PU.M.A. Vol. **17** (2006), No. 1–2, pp. 175–182

Gray codes for necklaces and Lyndon words of arbitrary base

Mark Weston
University of Victoria, PO BOX 3055, Victoria, BC Canada
e-mail: mweston@cs.uvic.ca

and

Vincent Vajnovszki
Université de Bourgogne, 21078 Dijon Cedex, FRANCE
e-mail: vvajnov@u-bourgogne.fr

(Received: October 31, 2006)

Abstract. Recently, a Gray code for unrestricted binary necklaces and their relatives was discovered by Vajnovszki [Discrete Mathematics & Theoretical Computer Science, to appear]. The Gray code is constructed by modifying the classical FKM algorithm for generating necklaces in lexicographic order. We present a generalisation of Vajnovszki's algorithm, giving a Gray code for necklaces and their relatives over an arbitrarily-large alphabet. Each string in the resulting code differs from adjacent strings in at most three positions. To our knowledge this is the first Gray code for necklaces of arbitrary base.

Mathematics Subject Classifications (2000). 68R15, 05A99, 05C45

1 Introduction

Consider a string of beads of various colours arranged on a loop of string: such an object is typically called a necklace, and, conveniently, this corresponds to the analogous mathematical object: a *necklace* is a string of characters which is lexicographically minimal out of all of its possible rotations. If the string is also lexicographically minimal with respect to reversing it, it is a *bracelet*. Two other, more abstract, concepts are related: a *pre-necklace* is a string which is a prefix of some necklace, and a *Lyndon word* is an aperiodic necklace (that is, a necklace that is not composed of a whole number of repeated copies of some smaller string).

Throughout this paper we discuss strings over a fixed alphabet $\Sigma = \{0, 1, \ldots, \sigma - 1\}$, where $|\Sigma| = \sigma \geq 2$. Define an equivalence relation $\sim$ on Σ^n by $x \sim y$ iff there exist non-empty strings u, v such that $x = uv$ and $y = vu$, and we say that $x \leq y$ iff x is lexicographically (in dictionary order) smaller than y. We denote the reversal of a string x as x^R. The sets of necklaces, bracelets, pre-necklaces and Lyndon words of length n, can now be defined as

$$N_n \stackrel{\text{def}}{=} \{x \in \Sigma^n \mid x \leq y \text{ for all } y \text{ such that } y \sim x\}$$

$$B_n \stackrel{\text{def}}{=} \{x \in \Sigma^n \mid x \leq y \text{ and } x \leq y^R \text{ for all } y \text{ such that } y \sim x\}$$

$$P_n \stackrel{\text{def}}{=} \{x \in \Sigma^n \mid xy \in N_{(n+k)} \text{ for some } y \text{ where } |y| = k\}$$

$$L_n \stackrel{\text{def}}{=} \{x \in N_n \mid x \neq y^k \text{ for all } y \in \Sigma^* \text{ and } k \geq 2\}$$

respectively.

For example, Figure 1 shows the length 4 pre-necklaces over a 3-ary alphabet and indicates which strings are also necklaces and Lyndon words.

Output	Necklace	Lyndon Word
0000	✓	
0001	✓	✓
0002	✓	✓
0010		
0011	✓	✓
0012	✓	✓
0020		
0021	✓	✓
0022	✓	✓
0101	✓	
0102	✓	✓
0110		
0111	✓	✓
0112	✓	✓
0120		
0121	✓	✓

Output	Necklace	Lyndon Word
0122	✓	✓
0202	✓	
0210		
0211	✓	✓
0212	✓	✓
0220		
0221	✓	✓
0222	✓	✓
1111	✓	
1112	✓	✓
1121		
1122	✓	✓
1212	✓	
1221		
1222	✓	✓
2222	✓	

Figure 1: Base 3 pre-necklaces of length 4 (read down columns in order).

Generating necklaces and their relatives is a relatively well-studied problem; Fredricksen with Kessler and Maiorana [3, 4] developed the FKM algorithm for generating necklaces, pre-necklaces, and Lyndon words of any base; Duval [2] independently developed a version of the algorithm for Lyndon words. A recursive version, along with counting results and further analysis, can be found in [1, 8].

The table in Figure 1 shows the pre-necklaces in the order outputted from the FKM algorithm; note that there are strings that differ from adjacent strings by up to n positions (for example, $0222 \to 1111$). In general this is true of the output from the FKM algorithm. A k-Gray code for a set of strings X is an ordering for X such that the Hamming distance (number of positions that are different) between any two consecutive strings in the ordering is at most k. Gray codes were introduced by Gray [5] in the context of listing binary strings such that adjacent strings differ by exactly 1 bit. They have since been extensively studied and generalised to various combinatorial objects, see [11] for an excellent survey on combinatorial applications. A k-Gray code is *circular* if it also has the property that its first and last elements also differ in at most k positions. A 1-Gray code gives a Hamilton path in the appropriate adjacency graph, and a circular 1-Gray code gives a Hamilton circuit.

Binary strings of a fixed length and fixed number of 1s are said to have *fixed density*. There have previously been published two algorithms for generating

Gray codes for binary necklaces of fixed density; the first is due to Ueda [15], which also considers binary Lyndon words, and the second is from Savage and Wang [19]. These two Gray codes are both optimal in the sense that they are 2-Gray codes, and since there can be no 1-Gray code for any set of fixed density binary strings this is the best that can be done.

For necklaces and their relatives of unrestricted density (*i.e.* considering the entire set N_n, etc) over a binary alphabet, the first Gray code was due to Vajnovszki [17]. This Gray code is based on modifying the recursive FKM algorithm [3, 4, 8] to output strings in a 3-Gray code order; this result provides the first Gray code of any type for binary necklaces, pre-necklaces and Lyndon words. At this time it is still unknown whether this is optimal, or whether a 2-Gray code for binary necklaces and their relatives can be constructed.

The existence of a 3-Gray code for these objects is, in fact, not surprising in itself. Consider the graph constructed where nodes are elements of the sets under consideration, and nodes are adjacent where they differ by one character in one position. Sekanina [14] showed that if such a graph is connected, then some ordering where adjacent elements differ in at most 3 places is always possible; see also [7, p. 32]. The difficulty lies in the construction of such an ordering.

The contribution of this paper is to generalise the result of [17] to the general case of arbitrary base. We give a 3-Gray code listing for necklaces and their relatives of length n over the alphabet $\Sigma = \{0, 1, \ldots, \sigma - 1\}$. To our knowledge it is the first example of a Gray code for arbitrary base necklaces.

2 Preliminaries

Let x, $y \in \Sigma^n$ be two strings of length n. We define a total order relation, $\prec$, called the *local reflected order*, which is a natural generalisation of the binary local reflected order from [16].

DEFINITION 1 ($\prec$) *Let k be the leftmost position in which x and y differ, and let m be the number of characters that are not $\sigma - 1$ in the length $k - 1$ prefix of x. $x \prec y$ if either m is even and $x_k < y_k$ or m is odd and $x_k > y_k$; otherwise $y \prec x$.*

Note that this ordering has the property that strings with a common prefix are contiguous in the ordering; orderings with this property are called *genlex* [18]. For the binary case the local reflected order $\prec$ induces a 1-Gray code on $\{0, 1\}^n$, the set of all binary strings of length n, and a 2-Gray code on the strings in $\{0, 1\}^n$ with fixed density (a fixed number of non-zero bits) [17].

DEFINITION 2 ([17]) *A string set $X \subset \Sigma^n$ is called* absorbent *if for any $x = x_1 x_2 \ldots x_n \in X$ and any $k \leq n$, $x_1 x_2 \ldots x_k (\sigma - 1)^{n-k}$ is also a string in X.*

The following Lemma is proved in [17] for the binary case, but the proof also applies for the σ-ary case, substituting "$\sigma - 1$" for "1" in the proof in [17].

LEMMA 1 ([17]) *The sets of σ-ary strings, pre-necklaces, necklaces, bracelets, and Lyndon words of length n are all absorbent.*

3 The 3-Gray code for σ-ary alphabets

In this section we show that the local reflected order yields a 3-Gray code on the sets of pre-necklaces, necklaces and Lyndon words of length n over an arbitrary alphabet. This presentation very much follows [17] in that Lemma 2, Corollary 1, and Theorem 1 are virtually identical, except extended to the σ-ary case.

Let $X \subset \Sigma^n$, $x \in X$ and let k be the leftmost position where x differs from its predecessor in $\prec$ order (and x is not the initial string in the order). Since $\prec$ is a genlex ordering, $x_1x_2 \dots x_k$ is the shortest prefix of x such that for all $y \prec x$ we have $y_1y_2 \dots y_k \prec x_1x_2 \dots x_k$. We call $x_{k+1}x_{k+2} \dots x_n$ the *first discriminant suffix* of x. Similarly the *last discriminant suffix* of x, $x_{k+1}x_{k+2} \dots x_n$, is the $n-k$ suffix of x where k is the leftmost position where x differs from its successor in $\prec$ order (assuming x is not the final string in the order).

Let $|y|_{\overline{\sigma-1}}$ denote the number of characters that are not $\sigma - 1$ in the string y.

LEMMA 2 *If x is a string in an absorbent subset $X \subset \Sigma^n$ and $x_{k+1}x_{k+2} \dots x_n$ is the first or the last discriminant suffix of x, then $|x_{k+1}x_{k+2} \dots x_n|_{\overline{\sigma-1}} \leq 1$.*

Proof. We derive a contradiction by supposing that $|x_{k+1}x_{k+2} \dots x_n|_{\overline{\sigma-1}} > 1$, where $x_{k+1}x_{k+2} \dots x_n$ is the first discriminant suffix of x. Then there are at least two indices i and j, with $k+1 \leq i < j = n$, such that $x_i \neq \sigma - 1$ and $x_j \neq \sigma - 1$ and $x_{i+1} = x_{i+2} = \dots = x_{j-1} = \sigma - 1$. Since X is absorbent, $x' = x_1x_2 \dots x_{i-1}(\sigma-1)^{n-i+1}$ and $x'' = x_1x_2 \dots x_{j-1}(\sigma-1)^{n-j+1}$ are both in X. By Definition 1, either $x' \prec x$ or $x'' \prec x$, since the number of non-$(\sigma-1)$ characters of the relevant prefixes of x' and x'' must differ in parity. Thus since X ordered by $\prec$ is prefix partitioned (it is a genlex ordering), the leftmost position where x differs from its predecessor is greater than k, and so $x_{k+1}x_{k+2} \dots x_n$ is not the first discriminant suffix of x, which contradicts the assumption that it was. The case where $x_{k+1}x_{k+2} \dots x_n$ is the last discriminant suffix is treated similarly. □

COROLLARY 1 *If X is an absorbent subset of Σ^n, then X listed in $\prec$ order is a 3-Gray code.*

Proof. Let y be the successor of x in X with respect to $\prec$ order and let k be the leftmost position where x differs from y. Then $x_{k+1}x_{k+2} \dots x_n$ is the last discriminant suffix of x, and $y_{k+1}y_{k+2} \dots y_n$ is the first discriminant suffix of y. By Lemma 2, $x_i = y_i = \sigma - 1$ for all $k+1 \leq i \leq n$ except at most two of them. Since $x_i = y_i$ for all $1 \leq i \leq k-1$, and $x_k \neq y_k$, the Hamming distance between x and y is at most three. □

The main theorem now follows directly from Corollary 1 and Lemma 1.

THEOREM 1 *The $\prec$ order gives a 3-Gray code on the sets of σ-ary pre-necklaces, necklaces, and Lyndon words of length n.*

Note that the first element in these lists is $0(\sigma-1)^{n-1}$, and the last is $(\sigma-1)(\sigma-1)(\sigma-1)^{n-1}$ for Lyndon words and $(\sigma-1)^n$ for necklaces and pre-necklaces, and thus the Gray codes produced are circular, as the first and last elements of the lists differ in at most two positions.

4 Algorithm

In this section we give a modification of Vajnovszki's algorithm [17], which itself is a modification of the recursive FKM algorithm [3, 4, 8]. Like the algorithms in [17] and [3, 4, 8], it generates pre-necklaces, necklaces, and Lyndon words, according to the behaviour of the function `Print(`p`)`: if `Print` outputs the array a at every call, it produces the pre-necklaces, if `Print` outputs only when $p = n$ it produces the Lyndon words, and if `Print` outputs only if p is a divisor of n the necklaces are produced.

In the algorithm the parameter z keeps track of the number of $(\sigma-1)$s in the prefix of the string up to position t. The recursive FKM algorithm is modified by changing the direction of the for loop, depending on the parity of z.

The initial call is `gen(`$0, 1, 1$`)` with the initialisation $a[0] \leftarrow 0$. See Figure 2 for the pseudocode.

In [9] it is shown that the recursive implementation of the FKM algorithm works in constant amortised time per string outputted, which is as fast as could be expected, and thus our algorithm does as well.

5 Final remarks

Vajnovszki points out that the binary version of this Gray code can be easily extended to produce bracelets [17] and, though we have not yet verified it, we suspect that the general (σ-ary) version can also be used to produce a 3-Gray code for bracelets.

Experimental analysis of our algorithm suggests that the proportion of 1-, 2-, and 3-changes between successive words generated is very different: for example there tend to be very few (less than 1%) of 3-changes, and the ratio of 1-changes to 2-changes increases as n increases. Further analysis of the algorithm would shed light on the causes of these phenomena.

An open question is still whether there exists any 2-Gray code for necklaces and their relatives; as evidenced by the earlier work on fixed-density necklaces [19, 15] this seems to be a harder question.

There has been much recent work in generating other variations of necklaces such as unlabelled necklaces (necklaces for strings equivalent under rotation and symbol permutation) [1], necklaces with forbidden substrings [10], plane trees [13, 6] and bracelets [12]. We suspect that Gray codes can be found using techniques similar to ours for many of these objects.

```
PROCEDURE gen( z, t, p :  integer );
    local j :  integer;

    if ( t > n ) then Print(p);
    else
        if ( a[t − p] = σ − 1 ) then
            a[t] ← a[t − p];
            gen( z, t + 1, p );
        else
            if ( z is even )
                for j ← a[t − p] to σ − 1 do
                    a[t] ← j;
                    if ( j ≠ σ − 1 ) then z ← z + 1 end if;
                    if ( j ≠ a[t − p] ) then p ← t end if;
                    gen( z, t + 1, p );
                end for;
            else
                for j ← σ − 1 downto a[t − p] do
                    a[t] ← j;
                    if ( j ≠ σ − 1 ) then z ← z + 1 end if;
                    if ( j ≠ a[t − p] ) then p ← t end if;
                    gen( z, t + 1, p );
                end for;
            end if;
        end if;
    end if;

end PROCEDURE;
```

Figure 2: Modified recursive FKM algorithm to generate 3-Gray code for σ-ary necklaces, pre-necklaces, and Lyndon words.

Acknowledgements. The authors would like to thank Frank Ruskey for suggesting the problem and providing helpful suggestions.

References

[1] K. Cattell, F. Ruskey, J. Sawada, M. Serra and C.R. Miers, *Fast algorithms to generate necklaces, unlabeled necklaces, and irreducible polynomials over* GF(2), J. Algorithms, **37** (2) (2000), 267–282.

[2] J-P. Duval, *Génération d'une section des classes de conjugaison et arbre*

des mots de Lyndon de longueur bornée, Theoret. Comput. Sci., **60** (3) (1988), 255–283.

[3] H. Fredricksen and I.J. Kessler, *An algorithm for generating necklaces of beads in two colors*, Discrete Mathematics, **61** (1986), 181–188.

[4] H. Fredricksen and J. Maiorana, *Necklaces of beads in k colours and k-ary de Bruijn sequences*, Discrete Mathematics, **23** (1978), 207–210.

[5] F. Gray, *Pulse code communications*, U.S. Patent 2632058, 1953.

[6] F. Harary, G. Prins and W.T. Tutte, *The number of plane trees*, Nederl. Akad. Wetensch. Proc. Ser. A 67=Indag. Math., **26** (1964), 319–329.

[7] D. Knuth, *The Art of Computer Programming: Vol. 4 Fascicle 4*. Addison-Wesley, 2006.

[8] F. Ruskey, *Combinatorial Generation*, In preparation.

[9] F. Ruskey, C.D. Savage and T.M. Wang, *Generating necklaces*, Journal of Algorithms, **13** (1992), 414–430.

[10] F. Ruskey and J. Sawada, *Generating necklaces and strings with forbidden substrings*, In: Computing and combinatorics (Sydney, 2000), volume 1858 of Lecture Notes in Comput. Sci., pages 330–339. Springer, Berlin, 2000.

[11] C.D. Savage, *A survey of combinatorial Gray codes*, SIAM Review, **39** (4) (1997), 605–629.

[12] J. Savage, *Generating bracelets in constant amortized time*, SIAM J. Comput., **31** (1) (2001), 259–268 (electronic).

[13] J. Sawada, *Generating rooted and free plane trees*, To appear in ACM Transactions on Algorithms, 2005.

[14] M. Sekanina, *On an ordering of the set of vertices of a connected graph*, Spisy Přírod. Fak. Univ. Brno, pages 137–141, 1960.

[15] T. Ueda, *Gray codes for necklaces*, Discrete Math., **219** (1–3) (2000), 235–248.

[16] V. Vajnovszki, *A loopless generation of bitstrings without p consecutive ones*, In: Combinatorics, computability and logic (Constanţa, 2001), Springer Ser. Discrete Math. Theor. Comput. Sci., pages 227–240. Springer, London, 2001.

[17] V. Vajnovszki, *Gray code order for Lyndon words*, Discrete Mathematics & Theoretical Computer Science, To appear.

[18] T. Walsh, *Generating Gray codes in $O(1)$ worst-case time per word*, In: Discrete mathematics and theoretical computer science, volume 2731 of Lecture Notes in Comput. Sci., pages 73–88. Springer, Berlin, 2003.

[19] T.M.Y. Wang and C.D. Savage, *A Gray code for necklaces of fixed density*, SIAM J. Discrete Math., **9** (4) (1996), 654–673.