HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

GRADUATION THESIS

Run length limited de Bruijn sequences for quantum communication

NGUYEN TIEN LONG
long.nt180129@sis.hust.edu.vn

Major: Computer Science

Supervisor: Dr. Vu Van Khu

Dr. Tran Vinh Duc ______________

Signature

Department: Computer Science

School: School of Information and Communications Technology

HANOI, 08/2022

ACKNOWLEDGMENT

First and foremost, I would like to express my sincere gratitude to the subject teachers at Hanoi University of Science and Technology for their continuous guidance throughout my study, especially ones from School of Information and Communication Technology. During the past three and a half years, I have learned from them needed knowledge for pursuing later stages in my Computer Science journey.

I thankfully acknowledge the support of Prof. Huynh Thi Thanh Binh, who accepted me into MSO Lab and guided me in the first steps of my research career. My appreciation also extends to my laboratory colleagues, I'm very happy to have had the opportunity to collaborate with all of you, especially M.Sc. Tran Ba Trung, who introduced and invited me to joint work with him on our very first theoretical topic. I would also like to send my sincerity to other members of MSO Lab, Dr. Nguyen Thi My Binh, M.Sc. Nguyen Hong Ngoc, Tran Cong Dao, Tran Huy Hung and Nguyen Dac Tam. Dr.Binh and M.Sc Ngoc supported me a lot when I first joint our lab. Brothers Dao, Hung, and Tam lent a hand to review my thesis.

I also cherish the moment of doing my thesis under the supervision of Dr. Tran Vinh Duc. Though time is short, but his style in research inspired me so much.

Taking my first step in doing theoretical research, I owe a great debt of gratitude to Dr. Ta Duy Hoang (ENS de Lyon) and Dr. Vu Van Khu (National University of Singapore) for training and mentoring me during the working on various problems including ones in this thesis. Without their support (both professionally and personally), it would not have been possible for this work to progress as far as it has.

I'm also grateful to FPT Young Talent's members, especially Do Hoang Khanh, Lai Ngoc Tan, and Thanh La. I'll always remember the late night we spent together sharing our stories and our plans. I had known the way to find my answer to "Who am I going to be? How do I want to live in the next 10 years? 20 years? . . . ?" since that night.

In Bach Khoa Street Workout, my gratitude goes to Xuan Nam, Duc Anh, and Thanh Hai. We've trained and shared our experiences in SW, life and work as brothers.

My student's life must be harder without the following idiots: Phan Viet Hoang, Phi Phuc , Khanh Doan, Ng Duc Long, Tran Anh, Ba Tan, and Hong Sang. They

are my best friends for the last four years.

I’m also very grateful to my aunt Hang and uncle Lich, who always look after me just like my mom and dad.

Last but not least, this thesis is dedicated to my parents for the two decades of your love and support. I would not have come this far without your loving weight behind me.

ABSTRACT

Quantum key distribution (Quantum key distribution) is a secure communication enabling two parties to produce a shared random secret key known only to them. Current commercial deployed QKD systems have transmission range restricted to under 1000 km because they rely on optical fiber. The alternative method, satellite QKD, is able to overcome this issue but faces a new challenge caused by noisy environments and swift relative motion between the transmitter and receiver.

Therefore, a classical channel, which actually is a timing and synchronization system, is used along with the quantum channel. In such systems, Peide Zhang .et.al proposed to transmit a positioning sequence (also known as a de Bruijn sequence). To consider the timing jitter performance, a long period of no-pulses should be forbidden. In Peide Zhang's method, two pulse slots are used to represent a single bit (on-on is 1 and on-off is 0) so that one can avoid two consecutive no-pulses. However, the above scheme, called Hybrid de Bruijn (Hybrid de Bruijn) code, requires $2n$ pulse slots to represent a de Bruijn sequence of length n and it needs to receive a sub-sequence of $2\log n$ pulse slots to locate its position.

Observe that it is possible to use less redundant pulse slots to achieve both goals: to synchronize accurately and to avoid a long period of no-pulses, in this thesis, Run length limited de Bruijn sequences are designed in which each binary bit is represented by only one pulse slot, 1 is on and 0 is off. The RdB sequences are shown to be more general and efficient than the previous work.

This thesis provides the first explicit formula for the maximal length of the run length limited de Bruijn sequences. Furthermore, using Lyndon words, an efficient construction of a run length limited de Bruijn sequence with the maximal length is presented. In addition, a sub-linear decoding algorithm that can locate the position of an arbitrary substring is also provided.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Definition
dBTS	a timing and synchronization system using Hybrid de Bruijn code (de Bruijn based Timing and Synchronization system)
FKM	Algorithm of Fredricksen, Kessler and Maiorana to generate granddaddy sequence
HdB	de Bruijn sequences encoded with a beacon model, on-on is 1 and on-off is 0 (Hybrid de Bruijn sequence)
LFSR	an algorithm to generate a de Bruijn sequence using a linear function (Linear feedback shift register)
LHS	Left hand side of a specific equation (Left Hand Side)
QKD	a secure communication method involving components of quantum mechanics for exchanging encryption keys (Quantum key distribution)
RdB	a sequence that is the combination of positioning sequence and run length sequence (Run length limited de Bruijn sequence)
RHS	Right hand side of a specific equation (Right Hand Side)
RLL	a line coding technique that is used to send arbitrary data over a communications channel with bandwidth limits (Run length limited)

Notation Table

Notation	Meaning
Σ_q	alphabet of size q, $\Sigma_q = (0, 1, 2, \ldots, q-1)$
$\boldsymbol{x} \in \Sigma^n$	binary sequence $\boldsymbol{x}$ of length n over alphabet Σ
0^i (1^i)	concatenation of i symbols 0 (1)
$W(n, s)$	set of all sequences of length n containing at most s consecutive bits 0
Ł$_n$	set of all Lyndon words of length n
Ł$^{(n)}$	set of all Lyndon words whose length are divisors of n
$\langle \boldsymbol{x} \rangle$	minimal rotation of sequence $\boldsymbol{x}$
G_k	de Bruijn graph of order k
$G_{k,s} = (V^{k-1,s}, E^{k,s})$	(k, s)-RdB graph with vertex set $V^{k-1,s}$, edges set $E^{k,s}$
$V_{i,j}^{k-1,s}$	set of all vertices in $G_{k,s}$ satisfying the first $i+1$ letters are $(0, 0, \ldots, 0, 1)$ and the last $j+1$ letters are $(1, 0, 0, \ldots, 0)$
$\ell(G_{k,s})$	length of the longest simple path in $G_{k,s}$
$N(k, s)$	length of the longest (k, s)-RdB sequence
$\mathbb{U}(k, s)$	upper bound for the length of the longest simple path in $G_{k,s}$
$\mathcal{U}(k, s)$	upper bound for the length of the longest (k, s)-RdB sequence

CHAPTER 1. INTRODUCTION

A de Bruijn sequence, or a positioning sequence, (of order k) is a binary sequence in which every possible length-k string appears exactly once as a substring. The uses of de Bruijn sequences have been found in various fields. Recently, a novel application of positioning sequences has been found in quantum communication by Zhang, Oi, Lowndes, *et al.* They adopt such sequences into HdB sequences to develop a system for synchronizing in quantum channels. Though having shown advantages against other methods [1], such implementation still has drawbacks and there is room for improvements. This thesis focuses on designing a new constrained de Bruijn sequence and proving its efficiency against HdB sequence.

Chapter 1 first briefly introduces the QKD protocol, along with the reasonable motivation to study the synchronization mechanisms in satellite quantum channels. Among such mechanisms, the timing and synchronization system proposed by Zhang, Oi, Lowndes, *et al.* in [1] is analyzed to recognize its disadvantages. The main results of this thesis, which aim to improve those weaknesses, are summarized. The organization of the whole thesis is given at the end of this chapter.

1.1 Timing and synchronization system in quantum channels

Symmetric, public-key (asymmetric), and hash-based cryptography are fundamental pillars of modern cryptography. While symmetric schemes and hash functions are less vulnerable to quantum attacks, the asymmetric schemes based on factoring or solving the discrete logarithm problem, for example, Rivest-Shamir-Adelman (RSA), Elliptic Curve Cryptography, are completely broken by a quantum adversary via Shor's algorithm [2]. Currently deployed public key cryptosystems are used to establish a common secret key between two parties. Doing the same jobs, QKD enables two parties to produce a shared random secret key known only to them. Moreover, QKD can guarantee the security of communication links making them immune to quantum computer-based attacks [3].

The first invented QKD protocols are BB84 [4], and E91 [5]. Since 2005, QKD has been initially implemented in real life. For example, in 2005, the University of Geneva and Corning Inc used a fiber optic wire of 307 km. In 2007, Los Alamos National Laboratory and the National Institute of Standards and Technology (NIST) used the BB84 protocol over a 148.7 km optical fiber. In 2018, Quantum Xchange launched the first quantum network in the U.S., offering $1,000$ km of fiber optic cable and 19 colocation centers along the Boston-to-Washington, D.C., corridor and metro hubs. However, due to the intrinsic exponential losses over optical fiber, the deployed QKD systems' range is restricted to under 1000 km. So as to establish intercontinental secure communication links, which usually require a range over 1000 km, satellite QKD has been proposed as an alternative, with the pioneering Micius satellite. In these systems, the transmitter (satellite) and the receiver (optical ground station) relentlessly exchange information after measuring the quantum states. But transmitting faint quantum optical pulses between a satellite and the Earth is challenging due to high channel losses cause by volatility environments and rapid relative motion between two parties.

To deal with this problem, reliable and efficient timing and synchronization systems have been proposed in [6], [7]. In this system, a classical channel is used along with the quantum channel, because of its advantage in synchronization.

Based on that concept, in [1], a de Bruijn based timing and synchronization system is introduced using a beacon with an on-off model. In this model, a de Bruijn sequence is transmitted from the satellite to the ground for synchronization. The superiority of this system relies on the intrinsic property of the positioning sequence, which is self-located. However, in dBTS, the method [1] use requires 2 pulse slots to modulate 1 bit to balance encoding sequence with timing jitter performance. Consequently, the (information) rate of the transmitted sequence, called HdB sequence is 0.5. The formal definition of information rate is given section 2.1.2. This quantity is usually expected to be as high as possible.

Moreover, to generate positioning sequences, dBTS applies Linear feedback shift register algorithm, which is fast, but depends on finding a suitable primitive polynomial first. To determine the location of a subsequence in the whole positioning sequence, the system uses a look up table, which is a costly approach. More details about the generate algorithm, look-up table, and the analysis of this system are presented in the next chapter.

1.2 The contributions and organization of this thesis

Such flaws of Zhang, Oi, Lowndes, *et al.*'s system, fortunately, can still be improved. That is the goal this thesis aims to. The main task here is to design a code satisfying the constraints of dBTS system.

Problem Statement: Designing a high rate sequence that is capable of positioning and avoids long periods with no pulse

This problem is similar to constructing a constrained positioning sequence. In this thesis, RdB are developed. The longest RdB sequences, not only have a high rate but also are generated and decoded rapidly. In summary, the contributions of this thesis are listed as follows:

- Proposing a new kind of sequence (RdB), more efficient (higher rate, more general and adaptive) than HdB sequences.
- Determining the length of the longest RdB sequences.
- Determining the maximal asymptotic rate of RdB sequences.
- Providing fast encoder and decoder based on state-of-the-art algorithms.

The rest of this thesis is organized as follows. **Chapter 2** gives a brief introduction to coding theory and the application of de Bruijn sequences in such a research area. Other important results surrounding de Bruijn sequences and their generalizations are also provided. **Chapter 3** describes precisely the proposed run length limited de Bruijn sequence. For more understanding, the graph presentation of such sequences is presented. **Chapter 4** studies the properties of the longest run length limited de Bruijn sequence. This chapter answers the major questions: How long is that sequence? What is its rate and maximal asymptotic rate? How to generate the longest run length limited de Bruijn sequence of order k? Also, how to locate the position of each length k sub-string of such a sequence?

CHAPTER 2. DE BRUIJN SEQUENCE AND ITS RELATED RESULTS

The combinatorial object is one of the major focuses in coding theory, for instance, binary complementary sequences, Hadamard matrices, and de Bruijn sequences, ... Introduced in 1946, de Bruijn sequences have been well studied and applied in various fields. This thesis is interested in its novel applications, synchronization in quantum communication.

In this chapter, section 2.1 gives a brief introduction to coding theory. The de Bruijn sequence, its generalizations and applications are presented in section 2.2, 2.3, and 2.4. Finally, section 2.5 analyzes the de Bruijn based timing and synchronization system proposed by Zhang, Oi, Lowndes, *et al.* The analysis points out that the coding schema in such a system can be improved to achieve a higher rate. Moreover, it's also necessary to design algorithms that help the system to encode and decode more efficiently.

2.1 Coding Theory

Transmitting, storing, protecting data (and so on) are challenging problems because of various factors: noisy channels, bandwidth, inter-symbol interference, ... Coding theory is the study of the properties of codes and their fitness that helps dealing with these issues. In academic research, codes are involved in data-transmission, data-storage, data-compression, cryptography, error-detection and correction.

2.1.1 Brief overview

The article, ”A Mathematical Theory of Communication” by Claude Shannon, published in 1948, was considered to mark the birth of Coding Theory. In his work, Shannon showed that when a noisy communication channel is given, he defined a number, called the capacity of the channel, such that reliable communication can be achieved at any rate below the channel capacity if proper encoding and decoding techniques are used.

For more than half a century, coding theory has seen phenomenal growth. Many codes have been well-studied and have various applications in real life. For example, Reed-Solomon code is used in 3G, and 4G networks and Turbo code is used in 5G networks. Both Turbo code and LDPC code are channel coding techniques that Data modems, telephone transmission, and NASA Deep Space Network use to get the bit through.

Usually, coding is divided into *source coding* and *channel coding*.

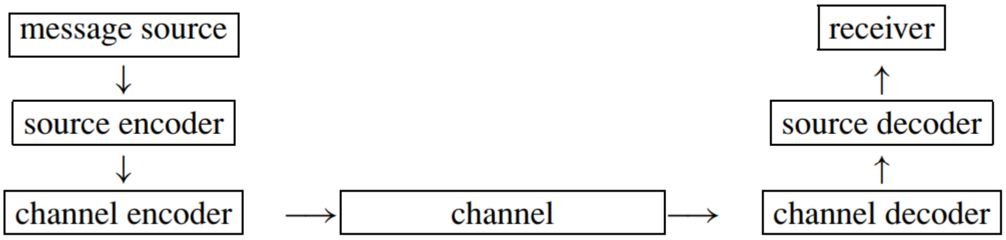

Figure 2.1: Model of source and channel coding [8].

Source coding plays the role of changing the message source to a code that is suitable for transmitting through the channel. For example, ASCII code is a source coding standard converting each character to a byte of 8 bits is an example of source coding. Another way to think about source coding is to treat it as a compress-decompress process. At the transmitter, the source encoder compresses the message for the purpose of economizing on the length of the transmission. At the other end, the source decoder decompresses the received signal or sequence. The commonly used compression algorithms include Huffman code used in JPEG, MPEG, MP3 files, Lempel-Ziv code used in ZIP files, ...

Because of physical and engineering limitations, channels are not ideal: their output may differ from their input because of noise or manufacturing defects. The transmitted message may become distorted and the receiver might not realize that the message was corrupted. Additionally, there are applications, such as magnetic and optical mass storage media, where certain patterns are not allowed to appear in the channel's bit stream. The main role of channel coding is to overcome such limitations by encoding the message again after the source coding while maintaining the channel as transparent as possible from the source and destination points of view.

Section 2.1.1 has introduced basic ideas of coding theory. In the next section, this thesis provides the commonly used notations and terminologies of this research field.

2.1.2 Notation and terminologies

The most essential element of coding theory is **codeword**, which is a sequence of code symbols taken from a code alphabet

Definition 1 (Code Alphabet)**.** A code alphabet is set $\Sigma = \{a_1, a_2, \ldots, a_q\}$ of size q. The elements of Σ are called code symbols, letters, and bits if $q = 2$. A q-ary word of length n over Σ is a sequence (or string) $\boldsymbol{w} = w_1 w_2 \ldots w_n \in \Sigma^n$ with each $w_i \in \Sigma$ for all i.

In practice, the size of a code alphabet is often the size of a finite field, which is the power of a prime number. Hence, for simplicity, Σ can be treated as a set of the first q non-negative integers without ambiguity. More particularly, the notation $\Sigma = 0, 1, 2, \ldots, q-1$ can be used instead.

Definition 2 (Code and Codeword)**.** A q-ary block code C over Σ is a nonempty set C of $q - ary$ words of the same length n. Each element of C is called a codeword in C.

The study of a code C involves the following process in an example of channel coding. Suppose that Σ and Σ' are finite input and output of the channel respectively. Let $\boldsymbol{m}$, taken out of M possible information words, be a message input to the channel encoder. Through a desired channel encoder, the message $\boldsymbol{m}$ is mapped to a longer codeword $\boldsymbol{c} \in \Sigma^n$. The word $\boldsymbol{c}$ is transmitted through the channel, become $\boldsymbol{y} \in \Sigma'^n$. After receiving $\boldsymbol{y}$, the role of the channel decoder is to produce codeword $\hat{c}$ and a decoded information word $\hat{\boldsymbol{u}}$, aiming to have $\boldsymbol{c} = \hat{\boldsymbol{c}}$ and $\boldsymbol{u} = \hat{\boldsymbol{u}}$. Consequently, the mapping at the channel encoder needs to be one-to-one, and the size of the code C is the maximal possible number of messages, or $|C| = M$.

Observe that, using code C, it takes a sequence of length n to encode a sequence of length $\log_{|\Sigma|}(|C|)$. In other words, $n - \log_{|\Sigma|}(|C|)$ ”redundant” bits were added to the message so that the channel can achieve its goal. Accordingly, a quantity concerning this redundancy was introduced, called *(information) rate*.

Definition 3 (Information rate)**.** The (information) rate of a code C over an alphabet of size q is defined as:

$$R_C = \frac{\log_q(|C|)}{n}. \tag{2.1}$$

Works in coding theory, including this thesis, are interested in designing codes with a high rate, along with efficient encoder, and decoder, that can be used in specific situations.

Based on their motive or their intrinsic properties, codes are categorized into linear codes, constrained codes, error-correcting codes, and error-detecting codes, ... This thesis focus on the combination of a constrained code, run length limited, and positioning code. A brief introduction to constrained code is given in the next section.

2.1.3 Constrained code

Constrained Code is a sub-field of Coding theory, studies to design codes satisfying given constrained. The inspiration for the research of constrained codes comes from real problems. The transmitted data needs to follow some given standards which are necessary for the code to surmount the flaw of the environment. For instance, in CD disc storage, errors tend to occur when there is a sequence of many consecutive 0 bits. Consequently, it's crucial to construct codes that should avoid a long sequence of 0 bits. A famous code invented to overcome this challenge is Run length limited code by Immink [9]. RLL codes are defined by 2 parameters: d, k, and denoted by (d, k)-RLL, where d and k are two non-negative integers such that $d \leqslant k$. A finite length binary sequence is said to satisfy the (d, k)-RLL constraint if its number of 0's between 2 consecutive 1 bits is at least d and at most k.

An illustration is a convenient way to begin understanding things. In constrained code, a graph, usually called a labeled graph, is a helpful visualization technique. More particularly, a labeled graph is a directed graph with its vertices and edges labeled. Vertices in the labeled graph are also called states. And the start and end vertices of a directed edge are called initial and terminal states respectively. Given a state v, in-edges of v are edges treating v as a terminal state. Similarly, out-edges of v are edges taking v as an initial state.

For example, the graph in figure 2.2 represents a (d, k)-RLL code. It can be verified that a sequence w satisfies the (d, k)-RLL constraint if and only if a path whose edge labeling is w exists in the graph.

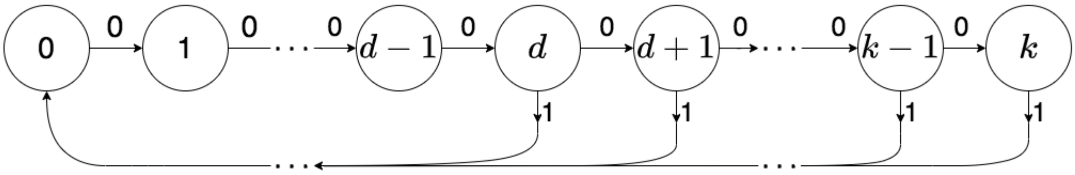

Figure 2.2: Graph representation of (d, k)-RLL code.

Labeled graphs are however more than just visualization tools. Using the finite state splitting algorithm, they become encoders. In a constrained system, a very common problem is designing an encoding algorithm, which maps arbitrary user sequences into sequences obeying the constraints. Nevertheless, it's crucial to note that there are many kinds of encoders, depending on their objectives. For instance, there are encoders not taking any sequences as input, their goal is just to generate sequences satisfying given constraints. Such encoders are focused on in this thesis.

Besides constrained code, another combinatorial object drawing much attentions in coding theory is positioning code, also known by the name de Bruijn code. The formal definition of this code and its important results are presented in the next section.

2.2 De Bruijn Sequence

A de Bruijn sequence (of order k), sometimes called a positioning sequence, over an alphabet Σ, is a sequence of symbols of Σ such that all subsequences over Σ of length k appear exactly once. This section first explains how to use a graph to represent de Bruijn sequences, and then introduces methods to generate or decode such sequences. Important results on the granddaddy, one of the most interesting de Bruijn sequences, which play a significant role in this work, are also given.

2.2.1 Graph presentation of de Bruijn sequences

Since the first time introduced in 1946 by de Bruijn himself, the de Bruijn graph and its related sequences have been well-studied and generalized under numerous names, including positioning sequences, m-sequences, shift register sequences [10]–[14]. The goal of de Bruijn was to find a recursive algorithm to enumerate the number of cyclic binary sequences of length 2^k such that each binary k-tuple appears as a window of length k exactly once in each sequence.

The first results in the de Bruijn graph focused on the alphabet of size 2. Later, in 1951, van Aardenne-Ehrenfest and de Bruijn [15] generalized the enumeration result for any arbitrary alphabet of finite size q, using a generalized graph for an alphabet Σ of size q.

Definition 4 (de Bruijn graph)**.** Formally, the de Bruijn Graph of order k, G_k is a directed graph with q^{k-1} vertices, each one is represented by a word of length $k-1$ over an alphabet Σ with q letters. A directed edge from the vertex $\boldsymbol{x} = (x_0, x_1, \ldots, x_{k-2})$ to the vertex $\boldsymbol{y} = (y_1, y_2, \ldots, y_{k-1})$, represented by the symbol x_k, where $x_i, y_i \in \Sigma$, if and only if $x_i = y_i$ for all $1 \leqslant i \leqslant k-2$. We call this edge x_k the out-edge of $\boldsymbol{x}$, and the in-edge of $\boldsymbol{y}$. Progressively, the in-degree and out-degree of a vertex $\boldsymbol{x}$ are the numbers of in-edges and out-edges of $\boldsymbol{x}$ respectively.

Deduced from the definition, the in-degree and out-degree of each vertex are q. Thus, a de Bruijn graph is an Eulerian graph. Figure 2.3 gives an illustration for the graph G_4.

Path: A *path* in the graph is a sequence of edges: $e_0, e_1, \ldots, e_n$ such that the terminal vertex of edge e_i is the the initial vertex of edge e_{i+1} for all $0 \leqslant i \leqslant n-1$. A *simple path* is a path going through each edge at most one time. Each longest

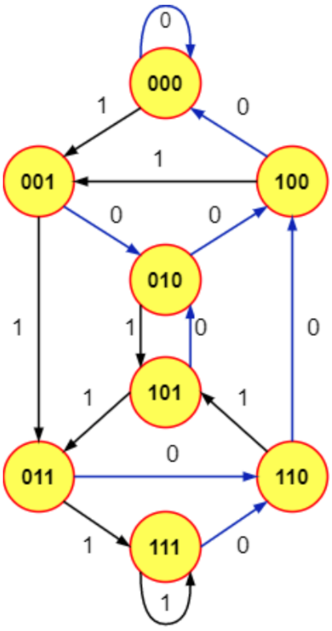

Figure 2.3: de Bruijn graph of order 4, G_4.

simple path in a de Bruijn graph is an Eulerian cycle. A sequence formed by concatenating the symbol of each edge in the longest simple path in G_k is called a (cyclic) de Bruijn sequence of order k. All the strings of length k appear exactly once in each such de Bruijn sequence. The acyclic version of the de Bruijn sequence can be obtained by prepending the sequence representing the first vertex in the corresponding Eulerian cycle to the cyclic one.

Example 2.1. Consider an Eulerian cycle starting at vertex 000, then the symbols representing the following edges $0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0$ form a de Bruijn sequence order 4. Adding 000 to its beginning results in an acylic one:

$$0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0$$

.

The number of longest simple path in G_k, and also the number of de Bruijn sequences, have been proved in [15] to be $\dfrac{q!^{q^{k-1}}}{q^k}$.

Example 2.2. For $q = 2,\ k = 4$, there are 16 distinct de Bruijn sequences. From figure 2.3, those de Bruijn sequences are found and listed as follows:

$$\begin{array}{ll}
0000100110101111 & 0000100111101011 \\
0000101001101111 & 0000101001111011 \\
0000101100111101 & 0000101101001111 \\
0000101111001101 & 0000101111010011 \\
0000110010111101 & 0000110100101111 \\
0000110101111001 & 0000110111100101 \\
0000111100101101 & 0000111101001011 \\
0000111101011001 & 0000111101100101
\end{array}$$

2.2.2 Encode and decode de Bruijn sequences

Encoding de Bruijn sequences concerns generating an arbitrary de Bruijn sequence or a de Bruijn sequence satisfying some given constraints. Finding a de Bruijn sequence is equivalent to seeking an Eulerian cycle in a de Bruijn graph. In [16], [17], efficient algorithms to find Eulerian cycles are presented. Especially, the approach in [16] can be used to generate all binary de Bruijn sequences. However, since the graph must be stored, applying such algorithms to find a positioning sequence requires exponential $O(q^k)$ space.

Besides the graph-based approach, there are other well-known methods to construct such sequences, including LFSR, recursive methods, greedy methods, and concatenation approaches.

The idea of LFSR is to design a feedback function f mapping length k strings to $\{0,1\}$. Starting with an initial length k string, f is repeatedly applied to the last k symbols of the current string to generate the next symbol until the maximal length of a de Bruijn sequence is reached. If f is linear, then, the function $F(\alpha) = \alpha f(\alpha)$ is said to be a LFSR, where α is a length k string. LFSRs based on primitive polynomials generate maximal length sequences (positioning sequences) having length $2^k - 1$ that miss only the all 0 string. The downside of this method is that it's compulsory to find a primitive polynomial first.

De Bruijn sequences can also be constructed via recursion by applying Lempel's D-morphism $D : \{0,1\}^m \to \{0,1\}^{m-1}$ which maps a string $\alpha = \alpha_1\alpha_2 \dots \alpha_m$ to $\beta = \beta_1\beta_2 \dots \beta_{m-1}$, where each $b_i = a_i + a_{i+1} \ (mod\, 2)$. Nevertheless, an exponential amount of space is also required by these recursive strategies.

Surprisingly, greedy approaches are also able to generate de Bruijn sequences. The greedy construction starts with a seed string, then repeatedly applies some greedy rule to determine the next symbol of a sequence. The algorithm stops when it is impossible to add another symbol without creating a duplicate substring of length k, or some termination condition is reached. The different explicit greedy rules result in different implementation greedy algorithms [12], [18]–[20]. Such constructions, however, have a major drawback: they require exponential space.

Despite many constructions being known, and even a useful survey has been given by Fredricksen [12], things are not quite the same for the decoding problem. This problem, discovering the position within a particular sequence of any specified k-tuple, has been much less well studied. There are just some classical de Bruijn sequences with sub-linear decoding algorithm [21]–[23].

2.2.3 Results on lexicographically minimal de Bruijn sequence

The lexicographically minimal de Bruijn sequence, or **granddaddy sequence** as called by Knuth [24], is one the most interesting among other de Bruijn sequences. An example of granddaddy is provided in example 2.3.

Example 2.3 (Granddaddy of order 6)**.**

$$0000001000011000101000111001001011001101001111010101110110111111$$

Both efficient encoder and decoder of this sequence have been found.

The encoding algorithm is actually a concatenation scheme, which is later called FKM algorithm, the abbreviation of Fredrickesen, Kessller, and Maiorana, who discovered this strategy [25], [26]. Its complexity has been proved to be constant amortized time per symbol by Frank Ruskey et.al [27] in 1992.

Though its construction and the related algorithm has been found in 1978, about 40 years ago, the granddaddy's decoder has just been discovered recently in 2016 by Kociumaka, Radoszewski, and W. Rytter [23]. Denote $\boldsymbol{x}$ as a granddaddy sequence of order k, and v is a length k arbitrary substring. Then the decoding algorithm, denoted by $\mathcal{D}_{KRR}$, returns $\mathcal{D}_{KRR}(v)$ being the one and only position of v in the whole sequence $\boldsymbol{x}$. Kociumala et.al also proved that $\mathcal{D}_{KRR}$ works in $O(k^2 \log(q))$-time in the word-RAM model and $O(k^2)$-time in unit-cost RAM model.

2.3 Universal Cycles

A more general way to look at de Bruijn sequences is the universal cycle (U-cycle).

Definition 5 (Universal cycle)**.** Given a finite set $\mathcal{T}_k$ of distinct of combinatorial objects of "rank k", an U-cycle of $\mathcal{T}_k$ is a cyclic sequence $\mathcal{U} = (a_0, a_1, \dots, a_n)$ such that $(a_{i+1}, \dots, a_{i+k})$, $0 \leqslant i \leqslant n$, run through each element of $\mathcal{T}_k$, where index addition is performed modulo n.

An order k binary de Bruijn sequence is eventually an U-cycle of the set of all length k binary strings. The studies of U-cycle are concerned with the existence and construction of U-cycles for many combinatorial objects such as strings, permutations, partitions, subsets, multisets, lattice paths, vector spaces weak orders, etc [28]–[33]. Section 2.3.1 provides results for permutations, partitions, and subsets of a set of n distinct symbols, where n is a positive integer. These results were first summarized by Chung, Diaconis, and Graham in [28].

2.3.1 Permutations, partitions and subsets of n distinct symbols

Consider the set S_n of all $n!$ permutations of $\{1, 2, \ldots, n\}$. With each value of n, set S_n may not always contain any U-cycles, such as $n = 3$. All 6 permutations of $\{1, 2, 3\}$ are $S_3 = \{123, 132, 213, 231, 312, 321\}$, and the longest cycle in S_3 one can travel is of length 4, for instance, $123 \rightarrow 231 \rightarrow 312 \rightarrow 123$, which still lacks $132, 321$.

However, if order-isomorphism is allowed instead of requiring exact matches, U-cycles of S_n exists. More precisely, an U-cycle $U_n = (a_0, a_1, \ldots, a_{n!-1})$, $a_i \in \{1, 2, \ldots, N\}$, for S_n is $n!$-tuple such that each element of S_n is order-isomorphic to exactly one block $(a_{i+1}, \ldots, a_{i+n})$, where $a_i = a_{j \equiv i (\mod n!)}$. Here, two n-tuples $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ are called order-isomorphic, written as $\boldsymbol{a} \sim \boldsymbol{b}$, if $a_i < a_j \Leftrightarrow b_i < b_j$ for all $0 < i, j \leqslant n$. An example of U-cycle for S_3 is :

$$1\ 4\ 5\ 2\ 4\ 3$$

By order-isomorphism, each 3-tuple in the above U-cycles can be mapped into elements of S_3 as follow:

$$145 \sim 123$$
$$452 \sim 231$$
$$524 \sim 312$$
$$243 \sim 132$$
$$431 \sim 312$$
$$314 \sim 213$$

and hence, the equivalent cycle is $123 \rightarrow 231 \rightarrow 312 \rightarrow 132 \rightarrow 312 \rightarrow 213 \rightarrow 123$. The construction of de Bruijn graphs can be imitated to construct the transition graph for S_n. Each permutation plays the role of a vertex. Their suffix of length $n-1$ is analyzed to establish its edges to other permutations. Takes the vertex 231 of S_3

as an example. From its suffix 31, one can go to 312. But since order-isomorphism is accepted, and note that $31 \sim 21 \sim 32$, there are also edges connecting 231 to 213 and 321. The whole transition graph of S_3 is shown in figure 2.4.

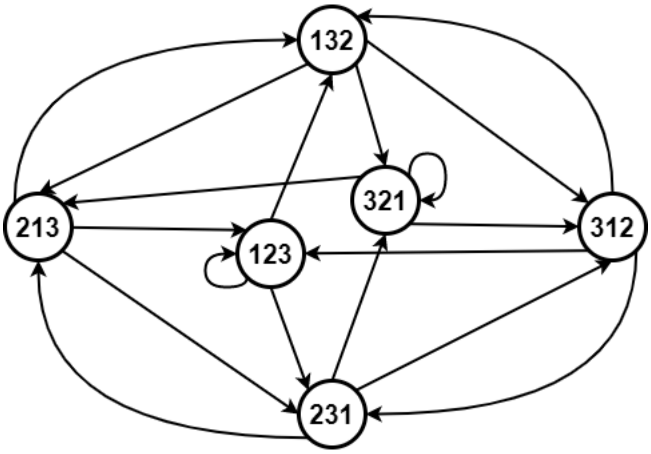

Figure 2.4: Transition graph of S_3.

It is proved that the transition graph of S_n is Hamiltonian, and a Hamiltonian cycle in the transition graph corresponds to an U-cycle in S_n. Now, the key question is how to represent a U-cycle of an Hamiltonian cycle, like the sequence $1\ 4\ 5\ 2\ 4\ 3$ represents $123 \rightarrow 231 \rightarrow 312 \rightarrow 132 \rightarrow 312 \rightarrow 213 \rightarrow 123$. Even with S_3, is 5 the smallest number of distinct symbols necessary for an U-cycle. More generally, how many distinct symbols does an U-cycle of S_n use at least?

Actually, in S_3, one can do better with 4 symbols. For example, the sequence $1\ 4\ 2\ 3\ 4\ 2$ is the representation of a Hamiltonian cycle:

$$132 \rightarrow 312 \rightarrow 123 \rightarrow 231 \rightarrow 321 \rightarrow 213 \rightarrow 132$$

The following sequence is an example with 5 symbols for S_4:

$$1\ 2\ 3\ 4\ 1\ 2\ 5\ 3\ 4\ 1\ 5\ 3\ 2\ 1\ 4\ 5\ 3\ 2\ 4\ 1\ 3\ 2\ 5\ 4$$

Let $N(n)$ be the minimum number required for an U-cycle of S_n, it is proved that:

$$N(2) = 2,\ N(3) = 4,\ N(4) = 5 \text{ and } n + 1 \leqslant N(n) \leqslant 6n \text{ for } n \geqslant 5$$

Fan Chung believes that the equation happens at $n + 1$. However, their belief remains an unsolved conjecture until now.

Conjecture 1. $N(n) = n + 1$.

Constructing transition graph is also help finding U-cycle for the set of P_n of partitions of the set $\{1, 2, \ldots, n\}$. The partitions can be represented by sequence of length n $(a_0, a_1, \ldots, a_n)$, where $a_i = a_j$ indicates the i-th element and j-th element are in the same group of the partition. The transition graph of P_n is illustrated in figure 2.5.

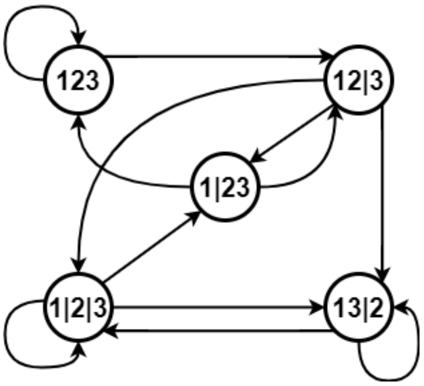

Figure 2.5: Transition graph of P_3.

The transition graph of P_n is proved to be Hamiltonian by showing that it can be clustered to be an Eulerian graph.

It is more challenging to study the family $\begin{bmatrix} n \\ k \end{bmatrix}$ of all k-element subsets of a set of n distinct elements $\{0, 1, \ldots, n-1\}$. This thesis provides an example of an U-cycle for the case $n = 5, k = 2$:

$$1\ 3\ 2\ 5\ 4\ 2\ 1\ 5\ 3\ 4$$

The question about the condition for the existence of the universal cycle for such a family is still not answered completely. The difficulty here is that a transition graph for $\begin{bmatrix} n \\ k \end{bmatrix}$ isn't able to be defined explicitly. This issue is caused by the distinguishing feature of an k-set. More precisely, a k-element subset might occur in any $k!$ possible order in the U-cycle, but it is only allowed to occur once. Fan Chung and Graham made a conjecture on this problem and the first person who can solve it would earn a prize awarded by the author's conjecture.

Conjecture 2 (Fan Chung's conjecture)**.** Universal cycle for $\begin{bmatrix} n \\ k \end{bmatrix}$ always exists provided k divides $\begin{pmatrix} n-1 \\ k-1 \end{pmatrix}$ and n is large enough.

It's easy to see that conjecture 2 is true for $k = 1, 2$. Effort on this question has just cracked completely the cases $k = 3, 4, 5$ (with some aid of a computer), and $k = 6$ whenever n and k are relatively prime([34], [35]). For $k \geqslant 7$, and $k = 6$ when n and k are not relatively prime, conjecture 2 remains open.

Besides studying the existence of U-cycle on different kinds of sets, designing algorithms that create universal cycles is also concerned.

2.3.2 Universal cycles algorithms for other classes of sets

There are researches focusing on using generalized the FKM algorithm and greedy algorithm to create universal cycles for a class of sets. Moreno proved that this method works for the set of rotations of the lexicographically largest i necklace [36]. The aperiodic strings in the set of all k-ary strings of length n can be generated the same way as shown by Au in [37]. All these results are later generalized by Joe Sawada et.al in [38]. More particularly, let S be the set of length n k-ary strings such that the following closure conditions are obeyed:

- The set of strings S is closed under rotation.
- Its subset of necklaces is closed under replacing any suffix of length i by k^i.

Then, the greedy and FKM algorithm create the lexicographically smallest universal cycle of S. Several such classes S are listed in example 2.4.

Example 2.4. Recall that $\Sigma_q = \{1, 2, 3, \ldots, q\}$ is a code alphabet of size q, and Σ_q^n is the set of q-ary sequences of length n over alphabet Σ. The following sets satisfy the closure conditions for the existence of universal cycles proved by Sawada [38].

- **Minimum Sum**: S $\in \Sigma_q^n$ is a set of length n strings with sum over all of its symbol at least s, where s is a given constant.
- **Frequency of** q: S $\in \Sigma_q^n$ contains the strings with at least l_q copies of q, where l_q is a given constant.
- **Frequency of** $i < q$: S $\in \Sigma_q^n$ contains the strings with at most u_i copies of $i < q$. Here, u_i is a given constant.
- **Avoiding a substring**: S $\in \Sigma_q^n$ contains the strings that do not contain a given pattern $\alpha \in \Sigma_{q-1}^m$, for some $m \geqslant 1$, as a cyclic substring.
- **Union and Intersection**: Let S$_1$ and S$_2$ be 2 set obeying the closure conditions, then both S$_1 \cup$ S$_2$ and S$_1 \cap$ S$_2$ also satisfy those conditions.

Note that, in example 2.4, the union and intersection of the proper sets S allow to combine the previous results to create more interesting classes of sets that have universal cycles.

Section 2.3 has provides different research directions and results on the universal cycle, which is a generalization of de Bruijn sequences. The applications of de Bruijn sequences and their generalizations will be presented in the next section.

2.4 Applications

The reason why the de Bruijn graph, its sequence, and its generalizations are having so much attention is due to their diverse important applications. Very soon after the formal definition of this graph was given birth, one of its first applications was found in the introduction of shift-register sequences in general and linear feedback registers in particular [39]. Throughout the years, these types of sequences and graphs have found a variety of applications.

In cryptography, for example, the Baltimore Hilton Inn used de Bruijn sequences to install a cipher lock system for each of its rooms in lieu of the conventional key-lock system [12]. The low-cost n-stage shift register was used to generate maximum-length pseudorandom sequences in stream cipher, though later, this method was proved to be vulnerable to known-plaintext attack [40].

De Bruijn sequences also opened a new field of research surrounding their complexity. Agnes Hui Chan et.al studied the complexity and the distribution of the complexities of de Bruijn sequences [41]. Especially, for binary sequences with period 2^n, they come up with a fast algorithm determining its complexity [42]. Edwin on himself analyzed the structure and complexity of nonlinear binary sequence generators [43]. Tuvi et.al studied the error linear complexity spectrum of binary sequences with period $2n$ [44]. Also Tuvi, in his joint work with Lampel [45], found a construction of the de Bruijn sequence to show that the lower bound of its complexity ($2^{n-1} + n$) is attainable for all n.

In [46], A.Lampel and M.Cohn are interested in designing a universal test sequence for VLSI (very large-scale integration chip). A binary sequence is called (s, t)-universal, $s > t$, if when shifted through a register of length s, it exercises every subset of t register positions. Their proposed method was concatenating a set of de Bruijn sequences of appropriate length. In [47], Zeev Barzilai .et.al also demonstrated an application of de Bruijn sequence in VLSI self-testing.

There are also other applications requiring a two-dimensional version of de Bruijn sequences. And the research about the two-dimensional generalization of de Bruijn sequences comes to call. One well-known version is called pseudo-random arrays. In 1976, Mac Williams and Neil Sloane [48] gave a simple description of pseudo-random arrays and studied several of their nice properties. In 1988, Tuvi [49], represented a new version of pseudo-random arrays to construct perfect maps. In another approach by Bruck Stein [50], he combined a de Bruijn sequence and a half de Bruijn sequence to study its robust and self-location properties. Studies [51]–[55] used pseudo-random arrays to with applications to robust undetectable digital watermarking of two-dimensional test images, and structured light.

More surprisingly, de Bruijn's modern applications are even combined with biology, like genome assembly as part of DNA sequencing. For example, Chaisson et.al [56] described a new tool, EULER-USR, for assembling mate-paired short reads and used it to analyze the question of whether the read length matters. Compeau et.al [57] represented a method using the de Bruijn graph for genome assembly. In 2001, Pevzner et.al [58] abandoned the classical "overlap - layout - consensus" approach in favor of a new Eulerian Superpath approach, that, for the first time, resolves the problem of repeats in fragment assembly. Later on, in 2003, Yu Zhang and Michael Waterman [59], adapted Pevzner's method to global multiple alignment for DNA sequences. In DNA storage, Han Mao et.al [60], [61] studied codes and their rates for DNA sequence profiles. Their studies were based on the de Brujn graph.

In some new memory technologies, mainly in racetrack memories, and other ones which can be viewed as an l-read channel, synchronization errors (which are shift errors known also as deletions and sticky insertions) occur. By proposing a new de Bruijn-based schema, using a locally-constrained de Bruijn sequence to construct such code, Chee et.al. [62] are able to increase the rate of codes that correct the synchronization errors. Locally constrained de Bruijn sequences and codes (sets of sequences) are of interest in their own right from both practical and theoretical points of view.

Recently, in 2021, a novel application of the de Bruijn sequence has been found in quantum communication. Generally, to transmit quantum information between a satellite and the ground station, a timing and synchronization system has been used. Having observed that the intrinsic properties of the positioning sequence are very suitable for this system, Zhang, Oi, Lowndes, *et al.* [1] have modulated it into HdB sequence to transmit along the quantum channel. Their system is analyzed in the next section.

2.5 Motivation

In satellite QKD, the quantum information is synchronized by transmitting along with the classical one. Figure 2.6 shows a high-level view of this schematic.

At the beacon source, a de Bruijn sequence is modulated before transmitting to the ground.

The encode process happens at the satellite, where it uses the LFSR algorithm

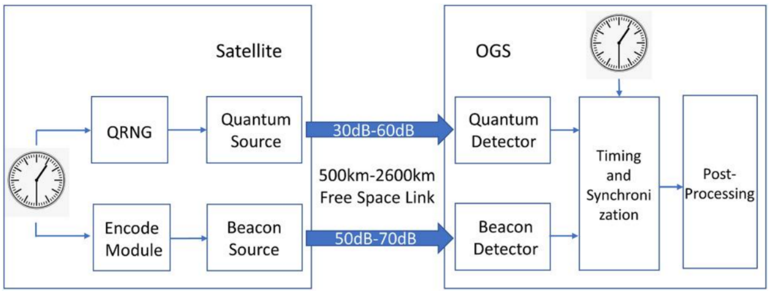

Figure 2.6: High-level satellite Quantum Key Distribution timing and synchronization schematic [1].

to generate a positioning sequence (of order k for example). The prerequisite of finding a proper primitive polynomial is the main drawback of this encoder.

The decoding process takes place at the ground station, a look-up table is used to identify the unique position of the received sequences of length k. The complexity of this method is exponential.

Furthermore, in this model, a sequence of beacon pulses is used to represent a binary de Bruijn sequence. Considering the timing jitter performance, a long period of no-pulses should be forbidden. If one pulse slot is used to represent a binary bit, for example, on is 1 and off is 0, a long run of 0's in the sequence (which is a long period of no-pulses) would impact the timing jitter. In [1], two pulse slots are used to represent a single bit (on-on is 1 and on-off is 0) so that one can avoid two consecutive no-pulses. The transmitted sequence is called HdB. However, the above scheme requires $2n$ pulse slots to represent a de Bruijn sequence of length n and needs to receive a sub-sequence of $2\log n$ pulse slots to locate its position. Formally, the HdB sequence's rate is just 0.5, where rate is a quantity that needs to be as high as possible (the definition of sequences' rate is given in section 4.2).

Table 2.1 summarized the process, applied methods and drawbacks of dBTS system using HdB sequence.

This thesis's target is to surmount the drawbacks listed above.

In order to avoid long period of no pulse, the positioning sequences are combined with run length limited constraints. Such sequences are called Run length limited de Bruijn (RdB) sequences, presented in chapter 3. The RdB sequences are not just suitable with dBTS system, but also have a higher rate than HdB sequences. More precisely, rate of the longest RdB sequences are

$$\log\left(\frac{1+\sqrt{5}}{2}\right) \approx 0.6942.$$

Table 2.1: Timing and synchronizing system use Hybrid de Bruijn code.

Satellite	**Transmited de Bruijn sequence**	**Ground Station**
Requirement: transmit a de Bruijn sequence of order k	**Requirement**: be able to be positioned, avoid periods of no-pulse	**Requirement**: locate the position of any length k subsequence
Method: use linear feedback shift register algorithm	**Method**: modulate on-on is 1, on-off is 0	**Method**: use look-up table
Drawback: the prerequisite of finding a proper primitive polynomial	**Drawback**: rate is 0.5	**Drawback**: exponential complexity

So as to generate one of the longest RdB sequences, based on FKM algorithm, chapter 4 provides an encoder whose time complexity is constant amortized time per symbol. Moreover, to locate the position of an arbitrary proper subsequence in the whole RdB sequence, a decoder is also presented. The proposed decoder modifies the decoding algorithm found by Kociumaka, Radoszewski, and Rytter in [23], which is currently the state of the art method to position a subsequence in the de Bruijn sequence, and therefore, is better than look-up table.

Beside, the RdB sequence is even more general and adaptive. More particularly, when the constraint of forbidding pattern 00 is relaxed, that is, a longer run of bit 0's is allowed, the RdB sequence can be easily adjusted to make the its rate higher and still suits the system.

CHAPTER 3. RUNLENGTH LIMITED DE BRUIJN SEQUENCE

In this chapter, new constrained de Bruijn sequences are introduced. Since a de Bruijn sequence is self-located, the run length limited constraint is the only requirement remaining that this sequence needs to satisfy to be used in the dBTS system. Therefore, combining a positioning sequence and a run length limited sequence is a natural solution. That is also how this thesis gave birth to the name: Run length limited de Bruijn sequence. Besides, just like other constrained codes, a convenient way to apprehend a code is using a labeled graph. Hence, the graph presentation of these sequences is also provided.

3.1 Run length limited de Bruijn sequence

Let n, k, s, q be some positive integers and $\Sigma_q = \{0, 1, 2, \ldots, q-1\}$ be an alphabet of size q. A *sequence* $\boldsymbol{s} = (s_1, s_2, \ldots, s_n) \in \Sigma_q^n$ is over an alphabet Σ, that is, $s_i \in \Sigma_q$. This thesis only focuses on the case $q = 2$ and thus drops q in the notation for simplicity. Sequence $\boldsymbol{s} = s_1 s_2 \ldots s_n \in \Sigma^n$ is also written without ambiguity. The window (substring) $(s_i, s_{i+1}, \ldots, s_j)$ is denoted by $s[i, j]$.

Given two sequences $\boldsymbol{x} = x_1 x_2 \ldots x_m$ and $\boldsymbol{y} = y_1 y_2 \ldots y_n$, denote the concatenation of $\boldsymbol{x}$ and $\boldsymbol{y}$ to be $\boldsymbol{xy} = x_1 x_2 \ldots x_m y_1 y_2 \ldots y_n$, and denote $\boldsymbol{x}^k$ the concatenation of k copies of $\boldsymbol{x}$. It is said that $\boldsymbol{x}$ is smaller than $\boldsymbol{y}$, denoted $\boldsymbol{x} < \boldsymbol{y}$, if there is an index $t \geqslant 1$, such that $x_i = y_i$, $\forall i \leqslant t$, and $x_{t+1} < y_{t+1}$. Note that empty sequence is smaller than 0.

Definition 6. A sequence $\boldsymbol{s} = (s_1, s_2, \ldots, s_n)$ is called a s-run length limited (RLL) sequence of length n if each run of 0's in the sequence $\boldsymbol{s}$ has length at most s, or in other words, the sequence $\boldsymbol{s}$ does not contain $s+1$ consecutive 0's as a substring. A set of s-RLL sequences of length n is called a s-RLL code and denoted $C(n, s)$.

Denote $W(n, s)$ the set of all s-RLL sequences of length n and note that $W(n, s)$ is the maximal s-RLL code. The s-RLL code $C(n, s)$ and the cardinality $|W(n, s)|$ has been well-studied in the literature [63], [64]. This thesis presents the recursive formula of $|W(n, s)|$ with proof.

Lemma 1 (Cardinality of $W(n, s)$)**.** Let n, s be two non-negative integers. Then

$$|W(n,s)| = 2^s,\ \forall 0 \leqslant n \leqslant s$$

$$|W(n,s)| = \sum_{i=0}^{s} |W(n-i-1,s)|,\ \forall n > s$$

Proof. For the first equation, when $n \leqslant s$, all sequences of length n belong to

$W(n,s)$. Hence $|W(n,s)| = 2^n$, $\forall 0 \leqslant n \leqslant s$.

When $n > s$, every sequence in $W(n,s)$ is of the form $0^i 1\boldsymbol{x}$, where $\boldsymbol{x} \in W(n-i-1,s)$ for $0 \leqslant i \leqslant s$. Additionally, for every $\boldsymbol{x} \in W(n-i-1,s)$, the sequence $0^i 1\boldsymbol{x}$ is an element of $W(n,s)$. This bijection brings the second equation. □

Table 3.1 lists the very first values of $|W(n,s)|$. Each $|W(i,j)|$ is stored at the crossed cell of column $n = i$ with row $s = j$.

Table 3.1: Values of $W(n,s)$ for all $n = \overline{0,12}$ and $s = \overline{1,9}$.

n s	0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	5	8	13	21	34	55	89	144	233	377
2	1	2	4	7	13	24	44	81	149	274	504	927	1705
3	1	2	4	8	15	29	56	108	208	401	773	1490	2872
4	1	2	4	8	16	31	61	120	236	464	912	1793	3525
5	1	2	4	8	16	32	63	125	248	492	976	1936	3840
6	1	2	4	8	16	32	64	127	253	504	1004	2000	3984
7	1	2	4	8	16	32	64	128	255	509	1016	2028	4048
8	1	2	4	8	16	32	64	128	256	511	1021	2040	4076
9	1	2	4	8	16	32	64	128	256	512	1023	2045	4088

Definition 7 (Run length limited de Bruijn (RdB) sequence)**.** A sequence $s = (s_1, s_2, \ldots, s_n) \in \Sigma^n$ is called a (k,s)-run length limited de Bruijn (RdB) sequence of length n if it is a de Bruijn sequence of order k and a s-RLL sequence of length n.

Example 3.1 gives an instance of RdB sequence.

Example 3.1 ($(5,2)$-RdB sequence)**.** For $k = 5, s = 2$, a $(5,2)$-RdB sequence of length 27 is $s = (0,0,1,1,0,0,1,0,1,0,0,1,1,1,0,1,0,1,1,0,1,1,1,1,1,0,0)$.

Note that, when $s \geqslant k$, a (k,s)-RdB sequence is just an original de Bruijn sequence. If $s = k-1$, any $(k,k-1)$-RdB sequence can be achieved from a de Bruijn sequence removing 1 letter 0 in the subsequence 0^k. Based on those observations, case $s \geqslant k-1$ is considered to be trivial. Therefore, this thesis concentrates on the case $s < k-1$. And thus, in the rest of this thesis, s is always assumed to be smaller than $k-1$.

It is well-known that given k, the maximal length n of a binary acyclic de Bruijn sequence is $n = 2^k + k - 1$. Let $N(k,s)$ be the maximal length of a (k,s)-RdB sequence. This thesis is interested in finding the exact value of $N(k,s)$. The motivation of this task is explained clearly in section 4.2, which concerns the rate of a sequence.

For further demonstration, the next section presents the graph presentation for the (k, s)-RdB sequence.

3.2 Graph presentation of RdB sequence

In this section, a labeled graph, called (k, s)-RdB graph, is used to represent (k, s)-RdB sequences. Just like a de Bruijn graph of order k, any simple path in (k, s)-Rdb graph represents a (k, s)-RdB sequence.

A (k, s)-RdB graph can be achieved by eliminating all the vertices containing more than s consecutive letter 0 in the de Bruijn graph G_k. As a result, the vertices of a (k, s)-RdB graph are represented by binary sequences of length $k - 1$ which don't contain pattern 0^{s+1}.

The illustration for de Bruijn graph of order 4, G_4, was given in figure 2.3. To obtain $(4, 1)$-Rdb graph from there, vertices $000, 001, 100$ are deleted. Figure 3.1 demonstrates the $(4, 1)$-RdB graph.

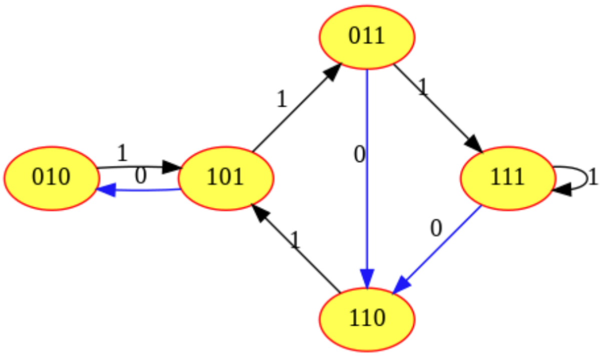

Figure 3.1: $(4, 1)$-RdB graph.

Denote the (k, s)-RdB graph to be $G_{k,s} = (V^{k-1,s}, E^{k,s})$, where $V^{k-1,s}$ is the set of all vertices and $E^{k,s}$ is the set of all edges. The following lemmas determining the cardinality of $V^{k-1,s}$ and $E^{k,s}$

Lemma 2 (Number of vertices).

$$\left|V^{k-1,s}\right| = |W(k, s)| \tag{3.1}$$

Proof. This proof is deduced directly from the construction of $G_{k,s}$ since its set of all vertices is the set of all length $k - 1$ sequences containing at most s consecutive letters 0. □

Lemma 3 (Number of edges).

$$\left|E^{k,s}\right| = |W(k, s)| \tag{3.2}$$

Proof. Observe that for both $\boldsymbol{x}, \boldsymbol{y} \in W(k-1, s)$, if there is an edge from vertex $\boldsymbol{x} = (x_1, \ldots, x_{k-1})$ to vertex $\boldsymbol{y} = y_1, \ldots, y_{k-1}$, then $(x_1, \ldots, x_{k-1}, y_{k-1}) \in W(k, s)$.

Besides, for each s-RLL sequence $\boldsymbol{x} = (x_1, \ldots, x_k) \in W(k, s)$, its prefix and suffix, $(x_1, \ldots, x_{k-1})$ and $(x_2, \ldots, x_k)$, are both s-RLL sequence of length $k-1$. Hence, they represent two vertices in the graph $G_{k,s}$ and their connecting edge is represented by the sequence $\boldsymbol{x}$.

So, there is $1-1$ correspondence between $E^{k,s}$ and $W(k, s)$. This results in $\left|E^{k,s}\right| = |W(k, s)|$. □

Example 3.2. According to lemma 2 and lemma 3, $(4, 1)$-RdB graph has $\left|V^{3,1}\right| = |W(3, 1)| = 5$, and $\left|E^{4,1}\right| = |W(4, 1)| = 8$. These results can be verified by counting the number of vertices and edges in figure 3.1.

Let $u = (u_1, u_2, \ldots, u_{k-1})$ and $v = (v_1, v_2, \ldots, v_{k-1})$ be arbitrary vertices in RdB graph. Starting at v, the following sequence of edges' labels $(1, u_1, u_2, \ldots, u_{k-1})$ apparently forms a proper path going from v to u in RdB graph. Similarly, the sequence of edges' labels $(1, v_1, v_2, \ldots, v_{k-1})$ beginning at u is also a directed path from u to v. This is sufficient to conclude that the connectivity of RdB graphs is preserved.

CHAPTER 4. PROPERTIES OF RUN LENGTH LIMITED DE BRUIJN SEQUENCE

Chapter 4 concerns in determining the rate of RdB sequence. Besides, designing efficient encoder and decoder for a longest RdB sequence are also critical contributions.

To calculate the rate and maximal asymptotic rate of RdB sequence, first, results on the maximal length of a RdB sequence are presented. After that, efficient algorithms to generate a longest RdB sequence and locate any sub-sequence in such sequence are also provided.

4.1 Longest simple path in RdB graph

The rate of de Bruijn sequences can be determined by understanding its longest length. Section 4.2 provides a formal definition of rate and maximal asymptotic rate of the de Bruijn sequence. This section concerns finding the longest simple path in $G_{k,s}$, which corresponds to the longest (k, s)-RdB sequence.

Let $N(k, s)$ is the maximal length of a (k, s)-RdB sequence, and $\ell(G_{k,s})$ be the length of the longest simple path in $G_{k,s}$. Recall that a length l simple path in $G_{k,s}$ is equivalent to a (k, s)-RdB sequence of length $l + k - 1$. Therefore, $N(k, s) = \ell(G_{k,s}) + k - 1$.

The de Bruijn graph G_k is actually an Eulerian graph because each vertex has exactly two in-coming edges and two out-coming edges. This results in its longest path visiting each edge exactly once and has a length of 2^k. However, (k, s)-RdB graph $G_{k,s}$ doesn't have the same property since the in-degree and out-degree of each vertex can be one or two. Thus, a simple path that visits all edges of the graph may not exist.

To overcome this issue, the upper bound $\mathbb{U}(k, s)$ for the length of the longest simple path is first determined in this section. Then, $\mathcal{U}(k, s) = \mathbb{U}(k, s) + k - 1$ is the upper bound for the length of longest (k, s)-RdB sequence. This work later proves that such bound can be achieved by proposing an efficient encoder returning a sequence of length $\mathcal{U}(k, s)$ in section 4.3.1. Hence, it's sufficient to conclude that the upper bound $\mathbb{U}(k, s)$ is also the length of the longest simple path. In other words, $\ell(G_{k,s}) = \mathbb{U}(k, s)$, and $N(k, s) = \mathcal{U}(k, s)$

Before deriving the explicit formula of maximal length, it's necessary to analyze more meticulously the in-degree and out-degree of all vertices. Given $0 \leqslant i, j \leqslant s$, define:

$$V_{i,j}^{k-1,s} = \Big\{ \boldsymbol{x} : \boldsymbol{x} \in V^{k-1,s}, x[1, i+1] = 0^i 1, \\ x[k-1-j, k-1] = 10^j \Big\}$$

That is, $V_{i,j}^{k-1,s}$ is the set of all vertices in $G_{k,s}$ satisfying the first $i+1$ letters are $(0, 0, \ldots, 0, 1)$ and the last $j+1$ letters are $(1, 0, 0, \ldots, 0)$. Lemma 4 summaries the properties of $V_{i,j}^{k-1,s}$ the helps finding $\mathbb{U}(k, s)$.

Lemma 4 (Properties of $V_{i,j}^{k-1,s}$).

1. $\left|V_{i,j}^{k-1,s}\right| = |W(k-i-j-3, s)|$.
2. $\sum_{0 \leqslant i,j \leqslant s} \left|V_{i,j}^{k-1,s}\right| = \left|V^{k-1,s}\right|$ $(= |W(k-1, s)|)$.
3. The in-degree and out-degree of each vertex in $V_{s,S}^{k-1,s}$ (if $V_{s,s}^{k-1,s} \neq \emptyset$) is exactly 1.
4. The in-degree and out-degree of each vertex in $V_{i,j}^{k-1,s}$ is exactly 1 for all $0 \leqslant i, j \leqslant s-1$.
5. For each vertex in $V_{s,i}^{k-1,s}$ $(0 \leqslant i \leqslant s-1)$, their in-degree is exactly one and their out-degree is exactly two.
6. For each vertex in $V_{i,s}^{k-1,s}$ $(0 \leqslant i \leqslant s-1)$, their in-degree is exactly two and their out-degree is exactly one.

Proof.

1. For each element $x \in V_{i,j}^{k-1,s}$, its subsequence, $x[i+1, k-2-j]$, can be any sequence of length $k-i-j-3$ such that more than s consecutive letter 0's is forbidden. Hence $x[i+1, k-2-j] \in W(k-i-j-3, s)$. Reversely, given an arbitrary sequence $y \in W(k-i-j-3, s)$, the string $0^i 1 y 1 0^j$ is a sequence in $V_{i,j}^{k-1,s}$. It comes to the conclusion that there is a bijection from $V_{i,j}^{k-1,s}$ to $W(k-i-j-3, s)$. In other word, $\left|V_{i,j}^{k-1,s}\right| = |W(k-i-j-3, s)|$.
2. Since $V_{i,j}^{k-1,s} \cup V_{i',j'}^{k-1,s} = \emptyset$ with $(i, j) \neq (i', j')$, and i, j cannot exceed s, thus, $\sum_{0 \leqslant i,j \leqslant s} \left|V_{i,j}^{k-1,s}\right| = \left|V^{k-1,s}\right|$.
3. The properties from (3) to (6) can be deduced directly by considering the prefix and suffix of each element in those sets.

□

Theorem 1 (Longest simple path). Let $C = \min(s-1, k-s-2)$. The length of the longest path in $G_{k,s}$, $\ell(G_{k,s})$, is equal to $\mathbb{U}(k,s)$, where:

$$\mathbb{U}(k,s) = |W(k,s)| - \left(\sum_{i=0}^{C} |W(k-i-s-3,s)| - s\right) \tag{4.1}$$

As mentioned above, proof of theorem 1 is divided into 2 parts. While the first one claims $\ell(G_{k,s}) \leqslant \mathbb{U}(k,s)$, the second one shows that there exists a sequence can achieve the length of $\mathbb{U}(k,s) + k - 1$. This section provides the proof of the first part (lemma 5). Proof for the second part is available in section 4.3.

Lemma 5. The longest simple path in $G_{k,s}$'s length cannot exceed $\mathbb{U}(k,s)$, that is, $\ell(k,s) \leqslant \mathbb{U}(k,s)$.

The following definitions and claims are essential to prove lemma 5.

Definition 8 (Balance and unbalanced vertex)**.** A vertex with the quantity of incoming edge equal to the quantity of out-coming edge is called a balanced vertex. A vertex is left-unbalanced if it has 2 edges coming in and 1 edges coming out. Reversely, a vertex is right-unbalanced if it has 1 edge coming in and 2 edges coming out.

Recall that a path is defined to be a sequence of edges. A vertex v is said to be (lying) in or belong to a path $\mathcal{P}$, denoted by $v \in \mathscr{P}$, if v has edges in $\mathcal{P}$. It's also fair to say that $\mathcal{P}$ goes through v. Besides, v is called the end (or the start) vertex of $\mathcal{P}$ if its last (first) edge ends (begins) at v.

Suppose $\mathscr{P}$ to be a longest simple path in $G_{k,s}$. In other word, $\mathscr{P}$ achieves the length $\ell(G_{k,s})$. Some observations about $\mathscr{P}$ are given in the following claims.

Claim 1. All the vertices in $\mathscr{P}$, if not a start or end vertex, must have the number of in-edges and out-edges equal in $\mathscr{P}$.

Proof. Let v be a vertex in $\mathscr{P}$, v is neither start nor end vertex. Then whenever $\mathscr{P}$ comes to v by an in-edge, it must go out of v by an out-edge. So the claim 1 is true. □

Claim 2. Every balance vertex in $\mathscr{P}$ has their quantity of in-edges and out-edges in $\mathscr{P}$ equal, even if one of them is the start or end vertex.

Proof. Let v be a balanced vertex and $\mathscr{P}$ goes through v. If v is neither end nor start vertex, by claim 2, v has the number of in-edges and out-edges in $\mathscr{P}$ equal.

Without loss of generality, assume that v is the start vertex. This results in the number of out-edges of v being equal or has 1 edges more than its number of out-edges. If these two quantities are equal, the proof is done. Otherwise, denote e' to be one of v's in-edges not belonging to $\mathscr{P}$. Then the path $\mathscr{P}_1 = \{e'\} \cup \mathscr{P}$ beginning at e' is a proper simple path RdB graph, but longer than $\mathscr{P}$, which contradicts to the assumption about the longest property of $\mathscr{P}$. □

Claim 3. Let $u = 0^s 1x1_t 0^j$ be a right-unbalanced vertex. Then the shortest path going from u to an arbitrary left-unbalanced vertex lengthen $s - j$.

Proof. A left-unbalanced vertex is represented by a sequence whose suffix of length s is filled by 0. Hence, a path from u to a left-unbalanced vertex must contain at least $s - j$ edges labeled 0. In fact, a path of length $s - j$ connecting u to a left-unbalanced vertex exists, which is the path of all 0 labeled edges. This path comes from u to $0^{s-j} 1x1_t 0^s$. □

Proof for lemma 5. Define $1x1_t$ to be a sequence of length t such that start and end letters are both 1. Let

$$\mathscr{L} = \bigcup_{j=0}^{C} \left\{ \boldsymbol{u} : \boldsymbol{u} = 0^j 1x1_t 0^s, s + t + j = k - 1 \right\}$$

be the set of all left-unbalanced vertices

Let $\boldsymbol{v} = 0^j 1x1_t 0^s$ be an arbitrary vertex in $\mathscr{L}$ such that $\boldsymbol{v}$ isn't the end-vertex of path $\mathscr{P}$.

1. It can be proved that there is a path $\mathcal{P}_v$ of length $s - j$ satisfying all of its edges not lying in $\mathscr{P}$ and its end-vertex is $\boldsymbol{v}$. The path $\mathcal{P}_v$ is constructed backwardly as follows:

   Starts with $\mathcal{P}_v = \emptyset$. As $\boldsymbol{v}$ has 2 in-edges and 1 out-edges, there's at least 1 in-edge e_v of $\boldsymbol{v}$ not lying in $\mathscr{P}$. Of course, e_v's label is 0. Adds e_v to $\mathscr{P}_v$. Let $a_1 0^j 1x1_t 0^{s-1} = \pi(e_v)$ be the initial state of e_v. If $\pi(e_v)$ is a balance vertex, then by the claim 2, there also must be at least 1 in-edge of $\pi(e_v)$ not lying in $\mathscr{P}$. Continue adding this edge to the head of $\mathcal{P}_v$. Denote $a_2 a_1 0^j 1x10^{s-2}$ to be the initial state of this edge. Now, $\mathcal{P}_v$ can be represented as below:

$$\mathcal{P}_v = \left[\left(a_2 a_1 0^j 1x1_t 0^{s-2}, a_1 0^j 1x1_t 0^{s-1} \right), \left(a_1 0^j 1x1_t 0^{s-1}, 0^j 1x1_t 0^s \right) \right]$$

   Assume inductively that:

$$
\begin{aligned}
\mathcal{P}_v = \Big[& \Big(a_l \cdots a_2 a_1 0^j 1x1_t 0^{s-l}, a_{l-1} \cdots a_1 0^j 1x1_t 0^{s-(l-1)}\Big), \cdots, \\
& \big(a_2 a_1 0^j 1x1_t 0^{s-2}, a_1 0^{j-1} 1x1_t 0^{s-1}\big), \\
& \big(a_1 0^{j-1} 1x1_t 0^{s-1}, 0^j 1x1_t 0^s\big) \Big]
\end{aligned}
$$

with $l \leqslant s-j$.

If $l = s-j$, the proof is done. Otherwise, it's obvious that

$$a_l \cdots a_2 a_1 0^j 1x1_t 0^{s-l}$$

is balance, hence, by claim 2, it also has at least 1 in-edge not lying in $\mathscr{P}$, and one can continue adding such edge to the head of $\mathcal{P}_v$.

2. It can be shown that for each $u, v \in \mathscr{L}$ such that neither u nor v is the end vertex of the last edge of $\mathscr{P}$, paths $\mathcal{P}_u$ and $\mathcal{P}_v$ are edge-disjoint.

   Represent $\mathcal{P}_u = [e_{u,1}, e_{u,2}, \cdots, e_{u,i}]$, $\mathcal{P}_v = [e_{v,1}, e_{v,2}, \cdots, e_{v,j}]$. Assume to the contrary that $\exists t \leqslant i, l \leqslant j$ satisfying $e_{u,t} = e_{v,l}$. Without loss of generalization, we suppose that $i-t \leqslant j-l$. As all edges in $\mathcal{P}_u$ and $\mathcal{P}_v$ are labeled 0, we must have $e_{u,t+1} = e_{v,l+1}, \cdots, e_{u,i} = e_{v,l+(i-t)}$.

   If $l+i-t = j$, we have $\mathcal{P}_u$ and $\mathcal{P}_v$ have the same last edge but different terminal states, which is impossible. Otherwise, the path $e_{v,l+i-t+1}, \cdots, e_{v,j}$ is the path connecting u to v. But the length of this path is smaller than s, which is also absurd.

Summary, for each vertex in $\mathscr{L}$ such that v isn't the end-vertex of the last edge in path $\mathscr{P}$, there is a path $\mathcal{P}_v$ of length $s-j$ satisfying all of its edges not lying in $\mathscr{P}$ and takes v to be its end-vertex. Moreover, all these such paths are edge-disjoint, and there can be only 1 end-vertex of the last edge in path $\ell(G_{k,s})$, the total edges of all such path $\mathcal{P}_u$ is:

$$\begin{aligned}
&\sum_{i=0, i\neq j}^{C} (s-i)|W(k-s-i-3,s)| \\
&\qquad +(s-j)(|W(k-s-j-3,s)|-1) \\
&= \sum_{i=0}^{C}(s-i)|W(k-s-i-3,s)|-(s-j) \\
&\geqslant \sum_{i=0}^{C}(s-i)|W(k-s-i-3,s)|-s
\end{aligned}$$

This means at least $\sum_{i=0}^{C}(s-i)|W(k-s-i-3,s)|-s$ edges not lying in $\mathscr{P}$. As the number of edges in (k,s)-RdB is $|W(k,s)|$, the length of the longest path $\mathscr{P}$ can not exceed :

$$|W(k,s)|-(\sum_{i=0}^{C}(s-i)|W(k-s-i-3,s)|-s)$$

This concludes our lemma. □

Define $\mathcal{U}(k,s) = \mathbb{U}(k,s)+(k-1)$, then $\mathcal{U}(k,s)$ is length of the longest (k,s)-RdB sequence.

4.2 Rate and maximal asymptotic rate of (k,s)-RdB sequence

For every (k,s)-RdB sequence, their rate is defined as follows:

Definition 9 (Rate)**.** Denote $R(\boldsymbol{x}_{k,s})$ to be the rate of a (k,s)-RdB sequence $\boldsymbol{x}_{k,s}$, then:

$$R(\boldsymbol{x}_{k,s}) = \frac{\log\left(|\boldsymbol{x}_{k,s}|\right)}{k} \tag{4.2}$$

where the base of logarithm function is $|\Sigma| = 2$.

The rates of sequences are the proportion of the data stream that is useful (non-redundant), which tell how much useful information is transmitted. The sequence's rate actually originates from the information rate. Recall that, in definition 3, the rate R_C of a code C consisting of length n q-ary sequences is $R_C = \dfrac{\log_q(|C|)}{n}$. If a (k,s)-RdB sequence $x_{k,s}$ is considered to be a code $C_{k,s}$, each of its size k windows is treated as a codeword. The size of $C_{k,s}$ is exactly the length of $x_{k,s}$ minus k, but the offset k can be omitted under log calculation. Hence:

$$R_{C_{k,s}} = \frac{\log_q \left(\left|C_{k,s}\right|\right)}{n} = \frac{\log(\left|\boldsymbol{x}_{k,s}\right|)}{k} = R(x_{k,s})$$

. That is to say, $R(\boldsymbol{x}_{k,s})$ is eventually a kind of information rate.

Note that, a high rate is usually preferred. Accordingly, with each given pair (k, s), the maximal rate of all (k, s)-RdB sequence is defined.

Definition 10 (Maximal rate)**.** Given k and s, the maximal rate $R_{k,s}$ of all (k, s)-RdB sequences is:

$$R_{k,s} = \frac{\log(N(k, s))}{k} \tag{4.3}$$

The maximal asymptotic rate is concerned in case k appears to be very large.

Definition 11 (Maximal asymptotic rate)**.** Denote R_s to be the maximal asymptotic rate of (k, s)-RdB sequences, then:

$$R_s = \lim_{k\to\infty} \frac{\log(N(k, s))}{k} \tag{4.4}$$

Having the explicit formula of $N(k, s)$ determined makes it easier to calculate the maximal asymptotic rate of the (k, s)-RdB sequence. The following equation is a direct consequence of theorem 1:

$$N(k, s) = |W(k, s)| - \left(\sum_{i=0}^{C} |W(k-i-s-3, s)| - s\right) + k - 1 \tag{4.5}$$

Theorem 2 below shows that the rate of $(k, 1)$-RdB sequence is better than the rate of HdB sequence. Therefore, it's able to use $(k, 1)$-RdB sequence for the system in [1] instead to increase the rate, speed of encoding, and decoding of the transmitted signals.

Theorem 2.

$$R_1 = 0.6942 \tag{4.6}$$

Proof. Substitute $s = 1$ into expression 4.5 gives $N(k, 1) = |W(k, 1)| - |W(k-4, 1)|$, and recall that $\{|W(k, 1)|\}$ is a Fibonacci sequence with:

$$|W(k, 1)| = \frac{\phi^{k+2} + \varphi^{k+2}}{\sqrt{5}}$$

. where $\phi = \dfrac{1+\sqrt{5}}{2}$ and $\varphi = \dfrac{1-\sqrt{5}}{2}$. Here, observe that $\phi\varphi = -1$, so $\phi^2\varphi^2 = 1$, consequently, it's able to write:

$$\begin{cases} \phi^{-2} = \varphi^2 \\ \varphi^{-2} = \phi^2 \end{cases}$$

As a result:

$$\begin{aligned} |W(k,1)| - |W(k-4,1)| &= \frac{\phi^{k+2} + \varphi^{k+2}}{\sqrt{5}} - \frac{\phi^{k-2} + \varphi^{k-2}}{\sqrt{5}} \\ &= \frac{\phi^{k+2} + \varphi^{k+2} - \phi^k\varphi^2 - \varphi^k\phi^2}{\sqrt{5}} \\ &= \left(\phi^k + \varphi^k\right).\frac{\phi^2 - \varphi^2}{\sqrt{5}} \end{aligned}$$

And hence:

$$\begin{aligned} R_1 &= \lim_{k\to\infty} \frac{\log(N(k,1))}{k} \\ &= \lim_{k\to\infty} \frac{\log\left(\left(\phi^k + \varphi^k\right).\frac{\phi^2 - \varphi^2}{\sqrt{5}} + k\right)}{k} \\ &= \log(\phi) \approx 0.6942 \end{aligned}$$

□

Note that ϕ is the largest root of equation $x^2 - x - 1 = 0$. Based on that observation, the prediction here is that if ω is the root of equation:

$$x^{s+1} - x^s - \ldots - x - 1 = 0 \tag{4.7}$$

satisfying $|\omega|$ is the largest, the maximal asymptotic rate $R_s = \log(|\omega|)$. Fortunately, this is proved to be true in the following theorem, the generalization of theorem 2.

Theorem3 (Maximal asymptotic rate of RdB sequence)**.** Let s be a positive integer. Then:

$$R_s = \log(|\omega|) \tag{4.8}$$

Proof. The root ω is actually a Pisot number. More particularly, ω is the only positive roots of 4.7 lying in the interval $(1,2)$, the other roots are in the open disk $\{z \in \mathbb{C}, |z| < 1\}$. Also, note that $x^{s+1} - x^s - \ldots - x - 1 = 0$ is the characteristic equation of $W(k,s)$. Therefore, with k big enough, $|W(k,s)|$ can be estimated as follows:

$$|W(k,s)| \approx a.|\omega|^k$$

whereas a is a positive constant. Thus, from expression 4.5:

$$\begin{aligned}
N(k,s) &= |W(k,s)| - \left(\sum_{i=0}^{C} |W(k-i-s-3,s)| - s\right) + k - 1 \\
&\approx \sum_{t=k-s}^{k} (k-t+1)a|\omega|^{t-2} + s + k - 1 \\
&= |\omega|^{k-s-2}\left(\sum_{t=k-s}^{k} a(k-t+1)|\omega|^{t-k+s} + \frac{s+k-1}{|\omega|^{k-s-2}}\right) \\
&= |\omega|^{k-s-2}\left(\sum_{i=0}^{s} a(i+1)|\omega|^{s-1} + \frac{s+k-1}{|\omega|^{k-s-2}}\right)
\end{aligned}$$

This results in:

$$\begin{aligned}
R_k &= \lim_{k\to\infty} \frac{\log\left(|\omega|^{k-s-2}\left(\sum_{i=0}^{s} a(i+1)|\omega|^{s-1} + \frac{s+k-1}{|\omega|^{k-s-2}}\right)\right)}{k} \\
&= \lim_{k\to\infty} \frac{(k-s-2)\log(|\omega|)}{k} + \lim_{k\to\infty} \frac{\log\left(\sum_{i=0}^{s} a(i+1)|\omega|^{s-i} + \frac{s+k-1}{|\omega|^{k-s-2}}\right)}{k} \\
&= \log(|\omega|)
\end{aligned}$$

□

Table 4.1 lists the first 8 values of $\log(\omega)$ with respect to s from 1 to 8.

s	1	2	3	4	5	6	7	8
$\log(\omega)$	0.6942	0.8791	0.9468	0.9752	0.9881	0.9942	0.9971	0.9986

Table 4.1: Values of $\log(\omega)$ with s from 1 to 8

To demonstrate the convergence of $R_{k,s}$ to $\log(|\omega|)$, table 4.2 lists the first values of $R_{k,s}$ with $k = \overline{2,4}$ and then increase to the large $k = \overline{71,74}$.

4.3 Construction of RdB sequence

This section presents a construction of a (k,s)-RdB sequence $\boldsymbol{c}_{k,s}$. Furthermore, given a substring of length k of the sequence $\boldsymbol{c}_{k,s}$, a fast decoding algorithm to determine the location of the given substring is provided. The complexity of the decoding algorithm is sub-linear with respect to the length of $\boldsymbol{c}_{k,s}$. There are some

k s	2	3	4	...	72	73	74	$\log(\omega)$
1	1	0.9358	0.8649	...	0.6942	0.6942	0.6942	0.6942
2	1.1610	1.0566	1	...	0.8791	0.8791	0.8791	0.8791
3	1.1610	1.1073	1.0425	...	0.9468	0.9468	0.9468	0.9468
4	1.1610	1.1073	1.0620	...	0.9752	0.9752	0.9752	0.9752
5	1.1610	1.1073	1.0620	...	0.9881	0.9881	0.9881	0.9881
6	1.1610	1.1073	1.0620	...	0.9942	0.9942	0.9942	0.9942
7	1.1610	1.1073	1.0620	...	0.9971	0.9971	0.9971	0.9971
8	1.1610	1.1073	1.0620	...	0.9986	0.9986	0.9986	0.9986

Table 4.2: The convergence of $R_{k,s}$
with s from 1 to 8

classical de Bruijn sequences with sub-linear decoding algorithm [21]–[23]. This work uses the minimal de Bruijn sequence, constructed in [23], [25], and some special properties of Lyndon words to construct a (k, s)-RdB.

Definition 12 (Lyndon words). A sequence w is a Lyndon word if and only if it's strictly smaller than all of its rotation.

For more intelligible, several Lyndon words and non-Lyndon words are provided in example 4.1.

Example 4.1 (Lyndon words)**.** The word 00101 is a Lyndon word since it is smaller than all of its rotations: 01010, 10100, 01001, 10010. The word 01100 is not a Lyndon word, since one of its rotations, 00011, is smaller than it. The word 011011 is also not a Lyndon word, as its cyclic rotation by 3 letters is equal to it.

4.3.1 Encoder for a(k, s)-RdB sequence

In 1978, Fredricksen and Maiorana [25] proposed the FKM algorithm to efficiently construct the lexicographically minimal de Bruijn sequence, which is later called the granddaddy sequence by Knuth [24]. The algorithm was based on their finding of the connection between the de Bruijn sequence and Lyndon words.

Lemma 6 ([25])**.** The lexicographically minimal de Bruijn sequence of order k (k-MdB) is the concatenation of all Lyndon words whose length is a divisor of k in the lexicographical order.

For example, the 6-MdB sequence is decomposed into Lyndon words as follows:

Example 4.2 (Decomposition of 6-MdB sequence)**.** Recall that 6-MdB sequence is already given in example 2.3, which is:

$$0000001000011000101000111001001011001101001111010101110110111111$$

As stated in lemma 6, it can be decomposed in lexicographically order into Lyndon words listed follows:

0

000001

000011

000101

000111

001

001011

001101

001111

01

010111

011

011111

1

This thesis observes that it is able to append a prefix to a suffix of k-MdB to obtain a (k, s)-RdB sequence, i.e., in the cycle representing k-MdB, there are arcs representing (k, s)-RdB sequences. To illustrate this idea, figure 4.1 gives an example for $k = 6$ and $s = 2$, where the blue arc and the red arc indicate the prefix and the suffix respectively. Any substring of the concatenation of the prefix and suffix is a $(6, 2)$-RdB sequence.

More precisely, the construction is described as follows:

Construction 1. Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ be the k-MdB constructed from Lemma 6 and $u_{k,s} = 0^{s+1}1^{k-s-1}$ be a Lyndon word of length k satisfying the first $s+1$ letters are all 0's and the last $k-s-1$ letters are all 1's. By the intrinsic property of de Bruijn sequences, there exists only one index i such that $\boldsymbol{u}_{k,s} = \boldsymbol{x}[i, i+k-1] = (x_i, \ldots, x_{i+k-1})$. Denote the word $\boldsymbol{c}_{k,s} = (x_{i+1}, \ldots, x_n, 0, 0, \ldots, 0)$, obtained by adding s letters 0 to the end of the suffix of $\boldsymbol{x}$, from index $i+1$ to the end. Theorem 4 below claims that $\boldsymbol{c}_{k,s}$ is a (k, s)-RdB sequence.

This thesis later proves that $\boldsymbol{c}_{k,s}$ is even the longest (k, s)-RdB sequence.

Denote Ł$_n$ to be the set of all Lyndon words of length n, and Ł$^{(n)} = \cup_{d|n}$Ł$_d$ to

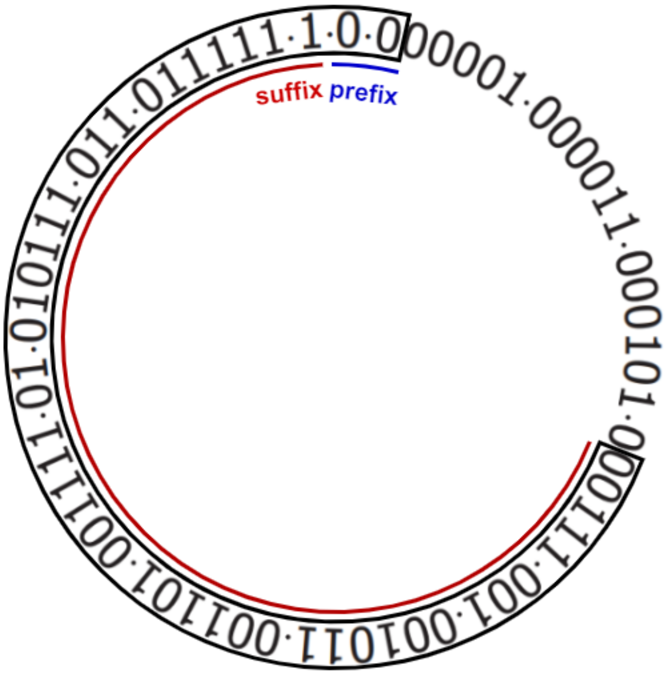

Figure 4.1: Example for $k = 6,\ s = 2$. In the circle of 6-MdB, an arbitrary substring of the concatenation of suffix and prefix in the picture is a $(6, 2)$-RdB sequence.

be the set of all Lyndon words whose lengths are divisors of n. The formal encoder to construct a (k, s)-RdB sequence is given in algorithm 1.

Algorithm 1: Encode (k,s)-RLL dBs

```
Input  : $k$, and descending ordered set Ł$^{(n)}$.
Output: $(k, s)$-RLL dBs
w ←emptystring
for $\lambda$ ∈Ł$^{(n)}$ do
    w.$prepend(\lambda)$
    if $\lambda == 0^{s+1}1^{k-s-1}$ then
        /* remove the first letter of $w$, which is 0,
           and add $s$ letters 0 to the end              */
        w = w$[2, \ell]0^s$
        break
return w
```

The set Ł$^{(n)}$ in lexicographically order can be generated in constant amortized time by applying FKM algorithm (analyzed in [27]), or by another algorithm developed by Duval in [65]. In algorithm 1, the most consuming time step is to produce the set Ł$^{(n)}$, and hence, its complexity is the complexity of the algorithm used to bring out Ł$^{(n)}$.

Here presents an example for $k = 6$ and $s = 2$.

Example 4.3 (Construction of $(6,2)$-RdB sequence)**.** The suffix:

$$00111\ 001011\ 001101\ 001111\ 01\ 010111\ 011\ 011111\ 1$$

is taken from the granddaddy of order 6 given above. Adding 2 letter 0 to the end of it obtains:

$$00111\ 00\ 001011\ 001101\ 001111\ 01\ 010111\ 011\ 011111\ 1\ 00$$

which is indeed a $(6,2)$-RdB sequence.

Now, theorem 4 proves that the Construction 1 always return a (k,s)-RdB sequence.

Theorem 4. The sequence $\boldsymbol{c}_{k,s}$ obtained from Construction 1 is a (k,s)-RdB sequence.

Proof. First, it's necessary to show that each substring of length k appears at most once in $\boldsymbol{c}_{k,s}$. Note that the granddaddy sequence $\boldsymbol{x}$ obtained from Lemma 6 is a cyclic de Bruijn sequence, and $\boldsymbol{c}_{k,s}$ is actually a substring of $\boldsymbol{x}$. Hence, $\boldsymbol{c}_{k,s}$ just contains each substring of size k at most once.

Now, claiming that $\boldsymbol{c}_{k,s}$ doesn't contain any patterns 0^{s+1} will complete the theorem. This can be proved by considering the property of Lyndon words. It's obvious to see that $\boldsymbol{u}_{k,s} = 0^{s+1}1^{k-s-1}$ is the largest Lyndon word containing $s+1$ consecutive symbols 0. Hence, every Lyndon word decomposed from $\boldsymbol{c}_{k,s}$ doesn't take 0^{s+1} as a substring. Moreover, the last symbol of all Lyndon words but 0 is 1. Therefore, 0^{s+1} will not appear in the combination of Lyndon words larger than $\boldsymbol{u}_{k,s}$. Adding 0^s1^{k-s-1} to the beginning and s letters 0 to the end of this combination resulting in $\boldsymbol{c}_{k,s}$ won't change this property. So, it's able to conclude that $\boldsymbol{c}_{k,s}$ is indeed a (k,s)-RdB sequence. □

4.3.2 Decoder for a (k,s)-RdB sequence

In [1], the Hybrid de Bruijn sequence of order k after being received needs to be decoded for correcting errors. More particularly, it's necessary to indicate the exact location of an arbitrary sequence of length k in the Hybrid de Bruijn sequence. To do that, they proposed to use a look-up table, which is an exponential complex method.

Similarly, it's essential for this work to decode $\boldsymbol{c}_{k,s}$. In 2016, Kociumaka, Radoszewski, and Rytter presented the first sub-linear decoding algorithm $\mathcal{D}_{KRR}$ for the minimal de Bruijn sequences. And since $\boldsymbol{c}_{k,s}$ is a substring of a minimal de

Bruijn sequence, it's able to modify $\mathcal{D}_{KRR}$ to decode $\boldsymbol{c}_{k,s}$ in sub-linear time.

Let $i = \mathcal{D}_{KRR}(\boldsymbol{u}_{k,s})$ be the position of the word $\boldsymbol{u}_{k,s}$ in the granddaddy sequence of order k $\boldsymbol{x}$. Recall that, from Construction 1, we have $\boldsymbol{c}_{k,s} = (x_{i+1}, \ldots, x_n, 0^s)$. Thus, for each length k word v lying in $\boldsymbol{c}_{k,s}$, its location in $\boldsymbol{c}_{k,s}$ is to location of v in x minus j, unless they are of the form $1^j 0^{k-j}$ for all $1 \leqslant j \leqslant s$ which appear at the end of $\boldsymbol{c}_{k,s}$. The formal description of our decoding algorithm is shown in algorithm 2.

Algorithm 2: Decode (k,s)-RdB $\boldsymbol{c}_{\mathbf{k,s}}$

```
Input  : A word $\boldsymbol{v} = (v_1, \ldots, v_k)$ of length $k$
Output: a is the location of $\boldsymbol{v}$ in $\boldsymbol{c}_{\mathbf{k,s}}$
$i \leftarrow \mathcal{D}_{KRR}(\boldsymbol{u}_{k,s})$;
/* $\mathcal{D}_{KRR}$ is the decoder of the minimal de Bruijn
   sequence in [23]                                      */
if $\boldsymbol{v} = 1^j 0^{k-j}$, then
    return $n - i + 1 - (k - j)$;
else
    return $\mathcal{D}_{KRR}(\boldsymbol{v}) - i$;
```

4.3.3 The optimality of our construction

This section gives proof for the claim stated in section 4.1, that is, the encoder produces sequence $\boldsymbol{c}_{k,s}$ whose length equals to to upper bound $\mathcal{U}(k,s)$, and thus, $\boldsymbol{c}_{k,s}$ is the longest the (k,s)-RdB sequence. In order to do so, $\boldsymbol{c}_{k,s}$'s length, denoted by $\ell(\boldsymbol{c}_{k,s})$, is needed calculating first. It's then essential to show that $\ell(\boldsymbol{c}_{k,s})$ is equal to $\mathcal{U}(k,s)$ by some algebraic transformations.

Given a word $\boldsymbol{u}$, denote $\langle \boldsymbol{u} \rangle$ to be its minimal rotation. For instance, the minimal rotation of 010110 is 001011, or, the minimal rotation of 010101 is itself. For every word $\boldsymbol{v}$, we define:

$$S(\boldsymbol{v}) = \left\{ \boldsymbol{u} : \boldsymbol{u} \in \Sigma^{|\boldsymbol{v}|}, \langle \boldsymbol{u} \rangle \leqslant \boldsymbol{v} \right\}$$

to be the set of all sequence of length $|\boldsymbol{v}|$ satisfying their minimal rotation doesn't exceed $\boldsymbol{v}$. The following example 4.4 lists all element of $S(\boldsymbol{v})$ with $\boldsymbol{v} = 01101$.

Example 4.4 (Example of $S(\boldsymbol{v})$)**.** Given $\boldsymbol{v} = 01101$, all sequence of length $|\boldsymbol{v}| = 5$

whose minimal rotations are at most $\boldsymbol{v}$ is:

$$\begin{aligned} S(01101) = \big\{&00000, \\ &00001, 00010, 00100, 01000, 10000, \\ &00011, 00110, 01100, 11000, 10001, \\ &00111, 01110, 11100, 11001, 10011\big\} \end{aligned}$$

If $\boldsymbol{v}$ is a Lyndon word, Lemma 29 in [23] tells that the cardinality of $S(\boldsymbol{v})$, $|S(\boldsymbol{v})|$, equals to the length of the prefix of the granddaddy sequence $\boldsymbol{x}$, from the beginning to the sub-string $\boldsymbol{v}$. Recall that $\boldsymbol{u}_{k,s}$ is also a Lyndon word, one has:

$$\begin{aligned} \big|S(\boldsymbol{u}_{k,s})\big| &= 2^k - (\ell(\boldsymbol{c}_{k,s}) - (k-1) - s) \\ \Leftrightarrow \ell(\boldsymbol{c}_{k,s}) &= 2^k + (k-1) + s - \big|S(\boldsymbol{u}_{k,s})\big| \end{aligned} \tag{4.9}$$

This brings the idea determining $\ell(\boldsymbol{c}_{k,s})$ by computing the size of the set $S(\boldsymbol{u}_{k,s})$.

Lemma 7. Let $A_t = 2^{t-2}$ for all $t > 1$, $A_1 = 1$, and $M = \max(k-s, s+3)$. Then:

$$\big|S(\boldsymbol{u}_{k,s})\big| = 1 + \sum_{t=M}^{k} (k-t+1)\,|C(t-2,s)| + \sum_{t=1}^{k-s-1} A_t \tag{4.10}$$

Proof. Let i, j be two non-negative integers such that $i + j < k$, denote:

$$U_{i,j} = \big\{0^i 1 x 1_t 0^j \in S(\boldsymbol{u}_{k,s}) : i + j + t = k\big\}$$

to be the set of all words in $S(\boldsymbol{u}_{k,s})$ satisfying its prefix of length $i+1$ is 0^i1 and its suffix of length $j+1$ is 10^j.

Since $S(\boldsymbol{u}_{k,s})$ is the disjoint union of 0^k and all sets $U_{i,j}$ for $i, j \geqslant 0$ and $i+j < k$, we obtain:

$$\big|S(\boldsymbol{u}_{k,s})\big| = 1 + \sum_{i,j\geqslant 0; i+j\leqslant s} |U_{i,j}| + \sum_{i,j\geqslant 0; s<i+j<k} |U_{i,j}|\,. \tag{4.11}$$

If we fix $1 \leqslant t \leqslant k$, there are $k - t + 1$ pairs (i, j) such that $t = k - i - j$. If $i + j \leqslant s$ then the sub-string $1x1_t$ must contain $s + 1$ consecutive 0's (consequently, $k - i - 1 - (i + 2) + 1 \geqslant s + 1 \Rightarrow t = k - i - j \geqslant s + 3$), and therefore, $|U_{i,j}| =$

$|C(t-2,s)|$. Hence,

$$\mathcal{C}_1 = \sum_{\substack{i,j \geqslant 0;\\ i+j \leqslant s}} |U_{i,j}| = \sum_{t=M}^{k} (k-t+1)\,|C(t-2,s)|. \tag{4.12}$$

where $M = \max(s+3, k-s)$.

If $k > i+j > s$ then the sub-string $(x_{i+2}, \ldots, x_{k-j-1})$ can be any word of length $t-2$ and thus $|U_{i,j}| = A_t = 2^{t-2}$. Hence:

$$\mathcal{C}_2 = \sum_{\substack{i,j \geqslant 0;\\ s<i+j<k}} |U_{i,j}| = \sum_{t=1}^{k-s-1} (k-t+1)A_t. \tag{4.13}$$

From Equations (4.11), (4.12), (4.13), we get the result in Lemma 7. □

Combining the results from Lemma 7 and equation 4.9 gives:

$$\ell(\boldsymbol{c}_{k,s}) = 2^k + k + s - 2 - \mathcal{C}_1 - \mathcal{C}_2$$

where $\mathcal{C}_1$ and $\mathcal{C}_2$ are defined in Equation 4.12 and 4.13. It's now ready to prove the following lemma, which states that the proposed construction is optimal.

Lemma 8. The length the sequence $\boldsymbol{c}_{k,s}$ returned from Construction 1 is optimal, that is:

$$\ell(\boldsymbol{c}_{k,s}) = \mathcal{U}(k,s) \tag{4.14}$$

Proof. Equation 4.14 is equivalent to:

$$\begin{aligned} 2^k + k + s - 2 - \mathcal{C}_1 - \mathcal{C}_2 &= |W(k,s)| - \left(\sum_{i=0}^{C} |W(k-i-d-3,s)| - s\right) + (k-1) \\ \Leftrightarrow 2^k - (1 + \mathcal{C}_1 + \mathcal{C}_2) &= |W(k,s)| - \left(\sum_{i=0}^{C} |W(k-i-d-3,s)|\right) \end{aligned} \tag{4.15}$$

First, the value of C and M is necessarily explicated by considering the relation between k and s. In short, there are 3 following cases:

$$\begin{cases} M = k-s,\ C = s-1 \text{ if } s+3 \leqslant k-s \\ M = s+3,\ C = s-1 \text{ if } k = 2s+2, s < k-1 \\ M = s+2,\ C = k-s-2 \text{ if } k \leqslant 2s+1 \end{cases}$$

Case 1: $M = k - s,\ C = s - 1$ when $s + 3 \leqslant k - s$. The equation needing to be proved 4.15 becomes:

$$\begin{aligned}
&|W(k,s)| - \left(\sum_{i=0}^{s-1}(s-i)\,|W(k-s-i-3,s)|\right)\\
&= 2^k - \left(1 + \sum_{t=k-s}^{k}(k-t+1)\,|C(t-2,s)| + \sum_{t=1}^{k-(s+1)}(k-t+1)A_t\right)
\end{aligned}$$

In the right hand side, recall that $|C(t-2,s)| = 2^{t-2} - |W(t-2,s)|$, so:

$$\begin{aligned}
\text{RHS} &= 2^k - \left(1 + \sum_{t=k-s}^{k}(k-t+1)\,|C(t-2,s)| + \sum_{t=1}^{k-(s+1)}(k-t+1)A_t\right)\\
&= 2^k - \left(1 + \sum_{t=1}^{k}(k-t+1)A_t - \sum_{t=k-s}^{k}(k-t+1)\,|W(t-2,s)|\right)\\
&= 2^k - \left(2^k - \sum_{t=k-s}^{k}(k-t+1)\,|W(t-2,s)|\right)\\
&\quad \text{(the term } 1 + \sum_{t=1}^{k}(k-t+1)A_t \text{ can be easily shown to be equal to } 2^k\text{)}\\
&= \sum_{t=k-s}^{k}(k-t+1)\,|W(t-2,s)|
\end{aligned}$$

Thus:

$$\begin{aligned}
&\text{LHS} = \text{RHS}\\
\Leftrightarrow\ & |W(k,s)| - \left(\sum_{i=0}^{s-1}(s-i)\,|W(k-s-i-3,s)|\right) = \sum_{t=k-s}^{k}(k-t+1)\,|W(t-2,s)|\\
\Leftrightarrow\ & |W(k,s)| = \sum_{t=2}^{s+2}(t-1)\,|W(k-t,s)| + \sum_{t=s+3}^{2s+2}(2s+3-t)\,|W(k-t,s)|\ (*)
\end{aligned}$$

Recall that:

$$\begin{aligned}
|W(k,s)| &= \sum_{i=1}^{s+1}|W(k-i,s)|\ \forall k > s\\
&= |W(k-1,s)| + |W(k-2,s)| + \cdots + |W(k-s-1,s)|\ \forall k > s
\end{aligned}$$

Therefore, when $k \geqslant 2s+3$, the following system of equations is obtained:

$$\begin{cases} |W(k-1,s)| = & |W(k-2,s)| + |W(k-3,s)| + \cdots + |W(k-s-2,s)| \\ |W(k-2,s)| = & |W(k-3,s)| + |W(k-4,s)| + \cdots + |W(k-s-3,s)| \\ \quad \cdots & \\ |W(k-s-1,s)| = & |W(k-s-2,s)| + |W(k-s-3,s)| + \cdots + |W(k-2s-2,s)| \end{cases}$$

Adding side by side the above equations results in the equation $(*)$, which is needed to be verified.

Case 2: $M = s+3,\ C = s-1$ when $k = 2s+2, s < k-1$. The LHS is the same as in the first case, meanwhile, the RHS is :

$$\begin{aligned} \text{RHS} =& 2^k - \left(1 + \sum_{t=s+3}^{2s+2} (k-t+1)\,|C(t-2,s)| + \sum_{t=1}^{k-(s+1)} (k-t+1)A_t\right) \\ =& 2^k - \left(1 + \sum_{t=1}^{2s+2} A_t - \sum_{t=s+3}^{2s+2} (k-t+1)\,|W(t-2,s)| - (s+1)2^s\right) \\ =& \sum_{t=s+2}^{2s+2} (k-t+1)\,|W(t-2,s)| \\ =& \sum_{t=k-s}^{k} (k-t+1)\,|W(t-2,s)| \end{aligned}$$

what's left to be proved is similar to the first case.

Case 3: $M = s+3,\ C = k-s-2$ when $s+2 \leqslant k \leqslant 2s+1$. Again, the RHS is $\text{RHS} = \sum_{t=k-s}^{k}(k-t+1)\,|W(t-2,s)|$, and the LHS is:

$$|W(k,s)| - \left(\sum_{i=0}^{k-s-2} (s-i)\,|W(k-s-i-3,s)|\right)$$

Therefore:

$$\begin{aligned} &\text{LHS} = \text{RHS} \\ \Leftrightarrow |W(k,s)| = \sum_{t=2}^{s+2} &(t-1)\,|W(k-t,s)| + \sum_{t=s+3}^{k+1} (2s+3-t)\,|W(k-t,s)|\,(**) \end{aligned}$$

The situation is quite similar to the first case, and recall that : $|W(n,s)| = 2^s\ \forall 0 \leqslant n \leqslant s$, $|W(-1,s)| = 1$. Thus:

$$\begin{cases} |W(k-1,s)| = |W(k-2,s)| + |W(k-3,s)| + \ldots + |W(k-s-2,s)| \\ |W(k-2,s)| = |W(k-3,s)| + |W(k-4,s)| + \ldots + |W(k-s-3,s)| \\ \quad \ldots \\ |W(k-t,s)| = |W(k-t-1,s)| + |W(k-t-2,s)| + \ldots + |W(k-t-s-1,s)|, \\ \text{with } k-t=s+1 \\ |W(k-t-1,s)| = |W(k-t-2,s)| + |W(k-t-3,s)| + \ldots + |W(0,s)| + |W(-1,s)| \\ \quad \ldots \\ |W(k-s-1,s)| = |W(k-s-2,s)| + |W(k-s-3,s)| + \ldots + |W(-1,s)| \end{cases}$$

Once again, adding side by side the above equations gives $(**)$.

In conclusion, in all 3 cases, the correctness of equation 4.15 is verified, hence, Lemma 8 is proved. □

CONCLUSIONS

Summary

In this thesis, Run length limited de Bruijn sequences are introduced and studied to replace Hybrid de Bruijn sequence in dBTS system. Compare to HdB sequences, RdB sequences not only have a higher rate but are also more general and adaptive. The main results of this thesis include the explicit formula of the maximal length and maximal asymptotic rate of RdB sequences. To achieve such length and rate, an encoding algorithm is presented. This thesis also provides proof of the optimality of the encoder. To locate the position of a proper substring in the whole encoded sequence, a decoding algorithm is proposed based on the decoder of the granddaddy sequence. The encoder and decoder are both based on state-of-the-art algorithms. The encoder's complexity is constant amortized time per symbol, and the decoder's complexity is sub-linear with respect to the length of the RdB sequence.

Future works

In future work, it's critical to analyze deeper about the RdB sequence under some other constraints like weight constraint or local constraint. The current results right now just focus on the alphabet of size 2. The study of the more general alphabet will raise many more questions in combinatorics and algorithm. Especially, results for the alphabet of size 4 will be valuable in the research of DNA storage as well as DNA sequencing, a very interesting field recently.

Publications

1. Yeow Meng Chee, Duc Tu Dao, **Tien Long Nguyen**, Duy Hoang Ta, Van Khu Vu. ”Run Length Limited de Bruijn Sequences for Quantum Communications”, The 2022 IEEE International Symposium on Information Theory.
2. Tran Ba Trung, Lijun Chang, **Nguyen Tien Long**, Kai Yao, Huynh Thi Thanh Binh. ”Verification-Free Approaches to Efficient Locally Densest Subgraph Discovery”, The 39th IEEE International Conference on Data Engineering.

REFERENCE

[1] P. Zhang, D. K. Oi, D. Lowndes, and J. G. Rarity, “Timing and synchronisation for high-loss free-space quantum communication with hybrid de bruijn codes,” *IET Quantum Communication*, vol. 2, no. 3, pp. 80–89, 2021.

[2] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[3] V. Gheorghiu and M. Mosca, “Benchmarking the quantum cryptanalysis of symmetric, public-key and hash-based cryptographic schemes,” *arXiv preprint arXiv:1902.02332*, 2019.

[4] C. H. Bennett and G. Brassard, “Quantum cryptography: Public key distribution and coin tossing,” in *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, IEEE, vol. 175, 1984, p. 8.

[5] A. K. Ekert, “Quantum cryptography and bell’s theorem,” in *Quantum Measurements in Optics*, Springer, 1992, pp. 413–418.

[6] I. Khader, H. Bergeron, L. C. Sinclair, W. C. Swann, N. R. Newbury, and J.-D. Deschênes, “Time synchronization over a free-space optical communication channel,” *Optica*, vol. 5, no. 12, pp. 1542–1548, 2018.

[7] S. Duan, S. Cong, and Y. Song, “A survey on quantum positioning system,” *International Journal of Modelling and Simulation*, vol. 41, no. 4, pp. 265–283, 2021.

[8] S. Ling and C. Xing, *Coding theory: a first course*. Cambridge University Press, 2004.

[9] K. S. Immink, “Runlength-limited sequences,” *Proceedings of the IEEE*, vol. 78, no. 11, pp. 1745–1759, 1990.

[10] L. Song, F. Geng, Z. Song, B.-Z. Li, and Y.-J. Yuan, “Robust data storage in dna by de bruijn graph-based decoding,” 2021.

[11] T. Etzion and A. Lempel, “Algorithms for the generation of full-length shift-register sequences,” *IEEE Transactions on Information Theory*, vol. 30, no. 3, pp. 480–484, 1984.

[12] H. Fredricksen, “A survey of full length nonlinear shift register cycle algorithms,” *SIAM review*, vol. 24, no. 2, pp. 195–221, 1982.

[13] A. Lempel, “On a homomorphism of the de bruijn graph and its applications to the design of feedback shift registers,” *IEEE Transactions on Computers*, vol. 100, no. 12, pp. 1204–1209, 1970.

[14] M. Cohn and A. Lempel, “On fast m-sequence transforms (corresp.),” *IEEE Transactions on Information Theory*, vol. 23, no. 1, pp. 135–137, 1977.

[15] T. van Aardenne-Ehrenfest, “Circuits and trees in oriented linear graphs,” *Simon Stevin*, vol. 28, pp. 203–217, 1951.

[16] M. Fleury, “Deux problemes de geometrie de situation,” *Journal de mathematiques elementaires*, vol. 2, no. 2, pp. 257–261, 1883.

[17] C. Hierholzer and C. Wiener, “Über die möglichkeit, einen linienzug ohne wiederholung und ohne unterbrechung zu umfahren,” *Mathematische Annalen*, vol. 6, no. 1, pp. 30–32, 1873.

[18] M. H. Martin, “A problem in arrangements,” *Bulletin of the American Mathematical Society*, vol. 40, no. 12, pp. 859–864, 1934.

[19] A. M. Alhakim, “A simple combinatorial algorithm for de bruijn sequences,” *The American Mathematical Monthly*, vol. 117, no. 8, pp. 728–732, 2010.

[20] A. Alhakim, E. Sala, and J. Sawada, “Revisiting the prefer-same and prefer-opposite de bruijn sequence constructions,” *Theoretical Computer Science*, vol. 852, pp. 73–77, 2021.

[21] C. J. Mitchell, T. Etzion, and K. G. Paterson, “A method for constructing decodable de bruijn sequences,” *IEEE Transactions on Information Theory*, vol. 42, no. 5, pp. 1472–1478, 1996.

[22] J. Tuliani, “De bruijn sequences with efficient decoding algorithms,” *Discrete Mathematics*, vol. 226, no. 1-3, pp. 313–336, 2001.

[23] T. Kociumaka, J. Radoszewski, and W. Rytter, “Efficient ranking of lyndon words and decoding lexicographically minimal de bruijn sequence,” *SIAM Journal on Discrete Mathematics*, vol. 30, no. 4, pp. 2027–2046, 2016.

[24] D. E. Knuth, *Art of Computer Programming, Volume 4, Fascicle 4, The: Generating All Trees–History of Combinatorial Generation*. Addison-Wesley Professional, 2013.

[25] H. Fredricksen and J. Maiorana, “Necklaces of beads in k colors and k-ary de bruijn sequences,” *Discrete Mathematics*, vol. 23, no. 3, pp. 207–210, 1978.

[26] H. Fredricksen and I. J. Kessler, “An algorithm for generating necklaces of beads in two colors,” *Discrete mathematics*, vol. 61, no. 2-3, pp. 181–188, 1986.

[27] F. Ruskey, C. Savage, and T. M. Y. Wang, “Generating necklaces,” *Journal of Algorithms*, vol. 13, no. 3, pp. 414–430, 1992.

[28] F. Chung, P. Diaconis, and R. Graham, “Universal cycles for combinatorial structures,” *Discrete Mathematics*, vol. 110, no. 1-3, pp. 43–59, 1992.

[29] V. Horan and G. Hurlbert, “Universal cycles for weak orders,” *SIAM Journal on Discrete Mathematics*, vol. 27, no. 3, pp. 1360–1371, 2013.

[30] B. Jackson, B. Stevens, and G. Hurlbert, “Research problems on gray codes and universal cycles,” *Discrete Mathematics*, vol. 309, no. 17, pp. 5341–5348, 2009.

[31] J. R. Johnson, “Universal cycles for permutations,” *Discrete Mathematics*, vol. 309, no. 17, pp. 5264–5270, 2009.

[32] G. Hurlbert, T. Johnson, and J. Zahl, “On universal cycles for multisets,” *Discrete mathematics*, vol. 309, no. 17, pp. 5321–5327, 2009.

[33] B. W. Jackson, J. Buhler, and R. Mayer, “A recursive construction for universal cycles of 2-subspaces,” *Discrete mathematics*, vol. 309, no. 17, pp. 5328–5331, 2009.

[34] G. Hurlbert, “On universal cycles for k-subsets of an n-set,” *SIAM Journal on Discrete Mathematics*, vol. 7, no. 4, pp. 598–604, 1994.

[35] B. W. Jackson, “Universal cycles of k-subsets and k-permutations,” *Discrete mathematics*, vol. 117, no. 1-3, pp. 141–150, 1993.

[36] E. Moreno, “On the theorem of fredricksen and maiorana about de bruijn sequences,” *Advances in Applied Mathematics*, vol. 33, no. 2, pp. 413–415, 2004.

[37] Y. H. Au, “Generalized de bruijn words for primitive words and powers,” *Discrete Mathematics*, vol. 338, no. 12, pp. 2320–2331, 2015.

[38] J. Sawada, A. Williams, and D. Wong, “Generalizing the classic greedy and necklace constructions of de bruijn sequences and universal cycles,” *the electronic journal of combinatorics*, P1–24, 2016.

[39] S. Golomb, S.-R. Sequences, and S. F. Holden-Day, “Ca, 1967,” *Aegean Park*, 1982.

[40] A. Lempel, “Cryptology in transition,” *ACM Computing Surveys (CSUR)*, vol. 11, no. 4, pp. 285–303, 1979.

[41] A. H. Chan, R. A. Games, and E. L. Key, “On the complexities of de bruijn sequences,” *Journal of Combinatorial Theory, Series A*, vol. 33, no. 3, pp. 233–246, 1982.

[42] R. Games and A. Chan, “A fast algorithm for determining the complexity of a binary sequence with period 2ˆ n (corresp.),” *IEEE Transactions on Information Theory*, vol. 29, no. 1, pp. 144–146, 1983.

[43] E. Key, “An analysis of the structure and complexity of nonlinear binary sequence generators,” *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 732–736, 1976.

[44] T. Etzion, N. Kalouptsidis, N. Kolokotronis, K. Limniotis, and K. G. Paterson, “Properties of the error linear complexity spectrum,” *IEEE Transactions on Information Theory*, vol. 55, no. 10, pp. 4681–4686, 2009.

[45] T. Etzion and A. Lempel, “Construction of de bruijn sequences of minimal complexity,” *IEEE Transactions on Information Theory*, vol. 30, no. 5, pp. 705–709, 1984.

[46] A. Lempel and M. Cohn, “Design of universal test sequences for vlsi,” *IEEE transactions on information theory*, vol. 31, no. 1, pp. 10–17, 1985.

[47] Z. Barzilai, D. Coppersmith, and A. L. Rosenberg, “Exhaustive generation of bit patterns with applications to vlsi self-testing,” *IEEE Transactions on Computers*, vol. 32, no. 02, pp. 190–194, 1983.

[48] F. J. MacWilliams and N. J. Sloane, “Pseudo-random sequences and arrays,” *Proceedings of the IEEE*, vol. 64, no. 12, pp. 1715–1729, 1976.

[49] T. Etzion, “Constructions for perfect maps and pseudorandom arrays,” *IEEE Transactions on information theory*, vol. 34, no. 5, pp. 1308–1316, 1988.

[50] A. M. Bruckstein, T. Etzion, R. Giryes, N. Gordon, R. J. Holt, and D. Shuldiner, “Simple and robust binary self-location patterns,” *IEEE transactions on information theory*, vol. 58, no. 7, pp. 4884–4889, 2012.

[51] Y.-C. Hsieh, “Decoding structured light patterns for three-dimensional imaging systems,” *Pattern Recognition*, vol. 34, no. 2, pp. 343–349, 2001.

[52] R. A. Morano, C. Ozturk, R. Conn, S. Dubin, S. Zietz, and J. Nissano, “Structured light using pseudorandom codes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 322–327, 1998.

[53] J. Pages, J. Salvi, C. Collewet, and J. Forest, “Optimised de bruijn patterns for one-shot shape acquisition,” *Image and Vision Computing*, vol. 23, no. 8, pp. 707–720, 2005.

[54] J. Salvi, S. Fernandez, T. Pribanic, and X. Llado, “A state of the art in structured light patterns for surface profilometry,” *Pattern recognition*, vol. 43, no. 8, pp. 2666–2680, 2010.

[55] R. G. Van Schyndel, A. Z. Tirkel, and C. F. Osborne, “A digital watermark,” in *Proceedings of 1st international conference on image processing*, IEEE, vol. 2, 1994, pp. 86–90.

[56] M. J. Chaisson, D. Brinza, and P. A. Pevzner, “De novo fragment assembly with short mate-paired reads: Does the read length matter?” *Genome research*, vol. 19, no. 2, pp. 336–346, 2009.

[57] P. E. Compeau, P. A. Pevzner, and G. Tesler, “How to apply de bruijn graphs to genome assembly,” *Nature biotechnology*, vol. 29, no. 11, pp. 987–991, 2011.

[58] P. A. Pevzner, H. Tang, and M. S. Waterman, “A new approach to fragment assembly in dna sequencing,” in *Proceedings of the fifth annual international conference on Computational biology*, 2001, pp. 256–267.

[59] Y. Zhang and M. S. Waterman, “An eulerian path approach to global multiple alignment for dna sequences,” *Journal of Computational Biology*, vol. 10, no. 6, pp. 803–819, 2003.

[60] Z. Chang, J. Chrisnata, M. F. Ezerman, and H. M. Kiah, “Rates of dna sequence profiles for practical values of read lengths,” *IEEE Transactions on Information Theory*, vol. 63, no. 11, pp. 7166–7177, 2017.

[61] H. M. Kiah, G. J. Puleo, and O. Milenkovic, “Codes for dna sequence profiles,” *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 3125–3146, 2016.

[62] Y. M. Chee, T. Etzion, H. M. Kiah, S. Marcovich, A. Vardy, E. Yaakobi, *et al.*, “Locally-constrained de bruijn codes: Properties, enumeration, code constructions, and applications,” *IEEE Transactions on Information Theory*, vol. 67, no. 12, pp. 7857–7875, 2021.

[63] I. F. Blake, “The enumeration of certain run-length sequences,” *Inf. Control.*, vol. 55, no. 1-3, pp. 222–236, 1982.

[64] O. F. Kurmaev, “Constant-weight and constant-charge binary run-length limited codes,” *IEEE transactions on information theory*, vol. 57, no. 7, pp. 4497–4515, 2011.

[65] J.-P. Duval, “Génération d’une section des classes de conjugaison et arbre des mots de lyndon de longueur bornée,” *Theoretical computer science*, vol. 60, no. 3, pp. 255–283, 1988.