Efficient generation of restricted growth words

Toufik Mansour
Department of Mathematics, University of Haifa
31905 Haifa, Israel
tmansour@univ.haifa.ac.il

Vincent Vajnovszki
LE2I, Université de Bourgogne
BP 47870, 21078 Dijon Cedex, France
vvajnov@u-bourgogne.fr

May 30, 2013

Abstract

A length n *restricted growth word* is a word $w = w_1 w_2 \dots w_n$ over the set of integers where $w_1 = 0$ and each w_i, $i > 1$, lies between 0 and the value of a word statistics of the prefix $w_1 w_2 \dots w_{i-1}$ of w, plus one. Restricted growth words simultaneously generalize combinatorial objects as restricted growth functions, staircase words and ascent or binary sequences. Here we give a generic generating algorithm for restricted growth words. It produces a Gray code and runs in constant average time provided that the corresponding statistics has some local properties.

Keywords: Algorithms, Restricted growth functions/words, Gray codes.

1 Introduction

A list of words is a *Gray code* if successive words in the list differ in a 'small prespecified way' [12]. Here we adhere to the following (quite restrictive) definition: a list of words is a Gray code if the words are listed so that two successive words differ in a single position and by a bounded amount in this position.

In this paper we give a generating algorithm producing a Gray code for words over the set of integers, and satisfying the following constraint: the ith symbol of each word is upper bounded by the value of a statistics of its length $i-1$ prefix, plus one.

A *statistics* on a set of words is an association of an integer to each word in the set. Classical examples of statistics are defined as: if $w = w_1 w_2 \dots w_n$, then

$$\mathsf{m}(w) = \max\{w_1, w_2, \dots, w_n\},$$

$$\mathsf{last}(w) = w_n,$$

$$\mathsf{asc}(w) = \mathrm{card}\{i \,|\, 1 < i \le n \text{ and } w_{i-1} < w_i\},$$

$$\mathsf{len}(w) = n - 1,$$

$$\mathsf{bin}(w) = 0,$$

and we will consider only statistics st satisfying $\mathsf{st}(w_1 w_2 \dots w_n) \le n - 1$, which agrees with most of naturally defined statistics.

Definition 1. Let st be a statistics on words. An st-*restricted growth word* is a word $w = w_1 w_2 \dots w_n$ with $w_1 = 0$, and

$$w_i \leq \mathsf{st}(w_1 w_2 \dots w_{i-1}) + 1, \text{ for } 1 < i \leq n. \quad (1)$$

For particular cases, known sets of words are obtained, and they code various classes of combinatorial objects. For example, the set of

- m-restricted growth words is the set of *restricted growth functions* [10, 14],
- last-restricted growth words is the set of *staircase words* [13, exercise u, p. 222],
- asc-restricted growth words is the set of *ascent sequences* [3, 8],
- len-restricted growth words is the set of *subexcedant sequences* [4, 16],
- bin-restricted growth words is the set of binary words over $\{0, 1\}$, and beginning with a zero.

Notice that, if $w = w_1 w_2 \dots w_n$ is an st-restricted growth word, then so is any of its prefixes, and any word $w_1 w_2 \dots w_n a_{n+1} a_{n+2} \dots a_m$ with $a_i \in \{0, 1\}$, $n + 1 \leq i \leq m$, is still an st-restricted growth word. Also, with the restriction on statistics imposed before Definition 1 it follows that $0 \leq w_i \leq i - 1$, for $1 \leq i \leq n$.

Definition 2. A statistics st is called *local* if the value of $\mathsf{st}(w)$ of any length $n \geq 2$ st-restricted growth word $w = w_1 w_2 \dots w_n$ can be computed in constant time from $\mathsf{st}(w_1 w_2 \dots w_{n-1})$ and $w_1 w_2 \dots w_n$.

Example 1. The statistics m, last, asc, len and bin are local. Indeed, for $n \geq 2$ we have:

- $\mathsf{m}(w_1 w_2 \dots w_n) = \max\{\mathsf{m}(w_1 w_2 \dots w_{n-1}), w_n\}$,
- $\mathsf{last}(w_1 w_2 \dots w_n) = w_n$,
- $\mathsf{asc}(w_1 w_2 \dots w_n) = \begin{cases} \mathsf{asc}(w_1 w_2 \dots w_{n-1}) & \text{if} \quad w_{n-1} \geq w_n, \\ \mathsf{asc}(w_1 w_2 \dots w_{n-1}) + 1 & \text{if} \quad w_{n-1} < w_n, \end{cases}$
- $\mathsf{len}(w_1 w_2 \dots w_n) = \mathsf{len}(w_1 w_2 \dots w_{n-1}) + 1$.

By contrast, the following statistics is not local:

$$\mathsf{st}(w_1 w_2 \dots w_n) = \text{card}\{i \,|\, 1 \leq i < n, \text{ and } w_i < w_n\}.$$

This statistics counts the number of occurrences of the *vincular pattern* $\underline{0\,1}]$ (defined in [1]), and is not local. Indeed, in general, the value of $\mathsf{st}(w_1 w_2 \dots w_n)$, $n \geq 2$, can not be computed in constant time from $\mathsf{st}(w_1 w_2 \dots w_{n-1})$ and $w_1 w_2 \dots w_n$.

1	00000	15	00101	29	01211	43	01000
2	00001	16	01101	30	01231	44	01002
3	00011	17	01102	31	01233	45	01001
4	00012	18	01100	32	01234	46	01021
5	00010	19	01120	33	01232	47	01023
6	00110	20	01122	34	01230	48	01022
7	00112	21	01123	35	01220	49	01020
8	00111	22	01121	36	01222	50	01010
9	00121	23	01111	37	01223	51	01012
10	00123	24	01112	38	01221	52	01013
11	00122	25	01110	39	01201	53	01011
12	00120	26	01210	40	01203		
13	00100	27	01212	41	01202		
14	00102	28	01213	42	01200		

Table 1: The ascent sequences (corresponding to the statistics asc) of length 5 generated by procedure `GenRGW`; the positions where two successive words differ are underlined.

2 Generating algorithm

Definition 3. For an integer m we define two (ordered) lists $\mathcal{L}(m)$ and $\overline{\mathcal{L}}(m)$:

- $\mathcal{L}(m)$ is the list of even numbers in the set $\{0, 1, \ldots, m\}$ in increasing order, followed by the list of odd numbers in the same set, in decreasing order;
- $\overline{\mathcal{L}}(m)$ is the reverse of $\mathcal{L}(m)$, that is, the list of odd numbers in the set $\{0, 1, \ldots, m\}$ in increasing order, followed by the list of even numbers in the same set, in decreasing order.

For example, $\mathcal{L}(1) = 0, 1$; $\overline{\mathcal{L}}(2) = 1, 2, 0$; $\mathcal{L}(5) = 0, 2, 4, 5, 3, 1$; and $\mathcal{L}(6) = 0, 2, 4, 6, 5, 3, 1$; and these lists will be used in our generating algorithm. The main idea is that $\mathcal{L}$ and $\overline{\mathcal{L}}$ list the sets under consideration by imposing their first and last elements, namely 0 and 1, and successive elements differ by at most two. Similar techniques appear in an ECO-based context in [2] where the first and last value for each entry to update are imposed; and in [9] where concatenation of lists and reversed lists is used.

Now we explain the generating algorithm `GenRGW` given in Figure 1(a). In each recursive call of `GenRGW`, $w = w_1 w_2 \ldots w_n$ is a global variable, and we say that `GenRGW` *acts* on w; and the main call `GenRGW(2,0)` acts on $00 \ldots 0$. The first parameter, k, of `GenRGW` is the position in w updated by the current call; and the second one, u, gives the value of the statistics st for $w_1 w_2 \ldots w_{k-1}$. The call `GenRGW(`k`,`u`)`, $2 \leq k \leq n$, acting on the current word $w_1 w_2 \ldots w_n$

- exhausts all possible values for w_k (with respect to the prefix $w_1 w_2 \ldots w_{k-1}$), in $\mathcal{L}$ or $\overline{\mathcal{L}}$ order, and
- prints w if $k = n$, or calls recursively `GenRGW(`k`+1,`v`)` for each of these values, where $v = \mathsf{st}(w_1 w_2 \ldots w_k)$, if $k < n$.

Procedure `GenRGW` induces a generating tree covered in depth first way; see Figure 1(b) where the length four `m`-restricted growth words are at the last level (rightmost one) of the generating tree. Theorem 1 states that it generates exhaustively the set of `st`-restricted growth words of length n and Proposition 1 that this is done in a Gray code manner. Moreover, if `st` is a local statistics, then `GenRGW` is efficient.

Theorem 1. *Algorithm* `GenRGW` *produces exhaustively length* n `st`*-restricted growth words.*

Proof. We will show that `GenRGW` produces, with no omissions nor repetitions `st`-restricted growth words.

Let us consider the generating tree induced by `GenRGW` where the root call is `GenRGW(2,0)`. In this tree, a recursive call with its first parameter k and acting on $w_1w_2\dots w_n$ produces only calls acting on words with fixed length $k-1$ prefix equal to $w_1w_2\dots w_{k-1}$. More precisely, for each $x \in \{0, 1, \dots, \mathtt{st}(w_1w_2\dots w_{k-1}) + 1\}$, it produces in the `for` loop a call acting on a word beginning with $w_1w_2\dots w_{k-1}x$. Since the main call `GenRGW(2,0)` produces directly two recursive calls acting on words with prefix 00 and 01 respectively (the only two `st`-restricted growth words of length two), it follows by induction that for each $k \leq n$ and each `st`-restricted growth word $w_1w_2\dots w_k$, all words with the prefix $w_1w_2\dots w_k$ are produced, and in particular the whole list of length n words.

In addition, if s and t are two words printed by terminal calls, then either s and t are generated by the same terminal call, and so they differ in the last position, or there is a $k < n$ such that s and t are produced by two different calls produced in turn by the same call with the first parameter equal to k, and so s and t differ in position k. □

A *prefix partitioned list* is a list of words where words with a given prefix are contiguous. For a given prefix, procedure `GenRGW` exhausts all restricted words with this prefix, and so it generates prefix partitioned list. In addition, the set of extremal values (first and last value) in the list $\mathcal{M}$ equals $\{0, 1\}$, and thus, the first and the last word with a given prefix $w_1w_2\dots w_k$ generated by `GenRGW` have the form $w_1w_2\dots w_k a_{k+1}a_{k+2}\dots a_n$, with $a_i \in \{0, 1\}$ for $k+1 \leq i \leq n$.

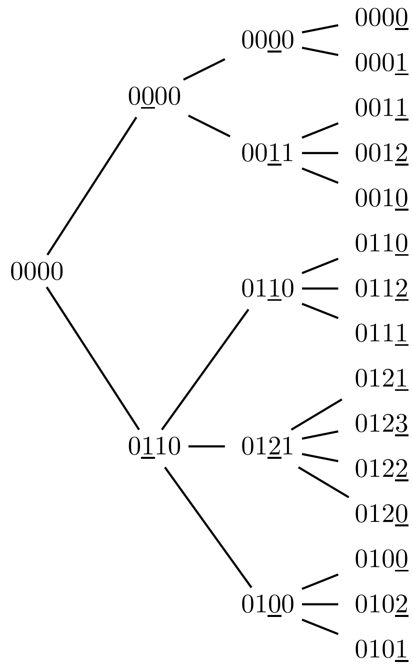
In a given call in the generating tree, the statement '$w_k := i$' is performed several times. The first of them does not change the value of w_k, which is either 0 or 1, the previous value of w_k. By induction the next proposition follows. See Table 1 for an example.

Proposition 1. *For any* n *and statistics* `st`*, procedure* `GenRGW` *generates a Gray code for the set of* `st`*-restricted growth words of length* n *where two successive words differ in a single position and by* ± 1 *or by* ± 2 *in this position.*

Moreover, if the statistics `st` is local, then procedure `GenRGW` has a constant average time complexity. Indeed, the total amount of computation done by a terminal call is proportional to the number of generated words by this call, and each non-terminal call produces at least two recursive calls. Since for any prefix $w_1w_2\dots w_k$ the value of $\mathtt{st}(w_1w_2\dots w_k)$ can be computed in constant time, by Frank Ruskey's *'CAT'* principle [11], it follows that `GenRGW` runs in constant amortized time.

```
procedure GenRGW(k,u)
global n, w;
if w_k = 0
then M := L(u+1);
else M := L̄(u+1);
end if
for i in M do
      w_k := i;
      if k = n
      then print w;
      else v := st(w_1 w_2 ... w_k);
           GenRGW(k+1,v);
      end if
      end do
end procedure.
```

(a)



(b)

Figure 1: (a) Algorithm generating st-restricted words of length n; $v = \mathsf{st}(w_1 w_2 \dots w_k)$ is computed in constant time from $w_1 w_2 \dots w_k$ and $u = \mathsf{st}(w_1 w_2 \dots w_{k-1})$. Initially $w = 00\dots0$, and the main call is `GenRGW(2,0)`. (b) The generating tree induced by the call of `GenRGW(2,0)` with $n = 4$ and the statistics st is m. At level k, the kth digit (underlined) is changed if the current call is not the first recursive call produced by its parent, and words are generated at the last level.

Pattern involvement statistics

A *pattern* over the alphabet $\{0, 1, \dots, k\}$ is a word where each letter of the alphabet occurs at least once. An occurrence of the (consecutive) pattern $a = a_1 a_2 \dots a_j$ in the word $w = w_1 w_2 \dots w_n$ is a factor $w_s w_{s+1} \dots w_{s+j-1}$ of w order isomorphic with a, that is, for any $u \in \{1, 2, \dots, j-1\}$, w_{s+u-1} and w_{s+u} are in same order relation ($>$, $<$ or $=$) as a_u and a_{u+1}. And for a pattern a, $\sharp\, a$ denotes the (pattern involvement) statistics giving the number of occurrences of this pattern. It is easy to see that

Remark 1. Each pattern involvement statistics is local.

Pattern involvement allows to re-express known statistics and formulate new ones. For example $\mathsf{asc} = \sharp\, 01$, and below we give other examples of pattern involvement-based statistics. Each of them is local, and so our algorithm can be applied in order to generate exhaustively, in Gray code order, its corresponding list of length n restricted growth words.

For example, the following (combinations of) statistics are pattern involvement based, and so local. Number of descents: $\mathsf{des} = \sharp\, 10$; of double ascents: $\mathsf{dasc} = \sharp\, 012$; of valleys: $\mathsf{val} = \sharp\, 101 + \sharp\, 201 + \sharp\, 102$; of levels: $\mathsf{lev} = \sharp\, 00$.

3 Final remarks

A possible generalization of st-restricted growth words is to replace the condition in relation (1) by one of the following conditions:

$$w_i \le \mathsf{st}(w_1 w_2 \ldots w_{i-1}) + t,$$

$$w_i \le \min\{\mathsf{st}(w_1 w_2 \ldots w_{i-1}) + 1, t\},$$
$$w_i \le \max\{\mathsf{st}(w_1 w_2 \ldots w_{i-1}) + 1, t\},$$

for a given $t > 1$. When st is m we obtain in the first case the set of e-restricted growth functions [7]; in the second case, the set of restricted growth functions coding set partitions with at most $t+1$ blocks [10]; and in the last case the set of restricted growth tails [10]. The corresponding generating algorithms are obtained simply by replacing $\mathcal{L}(u+1)$ in procedure `GenRGW` by $\mathcal{L}(u+t)$, $\mathcal{L}(\min\{u+1,t\})$ and $\mathcal{L}(\max\{u+1,t\})$, respectively, and similarly for $\overline{\mathcal{L}}(u+1)$.

In each case, our Gray code is different from the previous ones. Also, as a degenerate case, when st is constant and equal to zero for each word, then the list generated by procedure `GenRGW` is the first half of Binary Reflected Gray Code list [5] consisting on binary words with a zero in the first position; and so our Gray codes can be seen as a generalization of Binary Reflected Gray Code.

T. Walsh gave in [17] a general generating algorithm for Gray code lists satisfying the following two properties: (1) words with the same prefix are successive (that is, the list is prefix partitioned); (2) for each proper prefix $w_1 w_2 \ldots w_k$ of a word in the list there are at least two values a and b such that $w_1 w_2 \ldots w_k a$ and $w_1 w_2 \ldots w_k b$ are both prefixes of words in the list. Our Gray code lists satisfy Walsh's previous desiderata and so it can be generated by a loop-free algorithm by applying his general method. See also [15] 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 [6].

We conclude by a remark on the generating order induced by the procedure `GenRGW`. It is clear that, for a fixed length, the set of staircase words is a subset of the set of restricted growth functions which in turn is a subset of the set of ascent sequences. These relations are not preserved in terms of sublists, that is, words do not necessarily appear in same relative order in various lists. Let consider for example the three staircase words $x = 010111$, $y = 010100$ and $z = 010110$. Procedure `GenRGW` generates x before y as staircase word, but after y as restricted growth function. Similarly, x is generated before z as restricted growth function but after z as ascent sequence.

Acknowledgment

The authors would like to thank one of the anonymous referees for helpful suggestions which have improved the accuracy of this paper.

References

[1] E. Babson and E. Steingrímsson, Generalized permutation patterns and a classification of Mahonian statistics, *Sém. Lothar. Combin.*, 44:Art. B44b, 18pp. (electronic), 2000.

[2] A. Bernini, E. Grazzini, E. Pergola and R. Pinzani, A general exhaustive generation algorithm for Gray structures, *Acta Informatica*, **44**(5) (2007), 361–376.

[3] M. Bousquet-Mélou, A. Claesson, M. Dukes and S. Kitaev (2+2)-free posets, ascent sequences and pattern avoiding permutations, *J. Comb. Theory Ser. A*, **117**(7) (2010), 884-9-09.

[4] D. Foata and G.-N. Han, New permutation coding and equidistribution of set-valued statistics *Theor. Comput. Sci.* **410**(38–40) (2009), 3743–3750.

[5] F. Gray, Pulse code communication, U.S. Patent 2632058 (1953).

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

[7] T. Mansour, G. Nassar and V. Vajnovszki, Loop-free Gray code algorithm for e-restricted growth functions, *Information Processing Letters*, **111**(11) (2011), 541–544.

[8] T. Mansour and M. Shattuck, Some enumerative results related to ascent sequences, *arXiv:1207.3755.*

[9] F. Ruskey, Simple combinatorial Gray codes constructed by reversing sublists, *4*th *ISAAC–Lecture Notes in Computer Science*, **762** (1993) 201–208.

[10] F. Ruskey and C. Savage, Gray codes for set partitions and restricted growth tails, *Australian Journal of Combinatorics*, **10** 1994, 85–96.

[11] F. Ruskey, *Combinatorial generation*. Book in preparation.

[12] C. Savage, A Survey of Combinatorial Gray Codes, *SIAM Rev.* **39**(4) (1997), 605–629.

[13] R.P. Stanley, Enumerative Combinatorics, Vol. 2, *Cambridge University Press*, 1999.

[14] D. Stanton and D. White, Constructive combinatorics, *Springer-Verlag*, 1986.

[15] V. Vajnovszki, On the loopless generation of binary tree sequences, *Information Processing Letters*, **68** (1998), 113–117.

[16] V. Vajnovszki, Lehmer code transforms and Mahonian statistics on permutations, *Discrete Mathematics*, **313** (2013), 581–589.

[17] 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).