Ranking and Unranking Permutations with Applications

F. Critani, M. Dall'Aglio, G. Di Biase
Dept. of Science, G. D'Annunzio University, viale Pindaro 42, 65127 Pescara, Italy.
critani@sci.unich.it, maglio@sci.unich.it, dibiase@sci.unich.it

1 Introduction

Permutation theory has many applications in several fields of science and technology and it also has a charm in itself. *Mathematica* is particularly suitable for writing combinatorial algorithms because it provides many easy-to-use tools for handling lists.

Several combinatorial built-in functions which involve permutations and combinations are avaliable as standard add-on *Mathematica* packages. They are grouped under the name of `DiscreteMath` and are described by their author Steven Skiena in his book [7].

In this paper we focus on ranking and unranking procedures and we examine and implement alternative algorithms to the `RankPermutations` and `NthPermutations` already contained in the above mentioned add-on packages. Moreover we provide some applications of these ranking procedures in different topics such as probability, statistics and elementary calculus.

Let $I_n=\{0, 1, ..., n-1\}$ be the set of the nonnegative integers smaller than n and let A be a n-set. We denote the set of all permutations of A by the $n!$-set:

$$S_n = \{\sigma_i;\ i \in I_{n!}\}.$$

Each element of S_n is an ordered string of n objects:

$$\sigma_i = (a_{i0}, a_{i1}, ..., a_{ij}, ..., a_{i,n-1}) \qquad (1)$$

where $a_{ij} \in A, i \in I_{n!}, j \in I_n$ with $a_{ij} \neq a_{ik} \ \forall \ j \neq k$. In other words a permutation specifies an arrangement of the n objects of A. Algorithms for constructing permutations can be found e.g. in Knuth [3] or Sedgewick [6].

By default, *Mathematica* considers the lexicographic order among permutations. In fact, the built-in function `Permutations` lists all $n!$ permutations of the elements of A according to this order. The same choice is adopted here and S_n is assumed to be lexicographically ordered.

2 Ranking and unranking functions

When we are interested in a particular permutation with a certain property, it is more efficient to generate it directly, rather than construct all $n!$ permutations in advance and search for it afterwards. A particular case of this is when we look for a subset of S_n corresponding to some given position in the ordered array of permutations.

Moreover, we might need to work out the converse problem, i.e. to locate a given permutation in the ordered set S_n. The former is the problem of finding a *ranking* algorithm able to determine the position of a given permutation, with respect to the considered order, that is the number $p \in I_{n!}$ for a given $\sigma_p \in S_n$. Conversely, the latter is the problem of finding an *unranking* algorithm able to generate the p-th permutation σ_p for a given $p \in I_{n!}$. Many authors have tackled these problems, see e.g. Critchlow [2], Knuth [3] and Lehmer [4]. A large number of related algorithms have been codified on computer system languages and *Mathematica* is among them.

In particular Skiena [7] wrote the recorsive *Mathematica* function `RankPermutation` based on the observation that "...all permutations sharing the same first element k are ranked $(k\text{-}1)(n\text{-}1)!$ to $k(n\text{-}1)!\text{-}1$ in the total order, and after deleting the first element we may recur on a permutation of length n-1 to determine the exact rank". We refer to [7] for the detailed *Mathematica* code.

The following is an example of the use of this function together with an evaluation of its time efficiency. This and the following performaces are based on a 166 MHz Pentium machine with *Mathematica* 3.0.

```
In[1]:= σ=RandomPermutation[50];
        Timing[RankPermutation[σ]]

Out[1]:={0.32 Second,16443658893335817203146 5˙˙.
     9855367062375103274016081498081428532377 9}
```

The converse problem has been solved by Skiena [7] by means of the function `NthPermutation[p,σ]` based on the quotient algorithm (see e.g. Lehmer [4]). An example of the time performance of this function is:

```
In[2]:= Timing[Short[NthPermutation[3,Range[2000]],3]

Out[2]= {45.8 Second, {1, 2, 3, 4, 5, 6, <<1988>>,
          1995, 1996, 1997, 1999, 2000, 1998}}
```

Our *Mathematica* ranking-unranking permutation functions are based on the following results. We need a preliminary definition.

Definition 2.1 For any $\sigma_p \in S_n$ let p be its rank. We call *factorial base representation* of p the n-vector:

$$\omega_p = (\omega(a_{p0}), \omega(a_{p1}), \ldots, \omega(a_{p,n-1})) \tag{2}$$

where $\omega(a_{pi})$ $(\forall i \in I_n)$ represents the position of a_{pi} inside the set obtained from A by dropping all a_{pj} such that $j < i$.

Proposition 2.2 For any $\sigma_p \in S_n$ and $i \in I_n$ the elements of the *factorial base representation* of p satisfy the following inequality:

$$0 \le \omega(a_{pi}) \le n - i - 1 \tag{3}$$

Proposition 2.3 For any $\sigma_p \in S_n$ and $i \in I_n$ let $\rho(a_{pi})$ be the position of a_{pi} with respect to the identity permutation. Moreover denote by $\delta(a_{pi}, a_{pk})$ the precedence indicator function between two elements of σ_p (namely $\delta(a_{pi}, a_{pk}) = 1$ if a_{pi} precedes a_{pk} and it holds 0 otherwise). Any element of the *factorial base representation* of the rank of σ_p can be found by means of

$$\omega(a_{pi}) = \rho(a_{pi}) - \sum_{k:k\in I_i} \delta(a_{pi}, a_{pk}) \tag{4}$$

The proofs of Propositions 2.2 and 2.3 follow directly from Definition 2.1.

In Critani & Di Biase [1] it has been proved that the *factorial base representation* of the rank of a given permutation is univocally determined and that the following result holds.

Theorem 2.4 For any $\sigma_p \in S_n$, its rank p can be obtained by the following:

$$p = \sum_{i\in I_n} \omega(a_{pi}) \cdot (n - i - 1)! \tag{5}$$

Sketch of the Proof p is the number of permutations which precede σ_p. There are $\omega(a_{p0})(n-1)!$ permutations whose first position precedes a_{p0}. Similarly, there are $\omega(a_{p0})(n-1)!+\omega(a_{p1})(n-2)!$ permutations whose first two positions precede (a_{p0}, a_{p1}) in lexicographic order, and so on.

The rank p of a given permutation can be easily found with the help of *Mathematica*. First we find out the factorial base representation of σ (see **In[1]**) by means of:

```
In[3]:= δ[x_Integer,y_Integer]:= If[x>y,1,0];
        ω[z_List]:= Table[z[[i]]-
        Sum[δ[z[[i]],z[[j]]],{j,1,i}]-1,{i,1,Length[z]}];
        ω[σ]

Out[3]={27,1,29,24,20,44,40,40,36,19,22,18,35,34,
       12,27,25,5,9,25,26,4,1,23,9,12,13,1,11,18,
       9,0,16,16,4,7,8,7,4,9,9,3,2,0,3,2,1,2,1,0}
```

Next we obtain the rank by implementing the translation of formula (5):

```
In[4]:= Length[σ]=s; Timing[Apply[Plus,
        Table[ω[i] ((s-1)-(i-1))!, {i,1,s}]]]

Out[4]:={0.33 Second,164436588933358172031465...
     985536706237510327401608149808142853237 79}
```

The unranking function is based on the following procedure, called *module algorithm* (see Critani & Di Biase [1]).

Algorithm 2.4 We formulate the algorithm directly as a *Mathematica* code

```
In[5]:= unranking[n1_Integer, l_List]:=
        Block[{A={}, R, i, q=n1, module, s=l, el},
          R=Table[module=Mod[q,i];q=Quotient[q,i];
                  module, {i,1,Length[l],1}];
          Table[el=R[[i]]+1; A=s[[el]];
                s=Complement[s, {s[[el]]}];
                A, {i,Length[l],1,-1}]]
```

Explanation of the module algorithm. First of all notice that ineq (3) implies $\omega(a_{p,n-1}) = 0$. This term can be erased from the sum on the r.h.s of eqn (5).

Step 1. The resulting sum contains terms that are multiple of 2 with the exception of the last one. This is either 0 or 1 by (3) and can be obtained as the module of (5) divided by 2.
Step 2. Since the quotient of the last division is

$$\omega(a_{p0})\frac{(n-1)!}{2}+\ldots+\omega(a_{p,n-4})\frac{3!}{2}+\omega(a_{p,n-3})\frac{2!}{2} \qquad (6)$$

all terms but the last are multiples of 3. The last one is an integer in the range {0, 1, 2} and represents the module of (6) divided by 3.
Step 3. This procedure can be iterated until the quotient becomes null. The modules obtained in the repeated divisions are the elements of the factorial base. Therefore this allows us to determine the sought permutation.

When card(A) is large the function `unranking` runs faster than `NthPermutation`. Indeed, for the same permutation used in **`In[2]`**, we have:

```
In[6]:= Timing[Short[unranking[3,Range[2000]],3]

Out[6]= {32.63 Second, {1, 2, 3, 4, 5, 6,<<1988>>,
          1995, 1996, 1997, 1999, 2000, 1998}}
```

The faster time performance is not surprising if we observe that Skiena's function has to compute many factorial numbers.

3 Applications in probability, statistics and elementary calculus

3.1 Rank and parity of a permutation.

A given permutation (1) of n items has an *inversion* each time $a_{ij} > a_{ik}$ while $j < k$. A permutation is called *even* (respectively *odd*) if it contains an even (odd) number of inversions. This notion is fruitfully employed, for instance, in the definition of determinant of a matrix. The relationship between the parity of a permutation and its lexicographic order is now considered. The following *Mathematica* function, `findeven[n]`, returns the rank of the even permutations in S_n.

```
findeven[1]={1};
findodd[n_]:=Complement[Range[1,n!],findeven[n]];
findeven[n_]:=Flatten[Table[i(n-1)!+Which[EvenQ[i],
        findeven[n-1],True,findodd[n-1]],{i,0,n-1}]]
```

The function is defined by recursion. It complements over the range of the first $n!$ integers to find the position of odd permutations. For any given $n > 1$,

`findeven[n]` is obtained by alternating `findeven[n-1]` with `findodd[n-1]`. This simple function, in conjunction with the ranking permutation procedure, enables us to find all the even permutations of n items without a previous listing of all the permutations by means of the following.

```
evenp = findeven[n];
Table[unranking[evenp[[i]]-1,Range[n]],{i,n!/2}]
```

3.2 Random permutations without repetition.

When sampling from a finite population, we often need a procedure that allows us to generate a sample of k random permutations of n items without repetition (i.e. no permutation can match any other). This is needed, for instance, in sampling theory and design experiments. We proceed as follows: we extract a random sample without repetition of k elements from a population of $n!$ and we take the corresponding elements in the ordered list of all permutations. *Mathematica* does not provide for a built-in statistical function that extracts a random subpopulation, but the function `RandomKSubset` accomplishes the same task in the context of combinatorial analysis. The following function generates k random permutations without repetition.

```
draw[k_,n_]:= Module[{a,b,c},
                a=RandomKSubset[n!,k]-1; b=Range[n];
                Table[unranking[a[[i]],b}],{i,k}]
```

3.3 Permutation models.

Permutation models are statistical models in which a probability value is assigned to each of the possible permutations of n items. Such models have immediate application wherever an "order of arrival" is considered. A typical example is given by horse racing (Plackett [5]). Also this kind of model is important whenever we want to assess whether the order of presentation of certain items is relevant. Take for instance the order of candidates in a ballot.

Let $A=\{1, 2, \ldots, n\}$ and $\{\pi_i;\ i \in I_{n!}\}$ be a probability distribution assessed on S_n. An interesting model is described in Plackett [5]. This is a hyerarchical model which can include up to $n!$ - 1 parameters. The first stage parameters are defined as follows

$$p_j = \sum_{i \in S_n(j)} \pi_i ; \quad j \in I_n$$

with $S_n(j) = \{\sigma_i \in S_n : \sigma_i \text{ has item } j \text{ in the first place}\}$. The second stage parameters consider the first two places of the permutation

$$p_{jk} = \sum_{i \in S_n(j,k)} \pi_i ; \qquad j,k \in I_n ;\ j \neq k$$

with $S_n(j,k) = \{\sigma_i \in S_n : \sigma_i \text{ has item } j \text{ (respect. } k\text{) in the first (second) place}\}$.

The third stage parameters p_{jkl} fix the first three positions, and so on up to stage n-1. A hyerarchical model can be designed so to include up to the r-th stage parameters where $r \in I_n \setminus \{0\}$. In order to estimate the parameters, some data is needed. Let m_i be the absolute frequency (i.e. the number of observations) of permutation σ_i. In order to give a maximum likelihood estimate of the first stage parameters p_j the following functions of the observed data are needed

$$w_j = \sum_{i \in S_n(j)} m_i ; \quad w_{jk} = \sum_{i \in S_n(j,k)} m_i ; \quad j,k \in I_n , \; j \neq k$$

and similar quantities are defined for the remaining stages. Moreover we need:

$$w_{-j} = \sum_{i \in S_n(-j)} m_i \quad \text{with} \quad S_n(-j) = \{\sigma_i \in S_n : \sigma_i \text{ has item } j \text{ in the last place}\}.$$

Those quantities can be easily computed from the original array of data when we the lexicographic order is considered. Let us denote with `mfreq` the ordered array of observed frequencies $\{m_i ; \; i \in I_n\}$. The function `unranking` and `ranking` can be used respectively to detect the permutation associated to the i-th element of the array of data and to read the observed frequency associated to a given permutation σ:

```
unranking[i,Range[n]]; mfreq[[ranking[σ]]]
```

Last instruction can be slightly modified in order to input or modify the frequency of a permutation. The computation of the first stage parameters is fairly easy when the lexicographic order is used. In fact the first (n-1)! permutations have "1" in first place, followed by other (n-1)! permutations starting with "2", and so on. Thus we can define:

```
w1[j_Integer,n_Integer]:=Sum[mfreq[[i]],
                        {i,(j-1)(n-1)!+1,j(n-1)!}].
```

When second stage parameters are considered, things are conceptually just as simple, while the *Mathematica* function is slightly more complicated due to the presence of two indexes which never match each other:

```
w2[j_Integer,k_Integer,n_Integer]:= Block[{q},
   If[k<j,q=0,q=1]; Sum[mfreq[[i]],{i,j-1)(n-1)!+
     (k-1-q)(n-2)!+1,(j-1)(n-1)!+(k-q)(n-2)!}]]
```

The lexicographic order does not handle permutations with fixed last position as easily as those with fixed first place(s). However this is not a major difficulty since we can write the following function defined by recursion:

```
posend[1,1]={1};
posend[k_Integer,n_Integer]:=Flatten[Table[
       Which[i<k,(i-1)(n-1)!+posend[k-1,n-1],i==k,
       {},True,(i-1)(n-1)!+posend[k,n-1],{i,1,n}]]
```

The function `posend[k,n]` returns the position in S_n of the permutations that have element "k" in last place. It relies on the lists returned by `posend[k-1,n-1]` and `posend[k,n-1]`. It is now easy to compute the w_{-j}'s by a straightforward application of their defining formula:

```
wm[j_Integer,n_Integer]:= mfreq[[posend[j,n]]];
Table[1,{i,(n-1)!}].
```

References

1. Critani, F. and Di Biase, G. Ordinamento e generazione diretta di permutazioni, in Il Fascino Discreto della Matematica, pp. 132-146, *Atti del Convegno MATHESIS*, L'Aquila, Italia, 1996, Libreria Benedetti, L'Aquila, 1996.

2. Critchlow, D. E. Metric *Methods for analyzing Partially Ranked Data*, Lecture Notes in Statistics, 34, Springer-Verlag, New York, 1987.

3. Knuth, D. E. *The Art of Computer Programming*, Addison Wesley, Reading, Massachusetts, 1973.

4. Lehmer, D. H. The machine tools of combinatorics, *Applied Combinatorial Mathematics*, ed. E.F. Beckenbach, pp. 5-31, Wiley, New York, 1964.

5. Plackett, R.L. The Analysis of Permutations, *Applied Statistics*, 1975, **24**, 193-202.

6. Sedgewick, R. *Algorithms*, Addison Wesley, Reading, Massachusetts, 1988.

7. Skiena, S. *Implementing Discrete Mathematics*, Addison Wesley, New York, 1990.