ALGORITHM 23
MATH SORT
WALLACE FEURZEIG
Laboratories for Applied Science, University of Chicago, Chicago, Ill.

```
procedure MATHSORT (INVEC, OUTVEC, TOTEVEC,
    n, k, SETFUNC) ; value n, k ;
  array INVEC, OUTVEC ;
    integer array TOTEVEC ;
    integer procedure SETFUNC ;
  integer n, k ;
begin comment MATHSORT is a fast sorting algorithm which
    produces a monotone rearrangement of an arbitrarily ordered
    set of n numbers (represented by the vector INVEC) by a
    surprising though familiar device. The resultant sorted set is
    represented by the vector OUTVEC. The key field, i.e. the
    ordered set of bits (or bytes) on which the sort is to be done,
    is obtained by some extraction-justification function denoted
    SETFUNC. The key field allows the representation of k pos-
    sible values denoted 0, 1, ... , k-1.
    The procedure determines first of all the exact frequency
    distribution of the set with respect to the key, i.e. the number
    of elements of INVEC with key field value precisely equal to
    j for all j between 0 and k-1. The cumulative frequency dis-
    tribution TOTEVEC [i] ≡ ∑(j=0..i) (Number of elements of
    INVEC with key value = j) is then computed for 0 ≦ i ≦ k-1.
    This induces the direct assignment (storage mapping func-
    tion) of each element of INVEC to a unique cell in OUTVEC.
    This assignment (like the determination of the frequency
    distribution) requires just one inspection of each element of
    INVEC. Thus the algorithm requires only 2n "look and do"
    operations plus k-1 additions (to get the cumulative fre-
    quency distribution).
    The algorithm can be easily and efficiently extended to
    handle alphabetic sorts or multiple key sorts. To sort on
    another key the same algorithm is applied to each new key
    field with the new INVEC designated as the last induced
    ordering (i.e. the current OUTVEC). The algorithm has been
    used extensively at LAS on binary as well as decimal machines
    both for internal memory sorts and (with trivial modification)
    for large tape sorts ;
  for i := 1 step 1 until n do
      TOTEVEC[SETFUNC(INVEC[i])] := TOTEVEC
        [SETFUNC(INVEC[i])] + 1 ;
  for i := 1 step 1 until k-1 do
      TOTEVEC[i] := TOTEVEC[i] + TOTEVEC[i-1] ;
  for i := 1 step 1 until n do
      begin OUTVEC[TOTEVEC[SETFUNC(INVEC[i])]]
          := INVEC[i] ;
        TOTEVEC[SETFUNC(INVEC[i])] :=
          TOTEVEC[SETFUNC(INVEC[i])] - 1 ;
          end
end MATHSORT.
```

CERTIFICATION OF ALGORITHM 23
MATHSORT (Wallace Feurzeig, *Comm. ACM*, Nov., 1960)
RUSSELL W. RANSHAW
University of Pittsburgh, Pittsburgh, Pa.

The MATHSORT procedure as published was coded for the IBM 7070 in FORTRAN. Two deficiencies were discovered:

1. The TOTVEC array was not zeroed within the procedure. This led to some difficulties in repeated use of the procedure.

2. Input vectors already in sort on nonsort fields were unsorted. That is, given the sequence

31, 21, 32, 22, 33,

Mathsort would produce, for a sort on the 10's digit:

22, 21, 33, 32, 31,

which is definitely out of sequence.

The following modified form of the procedure corrects these difficulties. Note the transformation of symbols.

```
procedure MATHSORT (I, O, T, n, k, S); value n, k;
    array I, O; integer array T; integer procedure S;
    integer n, k;
begin for i := 0 step 1 until k - 1 do T[i] := 0;
    for i := 1 step 1 until n do T[S(I[i])] := T[S(I[i])] + 1;
    for i := k - 2 step -1 until 0 do T[i] := T[i] +
    T[i + 1];
    for i := 1 step 1 until n do
        begin O[n + 1 - T[S(I[i])]] := I[i];
            T[S(I[i])] := T[S(I[i])] - 1;
        end
end MATHSORT.
```

Using the MATHSORT procedure ten times and having the procedure S supply each digit in order, 1000 random numbers of 10 digits each were sorted into sequence in 31 seconds. The method of locating the lowest element, interchanging with the first element, and continuing until the entire list has been so examined yielded a complete sort on the same 1000 random numbers in 227 seconds. Using the Table-Lookup-Lowest command in the 7070 yielded 56 seconds for the same set of random numbers.