OPTIMALITY IN MERGING (N,2) AND (N,3) ELEMENTS

Ahmed A. Belal

Computer Sciences Department, Faculty of Engineering,
Alexandria University, Alexandria, Egypt.

ABSTRACT

Although abandoned for several years, the problems of sorting and merging in the least number of comparisons, is an area full of unanswered questions. Despite the fact that the theoretical bound is known in all the cases, the actual minimum is only known in just a very few cases. For example in the problem of sorting n elements, the theoretical minimum is log (n!) but the actual minimum is only known for values up to n=14. For values greater than 14, it is still not known whether the published values obtained by FORD and JOHNSON in 1959 are indeed the best possible. Prior to 1959, the best known results where those obtained then years earlier in 1949. In merging algorithms of n and m elements, the widely used algorithm using n+m-1 comparisons is only optimum when m is close to n. For other values optimal solutions are not known, except when they coincide with the theoretical minimum. The reason those problems were left unchallenged for so many years, is because of their tremendous complexities. To guarantee an optimal solution using the minimum number of comparisons, one has to consider all the possible routes emerging from every possible comparison of two elements,a really formidable task. In the sixties it book over 100 hours on a large IBM to prove, beyond doubt, that the minimum number of comparisons needed to sort 12 elements by the method of FORD and JOHNSON (30 comparisons) is indeed the best possible, although the theoretical minimum based on the information theory measure is log (12!)=29.In this paper, strategies for merging n and m elements based on maximizing the amount of information obtained in each step are investigated for the two cases m=2 and m =3.The results obtained in this paper are of a more theoretical interest. The results will show that by maximizing the amount of information obtained for several steps, one cannot do worse, as far as the number of caparisons is concerned, than the best known existing algorithm of HWANG and LIN. Simple algorithms based on a three-step analysis are obtained and shown to use a smaller number of comparisons than the currently used algorithms.

INTRODUCTION

Methods for merging n,m, elements in the least number of comparisons, have been extensively treated in the literature [1], [2], [3], [4], [5], [6].

The information theoretic bound $\lceil \log \binom{n+m}{m} \rceil$ comparisons is hard to achieve in most cases.

The least number of comparisons needed to merge n, m, elements is only known in a few cases, [2], [6].

Analysis of merging strategies are best treated using transitive digraphs [2], [5]. Figure (1) shows the representation for merging n,m elements where $X_1<X_2<X_3....<X_n; Y_1<Y_2<Y_3<....<Y_m$. The number of permutations of the (n+m) elements consistent with Figure (1). is

$$P=\binom{n+m}{m} \qquad (1)$$

which is just the number of ways of choosing m out of (n+m) objects.

Connecting vertices X_i, Y_j in Figure (1). will result in two different digraphs, one for the case $X_i < Y_j$ Figure (2) and one for the case $X_i > Y_j$ Figure (3).

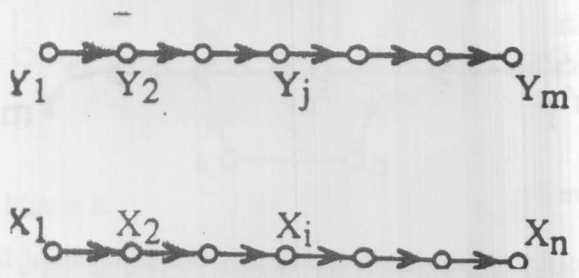

Figure 1.

If P_1, P_2 are the number of permutations now consistent with Figure (2). and Figure (3) respectively, then obviously

$$P_1 + P_2 = P = \binom{n+m}{m}$$

if $P_1 \geq P_2$ then the amount of information obtained in the worst case is

$$I = \log \frac{P}{P1} \qquad (2)$$

Maximizing (2) implies minimizing the difference between P_1, P_2. Merging strategies based on maximizing the amount of information obtained in every step is in general a hard task to perform for general n,m.

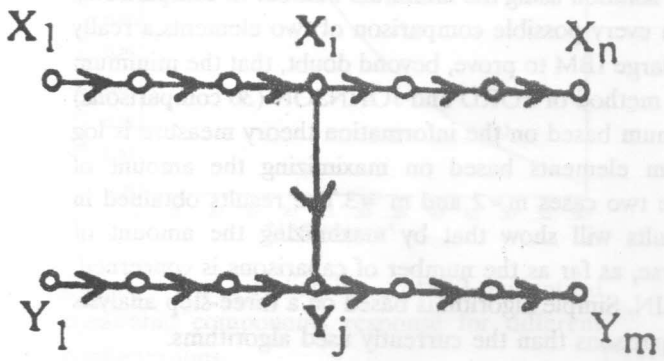

Figure 2.

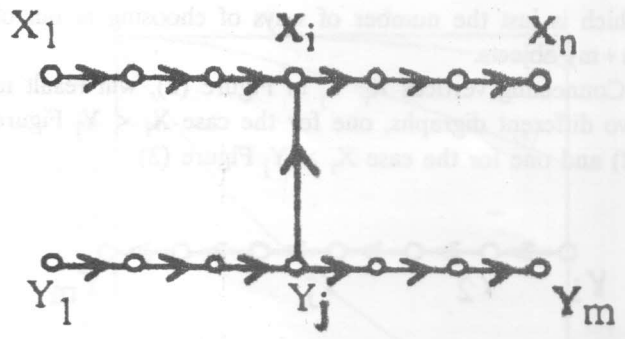

Figure 3.

In cases where simple expressions can be obtained for P_1, P_2 this optimization procedure can be carried out for several steps resulting in more optimal, yet more complicated, algorithms than the currently used ones.

III. The Case m = 2

The minimum number of comparisons needed to merge n and 2 elements was found by Hwang, Lin, and Graham [2] to be

$$M(n) = \lceil \log \frac{7}{12}(n+1) \rceil + \lceil \log \frac{14}{17}(n+1) \rceil \qquad (3)$$

No existing algorithm can achieve this minimum for all values of n.

In what follows an algorithm based on maximizing equation (2) is developed and compared to the very simple and efficient algorithm developed by Hwang and Lin [2], for different values of n.

In Figure (4), connecting the vertices a, j will give

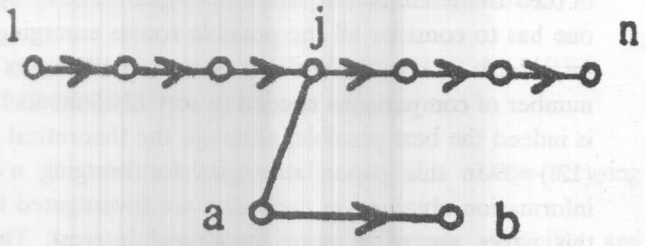

Figure 4.

$$P_1 = \frac{(n+1)(n+2)}{2} - P_2$$

$$P_2 = \frac{j}{2}(2n - j + 3)$$

Setting $P_1 = P_2$ and solving for j we get,

$$j_o = \left| \frac{2n+3-d}{2} \right|,$$

$$d^2 = 2n^2 + 6n + 5 \qquad (4)$$

where | | means the closest integer.

step 1

So, the first step of the algorithm is to determine j_o and compare Y (a), X (j_o).

Figures (5) ,(6) are the resulting digraphs after the comparison

If Y (a) > X (j_o) - Figure (5)- then we merge X (j_o+1), ..., X (n) with Y (a), Y (b).

If Y (a) < X (j_o) - Figure (6)- then we perform an optimization step on Figure (6).

Analysis of the more general graph G (m,k) of Figure (7). will show that the optimal step is one of the following three:

(A) Connect a, $j_1 < k$ where

$$j_1 = \left|\frac{2m+3-r}{2}\right|,\quad r^2 = 4m^2+m(12-4k)+2k^2-6k+9$$

$$\text{Here } P_1 = \frac{(k-j_1)(2m+3-k-j_1)}{2} \qquad (A)$$

(B) Connect b, $j_1 < k$ where

$$j_1 = \left|\frac{r-1}{2}\right|,\quad r^2 = 1+2(2km+3k-k^2)$$

$$\text{Here } P_1 = \frac{(k-j_1)(k+j_1+1)}{2} + k(m-k+1) \qquad (B)$$

(C) Connect b, $j_1 \geq k$ where

$$j_1 = |(2m+k+1)/4|$$

$$\text{Here } P_1 = (m-j_1+1)k \qquad (C)$$

The step to choose is the one for which P_1 is closest to $\frac{P}{2}$ where

$$P = \frac{k(2m-k+3)}{2}$$

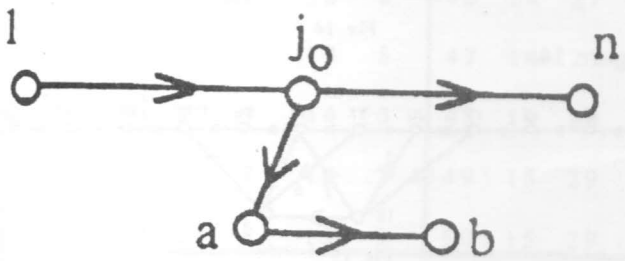

Figure 5.

By applying these steps to the graph G (n,j_o) of Figure (6) it turns out that the best step here is (C)for all values of n.

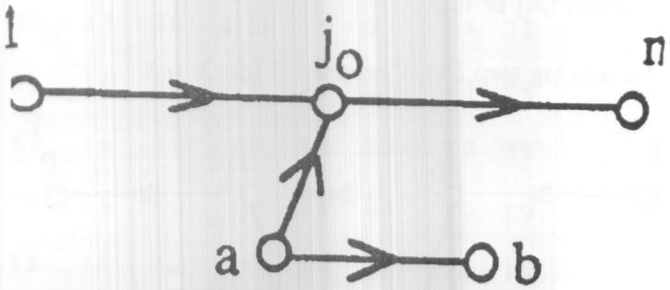

Figure 6.

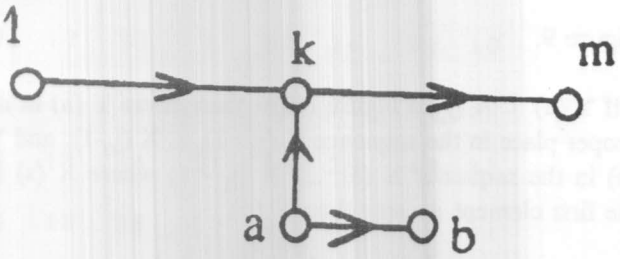

Figure 7.

Step 2

The second step of the algorithm if Y (a) < X (j_o) in the first step is to calculate

$$j_1 = \left|\frac{2m+k+1}{4}\right| \text{and compare } Y(b), X(j_1).$$

If Y (b) > X (j1) - Figure (8)- then insert Y (a) in its proper place in the sequence X (1),..., X (j_o-1)Y(b) in its proper place in the sequence X (j_1+1),..., X (n).

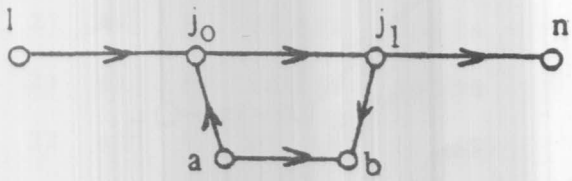

Figure 8.

If Y (b) < X (j_1) - Figure (9)- then if $j_1 = j_o$ merge Y (a), Y (b) with X (1),..., X (j_o-1).

If $j_1 > j_o$ we repeat steps A,B,C on the graph G (j_1-1,j_o).

Step 3

We choose our last optimizing step for the graph G $(j_1\text{-}1,j_o)$. It turns out that the optimizing step here is always (A) or (C).

* Choosing step (A): Compare Y (a), X (j_2).

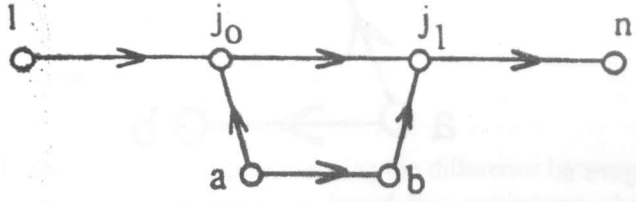

Figure 9.

If Y (a) < X (j_2) - Figure (10) - then insert Y (a) in its proper place in the sequence X (j_2+1), ... X $(j_o\text{-}1)$; and Y (b) in the sequence X (s), ..., X $(j_1 - 1)$ where X (s) is the first element greater than Y (a).

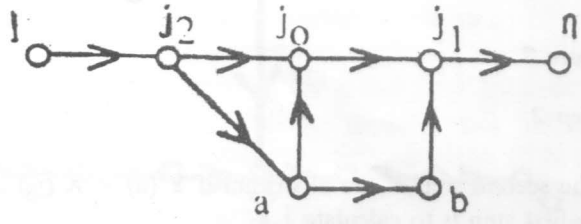

Figure 10.

If Y (a) $\nless$ X (j_2) - Figure (11) - then insert Y (a) in X (1), ..., X $(j_2\text{-}1)$ and Y (b) in X (s),...., X $(j_1 - 1)$.

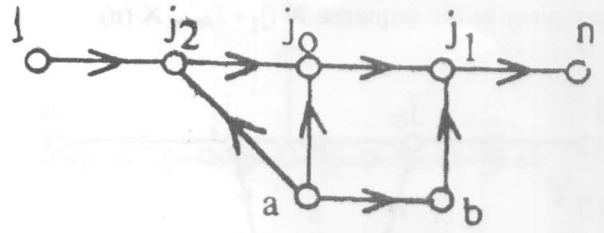

Figure 11.

* Choosing step (C): Compare Y (b), X $(j_2) \geq$ X (j_o)

If Y (b) > X (j_2) - Figure (12) - insert Y (a) in X (1),... X $(j_o\text{-}1)$ and Y (b) in X (j_2+1),..., X $(j_1\text{-}1)$.

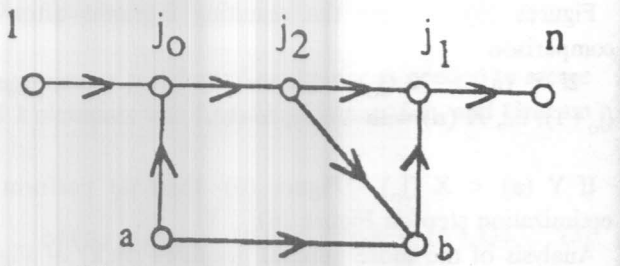

Figure 12.

If Y (b) $\nless$ X (j_2) - Figure (13) - then insert Y (a) in X (1), ..., X $(j_o\text{-}1)$ and Y (b) in X (s),...., X $(j_2 - 1)$ when $j_2 \neq j_o$, or merge Y (a), Y (b) with X (1),..., X $(j_o\text{-}1)$when $j_2 = j_o$, [X (s) is the first element greater than Y (a)].

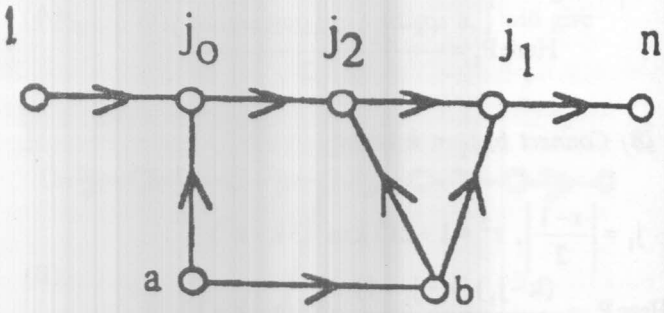

Figure 13.

Table (1), gives values of j_o,j_1,j_2 for n up to 100. Two examples using this algorithm are shown below for n = 12. The first example Figure (14) requires 7 comparisons and the second Figure (15), 6 comparisons. The maximum number of comparisons K (n) used by the algorithm presented in this paper is compared in Table (2), with the theoretical bound M (n) of equation (3) and with the number of comparisons H (n) required by the very simple and efficient algorithm of Hwang. and Lin.

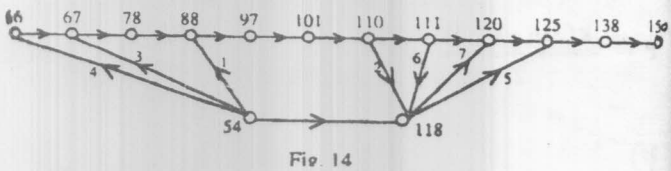

Fig. 14

Figure 14.

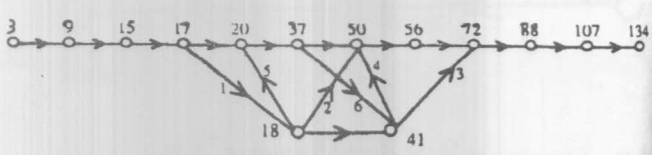

Fig. 15

Figure 15.

Table (1).

n	j_0	j_1	j_2	n	j_0	j_1	j_2	n	j_0	j_1	j_2	n	j_0	j_1	j_2
,	,	,	,	26	8	15	9	51	15	30	6,19	76	23	44	28
2	1	2	1	27	8	16	10	52	16	30	19	77	23	45	28
3	1	2	1	28	9	17	4	53	16	31	19	78	23	45	28
4	2	3	1,2	29	9	17	4	54	16	31	19	79	24	46	10
5	2	3	1,2	30	9	18	11	55	17	32	20	80	24	46	10
6	2	4	1,2	31	10	18	4	56	17	33	7	81	24	47	10
7	2	4	1,2	32	10	19	4,12	57	17	33	7	82	24	47	10
8	3	5	3	33	10	19	4,12	58	17	34	21	83	25	48	30
9	3	6	1,3,4	34	10	20	12	59	18	34	21	84	25	49	31
10	3	6	1,3,4	35	11	21	13	60	18	35	22	85	25	49	31
11	4	7	4	36	11	21	13	61	18	35	22	86	26	50	31
12	4	7	4	37	11	22	14	62	19	36	8	87	26	50	31
13	4	8	5	38	12	22	5	63	19	37	23	88	26	51	11
14	5	9	2	39	12	23	5	64	19	37	23	89	27	52	11
15	5	9	2	40	12	23	5	65	19	38	8	90	27	52	11
16	5	10	6	41	12	24	5	66	20	38	24	91	27	53	33
17	5	10	6	42	13	25	16	67	20	39	24	92	27	53	33
18	6	11	7	43	13	25	16	68	20	39	24	93	28	54	34
19	6	11	7	44	13	26	16	69	21	40	25	94	28	54	34
20	6	12	7	45	14	26	16	70	21	41	9	95	28	55	34
21	7	13	8	46	14	27	6,17	71	21	41	9	96	29	56	35
22	7	13	8	47	14	27	6,17	72	22	42	9	97	29	56	35
23	7	14	3	48	14	28	6	73	22	42	9	98	29	57	12
24	7	14	3	49	15	29	18	74	22	43	27	99	29	57	12
25	8	15	9	50	15	29	18	75	22	43	27	100	30	58	36

IV. The Case m = 3

$$\text{Here } P = \binom{N+3}{3} = \frac{(n+3)(n+2)(n+1)}{6} \qquad (5)$$

The first optimizing step will depend on whether n is odd or even Figure (16)

Because of the symmetry involved it is clear that by connecting the two mid-points $j = \frac{n+1}{2}$ and b, P is split into two equal halves and one full bit of information is obtained, hence we consider the case n odd separately.

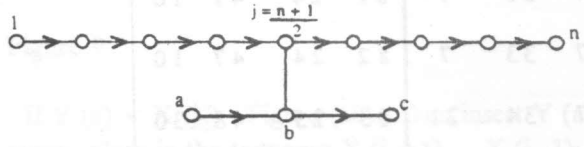

Figure 16.

First n odd:

Step 1: Compare the two mid points X $(\frac{n+1}{2})$, Y (b). By symmetry we now consider the graph of Figure (17) for which the number of consistent permutations is

$$P = \frac{(n+3)(n+2)(n+1)}{1\ 2} \qquad (6)$$

$$j_o = \frac{n+1}{2}$$

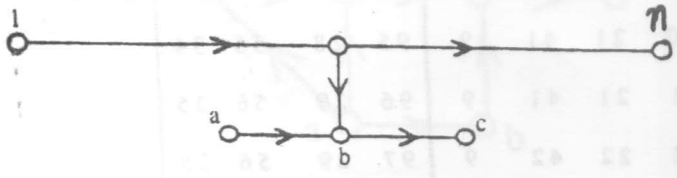

Figure 17.

For the second step, we have to consider one of the three connections of Figure (18).

Again the one to choose is the one for which the absolute value of (P_1-P_2) is the smallest. The best second connection is given in Table (3) up to n = 99.

Although, as shown in the Table, connection to vertex "a" is not the optimizing step in a few cases, we nevertheless for simplicity choose it as the second optimizing step in our algorithm.

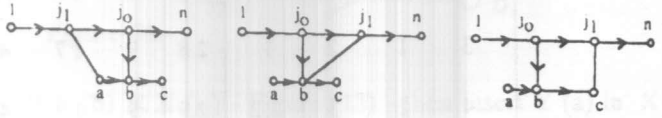

Figure 18.

Step 2: Consider the graph of Figure (19). The number of consistent permutations with this graph is

Table (2)

n	2	3	4	5	6	7	8	9	10	11	12	14	15	18	22	26	31	38
M(n)	3	4	5	5	6	6	6	7	7	7	7	8	8	8	9	9	10	11
H(n)	3	4	5	5	6	6	7	7	7	7	8	8	8	9	9	10	10	11
K(n)	3	4	5	5	6	6	6	7	7	7	7	8	8	9	9	10	10	11

Table (3)

n	step	n	step	n	step	n	step	n	step
3	2.a	23	8.a	43	15.a	63	22.a	83	28.a
5	5.c	25	9.a	45	31.b	65	57.c	85	29. a
7	3.a	27	19.b	47	16.a	67	23.a	87	76.c
9	7.b	29	26.c	49	17.a	69	24.a	89	30.a
11	4.a	31	11.a	51	45.c	71	24.a	91	31.a
13	5.a	33	23.b	53	18.a	73	25.a	93	32.a
15	11.b	35	31.c	55	19.a	75	26.a	95	83.c
17	6.a	37	13.a	57	50.c	77	26.a	97	33.a
19	7.a	39	27.b	59	20.a	79	27.a	99	34.a
21	15.b	41	14.a	61	21.a	81	71.c		

$$P_2 = \frac{j(n+1)(n+3)}{8}$$

with P given by equation (6), P_2 is closest to $\frac{P}{2}$ when $j=j_1$ where

$$j_1 = \left|\frac{n+2}{3}\right| \qquad (7)$$

Therefore the second step in our algorithm is to compare X (j_1) and Y (a).

If X (j_1) > Y (a) - Figure (19)- then insert Y (a) in the sequence X (1),...X(j_1-1) and merge Y (b), Y(c) with the sequence X (j_o+1),..., X (n) using the merging algorithm for m = 2.

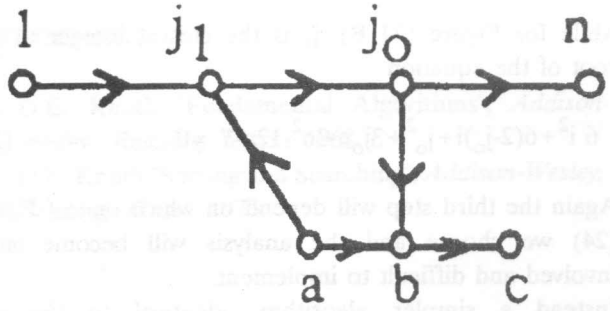

Figure 19.

If X (j_1) < Y (a) - Figure (20)- the following refinement step is used to reduce the number of needed comparisons.

Step 3:

Compare X (j_2) and Y (a) where

$$j_2 = j_1 + 2^r,\ r = \lceil \log j_1 \rceil - 1 \qquad (8)$$

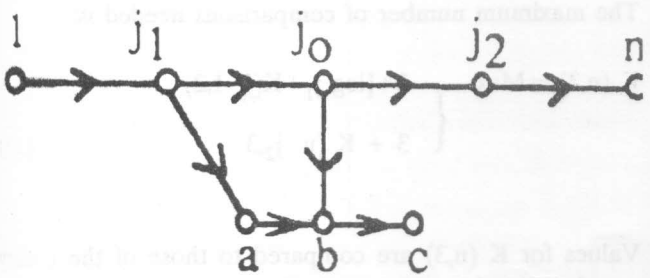

Figure 20.

Figure (21) gives the two graphs resulting from step 3.

If X (j_2) < Y (a), then merge the three elements Y (a), Y (b), Y (c) with the sequence X (j_2+1),...,X (n).

If X (j_2) > Y (a), insert Y (a) in X (j_1+1),..., X (j_2-1) and merge the two elements Y (b), Y (c) with either the sequence X (j_o+1),..., X (n) if Y (a) < X (j_o), or the sequence X (s), .., X (n) if Y (a) > X (j_o) where X (s) is the first element greater than Y (a).

The maximum number of comparisons K (n,3) needed by this algorithm is

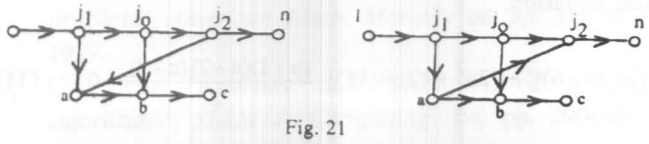

Fig. 21

Figure 21.

$$K(n,3) = \text{Max.} \begin{cases} 2+\lceil \log j_1 \rceil + K(n-j_o,2) \\ 3+\lceil \log (j_2-j_1) \rceil + K(n-j_o,2) \\ 3+K(n-j_2,3) \end{cases} \qquad (9)$$

Values for j_o,j_1, j_2 are given in Table (4) for different n.

Table (4)

n	3	5	7	9	11	13	15	17	19	21	23	25	27	29
j0	2	3	4	5	6	7	8	9	10	11	12	13	14	15
j1	2	2	3	4	4	5	6	6	7	8	8	9	10	10
j2	3	3	5	6	6	9	10	10	11	12	12	17	18	18

The example shown in Figure (22) uses 8 comparisons to merge 15 keys with 3 keys.

It should be noted that j_2 in step 3 was chosen so that the value of line 2 in equation (9) never exceeds that of line 1, so we can omit the second line and the equation becomes.

$$K(n,3) = \text{Max.} \begin{cases} 2+\lceil \log j_1 \rceil + K(n-j_o,2) \\ 3+ K(n-j_2,3) \end{cases} \qquad (10)$$

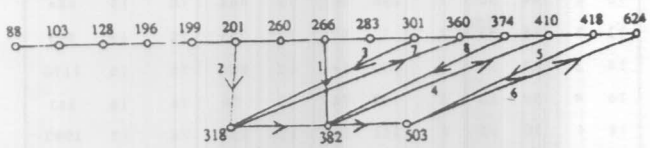

Figure 22.

* *Second n even::*

Here the first optimizing step is not obvious and we have to choose between the two connections shown in Figure (23)

Following the same optimization procedure as before, the value for j_o in Figure (23-A) is the closest integer to the real root of the equation.

$$j^3-(3n+6)j^2+(2n^2+12n+11)j-\frac{(n+1)(n+2)(n+3)}{2}=0 \qquad (11)$$

$$j_o=\frac{n}{2}+1 \qquad (12)$$

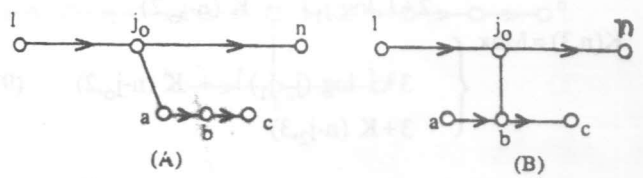

Fig. 23

Figure 23.

The step to choose is the one that gives a smaller difference between P_1, P_2. Solutions to equation (11) together with the corresponding values for

$$D = A\,bs\,(P_1-P_2) \qquad (13)$$

are given in Table (5).

For Figure (23-B) and j_O given by (12) the value for D is

$$D=(\frac{n}{2}+1)^2 \qquad (14)$$

Table (5).

n	j_o	D	n	j_o	D	n	j_o	D	n	j_o	D
4	1	5	24	5	155	44	9	657	64	14	1053
6	2	14	26	6	112	46	10	146	66	14	76
8	2	3	28	6	105	48	10	495	68	14	1365
10	2	44	30	7	256	50	11	466	70	15	484
12	3	15	32	7	7	52	11	253	72	15	915
14	3	48	34	7	350	54	12	880	74	16	1170
16	4	59	36	8	149	56	12	79	76	16	343
18	4	30	38	8	252	58	12	858	78	17	1992
20	5	139	40	9	373	60	13	511	80	17	361
22	5	20	42	9	90	62	13	520	82	17	1462

If the first step is to compare Y (a), X (j_o) Figure (23-a) then for the second step we have to consider either one of Figures (24) again choosing the has one that a has smaller value for D of equation (13).

The value for j_1 in Figure (24-A) that gives the maximum amount of information is the closest integer to the root of the equation

$$6\,i^2-6(2n+3)i+3n^2+6n+1+3j_on+6j_o-j_o^2=0 \qquad (15)$$

while for Figure (24 B), j_1 is the closest integer to the root of the equation

$$6\,i^2+6(2-j_o)i+j_o^2+3j_on-3n^2-12n-7=0 \qquad (16)$$

Again the third step will depend on which one of Figure (24) we choose and the analysis will become more involved and difficult to implement.

Instead a simpler algorithm, identical to the one implemented when n is odd, is also used for the case when n is even.

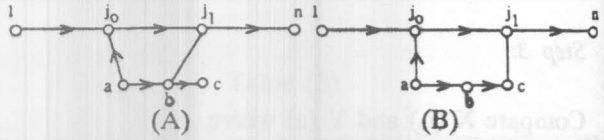

Figure 24.

The values for j_O, j_1, j_2 as given in - Table (4) are evaluated for n+1, for example for n = 8:$j_o=5$,$j_1=4$,$j_2=6$. The maximum number of comparisons needed is:

$$K(n,3)=\text{Max.}\begin{cases}2+[\log j_1+K(j_o-1,2)\\ 3+K(n\;\; j_2,3\end{cases} \qquad (17)$$

Values for K (n,3) are compared to those of the binary merging algorithm H (n,3) in Table (6).

Table (6).

n	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
H(n,3)	5	6	7	8	8	9	9	10	10	11	11	11	11	12	12	12	12
K(n,3)	5	6	7	8	8	9	9	9	10	11	11	11	11	11	11	12	12

CONCLUSION

Algorithms for merging (n,2) and (n,3) elements based on maximizing the amount of information of each comparison were developed.

The analysis in this paper used optimization for the first three comparisons only. General results were obtained and evaluated for values of n reaching 100, with several examples shown for illustration.

REFERENCES

[1] D.E. Knuth "Fundamental Algorithms", *Addison-Wesley, Reading, Mass.*, 1968.

[2] D.E. Knuth "Sorting and Searching", *Addison-Wesley, Reading, Mass.*, 1973.

[3] V.R. Pratt and F.F. Yao, "On lower bounds for computing the ith largest element", *Conference Record, IEEE 14th Annual Symposium on Switching and Automota Theory*, 70-81, 1973.

[4] R.W. Floyd and R.L. Rivest, "Expected time bounds for selection", *Computer Science Dept., Stanford University*, 1973.

[5] L.R. Ford and S.M. Johnson, "A tournament problem", *American Math. Monthly 66, pp. 387-389*, 1959.

[6] C.L. Liu "Analysis and Synthesis of sorting algorithms", *SIAM J. Computing 1:4*, pp. *290-304*, 1972.