Discrete Mathematics 61 (1986) 181–188
North-Holland

AN ALGORITHM FOR GENERATING NECKLACES OF BEADS IN TWO COLORS

Harold FREDRICKSEN
Mathematics Department, Naval Postgraduate School, Monterey, CA, U.S.A.

Irving J. KESSLER
Institute for Defense Analysis, Princeton, NJ, U.S.A.

Received 2 December 1985

In this paper we describe the necklaces of beads of length n in two colors and their equivalence to binary cycles from a circulating register of length n. We exhibit a correspondence between the binary cycles of length n and the lexicographic compositions of the integer n. We then give algorithms to generate the necklaces and the lexicographic compositions. We compare our algorithms to an exhaustive algorithm for generating the necklaces. We also give an algorithm for generating all necklaces of a specific density.

A necklace of beads of length n in two colors is a circular arrangement of the n beads. Two necklaces are inequivalent if they cannot be transformed from one to the other by a cyclic rotation of the beads.

The necklaces can also be viewed as the decomposition of the space V^n of n-tuples of zeros and ones into cycles by the cyclic permutation that takes $v_1v_2\cdots v_n \to v_2\cdots v_nv_1$. The number of necklaces and the number of cycles formed by this circulating permutation is well known [1], [2] to be Z_n where

$$Z_n = \frac{1}{n}\sum_{d|n} \phi(d)2^{n/d}.$$

The summation is taken over all divisors d of the dimension of the space n and ϕ is Euler's totient function.

Several authors [2–9] have studied these cycles and used them in a variety of applications. The cycles have always been generated in an ad hoc fashion. Basically, in order to find a new cycle a search has to be made to find an element of V^n not appearing on any of a previously determined set of cycles. We exhibit in this paper an algorithm which generates these cycles directly without any search through the space V^n of the 2^n binary n-tuples. The same methods work for k-ary n-tuples.

We also describe a companion algorithm which generates a list of lexicographic compositions of the integer n. A composition of n is a collection of positive

integers $a_1, a_2, \ldots, a_j$ with the property that $n = a_1 + a_2 + \cdots + a_j$. Two compositions of n are equivalent if the second uses the same parts as the first and in the same but cycled order. Among the equivalence classes of compositions of n one composition from each class can be singled out as the lexicographic representative of its class. We call it a lexicographic composition or say the composition $a_1, a_2, \ldots, a_j$ is lexicographic if $a_1 = a_i$, $a_2 = a_{i+1}, \ldots, a_{k-1} = a_{i+k-2}$, $a_k > a_{i+k-1}$ for some k and each $i > 1$. We reduce all subscripts bigger than j by j. The lexicographic compositions have also been given attention in the literature [10–13].

2.

In fact, there is a 1–1 correspondence between the non-zero cycles of the circulating permutation and the lexicographic compositions. To the composition $a_1, a_2, \ldots, a_j$ we assign the cycle $1 \cdots 101 \cdots 10 \cdots 01 \cdots 10$ which consists of $a_1 - 1$ ones followed by a single zero and $a_2 - 1$ ones followed by a single zero, etc. Each of these cycles ends with a zero. Then cycling the binary cycle to each of the j positions which contain a zero in the lowest order bit is exactly the same process as circularly cycling the parts of the composition $a_1, \ldots, a_j$. To the lexicographic composition corresponds the largest number on the binary cycle. Of course, there is one binary cycle which contains no zeros. Thus there are exactly $Z_n - 1$ lexicographic compositions of n.

There are 2^{n-1} parts in all of the periodic portions of the lexicographic compositions of n since exactly half of the bits in the circulating cycles are ones. The Z_n cycles of lengths d for d a divisor of n are made up of the primitive cycles of length d. The number of these primitive cycles of length d is

$$\Phi(d) = \frac{1}{d} \sum_{d' \mid d} \mu(d') 2^{d/d'},$$

where μ is the Möbius function and the summation is over all divisors d' of d. If we add $d \cdot \Phi(d)$ over all divisors d of n we get

$$\sum_{d \mid n} d\Phi(d) = \sum_{d \mid n} \sum_{d' \mid d} \mu(d') 2^{d/d'} = \sum_{d' \mid n} \mu(d') 2^{n/d'} = 2^n,$$

where the last equality follows by Möbius inversion.

The sequences of zeros and ones on these cycles are broken into a string of contiguous zeros followed by a string of contiguous ones followed by a string of contiguous zeros, etc. These contiguous strings of like bits are called runs of zeros and ones respectively. There are among all the n-tuple strings exactly one string of exact run length n of zeros and of ones. Of exact length $n - 1$ there is also one run of both zeros and ones as the $n - 1$ long string is then followed by a bit of opposite parity. The strings of zeros and ones alternate and a string of zeros (say)

of length j is followed by a one and then $n-j-1$ other bits. Thus, we see that the total number of parts of size j among all the periodic portions of the lexicographic compositions is 2^{n-j-1} for $j=1, 2, \ldots, n-1$ and one part of size n. As an example consider the set of circulating cycles of length 6 containing at least a single zero (periodic cycles are shorter) and the corresponding lexicographic compositions (periodic parts excluded). When the periodic portions are retained the cycles are also called necklaces of beads of length n in 2 colors of beads.

111110	6
111100	5 1
111010	4 2
111000	4 1 1
110	3 (3)
110100	3 2 1
110010	3 1 2
110000	3 1 1 1
10	2 (2 2)
101000	2 2 1 1
100	2 1 (2 1)
100000	2 1 1 1 1
0	1 (1 1 1 1 1)

Note there are 16 parts of size 1, 8 parts of size 2, 4 parts of size 3, 2 parts of size 4, 1 part of size 5 and 1 part of size 6 for 32 parts in all.

3.

We first give an algorithm to generate the necklaces of length n in 2 colors of beads. We need to define an operation $\theta: V^n \to V^n$ as follows: $\theta(a_1, a_2, \ldots, a_n) = (b_1, b_2, \ldots, b_n)$, where $b_i = a_i$ for $i = 1, 2, \ldots, j-1$ and j is defined as the largest subscript such that $a_j > 0$ and $a_k = 0$ for all $k > j$.

$$b_j = a_j - 1, \qquad b_{j+t} = b_t \quad \text{for } t = 1, 2, \ldots, n-j.$$

Necklace algorithm (for necklaces of length n).

0. The initial necklace is $11\cdots1 = 1^n$;
1. To find the $i+1$st necklace apply θ to the ith necklace;
2. The resulting string is the next necklace if and only if $j \mid n$.
3. If $j \nmid n$, then form $\theta^2, \theta^3, \ldots, \theta^k$ to the ith necklace until the smallest k is found so that $j \mid n$. The resulting string is the next necklace.
4. If we have not found the last necklace 0^n by steps 1, 2, 3 above return to step 1.

The reader is invited to apply the algorithm to the case $n=6$ to find the necklaces in the example above. Note that the first necklace 1^n does not appear in the example. Before we return to an analysis of the efficiency of the algorithm we give an algorithm to generate the lexicographic compositions.

LexComp Algorithm (lexicographic compositions of n).

1. The first lexicographic composition is n.

2. If $a_1, a_2, \ldots, a_j$ is the current lexicographic composition then (*) is the operation to find the next composition. We find the largest i such that $a_i > 1$ and $a_k = 1$ for $k > i$.

3. Form the partial composition $a = a_1, \ldots, a_{i-1}, a_i - 1$. Then with the remainder periodically reproduce a insofar as this is possible. That is, if $b = a_1 + a_2 + \cdots + a_{i-1} + a_i - 1$ we repeat $a[n/b]$ times, where $[x]$ is the greatest integer $\leqslant x$. If $b \cdot [n/b] < n$ we partially reproduce one more copy of a. Suppose $n - b \cdot [n/b] = c$. Then add $a_1, a_2, \ldots, a_j$ as long as $a_1 + a_2 + \cdots + a_j \leqslant c < a_1 + a_2 + \cdots + a_j + a_{j+1}$. Finally the last part added is $t = c - (a_1 + a_2 + \cdots + a_j)$. Thus the sequence produced is $a_1, a_2, \ldots, a_i - 1, a_1, a_2, \ldots, a_i - 1, a_1, a_2, \ldots, t$.

4. If step 3 terminates with $a_1 + a_2 + \cdots + a_j = n$, (i.e., $t = 0$) and $j \neq (k) \cdot (i)$, then the composition formed in 3 is not lexicographic. We re-apply (*) to the composition produced in 3 as many times as is necessary to either make $j' = (k) \cdot (i)$ or $a_1 + \cdots + a_j < n$. If $a_1 + a_2 + \cdots + a_j = n$ and $j = (k) \cdot (i)$ the composition formed in 3 is lexicographic and periodic of period i. If $a_1 + a_2 + \cdots + a_j < n$, i.e., if the last part t is smaller than a_{j+1}, the composition formed in 3 is lexicographic. Note when we reach the composition $11 \cdots 1$ after applying (*) we stop.

Before we verify the algorithm an example is in order.

Example. $n = 6$. The list of putative lexicographic compositions formed in step 3 is listed below

$$6 \to 51 \to 42 \to 411 \to 33 \to 321 \to 312 \to 3111 \to 222 \to 2211 \to 2121$$
$$\to (2112) \to 21111 \to 111111.$$

Note the composition 2112 is not lexicographic as the lexicographic representative of its equivalence class is 2211. The test for lexicography is simply this. We attempt to copy the first i integers $a_1, \ldots, a_i - 1$ as many times as possible using the remainder. If we can copy up to a_j, $j < i$ exactly with no remainder, the composition formed is not lexicographic. This follows since $a_1, a_2, \ldots, a_j, a_1, \ldots$ is 'bigger' than $a_1, a_2, \ldots, a_i - 1, a_1, \ldots$. However, if $j = ki$, (we copy $a_j = a_i - 1$) the composition is periodic and therefore lexicographic since the periodic portion obviously is lexicographic. If the remainder is not large enough to complete the part a_{j+1} the composition formed is lexicographic since the added part t is smaller than a_{j+1}.

Verification of the algorithm. To verify the algorithm we exhibit the inverse algorithm, $(*)^{-1}$, i.e., that algorithm which produces from the $k+1$st lexicographic composition the kth.

1. The last ($Z_n - 1$st) lexicographic composition is the composition $11\cdots1$.

2. If $a_1, a_2, \ldots, a_j$ is the $k+1$st lexicographic composition we find the smallest i such that $a_1, \ldots, a_i, a_{i+1}, \ldots, a_{2i}, \ldots, a_{ti}, a_{ti+1}, \ldots, a_j - 1$ is 'periodic'. That is $a_1, \ldots, a_i = a_{i+1}, \ldots, a_{2i} = \cdots = a_{(t-1)i+1}, \ldots, a_{ti}$ and $a_{ti+1}, \ldots, a_{j-1}$ is a 'prefix' of $a_1, \ldots, a_i$. Here prefix has a slightly different than the standard meaning. Recall that the composition was produced by periodically reproducing the initial string $a_1, a_2, \ldots, a_i - 1$ in so far as was possible. Then the last part which was the $k+1$st part may not be equal to a_{k+1} but in fact be smaller than a_{k+1}. Even in this case we wish to call this final string a prefix if the first k parts agree and the $k+1$st part is smaller than the $k+1$st part of the periodic string of length i.

3. Form the composition $a_1, a_2, \ldots, a_{i-1}, a_i + 1\ 1\ 1\cdots1$.

4. If the composition formed in 3 is lexicographic, then it is the kth lexicographic composition. We decrease k and return to Step 2. If the composition is not lexicographic we re-apply Step 2 on the composition produced in Step 3. When we reach the composition n we stop.

Obviously $(*)$ and $(*)^{-1}$ are strictly monotone operations. Since $(*)(*)^{-1}$ is the identity map on compositions it is evident that the two algorithms are inverse to one another and that the same set of lexicographic compositions are produced by the two algorithms.

The next lexicographic composition in the list following the lexicographic composition s_j is less than or equal to the composition (not necessarily lexicographic) formed by the process $(*)$. $(*)$ obviously takes the smallest possible step consistent with the idea of avoiding non-lexicographic compositions by repeating periodically the first part of the composition s_j (i.e., the initial part preceding the last non-one part of s_j).

The verification of the algorithm for necklaces is identical to the algorithm for lexicographic compositions and we omit it.

4.

In applying the operation θ to find the necklaces of length n, occasionally strings of length n are produced which are not necklaces. The efficiency of the algorithm can be measured by seeing how many strings are produced that are not necklaces. In Table 1 we present some statistics for how often θ produces good n-tuples (necklaces) and bad n-tuples (non-necklaces). The table contains statistics on the number of times that θ produces strings of good or bad n-tuples of length k for $k = 1, 2, 3, \ldots$.

The total number B_n of bad n-tuples is greater than the total number, Z_n, of necklaces. However, we see that asymptotically $B_n \sim Z_n$.

Table 1.
Distribution of θ^k to produce necklaces and non-necklaces of length $n \leq 19$

	$k=1$	2	3	4	5	6	7	8	9	
b_2	0									$B_2 = 0$
g_2	0	0	1							$N_2 = 3$
b_3	1									$B_3 = 1$
g_3	0	2								$N_3 = 4$
b_4	2									$B_4 = 2$
g_4	0	3								$N_4 = 6$
b_5	4	1								$B_5 = 6$
g_5	4	2								$N_5 = 8$
b_6	7	1								$B_6 = 9$
g_6	4	5								$N_6 = 14$
b_7	14	2	1							$B_7 = 21$
g_7	16	2								$N_7 = 20$
b_8	24	4	1							$B_8 = 35$
g_8	24	6								$N_8 = 36$
b_9	45	9	0	1						$B_9 = 67$
g_9	52	4								$N_9 = 60$
b_{10}	85	15	1	1						$B_{10} = 118$
g_{10}	90	9								$N_{10} = 108$
b_{11}	153	27	4	0	1					$B_{11} = 224$
g_{11}	184	2								$N_{11} = 188$
b_{12}	281	47	5	0	1					$B_{12} = 395$
g_{12}	318	17								$N_{12} = 352$
b_{13}	529	88	10	1	0	1				$B_{13} = 745$
g_{13}	628	2								$N_{13} = 632$
b_{14}	986	157	14	2	0	1				$B_{14} = 1356$
g_{14}	1 140	21								$N_{14} = 1182$
b_{15}	1 871	281	24	4	0	0	1			$B_{15} = 2528$
g_{15}	2 172	10								$N_{15} = 2192$
b_{16}	3 531	502	39	5	1	0	1			$B_{16} = 4684$
g_{16}	4 044	36								$N_{16} = 4116$
b_{17}	6 716	915	66	9	2	0	0	1		$B_{17} = 8798$
g_{17}	7 708	2								$N_{17} = 7712$
b_{18}	12 766	1 645	102	15	2	0	0	1		$B_{18} = 16440$
g_{18}	14 462	70								$N_{16} = 14602$
b_{19}	24 392	2 992	183	20	4	1	0	0	1	$B_{19} = 31040$
g_{19}	27 592	2								$N_{19} = 27596$

Theorem. *If* $a = a_1, a_2, \ldots, a_n$ *is a necklace,* $\neq 0^n$, *then one of* $\theta(a)$, $\theta^2(a), \ldots, \theta^{[(n-1)/2]}(a)$ *is a necklace.*

The proof is obvious and the longest gap between necklaces is between the necklaces $2\ 1^{[(n-4)/2]}2\ 1^{[(n-3)/2]}$ and $2\ 1^{n-2}$. Therefore, $B_n \leq [\frac{1}{2}(n-1)]Z_n$.

On the other hand there is almost always a bad n-tuple between every pair of necklaces as $\theta(a)$ can be a necklace for necklace a only if $\theta(a)$ is periodic of period $j \mid n$, $j < n$.

Theorem. *If $a = a_1, a_2, \ldots, a_n \neq 1^n$ is a necklace and $\theta(a)$ is a necklace, then $\theta(a)$ is periodic of period $j \mid n$, $j < n$.*

Proof. Since $a = a_1, a_2, \ldots, a_n$ is a necklace, not the all 1's necklace, it must end in a 0. Therefore $j < n$ and $\theta(a)$ can be a necklace only when $j \mid n$. □

A little more can be stated about consecutive necklaces.

Theorem. *If a is a necklace, then $\theta(a)$ and $\theta^2(a)$ are not both necklaces.*

Proof. Suppose a and $\theta(a)$ are necklaces. By the previous theorem $\theta(a)$ is periodic of period j. If $\theta(a) \neq 0^n$, then $\theta(a)$ contains at least n/j 1's with the last 1 appearing at least in position $(n/j - 1)j + 1 = n - j + 1$ of $\theta(a)$. Moreover, $2 \leqslant j \leqslant \frac{1}{2}n$. Thus the last 1 in $\theta(a)$ occurs no earlier than position $n - \frac{1}{2}n + 1 = \frac{1}{2}n + 1$ and $\theta^2(a)$ cannot be a necklace since it can't be periodic. □

5.

We may generate necklaces of a specific density d/n in a similar manner to that above. We use the algorithm for producing lexicographic compositions but restrict the compositions to those of d parts.

The algorithm is only slightly complicated in that we must reserve d of the n positions as place holders and then make a lexicographic composition of $n - d$ with exactly d non-negative parts. The stopping rule is less precise. However, we can stop when the first part is less than n/d as there is no lexicographic composition following.

Algorithm. Lexicographic compositions of n with d parts.

1. The first lexicographic composition is $n - d + 1\ 1\ 1 \cdots 1$.
2. If $a_1, a_2, \ldots, a_d$ is the kth lexicographic composition we find the largest $i \leqslant d - 1$ such that $a_i > 1$ and $a_k = 1$ for $i < k < d$.
3. Form the partial composition $a = a_1, a_2, \ldots, a_i - 1\ 1\ 1\ 1 \cdots 1$ of d parts. Then with the remainder we form as far as possible the ultimately 'periodic' composition $b = 0, 0, \ldots, 0, a_1 - 1, \ldots, a_i - 2, a_1 - 1, \ldots$ whose initial i parts are all zero. (Any extra remainder must be added into the dth part.) Add b to a componentwise.
4. If the sum composition is lexicographic then it is the $k + 1$st composition. We increment k and return to 2. If the composition is not lexicographic we apply Step 2 to the sum composition. We stop when the first part of the sum composition is less than n/d.

The test for lexicography is similar to the test we gave in our original algorithm.

We examine the vector b to form the test. b is a d-long vector whose first i parts are zero. Suppose the next parts of b are $b_{i+1}b_{i+2}\cdots b_d$. We write $d=pi+j$ with $0<j\leqslant i$. If $b_d<a_j$ the composition is lexicographic. If $b_d=a_j$ and $j=i$ the composition is periodic and lexicographic. If $b_d=a_j$ and $j\neq i$ the composition is not lexicographic. If $b_d>a_j$ the composition is not lexicographic.

The verification is straightforward and similar to the original algorithm and we omit it.

As an example we have $n=8$, $d=4$. We produce the following compositions. Bracketed compositions are not lexicographic.

$$5111\to 4211\to 4121\to 4112\to 3311\to 3321\to 3212\to 3131$$
$$\to 3122\to (3113)\to 2222\to (2213)\to (2123)\to (2114)\to (1115).$$

A more elegant stopping rule which is a function of n and d can no doubt be used to suppress the generation of the obviously non-lexicographic compositions at the end.

This algorithm was programmed to produce all density d necklaces of length n for $d\leqslant n\leqslant 25$. Curiously, the total number of 'bad' compositions formed equaled the number of 'bad' compositions for the first algorithm.

Obviously we can by the same correspondence as above generate the binary necklaces of density d by an algorithm similar to the above algorithm.

References

[1] J. Riordan, An Introduction to Combinatorial Analysis (Wiley, New york, 1958).
[2] S.W. Golomb, Shift Register Sequences (Holden-Day, San Francisco, 1967).
[3] S.W. Golomb, Irreducible polynomials, synchronization codes, primitive necklaces and the cyclotomic algebra, in: Combinatorial Mathematics and its Applications (Univ. of North Carolina Press, Chapel Hill, 1969).
[4] H. Fredricksen, A class of nonlinear de Bruijn cycles, J. Combin. Theory 19(2) (1975).
[5] A. Lempel, On extremal factors of the de Bruijn graph, J. Combin. Theory 11 (1971).
[6] J. Mykkeltveit, A proof of Golomb's conjecture for the de Bruijn graph, J. Combin. Theory 13 (1973).
[7] J. Mykkeltveit, Generating and counting the double adjacencies in a pure cycling register, IEEE Trans. Comput. 24 (1975).
[8] E.J. van Lantschoot, Double adjacencies between cycles of a circulating shift register, IEEE Trans. Comput. 22 (1973).
[9] E.J. van Lantschoot, Double adjacency diagrams of circulating shift registers, IEEE Trans. Comput., personal communication.
[10] H. Fredricksen and I. Kessler, Lexicographic compositions and de Bruijn sequences, J. Combin. Theory 22 (1977).
[11] T. Motzkin, Ordered and cyclic partitions, Riveon Lematematica 1 (1947).
[12] H. Fredricksen and J. Maiorana, Necklaces of beads in k colors and k-ary de Bruijn sequences, Discrete Math. 23(3) (1978).
[13] H. Fredricksen, A survey of full length nonlinear shift register cycle algorithms, SIAM Review 24(2) (1982).