LOOP-FREE GRAY CODE ALGORITHM FOR THE e-RESTRICTED GROWTH FUNCTIONS

TOUFIK MANSOUR, GHALIB NASSAR, AND VINCENT VAJNOVSZKI

Abstract

The subject of Gray codes algorithms for the set partitions of $\{1, 2, \ldots, n\}$ had been covered in several works. The first Gray code for that set was introduced by Knuth [3], later, Ruskey presented a modified version of Knuth's algorithm with distance two, Ehrlich [5] introduced a loop-free algorithm for the set of partitions of $\{1, 2, \ldots, n\}$, Ruskey and Savage [16] generalized Ehrlich's results and give two Gray codes for the set of partitions of $\{1, 2, \ldots, n\}$, and recently, Mansour et al. [11] gave another Gray code and loop-free generating algorithm for that set by adopting plane tree techniques.

In this paper, we introduce the set of **e**-restricted growth functions (a generalization of restricted growth functions) and extend the aforementioned results by giving a Gray code with distance one for this set; and as a particular case we obtain a new Gray code for set partitions in restricted growth function representation. Our Gray code satisfies some prefix properties and can be implemented by a loop-free generating algorithm using classical techniques; such algorithms can be used as a practical solution of some difficult problems. Finally, we give some enumerative results concerning the restricted growth functions of order d.

Keywords: Gray codes, Loop-free algorithms, Partitions, **e**-restricted growth functions

2000 Mathematics Subject Classification: Primary 05A05, 94B25, Secondary: 05A15

1. introduction

A *Gray code* for a combinatorial class is a listing of its objects in which only "small change" takes place between any two consecutive objects and does not depend on the size of the objects; the "small change" is considered with respect to the Hamming distance and it depends on the particular family. A *d-Gray code* is a Gray code such that the Hamming distance between any two consecutive objects is at most d. Several authors introduced Gray codes for permutations [6, 19], involutions [22], fixed-point free involutions [22], derangements [4], permutations with a fixed number of cycles [1], and partitions of a set [5, 16, 11]. A generating algorithm which takes only a constant amount of time between consecutive objects of a combinatorial class is said to be *loop-free*. The notion of loop-free algorithms was first formulated by Ehrlich [5]. Nowadays one can find many loop-free algorithms for various combinatorial classes such as permutations [5], multiset permutations [20], set partitions [5, 11], compositions [13] and others.

A *restricted growth function* of length n is an integer sequence $\pi = \pi_1\pi_2 \cdots \pi_n$ such that $\pi_1 = 1$ and $\pi_{i+1} \leq \max\{\pi_1, \ldots, \pi_i\} + 1$, for all $1 \leq i \leq n-1$ (see for example [18]). There is a bijection between the set of restricted growth functions $\pi_1\pi_2 \cdots \pi_n$ of length n and the set of partitions of

$\{1, 2, \ldots, n\}$, namely: $\pi_1\pi_2\cdots\pi_n \mapsto B_1/B_2/\cdots/B_k$ if and only if $\pi_j = i$ implies $j \in B_i$; or, conversely, $B_1/B_2/\cdots/B_k \mapsto \pi_1\pi_2\cdots\pi_n$ if and only if $j \in B_i$ implies $\pi_j = i$. We consider a natural extension of this definition.

Definition 1. Let $\mathbf{e} = e_1e_2\ldots e_n$ be a length-n integer sequence with $e_1 = 0$ and $e_i \geq 1$ for $i \geq 2$. An $\mathbf{e}$-*restricted growth function* is a sequence $\pi = \pi_1\pi_2\ldots\pi_n$ with

- $\pi_1 = 1$, and
- $1 \leq \pi_i \leq e_i + \max\{\pi_1, \pi_2, \ldots, \pi_{i-1}\}$, for $2 \leq i \leq n$.

In particular, if there exists an integer d such that $e_2 = e_3 = \ldots = e_n = d$, then π is called *restricted growth function of order* d. Thus the standard restricted growth functions correspond to the restricted growth functions of order $d = 1$. For a given integer n and an integer sequence $\mathbf{e}$ as in Definition 1 we denote by $P_{\mathbf{e},n}$ the set of $\mathbf{e}$-restricted growth functions; and for an integer d we denote by $P_{d,n}$ the set of restricted growth function of order d; and so, the standard restricted growth function set is $P_{1,n}$, see [18].

2. Gray code for $P_{\mathbf{e},n}$

Our main goal in this section is to give a Gray code, with distance 1, for $P_{\mathbf{e},n}$. By mean of a generating algorithm we define a list, $\mathcal{L}_{\mathbf{e},n}$, for the set $P_{\mathbf{e},n}$ and we will show that the obtained list is a Gray code.

A list for a set of sequences is *prefix partitioned* if all sequences in the list having the same prefix are consecutive. Our strategy in the construction of a prefix partitioned Gray code for $P_{\mathbf{e},n}$ is the following. We assign to each position of a sequence in $P_{\mathbf{e},n}$ a status: active or inactive; and initially all positions—except the leftmost one—are active. After the initialization step, the algorithm repeatedly does on the current sequence π in $P_{\mathbf{e},n}$ the following:

- find the rightmost active position i in π;
- change appropriately the ith element in π and output π;
- if all prefixes of the form $\pi_1\pi_2\ldots\pi_{i-1}x$ have been obtained, then set position i inactive;
- set all positions at the right of i active.

For a given prefix $\pi_1\pi_2\ldots\pi_{i-1}$ the algorithm above sketched will exhaust all possible values for $\pi_i \in \{1, 2, \ldots, m\}$, with $m = e_i + \max\{\pi_1, \pi_2, \ldots, \pi_{i-1}\}$ in an appropriate order. Now we define two such orders on the set $\{1, 2, \ldots, m\}$ depending on a parameter $f \in \{1, 2\}$, called *direction*. For an integer $m \geq 2$ let define the ordering $\mathrm{succ}_{f,m}$ on the set $\{1, 2, \ldots, m\}$ by

$$\mathrm{succ}_{f,m}(x) = \begin{cases} m, & \text{if } x = f \text{ and } (m > 2 \text{ or } f = 1); \\ x - 1, & \text{if } x \neq f \text{ and } x - 1 \neq f \text{ and } x > 2; \\ 1, & \text{if } f = 2 \text{ and } (m = 2 \text{ or } x = 3). \end{cases} \tag{1}$$

For example, the successive elements of the set $\{1, 2, \ldots, m\}$ are

- listed in $\mathrm{succ}_{1,m}$ order: $1, m, m-1, \ldots, 2$, and
- listed in $\mathrm{succ}_{2,m}$ order: $2, m, m-1, \ldots, 3, 1$.

The implementation of the above algorithm needs three auxiliary array: $f = f_1f_2\ldots f_n$, $m = m_1m_2\ldots m_n$ and $a = a_1a_2\ldots a_n$; the meaning of them is given below.

- f_i is the direction of the next change of π_i. Initially $f_i = 1$ for all i,
- m_i is the largest value of π_i considering the prefix $\pi_1\pi_2\dots\pi_{i-1}$ fixed; that is $m_i = e_i + \max\{\pi_1, \pi_2, \dots, \pi_{i-1}\}$. Initially $m_i = e_i + 1$ for all i.
- a_i is 0 or 1 according as i is an active position or not in π. Initially $a_i = 1$ for all i, except $a_1 = 0$.

Let denote by $\mathcal{L}_{\mathbf{e},n}$ the list produced by the previous algorithm. Now we give a more formal expression of this algorithm, which after the initialization stage of the auxiliary arrays as above and of π by $11\dots1$ performs

```
output π
while not all a_i are zeros do
        NEXT
        output π
enddo
```

The procedure NEXT is given below and computes the successor of a sequence π in $P_{\mathbf{e},n}$ and updates arrays a, m and f.

```
global array: π, a, f, m, e
procedure NEXT
local: i, j
i := max_{1≤j≤n}{j | a_j = 1} /* i is the rightmost active position in π*/
π_i := succ_{f_i, m_i+e_i}(π_i)
if π_i = 1 and f_i = 2 or π_i = 2 and f_i = 1 /* π_i is the last value in its direction */
then a_i := 0 /* set position i inactive */
        f_i := π_i /* change the direction of π_i */
endif
for j from i + 1 to n do
        a_j := 1 /* set active all positions at the right of i */
        m_j := max(m_{i−1}, π_i)
enddo
end procedure
```

Because of the research of the largest i with $a_i = 1$ and of the inner loop **for** this generating algorithm is not efficient in general. At the end of this section we will explain how using general known techniques it can be implemented by a loop-free algorithm, and so efficiently.

A sequence $\pi' = \pi_1\pi_2\dots\pi_j$, $1 \le j < n$, is an *admissible proper prefix* for $P_{\mathbf{e},n}$ if there is (at least) a sequence in $P_{\mathbf{e},n}$ with the prefix π'. For a given admissible proper prefix π' our algorithm produces sequences with prefix $\pi'x$ for all $x \in \{1, 2, \dots, e_i + \max\{\pi_1, \pi_2, \dots, \pi_{i-1}\}\}$. Iteratively applying this fact we have that the list $\mathcal{L}_{\mathbf{e},n}$ defined by the previous algorithm is an exhaustive list for the set $P_{\mathbf{e},n}$. In addition, since a single element is changed in the current sequence (by the procedure NEXT) in order to obtain its successor, we have

Proposition 2. *The list $\mathcal{L}_{\mathbf{e},n}$ is a 1-Gray code for the set $P_{\mathbf{e},n}$, that is, two consecutive sequences in $\mathcal{L}_{\mathbf{e},n}$ differ in exactly one position.*

By construction first$(\mathcal{L}_{\mathbf{e},n}) = 1111\dots1$, and if $\ell_1\ell_2\dots\ell_n = \text{last}(\mathcal{L}_{\mathbf{e},n})$, then $\ell_1 = 1$, and $\ell_i \in \{1, 2\}$ for $i \ge 2$. For example:

- for $\mathbf{e} = 02322$, $\text{last}(\mathcal{L}_{\mathbf{e},5}) = 12221$;
- for $\mathbf{e} = 01111$, $\text{last}(\mathcal{L}_{\mathbf{e},5}) = \text{last}(\mathcal{L}_{1,5}) = 12121$, see Table 2;
- for $\mathbf{e} = 03333$, $\text{last}(\mathcal{L}_{\mathbf{e},5}) = 12111$.

T. Walsh gave in [23] a general generating algorithm for Gray code lists $\mathcal{L}$ satisfying the following two properties:

- sequences with the same prefix are consecutive (that is, the list is *prefix partitioned*);
- for each proper prefix $\pi_1\pi_2\cdots\pi_i$ of a sequence in $\mathcal{L}$ there are at least two values a and b such that $\pi_1\pi_2\cdots\pi_i a$ and $\pi_1\pi_2\cdots\pi_i b$ are both prefixes of sequences in $\mathcal{L}$.

Our Gray code list $\mathcal{L}_{\mathbf{e},n}$ satisfies Walsh's previous desiderata and so it can be generated by a loop-free algorithm by applying his general method. See also [21] where is given a general technique for the loop-free generation of particular subsets of the product space. Alternatively, a loop-free implementation can be obtained by using the *finished and unfinished lists* method, introduced in [14].

1	1 1 1 1	*11*	1 3 2 3	*21*	1 3 3 4	*31*	1 2 3 2
2	1 1 1 3	*12*	1 3 2 2	*22*	1 3 3 3	*32*	1 2 3 5
3	1 1 1 2	*13*	1 3 4 2	*23*	1 3 3 2	*33*	1 2 3 4
4	1 1 2 2	*14*	1 3 4 6	*24*	1 3 1 2	*34*	1 2 3 3
5	1 1 2 4	*15*	1 3 4 5	*25*	1 3 1 3	*35*	1 2 3 1
6	1 1 2 3	*16*	1 3 4 4	*26*	1 3 1 1	*36*	1 2 2 1
7	1 1 2 1	*17*	1 3 4 3	*27*	1 2 1 1	*37*	1 2 2 4
8	1 3 2 1	*18*	1 3 4 1	*28*	1 2 1 4	*38*	1 2 2 3
9	1 3 2 5	*19*	1 3 3 1	*29*	1 2 1 3	*39*	1 2 2 2
10	1 3 2 4	*20*	1 3 3 5	*30*	1 2 1 2		

TABLE 1. The 39 sequences in the list $\mathcal{L}_{\mathbf{e},4}$ with $\mathbf{e} = 0212$.

1	1 1 1 1 1	*13*	1 1 2 1 1	*25*	1 2 3 2 1	*37*	1 2 3 3 2
2	1 1 1 1 2	*14*	1 1 2 1 2	*26*	1 2 3 2 4	*38*	1 2 3 1 2
3	1 1 1 2 2	*15*	1 2 2 1 2	*27*	1 2 3 2 3	*39*	1 2 3 1 3
4	1 1 1 2 3	*16*	1 2 2 1 3	*28*	1 2 3 2 2	*40*	1 2 3 1 1
5	1 1 1 2 1	*17*	1 2 2 1 1	*29*	1 2 3 4 2	*41*	1 2 1 1 1
6	1 1 2 2 1	*18*	1 2 2 3 1	*30*	1 2 3 4 5	*42*	1 2 1 1 2
7	1 1 2 2 3	*19*	1 2 2 3 4	*31*	1 2 3 4 4	*43*	1 2 1 2 2
8	1 1 2 2 2	*20*	1 2 2 3 3	*32*	1 2 3 4 3	*44*	1 2 1 2 3
9	1 1 2 3 2	*21*	1 2 2 3 2	*33*	1 2 3 4 1	*45*	1 2 1 2 1
10	1 1 2 3 4	*22*	1 2 2 2 2	*34*	1 2 3 3 1		
11	1 1 2 3 3	*23*	1 2 2 2 3	*35*	1 2 3 3 4		
12	1 1 2 3 1	*24*	1 2 2 2 1	*36*	1 2 3 3 3		

TABLE 2. The 45 restricted growth functions of length 5 in $\mathcal{L}_{1,5}$.

3. Enumeration restricted growth function of order d

Let $p_{n,d,k}$ be the number of restricted growth functions $\pi = \pi_1\pi_2\cdots\pi_n$ of order d of length n such that $\max_{i\in\{1,2,\dots,n\}} \pi_i = k$. We define $P_{d,k}(x) = \sum_{n\geq 0} p_{n,d,k}x^n$ to be the generating function for the sequence $p_{n,d,k}$ according to the first parameter. Since each restricted growth function π of order d with $\max_{i\in\{1,2,\dots,n\}} \pi_i = k$ can be decomposed as $\pi = \pi' k \pi''$, where π'' is any sequence of integers in $\{1, 2, \dots, k\}$ and π' is a restricted growth function of order d such that the largest entry in π' is in the set $\{k-d, k-d+1, \dots, k-1\}$. Hence, the generating function $P_{d,k}(x)$ satisfies the recurrence relation

$$P_{d,k}(x) = \frac{x}{1-kx}\left(P_{d,k-1}(x) + P_{d,k-2}(x) + \cdots + P_{d,k-d}(x)\right) \tag{2}$$

with the initial conditions $P_{d,k}(x) = 0$ for all $k < 1$ and $P_{d,1}(x) = \frac{x}{1-x}$.

Theorem 3. *The generating function $P_{d,k}(x)$ is given by*

$$\sum_{\substack{1=i_1<i_2<\cdots<i_s=k,\\ i_j-i_{j-1}\leq d,\ j=2,3,\dots,s}} \frac{x^s}{(1-i_1x)\cdots(1-i_sx)}.$$

Proof. We proceed the proof by induction on k. Clearly, the theorem holds for $k < 0$ and $k = 1$. If we assume that the theorem holds $k < \ell$, then by (2) we obtain

$$\begin{aligned}
P_{d,k}(x) &= \frac{x}{1-kx}\sum_{j=k-d}^{k-1}\left(\sum_{\substack{1=i_1<i_2<\cdots<i_s=\ell,\\ i_\ell-i_{\ell-1}\leq d,\ \ell=2,3,\dots,s}} \frac{x^s}{(1-i_1x)\cdots(1-i_sx)}\right)\\
&= \sum_{\substack{1=i_1<i_2<\cdots<i_{s+1}=k,\\ i_\ell-i_{\ell-1}\leq d,\ \ell=2,3,\dots,s+1}} \frac{x^{s+1}}{(1-i_1x)\cdots(1-i_{s+1}x)}\\
&= \sum_{\substack{1=i_1<i_2<\cdots<i_s=k,\\ i_\ell-i_{\ell-1}\leq d,\ \ell=2,3,\dots,s}} \frac{x^s}{(1-i_1x)\cdots(1-i_sx)},
\end{aligned}$$

as claimed. □

As a corollary of the above theorem we obtain that the generating function for the number of restricted growth functions of order d and of length n is given by

$$P_d(x) = 1 + \sum_{k\geq 1}\left(\sum_{\substack{1=i_1<i_2<\cdots<i_s=k,\\ i_\ell-i_{\ell-1}\leq d,\ \ell=2,3,\dots,s}} \frac{x^s}{(1-i_1x)\cdots(1-i_sx)}\right).$$

For instance, if $d = 1$ then

$$P_1(x) = 1 + \sum_{k\geq 1}\frac{x^k}{(1-x)\cdots(1-kx)},$$

which is the ordinary generating function for the number of set partitions of $\{1, 2, \dots, n\}$, see [18].

References

[1] J.-L. Baril, Gray code for permutations with a fixed number of cycles, *Disc. Math.* **307:13** (2007) 1559–1571.

[2] J.R. Bitner, G. Ehrlich and E.M. Reingold, Efficient generation of the binary reflected Gray code and its applications, *Commun. ACM*, 19(9):517–521, 1976.

[3] D.E. Knuth, The art of computer programming, Vol. 1, Fundamental algorithms, Addison-Wesley, Reading, Mass.-London-Amsterdam, 1975.

[4] J.-L. Baril and V. Vajnovszki, Gray code for derangements, *Disc. App. Math.* **140** (2004) 207–221.

[5] G. Ehrlich, Loopless algorithms for generating permutations, combinations, and other combinatorial configurations, *J. Assoc. Comput. Mach.* **20** (1973) 500–513.

[6] S.M. Johson, Generating of permutations by adjacent transposition, *Math. Comput.* **17** (1963) 282–285.

[7] J.M. Lucas, D. Roelants van Baronnaigien and F. Ruskey, On rotations and the generation of binary trees, *J. Algorithms* **15** (1993) 343–366.

[8] M. Klazar, On abab-free and abba-free set partitions, *Europ. J. Combin.* **17** (1996) 53–68.

[9] C.W. Ko and F. Ruskey, Generating permutations of a bag by interchanges, *Inform. Processing Lett.* **41** (1992) 263–269.

[10] J.F. Korsh and S. Lipschutz, Generating multiset permutations in constant time, *J. Algorithms* **25** (1997) 321–335.

[11] T. Mansour and G. Nassar, Gray codes, loopless algorithm and partitions, *J. Math. Model Algor.* **7.3** (2008) 291–310.

[12] T. Mansour and G. Nassar, Up-staircase words, generating and enumeration, *to appear*.

[13] T. Mansour and G. Nassar, Loop-free Gray code algorithms for the set of compositions, *J. Math. Model Algor.* **9** 2010 343–356.

[14] J.M. Lucas, D.R. van Baronaigien and F. Ruskey, On rotations and the generation of binary trees, *J. Algorithms* **15** (1993) 1–24.

[15] F. Ruskey, Combinatorial generation, see http://www.cs.sunysb.edu/ algorith/implement/ruskey/implement.shtml.

[16] F. Ruskey and C.D. Savage, Gray codes for set partitions and restricted growth tails, *Aust. J. Combin.* **10** (1994) 85–96.

[17] R. Stanley, Enumerative combinatorics, Vol. 1, Cambridge University Press, Cambridge, England, 1997.

[18] D. Stanton and D. White, *Constructive Combinatorics*, Springer, 1986.

[19] H.F. Trotter, Algorithm 115, permutations, *Comm. ACM* **5** (1962) 434–435.

[20] V. Vajnovszki, A loopless algorithm for generating the permutations of a multiset, *Theoret. Comput. Sci.* **307** (2003) 415–431.

[21] V. Vajnovszki, On the loopless generation of binary tree sequences, *Inform. Processing Lett.* **68** (1998) 113–117.

[22] T. Walsh, Gray codes for involutions, *J. Combin. Math. Combin. Comput.* **36** (2001) 95–118.

[23] T. Walsh, Generating Gray codes in $O(1)$ worst-case time per word, *4th Discrete Mathematics and Theoretical Computer Science Conference*, Dijon-France, 7-12 July 2003 (LNCS 2731, 71–88).

Department of Mathematics, University of Haifa, 31905 Haifa, Israel

E-mail address: toufik@math.haifa.ac.il

Department of Mathematics, University of Haifa, 31905 Haifa, Israel

E-mail address: nassar_ghalib@hotmail.com

LE2I, Université de Bourgogne, BP 47870, 21078 Dijon Cedex, France

E-mail address: vvajnov@u-bourgogne.fr