On a subset of all the permutations of n marks

Y. L. Varol

Computer Science Department, Southern Illinois University, Carbondale, Illinois 62901, USA

A subset of all the permutations of n marks arises naturally in interval exchange transformation problems. Each permutation $p_1p_2 \ldots p_n$ in the set represents a transformation having a dense orbit, and has the properties $p_{i+1} \neq p_i+1$ and $p_1p_2 \ldots p_i$ is not a permutation of $\{1, 2, \ldots, i\}$ for all i, $1 \leqslant i \leqslant n-1$. A recursive formula is given for the cardinality of this subset, and two algorithms for generating in alphabetic order all the permutations in it are presented.

(Received February 1979)

1. Introduction

Enumerating and generating all the permutations of n distinct marks has caught the fancy of many people since campanology became a popular pastime (MacCallum, 1977). Subsets of permutations satisfying certain conditions such as a partial order on the marks to be permuted (Knuth and Szwarcfiter, 1974), or prescribed up-down and inversion sequences (Foulkes, 1976), and many other subsets (Riordan, 1958) have also been of interest. In this paper we consider a particular subset which arises naturally in Interval Exchange Transformation problems (Keane, 1975), where the permutations represent transformations having a dense orbit.

For $n > 1$, let $Z_n = \{1, 2, \ldots, n\}$ be the set of marks, and P_n denote the set of all $n!$ permutations $\pi = p_1p_2 \ldots p_n$ where $p_i \varepsilon Z_n$ for all $1 \leqslant i \leqslant n$. Consider the set $G_n \subset P_n$, where $\pi \varepsilon G_n$ if and only if

$$p_{i+1} \neq p_i + 1 \text{ for all } 1 \leqslant i \leqslant n-1, \text{ and} \quad (1)$$

$$\pi_i = p_1p_2 \ldots p_i \text{ is not a permutation of } \{1, 2, \ldots, i\} \quad (2)$$

i.e. $\pi_i \notin P_i$ for $1 \leqslant i \leqslant n-1$ (this is equivalent to saying

$$\sum_{j=1}^{i} p_j \neq \tfrac{1}{2}i(i+1)) .$$

For $n = 4$, G_4 contains the following seven permutations:

2413 3142 4132 4321
2431 3241 4213

Permutations not belonging to G_n can be divided into $2n-2$ mutually disjoint subsets depending on the nature in which they violate (1) and/or (2).

For $1 \leqslant i \leqslant n-1$ define $S_{i,n}$, $T_{i,n} \subset P_n$ as follows:

$\pi \varepsilon S_{i,n}$ if and only if $\pi_i \varepsilon P_i$ and there does not exist a $j < i$ such that $\pi_j \varepsilon P_j$

$\pi \varepsilon T_{i,n}$ if and only if $\pi \notin S_{k,n}$ for any k, mark $i+1$ appears immediately to the right of mark i, and there does not exist a $j < i$ such that $\pi \varepsilon T_{j,n}$.

For the case of $n = 4$:

$$S_{1,4} = \{1234, 1243, 1324, 1342, 1423, 1432\}$$
$$S_{2,4} = \{2134, 2143\}$$
$$S_{3,4} = \{2314, 3124, 3214\}$$
$$T_{1,4} = \{3412, 4123, 4312\}$$
$$T_{2,4} = \{2341, 4231\}$$
$$T_{3,4} = \{3421\}.$$

From the definitions it is clear that the above sets are mutually disjoint and that

$$P_n = G_n \cup (\cup_{i=1}^{n-1} S_{i,n}) \cup (\cup_{i=1}^{n-1} T_{i,n}) .$$

Using the small letters to represent the cardinality of sets denoted by capital letters; respectively g_n, $s_{i,n}$, $t_{i,n}$ for the cardinality of G_n, $S_{i,n}$, and $T_{i,n}$, we have that

$$g_n = n! - \sum_{i=1}^{n-1} s_{i,n} - \sum_{i=1}^{n-1} t_{i,n} . \quad (3)$$

Expressing $s_{i,n}$ and $t_{i,n}$ in terms of g_j's, $j \leqslant i$, we can obtain the following recursive formula for g_n, $n > 2$,

$$g_n = (n-1)(n-1)! - \sum_{i=2}^{n-1} g_i \left[\sum_{k=0}^{n-1} (n-i-k)! \binom{i+k-1}{k}\right] .$$

A sample of s, t, and g values is listed in **Table 1**.

2. Generation algorithms

We now present two procedures to generate in alphabetical order all the permutations in G_n. The first procedure constructs the subsets of marks that can be placed in each location and uses standard backtracking to update these subsets and generate the permutations.

Assume $p_1 \ldots p_i$ is a subsequence satisfying both of our conditions and let R be the set of remaining marks (the set REMSET in the procedure below). $R = Z_n/\{p_1, p_2, \ldots, p_i\}$ where / stands for the set difference operator. Constructing the set of all marks which could occupy location i is based on the fact that condition (2) is satisfied if and only if $m_i = \text{maximum}(p_1, p_2 \ldots, p_i) > i$. If $m_i = i+1$, then there exists a mark $x \varepsilon R$, $x \leqslant i$, and p_{i+1} must not be allowed to take on the value of x, except for the case when $i = n-1$. If $m_i > i+1$ then the choice of p_{i+1} is not constrained by (2) since for any p_{i+1} chosen from R, $m_{i+1} \geqslant m_i > i+1$ and this implies the presence of a mark larger than $i+1$ in the subsequence. Finally to exclude the possibility of violating condition (1), we simply consider $R/\{p_i + 1\}$ when we derive the set of marks that can occupy the $i+1$'st location.

Table 1 Sample values for s, t, and g

$s_{1,2} = 1$	$s_{1,3} = 2$	$s_{1,4} = 6$	$s_{1,5} = 24$	$s_{1,6} = 120$	$s_{1,7} = 720$
	$s_{2,3} = 1$	$s_{2,4} = 2$	$s_{2,5} = 6$	$s_{2,6} = 24$	$s_{2,7} = 120$
		$s_{3,4} = 3$	$s_{3,5} = 6$	$s_{3,6} = 18$	$s_{3,7} = 72$
			$s_{4,5} = 13$	$s_{4,6} = 26$	$s_{4,7} = 78$
				$s_{5,6} = 71$	$s_{5,7} = 142$
					$s_{6,7} = 461$
$t_{1,2} = 0$	$t_{1,3} = 1$	$t_{1,4} = 3$	$t_{1,5} = 13$	$t_{1,6} = 71$	$t_{1,7} = 461$
	$t_{2,3} = 1$	$t_{2,4} = 2$	$t_{2,5} = 10$	$t_{2,6} = 58$	$t_{2,7} = 390$
		$t_{3,4} = 1$	$t_{3,5} = 8$	$t_{3,6} = 48$	$t_{3,7} = 332$
			$t_{4,5} = 7$	$t_{4,6} = 40$	$t_{4,7} = 284$
				$t_{5,6} = 33$	$t_{5,7} = 244$
					$t_{6,7} = 211$
$g_2 = 1$	$g_3 = 1$	$g_4 = 7$	$g_5 = 33$	$g_6 = 211$	$g_7 = 1525$

Assuming array P contains a permutation in G_n, but not the alphabetically last one, the procedure below generates its immediate alphabetic successor. The sets LOCSET contain those marks which are larger than $P[i]$ and can replace it without changing the fact that the first i marks satisfy both of the conditions.

```
(*type subset = set of 1 . . N;                                  *)
(*var REMSET:subset; I,N,TEMP:integer;                           *)
(*    LOCSET:array[1 . . N] of integer;                          *)
(*    P,MAX:array[0 . . N] of integer;                           *)
(*To find the alphabetically first permutation                    *)
(*satisfying both conditions, initialise                          *)
(*LOCSET[1] := [2, 3, . . . , N];                                 *)
(*REMSET := [1, 2, . . . , N];                                    *)
(*MAX[0] := 0;                                                    *)
(*for K := 2 to N do Y[K] := [ ];                                 *)
(*and call PERMBACK with I := 1.                                  *)
(*For subsequent permutations set I := N and                      *)
(*REMSET := [P[N]] prior to calling PERMBACK.                     *)

procedure PERMBACK;
begin while I <= N do
      begin while LOCSET[I] = [ ] do
            begin I := I - 1;
                  REMSET := REMSET + [P[I]]
            end;
      TEMP := 1;
      while not (TEMP in LOCSET[I])
            do TEMP := succ(TEMP);
      P[I] := TEMP;
      LOCSET[I] := LOCSET[I] - [TEMP];
      REMSET := REMSET - [TEMP];
      if MAX[I - 1] > TEMP
        then MAX[I] := MAX[I - 1]
        else MAX[I] := TEMP;
      LOCSET[I + 1] := REMSET - [succ(TEMP)];
      if (I <> N - 1) and (MAX[I] = I + 1)
        then begin TEMP := 1;
             while not (TEMP in REMSET)
              do TEMP := succ(TEMP);
              LOCSET[I + 1] := LOCSET[I + 1] -
                                          [TEMP]
             end;
      I := I + 1
      end
end
```

A somewhat different algorithm which doesn't use explicit backtracking can be based on Dijkstra's (1976) solution to finding the next permutation. This would also eliminate the need for the extra memory required for the locational and remainder subsets in the previous algorithm.

Given $p_1p_2 \ldots p_n$, Dijkstra uses the following four steps to transform it into its immediate alphabetic successor.

Determine i: find the maximum value i less than n, such that $p_i < p_{i+1}$.

Determine j: find the value of j in the range $i + 1 \leqslant j \leqslant n$ such that p_j is the smallest value satisfying $p_j > p_i$.

Swap (i, j): exchange p_i with p_j.

Sort the tail; reverse the order of elements from p_{i+1} to p_n.

Clearly the resulting permutation will not in general satisfy our requirements and therefore, a number of checks and further exchanges need to be performed. Assuming the original permutation belonged to G_n the subsequence $p_1p_2 \ldots p_{i-1}$ would still satisfy our requirements following the four steps of Dijkstra's algorithm. If the exchange of p_i with p_j results in $p_i = p_{i-1} + 1$, then clearly, there can not be any permutation in G_n starting with $p_1p_2 \ldots p_i$. The algorithm loops back and determines new i and new j and swaps the corresponding marks. This process is repeated until $p_i \neq p_{i-1} + 1$, which can always be achieved if we start with a permutation different to the alphabetically last one and let $p_0 = 0$. Following the execution of this loop, p_i will not contradict (2) since the new p_i is larger in value than the old one.

After sorting the tail in increasing order, we must check that each of the new tail marks conforms with the two conditions. Assume that the check has been satisfactorily performed up to p_i. The addition of p_{i+1} can not violate (1) and (2) simultaneously, since this would imply that $p_1 \ldots p_i$ contradicted (2). If $p_{i+1} = p_i + 1$ or if the sum of the first $i + 1$ marks is equal to the sum of the integers from 1 to $i + 1$, then swap p_{i+1} with p_{i+2}. If this does not lead to an acceptable subsequence, it must result in having (1) compromised. In the latter case we execute one more swap and exchange the new p_{i+1} with p_{i+3} when $i \leqslant n - 3$. This must lead to an acceptable subsequence of $i + 1$ marks since p_{i+3} was larger than its two immediate predecessors. If $i > n - 3$ and the last swap could not be performed then we begin again and search for new i and new j.

```
(*var I, J, K, N, SUM, TEMP:integer;                             *)
(*    P, VSUM:array[0 . . N] of integer;                         *)
(*Initialise P[0] := 0;                                           *)
(*for K := 1 to N do VSUM[K] := K*(K + 1)/2;                      *)
(*For the alphabetically first permutation satisfying            *)
(*both conditions, set P[1 . . N] := [1, N, N - 1, . . . , 2].    *)
(*For subsequent permutations, P[1 . . N] will contain           *)
(*the previous permutation. In either case                        *)
(*call DIJKSTRA_MOD followed by PERMDIJK.                         *)

procedure SWAP(L, M:integer);
 begin TEMP := P[L]; P[L] := P[M]; P[M] := TEMP
 end;

procedure DIJKSTRA_MOD;
 begin repeat
       begin I := N; repeat I := I - 1
                     until P[I] < P[I + 1];
             J := N + 1; repeat J = J - 1
                         until P[J] > P[I];
       SWAP(I, J)
       end
       until P[I] <> P[I - 1] + 1;
 K := I + 1; J := N;
 while K < J do begin SWAP(J, K);
                 K := K + 1; J := J - 1
                 end;
 SUM := 0;
 for K := 1 to I - 1 do SUM := SUM + P[K]
 end;

procedure PERMDIJK;
 begin while I < N - 1 do
  begin SUM := SUM + P[I]; I := I + 1;
   if (P[I] = P[I - 1] + 1)
     or (SUM + P[I] = VSUM[I]) then
      begin SWAP(I, I + 1);
        if P [I] = P [I - 1] + 1 then
          if I < N - 1 then SWAP(I, I + 2)
                       else DIJKSTRA_MOD
      end
   end
 if P[N] = P[N - 1] + 1 then SWAP(N - 1, N)
 end
```

The above algorithm and the preceding outline of its correctness are considerably more complicated than our backtracking

based solution; however, empirical results indicate that it is more efficient.

Appendix Cardinality of G_n

Lemma 1

(i) $s_{1,n} = (n-1)!$

(ii) $s_{i,n} = (n-i)s_{i,n-1}$ for $i \leqslant n-2$

(iii) $s_{n-1,n} = (n-1)! - \sum_{i=1}^{n-2} s_{i,n-1}$

Proof:

(i) By definition $S_{1,n}$ consists of all the permutations having mark 1 in their leftmost position. This leaves $n-1$ locations that can be occupied by any one of the remaining $n-1$ marks, and there are $(n-1)!$ distinct ways this can be accomplished.

(ii) Consider an element in $S_{i,n-1}$. In it are $n-i$ positions into which mark n can be placed which would leave the subsequence $\pi_i \varepsilon P_i$ intact, and produce distinct permutations in $S_{i,n}$. Thus $s_{i,n} \geqslant (n-i)s_{i,n-1}$. To prove equality let $\pi_i \varepsilon P_i$. Mark n can not occupy the first i locations. Thus for some k, $i < k < n$, $\pi = p_1 \ldots p_i \ldots p_{k-1} n p_{k+1} \ldots p_n$. By removing n and shifting all the marks on its right one location to the left we obtain a permutation in $S_{i,n-1}$. This implies that by our previous construct π is obtained from $S_{i,n-1}$. Thus equality.

(iii) Let $\pi = p_1 \ldots p_{n-1} \varepsilon P_{n-1}$ and $\pi \notin S_{i,n-1}$ for any $i \leqslant n-2$. Then by definition $p_1 \ldots p_{n-1} n \varepsilon S_{n-1,n}$. Conversely, $\pi = p_1 \ldots p_n \varepsilon S_{n-1,n}$ implies that $p_1 \ldots p_{n-1} \varepsilon P_{n-1}$ and $\pi_i \notin P_{n-1}$ and $\pi_i \varepsilon P_i$ for any $i \leqslant n-2$. Thus $p_n = n$ and $\pi_{n-1} \notin S_{i,n-1}$ for any $i \leqslant n-2$. This one-to-one onto correspondence completes the proof.

Lemma 2

(i) $t_{1,n} = (n-1)! - \sum_{j=1}^{n-2} s_{j,n-1}$

(ii) $t_{i,n} = (n-1)! - \sum_{j=1}^{n-2} s_{j,n-1} - \sum_{j=1}^{i-1} t_{j,n-1}$, for $2 \leqslant i \leqslant n-1$.

Proof

For $1 \leqslant i \leqslant n-1$ define the mapping $f_i: T_{i,n} \to P_{n-1}$ by the following construction: given $\pi \varepsilon T_{i,n}$ shift all marks to the right of $i+1$ one place to the left erasing mark $i+1$ in the process, and then replace all marks $k > i+1$ by $k-1$. The proof now consists of showing that f_1 is a one-to-one mapping of $T_{1,n}$ on to $P_{n-1} - \bigcup_{j=1}^{n-2} S_{j,n-1}$, and similarly that f_i is a one-to-one mapping of $T_{i,n}$ on to $P_{n-1} - \bigcup_{j=1}^{n-2} S_{j,n-1} - \bigcup_{j=1}^{i-1} T_{j,n-1}$ for $2 \leqslant i \leqslant n-1$. The rest is straightforward.

In the theorem below, to simplify the notation G_1 is taken to be the empty set, and consequently $g_1 = 0$. For the case $n = 2$, $g_2 = 1$ since the permutation 21 satisfies both conditions.

Theorem

$$g_n = (n-1)(n-1)! - \sum_{i=2}^{n-1} g_i \left[\sum_{k=0}^{n-i} (n-i-k)! \binom{i+k-1}{k} \right] \text{ for all } n > 2 .$$

Proof

From Lemma 2,

$$t_{i,n} = (n-1)! - \sum_{j=1}^{n-2} s_{j,n-1} - \sum_{j=1}^{i} t_{j,n-1} + t_{i,n-1}$$
$$= t_{i+1,n} + t_{i,n-1}$$

Repeated application of this recursion formula yields

$$t_{i,n} = \sum_{k=0}^{n-1-i} \binom{n-1-i}{k} t_{n-1-k,n-k}$$

which can also be written as

$$t_{i,n} = \sum_{k=0}^{n-1-i} \binom{n-1-i}{k} g_{n-1-k}$$

since $t_{n-1-k,n-k} = g_{n-1-k}$ from (3) and Lemma 2. From Lemma 1 we have that

$$s_{1,n} = (n-1)!,$$

and for $1 < i \leqslant n-1$,

$$s_{i,n} = (n-i)s_{i,n-1} = \ldots = (n-i)!s_{i,i+1}$$
$$= (n-i)![(i-1)! - \sum_{j=1}^{i-1} s_{j,i}] = (n-1)![t_{1,i1}]$$
$$= (n-1)! \sum_{k=0}^{i-1} \binom{i-1}{k} g_{i-k} .$$

Substituting all this into (3) we get

$$g_n = n! - (n-1)! - \sum_{i=2}^{n-1} \left[(n-i)! \sum_{k=0}^{i-1} \binom{i-1}{k} g_{i-k} \right] - \sum_{i=1}^{n-1} \left[\sum_{k=0}^{n-1-i} \binom{n-1-i}{k} g_{n-1-k} \right].$$

For the last term above, letting $n-1-k = j$ and $i+1 = l$, we can combine like terms of g_j and transform the double sum into

$$\sum_{j=2}^{n-1} g_j \sum_{l=2}^{j+1} \binom{n-l}{n-j-1} = \sum_{j=2}^{n-1} g_j \binom{n-1}{n-j}$$
$$= \sum_{j=2}^{n-1} g_j \binom{j+(n-j)-1}{n-j}$$

The other double sum can also be transformed similarly by considering all possible pairs of i and k such that $i-k = j$ is constant. Combining like terms together we get

$$\sum_{j=2}^{n-1} g_j \sum_{l=0}^{n-j-1} (n-(j+l))! \binom{j+l-1}{l}$$

Putting all the pieces together and renaming j as i, and l as k completes the proof.

References

DIJKSTRA, E. W. (1976). *A discipline of programming*, Prentice Hall, Englewood Cliffs, N.J.

FOULKES, H. O. (1976). Enumeration of permutations with prescribed up-down and inversion sequences, *Discrete Math.*, Vol. 15, pp. 235-252.

KEANE, M. (1975). Interval exchange transformations, *Math. Z.*, 141, pp. 25-31.

KNUTH, D. E. and SZWARCFITER, J. I. (1974). A structured program to generate all topological sorting arrangements, *Inf. Proc. Letters*, Vol. 2, pp. 153-157.

MACCALLUM, I. R. (1977). Letter to the Editor: Surveyor's Forum, *Comput. Surv.*, Vol. 9 No. 4, pp. 316-317.

RIORDAN, J. (1958). *An introduction to combinatorial analysis*, John Wiley, New York.