and only if $n = m$.)

GB3. If $a_m > b_x$, merge a_m into the set $C - b_x$ by the simple binary algorithm. Note that $C - b_x$ has exactly $2^\alpha - 1$ elements and a_m can be merged into the set in exactly α more comparisons. Pull out a_m and the set D of all elements in $B > a_m$. We are then left with the problem of merging the set $A - a_m$ with the set $B - D$. Redefine m and n and go back to GB1.

For this algorithm g, $K_g(m, n)$ is given by the following theorem.

THEOREM 1. *Let* $\alpha = \lfloor \log_2 (n/m) \rfloor$. *Write* $n = 2^\alpha m + 2^\alpha p + \theta$, *where* p *and* θ *are uniquely determined nonnegative integers satisfying* $0 \leqq p < m$, $0 \leqq \theta < 2^\alpha$. *Then* $K_g(m, n) = (2 + \alpha)m + p - 1$.

Proof. If $\alpha = 0$, $n = m + p$, and it is clear that the worst possible data forces the algorithm g to be identical with the algorithm t discussed in the previous section.

Hence $K_g(m, n) = K_t(m, n) = m + n - 1 = 2m + p - 1$.

If $m = 1$, p must be zero and $n = 2^\alpha + \theta$. It is clear that $a_m > b_x$ is the worst outcome and hence $K_g(1, n) = K(1, n) = 1 + \alpha$.

We now prove Theorem 1 by induction on $m + n$. Assume the theorem true for all m', n' such that $m' + n' < m + n$, and for all m, n with $\alpha = 0$, or $m = 1$. We prove the theorem true for m, n with $\alpha > 0$ and $m > 1$. The theorem is trivially true for $m + n = 2$.

After making the first comparison of a_m with b_x, we have two possibilities:

(i) $a_m < b_x$, and we are left with the problem of merging two sets with m and $n - 2^\alpha$ elements.

(ii) $a_m > b_x$. After merging a_m into the set $C - b_x$ in α more comparisons, we are left, in the worst case, with the problem of merging two sets with $m - 1$ and n elements. Hence

$$K_g(m, n) = \max[1 + K_g(m, n - 2^\alpha), 1 + \alpha + K_g(m - 1, n)].$$

Now,

$$n - 2^\alpha = \begin{cases} 2^\alpha m + 2^\alpha(p - 1) + \theta & \text{if } p \neq 0, \\ 2^{\alpha-1}m + 2^{\alpha-1}(m - 1) + \theta - 2^{\alpha-1} & \text{if } p = 0 \text{ and } \theta \geqq 2^{\alpha-1}, \\ 2^{\alpha-1}m + 2^{\alpha-1}(m - 2) + \theta & \text{if } p = 0 \text{ and } \theta < 2^{\alpha-1}. \end{cases}$$

Hence by induction,

$$K_g(m, n - 2^\alpha) = \begin{cases} (2 + \alpha)m + (p - 1) - 1 & \text{if } p \neq 0, \\ (1 + \alpha)m + (m - 1) - 1 & \text{if } p = 0 \text{ and } \theta \geqq 2^{\alpha-1}, \\ (1 + \alpha)m + (m - 2) - 1 & \text{if } p = 0 \text{ and } \theta < 2^{\alpha-1}. \end{cases}$$

Similarly,

$$n = \begin{cases} 2^\alpha(m-1) + 2^\alpha(p+1) + \theta & \text{if } p < m-2, \\ 2^{1+\alpha}(m-1) + \theta & \text{if } p = m-2, \\ 2^{1+\alpha}(m-1) + 2\alpha + \theta & \text{if } p = m-1. \end{cases}$$

Hence by induction,

$$K_g(m-1, n) = \begin{cases} (2+\alpha)(m-1) + (p+1) - 1 & \text{if } p < m-2, \\ (3+\alpha)(m-1) - 1 & \text{otherwise}. \end{cases}$$

Therefore,

$$1 + K_g(m, n - 2^\alpha) = \begin{cases} (2+\alpha)m + p - 2 & \text{if } p = 0 \text{ and } \theta < 2^{\alpha-1}, \\ (2+\alpha)m + p - 1 & \text{otherwise}, \end{cases}$$

and

$$1 + \alpha + K_g(m-1, n) = \begin{cases} (2+\alpha)m + p - 2 & \text{if } p = m-1, \\ (2+\alpha)m + p - 1 & \text{otherwise}. \end{cases}$$

Since the conditions $p = 0$ and $p = m - 1$ are mutually exclusive for $m > 1$, we have

$$\begin{aligned} K_g(m, n) &= \max\,[1 + K_g(m, n - 2^\alpha), 1 + \alpha + K_g(m-1, n)] \\ &= (2+\alpha)m + p - 1, \end{aligned}$$

and hence the theorem is proved.

Comparing the general binary algorithm g with the tape merge algorithm t and the simple binary algorithm s, we have

$$\begin{aligned} K_t(m, n) - K_g(m, n) &= (m + n - 1) - [(\alpha + 2)m + p - 1] \\ &= m + 2^\alpha m + 2^\alpha p + \theta - 1 - (\alpha + 2)m - p + 1 \\ &= (2^\alpha - \alpha - 1)m + (2^\alpha - 1)p + \theta. \end{aligned}$$

Hence $K_t(m, n) = K_g(m, n)$ only if $\alpha = 0$, or $\alpha = 1$ and $p = \theta = 0$. Otherwise, $K_t(m, n) - K_g(m, n) = n - (\alpha + 1)m - p > 0$. Similarly,

$$\begin{aligned} K_s(m, n) - K_g(m, n) &= m\lceil \log_2 (n+1) \rceil - [(\alpha + 2)m + p - 1] \\ &\geqq m(\alpha + \lfloor \log_2 (m + p) \rfloor + 1) - [(\alpha + 2)m + p - 1] \\ &= m(\lfloor \log_2 (m + p) - 1 \rfloor - p + 1. \end{aligned}$$

Hence $K_s(m, n) = K_g(m, n)$ only if $m = 1$, or $m = 2$, $p = 1$. Otherwise $K_s(m, n) - K_g(m, n) \geqq m(\lfloor \log_2 (m + p) \rfloor - 1) - p + 1 > 0$.

It is often convenient to refer to a set of numbers $n_l(m, k)$ as the largest n such that $K_l(m, n) \leqq k$. Table 1 gives some of these numbers for the algorithms t, s and g. Also we have for $k = (2 + \alpha)m + p - 1$,

$$n_g(m,k) = 2^\alpha(m + p + 1) - 1,$$
$$n_t(m,k) = (1 + \alpha)m + p,$$
$$n_s(m,k) = 2^{\alpha+2} - 1 \quad \text{provided } 2^{\alpha+2} - 1 \geqq m.$$

TABLE 1

(m, k)	$(2, 4)$	$(2, 24)$	$(4,14)$	$(4,90)$	$(10^3, 10^4)$
$n_g(m, k)$	3	4095	15	$2^{23} - 1$	256,511
$n_t(m, k)$	3	23	11	87	9,001
$n_s(m, k)$	3	2047	7	$2^{22} - 1$	1,023

5. Bounds on $K_g(m, n) - I(m, n)$. Let $n = 2^\alpha m + 2^\alpha p + \theta$ with $0 \leqq p < m$, $0 \leqq \theta < 2^\alpha$, $\alpha \geqq 0$; and $k = K_g(m, n) = (2 + \alpha)m + p - 1$.

THEOREM 2. $K_g(m,n) - I(m,n) \leqq m - 1$.

Proof. We have

$$I(m,n) = \left\lceil \log_2 \binom{m+n}{m} \right\rceil,$$

and

$$\binom{m+n}{m} \geqq \frac{(n+1)^m}{m!} = \frac{(2^\alpha m + 2^\alpha p + \theta + 1)^m}{m!}$$
$$> \frac{[2^\alpha m(1 + p/m)]^m}{m!} = \frac{2^{\alpha m} m^m (1 + p/m)^m}{m!} \geqq 2^{\alpha m + m - 1 + p}$$

since

$$m! \leqq m^m 2^{1-m} \quad \text{and} \quad (1 + p/m)^m \geqq 2^p.$$

Hence

$$\log_2 \binom{m+n}{m} > (\alpha + 1)m + p - 1,$$

and

$$\left\lceil \log_2 \binom{m+n}{m} \right\rceil \geqq (\alpha + 1)m + p.$$

Therefore,

$$K_g(m,n) - I(m,n) \leqq m - 1.$$

COROLLARY 1. *For $m > 1$ and $\theta = 2^\alpha - 1$,*

$$K_g(m,n) - I(m,n) \leqq m - 2.$$

Proof. For $m > 1$ and $\theta = 2^\alpha - 1$, we have

$$\binom{m+n}{m} > \frac{(n+1)^m}{m!} \geqq \frac{[2^\alpha m(1 + (p+1)/m)]^m}{m!}$$

and the proof parallels the proof of Theorem 1.

For larger m, a much sharper bound for $K_g(m,n) - I(m,n)$ can be derived by means of Stirling's formula. First we prove a lemma.

LEMMA 1. *Let $\hat{e} = (1 + m/n)^{n/m}$ and x_m be defined by*

$$(n + x_m)^m = (n+m)(n+m-1)\cdots(n+1).$$

Then

$$x_m \geqq \max\left[1, \frac{\hat{e}}{e}(n+m) - n\right].$$

Proof. It is clear that $x_m \geqq 1$. From Stirling's formula, we have

$$\begin{aligned}(n + x_m)^m &= (n+m)(n+m-1)\cdots(n+1) = \frac{(n+m)!}{n!}\\ &= \frac{\sqrt{2\pi}(n+m)^{n+m+1/2}\, e^{-(n+m)+\theta_1/(12(n+m))}}{\sqrt{2\pi}n^{n+1/2}\, e^{-n+\theta_2/(12n)}} \qquad \left(0 < \begin{matrix}\theta_1\\ \theta_2\end{matrix} < 1\right)\\ &> \sqrt{\frac{n+m}{n}}\left(1 + \frac{m}{n}\right)^n (n+m)^m\, e^{-m-1/(12n)}\\ &> \hat{e}^m (n+m)^m\, e^{-m}\end{aligned}$$

since

$$\sqrt{\frac{m+n}{n}} = \left(1 + \frac{m}{n}\right)^{n/m\cdot m/(2n)} \geqq 2^{1/(2n)} = 4^{1/(4n)} > e^{1/(12n)}.$$

Therefore,

$$x_m > \frac{\hat{e}}{e}(n+m) - n = m\left[\frac{\hat{e}}{e}\left(1 + \frac{n}{m}\right) - \frac{n}{m}\right].$$

Some typical values of x_m/m are given below in Table 2.

TABLE 2

n/m	1	2	10	100	1000
$\hat{e}$	2	2.25	2.594	2.705	2.717
x_m/m	>0.4715	>0.4831	>0.4907	>0.5065	>0.5279

THEOREM 3. *Let* $\varepsilon = (\theta + x_m)/2^\alpha$ *and* $t = \min(p + \varepsilon, m)$. *Then*

$$I(m,n) = \left\lceil \log_2 \binom{m+n}{m} \right\rceil \geqq (1+\alpha)m + \lceil t + q_m \rceil,$$

where

$$q_m = (\log_2 e - 1)m - \frac{\log_2 e}{12m} - \tfrac{1}{2}\log_2(2\pi m) \sim 0.442695m - \frac{0.12}{m} - \tfrac{1}{2}\log_2(2\pi m).$$

Proof.

$$\begin{aligned}\binom{n+m}{m} &= \frac{(n+x_m)^m}{m!} \\ &> \frac{(n+x_m)^m}{\sqrt{2\pi m}\, m^m e^{-m+1/(12m)}} \\ &= \frac{(2^\alpha m + 2^\alpha p + \theta + x_m)^m e^{m-1/(12m)}}{\sqrt{2\pi m}\, m^m} \\ &= \frac{(2^\alpha m)^m (1 + (p+\varepsilon)/m)^m e^{m-1/(12m)}}{\sqrt{2\pi m}\, m^m} \\ &> 2^{\alpha m + t + (\log_2 e)(m - 1/(12m)) - \log_2\sqrt{2\pi m}}.\end{aligned}$$

Therefore

$$I(m,n) = \left\lceil \log_2 \binom{m+n}{m} \right\rceil \geqq (1+\alpha)m + \lceil t + q_m \rceil,$$

which is to be proved.

Since $K(m,n) \geqq I(m,n)$, we have the following corollary.

COROLLARY 2.

$$K_g(m,n) - K(m,n) \leqq K_g(m,n) - I(m,n) \leqq m + p - 1 - \lceil t + q_m \rceil.$$

Table 3 gives some values for

$$q_m = (\log_2 e - 1)m - \frac{\log_2 e}{12m} - \tfrac{1}{2}\log_2(2\pi m).$$

TABLE 3

m	1	2	3	4	5	6	16	1024
q_m	−2.08	−1.0005	−0.832	−0.585	−0.299	0.179	3.65	446.9

Note that $t > p$ and $q_m > -1$ for $m \geqq 3$ so that Corollary 2 implies Theorem 2 for $m \geqq 3$. For $m \geqq 6$, $q_m > 0$ and hence Corollary 2 also implies the conclusion of Corollary 1 regardless of the value of θ. For large m, say $m > 100$, we have $K_g(m, n) - I(m, n) < 0.6m - 1$, and the "best" bound occurs when $\alpha = 0$, $p \approx 0.52m$, $x_m \approx 0.48m$, $q_m \approx 0.44m$ and this gives $K_g(m, n) - I(m, n) \approx 0.08m$.

6. Computational results. In this section, we discuss the storage requirements when the generalized binary algorithm g is implemented by a computer program and compare its running time with a similar program implementing the commonly used tape merge algorithm t. We assume that the sorted lists A_m and B_n to be merged are stored on tapes (or other external devices) if they are too large to be accommodated in core. These can then be read in sequentially in sorted order as needed and the elements of the merged list C_{m+n} written in similar sorted order onto output devices as soon as they are sequentially determined. As can be seen from the description of the algorithm g, for efficient comparison we need the elements a_m and those in B_n from b_x to b_n in core. This requires a storage space of 2^α elements ($\alpha = \lfloor \log_2 (n/m) \rfloor$) which is approximately equal to n/m. In general, this will not be excessive. For example, if $n = 10^7$ and $m = 10^4$, an average of 10^3 elements of B are required to be in core and this ratio will be approximately maintained if the data in B_n and A_m are uniformly distributed in some interval. If n/m becomes too large, a slight modification of the algorithm can be made, say, to compare a_m with b_x, where $x = n - 2^\beta + 1$ for some smaller β, without substantially affecting its efficiency.

Assuming the data in A_m and B_n are uniformly distributed in some interval, the expected number of comparisons $E_t(m, n)$ required by the tape merge algorithm t can be seen to satisfy the following recurrence relation:

$$\text{(R)} \qquad E_t(m, n) = 1 + \frac{m}{m+n} E_t(m-1, n) + \frac{n}{m+n} E_t(m, n-1); \qquad E_t(1, 1) = 1.$$

Solving (R), we have

$$E_t(m, n) = mn\left(\frac{1}{m+1} + \frac{1}{n+1}\right)$$

$$= m + n - \left(\frac{m}{n+1} + \frac{n}{m+1}\right),$$

which is only slightly less than $K_t(m, n) = m + n - 1$.

When n/m is large, as in the case of updating telephone directories or library materials, we see that $E_t(m, n) \approx m + n - n/m$ can be considerably larger than $K_g(m, n) \approx (3 + \lfloor \log_2 (n/m) \rfloor)m$, the maximal number of comparisons required using the algorithm g. Even when the logic involved in making one comparison using the proposed algorithm g is more involved than making one comparison using the tape merge algorithm t, substantial savings in computer time can be achieved. A computer program (FORTRAN, GE-635), implementing both the tape merge algorithm t and the generalized binary algorithm g (hopefully with equal degrees of efficiency), was written to test our assertions on some problems with randomly generated data. The results are presented in Table 4. As can be seen, the saving in time is great when n/m is large.

TABLE 4

m	n	n/m	C_t	C_g	T_t	T_g	T_g/T_t
150	1500	10	1649	784	75.6	40.2	$\approx 1/2$
100	2000	20	2099	620	94.8	32.6	$\approx 1/3$
100	10000	100	10069	765	462.0	58.0	$\approx 1/8$
20	3000	150	2873	180	131.5	13.5	$\approx 1/10$

C_t = number of comparisons made by the tape merge algorithm t;
C_g = number of comparisons made by the generalized binary algorithm g;
T_t = time (in milliseconds) spent in making the comparisons using t;
T_g = time (in milliseconds) spent in making the comparisons using g.

REFERENCES

[1] F. K. HWANG AND S. LIN, *Optimal merging of 2 elements with n elements*, Acta Informatica, 1 (1971), pp. 145–158.
[2] ———, *Some optimal results for merging two disjoint linearly-ordered sets*, internal memorandum.