Minimum Comparison Merging of Sets of Approximately Equal Size

by

Paul E. Murphy
and
Marvin C. Paull

DCS-TR 72

Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

Abstract

The problem of merging ordered sets in the least number of binary comparisons has been solved completely only for a few special cases. When the sets to be merged are of size m and m' ($m \leq m' \leq m+4$) the tape merge algorithm has been shown to be optimum in the worst case. This paper significantly extends these results by showing that the tape merge algorithm is optimum in the worst case whenever one set is no larger than 1.5 times the size of the other. This result is obtained by defining an interesting and amusing two-player game isomorphic to the problem of merging ordered sets and analyzing the optimum strategies for each player. The form of this result should be applicable to the solution of similar sorting and selection problems.

Keywords: merging, sorting, comparison, mini-max, algorithm.

CR Categories: 5.31

August 8, 1978

Minimum Comparison Merging of Sets of Approximately Equal Size

by

Paul E. Murphy
and
Marvin C. Paull

Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

1. The Merging Problem.

Suppose we are given two disjoint linearly ordered sets of distinct elements and are asked to merge them into one set by a pairwise sequence of binary comparisons. Call the sets $A_m = a_1<a_2<\ldots<a_m$ and $B_n = b_1<b_2<\ldots<b_n$, this is called the (m,n) merging problem. We wish to determine the least number of comparisons which will always suffice to merge the sets. That is, given any algorithm x to solve this problem, let $M_x(m,n)$ be the maximum number of comparisons required by algorithm x as a function of m and n. Then,

$$M(m,n) = \min_x \{M_x(m,n)\}.$$

M(m,n) is called the mini-max, or best worst case, number of comparisons needed to merge A_m and B_n. Any algorithm which merges A_m and B_n in at most M(m,n) comparisons is said to be optimum in the mini-max sense. For the remainder of this paper "optimum" will be used exclusively in this sense and we will assume that $m \leq n$. The determination of M(m,n) has proven to be very difficult for general m and n. The following results have been obtained for special cases;

$$M(1,n) = \lceil \log_2 (n+1) \rceil, \tag{1}$$

$$M(m,m) = 2m-1, \tag{2}$$

$$M(m,m+1) = 2m, \tag{3}$$

$$M(m,m+2) = 2m+1 \qquad m \geq 2, \tag{4}$$

$$M(m,m+3) = 2m+2 \qquad m \geq 4, \tag{5}$$

$$M(m,m+4) = 2m+3 \qquad m \geq 6, \tag{6}$$

$$M(2,n) = \lceil \log_2 \tfrac{14}{17}(n+1) \rceil + \lceil \log_2 \tfrac{7}{12}(n+1) \rceil, \; n > 0. \tag{7}$$

Where $\lceil a \rceil$ denotes "the least integer not less than a."

When m=1 (1), the optimum algorithm is binary search/insertion. For the cases (2)-(6) the optimum algorithm is the so-called tape merge algorithm. This algorithm, which we will call t, continually choses the least remaining element in either A_m or B_n until one or the other of the sets is exhausted. Therefore, we must always have

$$M(m,n) \leq M_t(m,n) = m+n-1. \tag{8}$$

The case m=2 (7) is very interesting and was solved by Hwang and Lin [2] and Graham [cited in 4]. The case m=3 is equally interesting but quite a bit more complex. It has also been completely solved by Murphy [5], and by Hwang [3] and Nisselbaum [7]. This list completely exhausts the range of values for which M(m,n) is known precisely. A complete discussion of this problem, much useful introductory material, and a comprehensive bibliography may be found in Knuth's text [4].

The difficulty encountered in evaluating M(m,n) for general m and n is a result of the fact that heretofore the only known methods of solving these problems are essentially enumerative and thus the complexity of proofs increases very fast for each increase in n. Enumerative is used here in the sense that some basis set of configurations must be treated individually and at length prior to arguing

the general case. It is the purpose of this paper to introduce a significant generalization of the above results. We will show that

$$M(m,m+d) = 2m+d-1, \quad \text{for } 0 \leq d \leq \left\lceil \frac{m+1}{2} \right\rceil. \tag{9}$$

An interesting aspect of this fact is that its proof does not rely on enumeration. Equation (9) states that algorithm t is optimum as long as the size of one set is no larger than (roughly) 1.5 times the size of the smaller. In the sections which follow we will state some basic definitions and results and prove (9).

2. Basic Notions.

Merging two ordered sets in the minimum number of comparisons can be viewed as a two-player game in which the first player represents a merging algorithm and the second player represents nature. The problem of determining M(m,n) is then the problem of determining the optimum strategies for both players of this game. A move by the algorithm consists of a question denoted by the pair (i,j) representing:

"Is a_i < or > than b_j?"

The second player, nature, responds with either the relation '>' or '<' specifying the ordering of a_i and b_j. The response to any question may be either relation subject only to the restriction that it must be consistent with all previous responses in keeping with the transitive nature of these relations. The game ends when the merging algorithm has determined the relation between every member of A_m and every member of B_n. The algorithm's goal is to ask as few questions as possible, while nature is trying to force the algorithm to ask as many questions as possible.

The players of this game use a rectangular board to record the

'<' and '>' relations as they are determined. This board is an m by n array, denoted [m,n], whose individual entries are addressed in the usual (matrix) fashion. If the first player queries (i,j) and nature responds '>' then this relation is written at board position (i,j). Notice that because of the transitivity of this relation and the initial ordering of the sets A_m and B_n we may also write '>' in all matrix entries (x,y) such that $i \leq x \leq m$ and $1 \leq y \leq j$. Pictorially, this set of entries is the largest rectangle lying entirely within the game board whose upper right-hand corner is (i,j). Figure 1 illustrates [9,16] and the effect of a '>' response to a query of (5,7). A similar argument can be made if the result is '<'. In this event the set of board entries would be (x,y) such that $1 \leq x \leq i$ and $j \leq y \leq n$, and the pictorial representation would be the largest rectangle lying entirely within the game board whose lower left-hand corner is (i,j). In order to informally refer to the "largest rectangle lying ....." we say that the relation '>' ('<') propagates down and left (up and right) in the board. Responses to subsequent questions are entered in the same way. The game is over when all entries have been filled in with one relation or the other.

We have already pointed out in (8) that the tape merge algorithm is a strategy for the first player which never uses more than m+n-1 questions no matter how nature responds. On the other hand, we will show that when m and n are related as in (9) there is a strategy for the second player which guarantees that at least m+n-1 questions will be necessary to determine all the order relations between the two sets. In order to facilitate the description of this strategy and the proof that it is optimum we will define a number of very clever second

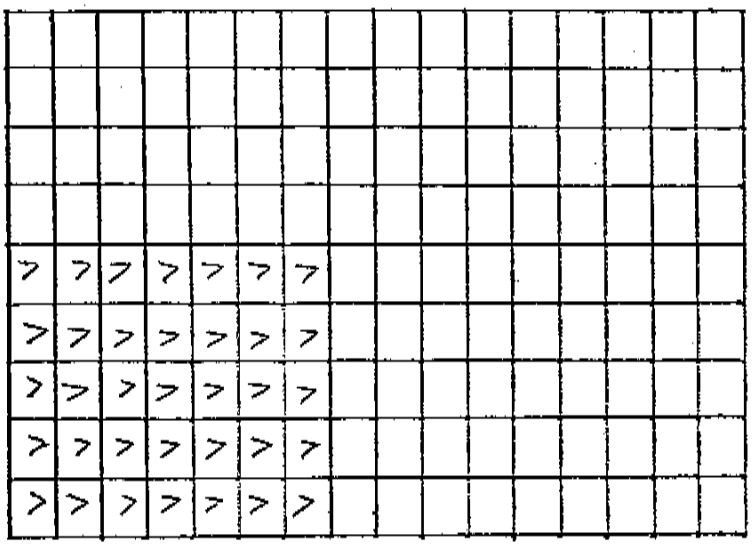

Figure 1. The effect of a '>' response to (5,7) on [9,16].

players called oracles. Each oracle is designated by two parameters, its name and the size of the board on which it is defined. The oracles of interest are named as follows:

t - which may respond freely to all questions,

s - which must respond to all questions so that $a_1 > b_1$,

s' - which must respond to all questions so that $a_m < b_n$,

r - which must respond to all questions so that both $a_1 > b_1$ and $a_m < b_n$.

We say that the s, s', and r oracles are constrained. (The s and s' versions of these players are identical under rotation.) The second parameter, board size, is specified directly by attaching a dimension to the name of the oracle as in t[m,n] or s[m,n], for example.

We will be describing strategies for different oracles. The same notation used to identify an oracle will also be used to denote

its strategy. The appropriate interpretation will be clear from the context. Oracles (with different properties) are used in [4] to prove (6) and elsewhere, when it is necessary to be precise about worst case situations.

We will denote the maximum number of questions the oracle named x, for $x \in \{t,s,s',r\}$, can always be assured of forcing any algorithm to ask by the upper case name of the oracle, X. X[m,n] is referred to as the value or cost of the oracle x[m,n].

Lemma 1. $$T[m,n] \geq S[m,n] \geq R[m,n]$$

Proof. If this were not the case then the less constrained oracle could simply adopt the strategy of the more constrained variety to get an equal minimum number of questions. Relaxing constraints allows the oracle the possibility of making more effective responses and so the lemma follows.

The proof of theorem 1 below describes an oracular strategy for [m,n] recursively in terms of a set of sub-oracles. Each sub-oracle in the set is assigned responsibility for a distinct sub-board of [m,n]. To precisely locate a sub-oracle on a sub-board we will use the notation {i,j}x[p,q] to specify the p by q oracle x whose (1,1) entry corresponds to the (i,j) entry of [m,n].

If the set which composes oracle x includes the sub-oracles $x_1, x_2, \ldots, x_n$, we write

$$x \supseteq x_1 \cup x_2 \cup \cdots \cup x_n. \tag{10}$$

In order for such a set of sub-oracles to unambiguously define a strategy for the oracle no position of the board can be assigned to more than one of the members of this set. If, in addition, the sub-oracles are constrained so that a response made by one cannot propagate to a

part of the board assigned to any other sub-oracle, then (10) implies,

$$X \geq X_1 + X_2 + \cdots + X_n.$$

Because the oracles described in theorem 1 will adopt different strategies (i.e. sets of sub-oracles) depending on the nature of the initial query, we say that they are adaptive.

We conclude this introduction with several observations and two lemmas. Consider figure 2 and notice that R[1,2] = 2 because no matter which board entry is queried first, the oracle can always respond so that a second question is required to determine the other relation. Also notice that the constraints r[1,2] must meet actually represent the optimum strategy for the oracle. A similar argument would show that R[1,4] = 3. The following two lemmas supply the structure needed to apply the induction hypothesis of theorem 1 and its generalization in theorem 2.

Lemma 2. If $S[2k+1,3k+2] \geq 5k+2$ then $S[2k+2,3k+3] \geq 5k+4$.

Proof. Let $s[2k+2,3k+3] \supseteq \{1,1\}r[1,2] \cup \{2,2\}s[2k+1,3k+2]$.

Lemma 3. If $R[2k+1,3k+4] \geq 5k+3$ then $R[2k+2,3k+5] \geq k+5$.

Proof. Let $r[2k+2,3k+5] \supseteq \{1,1\}r[1,2] \cup \{2,2\}r[2k+1,3k+4]$.

The construction in each case relies upon the constraints of the r[1,2] oracle at (1,1) and the s (or r) sub-oracle at (2,2) being "matched". That is, no information can propagate from one to the other in column 2 of the game board.

3. The Main Theorem.

The next result uses lemmas 2 and 3 to inductively construct adaptive strategies such that (9) is proven.

Comparison	Result	Sub-oracles
Part a. $T[2k,3k+1] \geq 5k$.		
1. $i=2p+1$, $j \leq 3p+1$	'>'	$\{1,1\}t[2p,3p+1]$ $\cup$ $\{2p+1,3p+2\}s[2(k-p),3(k-p)]$
2. $i=2p+1$, $j>3p+1$	'<'	$\{1,1\}s'[2p+1,3p+1]$ $\cup$ $\{2p+2,3p+1\}s[2(k-p-1)+1,3(k-p-1)+1]$
Part b. $S[2k+1,3k+2] \leq 5k+2$		
1. $i=2p+1$, $j \leq 3p+1$	'>'	$\{1,1\}r[2p,3p+2]$ $\cup$ $\{2p+1,3p+2\}s[2(k-p)+1,3(k-p)+1]$
2. $i=2p+1$, $j>3p+1$	'<'	$\{1,1\}s[2p+1,3p+1]$ $\cup$ $\{2p+2,3p+2\}t[2(k-p),3(k-p)+1]$
3. $i=2p$, $j \leq 3p$	'>'	$\{1,1\}r[2p-1,3p+1]$ $\cup$ $\{2p,3p+1\}s[2(k-p+1),3(k-p+1)-1]$
4. $i=2p$, $j>3p$	'<'	$\{1,1\}s[2p,3p]$ $\cup$ $\{2p+1,3p+1\}s[2(k-p)+1,3(k-p)+2]$
Part c. $R[2p+1,3k+4] \geq 5k+3$		
1. $i=2p$, $j \leq 3p+1$	'>'	$\{1,1\}r[2p-1,3p+1]$ $\cup$ $\{2p,3p+2\}s'[2(k-p+1),3(k-p+1)]$
2. $i=2p$, $j>3p+1$	'<'	$\{1,1\}s[2p,3p]$ $\cup$ $\{2p+1,3p+1\}r[2(k-p)+1,3(k-p)+4]$
3. $i=2p+1$, $j \leq 3p+2$	'>'	$\{1,1\}r[2p,3p+2]$ $\cup$ $\{2p+1,3p+3\}s'[2(k-p)+1,3(k-p)+2]$
4. $i=2p+1$, $j>3p+2$	'<'	$\{1,1\}s[2p+1,3p+2]$ $\cup$ $\{2p+2,3p+3\}r[2(k-p),3(k-p)+2]$

Table 1. Adaptive Oracles for Theorem 1, parts (a)-(c).

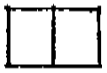

Figure 2. The board [1,2].

Theorem 1. There exist strategies for each of the following oracles such that the stated bounds are met or exceeded and all constraints are fulfilled. For all $k \geq 0$.

a. $T[2k,3k+1] \geq 5k$,

b. $S[2k+1,3k+2] \geq 5k+2$, and

c. $R[2k+1,3k+4] \geq 5k+3$.

Proof. By induction on k. Each part is true for $k = 0$, part (a) trivially so, parts (b) and (c) from the observations that $R[1,2] = 2$ and $R[1,4] = 3$. Therefore, assume the theorem is true for all integers less than k and that $k > 0$. We will prove the theorem for k. Notice that parts (b) and (c) of the induction hypothesis are precisely the assumptions of lemmas 2 and 3. In essence, when we are presented with a [2p,3p] sub-board (say) we "back-up" to a composition of sub-oracles as shown in the proof of lemma 2. The pieces of this composition satisfy the induction hypothesis so we may say $S[2p,3p] = 5p-1$.

Table 1 is the proof of the theorem. For each part of theorem 1 this table shows that there is an adaptive oracle such that after any initial query an algorithm can make there remains a sufficiently complex set of sub-oracles for those sections of the game board unaffected by the initial response. This set of sub-oracles is contructed from oracles satisfying the inductive assumption parts (a)-(c) either

directly or by repeated use of lemmas 2 and 3 and the observation that S[1,1] = 1. The sets constructed in this manner meet or exceed the specified bound and fulfill all the constraints of the theorem. We will describe in detail the proof for part (a) of the theorem, parts (b) and (c) are proven in a similar manner.

Let the initial query be board entry (i,j). The proof breaks these boards into odd and even numbered rows, but the board of part (a) consists of an even number of rows so that it is only necessary to show how the oracle responds to an initial query designating an odd numbered row. When an even row is queried first the oracle can, without loss of generality, rotate the board 180 degrees and map the indices of the original query into a new row, which will be odd, and a new column. The new indices will be (2k-i+1,3k-j+2). It can be seen from table 1 that on every row of the board there is an entry such that the oracle responds '>' if this entry or any entry to its left is queried initially, and responds '<' otherwise. Therefore assume that i = 2p+1 and the initial query is therefore node (2p+1,j). Notice that p < k.

The table specifies that if $j \leq 3p+1$ then the oracle responds '>' and plays a t[2p,3p+1] sub-oracle in the top left corner of the board and an s[2(k-p),3(k-p)] sub-oracle in the bottom right. T[2p,3p+1] = 5p by the induction hypothesis. By lemma 2 and the induction hypothesis S[2(k-p),3(k-p)] = 5(k-p)-1. The sum of these two strategies is 5k-1 as was claimed. On the other hand, if j > 3p+1 then the oracle responds '<' and plays a s'[2p+1,3p+1] sub-oracle in the top left corner of the board and an r[2(k-p-1)+1,3(k-p-1)+4] sub-oracle in the bottom right hand corner. The latter has cost 5(k-

p-1)+3 by the induction hypothesis, the former has cost 5p+1 by the induction hypothesis and two applications of lemma 2. (When p = 0 the observation that S[1,1] = 1 is sufficient.) Notice that the matched constraints of these sub-oracles prevent the propagation of information from one to the other in column 3p+1. Consequently, full value is assured from each and the total cost of these strategies is 5k-1, as was claimed.

In each case we have shown that there exists a set of strategies for any comparison the algorithm might make that meets or exceeds the stated bound while satisfying the necessary constraints and therefore the theorem is proved.

Using (8), lemmas 1, 2 and 3, and theorem 1 we can conclude

Theorem 2. (The Main Theorem).

$$M(m,m+d) = 2m+d-1, \quad \text{for } 0 \leq d \leq \left\lceil \frac{m+1}{2} \right\rceil. \tag{9}$$

Table 2 gives the values for M(m,n) for all $m \leq n \leq 20$ with the exception of several entries as yet unsolved and marked "?". The (7,12) merging problem is extremely interesting. It can be shown that if the first comparison by any algorithm to solve this problem is not a_3 to b_5 then the algorithm can do no better than tape merge in the worst case. A similar situation exists for the (9,15) problem if the first comparison is not a_5 to b_8. Proofs of these facts are given in [6] and are similar to the solution to the (5,9) problem given in [4, pg. 632].

4. Conclusions.

Theorem 2 does not specify all values of m and n for which the

m \ n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	2	2	3	3	3	3	4	4	4	4	4	4	4	4	5	5	5	5	5
2		3	4	5	5	6	6	6	7	7	7	7	8	8	8	8	8	8	9	9
3			5	6	7	7	8	8	9	9	10	10	10	11	11	11	11	12	12	12
4				7	8	9	10	10	11	11	12	12	13	13	13	14	14	14	15	15
5					9	10	11	12	12	13	13	14	14	15	15	16	16	17	17	17
6						11	12	13	14	15	15	16	16	17	17	18	18	19	19	20
7							13	14	15	16	17	17	18	19	19	20	20	21	21	22
8								15	16	17	18	19	20	20	21	22	22	23	23	24
9									17	18	19	20	21	22	22	23	24	?	25	?
10										19	20	21	22	23	24	25	25	26	27	?
11											21	22	23	24	25	26	27	28	28	29
12												23	24	25	26	27	28	29	30	30
13													25	26	27	28	29	30	31	32
14														27	28	29	30	31	32	33
15															29	30	31	32	33	34
16																31	32	33	34	35
17																	33	34	35	36
18																		35	36	37
19																			37	38
20																				39

Table 2. M(m,n) for all $n \leq 20$.

tape merge algorithm is optimum. For instance, the following is proven in [6];

$$M(2k+1,3k+3) = 5k+2 \quad \text{for } k > 4.$$

and similar extensions are possible for larger k. It therefore makes sense to ask "For fixed m, what is the least value of d such that M(m,m+d) = M(m,m+d+1)?" Hwang and Lin have shown that a d which satifies this relation is m-1 although m-1 is not necessarily the first such d [1]. Knowing d for each m would of course determine exactly the range of values of m and n for which algorithm t is optimal.

In this paper we significantly extended the range of values for which M(m,n) has been completely solved. This is the first time extensive exact values have been obtained for the mini-max number of comparisons required for any of the related problems of sorting, merg-

ing, or selecting. The precise results are not surprising and have been conjectured in various ways for a number of years, however, the method used is certainly unique and opens the way for a re-examination of a number of similar problems that might reasonably be viewed as question-response contests.

5. References.

[1] Hwang, F. K., and Lin, S. Some optimality results in merging two disjoint linearly ordered sets. BTL Internal Memo, (April 20, 1970).

[2] Hwang, F. K., and Lin, S. Optimal merging of 2 elements with n elements. Acta Informatica 1, 145-178 (1971).

[3] Hwang, F. K. Private Communication.

[4] Knuth, D. E. The Art of Computer Programming vol III: Sorting and Searching. Addison-Wesley, (1973).

[5] Murphy, P. E. A problem in optimal merging. Rutgers University, Dept. of Computer Science, DCS-TR-69.

[6] Murphy, P. E. Minimum comparison merging and merging algorithms. Dissertation. Rutgers University, Dept. of Computer Science. (May, 1978).

[7] Nisselbaum, Y. Abstract, SIGACT Notices. (Winter 1978).