ALGORITHM 87  
PERMUTATION GENERATOR  
John R. Howell  
Orlando Aerospace Division, Martin Marietta Corp., Orlando, Florida

procedure PERMUTATION (N, K);  
value K, N; integer K; integer array N;  
comment This procedure generates the next permutation in  
lexicographic order from a given permutation of the K marks  
0, 1, ··· , (K-1) by the repeated addition of (K-1) radix K.  
The radix K arithmetic is simulated by the addition of 9 radix  
10 and a test to determine if the sum consists of only the original  
K digits. Before each entry into the procedure the K marks  
are assumed to have been previously specified either by input  
data or as the result of a previous entry. Upon each such entry a  
new permutation is stored in N[i] through N[K]. In case the  
given permutation is (K-1), (K-2), ··· , 1, 0, then the next  
permutation is taken to be 0, 1, ··· , (K-1). A FORTRAN  
subroutine for the IBM 7090 has been written and tested for  
several examples;  
begin integer i, j, carry;  
for i := 1 step 1 until K do  
if N[i] = K + i ≠ 0 then go to add;  
for i := 1 step 1 until K do N[i] := i – 1;  
go to exit;  
add:  
for i := 1 step 1 until K-1 do  
begin if K > 10 then go to B;  
carry := (N[K-i+1]+10); go to C;  
B: carry := (N[K-i+1]+10);  
C: if carry = 0 then go to test;  
end i;  
if N[i] – (K – i) > 0 then go to add;  
exit:  
end PERMUTATION GENERATOR

CERTIFICATION OF ALGORITHM 87  
PERMUTATION GENERATOR [John R. Howell,  
Comm. ACM (Apr. 1962)]  
D. M. Collison  
P. R. Eaves, Comm. ACM 53 (Nov. 1962), 551  
ALGORITHM 102 [G6]  
PERMUTATION IN LEXICOGRAPHICAL ORDER  
[G. F. Schrack and M. Shimrat, Comm. ACM 5 (June  
1962), 346]  
ALGORITHM 130 [G6]  
PERMUTE  
R. J. Ord-Smith (Reed. 11 Nov. 1966, 28 Dec. 1966 and  
17 Mar. 1967)  
Computing Laboratory, University of Bradford, England

A comparison of the published algorithms which seek to generate  
successive permutations in lexicographic order shows that Alg.  
102 is the most efficient. Since, however, it is more than twice  
as slow as transposition Algorithm 115 [H. F. Trotter, Perm,  
Comm. ACM 5 (Aug. 1962), 434], there appears to be room for  
improvement. Theoretically a "best" lexicographic algorithm  
should be about one and a half times slower than Algorithm 115.  
See Algorithm 398 [R. J. Ord-Smith, Generation of Permutations  
in Pseudo-Lexicographic Order, Comm. ACM 10 (July 1967), 452]  
which is twice as fast as Algorithm 202.
ALGORITHM 87 is very slow.

ALGORITHM 102 shows a marked improvement.

ALGORITHM 130 does not appear to have been certified before. We find that, certainly for some forms of vector to be permuted, the algorithm can fail. The reason is as follows.

At execution of $A[f] := r$; on line prior to that labeled `check`, $f$ has not necessarily been assigned a value. $f$ has a value if, and only if, the Boolean expression $B[k] > 0 \land B[k] < B[m]$ is true for at least one of the relevant values of $k$. In particular when matrix $A$ is set up by $A[i] := i$; for each $i$ the Boolean expression above is false on the first call.

ALGORITHM 202 is the best and fastest algorithm of the exicographic set so far published.

A collected comparison of these algorithms is given in Table I. $t_n$ is the time for complete generation of $n!$ permutations. Times are scaled relative to $t_4$ for Algorithm 202, which is set at 100. Tests were made on an ICT 1905 computer. The actual time $t_4$ for Algorithm 202 on this machine was 100 seconds. $r_n$ has the usual definition $r_n = t_n/(n \cdot t_4)$.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$t_4$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>87</td>
<td>118</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>2.1</td>
<td>15.5</td>
<td>135</td>
<td>1.03</td>
<td>1.08</td>
<td>1.1</td>
</tr>
<tr>
<td>130</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>202</td>
<td>1.7</td>
<td>12.4</td>
<td>100</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>