J. Basic. Appl. Sci. Res., 2(4)3323-3328, 2012

ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

A Fast Algorithm to Enumerate Fixed Density Bracelets

S. Karim*, Z. Alamgir and S.M. Husnine

National University of Computer and Emerging Sciences, Lahore Campus
Block B, Faisal Town Lahore
PAKISTAN

ABSTRACT

A bracelet is the lexicographically smallest k-ary string equivalent under rotation and reversal. A bracelet is said to be of fixed density, if the number of occurrences of the symbol 0 in the string is fixed. In this work, we present a simple recursive scheme to list bracelets with fixed density d. We claim that our scheme is optimal in terms of time requirements, as it takes constant time on the average to list a binary bracelet with fixed density. We use sophisticated combinatorial techniques to prove our claim.
KEY WORDS: Bracelet with fixed density, Necklace, CAT algorithm, combinatorial generation.

1. INTRODUCTION

Generation of discrete combinatorial objects is of immense importance in mathematics and computer science. Knuth [5] and Ruskey [9] have recently compiled their books on the subject. Several schemes have been developed for the generation of combinatorial structures such as necklaces, Lyndon words and their variants. Kevin et al. [1], give a recursive framework to generate prenecklaces, necklaces and Lyndon words. This framework is used to list restricted classes of this family of objects. Ruskey and Sawada [8] develop the algorithms to generate necklace with fixed density and later Sawada [4] gives the algorithm for necklaces with fixed content using the same framework. Recently, Vajnovszki answers the problem of generating unrestricted binary necklaces in Gray code order [12]. Later Vajnovszki and Weston give a generalized and optimized algorithm for arbitrarily large alphabet size [10] [11]. However, no significant work is done to list restricted classes of bracelets. In [6], Lisonek proposes a linear algorithm for generating a list of all bracelets by modifying the necklace generation algorithm of Ruskey and Wang [7]. Sawada in [3] gives a constant amortized time (CAT) algorithm for bracelet generation. Bracelets are of great practical significance; the exhaustive list of bracelets is used in the calibration of color printers [2]. In this paper, we present a recursive scheme to list restricted class of bracelets, namely those with fixed density. We have designed our algorithm such that the total number of basic operations performed is proportional to the number of bracelets generated by our algorithm. Hence, our algorithm takes constant time on the average to list a bracelet. It is extremely desirable in generation algorithms that the number of steps taken in listing successive object remains constant.

2. PRELIMINARIES

A necklace is a lexicographically minimal k-ary string equivalent under string rotation, i.e. $a_1a_2 \ldots a_i \ldots a_n$ is equivalent to $a_ia_{i+1} \ldots a_na_1 \ldots a_{i-1}$ for all i, $1 < i \leq n$. The set of all necklaces of length n on k alphabets is represented as $\mathbf{N}_k(n)$ and the cardinality of $\mathbf{N}_k(n)$ is denoted by $N_k(n)$. A prenecklace of length n is a prefix of some necklace of length m where $m \geq n$. The set of all prenecklaces of length n is denoted by $\mathbf{P}_k(n)$. The cardinality of $\mathbf{P}_k(n)$ is $P_k(n)$. Another restricted class of necklaces is an aperiodic necklace, called Lyndon words. The set of Lyndon words of length n is denoted by $\mathbf{L}_k(n)$. A bracelet is a lexicographically minimal k-ary string that is symmetric under rotation and reversal. $\mathbf{B}_k(n)$, represents a set of length n bracelets and its cardinality is denoted by $B_k(n)$. A k-ary string is said to be of fixed density, if the number of occurrences of symbol 0 is fixed. Necklaces, prenecklaces and bracelets with fixed density are denoted in the similar manner. We add an additional parament d, number of non-zero symbols, to denote the density of the string. We use the notations $\mathbf{N}_k(n, d)$, $\mathbf{P}_k(n, d)$, $\mathbf{B}_k(n, d)$ to denote necklace with fixed density, prenecklaces with fixed density and bracelets with fixed density respectively. Similarly $N_k(n, d)$, $P_k(n, d)$, and $B_k(n, d)$ denote the cardinality of $\mathbf{N}_k(n, d)$, $\mathbf{P}_k(n, d)$ and $\mathbf{B}_k(n, d)$.

There are at most two necklaces in each equivalence class of the bracelet. Therefore, following relation holds.

$$N_k(n, d) \leq 2B_k(n, d) \qquad (1)$$

***Corresponding Author:** S. Karim, Department of Computer Science, National University of Computer and Emerging Sciences, Block B, Faisal Town, Lahore Pakistan. Office: +92111-128-128 Email: saira.karim@nu.edu.pk

Our recursive algorithm for generating bracelets with fixed density is based on the following theorem given in [1].

Theorem 1: *(Fundamental Theorem of Necklaces)*

Let $\alpha = a_1 a_2 \cdots a_{n-1} \in \mathbf{P}_k(n-1)$ *and* $p = lyn(\alpha)$. *The string* $\alpha b \in \mathbf{P}_k(n)$ *iff* $a_{n-p} \leq b \leq k-1$. *Furthermore,*

$$lyn(\alpha b) = \begin{cases} \text{p} & \text{if } a_{n-p} = b, \\ \text{n} & \text{if } a_{n-p} > b. \end{cases}$$

Here, $lyn(\alpha)$ is the longest Lyndon prefix of α that is $lyn(\alpha) = max\{1 \leq p \leq n-1 | (a_1 \cdots a_p) \in \mathbf{L}_k(p)\}$. Also, α b is a necklaces iff n mod $lyn(\alpha b) = 0$. $\mathbf{N}_k(n)$ is generated by recursive application of fundamental theorem of necklaces.

3. GENERATION ALGORITHM

Now, we present the basic idea of our recursive scheme. In our scheme to list $\mathbf{B}_k(n, d)$, we assume that $0 < d < n$. We develop a recursive algorithm which modifies the generation of necklaces with fixed density scheme given in [8] to list bracelets with fixed density. As defined earlier, bracelets are necklaces that are lexicographically least among all their reverse rotations. The basic idea is to check all reverse rotations of each generated necklace and discard all those necklaces which are smaller than any of its reverse rotations.

The algorithm in [8] to lists $\mathbf{N}_k(n, d)$ recursively generates prenecklaces in accordance with the fundamental theorem of necklaces. Each recursive call appends a string to the current prenecklace. Let $\gamma = c_1 c_2 \cdots c_{a(t)}$ be the current prenecklace generated by the algorithm. The following variables are maintained:

t: density of γ

$a(i)$: position of i^{th} non-zero symbol in γ for $1 \leq i \leq d$, and

p: density of the longest prefix of γ such that $c_1 c_2 \cdots c_{a(p)} \in \mathbf{L}_k(a(p), p)\}$.

Each recursive call determines valid value $c_{a(t+1)}$and valid position $a(t+1)$ of next non-zero symbol such that $0 < c_{a(t+1)} < k$. We use the notation APND(γ, $c_{a(t+1)}$) to denote the function which appends a string of zeros from position $a(t)+1$ to $a(t+1)-1$, and a non-zero symbol $c_{a(t+1)}$ to the current prenecklace and generates a prenecklace of density $t+1$. Time to append the string of zeros is reduced to a constant factor by initializing the data structure which holds γ to all zeros and restoring $c_{a(t+1)}$ to 0 before exiting the recursive function. $c_{a(t+1)}$ is chosen in such a way that $c_1 c_2 \cdots c_{a(t)} 0^{a(t+1)-a(t)-1} c_{a(t+1)} \in \mathbf{P}_k(a(t+1), t+1)$. In order to maintain lexicographic order, the algorithm computes maximum possible value of $a(t+1)$ and iterates from minimum to maximum value of $c_{a(t+1)}$ to generate next prenecklace. Same procedure is performed iteratively for successive values of $a(t+1)$ till $a(t+1) = a(t)+1$. As a necklace of positive density must not end with a zero, thus the recursion stops when current prenecklace has density $d-1$. The last non-zero symbol is appended at position n. Another test is performed in constant time to determine the valid values for last non-zero symbol. The value of $a(t+1)$ in γ depends on the length n and density d. For generation of necklace with fixed density, following conditions must hold:

I. $\lfloor (n-1)/d + 1 \rfloor \leq a(1) \leq n-d+1$

II. $a(i) \leq n-d+i$:

Once a necklace has been generated, a naive approach to list bracelets is to compare the necklace $\gamma = c_1 c_2 \cdots c_n$, generated by necklace with fixed density algorithm, with all its reverse strings and discard the necklace if $c_1 c_2 \cdots c_n$ is greater than any of its reverse rotations. We require $O(n^2)$ time test to perform such comparison. A necklace is a bracelet, if no such case arises. Adding this test for each necklace is computationally very expensive and will not result in a CAT algorithm.

We develop a CAT algorithm using couple of optimizations proposed in [3]. The first and simpler optimization is that if the necklace γ is of the form $c^i c_{i+1} \cdots c_n$ where $c \neq c_{i+1}$, then only those reverse rotations are required to check that also begins with c. For example, we only need to check two reverse rotations for the necklace 00121101211003 i.e. 00300112101121 and 00112101121003. Also, we do not need to wait for the generation of entire necklace. The reverse checking can be performed as soon as the prenecklace having above mentioned form is generated. Since, we assume that $d < n$, so all prenecklaces must start with symbol 0 and will be of the form $\gamma = 0^i c_{i+1} c_{i+2} \cdots c_{a(t)}$. Also note that necklace with fixed density algorithm does not generate a prenecklace that ends with 0. We perform the reverse checking only, if γ is of the form $0^i c_{i+1} c_{i+2} \cdots c_{a(t)-i-1} 0^i c_{a(t)}$. If $0^i c_{i+1} \cdots c_{a(t)-i-1} 0^i > 0^i c_{a(t)-i-1} \cdots c_{i+1} 0^i$, then we terminate further computation of γ. The second optimization uses the following Theorem in [3].

Theorem 2 *If $c_1c_2 \cdots c_n$ is a necklace where $c_1c_2 \cdots c_q = c_qc_{q-1} \cdots c_1$ and there exists an r in $\{q+1 \cdots n\}$ such that $c_1c_2 \cdots c_r = c_rc_{r-1} \cdots c_1$ and $c_{r+1} \cdots c_n \leq c_nc_{n-1} \cdots c_{r+1}$, then $c_{q+1} \cdots c_n \leq c_nc_{n-1} \cdots c_{q+1}$.* Let *r* be the longest prefix of the necklace $c_1c_2 \cdots c_n$ such that $c_1c_2 \cdots c_r = c_rc_{r-1} \cdots c_1$.

It follows clearly from above theorem that $c_1c_2 \cdots c_n$ is a bracelet if $c_{r+1}c_{r+2} \cdots c_n \leq c_nc_{n-1} \cdots c_{r+1}$. This means we need an additional comparison between $c_{r+1} \cdots c_n$ and its reverse string. In [3], additional comparison is performed symbol by symbol in each recursive call whenever $a(t) > (n-r/2) + r$. However, in our algorithm this cannot be done by a single comparison because function APND may append more than one symbol to the current prenecklace. Hence, non-constant amount of work may be required.

This comparison can be performed efficiently using the observation that there is only one non-zero symbol in the substring $0^{a(t)-a(t-1)-1}c_{a(t)}$. The comparison of whole substring can be answered in unit time if $c_{a(t)} \neq c_{n-a(t)+r+1}$. Otherwise, we determine the length of substring of zeros starting at position $n - a(t) + r + 2$. In order to compute the length of substring of zeros starting at position $n - a(t) + r + 2$, we define the following variables

s(i): density of prenecklace $c_1c_2 \cdots c_i$,

l(i) : length of substring of zeros starting at position *i*.

Let $e = n - a(t) + r + 1$. We can compute $l(e+1) = a_{s(e+1)} - a_{s(e)} - 1$. Hence we define *CMP* as follows

$$CMP(\gamma, a(t), r) = \begin{cases} \text{false} & \text{if } c_{a(t)} > c_e, \\ \text{true} & \text{if } c_{a(t)} < c_e, \text{ or } (a(t) - a(t-1) - 1 > l(e+1) \text{and } c_{a(t)} = c_e), \\ \text{CMP}(\gamma, a_{t-1}, r) & \text{if } c_{a(t)} = c_e \text{ and } a(t) - a(t-1) - 1 \leq l(e+1). \end{cases}$$

The default value of CMP is *false* and new value of CMP is computed only when $a(t) > (n - r)/2 + r$. If we keep the values of *CMP*(γ, *a(i), r*) for all $1 \leq i < t$, then we can compute the function *CMP*(γ, *a(i+1), r*) in constant number of steps. As we assume that $0 < d < n$, so each bracelet should begin with symbol 0. We define *u* to be the number of zeros at beginning of the current prenecklace. Let $\gamma = 0^u c_{u+1} \cdots c_{a(t)}$ be the current prenecklace. We design a recursive scheme $BF_u(t, r,$ CMP) to list the set $\mathbf{B}_k(n, d)$ such that there are *u* number of zeros in the beginning. The pseudo code of our scheme is shown as Algorithm: $BF_u(t, r, CMP(\gamma, a(t+1), r))$. We begin with $u := n - d$ and repeatedly compute BF_u for successive values of *u* to generate $\mathbf{B}_k(n, d)$. The variable *r* is initialized to *u* and is updated to one less than the length of the prenecklace, whenever the current prenecklace is equal to its reversal during reverse checking. Furthermore, *s(i)* is not assigned a valid value whenever $c_i = 0$. However, this information is not necessary, since we only use *s(i)* when $c_i = c_e \neq 0$. The function CHK_{REV} compares the current prenecklace with its reverse string. The values return by CHK_{REV} are given below.

$$CHK_{REV}(c_1c_2 \cdots c_{at}) = \begin{cases} 0 \;\; if\; c_1c_2 \cdots c_{a(t)} < c_{a(t)}c_{a(t)-1} \cdots c_1, \\ 1 \;\; if\; c_1c_2 \cdots c_{a(t)} = c_{a(t)}c_{a(t)-1} \cdots c_1, \\ -1 \; if\; c_1c_2 \cdots c_{a(t)} \geq c_{a(t)}c_{a(t)-1} \cdots c_1. \end{cases}$$

Since $\mathbf{B}_k(n, d)$ is a subset of $\mathbf{N}_k(n, d)$, we generate strings that belong to the set $\mathbf{N}_k(n, d)$ and we eliminate all those necklaces whose reversed rotations are less than the necklace itself. Hence, BF_u lists all bracelets with fixed density exactly once.

Theorem 3 $BF_u(1, u, false)$ *lists all elements of* $\mathbf{B}_k(n, d)$ *exactly once for all u such that* $(n-1)/d \leq u \leq n - d$ *in lexicographic order.*

4. ANALYSIS

In this section, we show that our algorithm takes constant time on average to generate a binary bracelet. The computation tree is a way to represent the computational steps of a recursive algorithm. In our algorithm, each node of the computation tree represents a recursive call to $BF_u(t, r,$ CMP(γ, *a(t+1), r*)) and each edge corresponds to transition from one recursive call to the other.

The size of the computation tree of our algorithm is smaller than the computation tree of necklaces with fixed density generation algorithm. Using the relation(1) and the fact that necklace with fixed density algorithm works in CAT, we claim that the size of the computation tree in our algorithm is proportional to $B_k(n, d)$. For each recursive call, the amount of work done by APND is constant. However, there are some nodes in our computation tree where *CHKREV* is executed. Thus, they perform more than constant amount of work. Figure 1 shows the computation tree of $\mathbf{B}_2(8, 4)$ and boxed nodes are the ones for which CHK_{REV} is called. We prove that our algorithm works in CAT by showing that total symbol comparisons performed by CHK_{REV} during the entire scheme is proportional to $B_k(n, d)$.

The total prenecklaces generated by our scheme is proportional to $B_k(n, d)$. Therefore, we map each comparison in CHK_{REV} to a unique prenecklace listed by our algorithm. Note that in case of binary bracelets we assume $d \leq n/2$. Fixed density bracelets with $d > n/2$ are listed by generating $\mathbf{B}_2(n, n-d)$ and then complementing the output. Each recursive call in the necklaces with fixed density algorithm increases the density of current prenecklace. This means that the algorithm does not generate any prenecklace that ends with 0. Furthermore, CHK_{REV} is computed at most once for each prenecklace $\gamma = 0^i c_{i+1}c_{i+2} \cdots c_{a(t)-i-1}0^i c_{a(t)}$ of length $a(t)$ where $c_{i+1} = c_{a(t)-i-1} \neq 0$ and $c_{a(t)} \geq c_{i+1}$. CHK_{REV} compares symbol c_j with $c_{a(t)-j}$ for $i+1 \leq j \leq a(t)/2$. We stop doing comparisons when either $j > a(t)/2$ or $c_j \neq c_{a(t)-j}$. Since, there is at most one unequal comparison for each prenecklace, so we count the cost of such comparison as constant.

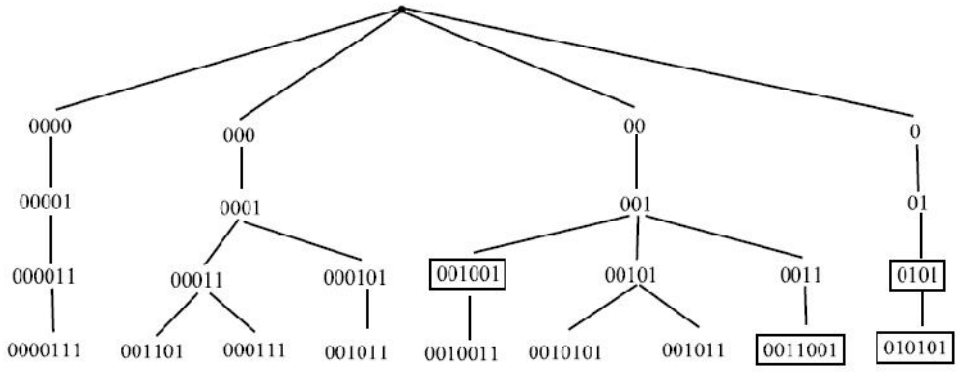

Figure 1: Computation Tree for $\mathbf{B}_2(8, 4)$ from $BF_u(t, r, CMP(\gamma, a(t+1), r))$

We define a mapping for the case when $k = 2$. Let us assume that γ and β are two different prenecklaces for which CHK_{REV} is computed.

$\gamma = 0^i c_{i+1}c_{i+2} \cdots c_{a(t)-i-1}0^i c_{a(t)}$,
$\beta = 0^i b_{i'+1}b_{i'+2} \cdots b_{a(t)-i'-1}0^{i'} b_{a(t)}$

Let $n_k(\gamma, j)$ denote the number of occurrences of symbol k in the substring $0^i c_{i+1}c_{i+2} \cdots c_j$. For example, $n_0(001111001011, 7) = 3$ and $n_1(001111001011, 7) = 4$. Consider the following mapping f:

$$f(\gamma, j) = \begin{cases} 0^{n_0(\gamma,j)}1^{n_1(\gamma,j)}c_{j+1}c_{j+2} \ldots c_{at-i-1}\, 0^i c_{a(t)} & \text{if } c_j = 0, \\ 0^{n_0(\gamma,j)}c_{j+1}c_{j+2} \ldots c_{at-i-1}\, 0^i c_{a(t)}1^{n_1(\gamma,j)} & \text{if } c_j = 1. \end{cases}$$

For example $f(001110011100111001, 3) = 001100111001110011$, and $f(001110011100111001, 6) = 000111011100111001$. Here, γ and j are valid only if there exists a corresponding equal comparison made by CHK_{REV}. It is evident, that the mapping preserves both length and contents.

Lemma 1 *For all valid γ and j, $f(\gamma, j)$ is a prenecklace .*

Proof: $f(\gamma, j)$ has maximum number of zeros in the beginning for the cases when $c_j = 0$, $n_0(\gamma, j) > i$ or $c_{j+1} = 0$, hence a valid prenecklace . Now to prove that $f(\gamma, j)$ is a prenecklace when $c_j = c_{j+1} = 1$ and $n_0(\gamma, j) = i$, let $lyn(\gamma) = p$. We can rewrite $\gamma = (0^i c_{i+1}c_{i+2} \cdots c_p)^d\, 0^i c_{i+1}c_{i+2} \cdots c_m$ and $f(\gamma, j) = 0^i c_{j+1}c_{j+2} \cdots c_p (0^i c_{i+1}c_{i+2} \cdots c_p)^{d-1}\, 0^i c_{i+1}c_{i+2} \cdots c_m 1^{n1(\gamma, j)}$, where d is a positive integer and $i < m < p$. Since $c_{i+1}c_{i+2} \cdots c_j = 1^{j-i}$ and $c_{j+1} = 1$, this means $j < p$ and $0^i c_{j+1}c_{j+2} \cdots c_p$ is Lyndon word. Also the following relation must hold:

$$0^i c_{j+1}c_{j+2} \cdots c_p < 0^i c_{i+1}c_{i+2} \cdots c_p \qquad (2)$$

Using relation (2) and repeated application of Lemma 2.2 in [8], we can conclude that $0^i c_{j+1}c_{j+2} \cdots c_p (0^i c_{i+1}c_{i+2} \cdots c_p)^{d-1}$ is also a Lyndon word. Again, using relation (2) we claim that $c_{j+1}c_{j+2} \cdots c_{j+m-i} \leq c_{i+1}c_{i+2} \cdots c_m$. This implies that $0^i c_{j+1}c_{j+2} \cdots c_p (0^i c_{i+1}c_{i+2} \cdots c_p)^{d-1}\, 0^i c_{i+1}c_{i+2} \cdots c_m$ is a prenecklace. Since 1 is the largest symbol, so by Theorem 1 $0^i c_{j+1}c_{j+2} \cdots c_p (0^i c_{i+1}c_{i+2} \cdots c_p{}^{d-1}\, 0^i c_{i+1}c_{i+2} \cdots c_m 1^{n1(\gamma, j)}$ is a prenecklace. □

Now, we prove that total comparisons made by all CHK_{REV} calls is proportional to the total prenecklaces generated. We claim that $f(\gamma, j)$ is $1-1$, for all γ valid and j.

Lemma 2. *For all valid γ and j such that γ is a binary prenecklace, the mapping $f(\gamma, j)$ is* $1-1$.

Proof: It is clear from the mapping that $f(\gamma, j) \neq f(\beta, j')$ whenever $j \neq j'$. Now, let γ and β be two prenecklaces of same length. Suppose $f(\gamma, j) = f(\beta, j')$ where $\gamma \neq \beta$. Observe that $f(\gamma, j)$ ends with 01 when $c_j = 0$ and ends with 11

when $c_j = 1$. Furthermore, the substring $c_{j+1}c_{j+2} \cdots c_{a(t)-i-1}0^i c_{a(t)}$ is not affected by the mapping f. Since, $f(\gamma, j) = f(\beta, j')$ for $j, j' \leq t/2$. This implies that $c_j = b_{j'}$ and the second half of γ and β must be equal that is

$$c_{a(t)/2+1}c_{a(t)/2+2} \cdots c_{a(t)-i-1}0^i c_{a(t)} = b_{a(t)/2+1}b_{a(t)/2+2} \cdots b_{a(t)-i'-1}0^{i'} b_{a(t)}.$$

Similarly, the trailing number of zeros in both strings must be equal that is $i = i'$. Now assume that q is the minimum index such that $c_q \neq b_q$ where $q \leq a(t)/2$. For all $i + 1 \leq j, j' < q$ clearly $f(\gamma, j) \neq f(\beta, j')$. When $j = j' = q$, either $c_j \neq c_{a(t)-j}$ or $b_j \neq b_{a(t)-j}$ because $c_{a(t)-j}$ lies in the second half of γ and $c_{a(t)-j} = b_{a(t)-j}$. So either j or j' is invalid, a contradiction. □

Let $P'_k(n, d)$ be the number of prenecklaces generated by necklace with fixed density algorithm, and $C_k(n)$ be the number of equal comparisons made by CHK_{REV} function on all prenecklaces of length n. The total number of equal comparisons made by our algorithm are

$$\sum_{i=1}^{n} C_k(i)$$

Following theorem proves that our algorithm works in CAT.

Theorem 4. $\sum_{i=1}^{n} C_k(i) \leq cB_k(n, d)$, *where c is a constant.*

Proof: In our algorithm, we stop recursion when a prenecklace of density $d-1$ has been generated and the last non-zero character is appended at position n such that the generated string is a necklace. This means that the algorithm does not generate prenecklaces of length n and by Lemma 2, the following bound holds.

$$\sum_{i=1}^{n-1} C_k(i) \leq P'_k(n, d).$$

From [8] one can show that the number of prenecklaces of length n and density d that does not end with a zero is less than $2N_k(n, d)$. This implies that $C_k(n) \leq 2N_k(n, d)$. Also $P'_k(n, d) \leq c'N_k(n, d)$. Hence, we obtain the following relation

$$\begin{aligned}\sum_{i=1}^{n} C_k(i) &\leq (c' + 2)N_k(n, d) \\ &\leq 2(c' + 2)B_k(n, d) \\ &= cB_k(n, d)\end{aligned}$$

As mentioned earlier, if the total number of comparisons made by CHK_{REV} are proportional to the total number of bracelets $\mathbf{B}_k(n, d)$ then our algorithm is CAT.

5. SUMMARY

In this paper, we develop an efficient scheme to generate binary bracelets with fixed density. We prove that in case of binary bracelets our algorithm works in constant amortized time and takes asymptotic linear space. As a future work, we aim to prove theoretically that our algorithm works in CAT for arbitrarily large alphabet size. Furthermore, we also want to develop schemes for listing other restricted classes of bracelets. The scheme has been implemented in C and can be obtained from the authors upon request.

ACKNOWLEDGEMENTS

This work is partially funded by FAST-NU under the grant 11L-270/NU-R/11. The authors would like to thank Dr. J. Sawada for his useful comments in the analysis part of this paper.

REFERENCES

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

[2] P. Emmel and R. Hersch. Exploring ink spreading. *In Proceedings of the 8th IS and T/SID Color Imaging Conference: Color Science and Engineering*, (2000).

[3] J. Sawada. Generating bracelets in constant amortized time. *SIAM J. Computing*, 31:259 – 268, (2001).

[4] J. Sawada. A fast algorithm to generate necklaces with fixed content. *Theoretical Computer Science*, 301:447 – 489, (2003).

[5] D. E. Knuth. *The Art of Computer Programming Volume 4A*. Addison-Wesley, (2011).

[6] P. Lisonek. Computer assisted studies in algebraic combinatorics. *Dissertation*, (1994).

[7] F. Ruskey, C. Savage, and T. Wang. Generating necklaces. *Journal of Algorithms*, 13:3247–3259, (1992).

[8] F. Ruskey and J. Sawada. An efficient algorithm for generating necklaces of fixed density. *SIAM J. Computing*, 29:671 – 684, (1999).

[9] F. Ruskey. *Combinatorial Generation*. Working version, (2003).

[10] V. Vajnovszki. Gray code order for Lyndon words. *Discrete Mathematics and Theoratical Computer Science*, 9:145 – 152, (2007).

[11] V. Vajnovszki. More restrictive Gray codes for necklaces and Lyndon words. *Information Processing Letters*, 106:96 – 99, (2008).

[12] M. Weston and V. Vajnovszki. Gray codes for necklaces and Lyndon words of arbitrary base. *Pure Math. Appl. (PU.M.A)*, 17:175–182, (2006).

APPENDIX - PSEUDO CODES

Algorithm: Initialize

- Initialize $c_1c_2 \cdots c_n = 0^n$ and $s_1s_2 \cdots s_n = 0^n$
- Initialize $a(d) := n$; $s(n) := d$ and $j := n - d$
    - repeat untill $j \geq (n-1)/d$
        - Initialize $u := j$; $a(1) := u + 1$; $s(u+1) = 1$ and $i := 1$
        - Repeat $k - 1$ times
            - $c(u+1) := i$
            - $BF_u(1, u, false)$
            - $c(u+1) := 0$
            - increment i by 1
        - increment j by 1 and $s(u+1) := 0$

Algorithm: $BF_u(t, r, CMP(\gamma, a(t+1), r))$

- if $a(t) > (n - r)/2 + r$ then
    - Compute $CMP(\gamma, a(t), r)$
    - if $t = d - 1$ then
        - Compute value for $c_{a(d)}$ and APND(γ, $c_{a(d)}$)
        - if $u = ad - ad-1 - 1$ then
            - Compute $CHK_{REV}(\gamma)$
            - if $CHKREV(\gamma) = -1$ then return
            - Compute $CMP(\gamma, a(d), r)$
            - if $CMP(\gamma, a(d), r) = false$ then $\gamma \in \mathbf{B}_k(n, d)$
            - $c_{a(d)} := 0$ and return
- Compute valid $c_{a(t+1)}$ and APND(γ, $c_{a(t+1)}$)
- $s_{a(t+1)} := a(t+1)$
- if $u = a(t+1) - a(t) - 1$ then
    - Compute $CHKREV(\gamma)$
    - CASE 1: $CHKREV(\gamma) = 0$
        - compute $BF_u(t + 1, r, CMP(\gamma, a(t), r))$
    - CASE 2: $CHKREV(\gamma) = 1$
        - $r := a(t+1) - 1$ and Compute $BF_u(t + 1, r,$ false)
    - CASE 3: $CHKREV(\gamma) = -1$
        - return
- else Compute $BF_u(t + 1, r, CMP(\gamma, a(t), r))$
- $c_{a(d)} := 0$