SIAM J. COMPUT.
Vol. 22, No. 2, pp. 261-271, April 1993

TWO PROBABILISTIC RESULTS ON MERGING*

WENCESLAS FERNANDEZ DE LA VEGA†, SAMPATH KANNAN‡, AND MIKLOS SANTHA†

Abstract. This paper contains two probabilistic results about merging two sorted lists of sizes n and m with $m < n$. This paper designs a probabilistic algorithm, which in the worst case is significantly faster than any deterministic one in the range $1.618 < n/m \leq 3$. This paper extends it into a simple general algorithm that performs well for any ratio n/m. In particular, for $n/m > 1.618$ it is significantly faster than binary merge. This paper also proves an average case lower bound for a widely studied class of merging algorithms, when $1 < n/m < \sqrt{2}+1$.

Key words. merging, randomized algorithm, information theory, lower bound

AMS(MOS) subject classifications. 68Q20, 68Q25

1. Introduction. Merging is one of the basic problems in theoretical computer science. Given two sorted lists $A = \{a_1 < \ldots < a_m\}$ and $B = \{b_1 < \ldots < b_n\}$, the task consists of sorting the union of the two lists. We assume that the $m + n$ elements are distinct and $m \leq n$. The merging is performed by pairwise comparisons between items in A and items in B. The measure of complexity of a merging algorithm is the number of comparisons made by the algorithm, and the complexity of the merging problem is the complexity of the best merging algorithm. As usual, we can speak of worst case and average case complexity.

The worst case complexity of the merging problem is quite well studied. Let $C(m, n)$ denote this complexity with input lists of sizes m and n. When $m = 1$, merging degenerates into binary search whose complexity is $\lceil \log_2(n+1) \rceil$. The case $m = 2$ was completely solved by Graham [7] and Hwang and Lin [5]; their result is

$$C(2, n) = \lceil \log_2 7(n+1)/12 \rceil + \lceil \log_2 14(n+1)/17 \rceil.$$

The exact complexity is also known when the sizes of the two lists are not too far apart. For $m \leq n \leq \lfloor 3m/2 \rfloor$ we have

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

and the optimal algorithm is the well-known tape-merge (two-way merge). This result was obtained by Stockmeyer and Yao [10] and by Christen [2], after it was observed by several people for $m \leq n \leq m + 4$. The best known algorithm—binary merge—which gives satisfactory results for all values of m and n is due to Hwang and Lin [6]. Let N denote the set of nonnegative integers. Set $n/m = 2^t x$, where $t \in N$ and $1 < x \leq 2$ are uniquely determined, and let $BM(m, n)$ denote the complexity of binary merge. Then

$$BM(m, n) = \lfloor (t + x + 1)m \rfloor - 1,$$

and Hwang and Lin have also shown that

$$BM(m, n) \leq \lceil L(m, n) \rceil + m,$$

*Received by the editors June 4, 1990; accepted for publication (in revised form) December 15, 1991.

†Centre National de la Recherche Scientifique, Unité de Recherche Associée 410, Université Paris-Sud, 91405 Orsay, France.

‡Computer Science Department, University of Arizona, Tucson, Arizona 85721. The work of this author was supported by National Science Foundation grant CCR91-08969.

where $L(m,n) = \log_2 \binom{m+n}{m}$ is the lower bound from information theory. This means that if the ratio n/m goes to infinity, then the relative extra work done by binary merge is $o(C(m,n))$. As the relative extra work might be significant for ratios n/m of constant order, it is important to look for improvements in this case.

Several algorithms were proposed, which in some range for n/m perform better than binary merge. Let us say that merging algorithm A_1 with running time $T_1(m,n)$ is *significantly* faster for some fixed ratio n/m than merging algorithm A_2 with running time $T_2(m,n)$, if $T_2(m,n) - T_1(m,n) = \Omega(m)$. The merging algorithm of Hwang and Deutsch [4] is better than binary merge for small values of m, but not significantly faster. The first significant improvement over binary merge was proposed by Manacher [8]; he improved it for $n \geq 8m$ by $31m/336$ comparisons. It was further improved by Christen [1], who designed an ingenious but quite involved merging algorithm. It is better than binary merge if $n > 3m$, it uses at least $m/4$ fewer comparisons if $n \geq 4m$, and asymptotically it uses at least $m/3 - o(m)$ fewer comparisons when n/m goes to infinity. On the other hand, this algorithm is worse than binary merge when $n < 3m$.

In the first part of this paper we propose a simple probabilistic algorithm for merging. The algorithm will at some points flip a (biased) coin, and its next step will depend on the result of the coin toss. The algorithm is quite simple, and works well for all values of m and n. It is significantly faster than binary merge for $n/m > (\sqrt{5}+1)/2 \approx 1.618$. To the best of our knowledge, binary merge is the best algorithm known for $n/m \leq 3$, thus our probabilistic algorithm is significantly faster than any deterministic one in this range.

In the second part of the paper we prove a nontrivial average case lower bound for a widely studied class of merging algorithms called *insertive* algorithms, defined as follows: During the run of a merging algorithm, we will say that an element a of a list is *active* if a has been compared to at least one element of the other list, but a's exact position in the other list has not been determined. We will call a merging algorithm *insertive* if there is no more than one active element in the smaller list at any time. Apart from this constraint the algorithm can be arbitrary. Many merging algorithms such as repeated binary search, tape merge, and binary merge are insertive. We can establish the lower bound for ratios n/m in the interval $(1, \sqrt{2}+1)$ which are bounded away by a constant from the endpoints of the interval. More precisely, we will prove that if $(1+\eta) \leq n/m \leq (\sqrt{2}+1-\eta)$ for some constant $\eta > 0$, then every insertive merging algorithm makes on the average at least a constant factor more comparisons than the information theoretic lower bound. We were unable to extend the lower bound for constant ratios $n/m \geq \sqrt{2}+1$. When $n/m \to \infty$, then the result of Hwang and Lin also implies that the information theoretic lower bound and the actual complexity are asymptotically equal.

The rest of the paper is organized as follows: In §2.1 we describe our algorithm. The analysis of its complexity will be done in §2.2 and we also prove there that the particular bias of the coin in the algorithm was optimally chosen. In §3 we prove the average case lower bound result for insertive algorithms. Finally, in §4 we mention some open problems. Throughout the paper the logarithm function in base 2 will be denoted by log, and in the natural base by ln.

2. The probabilistic algorithm.

2.1. Description of the algorithm. For this section let $s = (\sqrt{5}-1)/2 \approx 0.618$, and $r = (\sqrt{2} - 1 + \sqrt{2}s)^2 \approx 1.659$. These numbers play a considerable role in the algorithm and in its analysis. The heart of the algorithm MERGE is the probabilistic procedure PROBMERGE, which merges two already sorted lists, where the longer list contains more than $(1+s)$-times, but at most $2r$-times as many elements as the shorter one. The intuition underlying PROBMERGE is described below. We know that if $n \le \lfloor 3m/2 \rfloor$, then the tape merge algorithm is best possible. Given two sorted lists A and B as before, the tape merge algorithm can be thought of as inserting the a's in order in list B. At each stage the next a to be inserted is compared with the first "eligible" element in B. If, however, B is a list of length $2m$, then it might be better to compare the next a to be inserted with the second eligible b (since, on average, there are two b's between every two consecutive a's). However, the deterministic algorithm that tries to do this turns out to be no better than tape merge. Its worst case complexity is also $3m-1$. A natural idea is to try to compare the next a to be inserted with either the first or the second eligible element in B, the decision being made probabilistically. In fact this is our algorithm.

We will use sentinel elements in the algorithm. They make the description simpler for the price of a little loss of efficiency.

```
Procedure PROBMERGE
Input: $A = \{a_1 < \ldots < a_m\}$ and $B = \{b_1 < \ldots < b_n\}$, where $1+s < n/m \le 2r$, and
       sentinel elements $b_0 = -\infty$, $b_{n+1} = b_{n+2} = \infty$.
Output: The merged list.
  • $p := \begin{cases} s & \text{if } 1+s < n/m \le 2+s \\ \sqrt{n/m}-1 & \text{if } 2+s < n/m \le 2r \end{cases}$   $\left\{ \begin{array}{c} \textit{the choice of the} \\ \textit{probability value} \end{array} \right\}$
  • $i := 1; j := 1$ {$i$ indexes $A$, $j$ indexes $B$}
  • while $i \le m$ do
      – with probability $1-p$ compare $a_i$ with $b_j$
          * if $a_i < b_j$ then $i := i+1$ $\{b_{j-1} < a_i < b_j\}$
          * else $j := j+1$
      – with probability $p$ compare $a_i$ with $b_{j+1}$
          * if $a_i < b_{j+1}$ then
              · compare $a_i$ with $b_j$
              · if $a_i < b_j$ then $i := i+1$ $\{b_{j-1} < a_i < b_j\}$
              · else $i := i+1; j := j+1$ $\{b_j < a_i < b_{j+1}\}$
          * else $j := j+2$
  • end
```

The general algorithm MERGE uses PROBMERGE as a subroutine. Given lists A and B of size m and $n > (1+s)m$, respectively, the algorithm calls PROBMERGE directly if $n \le rm$. Otherwise it picks a uniformly spaced sublist C of B. The spacing between the elements of C is 2^t, where t is the unique integer such that cardinality of C is between rm and $2rm$. The algorithm first merges A with C probabilistically. This determines for each a in A a list of $2^t - 1$ b's into which a should be inserted. The insertion is done deterministically by binary search and takes t steps. Let us observe that MERGE actually coincides with the procedure PROBMERGE while $n \le 2rm$.

```
Algorithm MERGE
Input: $A = \{a_1 < \ldots < a_m\}$ and $B = \{b_1 < \ldots < b_n\}$, where $(1+s) < n/m$. For
       $r < n/m$ set $m = 2^t xm$, $t \in N$, $r < x \le 2r$.
Output: The merged list.
  1. If $n/m \le r$ then PROBMERGE($A, B$).
  2. Let $C = \{c_1 < \ldots < c_{\lceil xm \rceil}\}$ be the sublist of $B$, where $c_k = b_{(k-1)2^t+1}$ for
     $k = 1, \ldots, \lceil xm \rceil$, and define sentinel elements $c_0 = -\infty$, $c_{\lceil xm \rceil + 1} = \infty$.
  3. PROBMERGE($A, C$).
```

4. For $i = 1, \ldots, m$ let $0 \leq j_i \leq \lceil xm \rceil$ be such that $c_{j_i} < a_i < c_{j_i+1}$.
5. For $i = 1, \ldots, m$ insert a_i by binary search into the list $\{b_{(j_i-1)2^t+2} < \cdots < b_{j_i 2^t}\}$.

2.2. Analysis of the algorithm. We would like to analyze the expected number of comparisons made by the algorithm PROBMERGE. Let $T(m,n)$ be this number when the input of the procedure is two lists of size m and n, respectively, $1 + s < n/m \leq 2r$, and let p be the probability value defined inside the procedure.

THEOREM 2.1. *We have*

$$T(m,n) \leq \begin{cases} sn + (1+s)m & \text{if } 1+s < n/m \leq 2+s, \\ 2\sqrt{nm} & \text{if } 2+s < n/m \leq 2r. \end{cases}$$

Proof. The outcome of the procedure uniquely determines m nonnegative integers $k_1, \ldots, k_m$, where k_i is the number of b's between a_{i-1} and a_i. As the list B contains n elements, $\sum_{i=1}^m k_i \leq n$. The procedure puts the a's into B one by one. Let f_k be the expected number of comparisons to put an a into B, when the number of b's between a and its predecessor in A is k. f_k does not depend on the index of the element being inserted. Then

$$T(m,n) = \max_{\substack{k_1, \ldots, k_m \\ \sum_{i=1}^m k_i \leq n}} \sum_{i=1}^m f_{k_i}.$$

Thus we are interested in the values f_k. If $k \geq 2$, then after the first comparison the element a jumps over two b's with probability p, and jumps over one b with probability $1-p$. This means that we get the following recurrence relation for f_k:

$$f_k = (1-p)f_{k-1} + pf_{k-2} + 1,$$

with the initial conditions

$$f_0 = 2p + 1(1-p) = p + 1,$$

$$f_1 = 3p(1-p) + 2((1-p)^2 + p) = -p^2 + p + 2.$$

By standard technique, the solution of this linear recurrence is

$$f_k = \frac{k}{p+1} + \frac{p^3 + p^2 - p}{(p+1)^2}(-p)^k + \frac{2p^2 + 4p + 1}{(p+1)^2}.$$

This tells us that

$$\sum_{i=1}^m f_{k_i} = \sum_{i=1}^m \frac{k_i}{p+1} + \frac{p^3 + p^2 - p}{(p+1)^2} \sum_{i=1}^m (-p)^{k_i} + \frac{2p^2 + 4p + 1}{(p+1)^2} m.$$

The term $\sum_{i=1}^m \frac{k_i}{p+1}$ is always bounded from above by $\frac{n}{p+1}$. If $p = s$, we have $p^3 + p^2 - p = 0$, and the result follows. If $p > s$ (when $2 + s < n/m$), we have $p^3 + p^2 - p > 0$, and $\sum_{i=1}^m (-p)^{k_i}$ is maximized by choosing $k_i = 0$. Simple arithmetic gives the result also in this case. □

The global algorithm MERGE calls PROBMERGE with two lists of size m and $\lceil mx \rceil$ and then makes n binary searches, each in a list of size $2^t - 1$. Applying Theorem 2.1, we immediately get the following result.

THEOREM 2.2. *Let $E(m,n)$ denote the expected number of comparisons made by* MERGE. *For $n/m \leq r$ set $n/m = x$, $t = 0$, and for $n/m > r$ set $n/m = 2^t x$, $t \in N$, $r < x \leq 2r$. Then*

$$E(m,n) \leq \begin{cases} (t + sx + 1 + s)m + 1 & \text{if } (1+s) < x \leq 2+s, \\ (t + 2\sqrt{x})m + 1 & \text{if } 2+s < x \leq 2r. \end{cases}$$

COROLLARY 2.1. *The algorithm* MERGE *is significantly faster than binary merge for any fixed ratio $n/m > 1+s$.*

Proof. With the notation of Theorem 2.2 we have

$$BM(m,n) - E(m,n) \geq \begin{cases} (x - sx - s)m - 3 & \text{if } (1+s) < x \leq 2, \\ ((1-s) - x(s - 1/2))m - 3 & \text{if } 2 < x \leq 2+s, \\ (2 - \sqrt{x})^2 m/2 - 3 & \text{if } 2+s < x \leq 2r. \end{cases}$$

This difference is $\Omega(m)$. □

Let us point out an interesting special case of the above result. When $n = 2m$, PROBMERGE merges the two lists with less than $2.855m$ expected comparisons, whereas the best known deterministic algorithm [6] performs $3m - 2$ comparisons.

Some more calculation also yields a relation between the expected running time of MERGE and the information theoretic lower bound.

COROLLARY 2.2. $E(m,n) - L(m,n) \leq 0.471m$.

We also claim that the probability value p was optimally chosen in the procedure PROBMERGE. Let $T_p(m,n)$ be the expected number of comparisons made by PROBMERGE when the probability $0 \leq p \leq 1$ is taken as a variable. Let us observe that we have just analyzed $T_s(m,n)$ and $T_{\sqrt{n/m-1}}(m,n)$ in Theorem 2.1. For the purpose of this analysis let us permit any input lists such that the ratio n/m is at least 1. The following theorem gives optimal choices for the probability value for all ranges of n/m.

THEOREM 2.3. *For every p, for every large enough m and n, we have*

$$T_p(m,n) \geq \begin{cases} T_0(m,n) & \text{if } 1 \leq n/m \leq 1+s, \\ T_s(m,n) & \text{if } 1+s \leq n/m \leq 2+s, \\ T_{\sqrt{n/m-1}}(m,n) & \text{if } 2+s \leq n/m \leq 4, \\ T_1(m,n) & \text{if } 4 \leq n/m. \end{cases}$$

Proof. We will examine the function $T_p(m,n)$ according to two cases.

Case 1. $p \geq s$. For $k_1 = \ldots = k_{m-1} = 0$ and $k_m = n$, the expected number of comparisons made by the procedure PROBMERGE is

$$U_p(m,n) = \left(\frac{n/m}{p+1} + p + 1\right)m + \frac{p^3 + p^2 - p}{(p+1)^2}((-p)^n - 1).$$

By definition, for every p, m, n we have $U_p(m,n) \leq T_p(m,n)$. We claim that for every large enough m and n, for every $p \neq s$, we have

$$T_s(m,n) < U_p(m,n) \quad \text{if} \quad 1+s < n/m \leq 2+s,$$

and that for every large enough m and n, for every $p \neq \sqrt{n/m}-1$, we have

$$T_{\sqrt{n/m}-1}(m,n) < U_p(m,n) \quad \text{if} \quad 2+s \leq n/m.$$

Let us define the functions $V(p) = (\frac{n/m}{p+1}+p+1)m$ and $D(p) = \frac{p^3+p^2-p}{(p+1)^2}((-p)^n-1)$. Then $U_p(m,n) = V(p)+D(p)$. We will first prove the claim for the function $V(p)$ rather than the function $U_p(m,n)$. Its derivative is $V'(p) = (\frac{-n/m}{(p+1)^2}+1)m$, and it is easy to check that the function $V(p)$ takes its global minimum in the set of real numbers at the point $\sqrt{n/m}-1$. Moreover, $V(p)$ is an increasing function for $p \geq \sqrt{n/m}-1$. Thus the minimum of the function $V(p)$ in the interval $[s,1]$ is at the point s, if $\sqrt{n/m}-1 \leq s$, and is at the point $\sqrt{n/m}-1$, if $\sqrt{n/m}-1 \geq s$. If we observe that $\sqrt{n/m}-1 \leq s$ if and only if $n/m \leq 2+s$, then the claim follows for the function V.

If $\sqrt{n/m}-1 \geq s$ then for every p, we have $V(p) - V(\sqrt{n/m}-1) = \Omega(m)$, while $D(\sqrt{n/m}-1) - D(p) \leq |D(\sqrt{n/m}-1)| + |D(p)| = O(1)$. This implies the claim for $U_p(m,n)$ when $\sqrt{n/m}-1 \geq s$, and a similar argument works when $\sqrt{n/m}-1 \leq s$.

In conclusion, we have shown that for every p in the interval $[s,1]$, for every large enough m and n,

$$T_p(m,n) \geq T_s(m,n) \quad \text{if} \quad n/m \leq 2+s,$$

and similarly, for every p, for every large enough m and n,

$$T_p(m,n) \geq T_{\sqrt{n/m}-1}(m,n) \quad \text{if} \quad n/m \geq 2+s.$$

Case 2. $p \leq s$. We will show that for every p in the interval $[0,s]$, for every large enough m and n,

$$T_p(m,n) \geq T_0(m,n) \quad \text{if} \quad n/m \leq 1+s,$$

and similarly, for every p, for every large enough m and n,

$$T_p(m,n) \geq T_s(m,n) \quad \text{if} \quad n/m \geq 1+s.$$

This clearly implies the theorem. The function $U_p(m,n)$ that we use is when $k_1 = \dots k_{m-1} = 1$ and $k_m = n-m+1$. This is because, in this range for p, p^3+p^2-p is negative. For these choices of k_i we get,

$$U_p(m,n) = \left(\frac{-p^3+3p+1+n/m}{p+1}\right)m + \left(\frac{p^3+p^2-p}{(p+1)^2}\right)((-p)^{n-m+1}+p).$$

Once again we break up $U_p(m,n)$ into its two terms which we call $V(p)$ and $D(p)$. Thus $V(p) = (\frac{-p^3+3p+1+n/m}{p+1})m$ and $D(p) = (\frac{p^3+p^2-p}{(p+1)^2})((-p)^{n-m+1}+p)$. As before, we concentrate on $V(p)$, $D(p)$ being $O(1)$. We compute the derivative $V'(p)$ of $V(p)$ with respect to p:

$$V'(p) = \left(\frac{-2p^3-3p^2+2-n/m}{(p+1)^2}\right)m.$$

Note that $V'(p)$ is a decreasing function in the range $[0,\infty]$. This means that the minimum value for $V(p)$ for p in the range $[0,s]$ is attained either at 0 or at s. $V(0) =$

$m(n/m+1)$ and $V(s) = m((1+s)+sn/m)$. It is easy to see that $V(s) - V(0)$ decreases as n/m increases. At $n/m = (1+s)$, $V(s) = V(0)$. This proves both statements of the claim at the beginning of case 2. □

Thus while $n/m \leq (1+s)$, our probabilistic procedure gives deterministic tape-merge as a special case. In the range $1+s \leq n/m \leq 2+s$ the best choice is the constant $p = s$. For $n/m > 2+s$, the best choice grows with n/m until 1, when the procedure (obviously) degenerates into a deterministic one. There is no reason to use PROBMERGE for large values of n/m; MERGE is significantly faster when $n/m > 2r$. In fact that is how r is chosen.

3. The lower bound.

In this section we will prove the following theorem.

THEOREM 3.1. *For every $\eta > 0$ there exists $\epsilon > 0$ such that for every large enough m and n with $(1+\eta)m \leq n \leq (\sqrt{2}+1-\eta)m$, every insertive merging algorithm makes at least $(1+\epsilon)L(m,n)$ comparisons on the average.*

Proof. The proof depends on the following Unbalance Lemma.

UNBALANCE LEMMA. *Let T denote a binary decision tree with k leaves, let the set of internal nodes be $\{D_1, \ldots, D_{k-1}\}$, and let d_j be the number of leaves of the subtree with root D_j. Let us suppose that for some $J \subseteq \{1, 2, \ldots, k-1\}$, the subset of nodes $\{D_j : j \in J\}$ satisfies the following conditions:*

(i) *There exists $\epsilon_1 > 0$ such that $\sum_{j \in J} d_j \geq \epsilon_1 k \log k$.*

(ii) *There exists $\epsilon_2 > 0$ such that for every $j \in J$, the answer probabilities (fraction of leaves in left and right subtrees) at the node D_j lie outside the interval $(1/2 - \epsilon_2, 1/2 + \epsilon_2)$. Then, the average path length of T is at least*

$$(1 + \epsilon_1 \epsilon_2^2) \log k.$$

Proof. The average path length of T is

$$L(T) = \frac{1}{k} \sum_{1 \leq j \leq k-1} d_j.$$

Let p_j denote the answer probability at D_j. The information in T is

$$I(T) = \sum_{1 \leq j \leq k-1} d_j H_j,$$

where $H_j = p_j \log 1/p_j + q_j \log 1/q_j$ is the entropy function. Elementary information theory [9] tells us that we have necessarily

$$I(T) = k \log k.$$

For every j, we obviously have $H_j \leq 1$, and Taylor's formula gives $H_j \leq 1 - \epsilon_2^2$ for $p_j \leq 1/2 - \epsilon_2$. Thus we obtain

$$\sum_{j \notin J} d_j + \sum_{j \in J} d_j (1 - {\epsilon_2}^2) \geq k \log k,$$

and finally

$$L(T) \geq (1 + \epsilon_1 {\epsilon_2}^2) \log k. \qquad □$$

Let us fix some insertive algorithm, and let T be the binary decision tree associated with this algorithm. Let the internal nodes of T be $D_1, \ldots, D_{k-1}$, where $k = \binom{m+n}{m}$.

Since the average running time of the algorithm is the average path length of T, it will suffice to prove that the assumptions of the Unbalance Lemma are fulfilled for some subset of the nodes and for some proper ϵ_1 and ϵ_2. To begin, notice that there is a bijection between the outcomes of a merging algorithm and the set Z of words containing precisely m letters a and n letters b. For any word $w \in \{a,b\}^*$, let $|w|_a$ and $|w|_b$ denote, respectively, the number of occurrences of a and b in w.

It follows from the hypothesis that for some sufficiently small $\zeta > 0$ we have

$$1/2 + 3\zeta < n/(m+n) < 1/\sqrt{2} - 3\zeta.$$

Let $c_3 > 0$ be an appropriate constant to be specified later. For any $m \le i \le n$ and $w \in Z$, let *decomp*(w,i) denote the decomposition $uvxy = w$, where the factors u, v, x, y have the following lengths:

- $|u| = i - 3c_3$,
- $|v| = |x| = 3c_3$,
- $|y| = m + n - i - 3c_3$.

Let us denote by Z_i the set of words $w \in Z$ such that *decomp*(w,i) satisfies the following conditions:

1. $vx = (bba)^{c_3}(abb)^{c_3}$,
2. for every decomposition $y = y'y''$, where $|y'| = l$; and for every decomposition $u = u''u'$, where $|u'| = l$, we have

$$(1/2+\zeta)l - c_2 \le |y'|_b, |u'|_b \le (1/\sqrt{2} - \zeta)l + c_2,$$

where $c_2 > 0$ is an appropriate constant also to be specified later. We will define a subset of nodes satisfying the assumptions of the Unbalance Lemma using the sets Z_i. First we prove two claims about these sets.

CLAIM 1. *There exists a constant $\epsilon_3 > 0$ such that for all $m \le i \le n$ we have*

$$|Z_i| \ge \epsilon_3 |Z|\,.$$

Proof. Let w be a random element of Z, and let $m \le i \le n$. The first requirement is obviously satisfied by w with constant probability. We will show that under this hypothesis, the conditional probability that $|y'|_b$ falls outside the interval $((1/2+\zeta)l - c_2, (1/\sqrt{2} - \zeta)l + c_2)$ is less then 0.48 . As the second requirement is symmetrical in $|y'|_b$ and $|u'|_b$, this will imply the claim. We shall (rather crudely) bound this probability by the sum $\sum_{1 \le l \le |y|} s(l)$, where $s(l)$ denotes the probability that the left factor of y of length l violates the second requirement. Clearly $s(l) = q(l) + r(l)$, where $q(l)$ is the probability that the left factor of y of length l contains less than $(1/2+\zeta)l - c_2$ occurrences of b's, and $r(l)$ is the probability that the same left factor contains more than $(1/\sqrt{2} - \zeta)l + c_2$ occurrences of b's. Let us recall Hoeffding's bound [3] about sampling without replacement.

LEMMA (Hoeffding's bound). [3] *Let us suppose that we have an urn that contains $M + N$ letters, N of which are b's and M of which are a's. Let $p = N/(M+N)$. We are sampling the urn without replacement. Let X_j be the $0-1$ valued random variable which is 1 if and only if at the jth trial we get a letter b. Let $S_{i,p} = \sum_{j=1}^{i} X_j$. Then for every $\delta > 0$ we have*

$$P[S_{i,p} \ge (1+\delta)ip] \le \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{ip},$$

and similarly, for every $0 < \gamma \leq 1$ *we have*

$$P[S_{i,p} \leq (1-\gamma)ip] \leq \left(\frac{e^{\gamma}}{(1+\gamma)^{1+\gamma}}\right)^{ip}.$$

We will use Hoeffding's bound in the following way: set $N = n - 4c_3$ and $M = m - 2c_3$ (recall that $4c_3$ is the number of b's and $2c_3$ is the number of a's in vx). Let $p = N/(M+N)$. Then for large enough m, we have:

$$1/2 + 2\zeta < N/(M+N) < 1/\sqrt{2} - 2\zeta.$$

Let $q(l)$ (respectively, $r(l)$) be the probability that in a random w which satisfies the first condition, the left factor of y of length l contains more (fewer) occurrences of b than the second condition permits. Then

$$q(l) < P[S_{l,1/\sqrt{2}-2\zeta} \geq (1/\sqrt{2} - \zeta)l],$$

and

$$r(l) < P[S_{l,1/2+2\zeta} \leq (1/2+\zeta)l].$$

Hoeffding's bound implies that there exists a constant $0 < \xi < 1$ which depends only on ζ such that

$$q(l), r(l) < \xi^l.$$

Let $c_1 = c_1(\xi)$ be an integer such that

$$\sum_{l=c_1}^{\infty} \xi^l < 0.24.$$

Now, as a function of c_1 we choose c_2 large enough that $q(l) = r(l) = 0$, whenever $l \leq c_1$. Hence

$$\sum_{1 \leq l \leq |y|} s(l) < 2\sum_{l=c_1}^{\infty} \xi^l < 0.48\,. \qquad \square$$

We are now ready to choose the constant c_3 as a function of c_2 such that the following Claim becomes true. Let $c_3 = c_2/((3/\sqrt{2}) - 2 - 3\zeta)$.

CLAIM 2. *Let* $w \in Z_i$ *for some* $m \leq i \leq n$, *and let* $uvxy = decomp(w, i)$. *Let* z *be a nonempty prefix of* xy (*a suffix of* uv) *such that either the letter which follows* (*precedes*) z *in* w *is an* a, *or there is no more* a *in* w *after* (*before*) z. *Then we have*

$$(1/2+\zeta) \leq |z|_b/|z| \leq (1/\sqrt{2} - \zeta).$$

Proof. We show the upper bound. If z is a prefix of x, then $|z|_b/|z| = 2/3$. Otherwise this fraction is maximized when the corresponding prefix of y contains the most possible b's. If this prefix is of length l, we have

$$|z|_b/|z| \leq \frac{2c_3 + (1/\sqrt{2} - \zeta)l + c_2}{3c_3 + l} = 1/\sqrt{2} - \zeta. \qquad \square$$

We can now define a subset of the nodes of T which satisfies the conditions of the Unbalance Lemma. For every couple (w, i) such that $m \leq i \leq n$ and $w \in Z_i$, we first define an internal node $V(i, w)$ of T. Let $uvxy = decomp(w, i)$, and let $h = h(w, i)$ be such that a_h is the first instance of a in x (thus $h-1$ is the number of instances of a in uv). If on the path of T corresponding to w the comparisons involving a_{h-1} precede those involving a_h, then let $V(w, i)$ be the node of the decision tree at which the first comparison involving a_h takes place. Otherwise let $V(w, i)$ be the node at which the first comparison involving a_{h-1} takes place. Finally let

$$J = \{j : D_j = V(w, i) \text{ for some } w \in Z_i\}.$$

We now show that the two conditions of the Unbalance Lemma are met by the nodes whose indices are in J. First we will show that

$$\sum_{j \in J} d_j \geq \sum_{m \leq i \leq n} |Z_i|.$$

On the right-hand side every word $w \in \bigcup_{m \leq i \leq n} Z_i$ is counted with multiplicity t_w, where

$$t_w = |\{i : w \in Z_i\}|.$$

If $i \neq i'$, then $V(w, i) \neq V(w, i')$, because $|h(w, i) - h(w, i')| > 1$. Thus the leaf corresponding to w is counted on the left-hand side at least t_w times. Therefore by using Claim 1 we get

$$\sum_{j \in J} d_j \geq \epsilon_3 m \binom{m+n}{m} = \epsilon_1 k \log k.$$

We now turn to the second condition of the Unbalance Lemma. Let $D_j = V(w, i)$ for some $w \in Z_i$, and let us suppose without loss of generality that the comparisons involving a_{h-1} precede those involving a_h. Let a_{h+r} denote the leftmost right neighbour of a_h which has already been treated when the comparisons involving a_h begin at D_j. (If there is no such element we set $r = m-h+1$.) Let us further suppose that a_{h-1} and a_{h+r} are separated by s occurrences of b for some $s \geq 1$, which we denote by $b_e, \ldots, b_{e+s-1}$. Let us observe that $s/(r+s) = |z|_b/|z|$ for some initial segment z of xy, where z satisfies the conditions of Claim 2. Therefore we have

$$1/2 + \zeta \leq s/(r+s) \leq 1/\sqrt{2} - \zeta.$$

The comparison at D_j is $a_h : b_{e+l-1}$ for some $1 \leq l \leq s$. Let $p(r, s, l) = Pr[a_h > b_{e+l-1}]$ denote the answer probability at D_j. Among the leaves of $V(w, i)$, the relative rankings of the sets $\{a_h, \ldots, a_{h+r-1}\}$ and $\{b_e, \ldots, b_{e+s-1}\}$ are all equally represented. For any word belonging to the leaves of D_j, the relative rank of a_h within the pooled set is at least l if and only if $a_h > b_{e+l-1}$. Thus we have

$$p(r, s, l) = \frac{s(s-1)\ldots(s-l+1)}{(r+s)(r+s-1)\ldots(r+s-l+1)}$$

We claim that for every l, the answer probability $p(r, s, l)$ falls outside the interval $(1/2 - \epsilon_2, 1/2 + \epsilon_2)$, for $\epsilon_2 = \min\{\zeta, 1/2 - (1/\sqrt{2} - \zeta)^2\}$. We prove this in two cases according to the value of l. If $l = 1$, then $p(r, s, 1) = s/(r+s) \geq 1/2 + \zeta$. If $l \geq 2$, then we have

$$p(r, s, l) \leq \left(\frac{s}{r+s}\right)^2 \leq (1/\sqrt{2} - \zeta)^2,$$

which means that the second condition of the Unbalance Lemma is also satisfied. We now set $\epsilon = \epsilon_1\epsilon_2^2$. The conclusion of the Unbalance Lemma gives exactly what we wanted to prove. □

4. Open problems. We have shown how to make significant improvements upon the tape merge algorithm when the ratio n/m is at least the golden ratio. An interesting open question is the value of the smallest ratio n/m where a significant improvement — probabilistic or deterministic — can be obtained. Although the lower bounds in [2], [10] hold only for deterministic algorithms, we conjecture that not even a probabilistic algorithm can achieve improvements over tape merge for ratios less than the golden ratio.

It would be interesting to design other, more complex probabilistic algorithms. Generalizing our average case lower bound for arbitrary merging algorithms also remains open.

Acknowledgments. Thanks are due to Charles Delorme for his most valuable assistance. We would also like to thank Russell Impagliazzo for his help in simplifying the analysis of the probabilistic algorithm.

REFERENCES

[1] C. Christen, *Improving the bound on optimal merging*, in Proceedings of the 19th IEEE Symposium on Foundations of Computer Science, 1978, pp. 259–266.

[2] ———, *On the optimality of the straight merging algorithm*, Tech. Report Publication 296, Dép. d'Info. et de Rech. Op., Université de Montréal, Montréal, Québec, Canada, 1978.

[3] V. Chvatal, *Probabilistic methods in graph theory*, Ann. Oper. Res., 1 (1984), pp. 171–182.

[4] F. K. Hwang and D. N. Deutsch, *A class of merging algorithms*, J. Assoc. Comput. Mach., 20 (1973), pp. 148–159.

[5] F. K. Hwang and S. Lin, *Optimal merging of 2 elements with n elements*, Acta Inform., 1 (1971), pp. 145–158.

[6] ———, *A simple algorithm for merging two disjoint linearly ordered lists*, SIAM J. Comput., 1 (1972), pp. 31–39.

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

[8] G. K. Manacher, *Significant improvements to the Hwang–Ling merging algorithm*, J. Assoc. Comput., Mach., 26 (1979), pp. 434–440.

[9] C. F. Picard, *Graphes et questionnaires*, Gauthiers-Villars, Paris, 1973.

[10] P. K. Stockmeyer and F. F. Yao, *On the optimality of linear merge*, SIAM J. Comput., 9 (1980), pp. 85–90.