# Enumerating all the Irreducible Polynomials over Finite Field

Nader H. Bshouty
Technion
Dept. of Computer Science
Haifa, 32000
Israel
bshouty@cs.technion.ac.il

Nuha Diab
Sisters of Nazareth High School
Grade 12
P.O.B. 9422, Haifa, 35661
Israel
nuha15.3.98@gmail.com

Shada R. Kawar
Nazareth Baptist High School
Grade 11
P.O.B. 20, Nazareth, 16000
Israel
shadakawar.135@gmail.com

Robert J. Shahla
Sisters of Nazareth High School
Grade 11
P.O.B. 9422, Haifa, 35661
Israel
shahla.robert@gmail.com

*Abstract:* In this paper we give a detailed analysis of deterministic and randomized algorithms that enumerate any number of irreducible polynomials of degree $n$ over a finite field and their roots in the extension field in quasilinear* time cost per element. Our algorithm is based on an improved algorithm for enumerating all the Lyndon words of length $n$ in linear delay time and the known reduction of Lyndon words to irreducible polynomials.

*Key–Words:* Finite Field, irreducible polynomilas, Lyndon words.

## 1 Introduction

The problem of enumerating the strings in a language $L$ is to list all the elements in $L$ in some order. Several papers study this problem. For example, Enumerating all spanning trees, [25], minimal transversals for some Geometric Hypergraphs, [14], maximal cliques, [33], ordered trees, [13], certain cuts in graphs, [47, 53], paths in a graph, [39], bipartite perfect matchings, [45], maximum and maximal matchings in bipartite graphs, [44], and directed spanning trees in a directed graph [43]. See the list in [18] for other enumeration problems.

One of the challenges in enumeration problems is to find an order of the elements of $L$ such that finding the next element in that order can be done in quasilinear time in the length of the representation of the element. The time that the algorithm takes before giving the first element is called the *preprocessing time*. The time of finding the next element is called the *delay time*. In [3], Ackerman and Shallit gave a linear preprocessing and delay time for enumerating the words of any regular language (expressed as a regular expression or NFA) in lexicographic order.

Enumeration is also of interest to mathematicians without addressing the time complexity. Calkin and Wilf,[8], gave an enumeration of all the rational numbers such that the denominator of each fraction is the numerator of the next one.

Another problem that has received considerable attention is the problem of ranking the elements of $L$. In ranking the goal is to find some total order on the elements of $L$ where the problem of returning the $n$th element in that order can be solved in polynomial time. Obviously, polynomial time ranking implies polynomial time enumeration. In the literature, the problem of ranking is already solved for permutations [35, 42] and trees of special properties [21, 30, 36, 38, 46, 51, 52, 1, 49, 50]. Those also give enumerating algorithms for such objects.

Let $\mathbb{F}_q$ be a finite field with $q$ elements. Let $P_{n,q}$ be the set of irreducible polynomials over $\mathbb{F}_q$ of degree $n$ and their roots in $\mathbb{F}_{q^n}$. Several algorithms in the literature use irreducible polynomials of degree $n$ over finite fields, especially algorithms in coding theory, cryptography and problems that use the Chinese Remainder Theorem for polynomials [6, 31, 4, 12]. Some other algorithms use only the roots of those polynomials. See for example [4].

In this paper, we study the following problems

1. Enumeration of any number of irreducible poly-

*$O(N \cdot poly(\log N))$ where $N = n^2$ is the size of the output.

nomials of degree $n$ over a finite fields.

2. Enumeration of any number of irreducible polynomials of degree $n$ and their roots over the extended field.

3. Enumeration of any number of roots of irreducible polynomials of degree $n$ over the extended field. One root for each polynomial.

There are many papers in the literature that mention the result of enumerating all the irreducible polynomials of degree *less than or equal* to $n$ but do not give the exact algebraic complexity of this problem [7, 11, 37, 16, 17, 26]. In this paper we give a detailed analysis of deterministic and randomized algorithms that enumerate any number of irreducible polynomials of degree $n$ over a finite field and/or their roots in the extension field in quasilinear[1] time cost per element.

Our algorithm is based on an improved algorithm for enumerating all the Lyndon words of length $n$ in linear delay time and the well known reduction of Lyndon words to irreducible polynomials. In the next subsection we define the Lyndon word and present the result of the improved algorithm.

### 1.1 The Enumeration of Lyndon Words

Let $<$ be any total order on $\mathbb{F}_q$. A *Lyndon word* (or string) over $\mathbb{F}_q$ of length $n$ is a word $w = w_1 \cdots w_n \in \mathbb{F}_q^n$ where every rotation $w_i \cdots w_n w_1 \cdots w_{i-1}$, $i \neq 1$ of $w$ is lexicographically larger than $w$. Let $L_{n,q}$ be the set of all the Lyndon words over $\mathbb{F}_q$ of length $n$. In many papers in the literature, it is shown that there is polynomial time (in $n$) computable bijective function $\phi : L_{n,q} \to P_{n,q}$, where $P_{n,q}$ is the set of all polynomials of degree $n$ over $\mathbb{F}_q$. So the enumeration problem of the irreducible polynomials can be reduced to the problem of enumerating the elements of $L_{n,q}$.

Bshouty gave in [4] a large subset $L' \subseteq L_{n,q}$ where any number of words in $L'$ can be enumerated in a linear delay time. In fact, one can show that $L'$ has a small DFA and, therefore, this result follows from [8]. It is easy to show that the set $L_{n,q}$ cannot be accepted by a small size NFA, i.e., size polynomial in $n$, so one cannot generalize the above result to all $L_{n,q}$. Duval [11] and Fredricksen et. al., [16, 17] gave enumeration algorithms of all the words in $\cup_{m \leq n} L_{m,q}$ that run in linear delay time. Berstel and Pocchiola in [5] and Cattell et. al. in [7, 37] show that, in Duval's algorithm, in order to find the next Lyndon word in $\cup_{m \leq n} L_{m,q}$, the amortized number of *updates* is constant. The number of updtes is the number of symbols that the algorithm change in a Lyndon word in order to get the next word. Such an algorithm is called CAT algorithm. See the references in [7] for other CAT algorithms. Kociumaka et. al. gave an algorithm that finds the rank of a Lyndon word in $O(n^2 \log q)$ time and does unranking in $O(n^3 \log^2 q)$ time.

In this paper, we give an enumeration algorithm of $L_{n,q}$ with linear delay time. Our algorithm is the same as Duval's algorithm with the addition of a simple data structure. We show that this data structure enable us to find the next Lyndon word of *length* $n$ in constant updates per symbol and therefore in linear time. We also show that our algorithm is CAT algorithm and give an upper bound for the amortized update cost.

Another problem is testing whether a word of length $n$ is Lyndon word. In [10], Duval gave a linear time algorithm for such test. In this paper we give a simple algorithm that uses the suffix trie data structure and runs in linear time.

This paper is organized as follows. In Section 2 we give the exact arithmetic complexity of the preprocessing and delay time for enumerating any number of irreducible polynomials and/or their roots. In Section 3 we give a simple data structure that enable us to change Duval's algorithm to an algorithm that enumerates all the Lyndon words of length $n$ in linear delay time. We then show in Section 4 that the algorithm is CAT algorithm. In Section 5 we give a simple linear time algorithm that tests whether a word is a Lyndon word.

## 2 Enumerating Irreducible Polynomials

In this section we give the analysis for the algebraic complexity of the preprocessing time and delay time of enumerating irreducible polynomials of degree $n$ over a finite field and/or their roots in the extended field.

Let $q$ be a power of a prime $p$ and $\mathbb{F}_q$ be the finite field with $q$ elements. Our goal is to enumerate all the irreducible polynomials of degree $n$ over $\mathbb{F}_q$ and/or their roots in the extension field $\mathbb{F}_{q^n}$.

The best deterministic algorithm for constructing an irreducible polynomial over $\mathbb{F}_q$ of degree $n$ has time complexity $T_D := O(p^{1/2+\epsilon} n^{3+\epsilon} + (\log q)^{2+\epsilon} n^{4+\epsilon})$ for any $\epsilon > 0$. The best randomized algorithm has time complexity $T_R := O((\log n)^{2+\epsilon} n^2 + (\log q)(\log n)^{1+\epsilon} n)$ for any $\epsilon > 0$. For a comprehensive survey of this problem see [40] Chapter 3. Obviously, the preprocessing time for enumerating irreducible polynomials cannot be less than

[1] $O(N \cdot poly(\log N))$ where $N = n^2$ is the size of the output.

the time for constructing one. Therefore, $T_D$ for the deterministic algorithm, and $T_R$ for the randomized algorithm.

The main idea of the enumeration algorithm is to enumerate the roots of the irreducible polynomials in the extension field and then construct the polynomials from their roots. Let $\mathbb{F}_{q^n}$ be the extension field of $\mathbb{F}_q$ of size $q^n$. One possible representation of the elements of the field $\mathbb{F}_{q^n}$ is by polynomials of degree at most $n-1$ in $\mathbb{F}_q[\beta]/(f(\beta))$ where $f(x)$ is an irreducible polynomial of degree $n$. A *normal basis* of $\mathbb{F}_{q^n}$ is a basis over $\mathbb{F}_q$ of the form $N(\alpha) := \{\alpha, \alpha^q, \alpha^{q^2}, \ldots, \alpha^{q^{n-1}}\}$ for some $\alpha \in \mathbb{F}_{q^n}$ where $N(\alpha)$ is linearly independent. The *normal basis theorem* states that for every finite field $\mathbb{F}_{q^n}$ there is a normal basis $N(\alpha)$. That is, an $\alpha$ for which $N(\alpha)$ is linearly independent over $\mathbb{F}_q$. It is known that such an $\alpha$ can be constructed in deterministic time $O(n^3 + (\log n)(\log\log n)(\log q)n)$ and randomized time $O((\log\log n)^2(\log n)^4 n^2 + (\log n)(\log\log n)(\log q)n)$ [22, 27, 29]. The enumeration algorithm will use the normal basis for representing the elements of $\mathbb{F}_{q^n}$. Notice that the time complexity to find such an element $\alpha$ is less than constructing one irreducible polynomial. If we use the normal basis $N(\alpha)$ for the representation of the elements of $\mathbb{F}_{q^n}$, then every element $\gamma \in \mathbb{F}_{q^n}$ has a unique representation $\gamma = \lambda_1\alpha + \lambda_2\alpha^q + \lambda_3\alpha^{q^2} + \cdots + \lambda_n\alpha^{q^{n-1}}$ where $\lambda_i \in \mathbb{F}_q$ for all $i$.

It is known that any irreducible polynomial $g$ of degree $n$ over $\mathbb{F}_q$ has $n$ distinct roots in $\mathbb{F}_{q^n}$. If one can find one root $\gamma \in \mathbb{F}_{q^n}$ of $g$ then the other roots are $\gamma^q, \gamma^{q^2}, \ldots, \gamma^{q^{n-1}}$ and therefore $g_\gamma(x) := (x-\gamma)(x-\gamma^q)\cdots(x-\gamma^{q^{n-1}}) = g(x)$. The coefficients of $g_\gamma(x)$ can be computed in quadratic time $O(n^2 \log^3 n (\log\log n)^2)$. See Theorem A and B in [40] and references within. The element $\gamma = \lambda_1\alpha + \lambda_2\alpha^q + \lambda_3\alpha^{q^2} + \cdots + \lambda_n\alpha^{q^{n-1}}$ is a root of an irreducible polynomial of degree $n$ if and only if $\gamma, \gamma^q, \gamma^{q^2}, \ldots, \gamma^{q^{n-1}}$ are distinct. Now since

$$\gamma^{q^{n-k}} = \lambda_k\alpha + \lambda_{k+1}\alpha^q \cdots$$

$$+\lambda_n\alpha^{q^{n-k}} + \lambda_1\alpha^{q^{n-k+1}} + \cdots + \lambda_{k-1}\alpha^{q^{n-1}}, \quad (1)$$

$\gamma$ is a root of an irreducible polynomial of degree $n$ if and only if the following $n$ elements

$$(\lambda_1, \lambda_2, \lambda_3, \cdots, \lambda_n), (\lambda_2, \lambda_3, \lambda_4, \cdots, \lambda_n, \lambda_1)$$

$$, (\lambda_3, \lambda_4, \lambda_5, \cdots, \lambda_n, \lambda_1, \lambda_2), \cdots,$$

$$(\lambda_n, \lambda_1, \lambda_2, \cdots, \lambda_{n-1}) \quad (2)$$

are distinct.

When (2) happens then we call $\lambda = (\lambda_1, \lambda_2, \lambda_3, \cdots, \lambda_n)$ *aperiodic word*. We will write $\lambda$ as a word $\lambda = \lambda_1\lambda_2\lambda_3\cdots\lambda_n$ and define $\gamma(\lambda) := \lambda_1\alpha + \lambda_2\alpha^q + \lambda_3\alpha^{q^2} + \cdots + \lambda_n\alpha^{q^{n-1}}$. Therefore

**Lemma 1.** *We have*

1. *For any word $\lambda = \lambda_1\cdots\lambda_n \in \mathbb{F}_q^n$ the element $\gamma(\lambda)$ is a root of an irreducible polynomial of degree $n$ if and only if $\lambda$ is an aperiodic word.*
2. *Given an aperiodic word $\lambda$, the irreducible polynomial $g_{\gamma(\lambda)}$ can be constructed in time*[2] *$O((\log\log n)^2(\log n)^3 n^2) = \tilde{O}(n^2)$.*

Obviously, the aperiodic word $\lambda = \lambda_1\lambda_2\lambda_3\cdots\lambda_n$ and $R_k(\lambda) := \lambda_k\lambda_{k+1}\cdots\lambda_n\lambda_1\cdots\lambda_{k-1}$ corresponds to the same irreducible polynomial. See (1). That is, $g_{\gamma(\lambda)} = g_{\gamma(R_i(\lambda))}$ for any $1 \le i \le n$. Therefore to avoid enumerating the same polynomial more than once, the algorithm enumerates only the minimum element (in lexicographic order) among $\lambda, R_2(\lambda), \ldots, R_n(\lambda)$. Such an element is called *Lyndon word*. Therefore

**Definition 2.** *The word $\lambda = \lambda_1\lambda_2\lambda_3\cdots\lambda_n$ is called a Lyndon word if $\lambda < R_i(\lambda)$ for all $i = 2, \ldots, n$.*

To enumerate all the irreducible polynomials the algorithm enumerates all the Lyndon words of length $n$ and, for each one, it computes the corresponding irreducible polynomial.

In the next section, we show how to enumerate all the Lyndon words of length $n$ in linear delay time $O(n)$. Then from $\gamma(\lambda)$ (that corresponds to an irreducible polynomial) the algorithm constructs the irreducible polynomial $g_{\gamma(\lambda)}(x)$ and all the other $n-1$ roots in quadratic time $\tilde{O}(n^2)$. Since the size of all the roots is $O(n^2)$, this complexity is quasilinear in the output size. For the problem of enumerating only the roots (one root for each irreducible polynomial) the delay time is $O(n)$.

Let $L_{n,q}$ be the set of all Lyndon words over $\mathbb{F}_q$ of length $n$. We have shown how to reduce our problem to the problem of enumerating all the Lyndon words over $\mathbb{F}_q$ of length $n$ with linear delay time. Algorithm "Enumerate" in Figure 2 shows the reduction.

Putting all the above algebraic complexities together, we get the following

**Theorem 3.** *Let $\epsilon > 0$ be any constant. There is a randomized enumeration algorithm for*

[2]Here $\tilde{O}(N) = \tilde{O}(N \cdot poly(\log(N)))$

| $\sigma$ | $R_2(\sigma)$ | $R_3(\sigma)$ | $R_4(\sigma)$ | $R_5(\sigma)$ | $R_6(\sigma)$ | |
|---|---|---|---|---|---|---|
| 000000 | | | | | | |
| 000001 | 000010 | 000100 | 001000 | 010000 | 100000 | $x^6+x^5+1$ |
| 000011 | 000110 | 001100 | 011000 | 110000 | 100001 | $x^6+x+1$ |
| 000101 | 001010 | 010100 | 101000 | 010001 | 100010 | $x^6+x^3+1$ |
| 000111 | 001110 | 011100 | 111000 | 110001 | 100011 | $x^6+x^5+x^3+x^2+1$ |
| 001001 | 010010 | 100100 | | | | |
| 001011 | 010110 | 101100 | 011001 | 110010 | 100101 | $x^6+x^5+x^4+x+1$ |
| 001101 | 011010 | 110100 | 101001 | 010011 | 100110 | $x^6+x^5+x^4+x^2+1$ |
| 001111 | 011110 | 110100 | 111001 | 110011 | 100111 | $x^6+x^4+x^2+x+1$ |
| 010101 | 101010 | | | | | |
| 010111 | 101110 | 011101 | 111010 | 110101 | 101011 | $x^6+x^4+x^3+x+1$ |
| 011011 | 110110 | 101101 | | | | |
| 011111 | 111110 | 111101 | 111011 | 110101 | 101111 | $x^6+x^5+x^2+x+1$ |
| 111111 | | | | | | |

Figure 1: A table of the words over $\Sigma = \{0,1\}$ and all their rotations. The Lyndon words are in the gray boxes. The Lyndon words of length 6 are 000001, 000011, 000101, 000111, 001011, 001101, 001111, 010111 and 011111. The polynomial $f(x) = x^6+x+1$ is irreducible over $\mathbb{F}_2$ and therefore $\mathbb{F}_{2^6} = \mathbb{F}_2[\beta]/(\beta^6+\beta+1)$ and every element in $\mathbb{F}_{2^6}$ can be represented as $\lambda_5\beta^5+\cdots+\lambda_1\beta+\lambda_0$. For $\alpha = \beta^5+\beta^2+1$ the set $N(\alpha) = \{\alpha, \alpha^2, \alpha^4, \alpha^8, \alpha^{16}, \alpha^{32}\}$ is a Normal basis. The Lyndon word 001011 corresponds to the element $\gamma = \alpha^4+\alpha^{16}+\alpha^{32}$. The element $\gamma$ corresponds to the irreducible polynomial $g_\gamma(x) = (x-\gamma)(x-\gamma^2)(x-\gamma^4)(x-\gamma^8)(x-\gamma^{16})(x-\gamma^{32}) = x^6+x^5+x^4+x+1$.

**Enumerate**$(n,q)$
Preprocessing
1p) Find an irreducible polynomial $f(x)$ of degree $n$ over $\mathbb{F}_q$.
2p) Find a normal basis $\alpha, \alpha^q, \ldots, \alpha^{q^{n-1}}$ in $\mathbb{F}_q[\beta]/(f(\beta))$.
3p) Let $\lambda = 00\cdots01$
/* The first Lyndon word */
Delay
1d) Define
$\gamma = \lambda_1\alpha+\lambda_2\alpha^q+\cdots+\lambda_n\alpha^{q^{n-1}}$.
2d) Compute
$g_\gamma(x) := (x-\gamma)(x-\gamma^q)\cdots(x-\gamma^{q^{n-1}})$.
3d) Output$(g_\gamma(x), \gamma, \gamma^q, \cdots, \gamma^{q^{n-1}})$.
4d) Find the next Lyndon word: $\lambda \leftarrow \mathrm{Next}(\lambda)$.
5d) If $\lambda = 00\cdots01$ then Halt else Goto 1d.

Figure 2: An enumeration algorithm.

1. *the irreducible polynomial over $\mathbb{F}_q$ and their roots in $\mathbb{F}_{q^n}$ in preprocessing time $O((\log n)^4(\log\log n)^2n^2+(\log q)(\log n)^{1+\epsilon}n)$ and delay time $O((\log\log n)^2(\log n)^3n^2)$.*

2. *the roots in $\mathbb{F}_{q^n}$ of irreducible polynomials of degree $n$ over $\mathbb{F}_q$ in preprocessing time $O((\log n)^4(\log\log n)^2n^2+(\log q)(\log n)^{1+\epsilon}n)$ and delay time $O(n)$.*

**Theorem 4.** *Let $\epsilon > 0$ be any constant. There is a deterministic enumeration algorithm for*

1. *the irreducible polynomial over $\mathbb{F}_q$ and their roots in $\mathbb{F}_{q^n}$ in preprocessing time $O(n^{3+\epsilon}p^{1/2+\epsilon}+(\log q)^{2+\epsilon}n^{4+\epsilon}$ and delay time $O((\log\log n)^2\,(\log n)^3n^2)$.*

2. *the roots in $\mathbb{F}_{q^n}$ of irreducible polynomials of degree $n$ over $\mathbb{F}_q$ in preprocessing time $O(n^{3+\epsilon}p^{1/2+\epsilon}+(\log q)^{2+\epsilon}n^{4+\epsilon})$ and delay time $O(n)$.*

# 3 Linear Delay Time for Enumerating $L_{n,q}$

In this section we give Duval's algorithm, [11], that enumerates all the Lyndon words of length at most $n$, $\cup_{m\le n}L_{m,q}$, in linear delay time and change it to an algorithm that enumerates the Lyndon words of length $n$, $L_{n,q}$ in linear time. We will use a simple data structure that enable the algorithm to give the next Lyndon

word of length $n$ in Duval's algorithm in a constant update per symbol and therefore in linear time.

Let $\Sigma = \{0, 1, \ldots, q-1\}$ be the alphabet with the order $0 < 1 < \cdots < q-1$. We here identify $\mathbb{F}_q$ with $\Sigma$. We will sometime write the symbols in brackets. For example for $q = 5$ the word $[q-1]^2[q-3]$ is 442. Let $w = \sigma_1\sigma_2\cdots\sigma_m$ be a Lyndon word for some $m \leq n$. To find the next Lyndon word, (of length $\leq n$) Duval's algorithm first define the word $v = D(w) = w^h w'$ of length $n$ where $w$ is a non-empty prefix of $w$ and $h \geq 0$ (and therefore $h|w| + |w'| = n$). That is, $v = D(w) = \sigma_1\cdots\sigma_m\sigma_1\cdots\sigma_m\cdots\sigma_1\cdots\sigma_m\sigma_1\cdots\sigma_{(n \mod m)}$. Then if $v$ is of the form $v = ub[q-1]^t$ where $t \geq 0$ and $b \neq [q-1]$ then the next Lyndon word in Duval's algorithm is $P(v) = u[b+1]$. We denote the next Lyndom word of $w$ (in Duval's algorithm) by $N(w) := P(D(w))$. For example, for $q = 3$, $n = 7$ and $w = 0222$, $v = D(w) = 0222022$ and $N(w) = P(D(w)) = 02221$. Then $N(N(w)) = 022211$.

The following lemma is well known. We give the proof for completeness

**Lemma 5.** *If $w$ is a Lyndon word and $|w| < n$ then $|N(w)| > |w|$.*

*Proof.* Let $w = ub[q-1]^t$ where $b \neq [q-1]$. Then $u_1 \leq b$ because otherwise we would have $R_{|u|+1}(w) = b[q-1]^t u < ub[q-1]^t = w$ and then $w$ is not a Lyndon word. Let $D(w) = w^h w'$ where $h \geq 0$ and $w'$ is a nonempty prefix of $w$. Since $|D(w)| = n > |w|$ we have $h \geq 1$. Since $w'_1 = u_1 \leq b < q-1$, we have that $|N(w)| = |P(D(w))| \geq h|w| + 1 > |w|$. □

## 3.1 The Algorithm

In this subsection we give the data structure and the algorithm that finds the next Lyndon word of length $n$ in linear time.

We note here that, in the literature, the data structure that is used for the Lyndon word is an array of symbols. All the analyses of the algorithms in the literature treat an access to an element in an $n$ element array and comparing it to another symbol as an operation of time complexity equal to 1. The complexity of incrementing/decrementing an index $0 \leq i \leq n$ of an array of length $n$ and comparing two such indices are not included in the complexity. In this paper, the Lyndon words are represented with symbols and *numbers* in the range $[1, n]$. Every access to an element in this data structure and comparison between two elements are (as in literature) counted as an operation of time complexity equal 1. Operations that are done on the indices of the array (as in literature) are not counted but their time complexity is linear in the number of updates.

Let $v \in \Sigma^n$. We define the *compressed representation* of $v$ as $v = v^{(0)}[q-1]^{i_1}v^{(1)}[q-1]^{i_2}\cdots v^{(t-1)}[q-1]^{i_t}$ where $i_1, \ldots, i_{t-1}$ are not zero ($i_t$ may equal to zero) and $v^{(0)}, \ldots, v^{(t-1)}$ are nonempty words that do not contain the symbol $[q-1]$. If $v$ do not contain the symbol $[q-1]$ then $v = v^{(0)}[q-1]^0$ where $[q-1]^0$ is the empty word and $v^{(0)} = v$. The data structure will be an array (or double link list) that contains $v^{(0)}, i_1, v^{(1)}, \cdots, v^{(t-1)}, i_t$ if $i_t \neq 0$ and $v^{(0)}, i_1, v^{(1)}, \cdots, v^{(t-1)}$ otherwise.

Define $\|v\| = \sum_{j=0}^{t-1} |v^{(j)}| + t$. This is the *compressed length* of the compressed representation of $v$. Notice that for a word $v = v_1\cdots v_r$ that ends with a symbol $v_r \neq [q-1]$ we have $P(v) = v_1\cdots v_{r-1}[v_r+1]$ and for $u = v\cdot[q-1]^i$ we have $P(u) = P(v)$. Therefore $\|v\| - 1 \leq \|P(v)\| \leq \|v\|$.

Let $v = v^{(0)}[q-1]^{i_1}v^{(1)}[q-1]^{i_2}\cdots v^{(t-1)}[q-1]^{i_t}$ be any Lyndon word of length $n$. The next Lyndon word in Duval's algorithm is

$$u^{(1)} := N(v) =$$

$$v^{(0)}[q-1]^{i_1}v^{(1)}[q-1]^{i_2}\cdots[q-1]^{i_{t-1}}\cdot P(v^{(t-1)})$$

To find the next Lyndon word $u^{(2)}$ after $u^{(1)}$ we take $\left(u^{(1)}\right)^h z^{(1)}$ of length $n$ where $z^{(1)}$ is a nonempty prefix of $u^{(1)}$ and then $u^{(2)} = \left(u^{(1)}\right)^h \cdot P(z^{(1)})$. This is because $z^{(1)}_1 = u^{(1)} \neq [q-1]$. Since by Lemma 5, $|u^{(1)}| < |u^{(2)}| < \cdots$ we will eventually get a Lyndon word of length $n$. We now show that using the compressed representation we have

**Lemma 6.** *The time complexity of computing $u^{(i+1)}$ from $u^{(i)}$ is at most $|u^{(i+1)}| - |u^{(i)}| + 1$.*

*Proof.* Let $u^{(i)} = w^{(0)}[q-1]^{i_1}w^{(1)}[q-1]^{i_2}\cdots w^{(t-1)}[q-1]^{i_t}$ of length less than $n$. Then $u^{(i+1)} = (u^{(i)})^h \cdot P(z^{(i)})$ where $z^{(i)}$ is a nonempty prefix of $u^{(i)}$. So it is enough to show that $P(z^{(i)})$ can be computed in at most $|P(z^{(i)})| + 1$ time. Notice that the length of $z^{(i)}$ is $(n \mod |u^{(i)}|)$ (here the mod is equal to $|u^{(i)}|$ if $|u^{(i)}|$ divides $n$). Since $z^{(i)}$ is a prefix of $u^{(i)}$ we have that, in the compressed representation, $z^{(i)} = w^{(0)}[q-1]^{i_1}w^{(1)}[q-1]^{i_2}\cdots w^{(t'-1)}[q-1]^{i_{t'}}$ for some $t' \leq t$. Then $P(z^{(i)}) = w^{(0)}[q-1]^{i_1}w^{(1)}[q-1]^{i_2}\cdots P(w^{(t'-1)})$. Therefore the complexity of computing $P(z^{(i)})$ is $\|z^{(i)}\| \leq |P(z^{(i)})| + 1$. □

From the above lemma it follows that

**Theorem 7.** *Let $v$ be a Lyndon word of length $n$. Using the compressed representation, the next Lyndon word of length $n$ can be computed in linear time.*

*Proof.* To compress $v$ and find $u^{(1)} = N(v)$ we need a linear time. By Lemma 5 the Lyndon words after $v$ are $u^{(1)}, \ldots, u^{(j)}$ where $|u^{(1)}| < |u^{(2)}| < \cdots < |u^{(j)}| = n$. By Lemma 6 the time complexity of computing the next Lyndon word $u^{(j)}$ of length $n$ is $\sum_{i=1}^{j-1} |u^{(i+1)}| - |u^{(i)}| + 1 \le |u^{(j)}| + n = O(n)$. Then decompressing the result takes linear time. □

We now give a case where Duval's algorithm fails to give the next Lyndon word of length $n$ in linear time. Consider the Lyndon word $01^k01^{k+1}$ of length $n = 2k + 3$. The next Lyndon word in Duval's algorithm is $01^{k+1}$. Then $01^{k+2}, 01^{k+3}, \ldots, 01^{2k+2}$. To get to the next Lyndon word of length $n$, $01^{2k+2}$, the algorithm does $\sum_{i=1}^{k+2} i = O(n^2)$ updates.

# 4 Constant Amortized Time for Enumerating $L_{n,q}$

In this section, we show that our algorithm in the previous section is CAT algorithm. That is, it has a constant amortized update cost.

We first give some notation and preliminary results. Let $\ell_n$ be the number of Lyndon words of length $n$, $L_i = \ell_1 + \cdots + \ell_i$ for all $i = 1, \ldots, n$ and $\Lambda_n = L_1 + \cdots + L_n = n\ell_1 + (n-1)\ell_2 + \cdots + \ell_n$. It is known from [11] that for $n \ge 11$ and any $q$

$$\frac{q^n}{n}\left(1 - \frac{q}{(q-1)q^{n/2}}\right) \le \ell_n \le \frac{q^n}{n} \tag{3}$$

and for any $n$ and $q$

$$L_n \ge \frac{q}{q-1}\frac{q^n}{n} \tag{4}$$

and

$$\Lambda_n = \frac{q^2}{(q-1)^2}\frac{q^n}{n} \times \left(1 + \frac{2}{(q-1)(n-1)} + O\left(\frac{1}{(qn)^2}\right)\right). \tag{5}$$

Denote by $\ell_{n,i}$ the number of Lyndon words of length $n$ of the form $w = ub[q-1]^i$ where $b \in \Sigma\backslash\{q-1\}$. Then $\ell_n = \ell_{n,0} + \ell_{n,1} + \cdots + \ell_{n,n-1}$. Let $\ell_n^*$ be the number of Lyndon words of length $n$ that ends with the symbol $[q-2]$. That is, of the form $u[q-2]$.

For the analysis we will use the following.

**Lemma 8.** *Let $w = ub[q-1]^t \in \Sigma^n$ where $b \in \Sigma\backslash\{q-1\}$ and $t \ge 1$. If $w = ub[q-1]^t$ is a Lyndon word of length $n$ then $u[b+1]$ is a Lyndon word.*

*In particular,*

$$\ell_{n,t} \le \ell_{n-t}.$$

*If $w = u[q-2]$ is a Lyndon word of length $n$ then $u[q-1]$ is a Lyndon word. In particular,*

$$\ell_n^* \le \ell_{n,1} + \cdots + \ell_{n,n-1}.$$

*Proof.* If $w = ub[q-1]^t$ is Lyndon word of length $n$ then the next Lyndon word in Duvel's algorithm is $P(D(ub[q-1]^t)) = P(ub[q-1]^t) = u[b+1]$.

If $w = u[q-2]$ is a Lyndon word of length $n$ then $P(D(w)) = u[q-1]$ is the next Lyndon word in Duvel's algorithm. □

The amortized number of updates of listing all the Lyndon words of length at most $n$ in Duval's algorithm is [11]

$$\gamma_n \le \frac{2\Lambda_n}{L_n} - 1 = 1 + \frac{2}{q-1} + O\left(\frac{1}{qn}\right)$$

We now show that

**Theorem 9.** *Using the compressed representation the amortized number of updates for enumerating all the Lyndon words of length exactly $n$ is at most*

$$\frac{3(\Lambda_n - L_n) + \ell_n}{\ell_n} = 1 + \frac{3q}{(q-1)^2} + o(1)$$

*Proof.* The number of Lyndon words of length $n$ of the forms $w = ub$ where $b \in \Sigma$, $b \ne [q-1]$ and $b \ne [q-2]$ is $\ell_n - (\ell_{n,1} + \cdots + \ell_{n,n-1}) - \ell_n^*$. The next word of length $n$ is $u[b+1]$. So each such word takes one update to find the next word. For words that end with the symbol $[q-2]$ we need to change this symbol to $[q-1]$ and plausibly merge it with the previous one in the compressed representation. This takes at most two updates. One for removing this symbol and one for merging it with the cells of the form $[q-1]^t$. Therefore for such words we need $2\ell_n^*$ updates. Thus, for Lyndon words that do not ends with $[q-1]$ we need $\ell_n - (\ell_{n,1} + \cdots + \ell_{n,n-1}) + \ell_n^*$ updates.

For strings of the form $w = ub[q-1]^t$ where $b \ne [q-1]$ and $t \ge 1$ we need at most $3t$ updates and therefore at most $3t\ell_{n,t}$ for all such words. See the proof of Theorem 7. Therefore, the total updates is at most

$$\ell_n - (\ell_{n,1} + \cdots + \ell_{n,n-1}) + \ell_n^* +$$

$$3(\ell_{n,1} + 2\ell_{n,2} + \cdots + (n-1)\ell_{n,n-1})$$

By Lemma 4, this is at most

$$\ell_n + 3(\ell_{n-1} + 2\ell_{n-2} \cdots + (n-1)\ell_1).$$

Now, the amortized update is

$$\frac{\ell_n + 3(\ell_{n-1} + 2\ell_{n-2} \cdots + (n-1)\ell_1)}{\ell_n} =$$

$$\frac{3(\Lambda_n - L_n) + \ell_n}{\ell_n} = 1 + 3\frac{\Lambda_n - L_n}{\ell_n}.$$

By (3), (4) and (5) we get

$$1 + 3\frac{\Lambda_n - L_n}{\ell_n}$$

$$\leq 1 + 3\frac{\frac{q^2}{(q-1)^2}\left(1 + \frac{2}{(q-1)(n-1)} + O\left(\frac{1}{(qn)^2}\right)\right) - \frac{q}{q-1}}{1 - \frac{q}{(q-1)q^{n/2}}}$$

$$= 1 + 3\frac{\frac{q}{(q-1)^2} + \frac{q^2}{(q-1)^2}\left(\frac{2}{(q-1)(n-1)} + O\left(\frac{1}{(qn)^2}\right)\right)}{1 - \frac{q}{(q-1)q^{n/2}}}$$

$$= 1 + \frac{3q}{(q-1)^2} + O\left(\frac{1}{qn}\right).$$

□

# 5 Membership in $L_{n,q}$

In this subsection, we study the complexity of deciding membership in $L_{n,q}$. That is, given a word $\sigma \in \mathbb{F}_q^n$. Decide whether $\sigma$ is in $L_{n,q}$.

Since $\sigma \in L_{n,q}$ if and only if for all $1 < i \leq n$, $R_i(\sigma) > \sigma$, and each comparison of two words of length $n$ takes $O(n)$ operations, membership can be decided in time $O(n^2)$. Duval in [10] gave a linear time algorithm. In this subsection, we give a simple algorithm that decides membership in linear time. To this end, we need to introduce the suffix tree data structure.

The suffix tree of a word $s$ is a trie that contains all the suffixes of $s$. See for example the suffix tree of the word $s = 1010110\$$ in Figure 3. A suffix tree of a word $s$ of length $n$ can be constructed in linear time in $n$ [48, 15]. Using the suffix tree, one can check if a word $s'$ of length $|s'| = m$ is a suffix of $s$ in time $O(m)$.

Denote by $ST(s)$ the suffix tree of $s$. Define any order $<$ on the symbols of $s$. Define $\mathrm{Min}(ST(s))$ as follows: Start from the root of the trie and follow, at each node, the edges with the minimal symbol. Then $\mathrm{Min}(ST(s))$ is the word that corresponds to this path. One can find this word in $ST(s)$ in time that is linear in its length.

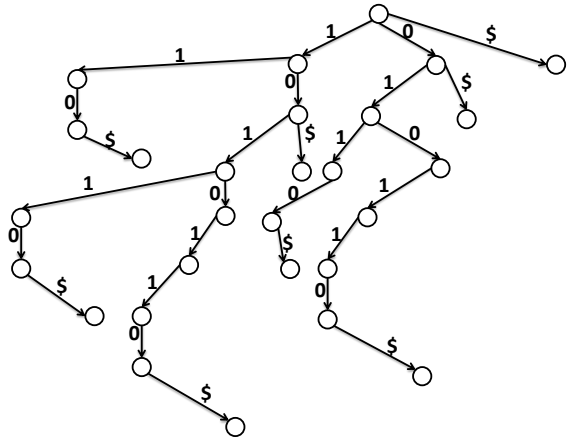


Figure 3: The suffix tree of $s = 1010110\$$. If $1 < 0 < \$$ then $\mathrm{Min}(ST(s)) = 110\$$. If $0 < 1 < \$$ then $\mathrm{Min}(ST(s)) = 010110\$$.

The function $\mathrm{Min}$ defines the following total order $\prec$ on the suffixes: Let $T = ST(s)$. Take $\mathrm{Min}(T)$ as the minimum element in that order. Now remove this word from $T$ and take $\mathrm{Min}(T)$ as the next one in that order. Repeat the above until the tree is empty. For example, if $0 < 1 < \$$ then the order in the suffix tree in Figure 3 is

$$010110\$, 0110\$, 0\$, 1010110\$, 10110\$, 10\$, 110\$, \$.$$

Obviously, for two suffixes $s$ and $r$, $s \prec r$ if and only if for $j = \min(|r|, |s|)$ we have $s_1 \cdots s_j < r_1 \cdots r_j$ (in the lexicographic order).

We define $ST_m(s)$ the suffix tree of the suffixes of $s$ of length at least $m$. We can construct $ST_m(s)$ in linear time in $|s|$ by taking a walk in the suffix tree $ST(s)$ and remove all the words of length less than $m$. In the same way as above, we define $\mathrm{Min}(ST_m(s))$.

We now show

**Lemma 10.** *Let $\$ \notin \mathbb{F}_q$ be a symbol. Define any total order $<$ on $\Sigma = \mathbb{F}_q \cup \{\$\}$ such that $\$ < \alpha$ for all $\alpha \in \mathbb{F}_q$. Let $\sigma \in \mathbb{F}_q^n$. Then $\sigma \in L_{n,q}$ if and only if*

$$\mathrm{Min}(ST_{n+2}(\sigma\sigma\$)) = \sigma\sigma\$.$$

*Proof.* First, notice that every word in $ST_{n+2}(\sigma\sigma\$)$ is of the form $\sigma_i \cdots \sigma_n \sigma\$$ for some $i = 1, \ldots, n$. Let $T = ST_{n+2}(\sigma\sigma\$)$.

If $R_i(\sigma) < \sigma$ then $\sigma_i \cdots \sigma_n \sigma_1 \cdots \sigma_{i-1} < \sigma$, and therefore $\sigma_i \cdots \sigma_n \sigma\$ = \sigma_i \cdots \sigma_n \sigma_1 \cdots \sigma_{i-1} \sigma_i \cdots \sigma_n\$ \prec \sigma\sigma\$$. Thus, $\mathrm{Min}(T) \neq \sigma\sigma\$$.

If $R_i(\sigma) = \sigma$ then $\sigma_i \cdots \sigma_n \sigma_1 \cdots \sigma_{i-1} = \sigma$, and then

$$\sigma_i \cdots \sigma_n \sigma =$$

$$\sigma_i \cdots \sigma_n \sigma_1 \cdots \sigma_{i-1} \sigma_i \cdots \sigma_n = \sigma \sigma_1 \cdots \sigma_{n-i+1}.$$

Thus, $\sigma_i \cdots \sigma_n \sigma\$ < \sigma \sigma_1 \cdots \sigma_{n-i+2}$ which implies $\sigma_i \cdots \sigma_n \sigma\$ \prec \sigma\sigma\$$. Therefore, we have $\mathrm{Min}(T) \neq \sigma\sigma\$$.

If $R_i(\sigma) > \sigma$ then $\sigma_i \cdots \sigma_n \sigma_1 \cdots \sigma_{i-1} > \sigma$, and therefore $\sigma_i \cdots \sigma_n \sigma\$ \succ \sigma\sigma\$$ and then $\mathrm{Min}(T) \neq \sigma_i \cdots \sigma_n \sigma\$$.

Now, if $\sigma \in L_{n,q}$ then $R_i(\sigma) > \sigma$ for all $1 < i \leq n$. Thus $\mathrm{Min}(T) \neq \sigma_i \cdots \sigma_n \sigma\$$ for all $i$. Therefore we have $\mathrm{Min}(T) = \sigma\sigma\$$. If $\sigma \notin L_{n,q}$ then there is $i$ such that $R_i(\sigma) \leq \sigma$, and then $\mathrm{Min}(T) \neq \sigma\sigma\$$. □

**Membership**$(\sigma, n, q)$
1) Define a total order on $\mathbb{F}_q \cup \{\$\}$ such that \$ is the minimal element.
2) $T \leftarrow$Construct the Suffix Tree of $\sigma\sigma\$$.
3) Take a walk in $T$ and remove all the words of length less than $n+2$.
4) Define $r$ the word of the path that starts from the root and takes, at each node, the edge with the smallest symbol.
5) If $r = \sigma\sigma\$$ then $\sigma \in L_{n,q}$ else $\sigma \notin L_{n,q}$.

Figure 4: Membership of $\sigma$ in $L_{n,q}$.

We now prove

**Theorem 11.** *There is a linear time algorithm that decides whether a word $\sigma$ is in $L_{n,q}$.*

*Proof.* The algorithm is in Figure 4. We use Lemma 10. The algorithm constructs the trie $ST_{n+2}(\sigma\sigma\$)$. The construction takes linear time in $\sigma\sigma\$$ and therefore linear time in $n$. Finding $\mathrm{Min}(ST_{n+2}(\sigma\sigma\$))$ in a trie takes linear time. □

*References:*

[1] F. Ashari-Ghomi, N. Khorasani, A. Nowzari-Dalini. Ranking and Unranking Algorithms for k-ary Trees in Gray Code Order. International Scholarly and Scientific Research & Innovation. 6(8). pp. 833–838. (2012)

[2] M. Ackerman, E. Mäkinen: Three New Algorithms for Regular Language Enumeration. COCOON 2009. pp. 178–191. (2009).

[3] M. Ackerman, J. Shallit. Efficient enumeration of words in regular languages. *Theor. Comput. Sci.* 410(37) pp. 3461–3470. (2009)

[4] N. H. Bshouty. Dense Testers: Almost Linear Time and Locally Explicit Constructions. Electronic Colloquium on Computational Complexity (ECCC). 22: 6 (2015).

[5] J. Berstel, M. Pocchiola. Average cost of Duval's algorithm for generating Lynon words. *Theoretical Computer Science*. 132(1). pp. 415-425. (1994).

[6] G. E. Collins. The Calculation of Multivariate Polynomial Resultants. *J. ACM*. 18(4): pp. 515–532. (1971).

[7] K. Cattell, F. Ruskey, J. Sawada, M. Serra, C. R. Miers. Fast Algorithms to Generate Necklaces, Unlabeled Necklaces, and Irreducible Polynomials over GF(2). *J. Algorithms.* 37(2). pp. 267-282. (2000).

[8] N. J. Calkin, H. S. Wilf. Recounting the Rationals. *The American Mathematical Monthly*. 107(4). pp. 360–363. (2000).

[9] P. Dömösi. Unusual Algorithms for Lexicographical Enumeration. *Acta Cybern.* 14(3). pp. 461–468. (2000)

[10] J. P. Duval. Factorizing words over an ordered alphabet. *Jornal of Algorithms*. 4(4). pp. 363-381. (1983).

[11] J.-P. Duval. Génération d'une Section des Classes de Conjugaison et Arbre des Mots de Lyndon de Longueur Bornée. *Theor. Comput. Sci.* 60, pp. 255–283. (1988).

[12] C. Ding, D. Pei, A. Salomaa. Chinese Remainder Theorem. Application in Computing, Coding, Cryptography. World Scientific Publication. (1996).

[13] M. C. Er. Enumerating Ordered Trees Lexicographically. *Comput. J.* 28(5). pp. 538–542. (1985).

[14] K. M. Elbassioni, K. Makino, I. Rauf. Output-Sensitive Algorithms for Enumerating Minimal Transversals for Some Geometric Hypergraphs. ESA 2009. pp. 143–154. (2009).

[15] M. Farach. Optimal Suffix Tree Construction with Large Alphabets. 38th IEEE Symposium on Foundations of Computer Science (FOCS '97), pp. 137–143. (1997)

[16] H. Fredricksen, I. J. Kessler. An algorithm for generating necklaces of beads in two colors. Discrete Mathematics 61(2-3): 181-188 (1986)

[17] H. Fredricksen, J. Maiorana. Necklaces of beads in $k$ color and $k$-aray de Bruijn sequences. *Discrete Mathematics*, 23(3). pp. 207–210. (1978).

[18] K. Fukuda and Y. Matsui. Enumeration of Enumeration Algorithms and Its Complexity. http://www-ikn.ist.hokudai.ac.jp/∼wasa/enumeration_complexity.html.

[19] D. T. Huynh. The Complexity of Ranking Simple Languages. *Mathematical Systems Theory.* 23(1). pp. 1–19. (1990)

[20] A. V. Goldberg, M. Sipser. Compression and Ranking. STOC 1985. pp. 440–448. (1985).

[21] U. Gupta, D. T. Lee, C. K. Wong. Ranking and Unranking of 2-3 Trees. *SIAM J. Comput.* 11(3). pp. 582–590. (1982).

[22] J. von zur Gathen, V. Shoup. Computing Frobenius Maps and Factoring Polynomials. *Computational Complexity*, 2. pp. 187-224. (1992).

[23] L. A. Hemachandra. On ranking. Structure in Complexity Theory Conference. (1987).

[24] L. A. Hemachandra, S. Rudich. On the Complexity of Ranking. *J. Comput. Syst. Sci.* 41(2). pp. 251–271. (1990)

[25] S. Kapoor, H. Ramesh. An Algorithm for Enumerating All Spanning Trees of a Directed Graph. *Algorithmica*. 27(2). pp. 120–130. (2000).

[26] T. Kociumaka, J. Radoszewski, W. Rytter. Efficient Ranking of Lyndon Words and Decoding Lexicographically Minimal de Bruijn Sequence. CoRR abs/1510.02637 (2015)

[27] H. W. Lenstra. Finding isomorphisms between finite fields. Mathematics of Computation, 56, 193, pp. 329–347. (1991).

[28] R. Lidl and H. Niederreiter. Finite Fields. Encyclopedia of Mathematics and its Applications. Addison-Wesley Publishing Company. (1984).

[29] A. Poli. A deterministic construction of normal bases with complexity $O(n^3+nlognloglognlogq)$. *J. Symb. Comp.*, 19, pp. 305–319. (1995).

[30] L. Li. Ranking and Unranking of AVL-Trees. *SIAM J. Comput.* 15(4). pp. 1025-1035. (1986).

[31] A. Lempel, G. Seroussi, S. Winograd: On the Complexity of Multiplication in Finite Fields. *Theor. Comput. Sci.* 22: pp. 285–296. (1983).

[32] E. Mäkinen. Ranking and Unranking Left Szilard Languages. University of Tampere. Report A-1997-2.

[33] K. Makino, T. Uno. New Algorithms for Enumerating All Maximal Cliques. SWAT 2004. pp. 260–272. (2004).

[34] E. Mäkinen. On Lexicographic Enumeration of Regular and Context-Free Languages. *Acta Cybern.* 13(1). pp. 55–61. (1997)

[35] W. J. Myrvold, F. Ruskey. Ranking and unranking permutations in linear time. *Inf. Process. Lett.* 79(6). pp. 281–284. (2001).

[36] J. M. Pallo. Enumerating, Ranking and Unranking Binary Trees. *Comput. J.* 29(2). pp. 171–175. (1986).

[37] F. Ruskey, C. D. Savage, T. M. Y. Wang. Generating Necklaces. *J. Algorithms*. 13(3). pp. 414–430. (1992).

[38] J. B. Remmel, S. G. Williamson. Ranking and Unranking Trees with a Given Number or a Given Set of Leaves. arXiv:1009.2060.

[39] Y. Shen. A new simple algorithm for enumerating all minimal paths and cuts of a graph. *Microelectronics Reliability*. 35(6). pp. 973–976. (1995).

[40] I. Shparlinski. Finite fields: theory and computation. Mathematics and Its Applications, Vol. 477. (1999).

[41] T. J. Savitsky. Enumeration of 2-Polymatroids on up to Seven Elements. *SIAM J. Discrete Math.* 28(4). pp. 1641–1650. (2014)

[42] P. Tarau. Ranking and Unranking of Hereditarily Finite Functions and Permutations. arXiv:0808.0554. (2008).

[43] T. Uno. An Algorithm for Enumerating all Directed Spanning Trees in a Directed Graph. ISAAC 1996. pp. 166–173. (1996).

[44] T. Uno. Algorithms for Enumerating All Perfect, Maximum and Maximal Matchings in Bipartite Graphs. ISAAC 1997. pp. 92–101. (1997).

[45] T. Uno. A Fast Algorithm for Enumerating Bipartite Perfect Matchings. ISAAC 2001. pp. 367–379. (2001)

[46] R-Y. Wu, J-M. Chang. Ranking and unranking of well-formed parenthesis strings in diverse representations. ICCRD - International Conference on Computer Research and Development. (2011).

[47] V. V. Vazirani, M. Yannakakis. Suboptimal Cuts: Their Enumeration, Weight and Number (Extended Abstract). ICALP 1992. pp. 366-377. (1992).

[48] P. Weiner., Linear pattern matching algorithms. 14th Annual IEEE Symposium on Switching and Automata Theory. pp. 1–11. (1973).

[49] R-Y. Wu, J-M. Chang, A-H. Chen, C-L. Liu. Ranking and Unranking $t$-ary Trees in a Gray-Code Order. *Comput. J.* 56(11). pp. 1388–1395. (2013).

[50] R-Y. Wu, J-M. Chang, A-H. Chen, M-T. Ko. Ranking and Unranking of Non-regular Trees in Gray-Code Order. *IEICE Transactions*. 96-A(6), pp. 1059–1065. (2013).

[51] R-Y. Wu, J-M. Chang, Y-L. Wang. Ranking and Unranking of $t$-Ary Trees Using RD-Sequences. *IEICE Transactions*. 94-D(2). pp. 226–232. (2011).

[52] R-Y. Wu, J-M. Chang, C-H. Chang. Ranking and unranking of non-regular trees with a prescribed branching sequence. Mathematical and Computer Modelling. 53(5-6). pp. 1331–1335. (2011).

[53] Li-Pu Yeh, B-F Wang, H-H Su. Efficient Algorithms for the Problems of Enumerating Cuts by Non-decreasing Weights. *Algorithmica*. 56(3). pp. 297-312. (2010).