SIAM J. COMPUT.
Vol. 7, No. 3, August 1978

0097-5397/78/0703-0001 $01.00/0

DATA MOVEMENT IN ODD-EVEN MERGING*

ROBERT SEDGEWICK†

Abstract. A complete analysis is given of the number of exchanges used by the well-known Batcher's odd-even merging (and sorting) networks. Batcher's method involves a fixed sequence of "compare-exchange" operations, so the number of comparisons required is easy to compute, but the problem of determining how many comparisons result in exchanges has not been successfully attacked before. New results are derived in this paper giving accurate formulas for the worst-case and average values of this quantity.

The worst-case analysis leads to the unexpected result that, asymptotically, the ratio of exchanges to comparisons approaches 1, although convergence to this asymptotic maximum is very slow.

The average-case analysis shows that, asymptotically, only $\frac{1}{4}$ of the comparators are involved in exchanges. The method used to derive this result can in principle be used to get any asymptotic accuracy. The derivation involves principles of the theory of complex functions; in particular, properties of the Γ-function and the generalized Riemann ζ-function are integral to the solution. Intermediate results in the analysis may be applicable to the average-case analysis of other merging methods, and the final portion of the derivation illustrates the utility of the "gamma function" method of asymptotic analysis.

Key words. analysis of algorithms, odd-even merge, merging networks, merge-exchange sort, sorting networks, gamma function, zeta function

1. Introduction. Suppose that we have two sorted arrays $B[1], \cdots, B[N]$ and $C[1], \cdots, C[N]$ which we wish to merge into a single sorted array $A[1], \cdots, A[2N]$, The straightforward algorithm

$i := j := 1$; $B[N+1] := C[N+1] := \infty$;
loop for $k := 1, 2, \cdots, 2N$:
  if $B[i] < C[j]$ **then** $A[k] := B[i]$; $i := i+1$
    else $A[k] := C[j]$; $j := j+1$
repeat

has been shown to be the "best possible" way to solve this problem (see [13, p. 199]) in that it requires the minimum number of comparisons between keys, not counting the ∞ sentinel keys. However, this method may not be appropriate if, for example, we wish to build hardware to do the merging, since it requires space for the output array and its comparison sequence depends on the arrangement of the input.

The "odd-even" merge introduced by K. E. Batcher in 1964 [3], [4] is a well-known method for merging in place with a fixed comparison sequence. To satisfy the in place condition we assume that the first sorted input array is stored in the odd positions $A[1], A[3], \cdots, A[2N-1]$ of the output array, and the second sorted input array is stored in the even positions $A[2], A[4], \cdots, A[2N]$ of the output array. Such files are called 2-*ordered*, and merging is equivalent to sorting 2-ordered files. Then

* Received by the editors March 7, 1977, and in revised form October 3, 1977.

† Computer Science Program and Division of Applied Mathematics, Brown University, Providence, Rhode Island 02912. This work was supported by the National Science Foundation under Grant MC575-23738.

Batcher's method may be implemented as follows:

> **loop for** $j := 1, 2, \cdots, N$:
>   **if** $A[2j-1] > A[2j]$ **then** $A[2j-1] := A[2j]$;
> **repeat**;
> **loop for** $\delta := 2^{\lceil \lg N \rceil - 1}, 2^{\lceil \lg N \rceil - 2}, \cdots, 1$:
>   **loop for** $j := 1, 2, \cdots, N - \delta$:
>   **if** $A[2j] > A[2j+2\delta-1]$ **then** $A[2j] :=: A[2j+2\delta-1]$;
>   **repeat;**
> **repeat;**

In this program, notice that the only statements which actually operate on the data are the "compare-exchange" statements of the form

$$\textbf{if } A[2j] > A[2j+2\delta-1] \textbf{ then } A[2j] :=: A[2j+2\delta-1];$$

and these are performed in the same order regardless of the input. Because of this, it is convenient to describe the algorithm as a merging *network* as in Fig. 1, which shows the algorithm operating on a typical 2-ordered file of sixteen numbers. The numbers move from left to right, encountering "compare-exchange" modules on the way. Each module exchanges its inputs, if necessary, to make the larger number appear on the lower line after passing. (Modules which actually perform exchanges are boxed in Fig. 1.) The networks for $N = 1$, 2, 4, 8, and 16 are shown in Fig. 2. Notice that the networks are composed of stages (an initial stage plus one for each value of δ) within which all of the compare-exchanges can be overlapped. This makes Batcher's algorithm particularly useful when parallelism is available.

Figure 3 shows the networks for $N = 1$, 2, 4, 8 and 16 with the comparators arranged somewhat differently to illustrate why the method is called the "odd-even" merge. First the "odd" members of the input files ($A[1]$, $A[5]$, $A[9]$, $\cdots$ and $A[2]$, $A[6]$, $A[10]$, $\cdots$) are merged, and, independently, the "even" members of the input files ($A[3]$, $A[7]$, $A[11]$, $\cdots$ and $A[4]$, $A[8]$, $A[12]$, $\cdots$) are merged. After this, it turns out that a single stage of compare-exchange modules connecting $A[2]$ with $A[3]$, $A[4]$ with $A[5]$, $A[6]$ with $A[7]$, etc., will complete the sort. Batcher gave a complete inductive proof that his method is valid, using this approach [2] (see also Knuth [13, pp. 224–225]). Knuth gives another proof [13, exercise 5.2.2-10] which we shall examine in some detail below.

To determine the running time of a program, we need to be able to determine the frequency of execution of each of its instructions. In the program above, these

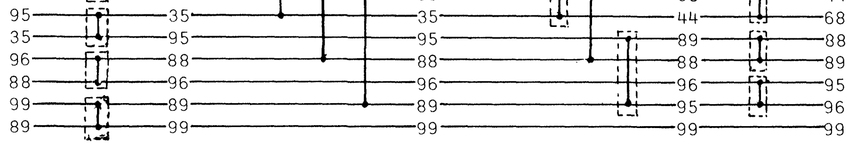


FIG. 1. *2-sorting a file of* 16 *elements.*

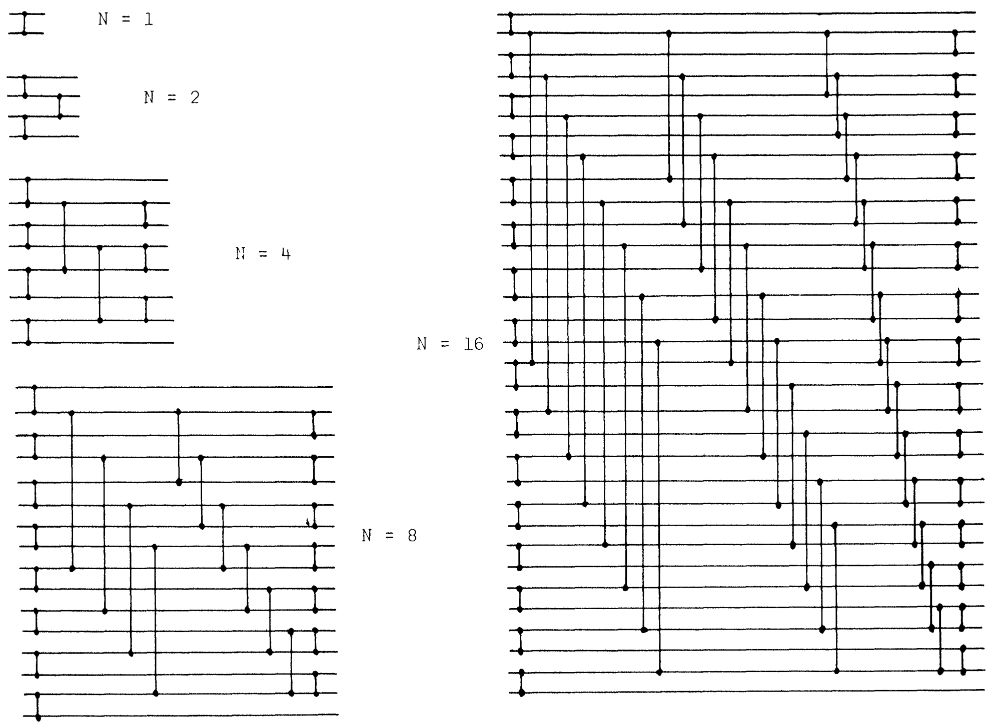


FIG. 2. *Odd-even merging (2-sorting) networks.*

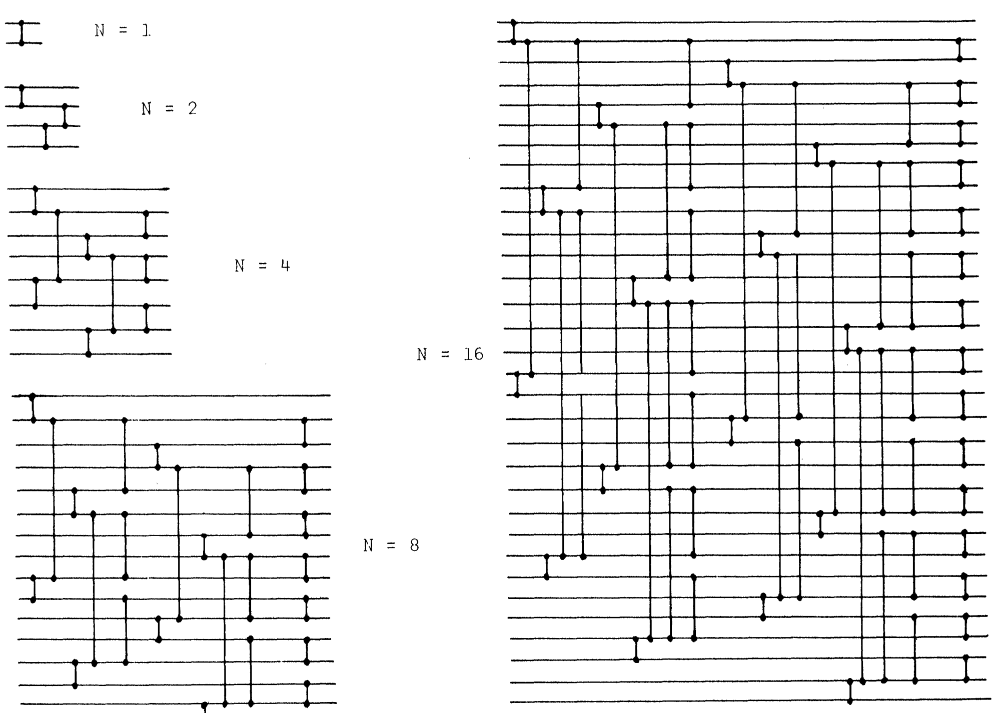


FIG. 3. *Odd-even merging networks (alternate arrangement).*

frequencies can be determined from N, the number of items in each file to be merged, and the following three quantities:

A—the number of stages,

B—the number of exchanges, and

C—the number of comparisons.

More precisely, A is the number of times a new value is assigned to δ, C is the number of times the tests "$A[2j-1]>A[2j]$" and "$A[2j]>A[2j+2\delta-1]$" are performed and B is the number of times these tests are successful.

The values of A and C are clearly independent of the input distribution. They do depend on the number of elements being merged, and we will write A_N and C_N to denote their values for an $N\times N$ merge. On inspection of the program, we see that

$$A_N = \lceil \lg N\rceil + 1 \qquad (\lg N \equiv \log_2 N)$$

and, counting the number of times the loop index j changes, we have

$$C_N = N + \sum_{\lceil \lg N\rceil > k \geqq 0} (N-2^k)$$

which evaluates to

$$C_N = N(\lceil \lg N\rceil + 1) - 2^{\lceil \lg N\rceil} + 1. \tag{1}$$

The number of comparisons is of order $(N \log N)$ so Batcher's merging algorithm cannot compare with the straightforward $O(N)$ algorithm on a serial computer. However, if parallelism is available, the comparisons on each stage can be performed independently, and the merge can be completed in $\lceil \lg N\rceil + 1$ parallel stages. Also, R. W. Floyd [13, p. 230] has shown that any merging method which can be represented as a network must use at least $\frac{1}{2}N \lg N + O(N)$ comparators to 2-sort N elements, so Batcher's method is, in this sense, nearly optimal.

The value of B does depend on the input distribution, and the subject of this paper is the analysis of the maximum and average values of this quantity when a random 2-ordered file is sorted. This is listed as an open problem by Knuth [13, p. 135]. The practical importance of this problem may be limited, since the method is best suited to a parallel implementation, and exchanges might not significantly affect the running time of a truly parallel implementation. However, it is essential to know the value of B for serial implementations, and, as we shall see, the analysis of B is of some theoretical interest. Our understanding of Batcher's method is incomplete without an understanding of how often it does exchanges. Moreover, the methods and results of the analysis may be applicable to the study of other algorithms.

To deal with the number of exchanges, it is useful to examine Knuth's alternate proof that the odd-even merge is valid. The proof is based on a natural correspondence between 2-ordered files of $2N$ elements and paths connecting opposite corners of $N\times N$ lattice diagrams. An example of this correspondence is shown in Fig. 4. Starting at the upper left corner, we form a path whose kth segment goes down if the kth smallest element is an odd position in the file, and to the right if the kth smallest element is in an even position in the file. We shall denote the lattice point reached after i steps down and j steps to the right by (i, j). The path must end up at (N, N) since there are N even positions and N odd positions, and the correspondence is clearly unique. One can think of the final sorted file as a chain with $2N$ links, and the path as the unique arrangement of the chain adhered at the upper left corner with each link vertical if the corresponding element is in an odd position in the file and horizontal if the corresponding element is in an even position in the file.

The sorted array corresponds to the diagonal path through the lattice whose first segment is vertical (the dotted line in Fig. 4), and the merging process consists of transformations from an arbitrary path to that particular path. As mentioned above, Batcher's method can be divided into $\lceil \lg N \rceil + 1$ stages of independent compare-exchange operations. The proof that the odd-even merge is valid consists of showing that the stages correspond to "folding" (interchanging horizontal and vertical) the path about certain diagonals in the lattice diagram.

For example, the first stage, which compare-exchanges $A[2]$ with $A[1]$, then $A[4]$ with $A[3]$, then $A[6]$ with $A[5]$, etc., corresponds to folding the path about the main diagonal. To show this, we first note that any path can be divided into sections which are either "high" (totally above the diagonal) or "low" (on or below the diagonal). (The path in Fig. 4 consists of a low section followed by a high section.) Now, the jth comparison in the first stage results in an exchange if and only if the jth horizontal path segment (which corresponds to $A[2j]$) appears before the jth vertical path segment (which corresponds to $A[2j-1]$). But this can happen if and only if both segments are above the diagonal. Therefore, all elements represented by high path sections are involved in exchanges and no elements represented by low sections are involved in exchanges. After the exchanges, low sections are unchanged, and horizontal and vertical are interchanged in high sections, making them low. In other words, the whole path is reflected down about the diagonal.

The first stage folds down about the main diagonal, ensuring that the path falls below the main diagonal, and successive stages fold up about the diagonal δ units below the main diagonal, ensuring that the path falls in a "band" between that diagonal and the main diagonal. After the stage when $\delta = 1$ the path must coincide with the main diagonal, and the corresponding permutation is sorted. Figure 5 shows the sample 2-ordered permutation in Fig. 4 being sorted. Shaded areas are the areas within which the path is guaranteed to fall, and each stage "folds" the shaded area in the middle. The reader may wish to check the correspondence and the proof by seeing that successive paths in Fig. 5 correspond to successive permutations in Fig. 1. (In particular, note that there are no exchanges on the third stage, and the path is unchanged.)

This proof that Batcher's method is valid also gives us an easy way to count the number of exchanges used to sort any particular 2-ordered permutation. First, we notice that if any segment on the path is on the main diagonal, then the element corresponding to it will not be involved in any exchanges during the sort (since it is in a

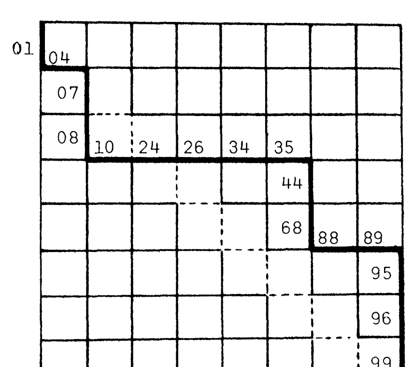


FIG. 4. *Lattice path for 2-ordered permutation of Fig.* 1.

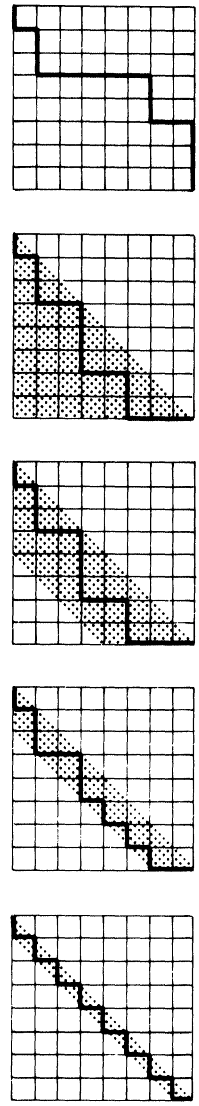


FIG. 5. *Sorting Fig.* 4.

"low" section for the first stage, and a "high" section for successive stages). If any segment on the path is on the diagonal one below the main diagonal, then the corresponding element must be involved in exactly one exchange (on the last stage). By following the "folding" process backwards in this way, we can assign a weight to each segment in the lattice which counts the number of exchanges the corresponding element will be involved in, if the path includes that segment. This process is illustrated, for $N=4$, in Fig. 6.

Now, for any path through the lattice, if we sum the weights of its segments and divide by two (since each exchange involves two elements), we get the total number of exchanges used to sort the corresponding 2-ordered permutation. In fact, the sum of the weights of a path's vertical segments must equal the sum of the weights of its horizontal segments, and both sums count the number of exchanges. From Fig. 7, which has only vertical weights, we see that the example in Fig. 4 takes 12 exchanges, which agrees with our count in Fig. 1. From now on, we shall consider only vertical weights.

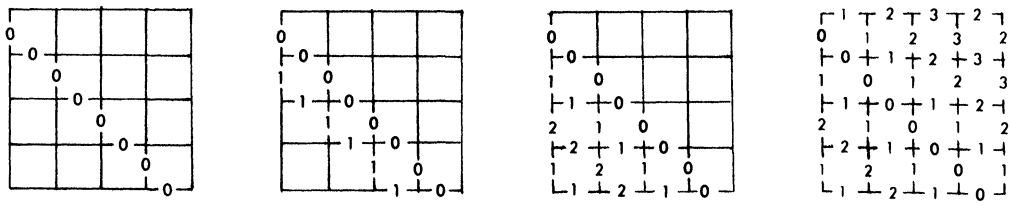


FIG. 6. *Assigning weights to the lattice.*

Continuing as in Fig. 6, the pattern of weights for general N is clear. First, we notice that all of the weights can be determined from the weights down the left edge. Since the folding is done along parallel diagonals, weights along diagonals are constant: if we denote the weight of the vertical segment from (i, j) to $(i+1, j)$ by $f(i, j)$, then we have

$$f(i,j)=\begin{cases} f(i-j,0) & \text{if } i \geqq j, \\ f(0,j-i) & \text{if } i \leqq j. \end{cases} \tag{2}$$

But from the first stage (the last "unfold") we know that

$$f(0, j+1)=f(j, 0)+1 \tag{3}$$

and from the other stages we can write down an algorithmic definition of $f(i, 0)$:

(4)
$f(0,0):=0$; $i:=1$;
loop:
  loop for $j:=i-1$ **step** -1 **until** 0: $f(i,0):=f(j,0)+1$; $i:=i+1$ **repeat**;
repeat;

In other words, in order to write down the values of $f(i, 0)$ for all i, first write down "0"; then repeatedly apply the following procedure: append to the string of numbers already written down the same string, but in reverse order, with each number incremented by 1. The value of $f(i, 0)$ is the ith number written.

This function, which is central to the study of data movement in Batcher's method, has a number of interesting properties. Since we shall be using it extensively, it will be convenient to drop the second argument and work with a more mathematical recursive definition:

$$\begin{aligned} f(0)&=0, \\ f(2^n+j)&=f(2^n-1-j)+1, \qquad n \geqq 0, \quad 0 \leqq j < 2^n. \end{aligned} \tag{5}$$

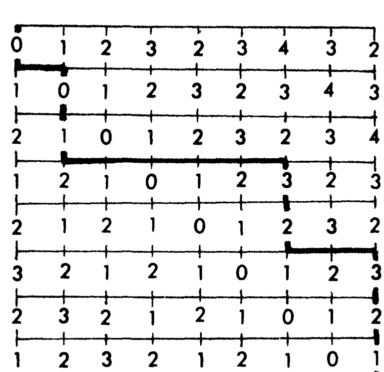
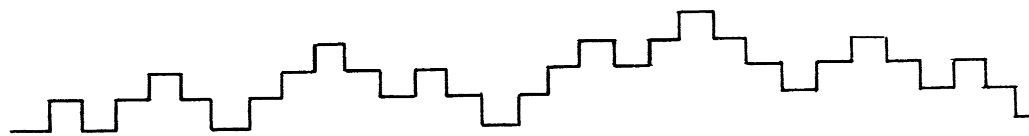


FIG. 7. *Vertical weights for* $N = 8$.



k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
f(k)	0	1	2	1	2	3	2	1	2	3	4	3	2	3	2	1	2	3	4	3	4	5	4	3	2	3	4	3	2	3	2	1
F(k)	0	0	1	3	4	6	9	11	12	14	17	21	24	26	29	31	32	34	37	41	44	48	53	57	60	62	65	69	72	74	77	79
∇f(k)		1	1	-1	1	1	-1	-1	1	1	1	-1	-1	1	-1	-1	1	1	1	-1	1	1	-1	-1	-1	1	1	-1	-1	1	-1	-1

FIG. 8. *The weight function.*

From this definition, we can explicitly evaluate the function for some arguments. For example, taking $j=2^n-1$, we find that $f(2^n-1)=1$ for $n\geqq 1$; then taking $j=0$ gives $f(2^n)=2$ for $n\geqq 0$ and taking $j=2^{n-1}$ gives $f(3\cdot 2^{n-1})=2$ for $n\geqq 2$. For other arguments, things are more complicated. However, there is a simple interpretation based on the binary representation of the argument. The binary representation of 2^n-1-j $(0\leqq j<2^n)$ is the "ones' complement" of the binary representation of 2^n+j (change 0's to 1's and 1's to 0's; then ignore leading zeros). Therefore, for example, $f(999)=f(1111100111_2)=f(11000_2)+1=f(111_2)+2=f(0)+3=3$. The value of $f(k)$ for all k is exactly the number of times the binary representation of k changes parity. Figure 8 gives the value of $f(k)$ for $0\leqq k<32$ along with a graph of the function and values for $F(k)=\sum_{0\leqq j<k} f(j)$ (the area under the curve) and $\nabla f(k)\equiv f(k)-f(k-1)$ (the slope), which we shall have use for later.

2. The worst case. To find the maximum number of exchanges that Batcher's algorithm will require, we can use the lattice diagram directly. The maximum number of exchanges is just the maximum possible weight of a path in the lattice diagram. Figure 9 shows the paths of highest weight for $N=2, 4, 8, 16$.

A cursory inspection of Fig. 9 could lead to the conjecture that, at least for $N=2^n$, the worst case might be the unique path through the lattice which contains the highest weights. Unfortunately, the situation is more complicated than this, as illustrated in Fig. 10 for $N=32$. However, it does turn out that we need to examine only a few paths. Consider the paths through the lattice defined, for each integer k, as follows: proceed right along the top until encountering the first line with weight k. Then proceed down and to the right (along the diagonal of lines with weight k). After reaching the right edge of the lattice, proceed down to the corner. Figure 11 illustrates these paths, which we shall refer to as *major diagonals*, for $N=32$. (Note that the last major diagonal is the unique path containing the highest weights.)

LEMMA. *The path of highest weight through the lattice must be one of the major diagonals.*

Proof. Clearly, for any path with segments below the main diagonal (the first major diagonal), there is a path of higher weight whose segments are all on or above

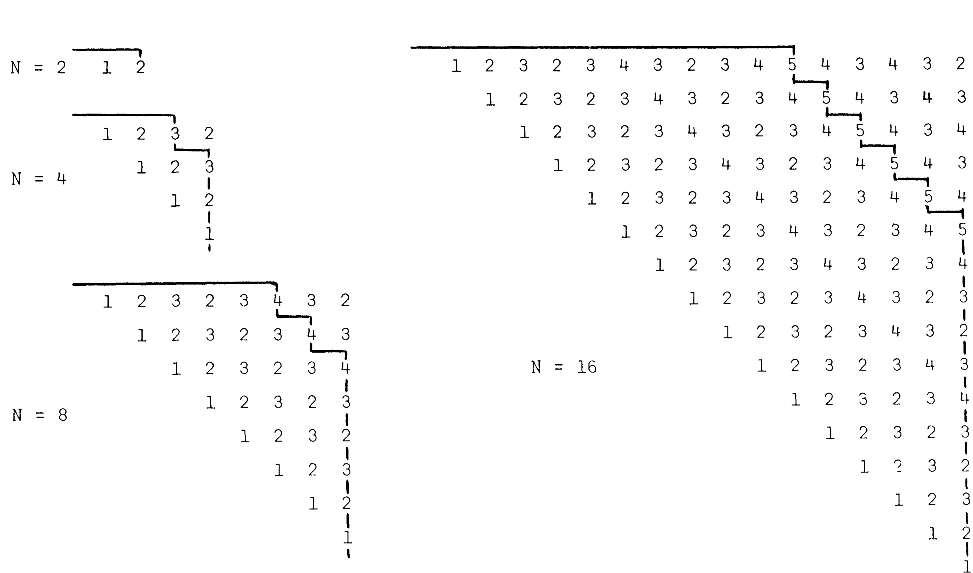


FIG. 9. *Worst-case lattice paths.*

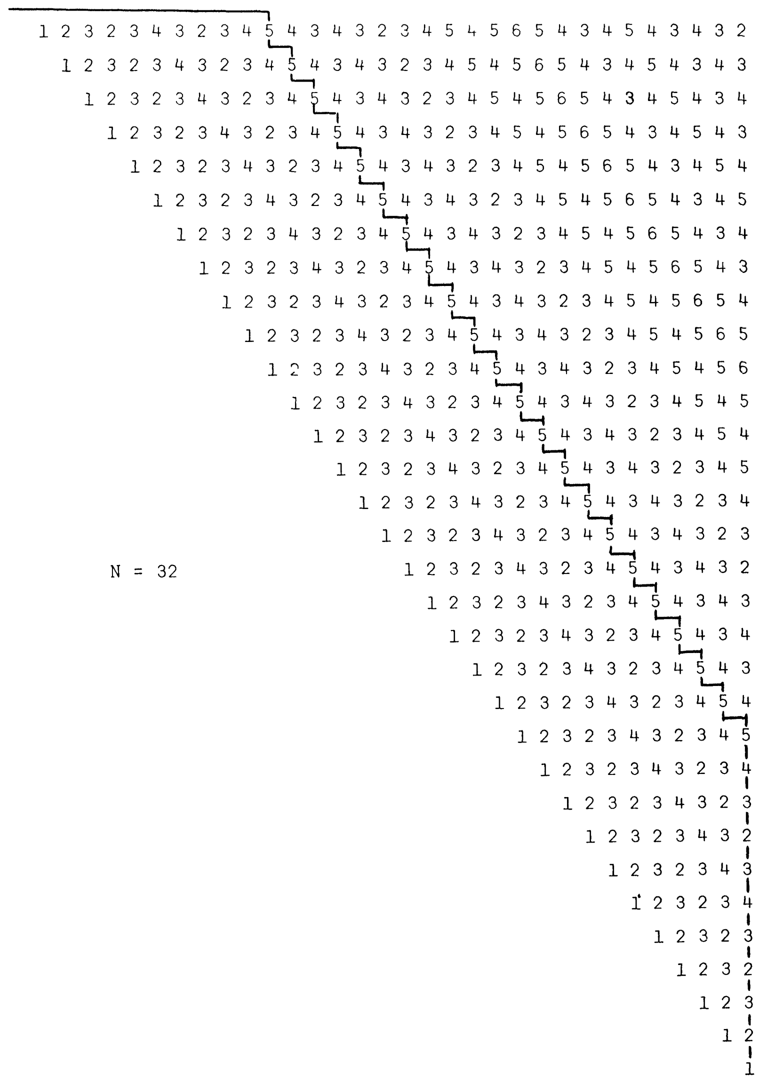


FIG. 10. *Worst-case path not containing highest weights.*

the main diagonal. Now, for any such path, consider the rightmost major diagonal which it crosses (has a segment in common with). The path must contain, sometime after the crossing, all of the weights which the major diagonal has on its vertical segment. None of the remaining weights can be higher than those on the major diagonal, because a higher weight would imply that the path crosses a major diagonal farther to the right. Therefore, for every path through the lattice, we can find a major diagonal whose weight is at least as high. □

Our problem is now reduced to finding the weights of all the major diagonal paths, and the maximum of these. To do so, we need to define

$$f^{-1}(k)=\{\text{smallest } j \text{ for which } f(j)=k\}$$

and

$$F(k)=\sum_{0\leq j<k} f(j).$$

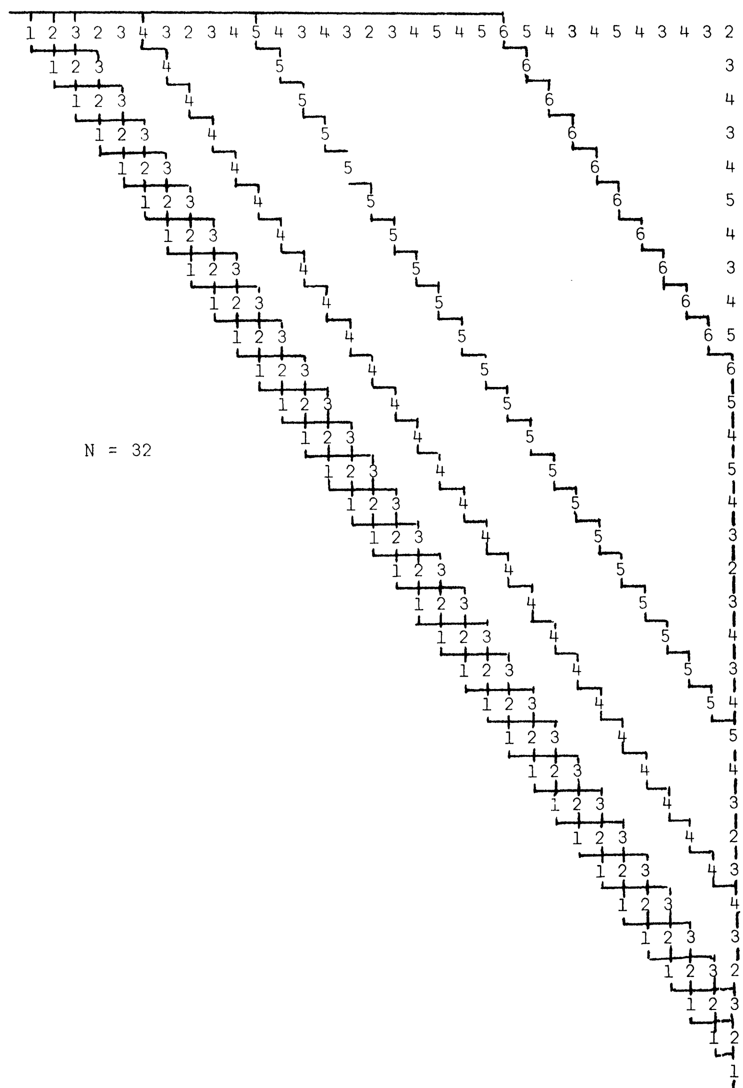


FIG. 11. *Major diagonals.*

The $(k+1)$st major diagonal path has $N-f^{-1}(k)$ segments along the diagonal, with weights $(k+1)$, and $f^{-1}(k)$ segments along the right edge, with total weight $F(f^{-1}(k))+f^{-1}(k)$. Therefore, if we let $w(k)$ denote the total weight of the $(k+1)$st major diagonal path, then

$$\begin{aligned} w(k) &= F(f^{-1}(k))+f^{-1}(k)+(k+1)(N-f^{-1}(k)) \\ &= (k+1)N+F(f^{-1}(k))+kf^{-1}(k), \end{aligned}$$

and we need to derive explicit expressions for $f^{-1}(k)$ and $F(f^{-1}(k))$.

A recurrence for $f^{-1}(k)$ follows immediately from the way that the function $f(k)$ "reflects" between powers of two. From (5) it is easy to prove by induction the difference between $f^{-1}(k-1)$ and $2^{k-1}-1$ must be the same as the difference between 2^{k-1} and $f^{-1}(k)$. (Also see Fig. 8.) In other words,

$$2^{k-1}-1-f^{-1}(k-1)=f^{-1}(k)-2^{k-1}.$$

Multiplying by $(-1)^k$ and telescoping, we find that

$$(-1)^k f^{-1}(k) = \sum_{0\le j\le k} (-2)^j - \sum_{0\le j\le k} (-1)^j$$

which leads, after the two geometric sums are evaluated, to the result

$$f^{-1}(k) = \frac{1}{6}(2^{k+2} - (-1)^k - 3). \tag{6}$$

As expected, these are all the numbers $(0, 1, 2, 5, 10, 21, \cdots)$ whose binary representations alternate between 0 and 1. These numbers change parity most often, and so have the highest values of $f(k)$.

The calculation of $F(f^{-1}(k))$ is more complicated. First, we can set up a recurrence similar to the one which defines $f(k)$. Suppose that $2^{n-1} < k \le 2^n$. We separate off the first 2^{n-1} terms of the sum and then apply the recurrence for $f(k)$ to the remaining terms:

$$\begin{aligned} F(k) &= \sum_{0\le j<2^{n-1}} f(j) + \sum_{2^{n-1}\le j<k} f(k) \\ &= F(2^{n-1}) + \sum_{0\le j<k-2^{n-1}} f(2^{n-1}+j) \\ &= F(2^{n-1}) + \sum_{0\le j<k-2^{n-1}} (f(2^{n-1}-1-j)+1) \\ &= F(2^{n-1}) + \sum_{2^n-k\le j<2^{n-1}} f(j) + k - 2^{n-1} \\ &= 2F(2^{n-1}) - F(2^n-k) + k - 2^{n-1} \quad \text{for } 2^{n-1} < k \le 2^n. \end{aligned}$$

In particular, if we take $k = 2^n$, then the formula becomes $F(2^n) = 2F(2^{n-1}) + 2^{n-1}$, which telescopes immediately to the solution

$$F(2^n) = n2^{n-1}.$$

Substituting this value, we find that

$$F(k) = -F(2^n - k) + (n-2)2^{n-1} + k \quad \text{for } 2^{n-1} < k \le 2^n. \tag{7}$$

As before, the form of this recurrence clearly suggests that the value of $F(k)$ depends on the binary representation of k (and the dependence is much more complicated than for $f(k)$). Fortunately, the points $f^{-1}(k)$ at which we need to evaluate the function have a simple binary representation. We can get an explicit formula for $F(f^{-1}(k))$ by noticing from the ones' complement of the binary representation that $2^k - f^{-1}(k) - 1 = f^{-1}(k-1)$, so $F(2^k - f^{-1}(k)) = F(2^k - f^{-1}(k) - 1) + f(2^k - f^{-1}(k) - 1) = F(f^{-1}(k-1)) + f(f^{-1}(k-1)) = F(f^{-1}(k-1)) + k - 1$, and, since $2^{k-1} < f^{-1}(k) \le 2^k$, the recurrence (7) becomes

$$F(f^{-1}(k)) = -F(f^{-1}(k-1)) - (k-1) + (k-2)2^{k-1} + f^{-1}(k).$$

This recurrence, after both sides are multiplied by $(-1)^k$, telescopes into a summation (note that $F(f^{-1}(0)) = F(0) = 0$):

$$(-1)^k F(f^{-1}(k)) = \sum_{1\le j\le k} (j-1)(-1)^{j-1} - \sum_{1\le j\le k} (j-2)(-2)^{j-1} + \sum_{1\le j\le k} f^{-1}(j)(-1)^j.$$

After substituting for $f(j)$, we are left with a number of elementary sums: they can all be evaluated using the well-known identities for $\sum_{0\le j\le k} x^j$ and $\sum_{0\le j\le k} jx^j$ (see, for example, Knuth [12, exercise 1.2.3-16]) with the result

$$F(f^{-1}(k))=\frac{1}{18}((3k-1)2^{k+1}-9k-(3k-2)(-1)^k). \tag{8}$$

Comparing this with the formula $F(2^n)=n2^{n-1}$, we find that both are of the form $F(N)=\frac{1}{2}N\lg N=O(N)$. In fact, it is possible to prove by induction that this does hold for all N, but the linear term is a complicated function of the binary representation of N.

Substituting these values for $f^{-1}(k)$ and $F(f^{-1}(k))$ into the formula given above for the total weight of the $(k+1)$st major diagonal path, we get the expression

$$\begin{aligned} w(k)&=(k+1)N+\frac{1}{18}((3k-1)2^{k+1}-9k-(3k-2)(-1)^k)-\frac{1}{6}k(2^{k+2}-(-1)^k-3) \\ &=(k+1)N-\frac{1}{9}((3k+1)2^k-(-1)^k). \end{aligned} \tag{9}$$

This function is clearly increasing for small k and decreasing for large k. The maximum number of exchanges required by Batcher's algorithm is the maximum value of the function. Note that the total weight of the last major diagonal is $\frac{2}{3}N\lg N+O(N)$. The following theorem shows that the proper choice of k leads to a path of much higher weight.

THEOREM 1. *Let $B_N^{\max}$ denote the maximum number of exchanges required when Batcher's odd-even merge is applied to a 2-ordered file of $2N$ elements. Then*

$$B_N^{\max}=(k'+1)N-\frac{1}{9}((3k'+1)2^{k'}-(-1)^{k'})$$

where k' is the largest integer satisfying $\frac{1}{9}((3k'+4)2^{k'-1}-\frac{2}{9}(-1)^{k'})\leqq N$. Asymptotically,

$$B_N^{\max}=N\lg N-N\lg\lg N+O(N).$$

Proof. Following the discussion above, the lemma says that we need only consider the major diagonals. We have:

$$B_N^{\max}=\max_{0\leqq k\leqq k''}(w(k)),$$

where k'' is the index of the last major diagonal (the largest integer satisfying $f(k'')\leqq N$). To calculate this maximum, consider the difference

$$w(k)-w(k-1)=N-\frac{1}{9}((3k+4)2^{k-1}-2(-1)^k).$$

The function $w(k)$ increases as long as this difference is positive, then decreases when the difference is negative. Clearly the maximum is $w(k')$, where k' is the largest integer for which the difference is positive. To complete the proof, it is necessary to show that this maximum is realizable, i.e. that $f^{-1}(k')\leqq N$. This is easily verified: we have $N\geqq\frac{1}{9}((3k'+4)2^{k'-1}-2(-1)^{k'})=\frac{1}{6}(\frac{2}{3}(3k'+4)2^{k'-1}-\frac{4}{3}(-1)^{k'})\geqq\frac{1}{6}(2^{k'+2}-(-1)^{k'}-3)=f^{-1}(k')$.

To find an asymptotic estimate of how the maximum grows with N, we start with the inequalities which define k':

$$\frac{1}{9}(3k'+4)2^{k'-1}-\frac{2}{9}(-1)^{k'}\leqq N<\frac{1}{9}(3k'+7)2^{k'}+\frac{2}{9}(-1)^{k'}.$$

After ignoring the $(-1)^{k'}$ factors (the inequalities still hold without them), if we multiply by 3, take logs (base 2) and solve for k', we get an inverted form of this formula:

$$\lg\left(3N-\frac{2}{3}\right)-\lg\left(k'+\frac{7}{3}\right)<k'<\lg\left(3N+\frac{2}{3}\right)-\lg\left(k'+\frac{4}{3}\right)+1.$$

These inequalities can now be iterated to give

$$\lg\left(3N-\frac{2}{3}\right)-\lg\left(\lg\left(3N+\frac{2}{3}\right)-\lg\left(k'+\frac{4}{3}\right)+1+\frac{7}{3}\right)$$
$$<k'<\lg\left(3N+\frac{2}{3}\right)-\lg\left(\lg\left(3N-\frac{2}{3}\right)-\lg\left(k'+\frac{7}{3}\right)+\frac{4}{3}\right)+1.$$

Now both sides reduce to the same asymptotic expression; we must have

$$k'=\lg 3N-\lg\lg 3N+O(1)$$
$$=\lg N-\lg\lg N+O(1)$$

and substituting this into the formula for $B_N^{\max}$ leads to the stated asymptotic estimate. □

The easiest way to actually compute $B_N^{\max}$ for any practical value of N is to use a table, since k' takes on relatively few values for realistic N. Table 1 gives the values of $B_{N_k}^{\max}$ for the inflection points N_k: numbers of the form $\frac{1}{9}((3k+4)2^{k-1}-2(-1)^k)$. Between the kth and $(k+1)$st inflection points the function is linear in N with slope $(k+1)$. Therefore, to compute $B_N^{\max}$ for arbitrary N, find the largest k for which $N_k \leqq N$, call it k', and set $B_N^{\max}=B_{N_{k'}}^{\max}+(k'+1)(N-N'_k)$. Table 2 shows the results of such a computation for $N=2^n$, $0\leqq n\leqq 20$.

TABLE 1
Inflection points for the worst case.

k	$N_k=\frac{1}{9}((3k+4)2^{k-1}-2(-1)^k)$	$B_{N_k}^{\max}$	C_{N_k}	$B_{N_k}^{\max}/C_{N_k}$
1	1	1	1	1.0000
2	2	3	3	1.0000
3	6	15	17	.8823
4	14	47	55	.8545
5	34	147	175	.8400
6	78	411	497	.8270
7	178	1,111	1,347	.8248
8	398	2,871	3,469	.8276
9	882	7,227	8,679	.8327
10	1,934	17,747	21,161	.8387
11	4,210	42,783	50,749	.8430
12	9,102	101,487	120,147	.8447
13	19,570	237,571	280,353	.8474
14	41,870	549,771	646,255	.8507
15	89,202	1,259,751	1,474.565	.8543
16	189,326	2,861,735	3,335,241	.8580
17	400,498	6,451,659	7,485,673	.8619
18	844,686	14,447,043	16,689,831	.8656
19	1,776,754	32,156,335	36,991,437	.8692

TABLE 2
The worst case at inflection points for the number of comparators.

N	k'	B_N^{max}	C_N	B_N^{max}/C_N
1	1	1	1	1.0000
2	2	3	3	1.0000
4	2	9	9	1.0000
8	3	23	25	.9200
16	4	57	65	.8769
32	4	137	161	.8509
64	5	327	385	.8493
128	6	761	897	.8484
256	7	1,735	2,049	.8468
512	8	3,897	4,609	.8455
1,024	9	8,647	10,241	.8444
2,048	10	19,001	22,529	.8434
4,096	10	41,529	49,153	.8449
8,192	11	90,567	106,497	.8504
16,384	12	196,153	229,377	.8552
32,768	13	422,343	491,521	.8593
65,536	14	904,761	1,049,477	.8621
131,072	15	1,929,671	2,228,225	.8660
262,144	16	4,099,641	4,718,593	.8668
524,288	17	8,679,879	9,961,473	.8713
1,048,576	18	18,320,953	20,971,521	.8736

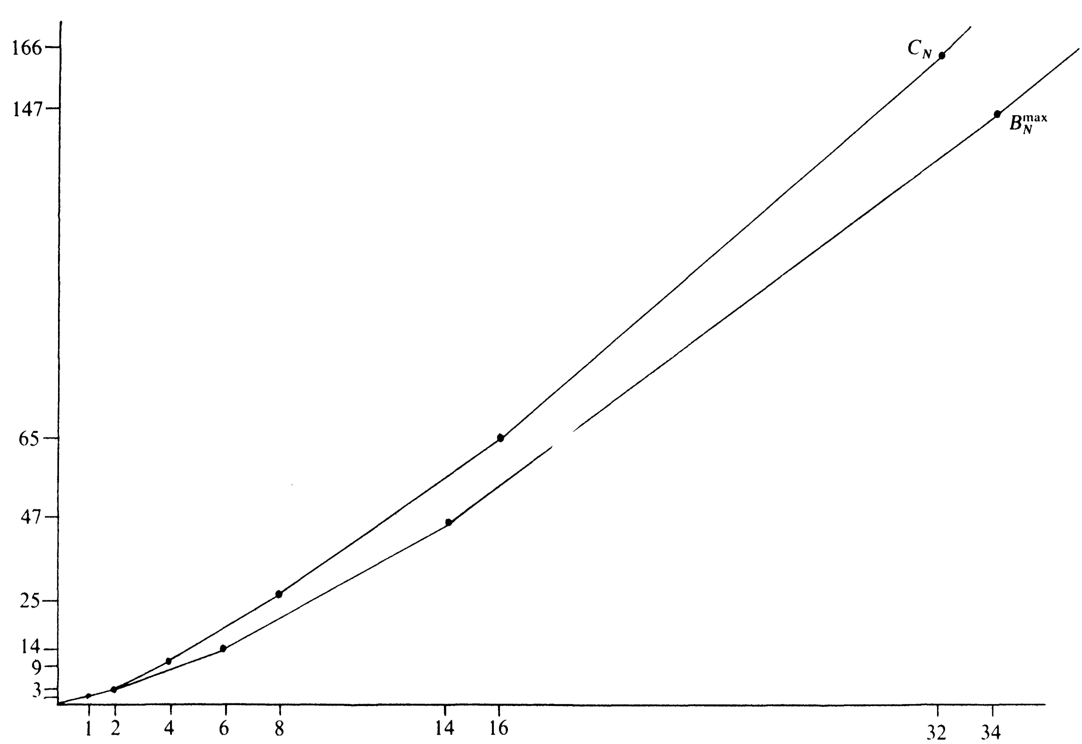


FIG. 12. *Number of comparators and maximum number of exchanges.*

Also given in Tables 1 and 2 is another quantity of interest: the percentage of comparators that perform exchanges in the worst case (the ratio $B_N^{\max}/C_N$). The graphs of C_N and $B_N^{\max}$ are both piecewise linear, with slope incrementing by 1 at each inflection point. This is illustrated in Fig. 12, for small N, where the effect is most pronounced. Both curves are therefore concave upwards, so they are closest together at the inflection points for C_N (numbers of the form 2^n) and farthest apart at the inflection points for $B_N^{\max}$. Tables 1 and 2 therefore show that the ratio is between 82% and 87% for all but very small and very large values of N. As $N\to\infty$ the ratio (slowly) approaches 1, which follows from a simple asymptotic calculation:

$$B_N^{\max}/C_N = \frac{N \lg N - N \lg\lg N + O(N)}{N \lg N + O(N)}$$

(10)
$$= \left(1 - \frac{\lg\lg N}{\lg N} + O\left(\frac{1}{\lg N}\right)\right)\left(1 + O\left(\frac{1}{\lg N}\right)\right)$$

$$= 1 - \frac{\lg\lg N}{\lg N} + O\left(\frac{1}{\lg N}\right).$$

The value of N must be truly astronomical for the ratio to be close to 1.

3. The average case. The lattice diagram correspondence of § 1 leads to an expression for B_N, the average number of comparisons taken by Batcher's method to sort a random 2-ordered file of $2N$ elements. The derivation is long, and conveniently divides into two parts. First, we shall perform some manipulations which are somewhat independent of our particular weight function $f(j)$. and so lead to results applicable to the analysis of other properties of 2-ordered permutations (or other merging algorithms). The second part of the derivation uses complex analysis and some particular properties of $f(j)$, and leads to a method for computing B_N to any desired asymptotic accuracy.

One way to determine B_N, using the lattice diagram correspondence, would be to find the weight of each of the $\binom{2N}{N}$ paths through the lattice, sum them, and divide by $\binom{2N}{N}$. An alternate way is to find the number of paths which pass through each vertical line in the lattice, multiply by the weight, sum over all vertical lines, and then divide by $\binom{2N}{N}$. In § 1, we defined the weight of the vertical segment from (i, j) to $(i+1, j)$ to be $f(i, j)$ and derived some simple properties of this function. Now, the number of paths from $(0, 0)$ to (i, j) is clearly $\binom{i+j}{i}$, and the number of paths from $(i+1, j)$ to (N, N) is $\binom{2N-i-j-1}{N-j}$, so the total number of paths which pass through the vertical segment from (i, j) to $(i+1, j)$ is the product of these two binomial coefficients. Therefore,

(11)
$$\binom{2N}{N} B_N = \sum_{0\le i<N}\ \sum_{0\le j\le N} f(i, j)\binom{i+j}{i}\binom{2N-i-j-1}{N-j}.$$

This can be transformed into an expression involving the single argument weight function defined in (5) because of the symmetries which are available. The first step is to split the sum on j:

$$\binom{2N}{N}B_N=\sum_{0\leqq i<N}\sum_{0\leqq j\leqq i}f(i,j)\binom{i+j}{i}\binom{2N-i-j-1}{N-j}$$
$$+\sum_{0\leqq i<N}\sum_{i<j\leqq N}f(i,j)\binom{i+j}{i}\binom{2N-i-j-1}{N-j}.$$

If we change j to $i-j$ in the first term and change j to $i+j+1$ and then i to $N-1-i$ in the second term, the terms can be recombined:

$$\binom{2N}{N}B_N=\sum_{0\leqq i<N}\sum_{0\leqq j\leqq i}f(i,i-j)\binom{2i-j}{i}\binom{2N-2i+j-1}{N-i+j}$$
$$+\sum_{0\leqq i<N}\sum_{0\leqq j\leqq i}f(N-1-i,N-i+j)\binom{2N-2i+j-1}{N-i-1}\binom{2i-j}{i-j}$$
$$=\sum_{0\leqq i<N}\sum_{0\leqq j\leqq i}(f(i,i-j)+f(N-1-i,N-i+j))\binom{2i-j}{i-j}\binom{2N-2i+j-1}{N-i-1}.$$

Now, equations (2) and (3) in § 1 tell us that $f(i,i-j)=f(j,0)$ and $f(N-1-i,N-i+j)=f(0,j+1)=f(j,0)+1$. Adopting the shorthand $f(j,0)\equiv f(j)$, we have

$$\binom{2N}{N}B_N=\sum_{0\leqq i<N}\sum_{0\leqq j\leqq i}(2f(j)+1)\binom{2i-j}{i-j}\binom{2N-2i+j-1}{N-i-1}.$$

Interchanging the order of summation and changing i to $i+j$ gives

$$\binom{2N}{N}B_N=\sum_{0\leqq j<N}(2f(j)+1)\sum_i\binom{2i+j}{i}\binom{2N-2i-j-1}{N-i-j-1}.$$

The inner sum remaining in this expression was studied as far back as 1902 by Jensen [9], who gave an identity which implies that

$$\sum_i\binom{2i+j}{i}\binom{2N-2i-j-1}{N-j-i-1}=\sum_{i\geqq 0}\binom{2N-1-i}{N-i-j-1}2^i$$

(see also Gould and Kaucký [8] or Knuth [12, exercise 1.2.6-28] for more general versions of this identity). This particular sum can be simplified even further, by applying the addition formula for binomial coefficients to set up a recurrence relation describing an alternate form of the sum. Denoting the sum by S_{Nj}, we have

$$S_{Nj}\equiv\sum_{i\geqq 0}\binom{2N-1-i}{N-i-j-1}2^i$$
$$=\sum_{i\geqq 0}\left(\binom{2N-2-i}{N-i-j-1}+\binom{2N-2-i}{N-i-j-2}\right)2^i$$
$$=\sum_{i\geqq 1}\left(\binom{2N-1-i}{N-i-j}+\binom{2N-1-i}{N-i-j-1}\right)2^{i-1}$$
$$=\frac{1}{2}S_{N(j-1)}+\frac{1}{2}S_{Nj}-\frac{1}{2}\binom{2N}{N-j}.$$

This implies that $S_{Nj}=S_{N(j+1)}+\binom{2N}{N-j-1}$, which telescopes to give the alternate form

$$\sum_i \binom{2i+j}{i}\binom{2N-2i-j-1}{N-i-j-1} = \sum_{0 \leqq k < N-j} \binom{2N}{k}. \tag{12}$$

When substituted into our formula for B_N, this leads to the following result:

THEOREM 2. *For any assignment of weights to an $N \times N$ lattice satisfying $f(i,j) = f(i-j, 0)$ for $i \geqq j$, $f(i,j) = f(0, j-i)$ for $j \leqq i$ and $f(0, j+1) = f(j, 0) + 1$, the average weight of a path through the lattice is*

$$B_N = \sum_{k \geqq 1} \frac{\binom{2N}{N-k}}{\binom{2N}{N}}(2F(k)+k)$$

where $F(k) = \sum_{0 \leqq j < k} f(j)$ with $f(j) \equiv f(j, 0)$.

Proof. From the discussion above, we have

$$\binom{2N}{N} B_N = \sum_{0 \leqq j < N} (2f(j)+1) \sum_{0 \leqq k < N-j} \binom{2N}{k}$$

which can be transformed into the stated result by changing k to $N-k$ and interchanging the order of summation. □

To proceed further, we need to examine the functions $f(k)$ and $F(k)$ in much more detail.

Digressing slightly, we can now easily compute the average number of inversions in a 2-ordered permutation as an example of the use of Theorem 2. (An inversion is an index pair (i, j) satisfying $i < j$ and $A[i] > A[j]$.) The lattice diagram correspondence and the initial expression (11) for B_N above are taken from Knuth's treatment of this problem [13, pp. 86–88 and exercises 5.2.1-12, 14, 15]. Knuth shows that the number of inversions in a 2-ordered permutation is equal to the area between its path in the lattice and the main diagonal. (Proof: changing ⌞ to ⌝ below the diagonal or ⌝ to ⌞ above the diagonal reduces the number of inversions by one and reduces the area by one unit.) The permutation in Fig. 4 has 12 inversions. The appropriate assignment of weights to the lattice is to take $f(i, j) = |i-j|$. This function satisfies (2) and (3), and we have $f(k) = k$ and $F(k) = \binom{k}{2}$. Then from the theorem we find that the average number of inversions must be

$$\sum_{k \geqq 1} \frac{\binom{2N}{N-k}}{\binom{2N}{N}} k^2.$$

This sum can be easily evaluated by writing

$$\binom{2N}{N-k} k^2 = \binom{2N}{N-k}(N^2 - (N-k)(N+k)) = N^2\binom{2N}{N-k} - 2N(2N-1)\binom{2N-2}{N-k-1}$$

and applying the identity $\sum_{k\geqq 1}\binom{2N}{N-k}+\frac{1}{2}\left(2^{2N}-\binom{2N}{N}\right)$. These calculations lead to the result

$$\frac{N2^{2N-2}}{\binom{2N}{N}}$$

for the average number of inversions in a 2-ordered permutation of $2N$ elements. (This checks with Knuth's result, but his derivation depends on particular properties of $f(k)=k$.) Knuth suggests that such a simple answer deserves a simple derivation; perhaps a direct combinatorial derivation of Theorem 2 could be devised. In any case, the weight function $f(k)$ for Batcher's method is much more complicated than $f(k)=k$ (we don't even have a closed formula for it), and our problem will involve much more analysis.

Theorem 2 does lead to an easy way to *compute* B_N for all practical values of N. Expanding the binomial coefficients in their factorial representations, we find that

$$B_N=\sum_{1\leqq k\leqq N}(2F(k)+k)\prod_{0\leqq j<k}\frac{N-j}{N+j+1}.$$

From this representation, we can see that the exact value of B_N can be computed in $O(N)$ steps, as follows:

(13)
```
product := 1; sum := 0;
loop for 1 ≦ k ≦ N:
    product := product*(N − k + 1)/(N + k);
    sum := sum + (2*F(k) + k)*product;
repeat;
```

This program assumes that $F(k)$ has been precomputed and stored in an array $F(1:N)$, say by using (4) to compute $f(k)$ and then passing through the array once more to compute $F(k)$. This requirement for N memory cells can be removed by computing $F(k)$ incrementally within the loop. [We have $F(k)=F(k-1)+f(k-1)$, and $f(k)$ can be computed from $f(k-1)$ by looking at the binary representations of $(k-1)$ and k. The binary representation of k is obtained from the binary representation of $k-1$ by changing the rightmost 0 to 1 and all the 1's to its right to 0's. (All numbers are assumed to have 0 as the leftmost digit.) This will increment by 1 the number of times the binary representation changes parity (the value of f) if the bit to the left of the rightmost 0 in $(k-1)_2$ is 0; otherwise it will decrement f by 1. Therefore, we need only test this one bit: this can be done by performing an "exclusive or" of $(k-1)_2$ with $(k)_2$, adding 1, then "and"ing the result with $(k-1)_2$ (or $(k)_2$). If the result is 0, then $f(k)=f(k-1)+1$, otherwise $f(k)=f(k-1)-1$.] The program can be further improved because the terms become very, very small as k gets large. If we put in a test to leave the loop when the terms to be added become smaller than the smallest representable number in our computer, then it turns out that the loop is iterated only about $O(\sqrt{N})$ times for large N (we shall see why later). Thus exact values of B_N can be computed very quickly.

TABLE 3
Average number of exchanges (exact).

N	B_N	$\frac{B_N-(1/4)N \lg N}{N}$
1	.500	.50000000
2	1.333	.41666667
4	3.600	.40000000
8	9.131	.39141414
16	22.221	.38881721
32	52.370	.38657069
64	120.735	.38647725
128	273.339	.38546127
256	610.795	.38591836
512	1,349.217	.38519013
1,024	2,955.039	.38578023
2,048	6,420.731	.38512273
4,096	13,868.014	.38574580
8,192	29,778.788	.38510590
16,384	63,663.918	.38573720
32,768	135,499.012	.38510170
65,536	287,423.532	.38573505
131,072	607,531.912	.38510065
262,144	1,280,765.989	.38573451
524,288	2,692,271.510	.38510038
1,048,576	5,647,351.813	.38573438

Table 3 shows exact values of B_N for $N=2^n$, computed in this way. By taking differences in this table, it is quickly discovered that these numbers grow with $N \lg N$, and the coefficient is apparently 1/4. Subtracting $\frac{1}{4}N \lg N$ from B_N and dividing by N gives the third column, which leads to the immediate conjecture that

$$B_N \approx \frac{1}{4} N \lg N + .385N$$

at least for $N=2^n$. In fact, a quick calculation with (13) proves that this formula is accurate to within 0.1% for $2^7 \leqq N \leqq 2^{20}$ (and to within 1% for $2<N<2^7$). From a practical standpoint, we are done, since we can accurately calculate B_N for any realistic value of N. From a theoretical standpoint, this answer is somewhat unsatisfactory, and the rest of the paper will be devoted to an analytic verification of this result. It turns out that precise formulas for B_N can be derived to any desired asymptotic accuracy; in particular, the coefficient of the linear term can be expressed in terms of classical mathematical functions. The derivation is an interesting example of a difficult type of asymptotic analysis, and it uncovers some interesting aspects of the structure of Batcher's method.

It will be convenient to begin by using the addition formula for binomial coefficients to transform the equation in Theorem 2 for B_N into a sum involving $\nabla f(k)$, which is simpler to work with than $F(k)$. First, just as in the derivation for the number of inversions, we can perform the summation $\sum_{k \geqq 1} \binom{2N}{N-k} k = \frac{1}{2} N \binom{2N}{N}$, which leaves

$$\begin{aligned}\binom{2N}{N}\left(B_N-\frac{N}{2}\right)&=2\sum_{k\geqq 1}\binom{2N}{N-k}F(k)\\&=2\sum_{k\geqq 1}\left(\binom{2N-2}{N-k}+2\binom{2N-2}{N-k-1}+\binom{2N-2}{N-k-2}\right)F(k)\\&=2\left(\sum_{k\geqq 0}\binom{2N-2}{N-k-1}F(k+1)+2\sum_{k\geqq 1}\binom{2N-2}{N-k-1}F(k)\right.\\&\qquad\left.+\sum_{k\geqq 2}\binom{2N-2}{N-k-1}F(k-1)\right)\\&=4\binom{2N-2}{N-1}\left(B_{N-1}-\frac{N-1}{2}\right)+2\sum_{k\geqq 1}\binom{2N-2}{N-k-1}\nabla f(k).\end{aligned}$$

When both sides are divided by 4^N, this recurrence telescopes to a sum, and leads to the formulation

$$B_N=\frac{1}{2}\frac{4^N}{\binom{2N}{N}}\sum_{1\leqq j<N}\frac{\binom{2j}{j}}{4^j}\sum_{k\geqq 1}\frac{\binom{2j}{j-k}}{\binom{2j}{j}}\nabla f(k)+\frac{1}{2}N. \tag{14}$$

We shall now concentrate on evaluating the inner sum

$$b_j=\sum_{k\geqq 1}\frac{\binom{2j}{j-k}}{\binom{2j}{j}}\nabla f(k). \tag{15}$$

After we have derived an asymptotic expression for b_j, we shall easily be able to deal with B_N.

Formulas of this type (involving a sum over the lower index of a binomial coefficient) appear relatively frequently in combinatorial analysis and the analysis of algorithms. We have already seen one example, counting inversions in a 2-ordered permutation. Knuth [13] gives several other specific examples which arise in the analysis of algorithms: bubble sort, digital searching, and radix exchange sort. Paths in a lattice may also be used to model other combinatorial problems, such as tree enumeration and the classical ballot problem, and formulas similar to Theorem 2 arise in the analysis. The method that we shall use is called the "gamma-function" method and is attributed by Knuth to N. G. de Bruijn. A derivation using the method is outlined in a paper on tree enumeration by de Bruijn, Knuth and S. O. Rice [5], and a similar description may be found in Knuth [13, pp. 132–134]. However, it will be useful to present the method in some detail here because our function $\nabla f(k)$ is more complicated than the corresponding functions for the prior derivations.

One goal in an asymptotic derivation is to use methods which could, at least in principle, yield an answer good to any given asymptotic accuracy. We shall be content to get a formula for B_N good to within $O(\sqrt{N}\log N)$; we are most interested in the coefficients of the $N\log N$ and linear terms. It turns out that it is sufficient to get b_j

with $O(j^{-1/2})$ to achieve this accuracy. In both cases, the methods can yield better asymptotic accuracy, if desired.

The first step in evaluating b_j is to use Stirling's approximation to replace the binomial coefficients with an exponential. Stirling's approximation says that

$$\ln n! = \left(n+\frac{1}{2}\right)\ln n - n + \ln\sqrt{2\pi} + O\left(\frac{1}{n}\right).$$

Applying this to the binomial coefficients in b_j, we have

$$\binom{2j}{j-k}\Big/\binom{2j}{j} = \exp\{2\ln j! - \ln(j+k)! - \ln(j-k)!\}$$

$$= \exp\left\{2\left(j+\frac{1}{2}\right)\ln j - \left(j+\frac{1}{2}\right)(\ln(j+k)+\ln(j-k))\right.$$

$$\left. -k(\ln(j+k)-\ln(j-k)) + O\left(\frac{1}{j}\right) + O\left(\frac{1}{j+k}\right) + O\left(\frac{1}{j-k}\right)\right\}.$$

Now, the $O(1/(j+k))$ and $O(1/(j-k))$ terms render this approximation useless unless the value of k is restricted in some way. In this case, the appropriate restriction is to take $|k| \leqq j^{1/2+\varepsilon}$ for some small positive constant $\varepsilon > 0$ (the reason for this will become apparent below). With this restriction, we can replace $O(1/(j+k))$ and $O(1/(j-k))$ by $O(1/j)$. Also, we get the asymptotic expansions

$$\ln(j+k) = \ln j + \frac{\kappa}{j} - \frac{k^2}{2j^2} + \frac{k^3}{3j^3} - O(j^{4\varepsilon-2})$$

and

$$\ln(j-k) = \ln j - \frac{k}{j} - \frac{k^2}{2j^2} - \frac{k^3}{3k^3} + O(j^{4\varepsilon-2}) \quad \text{for } |k| \leqq j^{1/2+\varepsilon}.$$

Substituting these and simplifying, we find that several terms cancel, leaving

$$\frac{\binom{2j}{j-k}}{\binom{2j}{j}} = e^{-k^2/j}(1+O(j^{4\varepsilon-1})) \quad \text{for } |k| \leqq j^{1/2+\varepsilon}. \tag{16}$$

This estimate can be used in our expression for b_j because the terms for $|k| \geqq j^{1/2+\varepsilon}$ are negligibly small. We have

$$\binom{2j}{j-k} < \binom{2j}{j-j^{1/2+\varepsilon}} \quad \text{for } |k| > j^{1/2+\varepsilon},$$

so (16) implies that

$$\frac{\binom{2j}{j-k}}{\binom{2j}{j}} = e^{-j^{2\varepsilon}}(1+O(j^{4\varepsilon-1})) \quad \text{for } |k| > j^{1/2+\varepsilon} \tag{17}$$

and this is $O(j^{-m})$ for all $m > 0$. Now, we can split the sum for b_j into two parts and apply (16) and (17) to replace the binomial coefficients with an exponential (recall that $|\nabla f(k)|=1$):

$$
\begin{aligned}
b_j &= \sum_{1\le k\le j^{1/2+\varepsilon}} \frac{\binom{2j}{j-k}}{\binom{2j}{j}} \nabla f(k) + \sum_{j^{1/2+\varepsilon}<k\le j} \frac{\binom{2j}{j-k}}{\binom{2j}{j}} \nabla f(k) \\
&= \sum_{1\le k\le j^{1/2+\varepsilon}} e^{-k^2/j}(1+O(j^{4\varepsilon-1}))\,\nabla f(k) + O(je^{-j^{2\varepsilon}}) \qquad (18)\\
&= \sum_{k\ge 1} e^{-k^2/j}\,\nabla f(k)(1+O(j^{4\varepsilon-1})).
\end{aligned}
$$

The terms for which the estimate (16) is not valid are exponentially small, as is $e^{-k^2/j}$; therefore it doesn't matter which we use in the "tail" of the sum.

If we had a simple expression for $\nabla f(k)$ we could proceed to get an asymptotic expression for b_j by applying the Euler–MacLaurin summation formula to approximate the sum with an integral, then do the integration. For example, we could apply the methods of the previous paragraph to the formula for B_N in Theorem 2 to get the asymptotic formula

$$B_N = \sum_{k\ge 1} e^{-k^2/N}(2F(k)+k)(1+O(N^{4\varepsilon-1})),$$

and from equation (7) it is easy to prove that $F(k)=\frac{1}{2}k\lg k+O(k)$ so that the Euler–MacLaurin summation gives the approximation

$$B_N = \int_1^\infty e^{-x^2/N}(x\lg x+(O(x))(1+O(N^{4\varepsilon-1}))\,dx$$

which, after the substitution $t=x^2/n$, leads to the well-known "exponential integral" function (see [1]), with the result

$$B_N = \frac{1}{4}N\lg N+O(N).$$

This method cannot be extended to find the coefficient of N, since the precise equation for $F(k)$ is quite complicated and depends on the binary value of k. Similarly, a simple equation for $\nabla f(k)$ is not available, and we need to resort to more advanced techniques to get an accurate estimate for b_j (and, eventually, B_N).

The "gamma-function" method that we shall use to evaluate b_j makes use of the residue theorem from the theory of functions of a complex variable. Knopp [10], [11] is the classical text on the theory of functions, and is an excellent introduction to the subject of complex analysis. Other aspects of complex analysis and properties of the two special functions that we use, the gamma (Γ) function and Riemann's zeta (ζ) function, may be found in Whittaker and Watson [15]. And we shall make use of a number of identities from Abramowitz and Stegun [1] and some other references noted below. The idea is to express $e^{-k^2/j}$ as an integral in the complex plane involving the Γ-function, then interchange the order of integration and summation. Although we don't have a simple closed formula for $\nabla f(k)$, we will be able to express the resulting complex series involving $\nabla f(k)$ in terms of classical analytic functions. This is the key to the analysis, for then the integral can be evaluated by finding residues within an appropriate contour of integration.

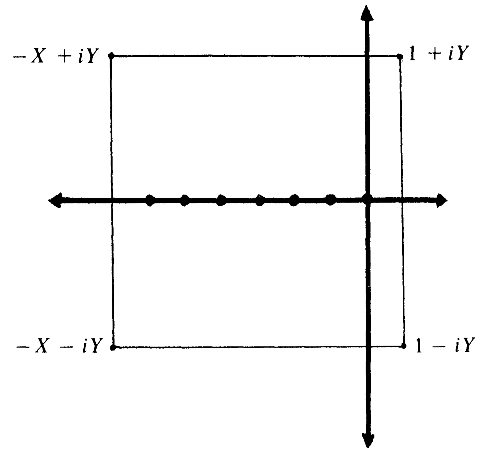


FIG. 13. *Contour of integration for Γ-function identity.*

We begin with the identity

$$e^{-r}=\frac{1}{2\pi i}\int_{1-i\infty}^{1+i\infty}\Gamma(z)r^{-z}\,dz. \tag{19}$$

This is the so-called Mellin transform of e^{-r} [7], a special case of Fourier inversion. We may prove this also directly from the residue theorem using the contour of integration R_{XY} shown in Fig. 13, and letting X and $Y\to\infty$. The function $\Gamma(z)r^{-z}$ has simple poles at $z=-k$, $k=0,1,2,\cdots$ with residue $r^k(-1)^k/k!$, so the value of the integral along R_{XY} is $\sum_{0\leqq k<X}(-r)^k/k!$ which becomes e^{-r} as $X\to\infty$. The integral in (19) is the integral along the right boundary of R_{XY}; the integrals along the other boundaries vanish as X, $Y\to\infty$ because the Γ-function becomes exponentially small on them. (We shall skip the precise bounds here because they may be found in Knuth [13, p. 132] and we shall be doing similar calculations later.) Applying this identity to our formula (18) for b_j, we have

$$\begin{aligned}b_j&=\sum_{k\geqq 1}\nabla f(k)\frac{1}{2\pi i}\int_{1-i\infty}^{1+i\infty}\Gamma(z)\left(\frac{k^2}{j}\right)^{-z}dz\,(1+O(j^{4\varepsilon-1}))\\&=\frac{1}{2\pi i}\int_{1-i\infty}^{1+i\infty}\Gamma(z)j^z\sum_{k\geqq 1}\frac{\nabla f(k)}{k^{2z}}\,dz\,(1+O(j^{4\varepsilon-1})).\end{aligned}$$

(The reader may wish to check that the interchange of summation and integration is justified here because of absolute convergence.)

In order to proceed further we need to know the properties of the function $\sum_{k\geqq 1}\nabla f(k)/k^z$. Remarkably, this function can be expressed in terms of the generalized Riemann (Hurwitz) ζ-function. Figure 14 shows the values of $\nabla f(k)$ broken up in a way that displays the pattern: the values for odd k go in the sequence $1,-1,1,-1,\cdots$; if those are removed, the odd values in the remaining sequence are $1,-1,1,-1,\cdots$; if those are removed, the odd values in the remaining sequence are $1,-1,1,-1,\cdots$; etc. (Proof: For $m>0$, the numbers $m\cdot 2^{n+2}+2^n$ and $m\cdot 2^{n+2}+2^n-1$ differ only in their last $(n+1)$ bits, so from the interpretation that $f(k)$ is the number of parity changes in the binary representation of k, we must have $\nabla f(m\cdot 2^{n+2}+2^n)=\nabla f(2^n)=1$ for all $m,n\geqq 0$. (See discussion following (5).) A similar argument shows that $\nabla f(m\cdot 2^{n+2}+3\cdot 2^n)=\nabla f(3\cdot 2^n)=-1$ for all $m,n\geqq 0$.) In terms of complex

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
∇f(k)	1	1	-1	1	1	-1	-1	1	1	1	-1	-1	1	-1	-1	1	-1	1	-1	1	1	-1	-1	-1	1	1	-1	-1	1	-1	-1
(k=2j+1)	1		-1		1		-1		1		-1		1		-1		1		-1		1		-1		1		-1		1		-1
(k=4j+2)		1				-1				1				-1				1				-1				1				-1	
(k=8j+4)				1								-1								1								-1			
(k=16j+8)								1																-1							
(k=32j+16)																1															

FIG. 14. *Decomposition of* $\nabla f(k)$.

functions this means that

$$\sum_{k\geqq 1}\frac{\nabla f(k)}{k^z}=\left(\frac{1}{1^z}-\frac{1}{3^z}+\frac{1}{5^z}-\frac{1}{7^z}+\cdots\right)\left(\frac{1}{1^z}+\frac{1}{2^z}+\frac{1}{4^z}+\frac{1}{8^z}+\cdots\right).$$

Both of these series can be expressed in terms of classical functions of complex variables. The second is a simple geometric series:

$$\sum_{k\geqq 0}\frac{1}{(2^k)^z}=\sum_{k\geqq 0}\left(\frac{1}{2^z}\right)^k=\frac{1}{1-1/2^z}=\frac{2^z}{2^z-1}.$$

The first factor involves the generalized Riemann ζ-function, which is defined, for $\operatorname{Re}(z)>1$, by the equation

$$\zeta(z,a)=\sum_{n\geqq 0}\frac{1}{(n+a)^z}.$$

Of course, we shall need to deal with the analytic continuation of this function, which is defined for all z except $z=1$, where there is a simple pole with residue 1. (The classical reference for properties of the ζ-function is Titchmarsh [14], though Whittaker and Watson [15] also have a full treatment, and Edwards [6] gives a nice historical perspective.) In terms of this function, we have

$$\sum_{k\geqq 0}\frac{(-1)^k}{(2k+1)^z}=2\sum_{k\geqq 0}\frac{1}{(4k+1)^z}-\sum_{k\geqq 1}\frac{1}{k^z}+\sum_{k\geqq 1}\frac{1}{(2k)^z}$$
$$=\frac{2}{4^z}\zeta\left(z,\frac{1}{4}\right)-\frac{2^z-1}{2^z}\zeta(z,1).$$

(It is customary to drop the second argument in $\zeta(z,1)$ and refer to it simply as $\zeta(z)$; this is the function originally studied by Riemann.) Therefore, we have found that

$$\sum_{k\geqq 1}\frac{\nabla f(k)}{k^z}=\frac{2\zeta(z,1/4)}{2^z(2^z-1)}-\zeta(z). \tag{20}$$

It is the existence of this simple formula which makes the gamma-function method applicable to this problem. (Functions of this form are well-known in analytic number theory as Dirichlet series, and many techniques have been developed for dealing with them. See, for example, [2].)

Substituting, we have

$$b_j=\frac{1}{2\pi i}\int_{1-i\infty}^{1+i\infty}\Gamma(z)j^z\left(\frac{2\zeta(2z,1/4)}{4^z(4^z-1)}-\zeta(2z)\right)dz\,(1+O(j^{4\varepsilon-1})).$$

To evaluate this integral, we first approximate it by integrating around the contour R_Y shown in Fig. 15 and letting $Y\to\infty$. As before, as $Y\to\infty$ the integral along the right-hand side of R_Y approaches the given integral, and the integrals along the top, bottom and left can be bounded by using well-known bounds on the Γ and ζ functions. We have

$$|\Gamma(x+iy)|=O(|y|^{x-1/2}\,e^{-\pi|y|/2}), \tag{21}$$

which follows from Stirling's approximation (see, for example, [1, eq. 6.1.45]), and

$$|\zeta(x+iy,a)|=O(|y|^{1-x})\quad\text{for } x\geqq -1 \tag{22}$$

(see, for example, Whittaker and Watson [15, p. 276]). Therefore, the integrals along

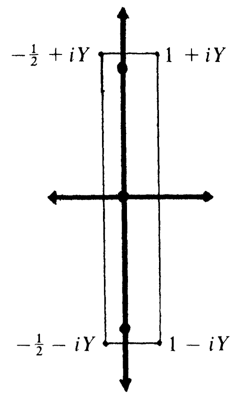


FIG. 15. *Contour of integral for final integral.*

the top and bottom lines of R_Y are

$$O\left(e^{-\pi|Y|/2}\int_{-1/2}^{1} j^x|Y|^{3/2-x}\,dx\right)$$

which is exponentially small and vanishes very quickly as $Y\to\infty$. The integral along the left line of R_Y is

$$O\left(j^{-1/2}\int_{-\infty}^{\infty} e^{-\pi|y|/2}|y|^2\,dy\right)=O(j^{-1/2})$$

so that we now have

$$(23)\qquad b_j=\frac{1}{2\pi i}\int_{R_Y}\Gamma(z)j^z\left(\frac{2\zeta(2z,1/4)}{4^z(4^z-1)}-\zeta(2z)\right)dz\,(1+O(j^{4\varepsilon-1}))+O(j^{-1/2}).$$

The value of the integral is the sum of the residues within R_Y.

The only singularities within R_Y are contributed by $\Gamma(z)$ and $1/(4^z-1)$: the function $\Gamma(z)$ has a simple pole at $z=0$ with residue 1, and $1/(4^z-1)$ has simple poles at $z=2k\pi i/\ln 4$ for $k=0,\pm1,\pm2,\cdots$ with residue $1/\ln 4$. (Both $\zeta(2z,\frac{1}{4})$ and $\zeta(2z)$ have simple poles with residue 1 at $z=1/2$, but they cancel out.) There is therefore a double pole at $z=0$ and we need to use Laurent series expansions to find the residue there. We have

$$\Gamma(z)=\frac{\Gamma(z+1)}{z}=\frac{1}{z}\exp\{-\gamma z+O(z^2)\}=\frac{1}{z}-\gamma+O(z),$$

$$j^z=e^{z\ln j}=1+z\ln j+O(z^2),$$

$$\frac{1}{4^z}=e^{z\ln(1/4)}=1-z\ln 4+O(z^2),$$

$$\frac{1}{4^z-1}=\frac{1}{e^{z\ln 4}-1}=\frac{1}{z\ln 4+(z\ln 4)^2/2+O(z^3)}=\frac{1}{z\ln 4}-\frac{1}{2}+O(z),$$

$$\zeta\left(2z,\frac{1}{4}\right)=\frac{1}{4}+z\left(2\ln\Gamma\left(\frac{1}{4}\right)-\ln(2\pi)\right)+O(z^2),$$

and

$$\zeta(2z)=-\frac{1}{2}+O(z).$$

The expansion for $\Gamma(z)$ is well known (see Abramowitz and Stegun [1, eq. 6.1.33]), and the next three expansions are elementary. The expansions for the ζ-functions, which are crucial to the derivation, follow directly from Whittaker and Watson [15, p. 271] where it is shown that $\zeta(0, a)=\frac{1}{2}-a$ and $\zeta'(0, a)=\ln\Gamma(a)-\frac{1}{2}\ln(2\pi)$. Multiplying these series together, we find that

$$\Gamma(z)j^z\Big(\frac{2\zeta(2z, 1/4)}{4^z(4^z-1)}-\zeta(2z)\Big)=\Big(\frac{1}{z}+\ln j-\gamma+O(z)\Big)\Big(\frac{1}{4z\ln 2}-\frac{1}{4}+\lg\frac{\Gamma(1/4)^2}{2\pi}+O(z)\Big)$$

(24)

$$=\frac{1}{4z^2\ln 2}+\frac{1}{z}\Big(\frac{1}{4}\lg j-\frac{\gamma}{4\ln 2}-\frac{1}{4}+\lg\frac{\Gamma(1/4)^2}{2\pi}\Big)+O(1).$$

This gives the residue at $z=0$ (the coefficient of $1/z$).

To this we must add the residue at the other poles of $1/(4^z-1)$. The effect of these other terms is small (but not insignificant), and we shall encapsulate them in a single term,

$$\begin{aligned}\varepsilon(j)&\equiv\frac{2}{\ln 4}\sum_{k\neq 0}\Gamma\Big(\frac{2k\pi i}{\ln 4}\Big)j^{2k\pi i/\ln 4}\zeta\Big(\frac{4k\pi i}{\ln 4}, \frac{1}{4}\Big)\\&=\frac{1}{\ln 2}\sum_{k\geqq 1}2\operatorname{Re}\Big(\Gamma\Big(\frac{k\pi i}{\ln 2}\Big)j^{k\pi i/\ln 2}\zeta\Big(\frac{2k\pi i}{\ln 2}, \frac{1}{4}\Big)\Big)\\&=\sum_{k\geqq 1}(\xi_k\cos(k\pi\lg j)-\eta_k\sin(k\pi\lg j)),\end{aligned}$$

where

$$\frac{2}{\ln 2}\Gamma\Big(\frac{k\pi i}{\ln 2}\Big)\zeta\Big(\frac{2k\pi i}{\ln 2}, \frac{1}{4}\Big)\equiv\xi_k+i\eta_k.$$

To finish the evaluation of our b_j and B_N we need to estimate the Γ and ζ functions at these points along the imaginary axis. The Γ-function is easy to bound from Stirling's approximation (see Edwards [5, § 6.3]), and the ζ-function can be estimated by writing

$$\zeta(z, a)=\sum_{0\leqq k<K}\frac{1}{(k+a)^z}+\sum_{k\geqq K}\frac{1}{(k+a)^z},$$

and then applying Euler–MacLaurin summation to the second sum, for appropriate K. (These manipulations are valid for $\operatorname{Re} z>1$ only, but the resulting formulas are valid for all z, by analytic continuation—see Edwards [6, § 6.4] for details.) Table 4 shows the values of ξ_k and η_k for $k=1, 2, 3$ computed in this way. The values get exceedingly small for larger k, as can be verified from the bounds (21) and (22).

Adding all the residues, we have, from (23):

$$b_j=\frac{1}{4}\lg j+\lg\frac{\Gamma(1/4)^2}{2\pi}-\frac{1}{4}-\frac{\gamma}{4\ln 2}+\varepsilon(j)+O(j^{-1/2}).\tag{25}$$

This leads to our final result.

THEOREM 3. *The average number of exchanges used by Batcher's odd-even merge for a random 2-ordered file of 2N elements is*

$$B_N=\frac{1}{4}N\lg N+\Big(\lg\frac{\Gamma(1/4)^2}{2\pi}+\frac{1}{4}-\frac{\gamma+2}{4\ln 2}+\delta(N)\Big)N+O(\sqrt{N}\log N),$$

where $\delta(N)$ is a periodic function of $\log N$, *with $\delta(4N)=\delta(N)$, $|\delta(N)|<.000490$, and*

$\delta(2^n) \approx .000317(-1)^n$. (*The constant*

$$\lg \frac{\Gamma(1/4)^2}{2\pi} + \frac{1}{4} - \frac{\gamma+2}{4\ln 2}$$

has the approximate value .385417224.)

Proof. From the discussion above, we need only substitute our result (25) for b_j into our equation (14) for B_N and perform the summation. We have

$$B_N = \frac{1}{2} \frac{4^N}{\binom{2N}{N}} \sum_{1 \leqq j < N} \frac{\binom{2j}{j}}{4^j} \left(\frac{1}{4} \lg j + \lg \frac{\Gamma(1/4)^2}{2\pi} - \frac{1}{4} - \frac{\gamma}{4 \ln 2} + \varepsilon(j) + O(j^{-1/2}) \right) + \frac{1}{2} N.$$

The terms not involving j are easily taken care of, since it is trivial to prove by induction that

$$\frac{1}{2} \frac{4^N}{\binom{2N}{N}} \sum_{0 \leqq j < N} \frac{\binom{2j}{j}}{4^j} = N.$$

(Direct proof:

$$\sum_{0 \leqq j < N} \frac{1}{4^j} \binom{2j}{j} = \sum_{0 \leqq j < N} (-1)^j \binom{-1/2}{j} = (-1)^{N-1} \binom{-3/2}{N-1} = \binom{N-1/2}{N-1} = \frac{N}{4^{N-1}} \binom{2N-1}{N-1};$$

for supporting identities, see Knuth [12].)

For the other terms, we can remove the binomial coefficients with Stirling's approximation, as in the derivation of (16). We have

$$\frac{\binom{2j}{j}}{4^j} = \frac{1}{\sqrt{\pi j}} + O(j^{-3/2}) \quad \text{and} \quad \frac{4^N}{\binom{2N}{N}} = \sqrt{\pi N} + O(N^{-1/2}).$$

TABLE 4

Values of constants in the asymptotic expansion for the average number of exchanges.

$$\xi_k + i\eta_k = \frac{2}{\ln 2} \Gamma\left(\frac{k\pi i}{\ln 2}\right) \zeta\left(\frac{2k\pi i}{\ln 2} \frac{1}{4}\right)$$

k	ξ_k	η_k
1	.003704670+	.002500177+
2	.000001560+	−.000000832−
3	.000000001−	.000000002+

$$\Gamma\left(\frac{1}{4}\right) = 3.6256099082+$$

$$\frac{1}{\ln 2} = 1.4426950408+$$

$$\gamma = 0.5772156649+$$

$$\pi = 3.1415926535+$$

Therefore the $O(j^{-1/2})$ term sums to $O(\sqrt{N}\log N)$, and

$$\frac{1}{2}\frac{4^N}{\binom{2N}{N}}\sum_{1\leq j<N}\frac{\binom{2j}{j}}{4^j}\frac{1}{4}\lg j=\frac{\sqrt{N}}{8}\sum_{1\leq j<N}\frac{\lg j}{\sqrt{j}}+O(\sqrt{N})$$

$$=\frac{\sqrt{N}}{8}\int_1^N\frac{\lg x}{\sqrt{x}}\,dz+O(\sqrt{N})$$

$$=\frac{\sqrt{N}}{2\ln 2}\int_1^{\sqrt{N}}\ln y\,dy+O(\sqrt{N})$$

$$=\frac{1}{4}N\lg N-\frac{1}{2\ln 2}N+O(\sqrt{N}).$$

Here the second step follows from Euler–MacLaurin summation (see, for example, Knuth [13, p. 110]) and the third step from the substitution $x=y^2$.

We have proved that

$$B_N=\frac{1}{4}N\lg N+\left(\lg\frac{\Gamma(1/4)^2}{2\pi}+\frac{1}{4}-\frac{\gamma+2}{4\ln 2}+\delta(N)\right)N+O(\sqrt{N}\log N);$$

it remains to evaluate the oscillatory term

$$N\delta(N)=\frac{1}{2}\frac{4^N}{\binom{2N}{N}}\sum_{1\leq j<N}\frac{\binom{2j}{j}}{4^j}\varepsilon(j).$$

After substituting for $\varepsilon(j)$, we proceed in the same way as we did for the $\lg j$ term. The result of using Stirling's approximation on the binomial coefficients and Euler–MacLaurin summation on the resulting sums is

$$\delta(N)=\frac{1}{2\sqrt{N}}\sum_{k\geq 1}\left(\xi_k\int_1^N\frac{\cos(k\pi\lg x)}{\sqrt{x}}\,dx-\eta_k\int_1^N\frac{\sin(k\pi\lg x)}{\sqrt{x}}\,dx\right)+O\left(\frac{1}{\sqrt{N}}\right).$$

These integrals are elementary; the substitutions $x=y^2$, then $t=2\pi k\lg y$, transform them into standard integrals (for example, Abramowitz and Stegun [1, eqs. 4.3.136, 4.3.137]) with the eventual result

$$\delta(N)=\sum_{k\geq 1}\frac{\sigma_k}{\sigma_k^2+1}(\xi_k(\sigma_k\cos(\pi k\lg N)+\sin(\pi k\lg N))$$
$$-\eta_k(\sigma_k\sin(\pi k\lg N)-\cos(\pi k\lg N)))$$

where σ_k is $(\ln 2)/(2\pi k)$. From this formula, we see that $\delta(N)$ has the stated properties. With the aid of Table 4, we can easily compute the values

$$\delta(2^{2n})=\sum_{k\geq 1}\frac{\sigma_k}{\sigma_k^2+1}(\sigma_k\xi_k+\eta_k)\approx .000317000\cdots \tag{26}$$

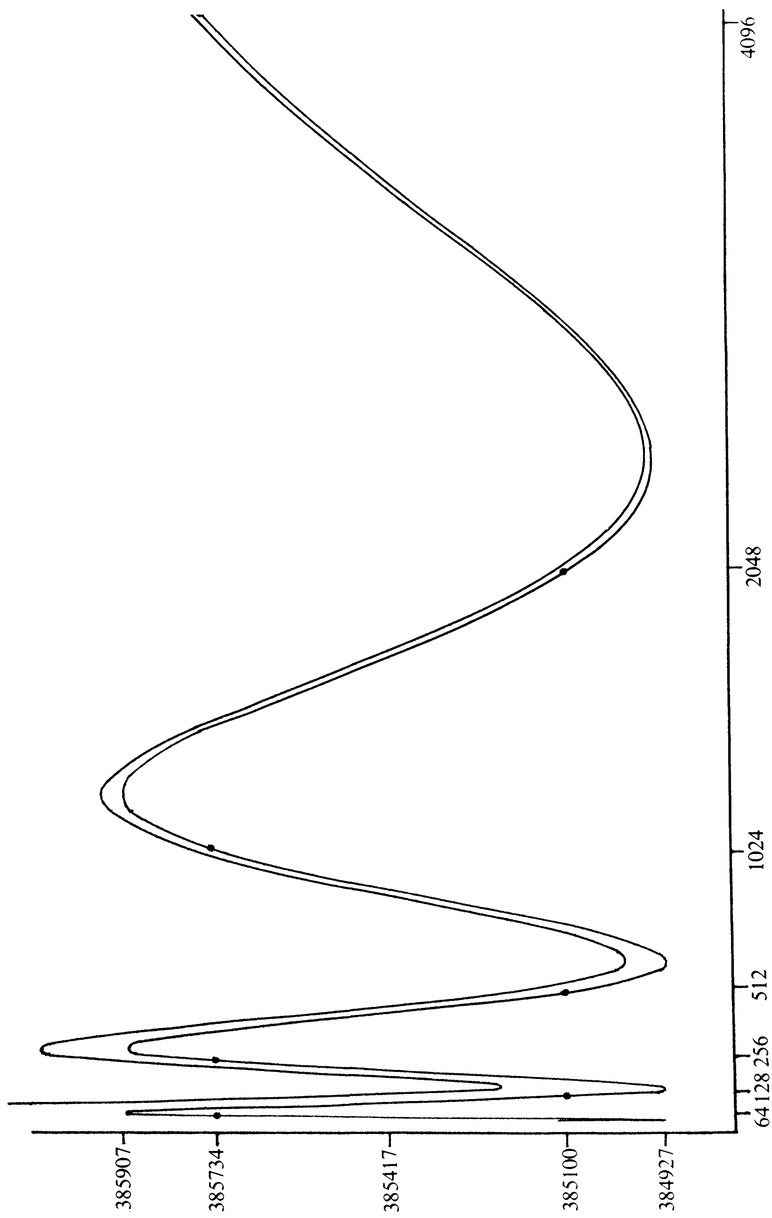


FIG. 16. *Coefficient of the linear term.*

and

$$\delta(2^{2n+1}) = \sum_{k \geqq 1} (-1)^k \frac{\sigma_k}{\sigma_k^2+1}(\sigma_k\xi_k + \eta_k) \approx -.000317082\cdots. \tag{27}$$

Finally, since the function $a \sin x + b \cos x$ has the extreme values $\pm\sqrt{a^2+b^2}$, we have the extreme values

$$|\delta(N)| \leqq \sum_{k \geqq 1} \frac{\sigma_k}{\sigma_k^2+1}\sqrt{(\xi_k\sigma_k+\eta_k)^2+(\xi_k-\eta_k\sigma_k)^2} \approx .000490177. \quad \square$$

From Theorem 3 we see that, asymptotically, only 1/4 of the comparators in Batcher's merge are involved in exchanges, on the average. The analytic result for the coefficient of the linear term given in Theorem 3 matches the exact computed value (Table 3) to six decimal places.

In principle we could extend the methods used to get any desired accuracy whatsoever. This would mainly involve carrying the asymptotic series expansions further, which gets very complicated in the applications of Euler–MacLaurin summation. Also, the left boundary has to be moved left for sharper asymptotic accuracy in (23). Each negative integer enclosed contributes another simple pole from the Γ function.

Figure 16 shows a graph of the coefficient of the linear term from Theorem 3,

$$\lg \frac{\Gamma(1/4)^2}{2\pi} + \frac{1}{4} - \frac{\gamma+2}{4 \ln 2} + \delta(N),$$

together with the true values of $(B_N - \frac{1}{4}N \lg N)/N$, computed with (13). The upper curve is the actual values, and the lower curve is the asymptotic estimate. The difference between the curves is reflected in the $O(\sqrt{N}\log N)$ term in Theorem 3. The curves get very close for large N.

4. Sorting. Any merging method may be extended into a sorting method with the following recursive procedure: To sort a file of N elements, use the method to independently sort the odd elements and the even elements of the file, thus producing a 2-ordered file of N elements. Then apply the merging method. Figure 17 shows the sorting network resulting from applying this procedure to Batcher's odd-even merge. If merge stages are overlapped, the sort can be accomplished in $\frac{1}{2}\lceil \lg N \rceil$ $(\lceil \lg N \rceil + 1)$ independent stages. Knuth gives a formula describing the number of comparators required [13, exercise 5.2.2-15]; it depends heavily on the binary representation of N. For simplicity, we shall assume throughout this section that $N = 2^n$. The number of comparators required is then described by the relation (see (1))

$$C^*_{2^n} = 2C^*_{2^{n-1}} + (n-1)2^{n-1} + 1$$

which telescopes, after division by 2^n, to the solution

$$C^*_N = \frac{1}{4}N(\lg N)^2 - \frac{1}{4}N \lg N + N - 1, \qquad N = 2^n. \tag{28}$$

Again, this method cannot compete with known $O(N \log N)$ sorting methods on serial computers, but it might do well if parallelism is available.

The average number of exchanges required can be calculated from a similar recurrence, using Theorem 3, since the odd and even elements are sorted independently. If

$$\alpha \equiv \lg \frac{\Gamma(1/4)^2}{2\pi} + \frac{1}{4} - \frac{\gamma+2}{4\ln 2}$$

we have the following expression for the average number of exchanges:

$$B^*_{2^n} = 2B^*_{2^{n-1}} + \left(\frac{1}{4}(n-1) + \alpha + \delta(2^{n-1})\right)2^{n-1} + O(\sqrt{N}\log N).$$

Iterating this recurrence (applying the same recurrence to $B^*_{N/2}$), we get

$$B^*_{2^n} = 4B^*_{2^{n-2}} + \left(\frac{1}{2}n + 2\alpha - \frac{3}{4} + \delta(2^{n-2}) + \delta(2^{n-1})\right)2^{n-1} + O(\sqrt{N}\log N).$$

If we define $\delta^*(2^n) \equiv \delta(2^{n-2}) + \delta(2^{n-1})$, then we know that $\delta^*(2^n) = \delta^*(2^{n-2})$ as in Theorem 3. Our recurrence then telescopes when divided by 2^n to the solution

$$\begin{aligned}\frac{B^*_{2^n}}{2^n} &= \sum_{0 \leqq j \leqq n/2} \left(\frac{1}{4}(n-2j) + \alpha - \frac{3}{8} + \frac{1}{2}\delta^*(2^n)\right) + O(1)\\ &= \frac{n^2}{16} + \frac{1}{2}\left(\alpha - \frac{1}{8} + \delta^*(2^n)\right)n + O(1)\end{aligned}$$

or, in terms of N:

$$B^*_N = \frac{1}{16}N(\lg N)^2 + \frac{1}{2}\left(\alpha - \frac{1}{8} + \delta^*(N)\right)N\lg N + O(N), \qquad N = 2^n. \tag{29}$$

The value of $\frac{1}{2}(\alpha - \frac{1}{8})$ is about .130208 $\cdots$ which is the value of the coefficient of the $N \lg N$ term to six places, since we know from (26) and (27) that $|\delta^*(N)| < 10^{-6}$.

In the same way, we could find from Theorem 1 that, asymptotically, all of the comparators could be involved in exchanges in the worst case. However, this asymptotic maximum is approached even more slowly than for the merging method, since the recursive nature of the sorting method guarantees that many small files will be merged.

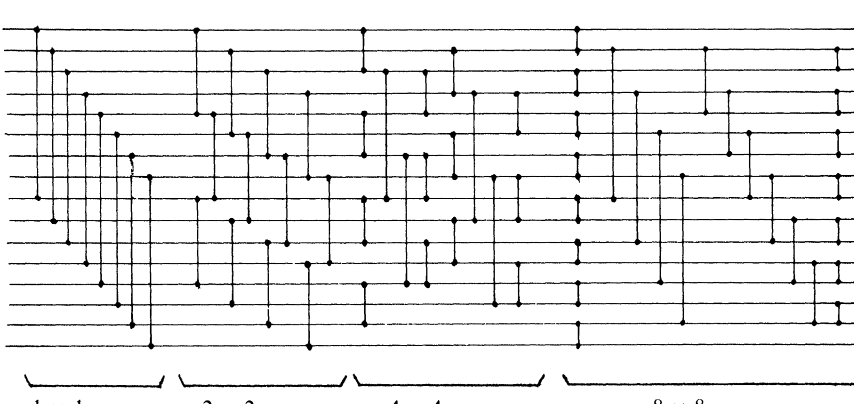


FIG. 17. *Odd-even sorting network.*

5. Conclusion. In this paper, we have derived formulas which accurately describe the number of exchanges involved in Batcher's odd-even merge, both on the average and in the worst case. This completes our understanding of Batcher's method, which is of some theoretical importance as a near-optimal nonadaptive method, and of some practical importance when parallelism is available.

The main results are the exact formulas for the worst case and the average given in Theorems 1 and 3. These lead to asymptotic statements that, as $N \to \infty$, about $\frac{1}{4}$ of the comparators do exchanges on the average and nearly all of them do exchanges in the worst case.

We have emphasized the methods of analysis, as well as the results, because they may have more general applicability. In particular, Theorem 2 could be of use in the analysis of other merging problems and other combinatorial problems which can be modeled with paths in a lattice. Also, the problem of determining the average number of exchanges has provided an excellent example of the application of de Bruijn's "gamma-function" method of asymptotic analysis.

Acknowledgments. I had thought this problem hopelessly difficult until Dave Notkin brought its details to my attention in a classroom project.

Note. The kind of asymptotic analysis that we used in determining the average number of exchanges has recently been used to solve yet another problem: determining the average number of registers needed to evaluate arithmetic expressions. See the recent reports by P. Flajolet, J. C. Raoult and J. Vuillemin, *On the average number of registers required for evaluating arithmetic expressions*, Proc. 18th Symp. on Foundations of Computer Science, Providence, RI; and by R. Kemp, *The average number of registers needed to evaluate a binary tree optimally*, Saarbrücken University Report A 77104, Saarbrücken, Germany.

REFERENCES

[1] M. ABRAMOWITZ AND I. A. STEGUN, *Handbook of Mathematical Functions*, Dover, New York, 1970.

[2] T. M. APOSTOL, *Introduction to Analytic Number Theory*, Springer-Verlag, New York, 1976.

[3] K. E. BATCHER, *A new internal sorting method*, Rep. GER 11759, Goodyear Aerospace Corp., Akron, OH, 1964.

[4] ———, *Sorting networks and their applications*, Proc. 1968 Spring Joint Comp. Conf., AFIPS Press, Montvale, NJ, 1968, pp. 307–314.

[5] N. G. DE BRUIJN, D. E. KNUTH AND S. O. RICE, *The average height of planted plane trees*, Graph Theory and Combinatorics, R. C. Read, ed., Academic Press, New York, 1972, pp. 15–22.

[6] H. M. EDWARDS, *Riemann's Zeta Function*, Academic Press, New York, 1974.

[7] A. ERDÉLYI, ed., *Tables of Integral Transforms*, McGraw-Hill, New York, 1954.

[8] H. W. GOULD AND J. KAUCKÝ, *Evaluation of a class of binomial coefficient summations*, J. Combinatorial Theory, 1 (1966), pp. 233–247.

[9] J. L. w. V. JENSEN, *Sur une identité d'Abel et sur d'autres formules analogues*, Acta Math., 26 (1902), pp. 307–318.

[10] K. KNOPP, *Theory of Functions Part I*, Dover, New York, 1945.

[11] ———, *Theory of Functions Part II*, Dover, New York, 1945.

[12] D. E. KNUTH, *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*, Addison-Wesley, Reading, MA, 1968.

[13] ———, *The Art of Computer Programming, Vol. 3: Sorting and Searching*, Addison-Wesley, Reading, MA, 1972.

[14] E. C. TITCHMARSH, *The Theory of the Riemann Zeta-Function*, Clarendon Press, Oxford, 1951.

[15] E. T. WHITTAKER AND G. N. WATSON, *A Course of Modern Analysis*, Cambridge University Press, London, 1973.