Generating Necklaces

Frank Ruskey *
Department of Computer Science
University of Victoria, P. O. Box 1700
Victoria, B. C. V8W 2Y2 CANADA

Carla Savage †
Terry MinYih Wang
Department of Computer Science
North Carolina State University, Box 8206
Raleigh, NC 27612-8206

January 15, 1991

Abstract

A k-color, n-bead necklace is an equivalence class of k-ary n-tuples under rotation. In this paper, we analyze an algorithm due to Fredricksen, Kessler, and Maiorana (FKM), to show that necklaces can be generated in constant average time. We also present a new approach to generating necklaces which we conjecture can be implemented in constant worst case time.

The FKM algorithm generates a list of n-tuples which includes, among other things, the lexicographically smallest element of each k-color, n-bead necklace. Previously it had been shown only that the list contains at most $O(n \cdot N(k,n))$ elements, where $N(k,n)$ is the number of k-color, n-bead necklaces, and that successive elements can be generated in worst case time $O(n)$, giving a bound of $O(n^2 \cdot N(k,n))$ on the time for the algorithm. We show that the number of elements generated by the FKM algorithm approaches $(k/(k-1)) \cdot N(k,n)$ and the total time is only $O(N(k,n))$. A by-product product of our analysis is a precise characterization of the list generated by FKM, which makes a recursive description possible.

1 Introduction

In this paper, we consider the problem of efficiently generating necklaces. A *necklace of n beads in k colors* is an equivalence class of k-ary n-tuples under rotation. That is, if σ

*Research supported by the Natural Sciences and Engineering Research Council of Canada under grant A3379.

†Research supported by the National Science Foundation Grant No. CCR8906500, the National Security Agency Grant No. MDA904-H-1025 and DIMACS.

denotes the rotation

$$\sigma(x_1 \ldots x_n) = x_2 \ldots x_n x_1$$

then the k-ary n-tuples x and y are in the same equivalence class or *necklace* if and only if $\sigma^j(x) = y$ for some integer j.

Let $N(k,n)$ denote the number of n-bead necklaces in k colors. It is known that

$$N(k,n) \;=\; \frac{1}{n}\sum_{d|n}\phi(d)k^{n/d} \;=\; \frac{1}{n}\sum_{i=1}^{n}k^{\gcd(i,n)} \tag{1}$$

(See, e.g., [Ma]) where ϕ is the Euler totient function, so that

$$\frac{1}{n}(k^n) \;\leq\; N(n) \;\leq\; \frac{1}{n}(k^n + (n-1)k^{n/2}). \tag{2}$$

So, in applications where every rotation of a given k-ary n-tuple represents the same phenomenon, it would be more efficient to work with only one representative of each necklace, rather than all k^n of the n-tuples. A necklace is thus identified with its representative.

A simple and elegant algorithm was proposed in [FrMa] and [FrKe] to generate for each necklace the lexicographically smallest element. We will refer to this as the *FKM algorithm.* A disadvantage of the FKM algorithm is that there can be gaps in which as many as $\lfloor (n-1)/2 \rfloor$ non-necklaces are examined between any two necklaces generated. For the case $k = 2$ the authors of [FrKe] prove an upper bound of $\lfloor (n-1)/2 \rfloor N(2,n)$ on the number of n-tuples generated and examined by the algorithm. The time spent generating and checking each element is $O(n)$, giving an overall time for the algorithm of $O(n^2 \cdot N(2,n))$. However, this does not establish an asymptotic improvement over the direct approach of generating *every* k-ary n-tuple and checking each one to see if it is the representative of its necklace, at a cost of time $O(n)$ per check.

In this paper, we present a new analysis of the FKM algorithm to show that it can generate all necklaces, for given k and n, in *constant average time.* Our analysis is based on a new characterization of the list of k-ary n-tuples examined by the FKM algorithm. This new characterization is used to give a more illuminating proof of the algorithm, as well as a recursive description of the number of n-tuples examined and the total time required. As a consequence, we show that the number of n-tuples is $O(N(k,n))$ and, in fact, approaches $(k/(k-1)) \cdot N(k,n)$ in the limit. We also show a bound of $O(N(k,n))$ on the total time required to generate all necklaces. Thus, although there can still be gaps of $\Omega(n)$ between successive necklaces generated, the *average* time per necklace is constant.

We also describe in this paper a new algorithm which generates the lexicographically smallest element in each necklace. The number of k-ary n-tuples generated and examined by the new algorithm is at most *twice* the total number of necklaces. This algorithm has the advantage that it can be implemented so that there is a gap of at most one non-necklace examined between any two necklaces generated. However, in the current implementation, for each n-tuple examined, time $O(n)$ is spent to check whether it is a necklace, i.e., the lexicographically smallest of all of its rotations. This gives a total time of $O(n \cdot N(k,n))$. We conjecture that there is a more efficient scheme to handle and/or avoid necklace-testing which will admit an $O(N(k,n))$-time implementation.

In Section 2, we describe the FKM algorithm, prove its correctness, and analyze it, based on a new characterization of the n-tuples which it examines. In Section 3, we describe our new necklace algorithm, prove that it examines no more than twice the number of necklaces, and discuss how to implement it to remove gaps. Section 4 contains concluding remarks. In the remainder of this section, we introduce notation and terms.

We will regard a k-ary n-tuple, α, as a string over the alphabet $\Sigma_k = \{0, \ldots, k-1\}$. Σ_k^n is the set of all length-n strings over Σ_k and Σ_k^* is the set of all strings (of any length) over Σ_k, including the empty string λ. For $\alpha \in \Sigma_k^n$ and $1 \le i \le n$, we use α_i to denote the i-th symbol of α so that $\alpha = \alpha_1 \ldots \alpha_n$. For $\alpha, \beta \in \Sigma_k^n$, $\alpha\beta$ denotes the concatenation of α and β and for $a \in \Sigma_k$, a^t denotes the string of length t in which every symbol is a. For $\alpha, \beta \in \Sigma_k^*$, we use $\alpha \le \beta$ ($\alpha < \beta$) to denote that α precedes (strictly precedes) β in lexicographic order, and similarly for $\ge$ and $>$.

For the remainder of this paper, we use *necklace* to denote the lexicographically smallest member of an equivalence class of Σ_k^n under rotation. Formally,

Definition 1 *The string $\alpha \in \Sigma_k^n$ is a* **necklace** *if $\alpha_1 \ldots \alpha_n \le \alpha_i \ldots \alpha_n \alpha_1 \ldots \alpha_{i-1}$ for all $1 \le i \le n$.*

2 The FKM Algorithm

For a given n and k, the FKM algorithm generates a list, $\mathcal{F}(k,n)$ consisting of a certain subset of Σ_k^n in lexicographic order. The list $\mathcal{F}(k,n)$begins with the string 0^n and ends with $(k-1)^n$. For a given α on $\mathcal{F}(k,n)$, the successor of α, $succ(\alpha)$, is obtained from α as follows.

$k = 2$, $n = 6$

000000	000110 •	001101	011011
000001	000111	001110 •	011101 •
000010 •	001001	001111	011110 •
000011	001010 •	010101	011111
000100 •	001011	010110 •	111111
000101	001100 •	010111	

$k = 3$, $n = 4$

0000	0022	0122	1111
0001	0101	0202	1112
0002	0102	0210 •	1121 •
0010 •	0110 •	0211	1122
0011	0111	0212	1212
0012	0112	0220 •	1221 •
0020 •	0120 •	0221	1222
0021	0121	0222	2222

Figure 1: The FKM algorithm (read down columns).

Definition 2 *For* $\alpha < (k-1)^n$, $\mathrm{succ}(\alpha) = (\alpha_1 \ldots \alpha_{i-1}(\alpha_i + 1))^t \alpha_1 \ldots \alpha_j$, *where* i *is the largest integer* $1 \le i \le n$ *such that* $\alpha_i < k-1$ *and* t, j *are such that* $ti + j = n$ *and* $j < i$.

It is shown in [FrMa] that no necklace can lie strictly between two elements of $\mathcal{F}(k,n)$, so that all necklaces appear on $\mathcal{F}(k,n)$. Thus, discarding non-necklaces of $\mathcal{F}(k,n)$ would result in a list of all necklaces in increasing order. Figure 1 shows examples of lists $\mathcal{F}(k,n)$ where non-necklaces are indicated by bullets. Figure 2 illustrates how a gap of length $\lfloor (9-1)/2 \rfloor = 4$ can occur between successive necklaces when $k = 2$ and $n = 9$.

It is claimed in [FrKe] that $succ(\alpha)$ is a necklace if and only if the 'i' of Definition 2 is a divisor of n and we prove this in Lemma 3. Incorporating this test for necklace-checking, the entire algorithm can be summarized by the PASCAL code in Figure 3.

In order to analyze the FKM algorithm, we first show how to characterize the elements of $\mathcal{F}(k,n)$. This will result in a more illuminating proof of correctness for the algorithm and then allow us to bound the size of the list $\mathcal{F}(k,n)$ and the total amount of work done.

.
.
.
011101111
011110111 •
011111011 •
011111101 •
011111110 •
011111111
.
.
.

Figure 2: Gaps in the FKM algorithm for $k = 2$, $n = 9$.

FKM Algorithm

```
for i := 0 to n do a[i] := 0;
Print(a);
i := n;
repeat
   a[i] := a[i] + 1;
   for j := 1 to n-i do a[j+i] := a[j];
   if n mod i = 0 then Print(a);
   i := n;
   while a[i] = k-1 do i := i-1;
until i = 0;
```

Figure 3: The FKM Algorithm (note $a[0] = 0$.)

Definition 3 *The string $\alpha \in \Sigma_k^n$ is a* **pre-necklace** *if $\alpha\beta$ is a necklace for some $\beta \in \Sigma_k^*$.*

We will eventually show, in Theorem 2, that the elements of $\mathcal{F}(k, n)$ are exactly the pre-necklaces in Σ_k^n.

Theorem 1 *(Characterization of pre-necklaces) $\alpha \in \Sigma_k^*$ is a pre-necklace if and only if $\alpha_i \dots \alpha_n \geq \alpha_1 \alpha_{n-i+1}$ for every i with $1 \leq i \leq n$.*

Proof. If α is a pre-necklace, then by definition, $\alpha\beta$ is a necklace for some $\beta \in \Sigma_k^*$. Then for $1 \leq i \leq n$,

$$\alpha_i \dots \alpha_n \beta \alpha_1 \dots \alpha_{i-1} \geq \alpha_1 \dots \alpha_n \beta,$$

so $\alpha_i \dots \alpha_n \geq \alpha_1 \alpha_{n-i+1}$.

On the other hand, if

$$\alpha_i \dots \alpha_n \geq \alpha_1 \alpha_{n-i+1} \tag{3}$$

for all $i : 1 \leq i \leq n$, we claim that $\alpha(k-1)^n$ is a necklace. Otherwise, for some i,

$$\alpha_i \dots \alpha_n (k-1)^n \alpha_1 \dots \alpha_{i-1} < \alpha(k-1)^n \tag{4}$$

However, by (3) and (4), $\alpha_i \dots \alpha_n = \alpha_1 \alpha_{n-i+1}$, that is,

$$\alpha_j = \alpha_{j+i-1} \text{ for } 1 \leq j \leq n-i+1. \tag{5}$$

But then to satisfy (4),

$$(k-1)^{i-1} \leq \alpha_{n-i+2} \dots \alpha_n$$

which can hold only if

$$\alpha_i = k-1 \text{ for } 1 \leq n-i+2 \leq n. \tag{6}$$

But (5) and (6) together imply $\alpha = (k-1)^n$ which contradicts (4). So, $\alpha(k-1)^n$ is a necklace and therefore α is a pre-necklace. □

Corollary 1 *If $\alpha \in \Sigma_k^n$ is a pre-necklace, $\alpha(k-1)$ is a pre-necklace.*

Proof. For any $i : 1 \leq i \leq n$, if α is a pre-necklace, then by Theorem 1, $\alpha_i \dots \alpha_n \geq \alpha_1 \alpha_{n-i+1}$. Therefore, $\alpha_i \dots \alpha_n (k-1) \geq \alpha_1 \alpha_{n-i+2}$. Furthermore, the last symbol of $\alpha(k-1)$ is $k-1$ which satisfies $k-1 \geq \alpha_1$, so by Theorem 1, $\alpha(k-1)$ is a pre-necklace. □

Definition 4 *The string $\alpha \in \Sigma_k^n$ is a* **prime necklace** *if $\alpha_1 \ldots \alpha_n < \alpha_i \ldots \alpha_n \alpha_1 \ldots \alpha_{i-1}$ for all $1 < i \leq n$.*

Lemma 1 *For $\alpha \in \Sigma_k^n$ and $\alpha_n < k-1$, if α is a pre-necklace, then $\alpha_1 \ldots \alpha_{n-1}(\alpha_n + 1)$ is a prime necklace.*

Proof. Since α is a pre-necklace, for any $1 \leq i \leq n$,

$$\alpha_i \ldots \alpha_n \geq \alpha_1 \ldots \alpha_{n-i+1}.$$

Thus,

$$\alpha_i \ldots \alpha_{n-1}(\alpha_n + 1) > \alpha_1 \ldots \alpha_{n-i+1}$$

so that

$$\alpha_i \ldots \alpha_{n-1}(\alpha_n + 1)\alpha_1 \ldots \alpha_{i-1} > \alpha_1 \ldots \alpha_{n-1}(\alpha_n + 1)$$

which means $\alpha_1 \ldots \alpha_{n-1}(\alpha_n + 1)$ is a prime necklace. □

Lemma 2 *If α is a necklace, then α^t is a necklace for $t \geq 1$.*

Proof. This is clear for $t = 1$. For $t \geq 1$, any rotation of α^t has the form $\gamma\alpha^{t-1}\beta$ for some $\gamma, \beta \in \Sigma_k^*$ with $\alpha = \beta\gamma$. But since α is a necklace, $\gamma\beta \geq \beta\gamma$, so

$$\gamma\alpha^{t-1}\beta = (\gamma\beta)^t \geq (\beta\gamma)^t = \alpha^t$$

and α^t is a necklace. □

Lemma 3 *If $\alpha \in \Sigma_k^*$ is a prime necklace and $\alpha = \beta\gamma$, for $\beta, \gamma \in \Sigma_k^*, \gamma \neq \lambda$, then for any $t \geq 1$,*

(a) $\alpha^t\beta$ is a pre-necklace and

(b) $\alpha^t\beta$ is a necklace if and only if $|\beta| = 0$.

Proof. Since α is a necklace, α^{t+1} and α^t are both necklaces by Lemma 2. Thus, $\alpha^t\beta$ is a pre-necklace (since $\alpha^t\beta\gamma$ is a necklace) and, if $|\beta| = 0$, a necklace. If $|\beta| > 0$, then $\beta\gamma < \gamma\beta$ since α is prime and therefore

$$\beta\alpha^t = \beta(\beta\gamma)^t < \beta(\gamma\beta)^t = (\beta\gamma)^t\beta = \alpha^t\beta,$$

so that $\alpha^t\beta$ is not a necklace. □

The FKM algorithm, for a given n, generates a list $\mathcal{F}(n,k)$, of strings in Σ_k^n, beginning with 0^n. For $\alpha \neq (k-1)^n$, $\mathrm{succ}(\alpha)$, the successor of α on $\mathcal{F}(k,n)$, is described in Definition 2 as

$$\mathrm{succ}(\alpha) = (\alpha_1 \ldots \alpha_{i-1}(\alpha_i + 1))^t \alpha_1 \ldots \alpha_j$$

where $i \geq 0$ is the largest integer such that $\alpha_i < k-1$ and t, j are such that $ti + j = n$ and $j < i$. We can now show the following.

Theorem 2 *$\mathcal{F}(k,n)$ is a list of all pre-necklaces in Σ_k^n in lexicographic order.*

Proof. We first show by induction that every element of $\mathcal{F}(k,n)$ is a pre-necklace. This is clearly true for the first element, 0^n. If pre-necklace $\alpha \neq (k-1)^n$ appears on $\mathcal{F}(k,n)$, then

$$\mathrm{succ}(\alpha) = (\alpha_1 \ldots \alpha_{i-1}(\alpha_i + 1))^t \alpha_1 \ldots \alpha_j$$

for some j, k, t where

$$\alpha = (\alpha_1 \ldots \alpha_{i-1}\alpha_i(k-1)^{n-i})$$

and $\alpha_i < k-1$. Since α is a pre-necklace, so is $\alpha_1 \ldots \alpha_{i-1}\alpha_i$, and therefore $\alpha_1 \ldots \alpha_{i-1}(\alpha_i+1)$ is a prime necklace by Lemma 1. Thus, $\mathrm{succ}(\alpha)$ is a pre-necklace by Lemma 3.

Clearly, $\alpha < \mathrm{succ}(\alpha)$ for all $\alpha < (k-1)^n$ on $\mathcal{F}(k,n)$. To show every pre-necklace in Σ_k^n is on $\mathcal{F}(k,n)$, suppose ρ is the lexicographically smallest pre-necklace which does not appear on $\mathcal{F}(k,n)$. Then $\rho \neq (k-1)^n$ since $(k-1)^n = \mathrm{succ}((k-2)(k-1)^{n-1})$ and $(k-2)(k-1)^{n-1}$ would be a pre-necklace smaller than ρ. Thus there must be a pre-necklace α on $\mathcal{F}(k,n)$ such that $\alpha < \rho < \mathrm{succ}(\alpha)$. Let i be the largest integer such that $\alpha_i < k-1$. Then

$$\alpha = \alpha_1 \ldots \alpha_{i-1}\alpha_i(k-1)^{n-i}$$

and

$$\mathrm{succ}(\alpha) = (\alpha_1 \ldots \alpha_{i-1}(\alpha_i + 1))^t \alpha_1 \ldots \alpha_j$$

where $it + j = n$ and $j < i$. Then since $\alpha < \rho < \mathrm{succ}(\alpha)$, $\rho_x = \alpha_x$ for $1 \leq x \leq i-1$ and $\rho_i = \alpha_i + 1$. So, it is possible to find the largest $s \geq 1$ such that

$$\rho = (\alpha_1 \ldots \alpha_{i-1}(\alpha_i + 1))^s \beta \text{ for some } \beta \in \Sigma_k^*.$$

If $s = t$, then $\beta < \alpha_1 \ldots \alpha_j$. If $s \leq t$, $\beta_1 \ldots \beta_i < \alpha_1 \ldots \alpha_{i-1}(\alpha_i + 1)$. In either case, $\beta < \rho_1 \ldots \rho_{|\beta|}$, contradicting that ρ is a pre-necklace. $\square$

We now analyze the FKM algorithm, first by finding the size $|\mathcal{F}(k,n)|$ of the list generated.

Lemma 4

$$(N(k,n+1)-1)/(k-1) \leq |\mathcal{F}(k,n)| \leq |\mathcal{F}(k,n-1)| + N(k,n)$$

Proof. Note that by Theorem 2, each β on $\mathcal{F}(k,n)$ has the form $\beta = \alpha a$ for some $a \in \Sigma_k$ and α on $\mathcal{F}(k,n-1)$. If $a < k-1$, by Lemma 3(b), $succ(\alpha)$ will be a necklace. Thus

$$|\{\beta \text{ on } \mathcal{F}(k,n) : \beta_n < k-1\}| \leq N(k,n) \tag{7}$$

By Corollary 1, $\alpha(k-1)$ is on $\mathcal{F}(k,n)$ for every α on $\mathcal{F}(k,n-1)$, so

$$|\{\beta \text{ on } \mathcal{F}(k,n) : \beta_n = k-1\}| = |\mathcal{F}(k,n-1)| \tag{8}$$

The second inequality claimed in the lemma now follows from (7) and (8).

Note that each $\alpha \neq 0^n$ on $\mathcal{F}(k,n)$ is a prefix of at most $k-1$ *necklaces* on $\mathcal{F}(k,n+1)$ and 0^n is the prefix of k necklaces. Thus,

$$N(k,n+1) \leq (k-1)|\mathcal{F}(k,n)| + 1$$

and the first inequality of the lemma follows. □

If we define $|\mathcal{F}(k,0)| = 0$, then from the recursion of the second inequality in Lemma 4, we get the following:

$$|\mathcal{F}(k,n)| \quad \leq \quad \sum_{i=1}^{n} N(k,n) \tag{9}$$

In Theorem 3, we will show that

$$\lim_{n\to\infty} \frac{|\mathcal{F}(k,n)|}{N(k,n)} = \frac{k}{k-1}$$

so that, in particular, $|\mathcal{F}(k,n)|$ is $O(N(k,n))$. First a technical lemma is needed.

Lemma 5 *For all $k \geq 2$,*

$$\text{a.} \quad \lim_{n\to\infty} \frac{\sum_{i=1}^{n}(k^i/i)}{k^n/n} = \frac{k}{k-1}.$$

$$\text{b.} \quad \lim_{n\to\infty} \frac{\sum_{i=1}^{n}\sum_{j=1}^{i}(k^j/j)}{k^n/n} = \frac{k^2}{(k-1)^2}.$$

Proof: (a.) The ratio can be rewritten as

$$\frac{\sum_{i=1}^{n}(k^i/i)}{k^n/n} = \sum_{j=0}^{n-1}\frac{n}{(n-j)k^j} = \sum_{j=0}^{n-1}\frac{1}{k^j} + \sum_{j=0}^{n-1}\frac{j}{(n-j)k^j}$$

The first term on the right converges to $k/(k-1)$. We show the second term converges to zero. First, note that for $t \geq 3$,

$$\frac{t}{t+1} > \frac{1}{\sqrt{k}} \tag{10}$$

since $k \geq 2$. Let

$$f(j) = \frac{j}{(n-j)k^j}.$$

Then

$$\frac{f(j)}{f(j+1)} = k \cdot \frac{j}{j+1} \cdot \frac{n-j-1}{n-j}.$$

By (10), the last two factors are greater than $1/\sqrt{k}$ for $3 \leq j \leq n-4$, so $f(j)$ is strictly decreasing on the range $3 \leq j \leq n-3$.

Since $f(j)$ converges to 0 for each fixed j,

$$\lim_{n\to\infty}\sum_{j=0}^{n-1} f(j) = \lim_{n\to\infty}\sum_{j=3}^{n-3} f(j)$$

and

$$\sum_{j=3}^{n-3} f(j) = \sum_{j=3}^{\lfloor\sqrt{n}\rfloor} f(j) + \sum_{j=\lfloor\sqrt{n}\rfloor+1}^{n-3} f(j) \leq \sqrt{n}\cdot\frac{3}{(n-3)k^3} + (n-\sqrt{n})\cdot\frac{\sqrt{n}}{(n-\sqrt{n})k^{\sqrt{n}}}$$

and each of these terms converges to 0.

(b.) The ratio can be rewritten as

$$\frac{\sum_{i=1}^{n}\sum_{j=1}^{i}(k^j/j)}{k^n/n} = \sum_{j=0}^{n-1}\frac{n(j+1)}{(n-j)k^j} = \sum_{j=0}^{n-1}\frac{j+1}{k^j} + \sum_{j=0}^{n-1}\frac{j(j+1)}{(n-j)k^j}$$

The first term on the right converges to $k^2/(k-1)^2$. We show the second term converges to zero. First, note that for $t \geq 5$,

$$\frac{t}{t+2} > \frac{1}{\sqrt{k}} \tag{11}$$

since $k \geq 2$. Let

$$g(j) = \frac{j(j+1)}{(n-j)k^j}.$$

Then

$$\frac{g(j)}{g(j+1)} = k \cdot \frac{j}{j+2} \cdot \frac{n-j-1}{n-j}.$$

By (11), the last two factors are greater than $1/\sqrt{k}$ for $5 \le j \le n-6$, so $f(j)$ is strictly decreasing on the range $5 \le j \le n-5$.

Since $g(j)$ converges to 0 for each fixed j,

$$\lim_{n\to\infty} \sum_{j=0}^{n-1} f(j) = \lim_{n\to\infty} \sum_{j=5}^{n-5} f(j)$$

and

$$\sum_{j=5}^{n-5} g(j) \;=\; \sum_{j=5}^{\lfloor\sqrt{n}\rfloor} g(j) \;+\; \sum_{j=\lfloor\sqrt{n}\rfloor+1}^{n-5} g(j) \;\le\; \sqrt{n}\cdot\frac{5\cdot 6}{(n-5)k^5} \;+\; (n-\sqrt{n})\cdot\frac{\sqrt{n}(\sqrt{n}+1)}{(n-\sqrt{n})k^{\sqrt{n}}}$$

and each of these terms converges to 0. □

Theorem 3

$$\lim_{n\to\infty} \frac{|\mathcal{F}(k,n)|}{N(k,n)} = \frac{k}{k-1}$$

Proof. We bound the ratio $|\mathcal{F}(k,n)|/N(k,n)$ from above and below by functions which both converge to $k/(k-1)$.

For the upper bound, from (2) and (9),

$$|\mathcal{F}(k,n)| \;\le\; \sum_{i=1}^{n}(\frac{k^i}{i} \;+\; \frac{i-1}{i}k^{i/2}) \;\le\; \sum_{i=1}^{n}\frac{k^i}{i} \;+\; \sum_{i=1}^{n}k^{i/2}$$

and

$$N(k,n) \ge \frac{k^n}{n}.$$

So,

$$\frac{|\mathcal{F}(k,n)|}{N(k,n)} \;\le\; \frac{\sum_{i=1}^{n}(k^i/i)}{k^n/n} + \frac{(k^{\frac{n+1}{2}}-1)/(k^{\frac{1}{2}}+1)}{k^n/n}.$$

On the right side of the inequality, clearly the second term converges to 0 and by Lemma 5, the first term converges to $k/(k-1)$.

For the lower bound, from Lemma 4 and (2),

$$(k-1)|\mathcal{F}(k,n)| \ge N(k,n+1) - 1 \ge \frac{k^{n+1}}{n} - 1.$$

¿From (2) then,

$$(k-1)\frac{|\mathcal{F}(k,n)|}{N(k,n)} \geq \frac{(k^{n+1}/n)-1}{k^n/n+(n-1)k^{n/2}}$$

and the right hand side converges to k. This gives the lower bound of $k/(k-1)$ on $\lim_{n\to\infty}|\mathcal{F}(k,n)|/N(k,n)$. □

To analyze the time required by the FKM algorithm, note that the work done, once the initial string 0^n has been generated, consists of

(i) Finding for each α on $\mathcal{F}(k,n)$ the largest i such that $\alpha_i < k-1$.

(ii) Copying $\alpha_1 \ldots \alpha_{i-1}(\alpha_i+1)$ repeatedly into positions $i+1,\ldots,n$ to get $succ(\alpha)$, and

(iii) Testing whether i divides n to determine whether $succ(\alpha)$ is a necklace by Lemma 3(b).

We can tabulate the divisors of n at the beginning of the algorithm, so the time, for each α, to find $succ(\alpha)$ is proportional to $n-i+1$ For each α on $\mathcal{F}(k,n)$ define

$$w(\alpha) = n-i+1$$

where $\alpha_i < k-1$ but $\alpha_j = k-1$ for $j > i$. Note that

$$w(\alpha) = 1 \quad \text{if} \quad \alpha_n < k-1 \qquad \text{and} \qquad w(\alpha(k-1)) = w(\alpha)+1. \tag{12}$$

The total time for the FKM algorithm for given k and n is bounded by a constant times $W(k,n)$ where

$$W(k,n) = \sum_{\alpha \text{ on } \mathcal{F}(k,n)} w(\alpha)$$

Lemma 6 $W(k,n) = W(k,n-1) + |\mathcal{F}(k,n)|$.

Proof. Every β on $\mathcal{F}(k,n)$ either has the form $\alpha(k-1)$ for some α on $\mathcal{F}(k,n)$ or has $\beta_n < k-1$, so

$$W(k,n) \;=\; \sum_{\beta \text{ on } \mathcal{F}(k,n)} w(\beta) = \sum_{\alpha(k-1) \text{ on } \mathcal{F}(k,n)} w(\alpha(k-1)) \;+\; \sum_{\beta \text{ on } \mathcal{F}(k,n);\ \beta_k<k-1} w(\beta).$$

By (12) this gives

$$W(k,n) \;=\; \sum_{\alpha(k-1) \text{ on } \mathcal{F}(k,n)} w(\alpha) \;+\; \sum_{\alpha(k-1) \text{ on } \mathcal{F}(k,n)} 1 \;+\; \sum_{\beta \text{ on } \mathcal{F}(k,n);\ \beta_k<k-1} 1.$$

Since $\alpha(k-1)$ is on $\mathcal{F}(k,n)$ if and only if α is on $\mathcal{F}(k,n-1)$ (by Corollary 1 and Theorem 2), the first term is $W(k,n-1)$. The sum of the second two terms is just $|\mathcal{F}(k,n)|$. $\Box$

If we define $W(k,0)=0$, then from the recurrence of Lemma 6 we have

$$W(k,n) \;=\; \sum_{i=1}^{n} |\mathcal{F}(k,i)|. \tag{13}$$

The following theorem establishes that the total rumming time of the FKM algorithm is linear in the number of necklaces.

Theorem 4 *For any $c > (k/(k-1))^2$, n can be chosen large enough so that*

$$W(k,n) \le c \cdot N(k,n).$$

Proof. By (13) and (9),

$$W(k,n) \;=\; \sum_{i=1}^{n} |\mathcal{F}(k,i)| \;\le\; \sum_{i=1}^{n}\sum_{j=1}^{i} N(k,j).$$

So, by (2),

$$\frac{W(k,n)}{N(k,n)} \;\le\; \frac{\sum_{i=1}^{n}\sum_{j=1}^{i}(k^j/j)}{k^n/n} \;+\; \frac{\sum_{i=1}^{n}\sum_{j=1}^{i}((i-1)/i)k^{i/2}}{k^n/n}.$$

The last term converges to 0 and by Lemma 5(b), the first term on the right hand side of the inequality converges to $(k/(k-1))^2$. $\Box$

3 A New Algorithm for Generating Necklaces

In this section, we describe a new algorithm for generating necklaces in which the number of strings examined is never more that twice the number of necklaces. For simplicity, we focus on the case for two colors, but the generalization to k colors is described in detail in [WaSa].

Our idea for generating necklaces of two-color beads was inspired by a result in [LiHiCa] that a certain variation on the shuffle-exchange graph is hamiltonian. (The graph of [LiHiCa] is actually the deBruijn graph, which is known to be hamiltonian.)

When $k=2$, the n-tuples are bit strings which we regard as elements of $\{0,1\}^n$. Define a function τ on $\{0,1\}^n$ by

$$\tau(x_1 \ldots x_n) = x_1 \ldots x_{n-1}\overline{x_n},$$

where $\overline{x_n}$ denotes the complement of the bit x_n. As in Section 1, $\sigma(x)$ denotes the rotation of string x one position left.

In the new algorithm, a tree of necklaces is generated as follows. Starting with the string $x = 0^n$ as root, we generate as the children of x all those *necklaces* of the form $\tau\sigma^j(x)$ for some j satisfying $1 \leq j \leq n-1$. This procedure is applied recursively to each child of x. We show first that every necklace will be generated.

Lemma 7 *Every n-bead necklace will appear in the tree with root 0^n.*

Proof. Let x be an n-bead necklace. We show by induction on t, the number of ones in x, that x is in the tree. If $t = 0$ then $x = 0^n$, which is the root of the tree. For $t > 0$, note that $x = \alpha 1$ for some $\alpha \in \{0,1\}^{n-1}$ since x must be the lexicographically smallest in its class. Thus $x = \tau\sigma(0\alpha)$. The representative y of the class containing 0α has $t-1$ ones, so must be in the tree, by induction. Since 0α is a rotation of y, $x = \tau\sigma^j(y)$, for some j, so x will be a child of y in the tree. □

Although at first glance it would appear that we need to examine *every* string of the form $\alpha 1$ with this procedure, this is not the case. We show in Theorem 5 below that if x, $\tau(x)$, and $\sigma(x)$ are not necklaces, then $\tau\sigma(x)$ is not a necklace. It follows for a *necklace* $y \neq 00\ldots01$ that if $\tau\sigma^j(y)$ is *not* a necklace for some j, then, $\tau\sigma^{j+1}(y)$ is also *not a necklace.*

As a result, from a necklace y, we generate the $\tau\sigma^j(y)$, starting with $j = 1$, until the first j is found for which $\tau\sigma^j(y)$ is not a necklace. By Theorem 5, we are guaranteed that at this point all children of y have been found. An example is given in Figure 4.

The proof of Theorem 5 is based on the following lemma.

Lemma 8 *If $x = 0^t 1\alpha 1$, where $t > 0$ and $\alpha \in \{0,1\}^*$, then*

(a) if there are $\beta, \gamma \in \{0,1\}^*$ such that $\alpha = \beta 0^{t+1}\gamma$ then x is not a necklace.

(b) if x is not a necklace, then there are $\beta, \gamma \in \{0,1\}^*$ such that $\alpha = \beta 0^t \gamma$.

Proof. In case (a), the string $0^{t+1}\gamma 1 0^t 1\beta$ is a rotation of x which precedes x in lexicographic order.

In case (b), let $\sigma^k(x)$ be a rotation of x with $\sigma^k(x) < x$. Then $\sigma^k(x) = 0^t\delta$ for some $\delta \in \{0,1\}^+$ (with $\delta < 1\alpha 1$.) Since $\sigma^k(x) \neq x$, α must contain 0^t as a contiguous substring. □

	$\bullet 00000$		
	$\bullet 00001$		
$\bullet \tau\sigma(00001) =$ 00011	$\bullet \tau\sigma^2(00001) =$ 00101	$\otimes \tau\sigma^3(00001) =$ 01001, not necklace	
		$\bullet \tau\sigma(00101) =$ 01011	$\otimes \tau\sigma^2(00101) =$ 10101, not a necklace
$\bullet \tau\sigma(00011) =$ 00111	$\otimes \tau\sigma^2(00011) =$ 01101, not a necklace		$\otimes \tau\sigma(01011) =$ 10111, not a necklace
$\bullet \tau\sigma(00111) =$ 01111	$\otimes \tau\sigma^2(00111) =$ 11101, not a necklace		
$\bullet \tau\sigma(01111) =$ 11111			

Figure 4: The tree of necklaces for $k = 2$, $n = 5$.

Two Color Necklaces

```
procedure search(y);
        output(y);
        done := false;
        while (not(done)) do
                y := σ(y);
                x := τ(y);
                if x is a necklace then search(x)
                        else done := true;

main;
output(00...00);
search(00...01);
```

Figure 5: New algorithm to generate n-bead necklaces in two colors.

Theorem 5 *For bitstring $x \in \{0,1\}^n$, if x, $\tau(x)$, and $\sigma(x)$ are not necklaces, then $\tau\sigma(x)$ is not a necklace.*

Proof. If $x = 1\alpha$, then $\tau\sigma(x) = \alpha 0$ which is not a necklace unless $\alpha = 00\ldots0$, in which case $\sigma(x) = 0\ldots01$ is a necklace.

Also, since x is not a necklace, $x \neq 0^n$ and $x \neq 0^{n-1}1$ so, we may assume that either (i) $x = 0^t1\alpha1$ or (ii) $x = 0^t1\alpha0$ for some $t > 0$ and $\alpha \in \{0,1\}^*$

In case (i), since x is not a necklace, by Lemma 8 (b), $\alpha = \beta 0^t \gamma$ for some $\beta, \gamma \in \{0,1\}^*$. But then $\tau\sigma(x) = 0^{t-1}1\alpha11$. So, by Lemma 8 (a), $\tau\sigma(x)$ is not a necklace.

In case (ii), $\tau(x) = 0^t1\alpha1$ and $\tau\sigma(x) = 0^{t-1}1\alpha01$. Since $\tau(x)$ is not a necklace, application of Lemma 2 shows as in case (i) that $\tau\sigma(x)$ is not a necklace. □

We summarize in Figure 5 the recursive algorithm for generating the lexicographically smallest representatives of the n-bead necklaces in two colors. This algorithm makes use of the fact that a *necklace* of the form $y = 1\alpha$ will not have any children. Note that by Theorem 5 and as discussed above, while generating the children of the necklace y in the tree, at most one n-tuple is examined which is *not* a necklace. Thus the total number of n-tuples examined is at most $2 \cdot N(2,n)$.

A recursive implementation will give rise to *gaps* in this algorithm: as necklaces are implicitly stacked, it could occur that for several consecutive necklaces on the stack, the next candidate "child" to be tested turns out to be a non-necklace. We can avoid this gap with a nonrecursive implementation in which a necklace y is pushed on the stack only after checking that the next n-tuple x to be examined when y is popped is actually a necklace. In this case, x can be saved on the stack with y to avoid generating and checking x twice.

Although avoiding gaps allows some hope for a worst case constant time implementation, currently, we spend $O(n)$ time per string generated to determine whether the string is a necklace. This makes the total time for the algorithm $O(n \cdot N(k,n))$. However, we conjecture that necklace-checking could be implemented in amortized constant time, for example by a scheme like the following. At each necklace, x, we store information, $I(x,j)$ which is sufficient for us to determine, perhaps in constant time, whether (i) $\tau\sigma^j(x)$ is a necklace and (ii) the information $I(\tau\sigma^j(x),1)$. After the entire subtree of $\tau\sigma^j(x)$ has been generated and searched recursively, the information at the node x is updated to give $I(x,j+1)$, based on $I(x,j)$ and the current information at node $\tau\sigma^j(x)$.

4 Concluding Remarks

We have defined a new set of objects, called *pre-necklaces*, as those strings which are prefixes of necklaces. We have shown that for given n and k, the ratio of the number of n-bead, k-color pre-necklaces to necklaces is bounded by a constant and in the limit this ratio is $k/(k-1)$. For given n and k, we showed that the FKM algorithm generates all n-bead, k-color necklaces in constant average time.

We have also described a new approach to generating necklaces which we conjecture can be implemented in constant average time, or even constant worst case time. If so, this algorithm could be competitive with the FKM algorithm for $k=2$: although in both algorithms the number of strings examined approaches $2 \cdot N(k,n)$ when $k=2$, the FKM algorithm approaches this limit from above, whereas the new algorithm approaches it from below.

We ask whether it is possible, by our new algorithm or another strategy, to generate necklaces in worst case constant time? As a variation, is it possible to list necklaces in some Gray-code like order ([Gi], [Jo], [Lu], [Ru], [Sa])?

Acknowledgement

Thanks to Herb Wilf (for suggesting this problem and for help with the proof of Lemma 5) and Pete Winkler (for helpful discussions on the FKM algorithm.)

References

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

[FrMa] H. Fredericksen and J. Maiorana, "Necklaces of beads in k colors and k-ary de Bruijn sequences," *Discrete Mathematics* 23, No. 3 (1978) 207-210.

[Gi] E. N. Gilbert, "Gray codes and paths on the n-cube," *Bell Systems Technical Journal* (1958) 815-826.

[Jo] S. M. Johnson, "Generation of permutations by adjacent transpositions," *Math. Comp.* 17 (1963) 282-285.

[LiHiCa] W. Liu, T. H. Hildebrandt, and R. Cavin III, "Hamiltonian cycles in the shuffle-exchange network," *IEEE Transactions on Computers* C-38, No. 5 (1989) 745-750.

[Lu] J. Lucas, "The rotation graph of binary trees is Hamiltonian," *Journal of Algorithms* 8 (1987) 503-535.

[Ma] G. Mackiw, *Applications of Abstract Algebra*, John Wiley and Sons, New York (1985).

[Ru] F. Ruskey, "Adjacent interchange generation of combinations," *Journal of Algorithms* 9 (1988) 162-180.

[Sa] C. Savage, "Gray code sequences of partitions," *Journal of Algorithms* 10, No. 4 (1989).

[WaSa] T. Wang and C. Savage, "A new algorithm for generating necklaces," Report TR-90-20, Department of Computer Science, North Carolina State University (1990).