Finding the Largest Fixed-Density Necklace and Lyndon word

Joe Sawada Patrick Hartman

April 18, 2017

Abstract

We present an $O(n)$ time algorithm for finding the lexicographically largest fixed-density necklace of length n. Then we determine whether or not a given string can be extended to a fixed-density necklace of length n in $O(n^2)$ time. Finally, we give an $O(n^3)$ algorithm that finds the largest fixed-density necklace of length n that is less than or equal to a given string. The efficiency of the latter algorithm is a key component to allow fixed-density necklaces to be ranked efficiently. The results are extended to find the largest fixed-density Lyndon word of length n (that is less than or equal to a given string) in $O(n^3)$ time.

1 Introduction

A *necklace* is the lexicographically smallest string in an equivalence class of strings under rotation. A *Lyndon word* is a primitive (aperiodic) necklace. The *density* of a binary string is the number of 1s it contains. Let $\mathbf{N}(n, d)$ denote the set of all binary necklaces with length n and density d. In this paper we present efficient algorithms for the following three problems:

1. finding the largest necklace in $\mathbf{N}(n, d)$,
2. determining if an arbitrary string is a prefix of some necklace in $\mathbf{N}(n, d)$, and
3. finding the largest necklace in $\mathbf{N}(n, d)$ that is less than or equal to a given binary string of length n.

The first problem can be answered in $O(n)$ time, which is applied to answer the second problem in $O(n^2)$ time, which in turn is applied to answer the third problem in $O(n^3)$ time. The third problem can also be solved for fixed-density Lyndon words in $O(n^3)$ time, which can immediately be used to find the largest fixed-density Lyndon word of a given length. Solving the third problem efficiently for both necklaces and Lyndon words is a key step in the first efficient algorithms to rank and unrank fixed-density necklaces and Lyndon words [2]. When there is no density constraint, the third problem is known to be solvable in $O(n^2)$-time; one such implementation is outlined in [10]. This problem was encountered in the first efficient algorithms to rank and unrank necklaces and Lyndon words discovered independently by Kopparty, Kumar, and Saks [5] and Kociumaka, Radoszewski and Rytter [4].

To illustrate these problems, consider the lexicographic listing of $\mathbf{N}(8, 3)$:

$$00000111, 00001011, 00001101, 00010011, 00010101, 00011001, 00100101.$$

The largest necklace in this set is 00100101. The string 0010 is a prefix of a necklace in this set, however, 010 is not. Given an arbitrary string $\alpha = 00011000$, the largest necklace in this set that is less than or equal to α is 00010101.

Fixed-density necklaces were first studied by Savage and Wang when they provided the first Gray code listing in [11]. Since then, an algorithm to efficiently list fixed-density necklaces was given by Ruskey and Sawada [8] and another efficient algorithm to list them in cool-lex Gray code order was given by Sawada and

Williams [9]. The latter algorithm leads to an efficient algorithm to construct a fixed-density de Bruijn sequence by Ruskey, Sawada, and Williams [7]. When equivalence is further considered under string reversal, an algorithm for listing fixed-density bracelets is given by Karim, Alamgir and Husnine [3].

The remainder of this paper is presented as follows. In Section 2, we present some preliminary results on necklaces and related objects. In Section 3, we present an $O(n)$-time algorithm to find the largest necklace in $\mathbf{N}(n, d)$. In Section 4, we present an $O(n^2)$-time algorithm to determine whether or not a given string is a prefix of a necklace in $\mathbf{N}(n, d)$. In Section 5, we present an $O(n^3)$-time algorithm to finding the largest necklace in $\mathbf{N}(n, d)$ that is less than or equal to a given string. These results on necklaces are extended to Lyndon words in Section 6.

2 Preliminaries

Let α be a binary string and let $lyn(\alpha)$ denote the length of the longest prefix of α that is a Lyndon word. A *prenecklace* is a prefix of some necklace. The following theorem by Cattell et al. [1] has been called *The Fundamental Theorem of Necklaces*:

Theorem 2.1 *Let $\alpha = a_1a_2\cdots a_{n-1}$ be a prenecklace over the alphabet $\Sigma = \{0, 1, \ldots, k-1\}$ and let $p = lyn(\alpha)$. Given $b \in \Sigma$, the string αb is a prenecklace if and only if $a_{n-p} \leq b$. Furthermore,*

$$lyn(\alpha b) \;=\; \begin{cases} p & \text{if } b = a_{n-p} \\ n & \text{if } b > a_{n-p} \quad \text{(i.e., } \alpha b \text{ is a Lyndon word).} \end{cases}$$

Corollary 2.2 *If αb is a prenecklace then $\alpha(b+1)$ is a Lyndon word.*

Corollary 2.3 *If $\alpha = a_1a_2\cdots a_n$ is a necklace then αa_1 is a prenecklace and αb is a Lyndon word for all $b > a_1$.*

The following is well-known property of Lyndon words by Reutenauer [6].

Lemma 2.4 *If α and β are Lyndon words such that $\alpha < \beta$ then $\alpha\beta$ is a Lyndon word.*

Corollary 2.5 *If α is a Lyndon word and β is a necklace such that $\alpha \leq \beta$ then $\alpha\beta^t$ is a necklace for $t \geq 1$.*

Proof. If $\alpha = \beta$ then clearly $\alpha\beta^t$ is a (periodic) necklace. If β is a Lyndon word, then the result follows from repeated application of Lemma 2.4. Otherwise $\beta = \delta^i$ for some Lyndon word δ and $i > 1$. Note that $\alpha \leq \delta$ because otherwise $\alpha > \beta$. If $\alpha < \delta$, then repeated application of Lemma 2.4 implies that $\alpha\delta^j$ is a Lyndon word for all $j \geq 0$. If $\alpha = \delta$, then clearly $\alpha\delta^j$ is a (periodic) necklace for all $j \geq 1$. In both cases, $\alpha\beta^t$ will be a necklace for all $t \geq 1$. $\square$

Lemma 2.6 *A k-ary string $\alpha = a_1a_2\cdots a_n$ over alphabet $\{0, 1, \ldots, k-1\}$ is a necklace if and only if $0^{t-a_1}10^{t-a_2}1\cdots 0^{t-a_n}1$ is a necklace for all $t \geq k-1$.*

Proof. ($\Rightarrow$) Assume α is a necklace. Let $\beta = 0^{t-a_1}10^{t-a_2}1\cdots 0^{t-a_n}1$ for some $t \geq k-1$. If β is not a necklace then there exists some $2 \leq i \leq n$ such that $0^{t-a_i}10^{t-a_{i+1}}1\cdots 0^{t-a_n}10^{t-a_1}1\cdots 0^{t-a_{i-1}}1 < \beta$. But this implies $a_ia_{i+1}\cdots a_na_1\cdots a_{i-1} < \alpha$, contradicting the assumption that α is a necklace. Thus β is a necklace. ($\Leftarrow$) Assume $\beta = 0^{t-a_1}10^{t-a_2}1\cdots 0^{t-a_n}1$ is a necklace for all $t \geq k-1$. If α is not a necklace then there exists some $2 \leq i \leq n$ such that $a_ia_{i+1}\cdots a_na_1\cdots a_{i-1} < \alpha$. But this implies that $0^{t-a_i}10^{t-a_{i+1}}1\cdots 0^{t-a_n}10^{t-a_1}1\cdots 0^{t-a_{i-1}}1 < \beta$, contradicting the assumption that β is a necklace. Thus α is a necklace. $\square$

Lemma 2.7 *A k-ary string $\alpha = a_1a_2\cdots a_n$ over alphabet $\{0,1,\ldots,k-1\}$ is a necklace if and only if $01^{t+a_1}01^{t+a_2}\cdots 01^{t+a_n}$ is a necklace for all $t \geq 0$.*

Proof. ($\Rightarrow$) Assume α is a necklace. Let $\beta = 01^{t+a_1}01^{t+a_2}\cdots 01^{t+a_n}$ for some $t \geq 0$. If β is not a necklace there exists some $2 \leq i \leq n$ such that $01^{t+a_i}01^{t+a_{i+1}}\cdots 01^{t+a_n}01^{t+a_1}\cdots 01^{t+a_{i-1}} < \beta$. But this implies $a_ia_{i+1}\cdots a_na_1\cdots a_{i-1} < \alpha$, contradicting the assumption that α is a necklace. Thus β is a necklace. ($\Leftarrow$) Assume $\beta = 01^{t+a_1}01^{t+a_2}\cdots 01^{t+a_n}$ is a necklace for all $t \geq 0$. If α is not a necklace there exists some $2 \leq i \leq n$ such that $a_ia_{i+1}\cdots a_na_1\cdots a_{i-1} < \alpha$. But this implies that $01^{t+a_i}01^{t+a_{i+1}}\cdots 01^{t+a_n}01^{t+a_1}\cdots 01^{t+a_{i-1}} < \beta$, contradicting the assumption that β is a necklace. Thus α is a necklace. $\square$

3 Finding the largest necklace with a given density

Let $\textsc{LargestNeck}(n,d)$ denote the lexicographically largest binary necklace in $\mathbf{N}(n,d)$.

Lemma 3.1 *Let $0 < d \leq n$ and $t = \lfloor \frac{n}{d} \rfloor$. Then*

$$\textsc{LargestNeck}(n,d) = 0^{t-b_1}10^{t-b_2}1\cdots 0^{t-b_d}1,$$

where $b_1b_2\cdots b_d = \textsc{LargestNeck}(d, d-(n \bmod d))$.

Proof. Since $d > 0$, $\alpha = \textsc{LargestNeck}(n,d)$ can be written as $0^{c_1}10^{c_2}1\cdots 0^{c_d}1$ where each $c_i \geq 0$. Let $x = d - (n \bmod d)$. Observe that $\alpha \geq (0^t1)^{d-x}(0^{t-1}1)^x \in \mathbf{N}(n,d)$ (it is a simple calculation to verify the length). Thus, $c_1 \leq t$, and moreover each $c_i \leq t$ since α is a necklace. Therefore α can be expressed as $0^{t-b_1}10^{t-b_2}1\cdots 0^{t-b_d}1$ for some string $\beta = b_1b_2\cdots b_d$ over the alphabet $\{0,1,\ldots,t\}$. By Lemma 2.6, β is a necklace. Suppose there is some largest $1 \leq i \leq d$ such that $b_i > 1$. Thus, each element of $b_{i+1}\cdots b_d$ must be in $\{0,1\}$. Since β is a necklace, each of its rotations $b_j\cdots b_db_1\cdots b_{j-1} \geq \beta$. Thus, we can deduce that if $j > i$ then $b_j\cdots b_db_1\cdots b_{j-1} > b_1b_2\cdots b_{i-1}$. This implies that $b_j\cdots b_db_1\cdots b_{i-1} > b_1b_2\cdots b_{i-1}$. Now consider $\gamma = b_1b_2\cdots b_{i-2}(b_{i-1}{+}1)b_{i+1}\cdots b_d$. Since $b_1b_2\cdots b_{i-1}$ is a prenecklace, $b_1b_2\cdots b_{i-2}(b_{i-1}{+}1)$ is a Lyndon word by Corollary 2.2. Thus any proper rotation of γ starting before b_{i+1} will be strictly greater than γ. Now consider a rotation of γ starting from b_j for $i+1 \leq j \leq d$. Observe that a rotation starting from b_j has prefix $b_j\cdots b_db_1\cdots b_{i-2}(b_{i-1}+1)$. We have already noted that $b_j\cdots b_db_1\cdots b_{i-1} > b_1b_2\cdots b_{i-1}$, and therefore the complete rotation of γ starting with b_j must be greater than γ. Since every proper rotation of γ is strictly greater than γ, γ is a necklace. However, this means that $\gamma(b_i-1)$ is also a necklace by Corollary 2.3 and hence by Lemma 2.6, $0^{b_1}10^{b_2}1\cdots 0^{b_{i-2}}10^{b_{i-1}+1}10^{b_{i+1}}1\cdots 0^{b_d}10^{b_i-1}1$ is also a necklace. But this contradicts $\alpha = \textsc{LargestNeck}(n,d)$. Therefore there is no $b_i > 1$ and hence β is a binary string. Now, since the necklace β is binary it must have density x. For any $b'_1b'_2\cdots b'_d \in \mathbf{N}(d,x)$ we have $\alpha' = 0^{t-b'_1}10^{t-b'_2}1\cdots 0^{t-b'_d}1 \in \mathbf{N}(n,d)$ by Lemma 2.6. Clearly, α' will be largest when $b'_1b'_2\cdots b'_d = \textsc{LargestNeck}(d,x)$. Thus, $\beta = \textsc{LargestNeck}(d,x)$. $\square$

Lemma 3.2 *Let $0 \leq d < n$ and $t = \lfloor \frac{n}{n-d} \rfloor$. Then*

$$\textsc{LargestNeck}(n,d) = 01^{t-1+b_1}01^{t-1+b_2}\cdots 01^{t-1+b_{n-d}},$$

where $b_1b_2\cdots b_{n-d} = \textsc{LargestNeck}(n-d, n \bmod (n-d))$.

Proof. Since $d < n$, $\alpha = \textsc{LargestNeck}(n,d)$ can be written as $01^{c_1}01^{c_2}\cdots 01^{c_{n-d}}$ where each $c_i \geq 0$. Let $x = n \bmod (n-d)$. Observe that $\alpha \geq (01^{t-1})^{n-d-x}(01^t)^x \in \mathbf{N}(n,d)$. Thus, $c_1 \geq t-1$, and moreover each $c_i \geq t-1$ since α is a necklace. Therefore α can be expressed as $01^{t-1+b_1}01^{t-1+b_2}\cdots 01^{t-1+b_{n-d}}$ for some string $\beta = b_1b_2\cdots b_{n-d}$ over the alphabet $\{0,1,\ldots,d\}$. By Lemma 2.7, β is a necklace. Suppose there is some largest $1 \leq i \leq n-d$ such that $b_i > 1$. Thus, each element of $b_{i+1}\cdots b_{n-d}$ must be in $\{0,1\}$. Since β is a necklace, each of its rotations $b_j\cdots b_{n-d}b_1\cdots b_{j-1} \geq \beta$. Thus, we can deduce that if $j > i$ then $b_j\cdots b_{n-d}b_1\cdots b_{j-1} > b_1b_2\cdots b_{i-1}$. This implies that $b_j\cdots b_{n-d}b_1\cdots b_{i-1} > b_1b_2\cdots b_{i-1}$. Now consider $\gamma = b_1b_2\cdots b_{i-2}(b_{i-1}{+}1)b_{i+1}\cdots b_{n-d}$. Since $b_1b_2\cdots b_{i-1}$ is a prenecklace, $b_1b_2\cdots b_{i-2}(b_{i-1}{+}1)$ is a Lyndon word by Corollary 2.2. Thus any proper rotation of γ starting before b_{i+1} will be strictly greater than γ. Now consider a rotation of γ starting from b_j for $i{+}1 \leq j \leq n{-}d$. Observe that a rotation starting from b_j has prefix $b_j\cdots b_{n-d}b_1\cdots b_{i-2}(b_{i-1}+1)$. We have already noted that $b_j\cdots b_{n-d}b_1\cdots b_{i-1} > b_1b_2\cdots b_{i-1}$, and therefore the complete rotation of γ starting with b_j must be greater than γ. Since every proper rotation of γ is strictly greater than γ, γ is a necklace. However, this means that $\gamma(b_i-1)$ is also a necklace by Corollary 2.3 and hence by Lemma 2.7, $01^{b_1}01^{b_2}\cdots 01^{b_{i-2}}01^{b_{i-1}+1}01^{b_{i+1}}\cdots 01^{b_{n-d}}01^{b_i-1}$ is also a necklace. But this contradicts $\alpha = \textsc{LargestNeck}(n,d)$. Therefore there is no $b_i > 1$ and hence β is a binary string. Now, since the necklace β is binary it must have density x. For any $b'_1b'_2\cdots b'_{n-d} \in \mathbf{N}(n{-}d,x)$ we have $\alpha' = 01^{t-1+b'_1}01^{t-1+b'_2}\cdots 01^{t-1+b'_{n-d}} \in \mathbf{N}(n,d)$ by Lemma 2.7. Clearly, α' will be largest when $b'_1b'_2\cdots b'_{n-d} = \textsc{LargestNeck}(n-d,x)$. Thus, $\beta = \textsc{LargestNeck}(n-d,x)$. □

By combining the previous two lemmas, the following equation can be used to recursively compute $\textsc{LargestNeck}(n,d)$ letting $t = \lfloor \frac{n}{d} \rfloor$ and $s = \lfloor \frac{n}{n-d} \rfloor$:

$$\textsc{LargestNeck}(n,d) \;=\; \begin{cases} 0^n & \text{if } \; d=0 \\ 0^{t-b_1}10^{t-b_2}1\cdots 0^{t-b_d}1 & \text{if } \; 0<d\leq \frac{n}{2} \\ 01^{s-1+c_1}01^{s-1+c_2}\cdots 01^{s-1+c_{n-d}} & \text{if } \; \frac{n}{2}<d<n \\ 1^n & \text{if } \; d=n, \end{cases}$$

where $b_1b_2\cdots b_d = \textsc{LargestNeck}(d, d-(n \bmod d))$ and $c_1c_2\cdots c_{n-d} = \textsc{LargestNeck}(n-d, n \bmod (n-d))$. Note that the strings returned in each recursive application have length less than or equal to $\frac{n}{2}$. Given these strings, obtaining the largest necklace can easily be constructed in $O(n)$ time. Thus we arrive at the following theorem.

Theorem 3.3 $\textsc{LargestNeck}(n,d)$ *can be computed in* $O(n)$ *time for* $0 \leq d \leq n$.

We conclude this section with two interesting properties of $\textsc{LargestNeck}(n,d)$.

Lemma 3.4 *Let* $0 \leq d \leq n$. *If* $\textsc{LargestNeck}(n,d) = a_1a_2\cdots a_n$ *then* $\textsc{LargestNeck}(n,n-d) = \overline{a}_n\cdots\overline{a}_2\overline{a}_1$.

Proof. The proof is by induction on n. For any $n \geq 1$, $\textsc{LargestNeck}(n,0) = 0^n$ and $\textsc{LargestNeck}(n,n) = 1^n$. Thus the base case when $n=1$ clearly holds, along with the cases when $d=0$ and $d=n$. Consider $0<d<n$ and let $\textsc{LargestNeck}(n,d) = a_1a_2\cdots a_n$. By Lemma 3.1, $\alpha = 0^{t-b_1}10^{t-b_2}1\cdots 0^{t-b_d}1$ where $b_1b_2\cdots b_d = \textsc{LargestNeck}(d, d-(n \bmod d))$ and $t = \lfloor \frac{n}{d} \rfloor$. By induction, $\textsc{LargestNeck}(d, n \bmod d) = \overline{b}_d\cdots\overline{b}_2\overline{b}_1$. Now by applying Lemma 3.2,

$$\begin{aligned} \textsc{LargestNeck}(n,n-d) &= 01^{t-1+\overline{b}_d}\cdots 01^{t-1+\overline{b}_2}\cdots 01^{t-1+\overline{b}_1} \\ &= 01^{t-b_d}\cdots 01^{t-b_2}\cdots 01^{t-b_1} \\ &= \overline{a}_n\cdots\overline{a}_2\overline{a}_1. \end{aligned}$$

□

Lemma 3.5 *Let $0 \le d \le n$.* $\textsc{LargestNeck}(n,d) = \delta^j$ *where $j = gcd(n,d)$ and δ is some binary string of length $\frac{n}{j}$.*

Proof. The proof is by induction on n. Since $gcd(n,0) = gcd(n,n) = n$, $\textsc{LargestNeck}(n,0) = 0^n$ and $\textsc{LargestNeck}(n,n) = 1^n$, the result clearly holds for all $d = 0$ and $d = n$, and for $n = 1$. Suppose $0 < d < n$ and consider $\alpha = \textsc{LargestNeck}(n,d)$ for $n \ge 2$. By Lemma 3.1, $\alpha = 0^{t-b_1}10^{t-b_2}1 \cdots 0^{t-b_d}1$ where $t = \lfloor \frac{n}{d} \rfloor$ and $\beta = b_1b_2\cdots b_d = \textsc{LargestNeck}(d, d-n \bmod d)$. By induction, $\beta = \gamma^j$ where $j = gcd(d, d-n \bmod d)$. Thus, $\alpha = \delta^j$ for some δ of length $\frac{n}{j}$. Finally, by applying Euclid's algorithm we have $gcd(n,d) = gcd(d, n \bmod d) = gcd(d, d-n \bmod d) = j$. □

This final lemma implies that $\textsc{LargestNeck}(n,d)$ is a Lyndon word if and only if $gcd(n,d) = 1$.

4 Testing if a string is a prefix of some necklace in $\mathbf{N}(n,d)$

Let the boolean function $\textsc{IsPrefix}(\alpha,n,d)$ return True if and only if $\alpha = a_1a_2\cdots a_t$ is a prefix of some necklace in $\mathbf{N}(n,d)$. In this section we present an $O(n^2)$ implementation for this function using results from the previous section. There are two trivial conditions for the function to return true: the density constraint must be attainable and α must be a prenecklace. Let $den(\alpha)$ denote the density of α. Then for the density to be attainable we must have $0 \le d - den(\alpha) \le n - t$.

Let $\alpha = a_1a_2\cdots a_t$ be a prenecklace where $1 \le t \le n$. Let $\textsc{Extend}(\alpha,n) = a_1a_2\cdots a_n$ be the lexicographically smallest prenecklace of length n with prefix α.

Lemma 4.1 *Let $0 \le d \le n$ and let $1 \le t \le n$. Suppose $\alpha = a_1a_2\cdots a_t$ is a prenecklace and $a_1a_2\cdots a_n =$ $\textsc{Extend}(\alpha,n)$. Then α is a prefix of some necklace in $\mathbf{N}(n,d)$ if and only if $a_1a_2\cdots a_n \in \mathbf{N}(n,d)$ or there exists $t < j \le n$ such that $a_j = 0$ and $d' = d - den(a_1a_2\cdots a_{j-1}1) \ge 0$ and either:*

(1) $j = n$ and $d' = 0$ or

(2) $j < n$ and there exists $\beta \in \mathbf{N}(n-j, d')$ such that $a_1a_2\cdots a_{j-1}1 \le \beta$.

Proof. ($\Rightarrow$) Suppose α is a prefix of $b_1b_2\cdots b_n \in \mathbf{N}(n,d)$ and suppose $a_1a_2\cdots a_n$ is not in $\mathbf{N}(n,d)$. By applying Theorem 2.1, there must be some smallest $j > t$ such that $a_1a_2\cdots a_{j-1} = b_1b_2\cdots b_{j-1}$ with $a_j = 0$ and $b_j = 1$ which implies $a_1a_2\cdots a_{j-1}1$ is a Lyndon word. Clearly $b_{j+1}b_{j+2}\cdots b_n$ has length $n-j$ and density $d' = d - den(a_1a_2\cdots a_{j-1}1)$. If $j = n$ then $d' = 0$. Otherwise, $j < n$ and since $b_1b_2\cdots b_n$ is a necklace and $b_1b_2\cdots b_j$ is a Lyndon word, it must be that $b_1b_2\cdots b_j \le b_{j+1}b_{j+2}\cdots b_n$. Thus, if $b_{j+1}b_{j+2}\cdots b_n$ ($= \beta$) is a necklace we are done. Otherwise let $b_{j+1}b_{j+2}\cdots b_n = \delta\gamma$ such that its rotation $\gamma\delta$ ($= \beta$) is a necklace. Again, since $b_1b_2\cdots b_n$ is a necklace and $b_1b_2\cdots b_j$ is a Lyndon word, $b_1b_2\cdots b_j \le \gamma$. It follows that $a_1a_2\cdots a_{j-1}1 = b_1b_2\cdots b_j \le \gamma\delta$.

($\Leftarrow$) If $a_1a_2\cdots a_n \in \mathbf{N}(n,d)$ then clearly α is a prefix of some necklace in $\mathbf{N}(n,d)$. Otherwise, suppose there exists $t < j \le n$ such that a_j=0 and $d' = d - den(a_1a_2\cdots a_{j-1}1) \ge 0$ and either (1) $j = n$ and $d' = 0$ or (2) $j < n$ and there exists $\beta \in \mathbf{N}(n-j, d')$ such that $a_1a_2\cdots a_{j-1}1 \le \beta$. For either case $a_1a_2\cdots a_{j-1}1$ is a Lyndon word since $a_1a_2\cdots a_j$ is a prenecklace. Thus, if $j = n$ and $d' = 0$, then $a_1a_2\cdots a_{n-1}1$ is a necklace with density d. Otherwise, $a_1a_2\cdots a_{j-1}1\beta$ is a binary string of length n and density d. By Corollary 2.5 $a_1a_2\cdots a_{j-1}1\beta$ is a necklace. Thus α is a prefix of some necklace in $\mathbf{N}(n,d)$. □

Assuming the density constraints are attainable, and $\alpha = a_1a_2\cdots a_t$ is a prenecklace, we can directly apply Lemma 4.1 to determine $\textsc{IsPrefix}(\alpha,n,d)$. To apply this lemma, note that it suffices only to compare $\beta = \textsc{LargestNeck}(n-j, d - den(a_1a_2\cdots a_{j-1}1))$ to $a_1a_2\cdots a_j1$, for a given j. Repeated applications

Algorithm 1 Testing if $\alpha = a_1a_2\cdots a_t$ is a prefix of a necklace in $\mathbf{N}(n,d)$.

function IsPrefix(α, n, d) **returns boolean**
  if ($d < den(\alpha)$ **or** $d - den(\alpha) > n - t$) **then** **return** False
  if α is not a prenecklace **then** **return** False
  $a_1a_2\cdots a_n \leftarrow$ Extend(α)
  if $a_1a_2\cdots a_n \in \mathbf{N}(n,d)$ **then** **return** True
  for $j \leftarrow t+1$ **to** n **do**
    $d' = d - den(a_1a_2\cdots a_{j-1}1)$
    if $a_j = 0$ **and** $d' \geq 0$ **then**
      if $j = n$ **and** $d' = 0$ **then** **return** True
      if $j < n$ **and** $a_1a_2\cdots a_{j-1}1 \leq$ LargestNeck($n-j, d'$) **then** **return** True
  return False

of Theorem 2.1 can be used to test if α is a prenecklace and to compute Extend(α, n) in $O(n)$ time. Pseudocode for IsPrefix(α, n, d) is given in Algorithm 1.

Since LargestNeck(n, d) can be computed in $O(n)$ time, we obtain the following theorem.

Theorem 4.2 IsPrefix*(α, n, d) can be computed in $O(n^2)$ time for* $0 \leq d \leq n$.

5 The largest necklace that is less than or equal to a given string

Let Ln(α, n, d) be a function that returns the largest necklace in $\mathbf{N}(n,d)$ that is less than or equal to a given binary string $\alpha = a_1a_2\cdots a_n$, or ϵ (the empty string) if no such necklace exists. In this section we present an $O(n^3)$ implementation of this function by applying the results from the previous section.

Let $\beta =$ Ln(α, n, d). If $\alpha \in \mathbf{N}(n,d)$ then clearly $\beta = \alpha$. Otherwise, let $t > 0$ be the largest index such that $a_t = 1$ and $a_1a_2\cdots a_{t-1}0$ is a prefix of some necklace in $\mathbf{N}(n,d)$. If no such index t exists, then there is no necklace in $\mathbf{N}(n,d)$ that is less than α and thus $\beta = \epsilon$. If t exists, then since it was chosen to be the largest index satisfying the conditions, $a_1a_2\cdots a_{t-1}0$ will be the first t characters of $\beta = b_1b_2\cdots b_n$. The next character b_{t+1} will be the largest element so $b_1b_2\cdots b_{t+1}$ is a prefix of some necklace in $\mathbf{N}(n,d)$. This can be determined by calling IsPrefix($b_1b_2\cdots b_t1, n, d$); if it returns true, then $b_{t+1} = 1$ and otherwise $b_{t+1} = 0$. The remaining characters $b_{t+2}, b_{t+3}, \ldots, b_n$ can be computed in the same way. Pseudocode for Ln($a_1a_2\cdots a_n, n, d$) is given in Algorithm 2.

Algorithm 2 Computing the largest necklace less than or equal to a given string.

function Ln(α, n, d) **returns necklace**
  if $\alpha \in \mathbf{N}(n,d)$ **then** **return** α
  $t \leftarrow n$
  while $t > 0$ **and not** ($a_t = 1$ **and** IsPrefix($a_1a_2\cdots a_{t-1}0, n, d$)) **do** $t \leftarrow t-1$
  if $t = 0$ **then** **return** ϵ
  $b_1b_2\cdots b_t \leftarrow a_1a_2\cdots a_{t-1}0$
  for $j \leftarrow t+1$ **to** n **do**
    if IsPrefix($b_1b_2\cdots b_{j-1}1, n, d$) **then** $b_j \leftarrow 1$
    else $b_j \leftarrow 0$
  return $b_1b_2\cdots b_n$

Since IsPrefix(α, n, d) can be computed in $O(n^2)$ time, we obtain the following theorem.

Theorem 5.1 Ln*(α, n, d) can be computed in $O(n^3)$ time for* $0 \leq d \leq n$.

6 Lyndon words

In this final section, we extend the results for necklaces to Lyndon words.

Lemma 6.1 *Let α, β be two consecutive necklaces in the lexicographic ordering of $\mathbf{N}(n, d)$. Then at least one of α and β is a Lyndon word.*

Proof. Suppose $\alpha < \beta$. If α is a Lyndon word we are done. Otherwise, $\alpha = \gamma^i$ for some Lyndon word $\gamma = a_1 a_2 \cdots a_{\frac{n}{i}}$ with density $\frac{d}{i}$ where $i \geq 2$. Let $\delta = \text{LARGESTNECK}(\frac{n}{i}, \frac{d}{i})$. Suppose $\gamma = \delta$. By Lemma 3.5, $\text{LARGESTNECK}(n, d) = \sigma^j$ where $j = gcd(n, d)$ and σ has length $\frac{n}{j}$. Since i divides both n and d, i also divides j. By the definitions of δ and σ, $\delta = \sigma^{\frac{j}{i}}$, and thus $\alpha = \text{LARGESTNECK}(n, d)$. But this contradicts that $\alpha < \beta$. Thus, $\gamma \neq \delta$. Repeated application of Lemma 2.4 implies that $\gamma^{i-1}\delta$ is a Lyndon word that is greater than α. Thus the necklace β must have prefix γ^{i-1} and hence clearly is a Lyndon word. □

The following two-step algorithm will return the largest Lyndon word with length n and density $d > 0$ that is less than or equal to α, or ϵ if no such Lyndon word exists. Let $\beta = \text{LN}(\alpha, n, d)$. If β is a Lyndon word or ϵ, then return β. Otherwise $\beta = b_1 b_2 \cdots b_n$ is a necklace where $b_n = 1$ since $d > 0$. Thus, $b_1 b_2 \cdots b_{n-1} 0$ is the largest string less than β and hence $\gamma = \text{LN}(b_1 b_2 \cdots b_{n-1} 0, n, d)$ will give the second largest necklace that is less than or equal to α or ϵ if no such necklace exists. Thus the algorithm returns γ as either $\gamma = \epsilon$ (no such Lyndon word exists), or by Lemma 6.1, γ is a Lyndon word.

Since testing whether or not a string is a Lyndon word can easily be tested in $O(n)$ time by applying Theorem 2.1, the running time of this algorithm will be $O(n^3)$.

Lemma 6.2 *The largest Lyndon word of length n and density d that is less than or equal to $\alpha = a_1 a_2 \cdots a_n$ can be computed in $O(n^3)$ time for $0 < d < n$.*

Setting $\alpha = 1^n$, the previous lemma immediately implies the following result.

Corollary 6.3 *The largest Lyndon word of length n and density d can be computed in $O(n^3)$ time for $0 < d < n$.*

Finally, the following conjecture has been verified to be true for all $n < 600$ by applying the algorithm just described.

Conjecture 6.4 *Let $\alpha = a_1 a_2 \cdots a_n = \text{LARGESTNECK}(n, d)$ where $p = lyn(\alpha)$ for $0 < d < n$. If $p = n$, then the largest Lyndon word of length n and density d is α; otherwise it is $a_1 a_2 \cdots a_{p-1} 0 1 a_2 \cdots a_p (a_1 a_2 \cdots a_p)^{\frac{n}{p}-2}$.*

A proof of this conjecture implies that the largest Lyndon word of length n and density d can be computed in $O(n)$ time for $0 < d < n$.

7 Acknowledgement

The research of Joe Sawada is supported by the *Natural Sciences and Engineering Research Council of Canada* (NSERC) grant RGPIN 400673-2012.

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), 2000.

[2] P. Hartman and J. Sawada. Ranking fixed-density necklaces and Lyndon words. *manuscript*, 2016.

[3] S. Karim, Z. Alamgir, and S. M. Husnine. Generating fixed density bracelets of arbitrary base. *International Journal of Computer Mathematics*, 91(3):434–446, 2014.

[4] T. Kociumaka, J. Radoszewski, and W. Rytter. Efficient ranking of Lyndon words and decoding lexicographically minimal de Bruijn sequence. *SIAM Journal on Discrete Mathematics*, 30(4):2027–2046, 2016.

[5] S. Kopparty, M. Kumar, and M. Saks. Efficient indexing of necklaces and irreducible polynomials over finite fields. *Theory of Computing*, 12(7):1–27, 2016.

[6] C. Reutenauer. *Free Lie algebras*. London Mathematical Society monographs. Clarendon Press New York, Oxford, 1993.

[7] F. Ruskey, J. Sawada, and A. Williams. De Bruijn sequences for fixed-weight binary strings. *SIAM Journal on Discrete Mathematics*, 26(2):605–617, 2012.

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

[9] J. Sawada and A. Williams. A Gray code for fixed-density necklaces and Lyndon words in constant amortized time. *Theoretical Computer Science*, 502:46 – 54, 2013. Generation of Combinatorial Structures.

[10] J. Sawada and A. Williams. Practical algorithms to rank necklaces, Lyndon words, and de Bruijn sequences. *Journal of Discrete Algorithms (in press, available online)*, 2017.

[11] T. M. Wang and C. D. Savage. A Gray code for necklaces of fixed density. *SIAM J. Discrete Math*, 9:654–673, 1997.