JOURNAL OF ALGORITHMS 3, 41–44 (1982)

A Gray Code for Compositions

PAUL KLINGSBERG

Department of Mathematics and Computer Science, St. Joseph's University, Philadelphia, Pennsylvania 19131

Received May 3, 1981; revised July 1, 1981

The paper describes a rapid algorithm for the sequential listing of the compositions of n into k parts.

I. INTRODUCTION AND NOTATION

In [1], Nijenhuis and Wilf pose as an exercise the problem of finding, for each pair of positive integers n and k, a list $L(n, k)$ of the compositions of n into k parts (= ordered representations of n as a sum of k nonnegative integers) with this property: if an element π and its successor π' are viewed as configurations of unlabeled balls in labeled cells, then π' can be obtained from π by moving just one ball. These "Gray codes" $L(n, k)$ admit an elegant recursive description (see below); but straightforward translation of this description into a recursive algorithm for producing π' from π results in a procedure that seems unduly laborious, since its entire task is to change two entries in the array holding the composition.

The purpose of this note is to present an algorithm that performs this job in such a way that the labor per call, averaged over $L(n, k)$, is bounded. In Section II, the lists are described, and some of their properties are noted; in Section III, the algorithm is presented; and in Section IV, the average labor is calculated.

The following notation will be used. If A and B are lists, $A \oplus B$ will denote their concatenation, and $-A$ will denote the list obtained by reversing A. $A \otimes B$ will denote the cartesian product of A and B, ordered lexicographically. In particular, if $B = \{j\}$ is a singleton, $A \otimes \{j\}$ is the list $\{x \oplus \{j\}: x \in A\}$. Finally, if π and π' are as above, $\mathrm{i}(\pi)$ and $\mathrm{d}(\pi)$ will denote, respectively, the parts of π that must be increased and decreased to make π'.

II. Recursive Description of the Gray Codes[1]

The lists $L(n, k)$ are described by induction on k.

Basis ($k = 2$). $L(n,2)$ has for its $(j+1)$st entry the composition $(n-j, j)$, $0 \leq j \leq n$.

Induction. Assume Gray codes $L(n, k)$ for all n and fixed $k \geq 2$, with the additional property that the first composition on each list is $(n, \ldots, 0)$ and the last is $(0, \ldots, n)$. One sees easily that the lists

$$L(n, k+1) = \bigoplus_{j=0}^{n} (-1)^{j}[L(n-j, k) \otimes \{j\}]$$

inherit the Gray-code property as well as the additional property. Some other properties of these lists, easily shown by induction, are used to construct the algorithm:

(A) After a generic pass from π to π', if the first two parts of π' are positive, then $\mathrm{i}(\pi) = \mathrm{i}(\pi') \leq 2$, and $d(\pi) = d(\pi') \leq 2$.

(B) If π has positive first part, then $\mathrm{i}(\pi)$ and $\mathrm{d}(\pi)$ are both less than or equal to the position of the second positive part, if any; if π has first part equal to zero, then $\mathrm{i}(\pi)$ and $\mathrm{d}(\pi)$ are both at most one greater than the position of the first positive part.

(C) If π and π' agree in parts q through k, and if t is any integer between q and k inclusive, then the change from π to π' can be said to occur in $L(n_t, t)$, where n_t is the sum of the first t parts of π. Moreover, the move in $L(n_t, t)$ is to the successor if $(n - n_t)$ is even and to the predecessor otherwise.

(D) For $k \geq 3$, the next-to-last entry of $L(n, k)$ is the composition $(0, \ldots, 1, n-1)$ if n is odd, and it is $(1, \ldots, 0, n-1)$ if n is even.

III. The Algorithm

On a generic call, input consists of the current composition $\pi = (\pi(1), \ldots, \pi(k))$, the pointers i and d, and a pointer p indicating the first positive part of π. Output consists of the successor π' of π and updated pointers i, d, and p. The algorithm uses observations (A)–(D) above to calculate $\mathrm{i}(\pi)$ and $\mathrm{d}(\pi)$ and then changes π accordingly. One array of length k is used.

[1] These Gray codes were discovered by D. E. Knuth (unpublished answer to a question of Nijenhuis and Wilf).

Algorithm GRACOM

FIRST ENTRY

```
$\pi \leftarrow (n,0,\dots,0)$
$p \leftarrow 1$
**If** $\pi(k) = n$, **then** final return
return
```

LATER ENTRIES

```
**If** $p = 1$ **then**
  $b \leftarrow$ position of second positive part (There will be one for all later
  entries.)
  **If** $b = 2$ **then**
    **If** $d = \pi(1) = 1$ **then** $p \leftarrow 2$
  **Else if** $(n - \pi(1))$ is even **then** $(d, i, p) \leftarrow (1,2,2)$
  **Else if** $\pi(b)$ is odd **then** $(d, i, p) \leftarrow (1, b, b)$
  **Else** $(i, d) \leftarrow (1, b)$
  **End if**
**Else**
  **If** $(n - \pi(p))$ is odd **then**
    $(d, i) \leftarrow (p, p - 1)$
    **If** $\pi(p)$ is even, **then** $i \leftarrow 1$
    $p \leftarrow i$
  **Else if** $\pi(p + 1)$ is even **then**
    $(i, d) \leftarrow (p + 1, p)$
    **If** $\pi(p) = 1$ **then** $p \leftarrow p + 1$
  **Else** $(i, d) \leftarrow (p, p + 1)$
  **End if**
**End if**
$\pi(i) \leftarrow \pi(i) + 1$; $\pi(d) \leftarrow \pi(d) - 1$
 **If** $\pi(1) > 0$ **then** $p \leftarrow 1$
 **If** $\pi(k) = n$ **then** final return; return □
```

IV. Average Complexity

Since the only loop in the algorithm is the search for b (line 6), one can measure the average complexity by averaging over $L(n, k)$ the function

$$f(\pi) = \begin{cases} b - 1, & \pi(1) > 0, \\ 0, & \pi(1) = 0. \end{cases}$$

One can compute this average $\bar{f}$ as follows:

Step 1. The number of compositions of n into k parts with positive first part is

$$\binom{n+k-2}{k-1} \qquad (n, k \text{ positive}).$$

Step 2. For (n, k) positive, the average number of leading zeros of a partition of n into k parts is $(k-1)/(n+1)$ [1, pp. 28–29].

Step 3.

$$\bar{f} = \binom{n+k-1}{k-1}^{-1} \sum_{l=0}^{n-1} \binom{n-l+k-2}{k-2} \cdot \frac{k-2}{n-l+1}$$
$$= \binom{n+k-1}{k-1}^{-1} \sum_{l=1}^{n-1} \binom{n-l+k-2}{k-3}$$
$$=< \frac{k-1}{n+k-1} < 1.$$

References

1. A. Nijenhuis and H. Wilf, "Combinatorial Algorithms," 2nd ed., Academic Press, New York, 1978.