SIAM J. COMPUT.
Vol. 7, No. 1, February 1978

MINIMEAN MERGING AND SORTING: AN ALGORITHM*

R. MICHAEL TANNER†

Abstract. A simple partitioning algorithm for merging two disjoint linearly ordered sets is given, and an upper bound on the average number of comparisons required is established. The upper bound is $1.06 \log_2 \binom{n+m}{m}$, where n is the number of elements in the larger of the two sets, m the number of the smaller, and $\binom{n+m}{m} = (n+m)!/(m!\,n!)$. An immediate corollary is that any sorting problem can be done with an average number of comparisons within 6% of the information theoretic bound using repeated merges; it does not matter what the merging order used is. Although the provable bound is 6% over the lower bound, computations indicate that the algorithm will asymptotically make only 3% more comparisons than the lower bound. The algorithm is compared with the Hwang-Lin algorithm, and modifications to improve average efficiency of this well known algorithm are given.

Key words. merging, algorithm, sorting, average, bound, insertion, probability, information

Introduction. Given two disjoint linearly ordered sets or lists, A and B, with elements

$$A_1 < A_2 < \cdots < A_m,$$

$$B_1 < B_2 < \cdots < B_n,$$

we are interested in merging them into one ordered set with $m+n$ elements by making pairwise comparisons between an element of A and an element of B. Since such a comparison can yield at most 1 bit of information, the noiseless coding theorem of information theory gives $\log_2 \binom{n+m}{m}$ as a lower bound on the number of comparisons required to perform this task, assuming that a priori all possible combinations of the elements in the final set are equally likely. This model is appropriate for the merges of a merge sort of a random list. In this paper we give an algorithm, which we will refer to as fractile insertion, whose average number of comparisons or work, $W_f(m, n)$, is upper bounded by $1.06 \log_2 \binom{n+m}{m}$. The fractile used to achieve this is the median.

The noiseless coding theorem will be a central facet of our argument, and its relevance warrants some review and explanation. In its standard form [1, p. 37], it states that if a source emits symbols from an alphabet $S_1, \cdots, S_k$ with probabilities $p_1, \cdots, p_k$ respectively, then the average length of any binary code used to encode the source must be greater than or equal to the source entropy, $H(p_1, \cdots, p_k) = \sum_{i=1}^{k} p_i \log_2 (1/p_i)$. In the present context we can regard each possible permutation of the elements as being a symbol, and for each merging

* Received by the editors April 13, 1977.

† Board of Information Sciences, University of California, Santa Cruz, Santa Cruz, California 95064.

problem nature chooses one of the combinations with uniform probability $= \binom{n+m}{m}^{-1}$. Since the result of a comparison of two elements can be coded as a 0 or 1, the outcomes of the comparisons specified by any given algorithm constitute a binary code for the source; therefore, the entropy, $\log_2 \binom{n+m}{m}$, must be a lower bound. But more importantly, the additivity of the entropy function on conditionally independent distributions enables it to serve as an effective measure of the quality of a comparison. A perfect comparison gives one bit of information; the average number of comparisons will be close to the lower bound if each of the comparisons yields close to one bit. In particular, our algorithm calls for the insertion of one element in the smaller list into the larger list, thus decomposing the problem into two smaller problems. For example, suppose A_f is inserted into B. Let the probability that A_f is less than all B_i be p_0, that it is greater than B_1 but less than B_2 be p_1, and so forth. We could write a recursion relation for the average number of comparisons required by the algorithm as

$$W_f(m, n) = (\text{Average number of comparisons required to insert } A_f) + \sum_{i=0}^{n} p_i (W_f(f-1, i) + W_f(m-f, n-i)), \tag{1}$$

since after locating A_f between B_i and B_{i+1}, the conditionally independent problems of merging $f-1$ into i and $m-f$ into $n-i$ remain.

On the other hand, determining the correct merging can be regarded as the two step process of first encoding the location of A_f in B, and then encoding for the two remaining subproblems. This interpretation permits the grouping property of the entropy function to be used to give a recursion for the information theory bound on $W_f(m, n)$ which closely resembles (1) (see [1, p. 8]). Let q_j, $j = 1, \cdots, \binom{n+m}{m}$, be the probability of the ith merge order in an exhaustive list arranged such that $q_{r_i+1}, \cdots, q_{r_i+1}$ are associated with the merge orders which have A_f between B_i and B_{i+1}. Since all merging orders are assumed equally likely $q_j = 1 \Big/ \binom{n+m}{m}$ for all j. By definition of the p_i, $p_i = \sum_{j=r_i+1}^{r_{i+1}} q_j$, $\left(r_0 = 0, r_{n+1} = \binom{n+m}{m}\right)$. The group property then permits the entropy of the complete merge to be written as the entropy of the partitioning induced by insertion of A_f plus the weighted sum of the entropies remaining in each partition:

$$H(q_1, \cdots, q_{\binom{n+m}{m}}) = H(p_0, p_1, \cdots, p_n) + \sum_{i=0}^{n} p_i H\left(\frac{q_{r_i+1}}{p_i}, \cdots, \frac{q_{r_i+1}}{p_i}\right).$$

Let

$$I(m, n) = H\left(\frac{1}{\binom{n+m}{m}}, \cdots, \frac{1}{\binom{n+m}{m}}\right)$$

be the information theory lower bound on the number of comparison needed to merge m into n. The equation then can be rewritten as

$$(2) \qquad I(m,n) = H(p_0, p_1, \cdots, p_n) + \sum_{i=0}^{n} p_i[I(f-1,i) + I(m-f, n-i)].$$

In other words, the total entropy is sum of the entropy of the distribution of A_f in B plus the weighted sum of the entropies which remain, given the position of A_f.

This identity, (2), stands independent of any algorithm and, recalling that the definition of p_i, $i = 0, \cdots, n$, depends on f, holds for any f. The proof of the bound relies heavily on the close similarity between the bound identity (2) and the recursion relation (1): the essential ingredient is showing that the insertion of a fractile element A_f can be done in very little more than $H(p_0, p_1, \cdots, p_n)$ comparisons on the average.

An algorithm: Fractile insertion. Without loss of generality, m is assumed less than n. An element A_f is compared with a sequence B_{k_i} of elements from B until A_f has been located in the B order. The algorithm is then applied to the two smaller problems of merging $A_1, \cdots, A_{f-1}$ into $B_1, \cdots, B_i$ and $A_{f+1}, \cdots, A_m$ into $B_{i+1}, \cdots, B_n$, where $B_i < A_f < B_{i+1}$. If $A_f > B_n$ then $i = n$ and only the first problem remains. If $A_f < B_1$, then $i = 0$ and only the second remains. The sequence of subscripts k_i is given as follows:

1. Let $k_1 = \lceil n \cdot f/(m+1) \rceil$. Let $\alpha = \lfloor \frac{1}{2} \log_2 (n(1+n/m)) - 1.3 \rfloor$. Let $\Delta = 2^{\alpha}$.
2. If $A_f > B_{k_i}$ then continue to set $k_{i+1} = k_i + \Delta$ until either $k_{i+1} > n$ or $A_f < B_{k_{i+1}}$. If $k_{i+1} > n$, binary insert A_f in $B_{k_i+1}, \cdots, B_n$. Otherwise binary insert A_f in $B_{k_i+1}, \cdots, B_{k_{i+1}-1}$.
3. If $A_f < B_{k_i}$ then continue to set $k_{i+1} = k_i - \Delta$ until either $k_{i+1} < 1$ or $A_f > B_{k_{i+1}}$. If $k_{i+1} < 1$, binary insert A_f in $B_1, \cdots, B_{k_i-1}$. Otherwise binary insert A_f in $B_{k_{i+1}}, \cdots, B_{k_i-1}$.

Briefly, the first test is at the proportional position of A_f in B. The test position makes hops of size Δ, either up or down, depending upon the outcome of the first test, until A_f's location is known within Δ or less. Finally A_f is binary inserted. (See Fig. 1.)

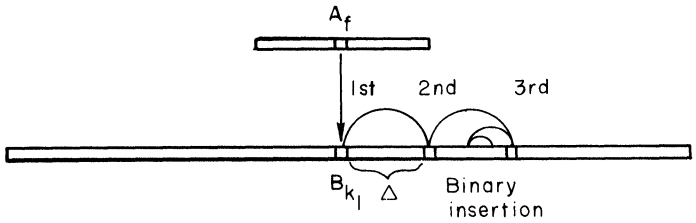

FIG. 1

When implemented as a recursive procedure, fractile insertion will require fewer than $\lceil X \rceil$ pointers to record the subdivisions, where X satisfies $m(f/(m+1))^X = 1$. Note that for $f = \lceil m/2 \rceil$, $X \approx \lceil \log_2 m \rceil$. Similarly, the amount of active store needed will depend on the particular fractile chosen.

As a simple example, suppose $f = \lceil m/2 \rceil$

$$A = \{1, 3, 8, 11\},$$

$$B = \{2, 4, 5, 6, 9, 10, 12, 16, 17, 19\}.$$

Then $k_1 = \lceil 10 \cdot \frac{2}{5} \rceil = 4$, $\alpha = \lfloor \frac{1}{2} \log_2 35 - 1.3 \rfloor = 1$, $\Delta = 2$. A_2 $(=3)$ is compared to B_4 $(=6)$; then B_2 $(=4)$. Since decreasing the B index by 2 again gives 0, A_2 is inserted by comparing with B_1 $(=2)$. The two subproblems which remain are $\{1\}$ into $\{2\}$, and $\{8, 11\}$ into $\{4, 5, 6, 9, 10, 12, 16, 17, 19\}$.

As heuristic motivation for the form of the algorithm consider the probability distribution of the location of A_f in the B list. To make the previous notation for the insertion probabilities more explicit, let $p_f(0; m, n)$ be the probability $A_f < B_1$, $p_f(n; m, n)$ the probability $A_f > B_n$, and $p_f(i; m, n)$ the probability $B_i < A_f < B_{i+1}$ for $0 < i < n$. Then

$$p_f(i; m, n) = \frac{\binom{i+f-1}{f-1}\binom{n-i+m-f}{m-f}}{\binom{n+m}{m}}, \qquad 0 \leqq i \leqq n. \tag{3}$$

By simple reorganization of the factorials, this can also be written as

$$p_f(i; m, n) = \frac{m}{m+n}\binom{m-1}{f-1}\frac{\binom{n}{i}}{\binom{m+n-1}{i+f-1}}. \tag{4}$$

For any fixed m, as $n \to \infty$, the distribution converges in law to a continuous Beta distribution of the form

$$p(X) = \frac{m!}{(f-1)!(m-f)!} X^{f-1}(1-X)^{m-f},$$

which can be seen most easily by observing that

$$\frac{(i+f-1)\cdots(i+1)(n-i+m-f)\cdots(n-i+1)}{(n+m)\cdots(n+1)} \sim \frac{1}{n+m/2}\left(\frac{i+f/2}{n+m/2}\right)\left\{\frac{n-i+[(m-f)/2]}{n+m/2}\right\}^{m-f}.$$

When $n \to \infty$ with m/n fixed, the distribution converges in law to a Gaussian with mean $((f-1)/(m-1)) \cdot n$, and variance $\frac{1}{4}n(1+n/(m-1))$. This can be derived either from the continuous Beta or directly from (3) using Gaussian approximations to the binomial distribution. In any event, the key to the success of the algorithm is that the search interval, Δ, is pegged to the standard deviation of the underlying distribution of $A_f(\Delta \approx \frac{3}{4}\sigma)$. Consequently, for large n, the initial comparisons are as efficient as on the smaller n cases, and the later comparisons are basically perfect binary insertions. As the latter begin to constitute the majority of the comparisons, over-all efficiency of each insertion becomes closer and closer to one.

Within this basic scheme there is some leeway in choosing the ratio of the search interval to the standard deviation. The term 1.3 in the expression for α was determined empirically to give the best balance between overshoot and undershoot for different n.

We will now show that when $f = \lceil m/2 \rceil$, representing the median, the average number of comparisons required by this algorithm is within 6% of the information theory bound. Hereafter, $p(i; m, n)$ denotes $p_{\lceil m/2 \rceil}(i; m, n)$. It is perhaps worth noting that, trivially, the optimum algorithm for merging one into two requires 5.2% more than the information theory bound. Thus, in a limited sense, no significantly tighter bound on all problems is possible for any algorithm. The actual worst case for the median insertion occurs at $m = 1$ and $n = 65$, where the average is 5.8% over the bound.

Since the proof is based on an intricate scaling argument, it is inconvenient to deal directly with the discrete distribution, (3). Instead we regard $p(i; m, n)$ as a function continuous in both i and n, but not m, by interpreting combinatorials of the form $\binom{x+j}{j}$ in (3), x real, j integer, as simply

$$\binom{x+j}{j} = \frac{(x+j)(x+j-1)\cdots(x+1)}{j!}.$$

By considering the number of comparisons the algorithm makes to insert $A_{\lceil m/2 \rceil}$ in position i to be defined for all real $i \in [-\frac{1}{2}, n+\frac{1}{2}]$, the average number of comparisons, $c(m, n)$, required by the algorithm to locate $A_{\lceil m/2 \rceil}$ can be viewed as a function continuous in n. The theorem is then proven based on the scaling properties of $c(m, n)$ and $h(m, n)$, the entropy of $p(i; m, n)$.

To validate the analytical proof for the actual discrete distribution, three types of potential error due to approximations implicit in this approach must be bounded. The first is approximation of the type

$$p(i; m, n) \approx \int_{i-\frac{1}{2}}^{i+\frac{1}{2}} p(i; m, n)\, di$$

which arises when a discrete sum over i is replaced by a continuous integral over i in calculating $c(m, n)$ and $h(m, n)$. The second is the possible interpolation error that occurs when the values of $c(m, n)$ and $h(m, n)$, computed only for integer n, are interpolated to define both for continuous n. Finally, there is the machine round-off and function error in the computation of the table for Region I (see proof of theorem). The first two sources of error are treated at the end of the proof. The third is deemed negligible, being of the order of 10^{-5} smaller than the margins allowed in the proof.

THEOREM. *Let $W_{\text{med}}(m, n)$ be the average number of comparisons required by the median insertion algorithm to merge m into n assuming all $\binom{n+m}{n}$ possible combinations are equally likely. Then $W_{\text{med}}(m, n) \leqq 1.06 \log_2 \binom{m+n}{m}$.*

Proof. The verification of the inequality has been carried out by dividing the table of values $W_{\text{med}}(m, n)$, $m \geqq 0$, $n \geqq 0$, into three regions. The first is a basis set

where the average number of comparisons and the efficiency of the insertion process can be evaluated numerically.

Region I: $m \leqq 40$, $n \leqq 100$. Using the above notation, (1) can be rewritten

$$W_{\text{med}}(m, n) = \sum_{i=0}^{n} p(i; m, n)(\text{comparisons made to locate } A_{\lceil m/2 \rceil} \text{ in position } i + W_{\text{med}}(\lceil m/2 \rceil - 1, i) + W_{\text{med}}(m - \lceil m/2 \rceil, n - i)).$$

This equation, along with $W_{\text{med}}(0, n) = 0$ for all $n \geqq 0$, defines $W_{\text{med}}(m, n)$. An ALGOL *W* program was written to calculate $W_{\text{med}}(m, n)$, as well as the entropy of the $p(i; m, n)$ distribution and the average number of comparisons required for the insertion. The results, when combined with the proofs used for Regions II and III, give the 1.06 $I(m, n)$ bound. For most m and n this region $W_{\text{med}}(m, n) \approx 1.03\, I(m, n)$.

Region II: $m \leqq 40$, $n > 100$. For fixed m we extend the bound by induction on n. Having calculated the efficiency of the remaining merges to be within the 1.06 bound, we need only show that the insertion efficiency satisfies

$$\frac{c(m, n')}{h(m, n')} < 1.06$$

for all $n' > 100$. The inductive strategy is to show for any n there exists an $n' \approx 2n$ such that efficiency for n implies efficiency for n'. Since we are allowing n to take on real values, this is equivalent to showing that for any n' there exists a corresponding n. Having computationally established the 6% bound for $40 \leqq n \leqq 100$, repeated use of this argument establishes the bound for all $n' > 100$.

We begin by comparing $p(i; m, n)$ with $p(i'; m, n')$ chosen to make the latter very closely the former scaled by a factor of 2.

Let $i' = ((f-1)/(m-1))n' + 2(i - ((f-1)/(m-1))n)$,

$$D_1 \triangleq \frac{d}{di} \ln \frac{p(i; m, n)}{p(i'; m, n')} = \sum_{k=1}^{f-1} \frac{1}{i+k} - \sum_{k=1}^{m-f} \frac{1}{n-i+k} - \sum_{k=1}^{f-1} \frac{1}{\frac{1}{2}\frac{f-1}{m-1}(n'-2n) + i + \frac{k}{2}} + \sum_{k=1}^{m-f} \frac{1}{\frac{1}{2}\frac{m-f}{m-1}(n'-2n) + n - i + \frac{k}{2}}$$

(5)

$$D_2 \triangleq \frac{d^2}{di^2} \ln \frac{p(i; m, n)}{p(i'; m, n')} = -\sum_{k=1}^{f-1} \frac{1}{(i+k)^2} - \sum_{k=1}^{m-f} \frac{1}{(n-i+k)^2} + \sum_{k=1}^{f-1} \frac{1}{\left(\frac{1}{2}\frac{f-1}{m-1}(n'-2n) + i + \frac{k}{2}\right)^2} + \sum_{k=1}^{m-f} \frac{1}{\left(\frac{1}{2}\frac{m-f}{m-1}(n'-2n) + n - i + \frac{k}{2}\right)^2}.$$

By making n' sufficiently large, it is clear that this second derivative, D_2, can be made negative. This will assure that the maximum of the ratio is achieved near the peaks of the distributions, where the first derivative, D_1, will be zero. The best comparison will be obtained by finding the smallest n' for which the second derivative is still negative for all i.

An upper bound to the second derivative can be obtained using an integral approximation to the sums. Specifically

$$\int_{i-\frac{1}{2}+\varepsilon}^{i+\frac{1}{2}+\varepsilon} \frac{1}{x^2}\,dx = \frac{-1}{i+\frac{1}{2}+\varepsilon}+\frac{1}{i-\frac{1}{2}+\varepsilon} = \frac{1}{i^2+2i\varepsilon+\varepsilon^2-\frac{1}{4}}.$$

If $\varepsilon > 1/(8i)$ the integral is less than $1/i^2$; if $\varepsilon = 0$ it is greater than $1/i^2$. Consequently

$$\sum_{k=1}^{f-1} \frac{1}{(i+k)^2} > \sum_{j=i+1}^{i+f-1} \int_{j=\frac{1}{2}+\varepsilon}^{j+\frac{1}{2}-\varepsilon} \frac{1}{x^2}\,dx = \int_{i=\frac{1}{2}+\varepsilon}^{i+f-\frac{1}{2}+\varepsilon} \frac{1}{x^2}\,dx = \frac{1}{(i+f-\frac{1}{2}+\varepsilon)(i+\frac{1}{2}+\varepsilon)}$$

for $\varepsilon > 1/(8(i+1))$, the largest value necessary for each of the parts. This bound can then be applied to the four sums in D_2.

$$-\sum_{k=1}^{f-1} \frac{1}{(i+k)^2} < -\frac{f-1}{\left(i+\frac{1}{2}+\frac{1}{8(i+1)}\right)\left(i+f-\frac{1}{2}+\frac{1}{i(i+1)}\right)} \triangleq B_1(i),$$

$$-\sum_{k=1}^{m-f} \frac{1}{(n-i+k)^2} < -\frac{m-f}{\left(n-i+\frac{1}{2}+\frac{1}{8(n-i+1)}\right)\left(n-i+m-f+\frac{1}{2}+\frac{1}{8(n-i+1)}\right)} \triangleq B_2(n-i),$$

(6)
$$\sum_{k=1}^{f-1} \frac{1}{\left(\frac{1}{2}\frac{f-1}{m-1}(n'-2n)+i+\frac{k}{2}\right)^2} < \frac{f-1}{\left(\frac{1}{2}\frac{f-1}{m-1}(n'-2n)+i+\frac{1}{4}\right)\left(\frac{1}{2}\frac{f-1}{m-1}(n'-2n)+i+\frac{f}{2}-\frac{1}{4}\right)} \triangleq B_3(i),$$

$$\sum_{k=1}^{m-f} \frac{1}{\left(\frac{1}{2}\frac{m-f}{m-1}(n'-2n)+n-i+k/2\right)^2} < \frac{m-f}{\left(\frac{1}{2}\frac{m-f}{m-1}(n'-2n)+n-i+\frac{1}{4}\right)\left(\frac{1}{2}\frac{m-f}{m-1}(n'-2n)+n-i+\frac{m-f}{2}+\frac{1}{4}\right)} \triangleq B_4(n-i),$$

$$D_2 < B_1(i)+B_2(n-i)+B_3(i)+B_4(n-i).$$

Leaving aside the cases $m = 1, 2$, and 3 for later attention, we now consider $m \geqq 4$. The sum of the four right-hand sides in (6) is a bound on D_2. The behavior of this bound is most easily discerned by grouping the $f-1$ terms, $B_1(i)$ and $B_3(i)$ and the $m-f$ terms, $B_2(n-i)+B_4(n-i)$. The form of $B_1(i)+B_3(i)$ is

$$B_1(i)+B_3(i)=\frac{ai+b}{c_1(i)}, \quad \text{where } c_1(i)\approx i^4.$$

(The reciprocal correction terms have been temporarily suppressed, since they do not affect the important behavior.) Similarly

$$B_2(n-i)+B_4(n-i)=\frac{c(n-i)+d}{c_2(n-i)}, \quad \text{where } c_2(n-i)\approx (n-i)^4.$$

Because of the $O(i^4)$ and $O((n-i)^4)$ terms in the denominators, $|B_1(i)+B_3(i)|>|B_2(n-i)+B_4(n-i)|$ for $i \leqq ((f-1)/(m-1))n$ whereas $|B_2(n-i)+B_4(n-i)|>|B_1(i)+B_3(i)|$ for $i \geqq n/2$. Consequently, by making $B_1(i)+B_3(i)\leqq 0$ for $i \leqq n/2$ and $B_2(n-i)+B_4(n-i)\leqq 0$ for $i \geqq ((f-1)/(m-1))n$, the sum, D_2, will be negative for all i.

For n' sufficiently large, all the numerator coefficients, a, b, c, and d, will be negative. For smaller n', $n'>2n$, a and c are positive while b and d remain negative. It follows that if $B_1(i)+B_3(i)\leqq 0$ at $i=n/2$, it is negative for $i<n/2$. Likewise, $B_2(n-i)+B_4(n-i)\leqq 0$ for $i=((f-1)/(m-1))n$ implies that it is negative for $i>((f-1)/(m-1))n$.

Equating the denominators of $B_1(i)$ and $B_3(i)$ in (6) for $i=n/2$ gives the first required condition for n':

$$\left(\frac{1}{2}\frac{f-1}{m-1}(n'-2n)+i+\frac{1}{4}\right)\left(\frac{1}{2}\frac{f-1}{m-1}(n'-2n)+i+\frac{f}{2}-\frac{1}{4}\right)$$
$$\geqq\left(i+\frac{1}{2}+\frac{1}{8(i+1)}\right)\left(i+f-\frac{1}{2}+\frac{1}{8(i+1)}\right) \quad \text{at } i=\frac{n}{2}.$$

The denominators of $B_2(n-i)+B_4(n-i)$ give second condition

$$\left(\frac{1}{2}\frac{m-f}{m-1}(n'-2n)+n-1+\frac{1}{4}\right)\left(\frac{1}{2}\frac{m-f}{m-1}(n'-2n)+n-i+\frac{m-f}{2}+\frac{1}{4}\right)$$
$$\geqq\left(n-i+\frac{1}{2}+\frac{1}{8(n-i+1)}\right)\left(n-i+m-f+\frac{1}{2}+\frac{1}{8(n-i+1)}\right) \quad \text{at } i=\frac{f-1}{m-1}n.$$

Substituting in the value of i and simplifying, the first becomes

$$\frac{1}{4}(n'+1)(n'+m)\geqq\left(n+1+\frac{1}{2n}\right)\left(n+m+\frac{1}{2n}\right) \tag{7}$$

for m odd, and $n'=(n''-2n)(m-f)/(2(f-1))+2n$, where n'' satisfies

$$\frac{1}{4}(n''+1)(n''+m-1)\geqq\left(n+1+\frac{1}{2n}\right)\left(n+m-1+\frac{1}{2n}\right) \tag{8}$$

for m even. The second inequality requires that

$$(9)\quad \frac{1}{4}\left(n'+\frac{m-1}{2(m-f)}\right)\left(n'+m-1+\frac{m-1}{2(m-f)}\right)\geqq\left(n+1+\frac{1}{2n}\right)\left(n+m+\frac{1}{2n}\right).$$

Note that this is virtually identical to the n' required to double the search interval Δ.

For m odd, then, the maximum of the ratio, $(p(i;m,n))/(p(i';m,n'))$, must occur at $i=n/2$, $i'=n'/2$. For m even, the maximum occurs very near $i=((f-1)/(m-1))n$, as can be seen by consideration of the zero of the first derivative in (5). Rather than seeking the exact maximum, we wish to show that must be less than $(p(n/2;m,n))/(p(n'/2;m,n'))$. This is even a possibility only because, for even m, when $i=n/2$, $i'=((f-1)/(m-1))(n'-2n)+n<n'/2$. Of course, as m becomes larger, the slight asymmetry becomes less important.

Let $i''=(n'/n)i$. Let

$$R(m,n,n')\equiv\max_i\frac{p(i;m,n)}{p(i';m,n')}.$$

Now

$$\frac{p(i;m,n)}{p(i'';m,n')}=\frac{p(i;m,n)}{p(i';m,n')}\quad\text{at } i=\frac{f-1}{m-1}n.$$

Furthermore

$$\frac{d^2}{di^2}\left(\ln\frac{p(i;m,n)}{p(i'';m,n'')}-\ln R(m,n,n')\right)$$

$$=\sum_{k=1}^{f-1}\frac{1}{(i+k\cdot(n/n'))}2-\sum_{k=1}^{f-1}\frac{1}{(i+k)^2}+\sum_{k=1}^{m-f}\frac{1}{(n-i+k\cdot(n/n'))^2}$$
$$-\sum_{k=1}^{m-f}\frac{1}{(n-i+k)^2}>0.$$

Since this difference achieves a minimum of virtually zero near $i=((f-1)/(m-1))n$, it is not too difficult to show that it is positive at $i=n/2$. Thus

$$R(m,n,n')\leqq\frac{p(n/2;m,n)}{p(n'/2;m,n')}$$

for both even and odd m.

To evaluate $R(m,n,n')$ is now quite easy. Starting from (4), we have

$$R(m,n,n')<\frac{m+n'}{m+n}\frac{\binom{n}{n/2}\binom{n'+m-1}{n'/2+f-1}}{\binom{n+m-1}{n/2+f-1}\binom{n'}{n'/2}}.$$

Feller [2, pp. 179–182] gives a bounded approximation to the central binomial

coefficient, namely

$$\binom{n}{n/2} = 2^n \frac{2}{\sqrt{n}} \frac{1}{\sqrt{2\pi}} e^{-\varepsilon}, \quad \text{with} \quad \frac{1}{4n} - \frac{1}{20n^3} < \varepsilon < \frac{1}{4n} + \frac{1}{360n^3}. \tag{10}$$

Using this for the coefficients in the above inequality, we obtain

$$R(m, n, n') < \frac{m+n'}{m+n} \frac{\sqrt{n+m-1}}{\sqrt{n'+m-1}} \frac{\sqrt{n'}}{\sqrt{n}} \exp\left[\frac{1}{4}\left(\frac{1}{n} - \frac{1}{n'} - \frac{1}{n+m-1} + \frac{1}{n'+m-1}\right) + \delta\right]$$

where $|\delta| < 1/(5n^3)$ and can be neglected for $n \geqq 40$. This holds for m odd. For m even the right-hand side must be multiplied by a factor less than $((n'+1)/(n'+2))((n+2)/(n+1))$, to compensate for the slightly noncentral evaluation of two of the coefficients. For m odd (7) and (9) are the same. Using equality in (7) to define n' in the bound above gives

$$R(m, n, n') < 2\left(1 - \frac{m+3}{(n'+m-1)(n'+1)}\right)^{1/2} \left(1 + \frac{m + \frac{m+2}{2n}}{n(n+m+\frac{1}{2}n)}\right)^{1/2} \cdot \exp\left[\frac{1}{4}\left(\frac{1}{n} - \frac{1}{n'} - \frac{1}{n+m-1} + \frac{1}{n'+m-1}\right)\right].$$

Over the region $m \leqq 40$, $n \geqq 40$ this evaluates to

$$R(m, n, n') < 2(1.008).$$

For m even the upper bounding is slightly less tight because of the additional factor but still gives

$$R(m, n, n') < 2(1.022). \tag{11}$$

We thus have a very close comparison between $p(i; m, n)$ and the scaled distribution $2p(i'; m, n')$. Because of the negative second derivative (6), we know that $p(i; m, n) - 2p(i'; m, n')$ is positive in an interval containing $i = n/2$, negative in the tails and at most a probability of $\frac{1}{2}(0.022)$ moves from the central region to the tails.

If the scaling were perfect, and the search interval Δ' corresponding to n' were exactly 2Δ, both the entropy and the average number of comparisons would be increased by one, implying perfect incremental efficiency. Our next object is to show that the actual case is not sufficiently different to destroy the 6% bound.

Let $w(i; m, n)$ be the number of comparisons made by the algorithm in locating $A_{\lceil m/2 \rceil}$ in the ith position in B. In what follows we will assume that for the n' defined by (7), (8), and (9), $\Delta' = 2\Delta$, so that $w(i'; m, n') = w(i; m, n) + 1$. This will be the case at all but a few n' near the discontinuities in Δ as a function of n.[1]

[1] A simplified expression has been used in the algorithm for defining α and Δ. This leads to discrepancies of less than 1 in the location of the discontinuities of $\Delta'/2$ and Δ; they can be accounted for within the framework of interpolation of the computed values of $c(m, n)$.

Consider the first variation of $c(m,n)-h(m,n)$, labeled δV, due to an additive variational function $\delta p(i)$:

$$\delta V \triangleq \int \frac{d}{dp}[(w(i;m,n)+\log_2 p(i;m,n))p(i;m,n)]\,\delta p(i)\,di,$$

$$\delta V = \int (w(i;m,n)+\log_2 p(i;m,n)+\log_2 e)\,\delta p(i)\,di,$$

$$\delta V < \int_{\delta p(i)>0} \max_{\delta p(i)>0} (w(i;m,n)+\log_2 p(i;m,n)+\log_2 e)\,\delta p(i)\,di$$

$$+ \int_{\delta p(i)<0} \min_{\delta p(i)<0} (w(i;m,n)+\log_2 p(i;m,n)+\log_2 e)\,\delta p(i)\,di.$$

Since any $\delta p(i)$ which preserves a probability density satisfies

$$\int \delta p(i)\,di = 0 \qquad \left(\text{implying } \int_{\delta p(i)>0} \delta p(i)\,di + \int_{\delta p(i)<0} \delta p(i)\,di = 0\right),$$

$$\delta V < \int_{\delta p(i)>0} \delta p(i)\,di\Big[\max_{\delta p(i)>0} (w(i;m,n)+\log_2 p(i;m,n)$$

$$- \min_{\delta p(i)<0} (w(i;m,n)+\log_2 p(i;m,n))\Big].$$

The $\delta p(i)$ of interest are those which carry $p(i;m,n)$ into $2p(i';m,n')$. By the previous analysis the region $\delta p(i)>0$ is in the tails, whereas $\delta p(i)<0$ is an interval about the center of the distribution. Using the upper bound

$$\frac{\left|i-\frac{f}{m+1}n\right|}{\Delta}+2+\log_2\Delta$$

for $w(i;m,n)$ in the maximum and the lower bound

$$\frac{\left|i-\frac{f}{m+1}n\right|}{\Delta}+1+\log_2\Delta$$

in the minimum yields

$$\delta V < \int_{\delta p(i)>0} \delta p(i)\,di\Big[1+\max_{j\geq k\geq (f/(m+1))n}\Big(\frac{j}{\Delta}+\log_2 p(j;m,n)-\frac{k}{\Delta}$$

$$-\log_2 p(k;m,n)\Big)\Big].$$

Because $(d^2/di^2)(-\log_2 p(i;m,n))$ increases monotonically from its central value as i goes to the tails, the growth of $-\log_2 p(i;m,n)$ is greater than quadratic. Consequently,

$$\delta V < \int_{\delta p(i)>0} \delta p(i)\,di\Big[1+\max_{j>0}\Big(\frac{j}{\Delta}-\frac{1}{2}\gamma j^2\Big)\Big] \tag{12}$$

where γ is $(d^2/di^2)(-\log_2 p(i; m, n))$ evaluated at the central point. The maximum occurs at $j = 1/(\gamma\Delta)$ and has value $\frac{1}{2}(1/(\gamma\Delta^2))$. Using a convex variational path of the form $p_\alpha(i; m, n) = (1-\alpha)p(i; m, n) + \alpha 2p(i'; m, n')$, $0 \leqq \alpha \leqq 1$, will guarantee that, from the sums in (5) as bounded in (6),

$$\gamma > 16\frac{m-1}{n'(n'+m)}\left(\frac{n'}{n'+1}\right)^2 \log_2 e.$$

Now $\Delta = \frac{1}{2}\Delta' \geqq \frac{1}{2}(1/4.93)\sqrt{n'(n'+m)/m}$. Substituting into (12) with $n' > 100$ we find that

$$\delta V < \int_{\delta p(i)>0} \delta p(i)\, di\left[1 + \frac{1}{2}\frac{4(4.93)^2}{16(\log_2 e)}\frac{m}{m-1}\left(\frac{101}{100}\right)^2\right]. \tag{13}$$

Using (11) we obtain

$$\delta V < \tfrac{1}{2}(0.022)(2.44) < 0.03. \tag{14}$$

This is sufficient to enable completion of the proof for this region.

The assumption $m \geqq 4$ can now be removed as follows: For $m = 1$ the distribution is uniform for all n and thus by setting $n' = 2n + 1$, perfect scaling is achieved. For $m = 2$, we set $n' = 2n + \frac{3}{2}$. Since $i' = \frac{1}{3}n' + 2i - \frac{2}{3}n$,

$$\frac{p(i; m, n)}{p(i'; m, n)} = \frac{n+1-i}{(n+1)(n+2)} \cdot \frac{(n'+1)(n'+2)}{n'+1-i} = \frac{(n+\frac{7}{4})(n+\frac{5}{4})}{(n+2)(n+1)} \cdot 2,$$

a constant, for $i \in [-\frac{1}{2}, n+\frac{1}{2}]$. Compared to perfect scaling, a probability less than

$$\left(\frac{(n+\frac{7}{4})(n+\frac{5}{4})}{(n+2)(n+1)} - 1\right) \cdot 1 = \frac{3}{16}\frac{1}{(n+1)(n+2)}$$

has been moved to the region $i' \in [2n+\frac{3}{2}, 2n+2]$. For $m = 2$, $\frac{2}{3}n/\Delta < 10$, and so the most this could increase the average number of comparisons for $n \geqq 40$ is

$$10 \cdot \frac{3}{16}\frac{1}{(41)(42)} = 0.0011$$

which is stronger than the 0.03 of (14). Finally, for $m = 3$ the general argument holds but the tighter 1.008 figure applicable for m odd must be used in (14) (see (11)). Specifically, (14) becomes

$$\delta V < \frac{1}{2}(0.008)\left(\frac{m}{m-1}2.44\right) = \frac{1}{2}(0.008)\left(\frac{3}{2} \cdot 2.44\right) < 0.03.$$

We have shown that for any $1 \leqq m \leqq 40$ and any $n \in [40, 100]$ there exists an n', the minimum n' satisfying (8) and (9), which gives almost perfect scaling by a factor of 2. From the computed values of $h(m, n)$ and $c(m, n)$, for the

corresponding n'

$$\frac{c(m,n')}{h(m,n')} \leqq \frac{c(m,n)+1.03}{h(m,n)+1} < 1.06.$$

However, the relationship between n and n' is continuous, monotone, and invertible. Consequently for any $100 < n' < 200$ there exists a corresponding n and the above inequality obtains. All bounds used grow tighter as n increases, and therefore, by repeated use of the same scaling argument

$$\frac{c(m,n)}{h(m,n)} < 1.06 \quad \text{for all } 1 \leqq m \leqq 40, \quad n > 100.$$

Since the insertions are all adequately efficient, it follows immediately from recursive formulas for the work and the information theory bound, (1) and (2), that $W_{\lceil m/2 \rceil} < 1.06\, I(m,n)$ for all $1 \leqq m \leqq 40$, $n \geqq 1$.

Region III: $n \geqq m > 40$. Again we need only show that all insertions which have not been computed are efficient. For this region the underlying principle is very simple: for any merge in this parameter range, $p(i; m, n)$ is effectively a Gaussian distribution; the efficiency of the insertion is affected only by the ratio between the search interval Δ and the standard deviation, σ, of the Gaussian distribution. In computing the cases $m = 39$ and 40, $40 \leqq n \leqq 100$, we have exhausted all possible ratios allowed by the definition of Δ, as well as seeing the maximum effect of the m even asymmetry. Thus to understand an insertion with parameters m' and n', we need only look at the $m = 39$ and 40 cases with the same Δ/σ ratio.

The form of the proof is basically the same as that for Region II, although here, rather than using an induction with each step adding a scale factor of 2, we make the comparison directly with the computed case, using whatever power of 2 is necessary to give correct scaling down to the computed range. Specifically, for any $40 < m' \leqq n'$ we find a $40 \leqq n \leqq 100$ and an integer $k \geqq 0$ such that

$$\frac{(m'+n'-1)(n'+1)}{(m'-1)} = 4^k \frac{\left(m-1+n+\frac{1}{2n}\right)\left(n+1+\frac{1}{2n}\right)}{(m-1)} \tag{15}$$

with $m = 39$ or 40.

This amounts to finding a computed case with $\sigma'^2 = 4^k \sigma^2$. Note that, by choosing k properly, an n in the range 40 to 100 satisfying (15) can always be found. In almost all cases this leads to $\Delta' = 2^k \Delta$. (As before, at the discontinuities of Δ as a function of n, interpolation of the computed data is required.)

We then wish to show that with $i' = ((f'-1)/(m'-1))n' + 2^k(i - ((f-1)/(m-1))n)$

$$\frac{p(i; m, n)}{p(i'; m', n')} \approx 2^k.$$

The derivation is exactly analogous to that for Region II:

$$\frac{d^2}{di^2}\ln\frac{p(i;m,n)}{p(i';m',n')}$$

$$<-\frac{f-1}{\left(i+\frac{1}{2}+\frac{1}{8(i+1)}\right)\left(i+f-\frac{1}{2}+\frac{1}{8(i+1)}\right)}$$

$$-\frac{m-f}{\left(n-i+\frac{1}{2}+\frac{1}{8(n-i+1)}\right)\left(n-i+m-f+\frac{1}{2}+\frac{1}{8(n-i+1)}\right)}$$

$$+\frac{f'-1}{\left(\frac{1}{2^k}\frac{f'-1}{m'-1}n'-\frac{f-1}{m-1}n+i+\frac{1}{2^{k+1}}\right)\left(\frac{1}{2^k}\frac{f'-1}{m'-1}n'-\frac{f-1}{m-1}n+i+\frac{f'-\frac{1}{2}}{2^k}\right)}$$

$$+\frac{m'-f'}{\left(\frac{1}{2^k}\frac{m'-f'}{m'-1}n'-\frac{m-f}{m-1}n+(n-i)+\frac{1}{2^{k+1}}\right)\left(\frac{1}{2^k}\frac{m'-f'}{m'-1}n'-\frac{m-f}{m-1}n+(n-i)+\frac{m'-f'+\frac{1}{2}}{2^k}\right)},$$

Again we choose n' large enough to guarantee that the sum of the $f-1$ term and the $f'-1$ term is negative for $i \leqq n/2$, and the sum of the $m-f$ term and the $m'-f'$ term are negative for $i \geqq n/2$.[2] (Note that because $f' \geqq f$ and $m'-f' \geqq m-f$, the constant terms in the denominators of the primed terms must be relatively larger than in the Region II proof, which has only the effect of making the entire second derivative more negative off center.) Substituting $i = n/2$, we need

$$\frac{f-1}{\left(\frac{n}{2}+\frac{1}{2}+\frac{1}{8(n/2+1)}\right)\left(\frac{n}{2}+f-\frac{1}{2}+\frac{1}{8(n/2+1)}\right)} \geqq \frac{f'-1}{\left(\frac{1}{2^k}\frac{f'-1}{m'-1}n'+\frac{1}{2^{k+1}}\right)\left(\frac{1}{2^k}\frac{f'-1}{m'-1}n'+\frac{f'-\frac{1}{2}}{2^k}\right)},$$

$$\frac{(n'+1)(n'+m'-1)}{m'-1} \geqq 4^k\frac{\left(n+1+\frac{1}{2n}\right)\left(n+m-1+\frac{1}{2n}\right)}{m-1},$$

where we have assumed $2(f-1)=m-1$ and $2(f'-1)=m'-1$. The $m-f$ and $m'-f'$ lead to the same inequality. With equality, this is, of course, equation (15).

[2] For simplicity we will deal explicitly only with m' odd in this region. The even cases require minor adjustments analogous to those for Region II. As the effect there was greatest at $m=4$, it is here much smaller and, in any event, less than the $m=40$ asymmetry.

The central ratio again gives a bound on the amount of probability moving from the center to the tails. From (4) and (10),

$$\frac{p(n/2;m,n)}{p(n'/2;m',n')} \leqq \frac{m}{m+n}\frac{m'+n'}{m'}\frac{\binom{m-1}{f-1}}{\binom{m'-1}{f'-1}}\frac{\binom{n}{n/2}}{\binom{n'}{n'/2}}\frac{\binom{n'+m'-1}{n'/2+f'-1}}{\binom{n+m-1}{n/2+f-1}}$$

$$\leqq \frac{m}{m+n}\frac{m'+n'}{m'}\frac{\sqrt{m'-1}\sqrt{n'}\sqrt{n+m-1}}{\sqrt{m-1}\sqrt{n}\sqrt{n'+m'-1}}$$

$$\cdot \exp\left[\frac{1}{4}\left(-\frac{1}{m-1}-\frac{1}{n}+\frac{1}{n+m-1}+\frac{1}{m'-1}+\frac{1}{n'}-\frac{1}{n'+m'-1}\right)\right].$$

Terms less than $1/(20\,m^3)$ in the exponent have been dropped. Equation (15) can then be used in the first factor.

$$\frac{m}{m+n}\frac{\sqrt{n+m-1}}{\sqrt{m-1}\sqrt{n}}\frac{m'-1}{m'}\frac{m'+n'}{m'+n'-1}\sqrt{\frac{n'}{n'+1}}\left(\frac{\sqrt{n'+1}\sqrt{m'+n'-1}}{\sqrt{m'-1}}\right)$$

$$= 2^k\frac{m}{m-1}\frac{((n+m-1)(n+m-1+1/(2n)))^{1/2}}{m+n}\left(\frac{n+1+1/(2n)}{n}\right)^{1/2}$$

$$\cdot\frac{m'-1}{m'}\frac{m'+n'}{m'+n'-1}\sqrt{\frac{n'}{n'+1}}.$$

With $m = 39$ the ratio can be made largest by letting m', n', and n go to infinity.

$$\frac{p(n/2;m,n)}{p(n'/2;m',n')} \leqq 2^k\left(1+\frac{1}{m-1}\right)\exp\left(-\frac{1}{4(m-1)}\right) \leqq 2^k\left(1+\frac{3}{4}\frac{1}{m-1}\right) \leqq 2^k(1.02).$$

The variational argument leading to (12) again holds. Here

$$\gamma > 4\frac{(m-1)\log_2 e}{(n+1+1/(2n))(m+n-1+1/(2n))}, \qquad \Delta > \left(\frac{n(n+m)}{m}\right)^{1/2}\frac{1}{4.93}.$$

The bound (14) becomes

$$< \frac{1}{2}(0.02)\left[1+\frac{(4.93)^2}{\log_2 e}\frac{(n+1+1/(2n))(m+n-1+1/(2n))}{4n(n+m)}\frac{m}{m-1}\right]$$

$$< \frac{1}{2}(0.02)(3.2) = 0.032.$$

This is the maximum amount of waste over perfect scaling that occurs for any m' and n'. Thus

$$c(m',n')-h(m',n') < c(39,n)-h(39,n)+0.032.$$

For all $40 \leqq n \leqq 100$ the right side is less than 0.256. Since the n corresponding to m' and n' lies in the range 40 to 100, $c(m',n')-h(m',n') < 0.256$ for all $40 < m' \leqq n'$. We have actually computed all cases $m \leqq 50$, $n \leqq 100$. All unverified

cases have $h(m', n') > 4.35$. Therefore

$$\frac{c(m', n')}{h(m', n')} < 1.06 \quad \text{for all } 40 < m' \leqq n'$$

and, moreover, since the entropy goes to infinity as n' goes to infinity,

$$\frac{c(m', n')}{h(m', n')} \sim 1.$$

As before, this implies that

$$W_{\lceil m/2 \rceil}(m, n) < 1.06 I(m, n) \quad \text{all } 40 < m \leqq n.$$

The only remaining detail is the use of $p(i; m, n)$ as continuous function of both i and n when it is actually discrete. For any fixed m and n the error is bounded using a Taylor series expansion in the ith interval

$$\left| p(i; m, n) - \int_{i-1/2}^{i+1/2} p(j; m, n)\, dj \right| < \frac{1}{16} \max_{i-1/2 < j < i+1/2} \left| \frac{d^2}{di^2} p(i; m, n) \right|_{i=j}.$$

Now

$$\frac{d^2}{di^2} p(i; m, n) = p(i; m, n)\left(\frac{d^2}{di^2} \ln p(i; m, n) + \left(\frac{d}{di} \ln p(i; m, n) \right)^2 \right).$$

Because this second derivative is negative in the center and positive in the tails, the effective central ratio for the actual discrete distributions is slightly larger than for the continuous approximation. However, using the expressions in (6) and recognizing that at the center $(d/di) \ln p(i; m, n) = 0$, the error in the ratio is less than

$$\frac{\frac{1}{16} p(n/2; m, n) \frac{d^2}{di^2} \ln p(i; m, n) \Big|_{i=n/2}}{p(n/2; m, n)} < \frac{m-1}{4n(n+m)} < 0.003$$

for all unverified cases.

Similarly, in implicitly defining an n, $40 \leqq n \leqq 100$, which corresponds to a larger n' we have tacitly assumed that the values of $c(m, n) - h(m, n)$ computed numerically for integer m and n can be accurately interpolated in n. By bounding $(\partial^2/\partial n^2)(c(m, n) - h(m, n))$, the interpolation error in any interval can be shown to be less than $\frac{1}{4}\sqrt{m}/((n+m)n)$, again negligible. This completes the proof.

In essence, the large subdividing insertions are done very efficiently, and they require only a small fraction of the total number of comparisons to be made. Consequently the algorithm's over-all performance is really determined by performance on the short merges, and this has been calculated.

Discussion of the result. Our concern here has been with an average number of comparisons measure. Although the demonstrated upper bound is within 6% of the information theory lower bound, for most problems the average number required is within 3% of $I(m, n)$. By the very nature of the measure, there is every reason to believe that the true asymptote is about 3%. Since any large problem gets broken into a spectrum of smaller problems, there is a blending and

smoothing effect which will utimately eliminate fluctuations in performance.

While establishing this tight bound on the average number of comparisons is of interest by itself, additional insights into the problem is provided by the method used:

First, large problems can be done with the almost same efficiency as small problems. Thus if some specialized hardware or a super algorithm were developed to give excellent speed on merges with say, $m < 10$, our algorithm enables that speed to be reflected in problems of arbitrary size.

Secondly, large problems can be done well only if small problems can be done well. Although it may seem trivial, this converse statement is in some ways more subtle than the original. Conceivably, any algorithm that decomposes the problem from the beginning is subdividing the tree of possibilities in a way that prevents "the perfect algorithm" from working. However, our technique breaks the problem down to problems of $m = k$ using a fraction roughly equal to $(1+\frac{1}{2}\log k)/k$ of the total work. To make $m = 2000$ requires only 0.003 of the total comparisons. Even ignoring the fact that the number of bits of information gained is virtually the same as the number of comparisons, this is so small as to be almost negligible. Therefore, given that this decomposition information is essentially free, "the perfect algorithm" has to be able to do $m \approx 2000$ problems well or it cannot do larger problems well either.

Although for the sake of proving the bound the median element, $A_{\lceil m/2 \rceil}$, was inserted, any fractile element can be used. The insertion technique is effective when the distribution is normal with variance $-\frac{1}{4}n(1+n/m)$. Using Gaussian approximations to the binomial coefficients in (4), it can be shown that this is the variance of the limiting distribution of every fractile. The only problem is when the element to be inserted is too near the end of the list for the normal approximation to be accurate. The first comparison of A_f should be with B_{k_1}, $k_1 = \lceil n \cdot f/(m+1) \rceil$. So long as

$$\min(k_1, n-k_1) > \frac{3(n+m)}{2\sqrt{m}}$$

the insertion should be very efficient. Near the ends, the distribution is more accurately given by an exponential, requiring a slightly different testing strategy. The discussion of the Hwang–Lin algorithm will return to this question.

As with any sophisticated technique that aims at minimization of one measure, there is the danger that the computation of the best next step dominates the useful computation. There is some leeway in the choice of α and Δ that should help this problem, however. Rather than calculating $\alpha = \lfloor \frac{1}{2}\log_2(n(1+n/m)) - 1.3 \rfloor$ and $\Delta = 2^\alpha$, one can instead straightforwardly set

$$\Delta = \text{ROUND}\left(\frac{n+m}{4\sqrt{m}}\right)$$

and, if convenient, use a binary insertion procedure that favors the middle. This combination has been shown by computation to be as good and perhaps better on small problems, and its asymptotic properties should be similar. It should also be noted that Δ is computed only once for every $2+\alpha$ comparisons on the average.

Even though the number of comparisons required by the fractile insertion algorithm can vary from problem to problem, the average measure used here by no means represents an unnaturally skewed weighting of the comparison tree nodes. For example, the assumption that both A and B have elements drawn from a uniform distribution on [0, 1] should not greatly affect the efficiency of the algorithm in performing the merge. On the other hand, the algorithm is not very good if the minimax (minimizing the maximum number of comparisons required) criterion is used.

Minimax properties. An exhaustive investigation of small problems reveals that the maximum number of comparisons required by the fractile insertion algorithm can be as much as 25% to 30% over the information theory bound. In contrast to the Hwang–Lin algorithm, the region where the difference is most pronounced is near $n \approx m$. The explanation for this weakness on approximately equal lists is easily found, however. It is known that for $m \geqq 6$([4, § 5.3.2]),

$$M(m, m+d) = 2m + d - 1, \qquad 0 \leqq d \leqq 4$$

where $M(m, n)$ is the lower bound for the best minimax algorithm merging m into n; and it is conjectured that the equality holds for any fixed d as $m \to \infty$. If this is true, then there is no difference in the maximum work that may have to be done if an (m, m) problem is subdivided by the fractile algorithm into two balanced problems or two slightly unbalanced problems. That is:

$$M(\lceil m/2 \rceil - 1, \lceil m/2 \rceil) + M(m - \lceil m/2 \rceil, m - \lceil m/2 \rceil)$$
$$= M(\lceil m/2 \rceil - 1, \lceil m/2 \rceil - d) + M(m - \lceil m/2 \rceil, m - \lceil m/2 \rceil + d) \quad \text{for } |d| \leqq 4.$$

Thus, in effect, the comparisons spent by the algorithm in determining the actual d do it no good against the minimax criterion. On the other hand, the $I(m, n)$ function does have the necessary convexity and, relative to the average metric, the same comparisons do pay off.

For unequal lists, the fractile insertion and the Hwang–Lin algorithms appear to have about the same maximum number of comparisons.

Sorting. It is well known that binary insertion and straight two way merging when used as sorting techniques are both asymptotically efficient, requiring only $n \log_2 n$ comparisons as $n \to \infty$. The underlying reason is that both are repeated merging schemes designed so that all the merging problems encountered can be done optimally by the associated merging procedures: Binary insertion is optimal for large n; the standard "pop the top", or two way merge ([4, p. 160]), is optimal when $m \approx n$. There may be times, however, when these list size constraints present awkward data handling problems. The implication of our theorem, combined with the noiseless coding theorem, is that any repeated merge sorting scheme based on the fractile insertion algorithm will require less than $1.06 \log_2 (n!)$ comparisons on the average to sort n elements. The proof is simply that every merge uses less than 1.06 times the corresponding information theory bound; by the noiseless coding theorem, the information theory bound for the whole problem is just the sum of the bounds for each part. But the average number of comparisons is likewise just the sum of the parts, and the result follows.

The Hwang–Lin algorithm. The best merging algorithm previously known for handling problems with ratios of m/n in the range between 0 and 1 is the generalized binary algorithm introduced by F. K. Hwang and S. Lin [3]. It has been shown to give minimax performance far superior to either straight binary insertion or "pop the top" merging. To use our information theory bound reference, the minimax upper bound is at worst only about 15% over $I(m, n)$. Although we have as yet been unable to prove a bound for arbitrarily large lists, on small problems the Hwang–Lin algorithm appears to be about as good if not slightly better than the fractile insertion algorithm even in terms of the average measure. In a way, it is actually fractile insertion adapted to the ends.

Briefly, the steps of their algorithm are:

1. Assuming $m \leqq n$, let $\alpha = \lfloor \log_2 (n/m) \rfloor$ and $x = n - 2^\alpha + 1$.
2. Compare A_m with B_x. If $A_m > B_x$, binary insert A_m in $B_{x+1}, \cdots, B_n$. Emit elements $A_m, B_{y+1}, \cdots, B_n$, where $B_y < A_m < B_{y+1}$, and apply the algorithm to $A_1, \cdots, A_m$ and $B_1, \cdots, B_y$.
3. If $A_m < B_x$, emit $B_x, \cdots, B_n$ and apply the algorithm to $A_1, \cdots, A_m$ and $B_1, \cdots, B_{x-1}$.

Given this recursive definition, it is easy to establish a recursion relation for the average number of comparisons required. This function was evaluated by computer for $m \leqq 25$, $n \leqq 100$. To summarize the results of a comparison of this function with the corresponding one for median insertion over the same region, neither one is uniformly better. Median insertion is slightly better for some problems, Hwang–Lin for others. On the average (that is, on a problem chosen at random), the Hwang–Lin is about half a percent better. The unresolved question is whether it continues to be so for large lists.

Suppose the fractile insertion algorithm were used with $f = m$. The first comparison would be between A_m and B_k,

$$k = \left\lceil n \cdot \frac{m}{m+1} \right\rceil = \left\lceil n \cdot \left(1 - \frac{1}{m+1}\right) \right\rceil \approx n + 1 - \lfloor n/m \rfloor \quad \text{for large } n \text{ and } m.$$

If $\Delta > \lfloor n/m \rfloor$ and the first comparison indicates $A_m > B_k$, A_m will be binary inserted in $B_{n+1-\lfloor n/m \rfloor + 1}, \cdots, B_n$. Otherwise it will be compared with $B_{n+1-\lfloor n/m \rfloor - \Delta}$. It does roughly the same thing as the Hwang–Lin method.

A closer look at distribution of location of A_m in B will show why this type of algorithm should give good average results. Let $p(0)$ be the probability $B_n < A_m$ and $p(i)$ the probability $B_{n-i} < A_m < B_{n-i+1}$.

$$\frac{p(i)}{p(i+1)} = \frac{\binom{n-i+m-1}{m-1} \Big/ \binom{n+m}{m}}{\binom{n-i-1+m-1}{m-1} \Big/ \binom{n+m}{m}} = \frac{(n-i+m-1)\cdots(n-i+1)}{(n-i-1+m-1)\cdots(n-i)}$$

$$= \frac{n-i+m}{n-i}.$$

So long as $i \ll n$, $p(i)/p(i+1) \approx (n+m)/n$. Thus an excellent approximation to $p(i)$ is

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

an exponential distribution. When $n = m$, for example, $p(0) = \frac{1}{2}$. It is not surprising that Hwang–Lin does well on equal length lists. Most of the comparisons are close to perfect! In general the probability that a first comparison of A_m with B_{n-i} will show $B_{n-i} < A_m$ is $q = \sum_{k=0}^{i} p(i) = 1-(n/(n+m))^i$, and the numbers of bits thereby obtained is $-q \log_2 q - (1-q)\log_2(1-q)$. Similar expressions involving sums can be derived for the binary insertion comparisons. Although the $\alpha - \lfloor \log_2 (n/m) \rfloor$ and $x = n - 2^\alpha + 1$ do a good job of balancing probabilities in most cases, the efficiency of some comparisons can be as low as 80%. Indeed, an improvement can be made by simplifying the algorithm: The first comparison probability is bracketed by

$$1-\left(\frac{n}{n+m}\right)^{n/(2m)} < q \leqq 1-\left(\frac{n}{n+m}\right)^{n/m}.$$

The lower sequence as $n = m, 2m, 3m, \cdots$ is

$$1-\sqrt{1/2}, 1-\sqrt{4/9}, \cdots \to 1-1/\sqrt{e}$$

and the upper is

$$1-1/2, 1-4/9, \cdots \to 1-1/e.$$

Except at extremely large ratios, the second sequence stays closer to 1/2. The first comparison will be more efficient if the index in the B list is closer to $n+1-(n/m)$ than $n+1-(n/2m)$.

Also it should be noted that a unique property of an exponential is that it is meaningful to talk about a "half-life". In the present context, the critical "half-life" is that of the probability distribution, and as we have seen, its value is approximately $\Delta \approx n/m$. But the implication is that if the first comparison is efficient and $A_m < B_{n+1-\Delta}$, the next test may as well be at $B_{n+1-2\Delta}$. This suggests the following modification to the Hwang–Lin algorithm:

1. Assuming $m \leqq n$, let $\Delta = \text{ROUND}\,(n/m)$ and $x = n+1-\Delta$.
2. Compare A_m with B_x. If $A_m > B_x$, insert A_m in $B_{x+1}, \cdots, B_n$ using a binary procedure that favors the larger index values. (If $n-x$ is odd and the middle element is unique, use it; if even, use the middle with highest index in B as the comparison point.) Emit $A_m, B_{y+1}, \cdots, B_n$, where $B_y < A_m < B_{y+1}$, and apply the algorithm to $A_1, \cdots, A_{m-1}$ and $B_1, \cdots, B_y$.
3. If $A_m < B_x$, emit $B_x, \cdots, B_n$, set $n = x-1$ and then set $x = x-\Delta$. Return to 2.

The linear testing pattern of fractile insertion is very naturally appropriate here. Quite precisely, this is fractile insertion with a Δ adjusted to the tail distribution. Unless there is some difficulty in implementing the more general binary insertion, this modified version should run faster on the average than the original Hwang–Lin algorithm. Not only does it eliminate the taking of a base 2 logarithm and the exponentiation, it should actually make fewer comparisons. In addition, Δ will be calculated less than half as often.

Unfortunately, because the comparisons prescribed by the Hwang–Lin algorithm are not uniformly efficient, it appears difficult to prove the asymptotic

quality of its performance on the average. A proof by scaling, similar to the one given for Region II, would seem to be the path of least resistance. On the other hand, the modified version is better balanced, and it appears that a 6% bound could be established based simply on the efficiency of the worst comparison.

Conclusions. We have exhibited a merging algorithm based on fractile insertion which requires on the average less than 6% more comparisons than the information theoretic lower bound, $\log_2 \binom{n+m}{m}$. Moreover, $\log_2 \binom{n+m}{m}$ is not even the tightest lower bound (See [4, p. 194]). Given the tightness of the upper bound the algorithm establishes, any algorithm which is noticeably more efficient than fractile insertion (or the modified fractile insertion-Hwang–Lin algorithm above) will most probably be impractically complex. Indeed, at one point in our study, we calculated the performance of a very sophisticated median insertion algorithm; it actually computed the sequence of testing points that would best balance the tree. The reduction in the number of comparisons was unimpressive, less than 1% for a typical problem.

As we have mentioned, our result implies that it is possible to sort with an average number of comparisons close to the optimum. While this was known previously, there is now the additional freedom of knowing that the order of merging does not substantially affect the required number of comparisons.

Finally, the fractile insertion algorithm provides a highly efficient technique for decomposing a large merging problem into many separately processable smaller problems. As such it may make possible efficient parallelism in merging.

REFERENCES

[1] R. B. ASH, *Information Theory*, Interscience, New York, 1965.

[2] W. FELLER, *An Introduction to Probability Theory and Its Applications*, vol. I, John Wiley, New York, 1963.

[3] F. K. HWANG AND S. LIN, *A simple algorithm for merging two disjoint linearly ordered sets*, this Journal, 1 (1972), pp. 31–39.

[4] D. E. KNUTH, *The Art of Computer Programming, Vol. 3: Sorting and Searching*, Addison-Wesley, Reading, MA, 1973.