Short Note

An Efficient Algorithm for Generating all Partitions of the Set $\{1, 2, \ldots, n\}$

ICHIRO SEMBA*

We consider the problem of generating all partitions of the set $\{1, 2, \cdots, n\}$. An efficient algorithm based on backtrack technique is presented. The average running time per partition is proved to be bounded by a constant. Experiments showed that our algorithm is faster than other algorithms so far proposed.

1. Introduction

We consider the problem of generating all partitions of the set $\{1, 2, \cdots, n\}$. A partition of $\{1, 2, \cdots, n\}$ consists of m classes $C_1, C_2, \cdots, C_m$, where $C_i \cap C_j = \phi$ $(i \neq j)$, $\bigcup_{i=1}^{m} C_i = \{1, 2, \cdots, n\}$ and $C_i \neq \phi$ $(1 \leq i \leq m)$. Therefore, for $n=3$, we have the following 5 partitions: (1 2 3), (1 2)(3), (1 3)(2), (1)(2 3), (1)(2)(3).

A well-known generating algorithm is given in Nijenhuis and Wilf [1]. Kaye [2] has considered another algorithm generating successively all partitions by changing the class of exactly one element and has shown that the average running time per partition is bounded by a constant. We propose a generating algorithm based on backtrack technique and prove that the average running time per partition is bounded by a constant. Computer tests indicated that our algorithm was faster than other algorithms.

2. Generating Algorithm

In this section, we describe a new algorithm generating all partitions of the set $\{1, 2, \cdots, n\}$.

We assume that a partition P of $\{1, 2, \cdots, n\}$ consists of m classes $C_1, C_2, \cdots, C_m$. By the *children* of P, we mean the following partitions $P_1, P_2, \cdots, P_{m+1}$ of $\{1, 2, \cdots, n, n+1\}$.

$$P_1 : C_1 \cup \{n+1\}, C_2, \cdots, C_m$$
$$P_2 : C_1, C_2 \cup \{n+1\}, C_3, \cdots, C_m$$
$$\vdots$$
$$P_m : C_1, C_2, \cdots, C_m \cup \{n+1\}$$
$$P_{m+1} : C_1, C_2, \cdots, C_m, \{n+1\}$$

The first m children of P are obtained from P by inserting $n+1$ into one of the classes of P and the last one is obtained by adding a singleton $\{n+1\}$ to P. Therefore, all partitions of $\{1, 2, \cdots, n\}$ can be represented in a tree as in Fig. 2.1.

Our generating algorithm is established by traversing this tree. Backtrack technique is used to traverse this tree. We use two arrays, a_i $(1 \leq i \leq n)$, indicating the class to which element i belongs and g_i $(1 \leq i \leq n)$, representing the number of classes in the partition under consideration at level i.

When we traverse the tree, three cases are considered. Let k be the level of the node under consideration.

Case 1. If $k<n$, then we move down to the first son. Namely, we set $k \leftarrow k+1$, $a_k \leftarrow 1$ and $g_k \leftarrow g_{k-1}$.

Case 2. If $k=n$ and $a_k \leq g_k$, then we print out a solution $a_1, \cdots, a_n$ and move left to right in the level n of the tree. Namely, we set $a_k \leftarrow a_k + 1$.

Case 3. If $k=n$ and $a_k = g_k + 1$, then we backtrack. Namely, we set $k \leftarrow k-1$, until g_{k-1} becomes equal to g_k. Then, we set $a_k \leftarrow a_k + 1$. If $a_k > g_k$, then we set $g_k \leftarrow a_k$.

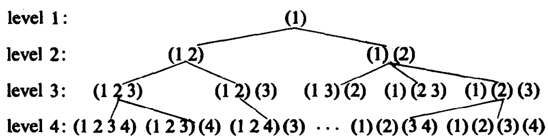

Fig. 2.1 A tree coresponding to partitions of $\{1, 2, \cdots, n\}$.

```
begin
a1:=1; g1:=1; k:=1;
1:
{Case 1}
while k<n do begin
    k:=k+1; ak:=1; gk:=gk-1
end;
output (a1, ···, an);
{Case 2}
while ak≤gk do begin
    ak:=ak+1; output (a1, ···, an)
end;
{Case 3}
repeat
    k:=k-1;
    if k=1 then stop;
until gk-1=gk;
ak:=ak+1;
if ak>gk then gk:=ak;
goto 1
end.
```

Fig. 2.2 Generating Algorithm.

*Department of Pure and Applied Sciences, College of General Education, University of Tokyo, Komaba, Meguro-ku, Tokyo 153, Japan.

Our algorithm is written in PASCAL-like notation in Fig. 2.2. The procedure "output $(a_1, \cdots, a_n)$" prints out the partition determined by $a_1, \cdots, a_n$.

3. Analysis of Generating Algorithm

In this section, we prove that the average running time per partition of $\{1, 2, \cdots, n\}$ is bounded by a constant.

The number of edges examined to traverse a tree is a reasonable measure of the work. We denote it by E_n.

Property 1. Let B_n be the number of partitions of $\{1, 2, \cdots, n\}$ (i.e., Bell number).

$$E_n < 2(B_n + \cdots + B_2) \quad (n \geq 2)$$

Proof. Obvious, since our generating algorithm is based on backtrack technique.

Property 2.

$$E_n/B_n < 4 \quad (n \geq 2)$$

Proof. Since $B_{i+1} > 2B_i$ $(2 \leq i \leq n-1)$, we have

$$E_n/B_n < 2(B_n + \cdots + B_2)/B_n$$
$$< 2\left(1 + \frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^{n-2}}\right) < 4$$

Theorem 1 Let $n \geq 2$.

The average running time per partition of $\{1, 2, \cdots, n\}$ is bounded by a constant.

Proof. By Property 2, it is easily shown.

4. Experimental Results

We have measured the time required to generate all partitions of $\{1, 2, \cdots, n\}$ for a well-known algorithm, Kaye's algorithm and our algorithm, coded in PASCAL, on a MELCOM-COSMO 900 II at Educational Computer Centre, University of Tokyo. The average running time required to generate all partitions of $\{1, 2, \cdots, n\}$ is shown in Table 1. We have also measured the time, coded in FORTRAN, on a M280H at the Computer Centre, University of Tokyo. The result is shown in Table 2. These results indicate that our algorithm is faster than other algorithms.

Table 1 The average running time required to generate all partitions of $\{1, 2, \cdots, n\}$. (times in milliseconds).

n	Nijenhuis's algorithm	Kaye's algorithm	Our algorithm
6	6.0	5.4	3.8
7	23.6	22.2	14.4
8	111.2	102.0	61.6
9	550.8	497.4	296.2
10	2,982.2	2,695.0	1,564.2

Table 2 The average running time required to generate all partitions of $\{1, 2, \cdots, n\}$. (times in milliseconds).

n	Nijenhuis's algorithm	Kaye's algorithm	Our algorithm
6	0.7	0.7	0.4
7	2.6	2.8	1.5
8	12.4	12.4	6.4
9	60.8	61.0	31.5
10	328.0	326.3	166.6
11	1,900.3	1,872.4	948.0

Acknowledgement

The author would like to thank referees for valuable comments.

References

1. NIJENHUIS, S. and WILF, H. S. Combinatorial Algorithms, Academic Press, New York, 1975, 81–86.
2. KAYE, R. A Gray Code for Set Partitions, Information Processing Letters, 5, 6 (1976), 171–173.

(Received May 27, 1982; revised Aug. 24, 1982)