New Results on Nyldon Words Derived Using an Algorithm from Hall Set Theory

Swapnil Garg

Massachusetts Institute of Technology, Cambridge, MA 02139
swapnilg@mit.edu

August 2019

Abstract

Grinberg defined Nyldon words as those words which cannot be factorized into a sequence of lexicographically nondecreasing smaller Nyldon words. He was inspired by Lyndon words, defined the same way except with “nondecreasing” replaced by “nonincreasing.” Charlier, Philibert, and Stipulanti proved that, like Lyndon words, any word has a unique nondecreasing factorization into Nyldon words. They also show that the Nyldon words form a right Lazard set, and equivalently, a right Hall set. In this paper, we provide a new proof of unique factorization into Nyldon words related to Hall set theory and resolve several questions of Charlier et al. In particular, we prove that Nyldon words of a fixed length form a circular code, we prove a result on factorizing powers of words into Nyldon words, and we investigate the Lazard procedure for generating Nyldon words.

1 Introduction

Nyldon words were introduced in 2014 by Darij Grinberg [7], with the name a play on the related Lyndon words, first studied in the 1950s by Shirshov [16] and Lyndon [12]. While Lyndon words were first defined as those words which are the smallest among their cyclic rotations, the Chen-Fox-Lyndon Theorem states that any word can be written uniquely as a sequence of lexicographically nonincreasing Lyndon words. That is, we can write $w = \ell_1 \ell_2 \cdots \ell_k$, where $\ell_1 \geq_{\text{lex}} \ell_2 \geq_{\text{lex}} \cdots \geq_{\text{lex}} \ell_k$ [4, 17]. In a sense, Lyndon words can act as “primes” in the factorization of all words. Thus, Lyndon words can also be defined recursively as those words which are either single letters, or which cannot be factorized into a sequence of nonincreasing smaller
Lyndon words. By changing the word “nonincreasing” to “nondecreasing” in this definition, we arrive at Nyldon words, which behave in a surprisingly different way. For example, it is much more difficult to determine whether a word is Nyldon from looking at its cyclic rotations.

In [3], Charlier, Philibert, and Stipulanti prove an analog of the Chen-Fox-Lyndon Theorem, showing that all words have a unique nondecreasing factorization into Nyldon words. They also give an algorithm for computing the Nyldon factorization of a word, investigate the differences between Nyldon and Lyndon words, and show that Nyldon words form a Hall set; see the end of Section 3 for more information on Hall sets. As unique factorization holds for Nyldon words, they seem to behave more nicely than another variant of Lyndon words studied recently, the inverse Lyndon words [2].

Lyndon words form a Hall set, with the ordering given by the standard lexicographical ordering. If we instead use the reverse lexicographical ordering to construct a Hall set, we arrive at Nyldon words, showing that these words arise in a quite natural way. As Lyndon introduced his namesake words with the intention of giving bases of free Lie algebras, Nyldon words could also shed light into this area, since they form a Hall set as well.

In this paper, we resolve several questions posed by Charlier, Philibert, and Stipulanti in [3]. In Section 3, we present the algorithm of Mélano, originating in Hall set theory, and demonstrate its power in elucidating certain properties of Nyldon words in Section 4. In Section 5, we show that the factorization algorithm conceived in [3] is faster than previously thought. In Section 6, we compute how fast the right Lazard procedure takes to generate all the Nyldon words over a given alphabet up to a given length. Finally, in Section 7, we prove a property of Lyndon words using only the recursive definition, answering a question of Charlier et al. in [3].

2 Background

Throughout this paper, let $A$ be an alphabet endowed with a total order $<$, with a size at least 2. We will only work with finite alphabets, denoted $\{0, 1, \ldots, m\}$ with $0 < 1 < \cdots < m$. Let $<_{\text{lex}}$ denote the lexicographical order on words. Let $A^+$ be the set of all finite words (including the empty word $\varepsilon$) over $A$, and let $A^+ = A^+ \setminus \varepsilon$. Let the length of a finite word $w$ be denoted by $|w|$. A word is primitive if it is not a power of another word, and a word $w$ is a conjugate (or cyclic rotation) of $x$ if $w = uv$, $x = vu$ for words $u, v$.

**Definition.** [7] [11] A nonempty word $w$ is Nyldon (resp. Lyndon) if $w = a \in A$ or $w$ cannot be factorized as $(w_1, w_2, \ldots, w_k)$ where $w_1, w_2, \ldots, w_k$ are Nyldon (resp. Lyndon), $k \geq 2$, and $w_1 \leq_{\text{lex}} w_2 \leq_{\text{lex}} \cdots \leq_{\text{lex}} w_k$ (resp. $w_1 \geq_{\text{lex}} w_2 \geq_{\text{lex}} \cdots \geq_{\text{lex}} w_k$). Such a factorization is referred to as a Nyldon (resp. Lyndon) factorization (of $w$).

Nyldon words were first extensively studied by Charlier et al., who proved that Nyldon
factorization was unique using the following two lemmas:

**Lemma 1.** [3] For a Nyldon word $x$ with a proper Nyldon suffix $s$, we have $s <_{lex} x$.

**Lemma 2.** [3] In the Nyldon factorization of the word $x$ as $(n_1, n_2, \ldots, n_k)$, the Nyldon word $n_k$ is the longest Nyldon suffix of $x$.

Clearly unique factorization follows from the second lemma. In the following section, we give an alternate, elementary proof of the above two lemmas using only the stated definition of Nyldon words. We also provide an elementary proof of the following theorem about Nyldon words:

**Theorem 3.** [3] Every primitive word has exactly one Nyldon word in its conjugacy class, and no periodic word is Nyldon.

We provide a table of short binary Nyldon words for reference here.

| Nyldon Words |
|--------------|
| 0            | 10011        | 101111       | 1001111       |
| 1            | 10110        | 1000000      | 1011000       |
| 10           | 10111        | 1000001      | 1011001       |
| 100          | 100000       | 1000010      | 1011010       |
| 101          | 100001       | 1000011      | 1011100       |
| 1000         | 1000010      | 1000100      | 1011101       |
| 1001         | 100011       | 1000110      | 1011110       |
| 1011         | 100110       | 1000111      | 1011111       |
| 10000        | 100111       | 1001010      |                |
| 10001        | 101100       | 1001100      |                |
| 10010        | 101110       | 1001110      |                |

Table 1: List of Binary Nyldon Words of Length at Most 7

### 3 Mélançon’s Algorithm

As explained by [3], the Nyldon words form a right Hall set, and so an algorithm attributed to Mélançon can be used to find the Nyldon conjugate of a primitive word. We develop the theory behind this algorithm and demonstrate how it can also be used to factorize words into Nyldon words. We also give a new way to prove unique factorization, among other properties of Nyldon
Lemma 4. Suppose \( u_1, u_2, \ldots, u_m \) are Nyldon words (also referred to as blocks) such that for any sequence of distinct numbers \( a_1, a_2, \ldots, a_i \) all between 1 and \( n \) inclusive, any Nyldon factorization of \( u_{a_1}u_{a_2}\cdots u_{a_i} \) does not break up the blocks (so we can write each word in the factorization as a consecutive sequence of whole blocks).

If \( u_k \) is the smallest of these blocks lexicographically, \( u_ku_{a_1}u_{a_2}\cdots u_{a_i} \) is not Nyldon for any sequence \( a_1, a_2, \ldots, a_i \).

Proof. Consider a Nyldon factorization of \( u_{a_1}u_{a_2}\cdots u_{a_i} \). This preserves the blocks, so it starts with some Nyldon word \( p \) that starts with \( u_{a_1} \). But \( p \geq_{\text{lex}} u_{a_1} \geq_{\text{lex}} u_k \), so a Nyldon factorization of \( u_ku_{a_1}u_{a_2}\cdots u_{a_i} \) is then just \( u_k \) prepended to the given factorization of \( u_{a_1}u_{a_2}\cdots u_{a_i} \), so the word is not Nyldon since it has multiple factors.

Lemma 5. Suppose \( u_1, \ldots, u_m \) and \( u_k \) are as above, with the additional condition that \( u_{k-1} >_{\text{lex}} u_k \). Then, if we combine \( u_{k-1} \) and \( u_k \) into a block, we still have the condition holding that any Nyldon factorization of a word made up of a subset of the blocks in any order preserves the blocks.

Proof. First, \( u_{k-1}u_k \) cannot have a Nyldon factorization \( (u_{k-1}, u_k) \), so \( u_{k-1}u_k \) must be Nyldon.

We want to show that any Nyldon factorization of \( w = u_{a_1}u_{a_2}\cdots u_{a_i}u_{k-1}u_ku_{a_{i+1}}u_{a_{i+2}}\cdots u_{a_j} \) does not break up \( u_{k-1}u_k \). For sake of contradiction, suppose one does. Then, there are two Nyldon factors, one ending with \( u_{k-1} \), and one beginning with \( u_k \). Let \( p = u_{a_1}u_{a_2}\cdots u_{a_i}u_{k-1} \) and \( q = u_ku_{a_{i+1}}u_{a_{i+2}}\cdots u_{a_j} \). We have that this Nyldon factorization of \( w \) gives a valid factorization for \( p \) next to a valid factorization for \( q \) such that the first factor of \( q \) is at least the last factor of \( p \) lexicographically.

By Lemma 4, \( u_ku_{a_1}u_{a_2}\cdots u_{a_i} \) cannot be Nyldon for any \( c \), so \( u_k \) is the first factor in any Nyldon factorization of \( q \).

But then, if the last Nyldon factor of \( p \) is \( u_{k-1} \), we have \( u_{k-1} >_{\text{lex}} u_k \), and if it is \( u_{a_m}u_{a_{m+1}}\cdots u_{a_i}u_{k-1} \), then since \( u_{a_m} \geq_{\text{lex}} u_k \), \( u_{a_m}u_{a_{m+1}}\cdots u_{a_i}u_{k-1} >_{\text{lex}} u_k \). So, we have a contradiction and \( u_{k-1}u_k \) cannot be broken up.
Let the operation of combining blocks be called \textit{contraction}, where we always contract a block into the block to its left. If we start with a word as a sequence of letters, we can repeat this algorithm to end up with a Nyldon word. It turns out that this algorithm is a special case of an algorithm attributed to Mélançon; see \cite{14} for more information, and also the last section of \cite{15} for an example of this algorithm. Specifically, suppose we start with a primitive word, with each digit being a block. We consider the word as circular (with the blocks in a circle) rather than linear. At each step, we repeat the contraction operation, eventually terminating in one block, which is a Nyldon word. As we will see shortly, this word is the unique Nyldon conjugate of the original word.

For convenience of the reader, we reproduce the pseudocode for this algorithm given on the last page of \cite{3}. $T(i)$ designates the $i$th element of the list $T$ while $T(-i)$ denotes the $(n-i+1)$th element of $T$ if $n$ is the length of $T$.

\begin{algorithm}
\textbf{Require:} $w \in A^+$ primitive
\textbf{Ensure:} $\text{NyIC}$ is the Nyldon conjugate of $w$

\begin{itemize}
\item $\text{NyIC} \leftarrow$ list of letters of $w$, $T \leftarrow$ list of letters of $w$
\item while $\text{length}(\text{NyIC}) > 1$ do
  \begin{itemize}
  \item if $T(1) = \min_{\leq_{\text{lex}}} \text{NyIC}$ and $T(1) <_{\text{lex}} T(-1)$ then
    \begin{itemize}
    \item $T \leftarrow (T(2), \ldots, T(-2), T(-1) \cdot T(1))$
    \end{itemize}
  \end{itemize}
  \item $i \leftarrow 2$, $j \leftarrow 2$
  \item while $j \leq \text{length}(T)$ do
    \begin{itemize}
    \item while $i \leq \text{length}(T)$ and $T(i) \neq \min_{\leq_{\text{lex}}} \text{NyIC}$ do
      \begin{itemize}
      \item $i \leftarrow i + 1$
      \end{itemize}
    \end{itemize}
    \item if $i \leq \text{length}(T)$ and $T(i) <_{\text{lex}} T(i - 1)$ then
      \begin{itemize}
      \item $T \leftarrow (T(1), \ldots, T(i - 1) \cdot T(i), \ldots, T(-1))$
      \end{itemize}
    \end{itemize}
  \item $j \leftarrow i + 1$, $i \leftarrow i + 1$
  \end{itemize}
\item $\text{NyIC} \leftarrow T$
\end{itemize}
return $\text{NyIC}$
\end{algorithm}

Algorithm 1: Mélançon’s algorithm

We now provide an example of Mélançon’s algorithm.

\begin{example}
Say we want to find the Nyldon conjugate of 10001011010101.
\end{example}
1. We start with $1,0,0,0,1,0,1,0,0,0,1,0,1,0,0,1$. 

2. We contract the $0$s, giving $1000,10,1,10,10,10,1$. 

3. We contract the $1$s, giving $1000,101,10,10,101$. 

4. We contract the $10$s, giving $1000,101010,101$. 

5. We contract the $1000$. It is the first block, so it is added to the end of the last block, giving $1011010,1011000$. 

6. We contract the $101$, giving a Nyldon conjugate $10110101011000$. 

Using only the recursive definition of Nyldon words and elementary methods, we will reprove the following: 

**Theorem 7.** 

1. Every primitive word has exactly one Nyldon conjugate. 

2. No periodic word is Nyldon. 

**Proof.** Suppose we start with a primitive word as a circular sequence where each letter is its own block, and we perform Mélacçon’s algorithm on the sequence of blocks. When the algorithm terminates, we are left with one block, which by Lemma 5 is Nyldon. Furthermore, suppose we have another Nyldon conjugate. Initially, this conjugate preserves the blocks. However, at some point the first block in this conjugate, $u_k$, will contract into the last block, as we are left with only one block at the end. Then $u_k$ cannot start this Nyldon conjugate by Lemma 4. Therefore, every primitive word has exactly one Nyldon conjugate. 

If we start with a periodic word with minimal unit $s$, suppose that the Nyldon conjugate of $s$ is $n$. If we perform Mélacçon’s algorithm on our word, we will end up with some number of blocks that are just $n$. Therefore, our periodic word is not Nyldon. 

As we end up with the same result each time, it doesn’t matter in what order we do contractions of blocks of the same type as long as we don’t contract a block into an equal block to its left. 

With a minor modification, we can use Mélacçon’s algorithm to find the Nyldon factorization of a word too. We start with the blocks in a line instead of a circle, and if at any point the first block is the lexicographically smallest, we remove it, adding it to the Nyldon factorization, and perform Mélacçon’s algorithm on the remaining blocks. We can use this modification to also reprove the fact from [3] that the Nyldon factorization is unique. 

**Theorem 8.** The above modification of Mélacçon’s algorithm gives the Nyldon factorization of a word. Furthermore, this factorization is unique.
Proof. At each step, if we have $u_1u_2\cdots u_k$ as our word with the $u_i$s being the blocks, we check whether $u_1$ is lexicographically smallest among the remaining blocks. If so, then by Lemma 4, $u_1u_2\cdots u_i$ cannot be Nyldon for any $i \neq 1$. By Lemma 5, the Nyldon factorization of $u_1u_2\cdots u_k$ preserves the blocks, so it must start with $u_1$. So, we take out $u_1$, as the first (and only possible first) Nyldon factor of our word, and perform the algorithmic procedure on $u_2u_3\cdots u_k$ (and check now whether $u_2$ is smallest). We will never have to combine a word $u_1$ with a word $u_k$ since if $u_1$ is the lexicographically smallest word, it will be taken out first. So, all contractions occur keeping the current word linear, and we end up with the unique Nyldon factorization of our word.

We give an example of Mélançon’s algorithm used for factorization, using the same string as the previous example.

**Example 9.** Say we want to find the Nyldon factorization of 10001011010101.

1. We start with 1,0,0,0,1,0,1,1,0,1,0,1.
2. We contract the 0s, giving 1000,10,1,10,10,1.
3. We contract the 1s, giving 1000,101,10,10,101.
4. We contract the 10s, giving 1000,1011010,101.
5. We contract the 1000. It is the first block, so it becomes the first Nyldon factor, and we are left with (1011010,101).
6. We contract 101, giving the block 1011010101. We thus get a Nyldon factorization of (1000,1011010101).

We can also prove the two lemmas from [3], using the following one.

**Lemma 10.** Let $s$ be a Nyldon suffix of $w$. Then, when applying the factorization version of Mélançon’s algorithm to $w$, the last block will be $s$ at some point.

**Proof.** The crucial idea is that information only travels from right to left, as a block always contracts onto the block to its left. Therefore, if we only look at the last $|s|$ letters, the algorithm acts identically as it would on $s$, until the leftmost block of $s$ must contract. Specifically, the factorization version of Mélançon’s algorithm on the last $|s|$ letters of $w$ acts the same as the conjugate version of Mélançon’s algorithm on $s$. If at some point the leftmost block in $s$ has to contract when applying the factorization version of Mélançon’s algorithm to $w$, then the same must be true when finding the Nyldon conjugate of $s$. However, as $s$ is Nyldon, the Nyldon conjugate of $s$ is just $s$, and the leftmost block in $s$ never contracts; instead, all of $s$ becomes one block. Therefore, in $w$, all of $s$ will become one block as well. □
Corollary 11. In the Nyldon factorization of the word $x$ as $(u_1, u_2, \ldots, u_k)$, the Nyldon word $u_k$ is the longest Nyldon suffix of $x$. Also, for a Nyldon word $x$ with a Nyldon suffix $s$, we have $s <_{\text{lex}} x$.

Proof. If $s$ is a Nyldon suffix of a Nyldon word $w$, $s$ will form when applying Mélançon’s algorithm to $w$ by Lemma 10. Then, since the final product is $w$, and a block formed by contraction is by definition lexicographically greater than the two previous blocks, we have $s <_{\text{lex}} w$. Furthermore, if we perform the factorization version of Mélançon’s algorithm on a word $w$, every Nyldon suffix of $w$ forms as a block at some point by Lemma 10, so $w$’s longest Nyldon suffix will form. As the last factor in the Nyldon factorization of $w$ must be a Nyldon suffix of $w$, and all such suffixes form as a block, the last factor will be the longest Nyldon suffix of $w$. \[\square\]

The algorithm we present is a case of a more general algorithm that works for all Hall sets. The Nyldon words form a right Hall set, as explained in Sections 11 and 12 of [3]. We will not define a Hall set here; for more information, see [14].

4 Applications of Mélançon’s Algorithm

In [3], Charlier et al. ask (Open Problem 38) whether for a primitive word $w$ with Nyldon conjugate $n$, and large enough $k$ depending on $w$, the word $w^k$ can be factorized as $(p_1, p_2, \ldots, p_i, n, n, \ldots, n, s_1, s_2, \ldots, s_j)$ for some $K$. We give a positive answer to this problem and provide a bound on $K$.

Furthermore, they ask whether the set of Nyldon words of a fixed length form a circular code (Open Problem 46). We explain the definition of a circular code below, and give a positive answer to the question.

We first demonstrate that there exists such a $K$ for powers of words $w$.

**Lemma 12.** [3] Starting with a Nyldon word $x$, suppose we add letters to the left one at a time, and take note of each time the whole word is Nyldon. Consider the first time this happens, and suppose that the last letter to be added is $d$, and right before that (one letter previously), the Nyldon factorization of our word is $u_1 u_2 \cdots u_k$, so that $du_1 u_2 \cdots u_k$ is Nyldon. Then, $du_1 u_2 \cdots u_i$ is Nyldon and $du_1 u_2 \cdots u_i >_{\text{lex}} u_{i+1}$ for all $1 \leq i \leq k$. Furthermore, $u_k = x$.

Proof. See the justification of Algorithm 1 in [3], in the proof of Proposition 18. \[\square\]

**Lemma 13.** Suppose $ab$ is Nyldon for words $a, b$ with a nonempty. Then, $a^k ab$ is not Nyldon for $k \geq 1$. 


Proof. Without loss of generality, let $a$ be primitive.

For sake of contradiction, suppose $a^k ab$ is Nyldon. Then, suppose we start with $ab = x_0$ and add letters of $a^k$ to the left of $ab$ one at a time. For each $i$, let $y_i$ be the word the $i$th time the whole word is Nyldon, so $y_0 = x_0$. Also, let $x_i$ be such that $y_i = x_ix_{i-1} \cdots x_0$. Therefore, after $i$ steps, we have a word $x_ix_{i-1} \cdots x_0$ such that $x_ix_{i-1} \cdots x_0$ is Nyldon for all $j \leq i$. By Lemma 12, $x_i$ is always Nyldon and $x_i \succ_{lex} x_{i+1}$. At the end of this process, we have $a^k ab = x_kx_{k-1} \cdots x_0$ for some $k$, with $x_0 = ab$. Then $x_k \succ_{lex} x_0$, but $x_k$ starts with $a$, so we have that $|x_k| > |a|$. For all $x_i$ with $0 < i < k$, we have that $x_i$ is between $ab$ and $x_k$ lexicographically, in particular $x_1$. The only way this is possible is if $x_1$ starts with the same $|a|$ letters as $a$, and since $a$ is primitive, $x_1 = a^j$ for some $1 \leq j \leq k$. But then if $j \geq 2$ we have that $x_1$ is not Nyldon, and if $x_1 = a$ then $x_1 \preceq_{lex} x_0$. So, we have a contradiction and we are done. 

**Theorem 14.** For a primitive word $w$ with Nyldon conjugate $n$, and large enough $k$ depending on $w$, the word $w^k$ can be factorized as $(p_1p_2, \ldots, p_1n, n, \ldots, n, s_1, s_2, \ldots, s_j)$ for some $K$.

Proof. Suppose $w = w_1w_2 \cdots w_t$ are the letters in $w$. By Lemma 13, in a word of the form $w^k$, there is a longest possible Nyldon factor that starts at $w_i$ and ends at $w_j$ for a fixed $i$ and $j$, since, if say $w_iw_{i+1} \cdots w_iw^a w_{i+1}w_{i+2} \cdots w_{j}$ is Nyldon, then $w_iw_{i+1} \cdots w_iw^b w_{i+1}w_{i+2} \cdots w_{j}$ cannot be Nyldon for $a \neq b$ and $a, b$ positive. So, there is a finite number of possible $p_\ell$ and $s_m$ that can appear in the Nyldon factorization of a power of $w$. In fact, we can have at most 2 factors starting at an index $i$ and ending at an index $j$ for given $i, j$, namely $w_iw_{i+1} \cdots w_iw^a w_{i+1}w_{i+2} \cdots w_j$ for some $a$ and possibly $w_iw_{i+1} \cdots w_iw^a w_{i+1}w_{i+2} \cdots w_j$. We cannot have a repeated such factor unless $i = j$ (otherwise the starting indices of two consecutive factors are different), which is only possible if the factor is $n$. So, for large enough $k$, $n$ must appear as a Nyldon factor in $w^k$, as the total possible length of the $p_\ell$ and $s_m$ is bounded.

Using Mélançon’s algorithm, we provide a logarithmic bound on $K$.

**Theorem 15.** Suppose $n$ is Nyldon. Let $s$ be a nonempty suffix of $n$ (possibly all of $n$). Then, for any word $a$, $sn^ka$ cannot be Nyldon if $k > \log_2(\ell)$ where $\ell$ is the length of $n$.

Proof. We perform the factorization version of Mélançon’s algorithm on $sn^ka$. The main idea is that if there are enough repeated blocks of a Nyldon word in the middle of a word, then the influence of extra digits on the right cannot reach the leftmost block.

We have $k$ consecutive factors $n$ in our word; let us call these $n$-groups. We aim to prove that the leftmost $n$-group fully forms as a block, that is, a block in the final factorization begins with the entire leftmost $n$-group. Say that an $n$-group has become infected when a block to the right
of the $n$-group contracts into the rightmost block of the $n$-group. In other words, an $n$-group becomes infected when it no longer preserves the blocks. An $n$-group that is not infected is called uninfected.

We will prove the following claim: consider two consecutive $n$-groups $n_1, n_2$. Note that before $n_1$ and $n_2$ are infected, they look identical. Suppose that they are made up of $m$ blocks each right before $n_2$ is infected. Then, $n_1$ cannot become infected while it consists of more than $\frac{m}{2}$ blocks.

Consider two consecutive $n$-groups in our word, which we hereafter refer to as the left $n$-group and right $n$-group. Suppose that they currently look identical, as blocks $u_1, u_2, \ldots, u_m$ and $v_1, v_2, \ldots, v_m$ where each $u_i = v_i$, and then a block $s$ on the right merges with $v_m$. Now, blocks $v_m$ and $u_m$ are different, so we say that the infection has spread to $v_m$, and importantly, $v_m$ is longer than $u_m$. Here, $v_m$ is the leftmost infected block, and $u_m$ is its corresponding block. We label states with a triple (condition, blocks left, merge counter). "Merge counter" is the number of merges that have happened in the left $n$-group, "blocks left" is the number of blocks to the left of the leftmost infected block in the right $n$-group, and "condition" is whether the leftmost infected block, initially $v_m$, is longer or shorter than its corresponding block in the left $n$-group, initially $u_m$. Thus, we start in a state (longer, $m - 1$, 0).

We now outline the possibilities for this state to change. Suppose we start in a state (condition, $c$, $d$) where condition is either "longer" or "shorter." Note that the first $c$ blocks in the left $n$-group, which we denote $u_1, u_2, \ldots, u_c$, are respectively equal to the first $c$ blocks in the right $n$-group, which we call $v_1, v_2, \ldots, v_c$. If for some $i < c$ we have that $u_i$ and $u_{i+1}$ merge, then $v_i, v_{i+1}$ merge as well, and we are in a state (condition, $c - 1$, $d + 1$).

If for some $i > c$ we have that $u_i, u_{i+1}$ merge, then "merge counter" goes up, while "blocks left" remains the same. So, we end up in a state (condition, $c$, $d + 1$).

The interesting situation is when merges happen involving $u_c$ and $u_{c+1}$ or $v_c$ and $v_{c+1}$. If $u_c$ and $u_{c+1}$ merge, and $v_c, v_{c+1}$ merge, then we end up in the state (condition, $c - 1$, $d + 1$). But perhaps a merge happens in one $n$-group and not the other. If $u_c$ merges with $u_{c+1}$ but $v_c$ does not merge with $v_{c+1}$, then we say that the infection has spread by one block, as the $c^{th}$ blocks in each $n$-group are different. So, we end up in the state (shorter, $c - 1$, $d + 1$). The last case is if $v_c$ and $v_{c+1}$ merge, but $u_c, u_{c+1}$ do not. This phenomenon is only possible if $u_{c+1} >_{\text{lex}} v_{c+1}$, or $v_{c+1}$ is shorter than $u_{c+1}$. We then end up with the $c^{th}$ block in the right $n$-group being longer than the respective block in the left $n$-group. So, we go from (shorter, $c$, $d$) to (longer, $c - 1$, $d$). Call this type of step quirky.

Finally, the infection can only spread to the left $n$-group when $v_1$ is shorter than $u_1$. So, we must end in a "shorter" state. We start in a state (longer, $m - 1$, 0), and normally "merge counter" goes up by 1 whenever "blocks left" goes down by 1. The case when "blocks left" goes down by 1 but "merge counter" does not increase is only possible if "condition" is "shorter,"
and it changes to “longer.” So, if it takes \( f \) steps to get to “shorter” for the first time, we go from \((\text{longer}, m-1, 0)\) to \((\text{shorter}, m-1-f, f)\). Every quirky step is accompanied by changing the condition from “shorter” to “longer,” and to go back to “shorter” we must have at least one normal step. So, starting from \((\text{shorter}, m-1-f, f)\), the number of normal steps is at least the number of quirky steps, and therefore the “merge counter” will be at least \( f + \frac{m-1-f}{2} \geq \frac{m}{2} \).

So, we end up with at most \( \left\lfloor \frac{m}{2} \right\rfloor \) blocks in the left \( n \)-group by the time the infection spreads.

Hence, by time the infection spreads to a new \( n \) factor, the number of blocks decreases by at least half. After the infection spreads to \( \left\lfloor \log_2(\ell) \right\rfloor \) \( n \)-groups, an \( n \)-group will fully form, and cannot merge to anything on the left.

Therefore, the leftmost of the \( k n \)-groups will fully form, and so appending the Nyldon factorization of \( s \) to the Nyldon factorization of \( n^k a \) will yield a valid Nyldon factorization of \( sn^k a \), as by Lemma 1 the last Nyldon factor in \( s \) is lexicographically smaller than the first Nyldon factor of \( n^k a \). Thus \( sn^k a \) is not Nyldon and we are done.

**Theorem 16.** For a primitive word \( w \) with Nyldon conjugate \( n \), and large enough \( k \) depending on \( w \), the word \( w^k \) can be factorized as \((p_1, p_2, \ldots, p_i, n, n, \ldots, n, s_1, s_2, \ldots, s_j)\) for \( K = \left\lfloor \log_2(\ell) \right\rfloor + 1 \), where \( \ell \) is the length of \( w \).

**Proof.** We exactly follow the proof of Theorem 15, but are able to get an improvement of 1.

Suppose we use Mélançon’s algorithm on a word \( w^k \) for \( k \geq \left\lfloor \log_2(\ell) \right\rfloor + 1 \). We then have \( \left\lfloor \log_2(\ell) \right\rfloor \) \( n \)-groups in the middle of our word, with some suffix on the right. Suppose that when something from the right suffix contracts onto the rightmost \( n \)-group, that \( n \)-group consists of \( x \) blocks. Then, each time a \( n \)-group is infected, the number of blocks in the current \( n \)-group must decrease by at least half. If the number of \( n \)-groups is at least \( \left\lfloor \log_2(x) \right\rfloor + 1 \), then the leftmost \( n \)-group will fully form, and so not be able to contract onto any block to its left, as such a block will be a Nyldon suffix of \( n \), and therefore lexicographically less than \( n \) by Lemma 1.

Clearly \( n \) starts with its maximal letter, so the right suffix (after the \( k-1 \) blocks of \( n \) in \( w^k \)) does as well. So, by the time the maximal digit has been contracted in Mélançon’s algorithm, all digits will have been contracted and there will be no blocks left of length 1. Thus, the total number of blocks in the rightmost \( n \)-group, \( x \), is at most \( \frac{k}{2} \) when something contracts onto it. Then, \( k-1 \geq K-1 = \left\lfloor \log_2(\ell) \right\rfloor \geq \left\lfloor \log_2(x) \right\rfloor + 1 \), and we are done.

We provide an example of how the above two proofs work, by showing Mélançon’s algorithm on a specific power. Consider the word \( w = 01111011011110110111 \), with Nyldon conjugate \( n = 1w1^{-1} = 10111101101111101111111111 \). We show that in this case, \( K = 4 \) (as shown in [3]), by factorizing \( w^5 \).
Note that \( w^5 = 011110110111101110111011 \). The prefix 011110110111101110111011 will be factorized and stay the same in front of \( n \), so we focus only on factorizing \( n^4 \). We use semicolons to show the barriers between the \( n \) factors.

1. Start with \( n^4 = 1 \):
   
   1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1;  
   1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1;  
   1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1;  
   1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1;  
   1.  

2. Contract the 0s to get:

   10, 1, 1, 1, 10, 1, 1, 1, 1, 10, 1, 1, 1, 10, 1, 1;  
   10, 1, 1, 1, 10, 1, 1, 1, 1, 10, 1, 1, 1, 10, 1, 1;  
   10, 1, 1, 1, 10, 1, 1, 1, 1, 10, 1, 1, 1, 10, 1, 1;  
   1.  

3. Contract the 1s to get:

   10111, 101, 101111, 10111, 1011; 10111, 101, 101111, 10111, 1011;  
   10111, 101, 101111, 10111, 1011; 10111, 101, 101111, 10111, 10111. The last of the four \( n \) factors is different from the first three, so the infection starts. The number of blocks per \( n \) factor is 5.

4. Contract the 101s, to get

   10111101, 101111, 101111, 101111, 101111, 101111, 101111;  
   10111101, 101111, 101111, 101111, 101111, 101111, 101111. The number of blocks per \( n \) factor is 4.

5. Contract the 1011s to get

   10111101, 101111, 10111101; 10111101, 101111, 10111101;  
   10111101, 101111, 10111101; 10111101, 101111, 101111, 101111. The number of blocks per correct \( n \) factor is 3.

6. Contract the 10111s to get

   10111101, 101111, 10111101; 10111101, 101111, 10111101;  
   10111101, 101111, 10111101; 10111101, 10111101110111.  

7. Contract the 101111s to get

   10111101, 10111101; 10111101110111, 10111101;  
   10111101110111, 10111101110111, 10111101. The number of blocks per uninfected \( n \) factor is 2.
8. Contract the 10111101 to get
   10111101101111, 1011110111011111, 101111011;
   10111101101111, 1011110111011111, 1011111011111. The infection has spread to the
   next $n$ factor, with the number of blocks decreasing from 5 to 2.

9. Contract the 101111011s to get
   10111101101111101111011101111101101111,
   1011110110111110111101110111101101111,
   10111101110111101
   1011111011110111. We now have $n$ fully formed! The next step will create
   10111101101111101111011101111101101111,
   10111101110111101
   1011111011110111, but the leftmost $n$ factor will remain unaltered.

As only one $n$ appears in the Nyldon factorization of $w^5$ in this example, we have $K = 4$.

**Definition.** A subset $F$ of $A^*$ is a *code* if all possible concatenations of (not necessarily distinct) words in $F$ yield distinct words.

**Definition.** Let $F \subseteq A^*$ be a code, and let $F^*$ be the set of all possible concatenations of words in $F$. Then $F$ is a *circular code* if for any words $u$ and $v$, we have $uv, vu \in F^*$ implies $u, v \in F^*$.

In other words, a code is circular if whenever we take a concatenation of words in the code and put the string in a circle, we can recover the original sequence of words. For example, \{00, 01, 10\} is not a circular code, as (00, 10) forms the same circular word as (01, 00). It turns out that Lyndon words over a given alphabet of a fixed length form a circular code \[\text{[1]}\]. We prove a similar result for Nyldon words.

**Theorem 17.** Nyldon words of any fixed length $\ell$ form a circular code.

**Proof.** Suppose we have a sequence of Nyldon words $w_1, w_2, \ldots, w_k$, each of length $\ell$, and we write the word $w_1 w_2 \cdots w_k$. Let us start $w_1 w_2 \cdots w_k$ as $u_1 u_2 \cdots u_{kt}$ for blocks $u_i$ just being individual letters, and perform the conjugate version of Mélançon’s algorithm on this as a circular word. We claim that the first time a block of length at least $\ell$ is formed is when some Nyldon word $w_j$ is fully formed as a block.

For sake of contradiction, suppose that a block of length $\ell$ forms without any word $w_j$ fully forming. Then, since the words $w_1 w_2 \cdots w_k$ are originally made up separately, at some point the first block in some word $w_i$ will be joined to the first block in a word $w_{i-1}$, where $w_1 = w_{n+1}, w_0 = w_n$, etc. Right before this happens, because by assumption no word $w_i$ is fully formed, let the first block in each $w_i$ be $p_i$ and the last block be $s_i$.

We have that $p_i$ is being joined to $s_{i-1}$, so $p_i$ is the lexicographically smallest of all the blocks still remaining. In particular, $p_i$ is the lexicographically smallest of the blocks still remaining that make up $w_i$, of which there are at least two. But then by Lemma 3, $w_{i+1}$ cannot be a
Nyldon word, which is a contradiction. Therefore, blocks will never cross the original Nyldon word boundaries until strictly after at least one is formed, and that will be the first word to form of length at least $\ell$.

Now, suppose for sake of contradiction that circularly shifting $w_1 w_2 \cdots w_k$ by a non-multiple of $\ell$ gives $v_1 v_2 \cdots v_k$, where the $v_i$ are all Nyldon words of length $\ell$. By the argument above, the first block of length $\ell$ formed is still some $w_j$. This means that a boundary is crossed in the $v_i$'s before any block $v_i$ is formed, which is impossible by the argument above and symmetry between the $w_j$'s and $v_j$'s. Therefore, there is no possible other sequence $v_1 v_2 \cdots v_k$ that forms the same circular word as $w_1 w_2 \cdots w_k$.

5 Another Algorithm

In [3], Charlier et al. provide an algorithm for computing the Nyldon factorization of a word. We prove that this algorithm is linear in the length of the word. Thus the Nyldon factorization can be computed in linear time, just like for the Lyndon factorization, as shown by Duval [5]. We reproduce the algorithm here.

Require: $w \in A^+$
Ensure: NylF is the Nyldon factorization of $w$

\begin{algorithm}
\begin{algorithmic}
\State $n \leftarrow \text{length}(w)$, NylF $\leftarrow (w[n])$
\For{$i = 1$ to $n - 1$}
\State NylF $\leftarrow (w[n - i], \text{NylF})$
\While{length(NylF) $\geq 2$ and NylF(1) $\geq_{\text{lex}}$ NylF(2)}
\State NylF $\leftarrow (\text{NylF}(1) \cdot \text{NylF}(2), \text{NylF}(3), \ldots, \text{NylF}(-1))$
\EndWhile
\EndFor
\Return NylF
\end{algorithmic}
\end{algorithm}

Algorithm 2 ([3]): Computing the Nyldon factorization

The number of initial lexicographic comparisons is equal to the number of digits in $w$, and the number of additional lexicographic comparisons is at most the number of times two words combine into a bigger word. There are $|w| - 1$ barriers between words, so the total number of lexicographical comparisons is at most $2|w| - 1$, which is linear in $w$. One way to do comparisons fast is with a suffix array, which takes $O(|w|)$ time to construct, giving a least common prefix (LCP) array [9, 8]. Comparing substrings of $w$ is equivalent to a range minimum query (RMQ) on the LCP array between the indices representing the suffixes where the substrings start. We
can do RMQ in constant time with linear preprocessing time [6]. Therefore, we obtain a runtime of $O(|w|)$.

We can also use Mélançon’s algorithm to factorize words. If we store the current blocks in a heap, and simply contract the current minimum each time, we get a time complexity of the previous algorithm increased by a logarithmic factor. So, Mélançon’s algorithm for factorization or finding the Nyldon conjugate has a runtime of $O(|w| \log |w|)$.

6 The Lazard Procedure

Let $A \leq^n n$ be the set of words on $A$ with length at most $n$. Also, for a set of words $X$ and word $w$, let $Xw^* = X \cup Xw \cup Xww \cup \cdots$, where $Xw^i$ denotes the set of words $\{xw^i : x \in X\}$.

**Definition.** A right Lazard set is a subset $F$ of $A^+$ with a total order $<$ satisfying the following property: suppose $F \cap A \leq^n = \{u_1, u_2, \ldots, u_k\}$ with $u_1 < u_2 < \cdots < u_k$. Let $Y_i$ be a sequence of sets defined as $Y_1 = A$, and for $i \geq 2$, $Y_i = (Y_{i-1} \setminus u_{i-1})u_{i-1}^*$. Then, for all $1 \leq i \leq k$, $u_i \in Y_i$, and $Y_k \cap A \leq^n = u_k$.

The Lazard procedure is the act of generating the $Y_i$ by choosing $u_i$ in $Y_i$, removing it from $Y_i$, and creating $Y_{i+1}$. We can think of the total order on $F$ being induced by the procedure itself by the choice of $u_i$.

In [3], Charlier et al. prove that the Nyldon words form a right Lazard set, and conjecture that the right Lazard procedure generates all the Nyldon words up to a certain length much before the procedure ends, unlike the procedure for Nyldon words (Open Problem 59). We explicitly determine the number of steps it takes to generate all the Nyldon words up to a given length. For reference, we reproduce the example in [3] of the right Lazard procedure on binary Nyldon words of length at most 5. Note that all words have been generated by the fourth step.

**Lemma 18.** Every Nyldon word can be generated in exactly one way by the Lazard procedure.

**Proof.** We can write every Nyldon word $w$ that is not a single letter as $au_{i_1}u_{i_2}\cdots u_{i_k}$ where $a \in A$ and $i_1 \leq i_2 \leq \cdots \leq i_k$ as it is generated by the procedure. However, $u_{i_1}u_{i_2}\cdots u_{i_k}$ is then the Nyldon factorization of $a^{-1}w$, which is unique, so there is only one way to generate $w$ during the Lazard procedure.

Now, we determine the step at which all Nyldon words have been generated. Suppose we work with an alphabet $\{0, 1, \ldots, m\}$.

**Lemma 19.** The largest Nyldon word lexicographically of length at most $\ell$ over an alphabet $A = \{0, 1, \ldots, m\}$ is $m(m-1)\underbrace{mm\cdots m}_{\ell-2 \text{ copies}}$. 

15
Proof. No Nyldon word \( w \) can start with \( mm \), as adding \( m \) to the Nyldon factorization of \( m^{-1}w \) yields a valid Nyldon factorization with at least 2 factors. So, \( m(m - 1)\underbrace{mm \cdots m}_{\ell - 2 \text{ copies}} \) is lexicographically the largest possible such word remaining. This word is clearly Nyldon by any factorization algorithm, so we are done.

Lemma 20. Over an alphabet \( A = \{0, 1, \ldots, m\} \), if a word \( v \) has length at most \( \ell - 3 \), then \( w = m(m - 1)\underbrace{mm \cdots m}_{\ell - 2 \text{ copies}} v \) is Nyldon.

Proof. If we perform Mélançon’s algorithm on \( w \), then a block of length \( \ell - 1 \) will form at the beginning of the word. This block is lexicographically greater than any block to the right, since any such block has length less than \( \ell - 1 \), and our first block is the lexicographically largest Nyldon word of length at most \( \ell - 1 \) by Lemma 19. So, the whole word will become Nyldon.

Proposition 21. Let the step at which the Lazard procedure generates all the Nyldon words of length up to \( \ell \) be the step corresponding to \( u_i \). If \( \ell \) is odd, then \( u_i = m(m - 1)\underbrace{mm \cdots m}_{\ell - 2 \text{ copies}} \) where...
there are $\frac{\ell - 5}{2}m$s at the end, and if $\ell$ is even, then $u_i = m(m - 1)mmm \cdots m(m - 1)$, where there are $\frac{\ell - 6}{2}m$s in the middle.

**Proof.** The main idea is that the chosen $u_i$ is the lexicographically largest Nyldon word that can still be affixed to the end of a larger Nyldon word within the length limits. Suppose that by the time $u_i$'s step happens, there is still some Nyldon word that has not yet appeared. Then, it must equal $ab$ where $a, b$ are Nyldon, and $a \succ_{\text{lex}} b \succ_{\text{lex}} u_i$.

If $\ell$ is odd, then if $a, b$ are greater than $u_i$ lexicographically, they each must have a length at least $\frac{\ell + 1}{2}$ since $u_i$ is the largest Nyldon word lexicographically with a length of at most $\frac{\ell - 1}{2}$ by Lemma 19.

If $\ell$ is even, then $a$ and $b$, being greater than $u_i$ lexicographically, must each have length at least $\frac{\ell}{2}$. But then the only possibility is $a = b = u_i(m - 1)^{-1}m$, which is impossible because $a$ and $b$ cannot be equal. So, we have proved that by step $u_i$, every Nyldon word is generated.

Furthermore, at step $u_i$, if $\ell$ is odd, then $(u_i,m)u_i$ is generated, and if $\ell$ is even, then $(u_i(m - 1)^{-1}m)u_i$ is generated, so a new word is generated. So, we are done. \qed

Now, we want to calculate how far from the end the Lazard procedure has generated all the Nyldon words, or equivalently, calculate how many Nyldon words of length at most $\ell$ are lexicographically greater than $u_i$.

**Proposition 22.** Suppose $\ell = 2n + 1$ for $n \geq 7$, and let $u_i = m(m - 1)m \cdots m$, where $u_i$ has length $n$, and the alphabet $A = \{0, 1, \ldots, m\}$. Then, the number of Nyldon words lexicographically greater than $u_i$ and with length at most $\ell$ is

$$\left|\frac{A^{n+2} - |A|}{|A| - 1}\right| - (|A|^3 + |A|^2 + 2|A| + 2).$$

**Proof.** Suppose $\ell = 2n + 1$ for $n \geq 7$. Most words of length at most $\ell$ beginning with $u_i$ are Nyldon, so we will instead count the words which are not. Let $w$ be a word beginning with $u_i$ and of length at most $\ell$, but $w$ is not Nyldon. We split into cases, where each time we perform Mélançon’s algorithm.

- **Case 1:** $w$ begins with $u_i m$.

  Then, when performing Mélançon’s algorithm, a block $u_i$ will form at the beginning of the word. For $w$ to not be Nyldon, the Nyldon block to the right of $u_i$ must be lexicographically at least $u_i$, at some point. Otherwise, it would contract into the first block, and any remaining blocks would have a length at most $n$ and therefore contract into the block to the left, making the whole word Nyldon. So, this second block must have length $n + 1$, and so it equals $u_i d$ for a letter $d$ or is simply $u_i$. There are $|A| + 1 = (m + 1) + 1$ possibilities in this case, depending on whether $w$ has length $2n + 1$ or $2n$, respectively.

- **Case 2:** $w$ begins with $u_i(m - 1)$.
Then, when performing Mélançon’s algorithm, a block of $u_im^{-1}$ will form at the beginning of the word, which is the largest Nyldon word lexicographically of length at most $n-1$. To the right of that, a block starting with $m(m-1)$, and therefore of length at least 2, will form. We claim that this “middle” block cannot combine with the block to its left. Suppose it does. Then, the leftmost block will be of length at least $n+1$ and start with $u_i$, and so no Nyldon block to the right of it can be lexicographically greater than it, so the whole word $w$ will become Nyldon.

So, the “middle” block, the one starting at the $n$th index, must eventually become lexicographically greater than $u_im^{-1}$. Therefore, the word made up from indices $n$ through $(2n-2)$ must be another copy of $u_i$. By Lemma 20, the suffix of $w$ starting at the $n$th index is then Nyldon as long as $n$ and $\ell$ are large enough, specifically $n \geq 7$. We then get $1+\sum k + \sum l + \sum m$, possible words that are not Nyldon, depending on whether $w$ has length $2n-2, 2n-1, 2n$, or $2n+1$, respectively.

- **Case 3:** $w$ begins with $u_id$ for a letter $d \leq m-2$.

When performing Mélançon’s algorithm, a block of $u_im^{-1}$ will form at the beginning of the word, and will be lexicographically greater than the block to its right. When they combine, the block will start with $u_i$, and be greater than any block to the right. So, the whole word will be Nyldon.

Therefore, the total number of words lexicographically greater than $u_i$ with length at most $\ell = 2n+1$ which are not Nyldon, for $n \geq 7$, is $|A|^3 + |A|^2 + 2|A| + 2$. There are $|A|^k$ words of length $n+k$ starting with $u_i$ for $1 \leq k \leq n+1$, and all are Nyldon except the aforementioned words. So, using the fact that $|A| + |A|^2 + \cdots + |A|^{n+1} = \frac{|A|^{n+2} - |A|}{|A| - 1}$, we have the result. 

**Proposition 23.** Suppose $\ell = 2n$ for $n \geq 9$, and let $u_i = m(m-1)m \cdots m(m-1)$, where $u_i$ has length $n$, and the alphabet $A = \{0, 1, \ldots, m\}$. Then, the number of Nyldon words lexicographically greater than $u_i$ and with length at most $\ell$ is $\frac{|A|^n - |A|}{|A| - 1} - (|A|^4 + |A|^3 + |A|^2 + |A| + 3)$.

**Proof.** As before, we will count the words beginning with $u_i$ of length at most $\ell$ that are not Nyldon. Let $w$ be such a word, and suppose we perform Mélançon’s algorithm on $w$. Either a block of length $\ell - 2$ or $\ell$ forms at the beginning of the word $w$, giving us two cases.

- **Case 1:** A block of length $n$ forms.

Then, the rest of $w$ must be either $u_i$ or $u_i(m-1)^{-1}$, since it starts with (and therefore is) a Nyldon word lexicographically at least $u_i$. So, we get 2 possible words.

- **Case 2:** A block of length $n-2$ forms.
Starting at index \( n - 1 \), we must have the word \( u_i(m(m - 1))^{-1} \). For \( n \) large enough, the last 0, 1, 2, 3, or 4 digits can be anything by Lemma 20. So, we get \(|A|^4 + |A|^3 + |A|^2 + |A| + 1\) possible words.

Now, note that the cases are distinct, since Case 1 requires \((m - 1)\) to be the \((n + 2)\)nd letter of \( w \), whereas it is \( m \) for Case 2. Therefore, the total number of non-Nyldon words is \(|A|^4 + |A|^3 + |A|^2 + |A| + 3\). There are \(|A|^k\) words of length \( n + k \) starting with \( u_i \) for \( 1 \leq k \leq n \), and all are Nyldon except the aforementioned words. So, using the fact that \(|A| + |A|^2 + \cdots + |A|^n = \frac{|A|^{n+1} - |A|}{|A| - 1}\), we have the result.

Using Lazard sets, we can find more codes made up of Nyldon words.

**Theorem 24 (Kraft-McMillan Inequality).** [10, 13] Suppose that a uniquely decodable code \( \{s_1, s_2, \ldots \} \) has codeword lengths \( \{\ell_1, \ell_2, \ldots \} \) over an alphabet of size \( r \). Then,

\[
\sum_i r^{-\ell_i} \leq 1.
\]

**Theorem 25.** Fix \( n \). For \( F \) the set of Nyldon words and \( F \cap A^{\leq n} = \{u_1, u_2, \ldots, u_k\} \) with \( u_1 <_{lex} u_2 <_{lex} \cdots <_{lex} u_k \), let \( Y_i \) be the sequence of Lazard sets defined earlier. Note that each \( Y_i \) is infinite. Then, each \( Y_i \) forms a uniquely decodable code and satisfies the equality case of the Kraft-McMillan inequality.

**Proof.** We use induction. Suppose our alphabet is \( A = \{0, 1, \ldots, m\} \). We start with the set \( Y_1 = \{0, 1, \ldots, m\} \), which is clearly a uniquely decodable code satisfying the equality case of the Kraft-McMillan equality. Suppose that \( Y_i = \{a_1, a_2, \cdots \} \) satisfies these conditions, and \( Y_{i+1} = (Y_i \setminus a_j) a_{j}^* \) for some \( j \), where \( a_j = u_i \) in the definition of a right Lazard set. Every word \( w \) that can be formed from the words in \( Y_i \) can clearly be formed in at most one way from the words in \( Y_{i+1} \), obtained by putting the non-\( a_j \) words forming \( w \) together, and adding the appropriate amount of \( a_j \)’s in between. So, \( Y_{i+1} \) is a uniquely decodable code. Now, we need to show that \( Y_{i+1} \) satisfies the equality case of the Kraft-McMillan inequality. By deleting \( a_j \), we get an exponential sum of

\[
\sum_{k \neq a_j} \frac{1}{(m+1)^{|a_k|}} \left( \frac{1}{(m+1)^{|a_j|}} + \frac{1}{(m+1)^2|a_j|} + \cdots \right) = \\
\frac{1 - \frac{1}{(m+1)^{|a_j|}}}{1 - \frac{1}{(m+1)^{|a_j|}}} = 1
\]

Notably, the proof of Theorem 25 does not depend on the choice of \( u_i \) in each step of the Lazard procedure.
7 Lyndon Words

Recall that Lyndon words are those words \( w \) that cannot be factorized into a sequence of smaller Lyndon words \( w_1 w_2 \cdots w_k \) with \( w_1 \geq_{\text{lex}} w_2 \geq_{\text{lex}} \cdots \geq_{\text{lex}} w_k \). It turns out that Mélançon’s algorithm also gives Lyndon words! The only change is that we contract the smallest word lexicographically with its right neighbor instead of its left neighbor at each step. The reason that the algorithm works with this modification is that the Lyndon words form a left Hall set; see [14] and Sections 11 and 12 of [3] for more information.

Furthermore, we give a positive answer to a question posed in [3].

**Theorem 26.** If a word \( w \) is lexicographically smaller than all of its Lyndon proper suffixes, then it is Lyndon.

**Proof.** We will only use the recursive definition of Lyndon words to prove this implication. Suppose \( w \) is not Lyndon. Then, by the recursive definition of Lyndon words, \( w \) has a factorization \( w_1 w_2 \cdots w_k \) into Lyndon words with \( w_1 \geq_{\text{lex}} w_2 \geq_{\text{lex}} \cdots \geq_{\text{lex}} w_k, \ k \geq 2 \). But then \( w_k \) is a Lyndon proper suffix of \( w \) and \( w >_{\text{lex}} w_1 \geq_{\text{lex}} w_k \), so \( w \) is not lexicographically smaller than all of its Lyndon proper suffixes, and we are done.

8 Further Directions

In [3], Charlier et al. give the problem of describing the forbidden prefixes of Nyldon words (Open Problem 11). They also ask whether prefixes of Nyldon words must always be sesquipowers (or fractional powers) of Nyldon words (Open Problem 12). These problems both remain open, and we in general know very little about prefixes of Nyldon words.

We also wonder how much the bound \( K \leq \left\lfloor \log_2(\ell) \right\rfloor + 1 \) can be improved in Nyldon factorizing \( w^k \), with \( w \) of length \( \ell \), as \((p_1, p_2, \ldots, p_i, n, n, \ldots, n, s_1, s_2, \ldots, s_j)\). In particular, we ask whether there is a way to construct examples of \( w \) with arbitrarily large \( K \). So far, we have not found any word \( w \) with a value of \( K \) more than 4.

9 Acknowledgments

This research was funded by NSF/DMS grant 1659047 and NSA grant H98230-18-1-0010. The author would like to thank Prof. Joe Gallian for organizing the Duluth REU where this research took place, as well as advisors Aaron Berger and Colin Defant. The author would also like to thank Amanda Burcroff and Sumun Iyer for providing helpful comments on drafts of this paper.
Finally, the author would like to thank Spencer Compton for pointing the author toward helpful resources discussing the complexity of certain algorithms.

References

[1] J. Berstel, D. Perrin, and C. Reutenauer. Codes and Automata. *Encyclopedia of Mathematics and its Applications, Cambridge University Press*, 129, 2010.

[2] P. Bonizzoni, C. D. Felice, R. Zaccagnino, and R. Zizza. Inverse Lyndon words and inverse Lyndon factorizations of words. *Adv. Appl. Math.*, 101:281–319, 2018.

[3] E. Charlier, M. Philibert, and M. Stipulanti. Nyldon words. *J. Combin. Theory, Ser. A*, 167:60–90, 2019.

[4] K.-T. Chen, R. H. Fox, and R. C. Lyndon. Free differential calculus. IV. The quotient groups of the lower central series. *Ann. of Math.*, 68(1):81–95, 1958.

[5] J.-P. Duval. Factorizing words over an ordered alphabet. *J. Algorithms*, 4(4):363–381, 1983.

[6] J. Fischer and V. Heun. Theoretical and practical improvements on the RMQ-problem, with applications to LCA and LCE. *Proceedings of the 17th Annual Symposium on Combinatorial Pattern Matching*, pages 36–48, 2006.

[7] D. Grinberg. “Nyldon words”: understanding a class of words factorizing the free monoid increasingly, 2014. [https://mathoverflow.net/questions/187451/](https://mathoverflow.net/questions/187451/).

[8] T. Kasai, G. Lee, H. Arimura, S. Arikawa, and K. Park. Linear-time longest-common-prefix computation in suffix arrays and its applications. *Proceedings of the 12th Annual Symposium on Combinatorial Pattern Matching*, pages 181–192, 2001.

[9] P. Ko and A. Srinivas. Linear time construction of suffix arrays. *Computer Science Technical Reports, Digital Repository @ Iowa State University*, 2002.

[10] L. G. Kraft. A device for quantizing, grouping, and coding amplitude-modulated pulses. Master’s thesis, Massachusetts Institute of Technology, 1949.

[11] M. Lothaire. *Combinatorics on Words*. Cambridge University Press, Cambridge, 1997.

[12] R. C. Lyndon. On Burnside’s problem. *Trans. Amer. Math. Soc.*, 77:202–215, 1954.

[13] B. McMillan. Two inequalities implied by unique decipherability. *IEEE Trans. Inf. Theory*, 2(4):115–116, 1956.

[14] G. Mélançon. Combinatorics of Hall trees and Hall words. *J. Combin. Theory, Ser. A*, 59(2):285–308, 1992.
[15] D. Perrin and C. Reutenauer. Hall sets, Lazard sets and comma-free codes. *Discrete Math.*, 341:231–243, 2018.

[16] A. I. Shirshov. Subalgebras of free Lie algebras. *Mat. Sb.*, 75(2):441–452, 1953.

[17] A. I. Shirshov. On free Lie rings. *Mat. Sb. N.S.*, 45(87):113–122, 1958.