*Short Note*

# A Note on Enumerating Combinations in Lexicographical Order

ICHIRO SEMBA*

The algorithm for generating combinations of $k$ things chosen from $n$ things in lexicographical order is well known. In this paper, we improve that algorithm and show that the improved algorithm is faster than the original for large $k$'s.

## 1. Introduction

The problem of generating combinations is an interesting example of the use of computers in combinatorial mathematics and provides a very instructive exercise in the implementation and analysis of algorithms. This problem is simply stated, but not easily solved. Various algorithms [1, 2, 3, 4] have been given for this problem. A detailed evaluation of performance and a summary of property of combination generating algorithms have been given by Payne and Ives [5].

In what follows, we improve the algorithm [2] which generates the sequence of combinations in lexicographical order, and show the improved algorithm is faster than the original for large $k$'s.

## 2. Definition

We consider generating all the $k$-subsets of the set $\{1, 2, \cdots, n\}$ in lexicographical order. We denote a $k$-subset by $X=(x_1, \cdots, x_k)$, where $1 \leq x_1 < \cdots < x_k \leq n$. Let the index $h$ be the largest $i$ such that $x_i < n-k+i$. If $x_i = n-k+i$ for all $i$ $(1 \leq i \leq k)$, then the index $h$ is defined to be 0. We denote the $m$-th $k$-subset in lexicographical order by $X^{(m)}=(x_1^{(m)}, \cdots, x_k^{(m)})$, where $1 \leq x_1^{(m)} < \cdots < x_k^{(m)} \leq n$. Let the index $h^{(m)}$ be the largest $i$ such that $x_i^{(m)} < n-k+i$. If $x_i^{(m)} = n-k+i$ for all $i$ $(1 \leq i \leq k)$, then the index $h^{(m)}$ is defined to be 0. We denote the number of different elements between $X^{(m)}$ and $X^{(m+1)}$ by $d^{(m)}$ $(1 \leq m < \binom{n}{k})$. We divide all the combinations into three classes, $C_1$, $C_2$ and $C_3$.

$$C_1 = \{X^{(m)} | x_{h^{(m)}}^{(m)} = n-k+h^{(m)}-1\},$$
$$C_2 = \{X^{(m)} | x_{h^{(m)}}^{(m)} < n-k+h^{(m)}-1\},$$
$$C_3 = \{X^{(m)} | h^{(m)} = 0\}.$$

We will give an example to gain a better understanding of our definitions.

Example. Let $n=6$, $k=4$. We show 4-subsets $X^{(m)}$, corresponding classes $C_i$, $h^{(m)}$ and $d^{(m)}$.

| $m$ | $X^{(m)}$ | class | $h^{(m)}$ | $d^{(m)}$ |
|---|---|---|---|---|
| 1 | 1234 | $C_2$ | 4 | 1 |
| 2 | 1235 | $C_1$ | 4 | 1 |
| 3 | 1236 | $C_2$ | 3 | 2 |
| 4 | 1245 | $C_1$ | 4 | 1 |
| 5 | 1246 | $C_1$ | 3 | 1 |
| 6 | 1256 | $C_2$ | 2 | 3 |
| 7 | 1345 | $C_1$ | 4 | 1 |
| 8 | 1346 | $C_1$ | 3 | 1 |
| 9 | 1356 | $C_1$ | 2 | 1 |
| 10 | 1456 | $C_2$ | 1 | 4 |
| 11 | 2345 | $C_1$ | 4 | 1 |
| 12 | 2346 | $C_1$ | 3 | 1 |
| 13 | 2356 | $C_1$ | 2 | 1 |
| 14 | 2456 | $C_1$ | 1 | 1 |
| 15 | 3456 | $C_3$ | 0 | |

## 3. Property of Original Algorithm

In this section, to begin with, we consider the original algorithm. The original algorithm generates $X^{(m+1)}$ from $X^{(m)}$ in a following way.

Step 1. Examine elements of $X^{(m)}$ from right to left to determine the index $h^{(m)}$.

Step 2. Change values of $x_i^{(m+1)}$ for $i = h^{(m)}, \cdots, k$.

It is interesting to measure the number of element examined to determine the index $h^{(m)}$ in step 1 and the number of element whose value is changed in step 2, in order to generate all the $k$-subsets in lexicographical order. Let us denote the former by $F_0(n, k)$ and the latter by $G_0(n, k)$.

Property 1. For $1 \leq k < n$,

$$F_0(n,k) = G_0(n,k) = \{(n+1)/(n-k+1)\}\binom{n}{k} - 1.$$

Proof. From the original algorithm, we immediately know that $F_0(n, k)$ is equal to $G_0(n, k)$. Thus we concentrate our attention upon determining $F_0(n, k)$. For a fixed $h$, the number of $k$-subsets whose number of element examined is $k-h+1$ is $\binom{n-k+h-1}{h}$, because $x_1^{(m)}, \cdots, x_h^{(m)}$ can be any $h$-subset of $\{1, \cdots, n-k+h-1\}$. We have to examine $k$ elements to know the last $k$-subset $(n-k+1, \cdots, n)$. Therefore it follows that

*Department of Pure and Applied Sciences, College of General Education, University of Tokyo.

$$F_0(n,k)=\sum_{h=1}^{k}(k-h+1)\binom{n-k+h-1}{h}+k$$
$$=\binom{n+1}{k}-1$$
$$=\{(n+1)/(n-k+1)\}\binom{n}{k}-1$$

This completes proof.

Thus the original algorithm does, on the average, less than $(n+1)/(n-k+1)$ examinations and changes per $k$-subset.

## 4. Property of Improved Algorithm

Now let us consider the improved algorithm. The improved algorithm is based on the following property.

Property 2. For $1\le k<n$,

(1) If $X^{(m)}\in C_1$, then $x_{h^{(m)}}^{(m+1)}=x_{h^{(m)}}^{(m)}+1$,
$x_i^{(m+1)}=x_i^{(m)}$ $(i\neq h^{(m)})$,
$h^{(m+1)}=h^{(m)}-1$.

(2) If $X^{(m)}\in C_2$, then $x_i^{(m+1)}=x_{h^{(m)}}^{(m)}+i-h^{(m)}+1$ $(h^{(m)}\le i\le k)$,
$x_i^{(m+1)}=x_i^{(m)}$ $(i<h^{(m)})$,
$h^{(m+1)}=k$.

Proof. The proof is direct consequence of the definition of lexicographical order.

This property suggests that we are able to determine $h^{(m+1)}$ from $h^{(m)}$ after the element $x_{h^{(m)}}^{(m)}$ is compared with $n-k+h^{(m)}-1$. Therefore we need not examine elements of $X^{(m+1)}$ from right to left. Especially, when $X^{(m)}\in C_1$, we have only to change the element $x^{(m+1)}$. Therefore, in this case, we need not change $k-h^{(m)}+1$ elements.

Let us denote the number of element examined by $F_i(n,k)$ and the number of element changed by $G_i(n,k)$.

Property 3. For $1\le k<n$,

(1) $F_i(n,k)=\binom{n}{k}$

(2) $G_i(n,k)=(1+k/n)\binom{n}{k}+1$.

Proof. It is obvious that $F_i(n,k)$ is equal to $\binom{n}{k}$. We try to determine $G_i(n,k)$. If $X^{(m)}\in C_1$, then $d^{(m)}=1$. The number of the set $C_1$ is $\sum_{h=1}^{k}\binom{n-k+h-2}{h-1}=\binom{n-1}{k-1}$, because for a fixed $h$, $x_1^{(m)},\cdots,x_{h-1}^{(m)}$ can be any $(h-1)$-subset of $\{1,\cdots,n-k+h-2\}$. If $X^{(m)}\in C_2$, then $d^{(m)}=k-h^{(m)}+1$. For a fixed $h$, the number of $k$-subsets whose value of $d^{(m)}$ is $k-h+1$ is $\binom{n-k+h-2}{h}$, because $x_1^{(m)},\cdots,x_h^{(m)}$ can be any $h$-subset of $\{1,\cdots,n-k+h-2\}$. We have to change $k$ times to initialize $X^{(1)}$.

It follows that

$$G_i(n,k)=\binom{n-1}{k-1}+\sum_{h=1}^{k}(k-h+1)\binom{n-k+h-2}{h}+k$$
$$=\binom{n-1}{k-1}+\binom{n}{k}-1$$
$$=(1+k/n)\binom{n}{k}-1$$

This completes proof.

Thus the improved algorithm does, on the average, one examination and less than $(1+k/n)$ changes per $k$-subset.

```
C     COMBINATION GENERATOR
C     LEXICOGRAPHICAL ORDER
      SUBROUTINE COMBI(X,N,K,H)
      INTEGER X(K),H
      IF(K-H) 10,20,30
   10 DO 11 I=1,K
         X(I)=I
   11 CONTINUE
      NK=N-K
      H=K
      IF(N.EQ.K) H=0
      GOTO 40
   20 X(H)=X(H)+1
      IF(X(H).EQ.N) H=H-1
      GOTO 40
   30 X(H)=X(H)+1
      IF(X(H).LT.NK+H) GOTO 31
      H=H-1
      GOTO 40
   31 H=H+1
      X(H)=X(H-1)+1
      IF(H.LT.K) GOTO 31
   40 RETURN
      END
```

Fig. 1 FORTRAN algorithm implementation.

## 5. Experimental Result

The improved algorithm, coded in FORTRAN, is given in Fig. 1. The $K$-subset of $\{1,\cdots,N\}$ is generated in the integer array $X$ in lexicographical order. The integer variable $H$ is nonlocal to COMBI and must be

Table 1 The average computing time to generate all the $K$-subsets of $\{1,\cdots,20\}$ without subroutine linkage time and the ratio of the improved algorithm to the original one. (times in milliseconds)

| $K$ | Original algorithm | Improved algorithm | Ratio |
|---|---|---|---|
| 1 | 0.23 | 0.20 | 0.870 |
| 2 | 0.97 | 0.97 | 1.000 |
| 3 | 5.90 | 6.27 | 1.063 |
| 4 | 24.67 | 25.33 | 1.027 |
| 5 | 94.60 | 81.23 | 0.859 |
| 6 | 254.47 | 214.93 | 0.845 |
| 7 | 562.30 | 421.13 | 0.749 |
| 8 | 975.63 | 719.80 | 0.738 |
| 9 | 1422.33 | 958.17 | 0.674 |
| 10 | 1640.83 | 1053.00 | 0.642 |
| 11 | 1738.43 | 990.07 | 0.570 |
| 12 | 1436.47 | 767.60 | 0.534 |
| 13 | 1029.50 | 484.87 | 0.471 |
| 14 | 597.37 | 242.47 | 0.406 |
| 15 | 282.37 | 98.90 | 0.350 |
| 16 | 106.00 | 31.30 | 0.295 |
| 17 | 31.73 | 8.03 | 0.253 |
| 18 | 7.20 | 0.97 | 0.135 |
| 19 | 1.40 | 0.27 | 0.193 |
| 20 | 0.30 | 0.07 | 0.233 |

greater than $N$ before the first call. While $K$-subset is generated, the integer variable $H$ remains $1 \leq H \leq K$. When the last $K$-subset is generated, the integer variable $H$ is set 0.

We have measured the computing time to generate all the $K$-subsets for the improved algorithm and the original algorithm, coded in FORTRAN, on a HITAC 8800/8700 at the Computer Centre of the University of Tokyo. We show the average computing time without the subroutine linkage time in Table 1. The result shows that the improved algorithm is faster than the original for large $K$'s.

**Acknowledgement**

The author would like to thank Prof. T. Shimizu for his hearty encouragement.

**References**

1. KURTZBERG, J. Combination (Algorithm 94), *Comm. ACM*, **5**, 6, (1962) 344.
2. MIFSUD, C. J. Combination in lexicographical order (Algorithm 154), *Comm. ACM*, **6**, 3, (1963) 103.
3. CHASE, P. J. Combinations of m out of n objects (Algorithm 382), *Comm. ACM*, **13**, 6, (1970) 368.
4. LIU, C. M. and TANG, D. T. Enumerating combinations of m out of n objects (Algorithm 452), *Comm. ACM*, **16**, 8, (1973) 485.
5. PAYNE, W. H. and IVES, F. M. Combination generators, *ACM Transactions on Mathematical Software*, **5**, 2, (1979), 163–172.