ALGORITHM 64
QUICKSORT

C. A. R. Hoare
Elliott Brothers Ltd., Borehamwood, Hertfordshire, Eng.

```
procedure  quicksort (A,M,N);  value M,N;
              array A;  integer M,N;
comment  Quicksort is a very fast and convenient method of
sorting an array in the random-access store of a computer. The
entire contents of the store may be sorted, since no extra space is
required. The average number of comparisons made is 2(M-N) ln
(N-M), and the average number of exchanges is one sixth this
amount. Suitable refinements of this method will be desirable for
its implementation on any actual computer;
begin      integer I,J;
           if M < N then begin partition (A,M,N,I,J);
                              quicksort (A,M,J);
                              quicksort (A, I, N)
                         end
end        quicksort
```

CERTIFICATION OF ALGORITHMS 63, 64, 65 PARTITION, QUICKSORT, FIND [C. A. R. Hoare, *Comm. ACM*, July 1961]

J. S. Hillmore
Elliott Bros. (London) Ltd., Borehamwood, Herts., England

The body of the procedure find was corrected to read:

```
begin integer I, J;
if M < N then begin partition (A, M, N, I, J);
                   if K ≦ I then find (A, M, J, K)
                   else if J ≦ K then find (A, I, N, K)
                   end
end find
```

and the trio of procedures was then successfully run using the Elliott Algol translator on the National-Elliott 803.

The author's estimate of $\frac{1}{3}(N-M)\ln(N-M)$ for the number of exchanges required to sort a random set was found to be correct. However, the number of comparisons was generally less than $2(N-M)\ln(N-M)$ even without the modification mentioned below.

The efficiency of the procedure quicksort was increased by changing its body to read:

```
begin integer I, J;
if M < N-1 then begin partition (A, M, N, I, J);
                     quicksort (A, M, J);
                     quicksort (A, I, N)
                     end
else if N-M = 1 then begin if A[N] < A[M] then
                              exchange (A[M], A[N])
                        end
end quicksort
```

This alteration reduced the number of comparisons involved in sorting a set of random numbers by 4–5 percent, and the number of entries to the procedure partition by 25–30 percent.

CERTIFICATION OF ALGORITHMS 63, 64 AND 65, PARTITION, QUICKSORT, AND FIND, [*Comm. ACM*, July 1961]

B. Randell and L. J. Russell
The English Electric Company Ltd., Whetstone, England

Algorithms 63, 64, and 65 have been tested using the Pegasus Algol 60 Compiler developed at the De Havilland Aircraft Company Ltd., Hatfield, England.

No changes were necessary to Algorithms 63 and 64 (Partition and Quicksort) which worked satisfactorily. However, the comment that Quicksort will sort an array without the need for any extra storage space is incorrect, as space is needed for the organization of the sequence of recursive procedure activations, or, if implemented without using recursive procedures, for storing information which records the progress of the partitioning and sorting.

A misprint ('if' for '**if**' on the line starting '**else** if $J \leqq K$ **then** ...') was corrected in Algorithm 65 (Find), but it was found that in certain cases the sequence of recursive activations of *Find* would not terminate successfully. Since *Partition* produces as output two integers J and I such that elements of the array $A[M:N]$ which lie between $A[J]$ and $A[I]$ are in the positions that they will occupy when the sorting of the array is completed, *Find* should cease to make further recursive activations of itself if K fulfills the condition $J < K < I$.

Therefore the conditional statement in the body of *Find* was changed to read

```
if K ≦ J then find (A,M,J,K)
else if I ≦ K then find (A,I,N,K)
```

With this change the procedure worked satisfactorily.