Sequences and Their Applications

Research thesis

Submitted In Partial Fulfillment of the Requirements for the
Joint Degree of Doctor of Philosophy

Johan Chrisnata

Submitted to the Senate of
the Technion - Israel Institute of Technology,
Tevet 5782 Haifa December 2021
The Research Thesis was done under the supervision of

**Asst. Prof. Han Mao Kiah** in the Department of Mathematical Science of Nanyang Technological University and

**Prof. Tuvi Etzion** in the Taub Faculty of Computer Science of Technion – Israel Institute of Technology

I acknowledge Nanyang Technological University for the NPGS scholarship granted to me during my PhD candidacy and Technion - Israel Institute of Technology for the generous financial aid for my studies.
List of Publications

This thesis contains material from 4 paper(s) published in the following peer-reviewed journal(s) / from papers accepted at conferences in which I am listed as an author.

- Chapter 5 contains Johan Chrisnata and Han Mao Kiah. Correcting Two Deletions with More Reads, accepted in the 2021 IEEE International Symposium on Information Theory (ISIT), Melbourne, Australia. Extended version is being prepared to be submitted for publication in “IEEE Transaction on Information Theory”


- Chapter 3 is developed from as Yeow Meng Chee, Johan Chrisnata, Tuvi Etzion and Han Mao Kiah, “Efficient Algorithm for the Linear Complexity of Sequences and Some Related Consequences,” 2020 IEEE International Symposium on Information Theory (ISIT), Los Angeles, CA, USA, 2020, pp. 2897-2902. Extended version is being prepared to be submitted for publication in “Codes, Designs and Cryptography”.

- Chapter 4 is published as Johan Chrisnata, Han Mao Kiah, Sankeerth Rao, Alexander Vardy, Eitan Yaakobi and Hanwen Yao, “On the Number of Distinct k-Decks: Enumeration and Bounds,” 2019 19th International Symposium on Communications and Information Technologies (ISCIT), Ho Chi Minh City, Vietnam, 2019, pp. 519-524. Extended version is accepted for publication in “Advances in Mathematical Communications”.

- Chapter 2 contains the introduction of all the four papers above.

20/12/2021

Date

JOHAN CHRISNATA
# Contents

## Summary

1 Preliminaries
   1.1 Basic Notation ........................................ 1
   1.2 Polynomial ............................................. 2
   1.3 Sequence ................................................ 3
   1.4 Relation ................................................ 3
   1.5 Coding Theory ........................................... 4

2 Introduction
   2.1 Sequences .............................................. 5
      2.1.1 Linear Complexity .................................. 5
      2.1.2 Periodic Sequences ................................ 6
   2.2 Deletion Channel ......................................... 9
   2.3 Sequence Reconstruction Problem ........................ 10
      2.3.1 All Subsequences : $k$-deck Problem .............. 12
      2.3.2 Partial Subsequences : Correcting Deletions with Distinct Reads .......................... 13
   2.4 Outline of the Thesis ................................... 14

3 Linear Complexity
   3.1 Shift Operator ......................................... 15
   3.2 Linear Complexity Computation by gcd .................. 19
   3.3 Games-Chan Algorithm ................................. 20
   3.4 A General Method for any Binary Sequence ............ 21
   3.5 Powers of Irreducible Polynomial ........................ 24
   3.6 Implementation for some Periods $p \cdot 2^n$, $p$ prime ... 26
   3.7 Implementation for Sequences with Odd Period ........ 27
   3.8 Conclusion ............................................... 28

4 The $k$-deck Problem
   4.1 Preliminaries ........................................... 29
   4.2 Polynomial-Time Enumeration ........................... 31
   4.3 Upper Bounds on Deck$_k(n)$ ............................ 36
## CONTENTS

4.4 Lower Bounds ................................................. 41
  4.4.1 Lower Bounds on Deck$_4(n)$ ............................... 43
  4.4.2 Lower Bounds on Deck$_3(n)$ ............................... 44
  4.4.3 Lower Bounds on Deck$_3,3(n)$ .............................. 47
4.5 Conclusion ...................................................... 51

5 Correcting Deletions with Multiple Distinct Reads ............... 53
  5.1 Read Coverage and Deletion Balls ............................ 54
  5.2 Single-deletion Reconstruction Code ........................ 55
  5.3 Two-deletion Reconstruction Code ............................ 62
    5.3.1 $N = 1$ .................................................. 62
    5.3.2 $N = 2n - 3$ ............................................. 64
    5.3.3 $N = 7$ .................................................. 64
    5.3.4 $N = n + 1$ .............................................. 65
    5.3.5 $N = 5$ .................................................. 68
  5.4 General $t$-deletion Reconstruction Code .................... 70
  5.5 Conclusion ...................................................... 70

6 Concluding Remarks ............................................... 73

A Proof of Theorem 3.18 ........................................... 75
  A.1 Proof of Theorem 3.18 part 1 ................................ 75
  A.2 Proof of Theorem 3.18 part 2 and part 3 .................... 78

B Proof of Theorem 5.25 and Theorem 5.27 ......................... 83
  B.1 Type-B-Confusability ........................................ 83
  B.2 Special Cases ................................................. 85
  B.3 Proof of Theorem 5.25 ........................................ 91
  B.4 Proof of Theorem 5.27 ........................................ 94

C Proof of Theorem 5.32 ........................................... 97
  C.1 Characteristics of Type-C confusable Sequences ............ 97
    C.1.1 Intersection of Two-Deletion Balls ....................... 98
    C.1.2 Proof of Theorem C.2 .................................... 100
  C.2 Proof of Theorem 5.32 ........................................ 105

List of Author's Awards, Patents, and Publications ................. 109

Bibliography ........................................................... 111
Summary

Binary and $q$-ary sequences have always been used in communication channel as the carrier or the vessel of information. In order to establish an efficient and error-free communication channel, investigations on the properties of sequences are crucial. This dissertation is devoted to the study of sequences to investigate its properties which can be useful to establish a more reliable communication channel.

The properties that we will investigate in this dissertation are the linear complexities of sequences and the reconstruction of a sequence from its sub-sequences. The linear complexity of a binary sequence is defined as the length of the shortest linear feedback shift-register that generates the binary sequence. In the first part of this dissertation, we devised a novel and efficient algorithm to find the linear complexity of any binary sequence. This algorithm is a generalization of the well-known Games-Chan algorithm. Furthermore, this algorithm can be applied in linear time and is faster than previous well-known algorithms for certain parameters of length of periodic sequences.

The second property that we will investigate in this dissertation is the reconstruction capability of binary sequences in particular from its sub-sequences. This is called the sequence reconstruction problem. The problem considers a communication scenario where the sender transmits a sequence from some codebook and the receiver obtains multiple noisy reads of the original sequence. Noisy reads here refer to possibly erroneous copies of the original sequence. The receiver/decoder then aims to reconstruct the original sequence from these noisy reads. The error that we consider in this dissertation is only deletion error where some elements of the original sequence might be deleted. Thus, the noisy reads here are in the form of subsequences of the original sequence.

There are two variants of the problem that we are going to consider for this problem. Firstly, we assume that the decoder receives all possible subsequences of a certain fixed length from the original sequence, including its multiplicity. In other words,
the decoder obtains the profile of sub-sequences of the original sequence, namely the $k$-deck of the binary sequence. The $k$-deck of a sequence is defined as the multiset of all its subsequences of length $k$. We determine the exact value of the number of distinct $k$-decks for all binary sequences of the same length for small values of $k$ and provide asymptotic estimates of this value when $k$ is fixed. Specifically, we introduce a trellis-based method to compute this value for fixed $k$ in polynomial time.

The second variant is under the assumption that the decoder does not receive every possible subsequence of the original sequence, but only receives some fixed number of noisy reads. In other words, the decoder receives a fixed number of subsequences. In this case, we also assume that all received noisy reads are distinct. We construct codes that are capable of correcting $t$ deletions with multiple noisy reads. Special attention is given to the case when $t = 1$ and $t = 2$. 
Chapter 1

Preliminaries

We introduce the common notations and terminologies used throughout this dissertation. We define also the common formulas and terms that will be used frequently.

1.1 Basic Notation

We denote the set of integers and the set of positive integers as $\mathbb{Z}$ and $\mathbb{Z} > 0$ respectively. We denote $\mathbb{R}$ to be the set of real numbers, and $\mathbb{F}$ to be a field in general. In particular, $\mathbb{F}_q$ denotes the finite field of order $q$, where $q$ is a prime power. For $r \in \mathbb{R}$, we let $\lfloor r \rfloor$ to denote the biggest integer that is not bigger than $r$, and $\lceil r \rceil$ to denote the smallest integer that is not smaller than $r$. For a positive integer $n \in \mathbb{Z}$, we denote $\mathbb{Z}_n$ to be the ring of integers modulo $n$. In other words $\mathbb{Z} = \{0, 1, \ldots, n - 1\}$, where $\overline{a} = \{a + nk : k \in \mathbb{Z}\}$.

Let $\Sigma$ denote the alphabet. For a positive integer $n$, we denote $\Sigma^n$ to be the set of sequences of length $n$ over $\Sigma$, in other words the entries of the sequences are in $\Sigma$. We denote a sequence in $\Sigma^n$ with lowercase but bold letters, for example $x$, where $x = x_1x_2 \cdots x_n$, and $x_i$'s are its entries in $\Sigma$. We denote also $\Sigma^*$ to be the set of sequences over $\Sigma$ with any length. In addition if $\Sigma = \{0, 1\}$, then $x \in \Sigma^*$ is called a binary sequence, while if $|\Sigma| = q$, then $x \in \Sigma^*$ is called a $q$-ary sequence. For a $q$-ary sequence, unless stated otherwise, we assume that the alphabet $\Sigma = \{0, 1, \ldots, q - 1\}$. For two sequences $x = x_1x_2 \cdots x_n$ and $y = y_1y_2 \cdots y_m$, we denote
the concatenation of $x$ with $y$ as $xy = x_1x_2 \cdots x_ny_1y_2 \cdots y_m$. We define the weight of a sequence $x$ to be the number of non-zero entries in the sequence $x$.

If $A$ and $B$ are subsets of $\Sigma^*$, then we denote them by $A, B \subseteq \Sigma^*$. We denote $A \circ B$ to be the set of sequences that can be obtained by concatenating a sequence $x \in A$ with $y \in B$. The intersection of two sets $A$ and $B$ is denoted as $A \cap B$, while the union of two sets $A$ and $B$ is denoted as $A \cup B$. We also denote $|A|$ to be the cardinality/size of the set $A$. We denote $A \setminus B$ to be the set difference between $A$ and $B$, to be precise we let $A \setminus B = \{a \in A : a \notin B\}$.

Let $f$ and $g$ be real-valued functions defined on $\mathbb{R}$. Then we have the following notations:

- $f(z) = O(g(z))$ if and only if there exists $z_0 \in \mathbb{R}$ and $M > 0$ such that $|f(z)| \leq M|g(z)|$ for all $z \geq z_0$.
- $f(z) = \Omega(g(z))$ if and only if there exists $z_0 \in \mathbb{R}$ and $M > 0$ such that $|f(z)| \geq M|g(z)|$ for all $z \geq z_0$.
- $f(z) = \Theta(g(z))$ if and only if $f(z) = O(g(z))$ and $f(z) = \Omega(g(z))$.
- $f(z) = o(g(z))$ if and only if for all $M > 0$, there exists $z_0 \in \mathbb{R}$ such that $|f(z)| < Mg(z)$ for all $z \geq z_0$.

A multiset is a modification of the definition of a set, in which it allows multiple instances of each of its elements. The number of instances that an element appear in the multiset is called the multiplicity of an element. For example, in the multiset $A = \{a, a, a, b, b\}$, the multiplicity of $a$ is 3, and the multiplicity of $b$ is 2.

## 1.2 Polynomial

Let $z$ be a variable or indeterminate. A polynomial $p(z)$ in $z$ can be written as

$$p(z) = a_0 + a_1z + a_2z^2 + \cdots + a_nz^n = \sum_{i=0}^{n} a_i z^i,$$

where $a_i \in \mathbb{F}$ for all $0 \leq i \leq n$ and some field $\mathbb{F}$. Furthermore, $p(z)$ is said to have degree $n$ if $a_n \neq 0$. A polynomial $p(z)$ is said to divide $q(z)$ if there exists a
polynomial \( r(z) \) such that \( q(z) = p(z) r(z) \). Let \( p(z) \) and \( q(z) \) be two polynomials whose coefficients are real numbers. The greatest common divisor of \( p(z) \) and \( q(z) \), denoted as \( \text{gcd}(p(z), q(z)) \) is defined as the polynomial \( r(z) \) with the biggest degree such that \( r(z) \) divides both \( p(z) \) and \( q(z) \).

### 1.3 Sequence

In general, there are two types of sequences. Firstly finite sequence, as explained in Section 1.1, which is basically an element of \( \Sigma^n \), where \( \Sigma \) is a set of alphabet and \( n \) is the length of the sequence. Second, there is also infinite sequence, \( x = \{x_i\}_{i \geq 0} = x_0 x_1 x_2 \ldots \), where \( x_i \in \Sigma \).

Let \( x = x_1 x_2 \cdots x_n \) be a finite binary sequence. A subsequence \( x' \) of \( x \) is defined to be any sequence that can be obtained from \( x \) by deleting some of its elements. To be formal, a subsequence \( x' \) of length \( t \) can be \( x_{a_1} x_{a_2} \cdots x_{a_t} \), where \( 1 \leq a_1 < a_2 < \cdots < a_t \leq n \), and \( t \leq n \). For example, if \( x = 011000 \), then \( x' = 0100 \) and \( x'' = 11 \) are subsequences of \( x \).

### 1.4 Relation

Let \( S \) and \( T \) be two sets. The Cartesian product of \( S \) and \( T \) is defined to be \( S \times T = \{(a, b) : a \in S, b \in T\} \). A binary relation \( \sim \) on \( S \) is defined to be a subset of the Cartesian product \( S \times S \). In other words, we write \( a \sim b \) if and only if the pair \((a, b) \in \sim\).

Given a set \( S \). A binary relation \( \sim \) on a set \( S \) is said to be an equivalence relation if it satisfies the following conditions:

- For every \( a \in S \), we have \( a \sim a \).
- For every \( a, b \in S \), if \( a \sim b \) then \( b \sim a \).
- For every \( a, b, c \in S \), if \( a \sim b \) and \( b \sim c \) then \( a \sim c \).

The equivalence class of \( a \in S \) under the binary relation \( \sim \) is \([a] = \{s \in S : a \sim s\}\).
1.5 Coding Theory

Let $\Sigma$ be an alphabet such that $|\Sigma| = q$. Let $\Sigma^n$ be the set of sequences of length $n$ over $\Sigma$, then a $q$-ary code $C$ of length $n$ is a subset of $\Sigma^n$. The elements of $C$ are called codewords of length $n$. Furthermore, the size of code $C$, $|C|$ is the number of codewords in $C$.

Given two codewords $x$ and $y$ in $C \subset \Sigma^n$, the Hamming distance between $x$ and $y$ is defined as the number of indices $1 \leq i \leq n$ such that $x_i \neq y_i$, and is denoted by $d_H(x, y)$.

The rate of a $q$-ary code $C$ of length $n$ is defined to be $\text{Rate}(C) = \frac{\log_q |C|}{n}$. While, the redundancy of the code $C$ is defined to be $n - \text{Rate}(C)$. Everything else being equal, we want to minimize the redundancy of a code $C$. Suppose that there is an infinite family of codes $\{C_i\}_{i \geq 0}$, where $C_i$ is a code of length $i$, then the asymptotic rate of the family is $R(\{C_i\}_{i \geq 0}) = \lim_{n \to \infty} \frac{\log_q |C_n|}{n}$. Unless explicitly written, a log would have a base of two. When $R(\{C_i\}_{i \geq 0}) = 1$, then we say that the family of codes $\{C_i\}_{i \geq 0}$ is asymptotically optimal.
Chapter 2

Introduction

2.1 Sequences

Data and information have always been stored in the form of sequences, be it binary or \( q \)-ary sequences. In order to store and retrieve the data efficiently, we need to understand the properties of sequences. This dissertation is devoted to the study of characteristics of sequences as well as the implementation on constructing error-correcting codes. The two main properties that we will discuss in this dissertation are the linear complexity of sequences and the profile of its subsequences. Gaining insight towards the properties of sequences, we will construct an error-correcting code capable of correcting a type of error, called the deletion error.

2.1.1 Linear Complexity

The linear complexity of a sequence is one important measure of a sequence; it represents the smallest degree of a linear recursion that the sequence satisfies. To be specific, for a sequence \( x \), its linear complexity \( c(x) \), is defined to be the length of the shortest linear feedback shift-register that generates \( x \). Binary sequences with high complexity and good pseudorandomness are widely used as keystreams in cryptographic applications [25, 26]. Sequences of low linear complexity can be fully determined or identified by solving a system of \( c(x) \) linear equations if \( 2c(x) \) consecutive terms of the sequence are known. Hence having higher linear complexity shows to be an advantage in cryptographic applications. Thus there
has been an extensive research done in the past to find an efficient algorithm that can determine the complexity of a sequence.

In regards to the efficiency of the algorithm to find linear complexity, there are two main components that we are concerned:

- **Time complexity** of the implementation of the algorithm: the time that the algorithm takes to run. The same algorithm may be implemented faster for one sequence compared to other sequence. In light of this, some developed algorithms focus on certain type of sequences, such as periodic sequences.

- **Space complexity** of the algorithm: the space/memory that the algorithm takes to run.

However, as far as this thesis are concerned, we will mainly discuss and focus on the time complexity of the algorithms. The linear complexity of a sequence \( x \) of period \( N \) over a finite field \( \mathbb{F}_q \) can be determined with the well-known Berlekamp-Massey algorithm [3, 23] in \( O(N^2) \) symbol field operations. This algorithm was implemented during the years in various ways, e.g. [16, 30]. The complexity of this algorithm was improved to \( O(N(\log N)^2 \log \log N) \) in [4–7]. Theoretically, the most efficient algorithm would have at least linear time algorithm, and thus **the ultimate goal is to find an algorithm that can be implemented in linear time**, in other words, it runs in \( kN \) symbol field operations for some small constant \( k \).

### 2.1.2 Periodic Sequences

The Berlekamp-Massey algorithm considered sequences of any period over any finite field \( \mathbb{F}_q \). But, in many applications, only periodic sequences are considered and hence the algorithm to find the linear complexity of such sequences, can be considerably improved. To be more formal, if \( x = \{x_i\}_{i \geq 0} \) is an infinite binary sequence, then \( x \) has a period \( N \), if \( N \) is the least positive integer such that \( x_i = x_{i+N} \) for each \( i \geq 0 \). Such a sequence is considered as a cyclic sequence and can be denoted by \([x_0, x_1, \ldots, x_{N-1}]\). One example of such periodic sequences with special interest is binary sequence whose period is a power of two. An important family of such sequences are de Bruijn sequences [8, 14, 17]. Their complexities was
considered in many papers, e.g. [9, 15]. The linear complexity of sequences whose period is \(2^n\) has some application also in other areas such as Gray codes [27].

One simple algorithm that finds the linear complexity \(c\) of any sequence \(x\) with period \(N = 2^n\) was proposed by Games and Chan [18]. Implementation of their algorithm requires \(N\) bit operations on the sequence \(x\) and another \(n\) bit operations are required from integer operations to compute \(c\). In the following two decades, a few algorithms were suggested to generalize this algorithm to binary sequences with other periods and periodic sequences over \(\mathbb{F}_q\). The time complexity of these proposed algorithms for sequences with period \(N\) were kept as low as \(\beta N\) for some constant \(\beta\) which means they are linear time algorithm. However, in practice, since the constant \(\beta\) is huge, the time complexity is still much higher than the \(N + \log N\) bit operations required for The Games-Chan Algorithm.

The following linear time algorithms are some of the generalizations, which can be summarized in Table 2.1 on the next page. One example is the generalization for sequences with period \(p^n\) over \(\mathbb{F}_{p^t}\), which was given in [12, 20]. Xiao et al. [31, 32] also gave an algorithm to compute the linear complexity of sequences with period \(N \in \{p^t, 2p^t\}\) over \(\mathbb{F}_q\), when \(q\) is a primitive root modulo \(p^2\). Furthermore, Chen [10] gave an algorithm for sequences over \(\mathbb{F}_{p^t}\) with period \(\ell \cdot 2^n\), where \(2^n|p^t - 1\) and \(\gcd(\ell, p^t - 1) = 1\). Moreover, Chen [11] generalized this algorithm to determine the linear complexity of sequences with period \(\ell \cdot n\) over \(\mathbb{F}_{p^t}\), where \(\ell|p^t - 1\) and \(\gcd(n, p^t - 1) = 1\). The main idea in [11] is to reduce the computation of linear complexity of a sequence with period \(\ell \cdot n\) over \(\mathbb{F}_{p^t}\) to the computation of linear complexities of \(\ell\) sequences, each with period \(n\) over \(\mathbb{F}_{p^t}\).

The algorithms in [10, 11, 31, 32] are designed for sequences over a field of odd order. The ideas in [10, 11] are generalized in [24] for binary sequences. Until recently, Meidl [24] presented the most efficient algorithm for computing the linear complexities of binary sequences of period \(N = \ell \cdot 2^n\). To apply Meidl’s algorithm on a sequence \(x\), one forms a family of sequences of period \(2^n\) from \(x\) and apply The Games-Chan Algorithm to each of these sequences. Then for specific values of \(\ell\), Meidl showed that the algorithm requires \(\beta N\) in time complexity, where \(\beta\) is a small constant. The algorithm in [24] is of interest for large \(N\), a small odd integer \(\ell\) such that the smallest \(k\) for which \(\ell\) divides \(2^k - 1\) is not large.
Table 2.1: Summary of linear time algorithms that find linear complexity of sequences for different periods and different fields.

<table>
<thead>
<tr>
<th>Period</th>
<th>Field</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^n$</td>
<td>$\mathbb{F}_2$</td>
<td>Games and Chan (1983)</td>
</tr>
<tr>
<td>$p^n$</td>
<td>$\mathbb{F}_{p^t}$</td>
<td>C. Ding (1991) and K. Imamura and T. Moriuchi (1993)</td>
</tr>
<tr>
<td>$p^t, 2p^t$ where $p$ is odd</td>
<td>$\mathbb{F}_q$, where $q$ is a primitive root modulo $p^2$</td>
<td>Xiao et al. (2000,2002)</td>
</tr>
<tr>
<td>$\ell \cdot 2^n$</td>
<td>$\mathbb{F}_{p^t}$, where $2^n</td>
<td>p^t-1$ and $\gcd(\ell, p^t - 1) = 1$</td>
</tr>
<tr>
<td>$\ell \cdot n$</td>
<td>$\mathbb{F}_{p^t}$, where $\ell</td>
<td>p^t - 1$ and $\gcd(n, p^t - 1) = 1$</td>
</tr>
<tr>
<td>$\ell \cdot 2^n$</td>
<td>$\mathbb{F}_2$</td>
<td>W. Meidl (2008)</td>
</tr>
</tbody>
</table>

As in [24], in this thesis we only focus on the most important class of sequences, namely the binary sequences. We present an algorithm which is very similar in nature to The Games-Chan Algorithm and can be viewed as its direct generalization compared to previous algorithms. The idea is different from Meidl’s paper, as we do not break down the original sequence into several sequences and apply the Games-Chan Algorithm. Moreover, the algorithm can handle efficiently sequences of even period as in Meidl’s paper (2008) and also binary sequences of some odd periods, which are not considered by previous algorithms.

Our contributions for this part can be summarized as follows:

- **Direct generalization of Games and Chan’s algorithm, namely the Powers of Irreducible Polynomial (PIP) algorithm for an important class of sequences.** This algorithm is a direct generalization of Games-Chan algorithm that deals only with binary sequences with period $N = 2^n$.

- **Using PIP algorithm, we present a general idea to find linear complexity of binary sequence of any period $N$.** The algorithm can handle efficiently sequences of even period as in Meidl’s paper (2008) and also binary sequences of some odd period. Further details will be elaborated in Chapter 3.

- **The minimal polynomial which generates the sequence is also computed in the algorithm.** This property does not exist in the algorithm of Meidl [24].
• The algorithm can be implemented in linear time for binary sequences of period $p_1^{n_1}p_2^{n_2} \cdots p_t^{n_t}$, where the $p_i$’s are primes, $n_i$’s are positive integers, and the polynomial $\sum_{i=0}^{p_i-1}x^{ip_jm}$ is primitive, for $0 \leq m \leq n_j - 1$.

• The algorithm requires $\beta N$ bit operations to compute the linear complexity of a binary sequence $x$ of period $N$, where the constant $\beta$ is relatively small.

Our exposition of the algorithm, properties of sequences and their linear complexities will be discussed in detail in Chapter 3. For the next section, we will move on to the second property of sequence that we will discuss in this thesis, namely to identify a sequence from its subsequences or more concisely called sequence reconstruction problem. But before that we need to introduce deletion channel.

## 2.2 Deletion Channel

In 1966, Levenshtein introduced the study of deletion channels where certain bits of a transmission are lost and their positions are not known to the receiver [67]. Formally, when a codeword $x$ of length $n$ from some codebook $\mathcal{C}$ is sent through a $t$-deletion channel, a subsequence $\tilde{x}$ of length $n - t$ is received. The receiver/decoder then aims to reconstruct the original sequence $x$ from its subsequence $\tilde{x}$ of length $n - t$. A $t$-deletion correcting code $\mathcal{C}$ is then a subset of length-$n$ binary sequences such that for any codeword $x \in \mathcal{C}$, we are able to uniquely identify $x$ from any length-$(n - t)$ subsequence of $x$. This is illustrated in Figure 2.1.

![Image of deletion channel diagram](https://example.com/deletion-channel-diagram.png)

**Figure 2.1:** Classic Deletion Channel Diagram

In the same paper, Levenshtein showed that the optimal redundancy (defined by $n - \log |\mathcal{C}|$) of a $t$-deletion correcting code $\mathcal{C}$ is between $t \log n + \Theta(1)$ and $2t \log n + \Theta(1)$. For $t = 1$, Levenshtein demonstrated that the Varshamov-Tenengolts (VT) code is a single-deletion correcting code with $\log n + O(1)$ redundant bits and is hence asymptotically optimal. However, for $t \geq 2$, the code construction (i.e. upper
bound) given by Levenshtein is essentially greedy and runs in exponential time (in $n$). Efficient constructions of $t$-deletion correcting codes with $C_t \log n$ redundant bits (here, $C_t$ is a constant dependent on $t$ only) were not known until a recent breakthrough by Brakensiek et al. [71].

Following which, there was a flurry of constructions [72–78] where the constant $C_t$ was gradually brought down (see Table 2.2 for a summary). Currently, the best known explicit construction has $4t \log n + o(\log n)$ redundant bits for general $t$ [77]. When $t = 2$, the best known explicit construction has $4 \log n + O(\log \log n)$ redundant bits [75] and this construction meets the existential upper bound. However, there remains a gap to the lower bound of $t \log n$ bits. In particular, for $t = 2$, it remains an open question whether there is a family of two-deletion correcting codes with $2 \log n$ redundant bits.

<table>
<thead>
<tr>
<th>Work</th>
<th>$t$</th>
<th>Redundancy*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brakensiek, Guruswami, and Zbarsky, 2018 [71]</td>
<td>$t \geq 2$</td>
<td>$C_t \log n$ where $C_t$ is a large constant</td>
</tr>
<tr>
<td>Sima and Bruck, 2020 [78]</td>
<td>$t \geq 3$</td>
<td>$8t \log n$</td>
</tr>
<tr>
<td>Sima and Gabrys, 2020 [77]</td>
<td>$t \geq 3$</td>
<td>$4t \log n$</td>
</tr>
<tr>
<td>Garbys and Sala, 2019 [72]</td>
<td>$t = 2$</td>
<td>$8 \log n$</td>
</tr>
<tr>
<td>Sima, Raviv, and Bruck, 2020 [73]</td>
<td>$t = 2$</td>
<td>$7 \log n$</td>
</tr>
<tr>
<td>Guruswami and Håstad, 2020 [75]</td>
<td>$t = 2$</td>
<td>$4 \log n$</td>
</tr>
</tbody>
</table>

TABLE 2.2: Summary of explicit $t$-deletion correcting codes. *Sublogarithmic terms, i.e. $o(\log n)$, are suppressed.

In the next section, we relax the coding problem by allowing the receiver to obtain more than one noisy version or read. This relaxation is reasonable and in fact, motivated by emerging storage media – such as DNA based storage [1, 55, 79, 80] and racetrack memories [56–58] – that provide users with multiple cheap (albeit noisy) reads.

### 2.3 Sequence Reconstruction Problem

Our genes consist of long strands/segments of DNA sequences. However, current sequencing technology either is unable to determine the long sequence directly or reads the sequence at a high error rate. Therefore, most sequencing methods obtain information about its short substrings or subsequences and attempt to infer or reconstruct the original string from this partial information. This gives rise to
a myriad of combinatorial problems, known as sequence reconstruction problems [33, 35, 40–42, 44, 47].

The sequence reconstruction problem, as shown in Figure 2.2 was introduced by Levenshtein in 2001 [59]. In Levenshtein’s seminal work, he considered a communication scenario where the sender transmits a codeword $x$ from some codebook $C$ and the receiver obtains multiple noisy reads $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_N$ of the codeword. Noisy reads here refer to copies of the original codeword $x$ with possibly some errors, and hence $\tilde{x}_i$ might be different from the original $x$, for all $1 \leq i \leq N$. The receiver/decoder then aims to reconstruct the original sequence $x$ from these noisy reads $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_N$ accurately instead of just one noisy read like the classical deletion channel in Figure 2.1.

![Figure 2.2: Sequence Reconstruction Problem Diagram](image)

The purpose of sequence reconstruction problem is to leverage on these multiple reads to increase the information capacity of these next-generation devices, or equivalently, reduce the number of redundant bits. The common setup assumes the codebook $C$ to be the entire space of $\Sigma^n$, where $\Sigma$ is the alphabet and $n$ is the length of the codeword. Thus the problem is to determine the minimum number of distinct reads $N$ that is required to reconstruct the transmitted codeword. In this dissertation, we will only focus on deletion channels, which means that after the codeword $x$ goes through the channel, some of its bits might be deleted and thus becomes noisy read $\tilde{x}_i$. This means, that what the receiver/decoder obtains is a set of subsequences of $x$.

There are two approaches that we will discuss in this dissertation. Firstly, we study the sequence reconstruction problem under the assumption that the receiver obtains all possible subsequences of the same length that can be obtained from $x$. This means that the channel will delete a constant number of elements from the original sequence $x$ and output a multiset of subsequences of the original sequence $x$. We call this multiset of all subsequences of length $k$ to be $k$-decks, which will
be elaborated in Subsection 2.3.1. Second, we study the sequence reconstruction problem under the assumption that the decoder receives a fixed number of noisy reads but not necessarily all possible noisy reads as in the previous assumption. Furthermore, under this case, we also assume that all received noisy reads are distinct. This study will be elaborated further in Subsection 2.3.2.

### 2.3.1 All Subsequences : $k$-deck Problem

First described by Kalashnik [43], the $k$-deck of a sequence is defined to be the multiset of all its subsequences of length $k$. Traditionally, the $k$-deck problem is to determine $S(k)$, the smallest value of $n$ such that all sequences of length $n$ have unique $k$-decks. The exact values of $S(k)$ are known for $k \leq 5$ and both upper and lower bounds for $S(k)$ have been extensively studied [37, 38, 43, 45, 49, 53].

Motivated by applications in DNA-based data storage (see [79] for a broad overview), we study the coded version of the $k$-deck problem. Consider sequences of length $n$. Instead of requiring all sequences in $\Sigma^n$ to have different $k$-decks, we choose a subset of these sequences, or a codebook, such that every codeword in this codebook can be uniquely identified by its $k$-deck. In this setting, we consider the following fundamental problem: how large can this codebook be?

Equivalently, this problem can be restated as an enumeration problem:

Consider all sequences of length $n$. How many distinct $k$-decks are there?

Let $\text{Deck}_k(n)$ denote this quantity and be the object of study for this subsection. In another context, the authors of [39, 46, 51] used the term $k$-binomial equivalence to describe sequences with the same $k$-deck and provided basic upper bounds on $\text{Deck}_k(n)$. In particular, Rigo and Salimov determined $\text{Deck}_2(n)$ and showed that $\text{Deck}_k(n) = O \left( n^{\Delta(k)} \right)$, where $\Delta(k) = 2((k - 1)2^k + 1)$. The values of $\text{Deck}_3(n)$ for $n \leq 16$ are listed in the On-Line Encyclopedia of Integer Sequences [50].

Our contributions for this part can be summarized as follows:

- Provide a trellis-based method to compute $\text{Deck}_k(n)$. See [48, 54] for the definition of a trellis.
• Determine the exact values of $\text{Deck}_k(n)$ for $k \in \{3, 4, 5, 6\}$ and $n \leq 30$.

• Provide asymptotic estimates of $\text{Deck}_k(n)$ for the case where $k$ is fixed. In particular, we improve the asymptotic upper bound to $O(n^{2\Delta(k)/4+1/2})$ for $k \geq 2$.

Our exposition of trellis-based method, the computation and proofs of estimates of $\text{Deck}_k(n)$, as well as possible extension to non-binary alphabet will be briefly discussed in Chapter 4.

### 2.3.2 Partial Subsequences: Correcting Deletions with Distinct Reads

Motivated by modern storage devices, Kiah et al. [60] introduced a variant of the sequence reconstruction problem where the number of noisy reads $N$ is fixed. Hence, our fundamental problem is then: how large can this codebook $\mathcal{C}$ be? Or equivalently, what is the minimum redundancy?

Under the assumption that the decoder receives $N$ distinct noisy reads, we will mainly consider $t$-deletion channel when $t = 1$ and $t = 2$. Our contributions for the following cases are also described below.

**Sequence Reconstruction for Single-deletion channel**: Modifying a code construction in [58], Kiah et al. proposed in [60] a reconstruction code for the single-deletion channel for $N = 2$ with $\log_2 \log_2 n + O(1)$ bits of redundancy. For this case, we focus on the converse of the problem and show that $\log_2 \log_2 n - O(1)$ redundant bits are necessary, and thereby demonstrating that the code construction provided in [60] is asymptotically optimal. Furthermore, we show that these reconstruction codes can be used in $t$-deletion channels (with $t \geq 2$) to uniquely reconstruct codewords from $n^{t-1} + O(n^{t-2})$ distinct noisy reads.

**Sequence Reconstruction for Two-deletion channel**: In 2018, Gabrys and Yaakobi showed that a single-deletion-correcting code is able to uniquely reconstruct from $N \geq 7$ reads from a two-deletion channel [61]. In other words, $\log n$ redundant bits are sufficient to reconstruct the original sequence from $N = 7$ noisy reads. Recall also that the best known two-deletion correcting code (i.e. reconstructs from one read) uses $4 \log n + o(\log n)$ redundant bits. Hence, we ask: for
“intermediate” values of $N$, i.e. $2 \leq N \leq 6$, are there codebooks that use “intermediate” amount of redundancy? In this work, we provide an explicit codebook with $2\log n + o(\log n)$ redundant bits that uniquely reconstructs using $N = 5$ reads.

Our proof of optimality of code construction for 1-deletion channel for $N = 2$ reads as well as the code construction for 2-deletion channel with $N = 5$ noisy reads will be elaborated further in Chapter 5.

## 2.4 Outline of the Thesis

Chapter 1 introduces the basic notations, symbols and terminologies in algebra and coding theory that are frequently used throughout this dissertation. This chapter is meant to align our basic definitions and notations to avoid ambiguity while reading this dissertation.

Chapter 2 introduces the background, motivation and more specific terminologies of each topic as well as the past results that has been made over the past few years. Furthermore, it also highlights the contribution of this dissertation for the rest of the chapters.

Chapter 3 describes our algorithm to find linear complexity of a sequence, as well as their properties that has been briefly discussed in Subsection 2.1.2. Furthermore, it also discusses in detail the efficient implementation of the algorithm for certain periodic sequences.

Chapter 4 provides the trellis-based method to compute the number of distinct $k$-decks for sequences of length $n$. Furthermore this chapter also provides the exact values for $\text{Deck}_k(n)$ for some values of $k$ and $n$ as described in Subsection 2.3.1, as well as the asymptotic upper bound for $\text{Deck}_k(n)$.

Chapter 5 provides the proof of optimality a certain reconstruction code for single-deletion channel with two reads as described in Subsection 2.3.2. We use a graph theoretical approach in this proof of optimality. Furthermore, we also proposed a reconstruction code for 2-deletion channel that can uniquely reconstruct the original sequence using 5 noisy reads.
Chapter 3

Linear Complexity

In this chapter, we discuss The Games-Chan Algorithm and its direct generali-

tation, namely the Powers of Irreducible Polynomial (PIP) algorithm. Using PIP

algorithm as a building block, we present a general idea to find linear complexity

of binary sequences of any period $N$. To be precise, our generalization is a linear

time algorithm that outputs the linear complexity of sequences with period $p \cdot 2^n$,

where $p$ is a prime and $n$ is a positive integer. The cases which will be discussed

are $p = 3$ and $p = 1(\text{mod} \ 4)$, where 2 is a generator modulo $p$.

As a benchmark of efficiency, we compare the complexity of our algorithm to the
algorithm presented in [24]. The comparison for the other cases when the period
is $\ell \cdot 2^n$ is more complicated and is not considered in this thesis. Furthermore, a

generalization that works for sequences with odd period is also presented at the end
of this chapter. Part of this chapter is already presented in the IEEE International
Symposium of Information Theory 2020 in [28]

3.1 Shift Operator

Let $x = \{x_i\}_{i \geq 0}$ be an infinite binary sequence. The sequence has a period $N$, if $N$
is the least positive integer such that $x_i = x_{i+N}$ for each $i \geq 0$. Such a sequence can

considered as a cyclic sequence of length $N$ and is denoted by $[x_0, x_1, \ldots, x_{N-1}]$.

For this chapter, we assume that all sequences are infinite binary sequences with a

period, and therefore can be denoted as a finite cyclic sequence $[x_0, x_1, \ldots, x_{N-1}]$.
3.1. Shift Operator

Any periodic sequence must satisfy a linear recursion

\[ x_{i+m} = a_1 x_{i+m-1} + \cdots + a_{m-1} x_{i+1} + a_m x_i , \quad i \geq 0, \]  

(3.1)

of some order \( m \leq N \), where \( a_j \in \{0, 1\} \) and all computations are done modulo 2. For instance, a periodic sequence of period \( N \), satisfies a linear recursion \( x_{i+N} = x_i \) of order \( N \). The linear complexity \( c(x) \) of \( x \) is defined as the least \( m \) for which (3.1) holds. Clearly \( c(x) \leq N \), for any sequence with period \( N \), since \( x_{i+N} = x_i \) by definition.

Now, we want to redefine equation (3.1) in a more formal setting. Therefore, we define \( \text{shift operator} \ E \) on the set of all cyclic sequences, such that

\[ E[x_0, x_1, \ldots, x_{N-1}] = [x_1, \ldots, x_{N-1}, x_0], \]

for any cyclic sequence \( x = [x_0, x_1, \ldots, x_{N-1}] \). Consequently, if the cyclic sequence \( x \) is clear from the context or is fixed, we can also define \( E x_i = x_{i+1} \). Now, we define \( E^k x \triangleq E(E^{k-1})x \), for \( k > 1 \). Therefore, if \( f(E) = \sum_{i=0}^{k} a_i E^i \), for some constant \( a_i \in \{0, 1\} \), then \( f(E)x = \sum_{i=0}^{k} (a_i E^i x) \), where addition of two sequences of the same length is performed bitwise. Thus, the linear recursion (3.1) can be restated as

\[ \left( E^m + \sum_{j=1}^{m} a_j E^{m-j} \right) x_i = 0 , \quad i \geq 0. \]

(3.2)

Since \( m \) is the smallest such integer such that (3.1) holds, it implies that \( a_m \neq 0 \). If we define

\[ f(E) \triangleq E^m + \sum_{j=1}^{m} a_j E^{m-j} , \]

then from equation (3.2) we have \( f(E)x_i = 0 \) for each \( i \geq 0 \). Replacing the shift operator \( E \) by the variable \( z \), we have that \( f(z) = z^m + \sum_{j=1}^{m} a_j z^{m-j} \), where \( a_m \neq 0 \). Let \( c_{m-j} = a_j \) for \( 1 \leq j \leq m \), and hence \( f(z) = z^m + \sum_{j=1}^{m} c_{m-j} z^{m-j} = z^m + \sum_{j=0}^{m-1} c_j z^j \), where \( c_0 \neq 0 \). This implies that (3.2) takes the form

\[ f(E)x_i = \left( E^m + \sum_{j=0}^{m-1} c_j E^j \right) x_i = 0 , \quad i \geq 0. \]
Based on the previous observation, we can summarize it in following lemma.

**Lemma 3.1.** If for \(x = [x_0, x_1, \ldots, x_{N-1}]\) the linear recurrence of the least degree \(m\) satisfies

\[
x_{i+m} = c_{m-1}x_{i+m-1} + c_{m-2}x_{i+m-2} + \cdots + c_1 x_{i+1} + c_0 x_i,
\]

for each \(i \geq 0\), then

\[
\left( E^m + \sum_{j=0}^{m-1} c_j E^j \right) x_i = 0,
\]

for each \(i \geq 0\).

We also define the following.

**Definition 3.2.** If \(f(z) = z^m + \sum_{j=0}^{m-1} c_j z^j\) and the sequence \(x = [x_0, x_1, \ldots, x_{N-1}]\) satisfies

\[
x_{i+m} = c_{m-1}x_{i+m-1} + c_{m-2}x_{i+m-2} + \cdots + c_1 x_{i+1} + c_0 x_i,
\]

for each \(i \geq 0\), then we say that the polynomial \(f(z)\) generates the sequence \(x\) (or \(x\) is generated by \(f(z)\)).

Henceforth, we denote by 0 (1, respectively) any sequence of any length which contains only zeroes (ones, respectively). The following corollaries are implications of Lemma 3.1

**Corollary 3.3.** For any nonzero polynomial \(f(z)\) and any cyclic nonzero sequence \(x\), \(f(E)x = 0\), if and only if the sequence \(x\) is generated by \(f(z)\).

**Corollary 3.4.** If \(x\) is a nonzero cyclic sequence generated by the nonzero polynomial \(f(z)^m\) and \(f(E)^{m-1}x = y\), then \(x\) is not generated by \(f(z)^{m-1}\) if and only if the sequence \(y\) is a nonzero sequence generated by \(f(z)\).

The sequence \(x\) generated by the polynomial \(f(z)\), can be described in terms of a linear feedback shift-register sequence with a feedback function \(x_{m+1} = f(x_1, x_2, \ldots, x_m)\), where \((x_1, x_2, \ldots, x_m)\) can be any binary \(m\)-tuple. The fundamental theory of shift-register sequences is given in [19].

**Definition 3.5.** A polynomial \(f(z)\) is irreducible if it has no nontrivial divisor polynomial of a smaller degree. An irreducible polynomial of degree \(k\) is called primitive if its nonzero roots are generators (primitive elements) of the field \(\mathbb{F}_{2^k}\).
The nonzero cyclic sequence $x$ generated by a primitive polynomial of degree $k$ has length $2^k - 1$, and each nonzero $k$-tuple appears exactly once in a window of length $k$ in $x$. Such a sequence is called an $m$-sequence for maximal length linear shift-register sequence [19]. The period of sequences generated by other irreducible polynomials (which are not primitive) can be also calculated by using the theory in [19]. For this purpose, the following definition is given.

**Definition 3.6.** The exponent of a polynomial $f(z)$ is the smallest integer $e$ such that $f(z)$ divides $z^e - 1$.

There is a connection between the exponent of any polynomial $f(z)$ and the periods of the related sequences generated by its shift-register. It does not give immediately the period of the sequences for every polynomial $f(z)$, but it does when $f(z)$ is an irreducible polynomial, which is described by the following well-known theorem.

**Theorem 3.7.** If $f(z)$ is an irreducible polynomial, then the nonzero sequences which it generates have period which is equal to the exponent of $f(z)$.

Each sequence $x$ can be generated by several distinct polynomials, but it appears that the structure of the polynomials which generate $x$ is determined by the polynomial of least degree which generates $x$. Therefore, we define the following.

**Definition 3.8.** For a sequence $x$, the polynomial $f(E)$ is a minimal zero polynomial for $x$ if $f(E)$ is a polynomial of the least degree such that $f(E)x = 0$.

In other words, in view of Corollary 3.3, $f(z)$ is a polynomial of least degree which generates $x$. The polynomial $f(z)$ will be called a minimal (connection) polynomial that generates $x$. We know that there exists at least one polynomial that generates a cyclic sequence of period $N$, namely $z^n - 1$. Therefore, we know the existence of a minimal polynomial that generates a sequence $x$. However, in order to show the uniqueness of the minimal polynomial, we need the following lemma.

**Lemma 3.9.** If the two polynomials $f(z)$ and $g(z)$ generate the same sequence $x$, then $h(z) = \gcd(f(z), g(z))$ also generates $x$, where $\gcd(\alpha(z), \beta(z))$ is the the greatest common divisor of the polynomials $\alpha(z)$ and $\beta(z)$.

**Proof.** By Corollary 3.3, $f(z)$ generates $x$ if and only if $f(E)x = 0$. Similarly, $g(z)$ generates $x$ if and only if $g(E)x = 0$. By the Euclidean algorithm, there
exists two polynomials \( a(z) \) and \( b(z) \) such that \( h(z) = a(z)f(z) + b(z)g(z) \). Hence, \( h(E)x = a(E)f(E)x + b(E)g(E)x \), and since \( f(E)x = 0 \) and \( g(E)x = 0 \), it follows that \( h(E)x = 0 \). Therefore, by Corollary 3.3, the polynomial \( h(z) \) generates \( x \).

**Corollary 3.10.** If \( x \) is a nonzero sequence, then it has a unique minimal zero polynomial.

### 3.2 Linear Complexity Computation by \( \text{gcd} \)

One conventional way to compute the linear complexity of the sequence \( x \) is using the greatest common divisors of two related polynomials as follows. Let \( x(z) \) be the generating function of \( x \) considered as an infinite sequence, defined by

\[
x(z) = x_0 + x_1 z + x_2 z^2 + x_3 z^3 + \cdots,
\]

or by its periodic sequence

\[
x^N(z) = x_0 + x_1 z + x_2 z^2 + x_3 z^3 + \cdots + x_{N-1} z^{N-1}.
\]

The generating function \( x(z) \) can be written as

\[
x(z) = \frac{x^N(z)}{1 - z^N} = \frac{x^N(z)/\gcd(x^N(z), 1 - z^N)}{(1 - z^N)/\gcd(x^N(x), 1 - z^N)} = \frac{g(z)}{f_x(z)},
\]

where

\[
g(z) = x^N(z)/\gcd(x^N(z), 1 - z^N),
\]

\[
f_x(z) = (1 - z^N)/\gcd(x^N(z), 1 - z^N).
\]

Obviously, \( \gcd(g(z), f_x(z)) = 1 \), \( \text{deg} \ g(z) < \text{deg} \ f_x(z) \), the polynomial \( f_x(E) \) is the minimal zero polynomial of \( x \), and \( \text{deg} \ f_x(z) = c(x) \) [12].

Since the sequence \( x \) can be any binary sequence of period \( N \), it follows that computing the linear complexity of \( x \) is equivalent to the computation of the degree of the greatest common divisor of \( x^N(z) \) and \( z^N - 1 \). Finding the minimal zero polynomial of \( x \) is equivalent to the computation of greatest common divisor of \( x^N(z) \) and \( z^N - 1 \). Hence, our algorithm can be regarded as an algorithm for the greatest common divisor of some cases.
The computation of the linear complexity in \([10, 11, 24, 31, 32]\) is based on these computations of the greatest common divisor. Our method is different as it is based on the direct definition of the linear complexity and on simple computations as done in the well-known Games-Chan Algorithm \([18]\).

### 3.3 Games-Chan Algorithm

Binary sequences of period \(2^n\) are very important among all binary sequences. The linear complexity \(c(x)\) of a binary sequence \(x = [L \ R]\) of period \(2^n\), where \(L\) and \(R\) are sequences of length \(2^{n-1}\), can be recursively computed by The Games-Chan Algorithm as follows \([18]\): when \(L + R = 0\), then \(c(x) = c(L)\), where the addition of two sequences \(L\) and \(R\) of the same length is defined as the bitwise addition for each position; otherwise we set \(c(x) = 2^{n-1} + c(L + R)\). This algorithm is described in more details as follows.

**The Games-Chan Algorithm:**

The input to the algorithm is a sequence \(x\) of period \(2^n\). If \(x \neq 0\), the complexity \(c\) of \(x\) is computed recursively as follows. Initially, set \(c_n = 0\) and \(A_n = x\). At a typical step of the algorithm the left half of \(A_m\), \(L(A_m) = [b_0, \ldots, b_{2^m-1}]\), is added to the right half, \(R(A_m) = [b_{2^m-1}, \ldots, b_{2^{m-1}-1}]\), the result being a sequence \(B_m\), of length \(2^{m-1}\). If \(B_m = 0\), \(A_m\) is replaced by \(A_{m-1} = L(A_m)\), and the complexity is left unchanged, i.e., \(c_{m-1} = c_m\). If \(B_m \neq 0\), \(A_m\) is replaced by \(A_{m-1} = B_m\), and \(c_m\) is replaced by \(c_{m-1} = c_m + 2^{m-1}\). The complexity of \(x\) is given by \(c(x) = c_0 + 1\).

The number of recursive steps in the algorithm is \(n\) for a sequence of period \(N = 2^n\), and this is the number of integer computations (on integers of length at most \(n\) bits) to obtain \(c(x)\). Since each such integer addition is for a distinct power of two, no more than a total of \(n\) bit operations are required. The algorithm also has to make at most \(N\) bit computations (comparisons or additions) for a sequence of period \(N = 2^n\). The reason is that in the first step \(2^{n-1}\) such operations are required; and the number of operations is reduced by half from one step to the following step. Thus, we have the following theorem.

**Theorem 3.11.** The complexity in The Games-Chan Algorithm is at most \(N + n\) bit operations, for a sequence of period \(N = 2^n\).
3.4 A General Method for any Binary Sequence

Let $x$ be a binary sequence of period $N$, which implies that $(E^N - 1)x = 0$. Assuming that we can factorize $z^N - 1$ (or equivalently $E^N - 1$) efficiently into irreducible factors, we want to find the minimal zero polynomial $g(E)$ of $x$, i.e. the smallest factor of $E^N - 1$ for which $g(E)x = 0$ (or equivalently the factor of $z^N - 1$, with the smallest degree, which generates $x$). For this purpose it is required to have an efficient algorithm for the factorization of the polynomial $z^N - 1$. This factorization is very important for other applications too. For example, each irreducible factor of $z^N - 1$ determines a binary cyclic code of length $N$. The overall complexity of a general algorithm depends on the complexity of this factorization. For many parameter this can be the bottleneck of the algorithm. For some information on the factorization of $z^N - 1$, the reader is referred to [22] and reference therein.

Let $z^N - 1 = q_1(z)^{\alpha_1}q_2(z)^{\alpha_2}\ldots q_t(z)^{\alpha_t}$, where the $q_i(z)$’s are distinct irreducible polynomials and $1 \leq \alpha_i \leq 2^{\gamma_i}$ for some nonnegative integer $\gamma_i$, $1 \leq i \leq t$. We want to find the polynomial of the smallest degree $g(z) = q_1(z)^{\delta_1}q_2(z)^{\delta_2}\ldots q_t(z)^{\delta_t}$, such that $g(E)x = 0$, where $0 \leq \delta_i \leq \alpha_i$, $1 \leq i \leq t$. Since at least one of the $\delta_i$’s is nonzero, we assume without loss of generality that $\delta_i \neq 0$. In order to do that, we need to do several observations below.

**Definition 3.12.** For any irreducible polynomial $q(z)$, let $S(q(z))$ be the set of all nonzero cyclic sequences generated by $q(z)$.

For example, $S(z^4 + z^3 + z^2 + z + 1) = \{00011, 01010, 11011\}$.

**Lemma 3.13.** Let $q(z)$ be an irreducible polynomial and $y \in S(q(z))$. Then for any polynomial $f(z)$, such that $\gcd(q(z), f(z)) = 1$, we have that $f(E)y \in S(q(z))$.

**Proof.** Suppose that $x = f(E)y$, for some sequence $y \in S(q(z))$. Assume first that $x = 0$. It implies by Corollary 3.3 that $f(z)$ generates $y$. Since $q(z)$ also generates $y$, it follows by Lemma 3.9 that $\gcd(q(z), f(z))$ generates $y$. But $\gcd(q(z), f(z)) = 1$ and hence $\gcd(q(z), f(z))$ does not generates $y$. Thus, $x$ is a nonzero sequence.

Since $y \in S(q(z))$ it follows that

$$q(E)x = q(E)(f(E)y) = f(E)(q(E)y) = f(E)0 = 0.$$
Since $x$ is a nonzero sequence, it follows that $x = f(E)y \in S(q(z))$. □

Lemma 3.13 leads to two interesting consequences.

**Corollary 3.14.** If $q(z)$ is an irreducible polynomial and $y$ is a nonzero sequence generated by $q(z)$, then for any polynomial $f(z)$, $f(E)y = 0$ if and only if $f(z)$ is divisible by $q(z)$.

**Proof.** If $f(z)$ is divisible by $q(z)$, i.e. $f(z) = h(z)q(z)$, then

$$f(E)y = h(E)q(E)y = h(E)0 = 0.$$  

Now, suppose that $f(E)y = 0$ and assume that $f(z)$ is not divisible by $q(z)$. Since $q(z)$ is an irreducible polynomial, it implies that $\gcd(q(z), f(z)) = 1$. Therefore, by Lemma 3.13, $f(E)y \in S(q(z))$ and hence $f(E)y \neq 0$, a contradiction. Thus, $f(z)$ is divisible by $q(z)$. □

The second consequence from Lemma 3.13 is the well known Shift and Add property for m-sequences [19]. The version given here generalizes the one which is usually used.

**Corollary 3.15.** Let $x$ be an m-sequence generated by a polynomial $q(z)$. If $f(z)$ is any other polynomial for which $\gcd(q(z), f(z)) = 1$, then $f(E)x = x$.

**Proof.** Since $x$ is an m-sequence, it follows that $q(z)$ is a primitive polynomial and $x$ is the only sequence in $S(q(z))$. The claim is now an immediate consequence from Lemma 3.13. □

Given a certain polynomial $f(z)$, now we are interested to know how the minimal polynomial of a sequence $f(E)x$ changes as compared to the original minimal polynomial of $x$. This next theorem analyzes the connection between the minimal polynomial of $x$ and $f(z)$.

**Theorem 3.16.** Let $x$ be a binary sequence, whose minimal polynomial is $g(z) = q_1(z)^{\delta_1} \cdots q_t(z)^{\delta_t}$, where the $q_i$'s are distinct irreducible polynomials and $\delta_i \geq 1$, $1 \leq i \leq t$. Then,

$$q_1(E)^{\delta_1}q_2(E)^{\delta_2} \cdots q_{t-1}(E)^{\delta_{t-1}}q_t(E)^{\delta_t-1}x \in S(q_t(z)).$$
Furthermore, for every \(1 \leq i \leq t-1\), let \(d_i\) be an integer such that \(d_i \geq \delta_i\). Then,

\[ q_1(t)q_2(t)^d_2 \ldots q_{t-1}(t)^{d_{t-1}}t_1(t)^{d_1-1}x \in S(t(z)). \]

Proof. If \(x' = q_1(t)^{\delta_1}q_2(t)^{\delta_2} \ldots q_{t-1}(t)^{\delta_{t-1}}t_1(t)^{\delta_1-1}x\), then since \(g(z)\) is the minimal polynomial of \(x\), it follows that \(x' \neq 0\) and \(q_1(t)x' = 0\). This implies that \(x' \in S(t(z))\). Since \(x' \neq 0\), \(q_1(t)x' = 0\), and

\[ \gcd(1(t)q_2(t)^{d_2-\delta_2} \ldots q_{t-1}(t)^{d_{t-1}-\delta_{t-1}}, q_1(t)) = 1, \]

it follows by Lemma 3.13 that

\[ q_1(t)^{d_1-\delta_1}q_2(t)^{d_2-\delta_2} \ldots q_{t-1}(t)^{d_{t-1}-\delta_{t-1}}x' \in S(t(z)), \]

and hence

\[ q_1(t)^{d_1}q_2(t)^{d_2} \ldots q_{t-1}(t)^{d_{t-1}}t_1(t)^{d_1-1}x \in S(t(z)). \]

\(\square\)

Now, for our given sequence \(x\) of period \(N\), we want to find the smallest \(\delta_i \geq 1\) such that \(q_1(t)^{\delta_1} \ldots q_{t}(t)^{\delta_{t}}x = 0\). By Theorem 3.16 it is sufficient to find the smallest \(\delta_i \geq 1\), such that

\[ q_1(t)^{2^{\gamma_1}}q_2(t)^{2^{\gamma_2}} \ldots q_{t-1}(t)^{2^{\gamma_{t-1}}}q(t)^{\delta_1-1}x \in S(t(z)), \]

for some integer \(\gamma_i\) where \(0 \leq \delta_i \leq 2^{\gamma_i}\). From our discussion so far, we can restate again our problem as follows.

**Problem 1.** Suppose that \(x\) is a binary sequence with period \(N\). Let the factorization of \(z^N - 1 = q_1(z)^{\alpha_1}q_2(z)^{\alpha_2} \ldots q_t(z)^{\alpha_t}\), where the \(q_i(z)\)'s are distinct irreducible polynomials and \(1 \leq \alpha_i \leq 2^{\gamma_i}\) for some nonnegative integer \(\gamma_i, 1 \leq i \leq t\). We want to find the unique polynomial of the smallest degree \(g(z) = q_1(z)^{\delta_1}q_2(z)^{\delta_2} \ldots q_t(z)^{\delta_t}\), such that \(g(x) = 0\), where \(0 \leq \delta_i \leq \alpha_i \leq 2^{\gamma_i}, 1 \leq i \leq t\).

Solving Problem 1 is equivalent to finding \(\delta_i\) for all \(1 \leq i \leq t\). Therefore we provide the following Algorithm 1 to find \(\delta_i\) for any \(1 \leq i \leq t\).
Algorithm 1 Find $\delta_i$ of $x$

1: Set $y \leftarrow q_1(E)^{2^1} q_2(E)^{2^2} \ldots q_{i-1}(E)^{2^i-1} q_{i+1}(E)^{2^i+1} \ldots q_t(E)^{2^t} x$.
2: if $y = 0$ then
3: $\delta_i \leftarrow 0$
4: else
5: $y' \leftarrow y$ and $\delta_i \leftarrow 0$ (if $\delta_i < 2^\gamma_i$ in this case).
6: for each $j$ from 1 to $\gamma_i$ do
7: Compute $A_j \leftarrow q_i^{2^\gamma_i-j}(E)y'$
8: if $A_j = 0$ then
9: proceed to the next iteration
10: else
11: Set $y' \leftarrow A_j$, $\delta_i \leftarrow \delta_i + 2^\gamma_i - j$.
12: end if
13: end for
14: Set $\delta_i \leftarrow \delta_i + 1$ (as $y' \in S(q_t(x))$).
15: end if

Remark 3.1. The time needed to compute $A_j$ in Line 7 varies depending on the polynomial $q_i$ and the current length of $y'$. The more summands that the irreducible polynomial $q_i$ has, the more time it takes to compute $A_j$. After each iteration in the for-loop in Line 6, it is possible that the period of the cyclic sequence $y'$ shrinks after applying $q_i^{2^\gamma_i-j}(E)$ on $y'$. Therefore, the implementation of Algorithm 1 can be done more efficiently as will be shown in Section 3.5 and 3.6.

3.5 Powers of Irreducible Polynomial

The Games-Chan Algorithm can be generalized in a trivial way to sequences generated by a power of an irreducible polynomial. If the primitive polynomial $f(z)$ has degree $k$, then the period of such sequences is given by the following lemma.

Lemma 3.17. Let $f(z)$ be a primitive polynomial of degree $k$. The polynomial $f(z)^m$ generates the all-zero sequence and $2^{(\ell-1)k-i}$ sequences whose period are $(2^k-1) \cdot 2^i$, for each $1 \leq \ell \leq m$, where $i = \lceil \log \ell \rceil$.

Lemma 3.17 is a generalization of a similar result for $f(z) = z+1$ [15]. Sequences of period $(2^k-1) \cdot 2^i$ can have also other minimal zero polynomials and they will be discussed later, as the linear complexity of sequences with these periods should be computed by the algorithm presented in this thesis. At this point we will consider
the linear complexity of such sequences which are generated by a polynomial \( f(z)^m \), for an irreducible polynomial \( f(z) \) i.e., we would like to find the minimal \( m \) for such sequences of period \( c \cdot 2^n \), where \( c \) is the exponent of \( f(z) \). This will be done with an algorithm which is a straightforward generalization of The Games-Chan Algorithm.

**Power of Irreducible Polynomial (PIP) Algorithm:**

The input for the algorithm is an irreducible polynomial \( f(z) \) of degree \( k \) and a nonzero sequence \( x \) of length \( c \cdot 2^n \) whose minimal zero polynomial is \( f(E)^m \) for some unknown \( m \), and \( c \) is the exponent of \( f(z) \). The output of the algorithm is \( m \), such that \( f(E)^m \) is the minimal zero polynomial of \( x \).

We are looking for the minimal \( m \) such that \( f(E)^m x = 0 \). By Corollary 3.4, it implies that \( f(E)^{m-1} x \in S(\mathbb{E}) \). Hence, if \( n = 0 \) then \( m = 1 \) (this will be recognized by the formal steps of the algorithm) and the algorithm comes to its end. The algorithm is described in Algorithm 2.

**Algorithm 2 PIP Algorithm**

1: Input : Binary sequence \( x \), and irreducible polynomial \( f(z) \).
2: Output : \( m \) such that \( f(z)^m \) is the minimal zero polynomial of \( x \).
3: Initiate \( x_n \leftarrow x \) and \( m_n \leftarrow 0 \)
4: for \( j \) from 1 to \( n \) do
5: \hspace{1em} Let \( [\mathcal{L}_{n-j+1} \mathcal{R}_{n-j+1}] \leftarrow x_{n-j+1} \), where \( |\mathcal{L}_{n-j+1}| = |\mathcal{R}_{n-j+1}| \).
6: \hspace{1em} Initiate \( m_n = 0 \).
7: \hspace{1em} Compute \( x' \leftarrow f(E)^{2n-j} x_{n-j+1} \), where \( |x'| = c \cdot 2^{n-j} \).
8: \hspace{1em} if \( x' \neq 0 \) then
9: \hspace{2em} Set \( m_{n-j} \leftarrow m_{n-j+1} + 2^{n-j} \), \( x_{n-j} \leftarrow x' \).
10: \hspace{1em} else
11: \hspace{2em} Set \( m_{n-j} \leftarrow m_{n-j+1} \), \( x_{n-j} \leftarrow \mathcal{L}_{n-j+1} \).
12: \hspace{1em} end if
13: end for
14: Set \( m \leftarrow m_0 + 1 \)

**Remark 3.2.** Line 11 is because if \( x = 0 \), then \( m - 1 < m_{n-j+1} + 2^{n-j} \). Line 14 is because after the last iteration of for-loop, we must have \( x_0 \in S(f(z)) \). The linear complexity of the input sequence \( x \) is therefore \( km \), if the degree of \( f(z) \) is \( k \).

The PIP algorithm is interesting for itself, but it is important in the implementation of the other algorithms presented in this work. Some of this implementation is given in Appendix A. Note that the complexity of the PIP algorithm depends on the input irreducible polynomial.
3.6 Implementation for some Periods $p \cdot 2^n$, $p$ prime

We are now in a position to implement our method for specific lengths of sequences, $\ell \cdot 2^n$, where $\ell$ is odd. We will consider as examples the cases in which $\ell = p$ is a prime such that $p = 3$ or $p \equiv 1(\mod 4)$, where 2 is a generator modulo $p$, and especially $p = 5$. We will concentrate on the number of bit operations required in these cases and compare them with the number of bit operations required by the method of Meidl [24]. The explanations for the exact steps in the implementation are long and sometimes it is required to split the sequences into a few parts to apply some tricky computations. The exact computations and proof of Theorem 3.18, part 1, part 2 and part 3 are given in Appendix A. We summarize the results in the appendix with the following concluding theorem.

**Theorem 3.18.** Let $x$ be a binary sequence of period $N$ on which the algorithm is applied.

1. If $N = 3 \cdot 2^n$ then $7 \cdot 2^n + 2n$ bit operations are required to implement the algorithm.

2. If $N = 5 \cdot 2^n$ then $16\frac{3}{4} \cdot 2^n + 2n$ bit operations are required to implement the algorithm.

3. If $N = p \cdot 2^n$, where $p \equiv 1(\mod 4)$ and 2 is a generator modulo $p$, then $\frac{p^2 + 7p + 7}{4} \cdot 2^n + 2n$ bit operations are required to implement the algorithm.

As for comparison with the method of Meidl [24], one can verify that for binary sequences of period $3 \cdot 2^n$, the implementation requires $8 \cdot 2^n + 4n$ bit operations, while for sequences of period $5 \cdot 2^n$, the implementation requires $20 \cdot 2^n + 10n$ bit operations. For general period $\ell \cdot 2^n$ it is too difficult to compare since it is difficult to compute the exact number of bit operations required in the algorithm of [24]. Therefore, this case will not be considered in this work. We just note that in some cases the algorithm in [24] is more efficient, e.g. for $N = 7 \cdot 2^n$.

The next section will discuss an implementation of Algorithm 1 for sequences with odd period, which is something that is not covered in Meidl’s paper.
Chapter 3. Linear Complexity

3.7 Implementation for Sequences with Odd Period

When we are given a binary sequence of odd period, the situation is more complicated and our algorithm is not efficient in many cases. This is because the factorization of \( x^N - 1 \) where \( N \) is the period of the binary sequence may be tedious and consists of many distinct irreducible polynomials. However, we consider two cases in which our algorithm is efficient for a binary sequence \( x \) of odd period \( \ell \).

**Case 1.** The input binary sequence \( x \) is of odd period \( \ell \), where \( \ell = p^n \), \( p \) prime, and 2 is a generator modulo \( p \).

In regards to Case 1, the factorization of \( z^\ell - 1 \) is easy with the help of the following well-known theorem.

**Theorem 3.19.** [21]. The polynomial \( \sum_{j=0}^{p-1} z^j \) is irreducible if and only if 2 is a generator modulo \( p \). Furthermore, if 2 is a generator modulo \( p \), for any given integer \( i \geq 0 \) the polynomial \( \sum_{j=0}^{p-1} z^{jp^i} \) is also irreducible.

Now, one can verify that \( z^{p^n} - 1 \) is a multiplication of \( n+1 \) irreducible polynomials, where the \( i \)th polynomial is \( g_i(z) = \sum_{j=0}^{p-1} z^{jp^i}, \) \( 0 \leq i \leq n - 1 \), i.e. \( z^{p^n} - 1 = (z + 1) \prod_{i=0}^{n-1} g_i(z) \). In this case, our Algorithm 1 can be implemented in a simpler and more efficient way in Algorithm 3.

**Remark 3.3.** At each iteration in Line 4, we have to check if \( x' \) whose length is \( p^{n-m+1} \) has period \( p^{n-m+1} \). For this, it is sufficient in Line 6 to just check if the first \( (p-1)p^{n-m} \) bits of \( y := (E^{p^{n-m}} + 1)x' \) are equal to \( 0 \) instead of checking if the whole sequence \( y = 0 \). Therefore in Line 5, only \( (p-1)p^{n-m} \) bit operations are required. Since Line 9 also requires \( (p-1)p^{n-m} \) bit operations, the total number of bit operations required by the algorithm (omitting the additions to compute \( c \)) is \( 1 + \sum_{m=1}^{n} 2(p-1)p^{n-m} = 1 + (2p^n - 2) \leq 2p^n = 2N \).

**Case 2.** The input binary sequence \( x \) is of odd period \( \ell \), where \( \ell = p_1^{n_1} p_2^{n_2} \cdots p_t^{n_t} \), \( p_i \) prime, 2 is a generator modulo \( p_i \), \( n_i \) is a positive integer. This case is solved similarly to Case 1, by considering \( t \) steps, one for each prime.
Algorithm 3 Implementation on Case 1

1: Input : Binary sequence $x$ of length $p^n$, and irreducible polynomial $f(z)$.
2: Output : minimal zero polynomial $f(z)$, linear complexity $c$ of $x$.
3: Initiate : $x' \leftarrow x$, $c \leftarrow 0$, and $f(z) \leftarrow 1$.
4: for $m$ from 1 to $n$ do
5: Let $y \leftarrow (E^{p^n-m} + 1)x'$.
6: if $y = 0$ then
7: $x' \leftarrow$ the first $p^n-m$ bits of $x'$.
8: else
9: $c \leftarrow c + (p-1)p^{n-m}$, $f(z) \leftarrow g_{n-m}(z)f(z)$ and $x' \leftarrow g_{n-m}(E)x'$.
10: end if
11: end for
12: if $x' = 1$ then
13: $c \leftarrow c + 1$ and $f(z) \leftarrow (z + 1)f(z)$
14: end if

3.8 Conclusion

We propose an algorithm that is a direct generalization of The Games-Chan Algorithm, namely the Powers of Irreducible Polynomial Algorithm as stated in Algorithm 2. This algorithm takes a binary sequence and a possible irreducible polynomial as inputs and outputs the power of the irreducible polynomial that generates the input sequence. Furthermore, we also present a general idea to find the linear complexity of any binary sequence in Algorithm 1. However, in many cases the implementation of the algorithm may not take time linear in length of the input sequence.

Nevertheless, we provide some cases, where the implementation is efficient. In fact, in some cases of periodic sequence of period $N$, the time complexity is $\beta N$, for a constant $\beta$ that is smaller than the constant from the previous best known algorithm of Meidl [24], as described in Section 3.6. However, the proof of the efficiency of the implementation for Theorem 3.18 is moved to Appendix A. Furthermore, our algorithm also outputs the minimal zero polynomial that generates the input sequence. This feature does not exist in the algorithm of Meidl. Lastly, we also show that the implementation/modification of Algorithm 1 is efficient in some cases of sequences with odd period in Section 3.7, as described in Algorithm 3.
Chapter 4

The $k$-deck Problem

In this chapter, our main goal is to compute or estimate $\text{Deck}_k(n)$, i.e. the number of sequences of length $n$ with distinct $k$-decks. In Section 4.2, we provide a trellis-based method to compute $\text{Deck}_k(n)$. We also provide the exact values of $\text{Deck}_k(n)$ for $k \in \{3, 4, 5, 6\}$ and $n \leq 30$. Furthermore, using tools from linear algebra, we provide asymptotic estimates of $\text{Deck}_k(n)$ for the case where $k$ is fixed. In particular, we improve the asymptotic upper bound to $O(n^{\Delta(k)/4+1/2})$ for $k \geq 2$ in Section 4.3. Lastly, in [49], Manvel designed a matrix $M^{(k,w)}$, and is now used in the construction of the lower bound of $\text{Deck}_k(n)$ in Section 4.4. Part of this chapter is already presented in the 2019 19th International Symposium on Communications and Information Technologies (ISCIT) in [36]. The result of this chapter has been accepted for publication in Advances in Mathematics of Communications.

4.1 Preliminaries

Let $x = x_1 x_2 \cdots x_n$ be a binary sequence of length $n$. For $A \subseteq \{1, 2, \ldots, n\}$, we use $x_A$ to denote the subsequence with indices in $A$. In other words, $x_A = x_{a_1} x_{a_2} \cdots x_{a_k}$ where $a_1 < a_2 < \cdots < a_k$ and $A = \{a_1, a_2, \ldots, a_k\}$. For $k \leq n$, the $k$-deck of $x$, denoted by $\text{Deck}_k(x)$, refers to the multiset of all $\binom{n}{k}$ subsequences of length $k$. We represent the $k$-deck of a sequence $x$ by an integer-valued vector of length $2^k$. Specifically, $\text{Deck}_k(x) \triangleq (x_\alpha)_{\alpha \in \{0,1\}^k}$, where $x_\alpha$ denotes the number of occurrences of $\alpha$ as a subsequence of $x$ and the indices in $\{0, 1\}^k$ are presented in an increasing lexicographic order.
Example 4.1. Let $x = 110011$. Then $x_{(3,5,6)} = x_{(4,5,6)} = 011$ and we check that $x_{011} = 2$. Furthermore,

$$
\begin{align*}
\text{Deck}_1(x) &= (x_0, x_1) = (2, 4), \\
\text{Deck}_2(x) &= (x_{00}, x_{01}, x_{10}, x_{11}) = (1, 4, 4, 6), \text{ and} \\
\text{Deck}_3(x) &= (x_{000}, x_{001}, x_{010}, x_{011}, x_{100}, x_{101}, x_{110}, x_{111}) = (0, 2, 0, 2, 8, 2, 4).
\end{align*}
$$

Definition 4.1. Two sequences $x$ and $y$ are said to be $k$-equivalent, or $x \sim_k y$ if their $k$-decks are the same, i.e. $\text{Deck}_k(x) = \text{Deck}_k(y)$.

Example 4.2. Let $x = 110011$ and $y = 101101$. Then $\text{Deck}_1(y) = (2, 4)$, $\text{Deck}_2(y) = (1, 4, 4, 6)$ and $\text{Deck}_3(y) = (0, 1, 2, 3, 1, 6, 3, 4)$. Hence, $x \sim_k y$ for $k \in \{1, 2\}$, but $x \not\sim_3 y$.

It can be shown that $\sim$ defines an equivalence relation on all binary sequences. Furthermore, if $x$ and $y$ have the same $k$-deck, then the lengths of $x$ and $y$ are necessarily the same. Hence, we fix $n$ and partition the binary sequences of length $n$ using the relation $\sim_k$. Then we set $\text{Deck}_k(n)$ to be the resulting number of equivalence classes. In this chapter, we determine the exact value of $\text{Deck}_k(n)$ for $k \in \{3, 4, 5, 6\}$ and $k \leq n \leq 30$ and provide asymptotic estimates of $\text{Deck}_k(n)$ when $k$ is fixed.

For $k \in \{1, 2\}$, the exact values on $\text{Deck}_k(n)$ and characterization of $\text{Deck}_k(x)$ have been determined [51, Lemma 4].

Proposition 4.2. Suppose that $x$ is a binary sequence of length $n$.

1. Then $\text{Deck}_1(x) = (n - w, w)$, where $x_1 = w$. Therefore, $\text{Deck}_1(n) = n + 1$.

2. Then $\text{Deck}_2(x) = (\binom{n-w}{2}, t, w(n-w) - t, \binom{w}{2})$, where $x_1 = w$ and $x_{01} = t$. Therefore, we have that $\text{Deck}_2(n) = (n^3 + 5n + 6)/6$.

For $k \geq 3$, the best known upper bound on $\text{Deck}_k(n)$ is below.

Theorem 4.3 (Rigo and Salimov [51, Proposition 2]). For all $n \geq k$, we have that $\text{Deck}_k(n) \leq \prod_{j=1}^{k} \left( \binom{n}{j} + 1 \right)^{2^{j-1}}$. When $k$ is fixed, we have that $\text{Deck}_k(n) = O \left( n^{2(k-1)2^k+1} \right)$.
Theorem 4.4 (Manvel et al. [49, Lemma 1]). If $x$ and $y$ are two binary sequences such that $x \sim_k y$, then $x \sim_{k-i} y$ for all $0 \leq i \leq k - 1$.

In addition to Deck$_k(n)$, we define $S(k) \triangleq \min\{n : \text{Deck}_k(n) < 2^n\}$. The exact values of $S(k)$ have been determined for $k \in \{3, 4, 5\}$ (see [38] for a survey). The first open case is $S(6)$ and the best known upper bound is given by Manvel et al. who constructed a pair of sequences of length thirty with the same 6-deck [49, Example 4].

Theorem 4.5. $S(6) \leq 30$.

For completeness, we present the best known upper bounds for $S(k)$ that were summarized in [38].

Theorem 4.6. We have that $S(k) = \Omega(k^2)$ and that

$$S(k) \leq \begin{cases} 
1.75 \cdot 1.62^k, & \text{for } 7 \leq k \leq 28, \\
0.25 \cdot 1.17^k k^3 \log k, & \text{for } 29 \leq k \leq 84, \\
3^{(3/2+o(1)) \log^2 k}, & \text{for } k \geq 85.
\end{cases}$$

In the next section, we will use a trellis structure to describe the $k$-decks and using this insight, we then provide an algorithm that enumerates all $k$-decks efficiently.

### 4.2 Polynomial-Time Enumeration

In this section, we introduce a trellis-based algorithm that calculates Deck$_k(n)$ for a fixed value of $k$. When $k$ is a constant, the technique computes Deck$_k(n)$ in polynomial time. We then compute Deck$_k(n)$ for $k \in \{3, 4, 5, 6\}$ and $k \leq n \leq 30$ and establish that $S(6) = 30$. To compute Deck$_k(n)$, we construct a trellis with levels $i \in \{k, k+1, \ldots, n\}$ such that we are able to compute Deck$_k(i)$ at level $i$. At level $i + 1$, instead of naively enumerating all $k$-decks for all possible $x \in \{0, 1\}^{i+1}$, our algorithm runs recursively and calculates the set $\{\text{Deck}_k(x) : x \in \{0, 1\}^{i+1}\}$ from the set $\{\text{Deck}_k(x) : x \in \{0, 1\}^i\}$, which reduces the complexity from $2^{i+1}$ down to Deck$_k(i)$.
To do so, we make two important combinatorial observations. In [51], it was observed that \( x \sim_{k} y \) implies \( x \sim_{s} y \) for all \( 1 \leq s < k \). The next proposition gives an explicit method to compute the \( s \)-deck from a \( k \)-deck. We remark that the proposition has appeared in [38] without proof and hence, we provide a proof here for completeness. Also, our proof illustrates the recursive nature of the trellis construction.

**Proposition 4.7** (Dudik and Schulman [38]). Let \( x \in \{0, 1\}^{n} \), \( \alpha \in \{0, 1\}^{s} \) with \( 1 \leq s < k \), then

\[
\binom{n-s}{k-s} x_{\alpha} = \sum_{\beta \in \{0,1\}^{k}} \beta_{\alpha} x_{\beta} \tag{4.1}
\]

*Proof.* For \( A \subseteq [n] \), recall that \( x_{A} \) denotes the subsequence of \( x \) with indices in \( A \). To demonstrate (4.1), we consider the two collections \( \mathcal{A} \) and \( \mathcal{B} \) of tuples. First, we set

\[
\mathcal{A} \triangleq \{ (A; S) : A, S \subset [n], x_{A} = \alpha, |S| = k-s, A \cap S = \emptyset \}.
\]

For each occurrence of \( \alpha \) in \( x \), we fix \( A \) and have \( \binom{n-s}{k-s} \) choices for \( S \). Therefore, the left hand side of (4.1) counts the number of tuples in \( \mathcal{A} \).

Next, we set

\[
\mathcal{B} \triangleq \{ (\beta; B; T) : B, T \subset [n], x_{B} = \beta, T \subseteq B, x_{B \setminus T} = \alpha \}.
\]

For each \( \beta \in \{0, 1\}^{k} \) and each occurrence of \( \beta \) in \( x \), we have \( \beta_{\alpha} \) choices for \( T \). Therefore, the right hand side of (4.1) counts the number of tuples in \( \mathcal{B} \).

To establish (4.1), it remains to exhibit a bijection between \( \mathcal{A} \) and \( \mathcal{B} \). Consider the map \( \phi : \mathcal{A} \rightarrow \mathcal{B} \) such that \( \phi(A; S) = (\beta; B; S) \), where \( B = A \cup S \) and \( \beta = x_{B} \).

For the inverse, we consider the mapping \( \psi : \mathcal{B} \rightarrow \mathcal{A} \) such that \( \psi(\beta; B; T) = (A; T) \), where \( A = B \setminus T \). It is not difficult to verify that both \( \phi \circ \psi \) and \( \psi \circ \phi \) are identity maps on their respective domains. Therefore, we establish (4.1). \( \square \)

For \( x \in \{0, 1\}^{n} \) and \( a \in \{0, 1\} \), recall from the preliminaries in Chapter 1 that \( xa \) denote the concatenation of \( x \) and \( a \). Our second observation states that we can compute \( \text{Deck}_{k}(xa) \) from \( \text{Deck}_{k-1}(x) \) and \( \text{Deck}_{k}(x) \). The following proposition was shown in Sakarovitch and Simon [52]. We provide a proof here, for completeness.
**Proposition 4.8** (Sakarovitch and Simon [52]). Let \( x \in \{0, 1\}^n, \beta \in \{0, 1\}^k \), we have

\[
\begin{align*}
(x0)_{\beta0} &= x_{\beta0} + x_{\beta}, \quad (4.2) \\
(x0)_{\beta1} &= x_{\beta1}, \quad (4.3) \\
(x1)_{\beta0} &= x_{\beta0}, \quad (4.4) \\
(x1)_{\beta1} &= x_{\beta1} + x_{\beta}. \quad (4.5)
\end{align*}
\]

**Proof.** Let \( x \in \{0, 1\}^n \). First we will show (4.2). Suppose \( \beta \in \{0, 1\}^k \). Consider the collection of index subsets:

\[ S = \{A \subset [n + 1] : (x0)_A = (\beta0)\}. \]

Then \( S \) can be written as a disjoint union of \( S_1 \) and \( S_2 \) where

\[ S_1 = \{A \in S : n + 1 \notin A\}, \quad S_2 = \{A \in S : n + 1 \in A\}. \]

Since \( (x0)_{\beta0} = |S|, x_{\beta0} = |S_1| \) and \( x_{\beta} = |S_2| \), we have (4.2). Now we want to show (4.3). Consider the collection of index subsets:

\[ S' = \{A \subset [n + 1] : (x0)_A x = (\beta1)\}. \]

Note that for every \( A \in S' \), \( n + 1 \notin A \), which implies that \( (x0)_{\beta1} = |S'| = |\{A \subset [n] : (x)_A = (\beta1)\}| = x_{\beta1} \), which is (4.3). Similarly, we can show (4.4) and (4.5). \( \square \)

A more general version of Proposition 4.8 was given in [51]. As the authors did not furnish a proof, we provide one here for completeness.

**Theorem 4.9.** Let \( x \) and \( y \) be binary sequences. Then \( x \sim_k y \) if and only if \( (x0)_k \sim (y0)_k \). Similarly \( x \sim_k y \) if and only if \( (x1)_k \sim (y1)_k \).

**Proof.** We first show the forward direction. Suppose that \( x \sim y \), which by Theorem 4.4 implies that \( x \sim_{k-1} y \). For any binary sequence \( \beta \) of length \( k-1 \), by (4.2), we have \( (x0)_{\beta0} = x_{\beta0} + x_{\beta} = y_{\beta0} + y_{\beta} = (y0)_{\beta0} \). Furthermore, for any binary sequence \( \beta \) of length \( k-1 \), by (4.3), we also have \( (x0)_{\beta1} = x_{\beta1} = y_{\beta1} = (y0)_{\beta1} \). For both
4.2. Polynomial-Time Enumeration

Figure 4.1: Trellis section for \( k = 2 \) and levels \( i \in \{ 4, 5 \} \). Left vertices belong to \( D_2(4) \), while right vertices belong to \( D_2(5) \). Blue solid edges correspond to label ‘0’, while red dashed edges correspond to label ‘1’.

In cases, it implies that \( (x_0)_k \sim_k (y_0) \).

On the other hand, suppose that \( (x_0)_k \sim_k (y_0) \) but \( x \not\sim_k y \). In other words, there exists a binary sequence \( \alpha \) of length \( k \), such that \( x_{\alpha} \neq y_{\alpha} \). If \( \alpha = (\beta_1) \), for some binary sequence \( \beta \) of length \( k - 1 \), then (4.3) implies that \( x_{\beta_1} = (x_0)_{\beta_1} = (y_0)_{\beta_1} = y_{\beta_1} \), which contradicts our assumption. That means \( \alpha = (\beta_0) \), for some binary sequence \( \beta \) of length \( k - 1 \). By (4.2), it implies that \( (x_0)_{\beta_0} - x_{\beta} = x_{\beta_0} \neq y_{\beta_0} = (y_0)_{\beta_0} - y_{\beta} \). Combined with the fact that \( (x_0)_{\beta_0} = (y_0)_{\beta_0} \), this implies \( x_{\beta} \neq y_{\beta} \), and hence
$x \not\sim_k y$. Furthermore, $(x_0) \sim_k (y_0)$ implies that $(x_0) \sim_{k-1} (y_0)$ by Theorem 4.4. And so our assumption of $(x_0) \sim_k (y_0)$ and $x \not\sim_k y$ implies that $(x_0) \sim_{k-1} (y_0)$ and $x \not\sim y$. Doing this inductively, we have that $(x_0) \sim_k (y_0)$ and $x \not\sim_k y$, which contradicts each other. Therefore our original assumption is wrong, and hence this proves that $(x_0) \sim_k (y_0)$ implies $x \sim_k y$.

We are ready to present our trellis. As mentioned earlier, the trellis has levels $i \in \{k, k+1, \ldots, n\}$. Each vertex at level $i$ represents a $k$-deck of some sequence of length $i$ and we denote the vertices at level $i$ with $\mathbb{D}_k(i)$. Therefore,

$$\mathbb{D}_k(i) = \{\text{Deck}_k(x) : x \in \{0, 1\}^i\}.\$$

Using (4.1) and Proposition 4.8, each vertex in $\mathbb{D}_k(i)$ is extended by two edges labeled ‘0’ and ‘1’ to two vertices in $\mathbb{D}_k(i+1)$. The resulting trellis is biproper, in other words, every vertex has exactly two outgoing arcs with distinct labels and at most two incoming arcs with distinct labels, which is due to Theorem 4.9. Furthermore, $\mathbb{D}_k(i)$ is the set of all $k$-decks of sequences of length $i$ while the set of paths to a vertex, or a $k$-deck, is the set of all binary sequences having this $k$-deck. See Figure 4.1 for a trellis section for $k = 2$ and levels $i \in \{4, 5\}$. A formal description of the enumeration method is detailed in Algorithm 4.

**Algorithm 4 Compute Deck_k(n)**

1: Initialize $\mathbb{D}_k(k) := \{(1, 0, \cdots, 0), (0, 1, \cdots, 0), \cdots, (0, \cdots, 0, 1)\}$
2: for $i = k$ to $n-1$ do
3: initialize $\mathbb{D}_k(i+1)$ as an empty set
4: for every $k$-deck Deck $\in \mathbb{D}_k(i)$ do
5: for $a \in \{0, 1\}$ do
6: Let $x$ be a sequence such that Deck_k(x) = Deck.
7: Using Eq. (4.1) with (4.2), (4.3), (4.4) or (4.5), compute Deck’ = Deck_k(xa)
8: if Deck’ $\notin \mathbb{D}_k(i+1)$ then
9: insert Deck’ to the set $\mathbb{D}_k(i+1)$
10: end if
11: end for
12: end for
13: end for
14: return Deck_k(n) = $|\mathbb{D}_k(n)|$

**Remark 4.1.** We discuss the computational complexity of Algorithm 4. First, we note that $x$ need not be explicitly found in Line 6. To compute Deck_k(xa), it suffices
4.3. Upper Bounds on Deck$_k(n)$

to apply (4.1) and Proposition 4.8 to Deck = Deck$_k(x)$. Also, since Equations (4.1), (4.2), (4.3), (4.4) and (4.5) involve sums with at most $2^k$ terms, Lines 6 and 7 take constant time.

The time complexity for Lines 8 and 9 depends on the data structure we used for D$_k(n)$. If we use a binary search tree, we can insert each “new” $k$-deck in $O(Deck_k(n) \log Deck_k(n))$ time using $O(Deck_k(n))$ space. Therefore, Algorithm 4 runs in $O(nDeck_k(n) \log Deck_k(n))$ time using $O(nDeck_k(n))$ space. For fixed $k$, since $Deck_k(n)$ is polynomial in $n$ by Theorem 4.3, the algorithm has space and time complexity polynomial in $n$.

To conclude this section, we compute the values of Deck$_k(n)$ for $k \in \{3, 4, 5, 6\}$ and $k \leq n \leq 30$ and present them in Table 4.1. In particular, we computed that Deck$_6(n) = 2^n$ for $n \leq 29$. Together with Theorem 4.5, we established the following.

**Theorem 4.10.** $S(6) = 30$.

In the next section, we will improve the asymptotic upper bound on Deck$_k(n)$. In particular, we will show that

$$Deck_k(n) = O \left( n^{(k-1)2^{k-1}+1} \right).$$

(4.6)

4.3 Upper Bounds on Deck$_k(n)$

Fix $k \geq 3$. In this section, we derive an upper bound on Deck$_k(n)$, and in particular Deck$_3(n)$. To this end, we fix some $(k-1)$-deck Deck’ and consider the collection $\mathcal{F}$ of all sequences of length $n$ whose $(k-1)$-deck is given by Deck’. Suppose that the number of $k$-equivalence classes in $\mathcal{F}$ is at most $U$ for all choices of $(k-1)$-decks. Then an upper bound for Deck$_k(n)$ is simply given by $Deck_k(n)U$.

Recall from Chapter 1 that the weight of a binary sequence is defined as the number of nonzero entries in this sequence. To find $U$, we consider an additional parameter $1 \leq m \leq k - 1$ and define the following

**Definition 4.11.** Let $J(k, m)$ be the set of all binary sequences of length $k$ and weight $m$. 
Table 4.1: Values of Deck\(_k(n)\) for 3 \(\leq k \leq 6\) and \(k \leq n \leq 30\). Values highlighted in bold correspond to Deck\(_k(S(k))\).

<table>
<thead>
<tr>
<th>(n\backslash k)</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>8</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>16</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>—</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>126</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>8</td>
<td>247</td>
<td>256</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>9</td>
<td>480</td>
<td>512</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>10</td>
<td>926</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
</tr>
<tr>
<td>11</td>
<td>1764</td>
<td>2048</td>
<td>2048</td>
<td>2048</td>
</tr>
<tr>
<td>12</td>
<td>3337</td>
<td>4092</td>
<td>4096</td>
<td>4096</td>
</tr>
<tr>
<td>13</td>
<td>6208</td>
<td>8176</td>
<td>8192</td>
<td>8192</td>
</tr>
<tr>
<td>14</td>
<td>11408</td>
<td>16328</td>
<td>16384</td>
<td>16384</td>
</tr>
<tr>
<td>15</td>
<td>20608</td>
<td>32604</td>
<td>32768</td>
<td>32768</td>
</tr>
<tr>
<td>16</td>
<td>36649</td>
<td>65075</td>
<td>65534</td>
<td>65536</td>
</tr>
<tr>
<td>17</td>
<td>63976</td>
<td>129824</td>
<td>131064</td>
<td>131072</td>
</tr>
<tr>
<td>18</td>
<td>109866</td>
<td>258906</td>
<td>262120</td>
<td>262144</td>
</tr>
<tr>
<td>19</td>
<td>185012</td>
<td>516168</td>
<td>524212</td>
<td>524288</td>
</tr>
<tr>
<td>20</td>
<td>306285</td>
<td>1028448</td>
<td>1048360</td>
<td>1048576</td>
</tr>
<tr>
<td>21</td>
<td>497190</td>
<td>2048272</td>
<td>2096586</td>
<td>2097152</td>
</tr>
<tr>
<td>22</td>
<td>792920</td>
<td>4077316</td>
<td>4192896</td>
<td>4194304</td>
</tr>
<tr>
<td>23</td>
<td>1241936</td>
<td>8111400</td>
<td>8385216</td>
<td>8388608</td>
</tr>
<tr>
<td>24</td>
<td>1913566</td>
<td>16124458</td>
<td>16769254</td>
<td>16777216</td>
</tr>
<tr>
<td>25</td>
<td>2898574</td>
<td>32034016</td>
<td>33536094</td>
<td>33554432</td>
</tr>
<tr>
<td>26</td>
<td>4323980</td>
<td>63579386</td>
<td>67067294</td>
<td>67108864</td>
</tr>
<tr>
<td>27</td>
<td>6353060</td>
<td>126076522</td>
<td>134124596</td>
<td>134217728</td>
</tr>
<tr>
<td>28</td>
<td>9206137</td>
<td>249736704</td>
<td>268228914</td>
<td>268435456</td>
</tr>
<tr>
<td>29</td>
<td>13158574</td>
<td>494124382</td>
<td>536416730</td>
<td>536870912</td>
</tr>
<tr>
<td>30</td>
<td>18576644</td>
<td>976302888</td>
<td>1072750464</td>
<td>1073741820</td>
</tr>
</tbody>
</table>

Similar to [49], we relax the notion of \(k\)-equivalence and define the \((k, m)\)-equivalence relation: \(x \sim (k, m) y\) if and only if \((x_\beta)_{\beta \in J(k, m)} = (y_\beta)_{\beta \in J(k, m)}\). Suppose that the number of \((k, m)\)-equivalence classes in \(\mathcal{F}\) is at most \(U(m)\) for all choice of \((k-1)\)-decks. Then we can obtain \(U \leq \prod_{m=1}^{k-1} U(m)\).

We now proceed to estimate \(U(m)\). Now, suppose \(x, y \in \mathcal{F}\). Since \(x \sim (k-1) y\), we have that \(x \sim y\). In other words, \(x\) and \(y\) have the same weight. Hence, we let \(w\) denote the weight of any sequence in \(\mathcal{F}\).

Let \(x \in J(n, w)\), the set of all sequences of length \(n\) and weight \(w\). Suppose that \(\beta \in J(k, m)\) and let \(\alpha\) be a subsequence of length \(k-1\) of \(\beta\). Then \(\alpha\) necessarily
belongs to $J(k-1, m-1) \cup J(k-1, m)$. Recall that $\beta_\alpha$ is the number of occurrences of $\alpha$ in $\beta$. Below we obtain a combinatorial relationship between $x_\alpha$, $x_\beta$ and $\beta_\alpha$, which is a refinement of Proposition 4.7.

**Proposition 4.12.** Let $x \in J(n, w)$ and $1 \leq m \leq k - 1$.

If $\alpha \in J(k - 1, m - 1)$, then

$$\sum_{\beta \in J(k, m)} \beta_\alpha x_\beta = (w - m + 1) x_\alpha.$$  \hspace{1cm} (4.7)

If $\alpha \in J(k - 1, m)$, then

$$\sum_{\beta \in J(k, m)} \beta_\alpha x_\beta = (n - k + m - w + 1) x_\alpha.$$  \hspace{1cm} (4.8)

**Proof.** Let $x = x_1 x_2 \cdots x_n$. Suppose that $\alpha \in J(k - 1, m - 1)$. To demonstrate (4.7), we proceed in a similar manner as in the proof of Proposition 4.7 and consider the following two collections $\mathcal{A}$ and $\mathcal{B}$ of tuples. Set

$$\mathcal{A} \triangleq \{(A; s) : A \subset [n], x_A = \alpha, x_s = 1, s \notin A\}.$$

Since $\alpha$ has weight $m - 1$ and $x$ has weight $w$, we have $(w - m + 1)$ choices for $s$ for each occurrence of $\alpha$ in $x$. Therefore, the left hand side of (4.7) counts the number of tuples in $\mathcal{A}$.

Next, set

$$\mathcal{B} \triangleq \{(\beta; B; t) : B \subset [n], x_B = \beta, t \in B, x_{B \setminus \{t\}} = \alpha\}.$$

For each $\beta \in J(k, m)$ and each occurrence of $\beta$ in $x$, we have $\beta_\alpha$ choices for $t$. Therefore, the right hand side of (4.7) counts the number of tuples in $\mathcal{B}$.

To establish (4.7), it remains to exhibit a bijection between $\mathcal{A}$ and $\mathcal{B}$. Consider the maps $\phi$ and $\psi$ defined in the proof of Proposition 4.7. When we restrict the domains of $\phi$ and $\psi$ to $\mathcal{A}$ and $\mathcal{B}$, respectively, the maps are well-defined bijections from $\mathcal{A}$ to $\mathcal{B}$ and vice versa. Hence, we obtain (4.7). When $\alpha \in J(k - 1, m)$, (4.8) can be similarly established by requiring $x_s = 0$ in the definition of $\mathcal{A}$. \hfill $\Box$

We define the following notations for convenience.
Definition 4.13. We let $H^{(k,m)}$ to be the $\binom{k}{m} \times \binom{k}{m}$ matrix whose rows and columns are indexed by $J(k-1,m-1) \cup J(k-1,m)$ and $J(k,m)$, respectively, with increasing lexicographic order. The entries of $H^{(k,m)}$ are given by $H^{(k,m)}_{\alpha\beta} \equiv \beta_{\alpha}$.

Further define a column vector $z$ of length $\binom{k}{m}$ such that the first $\binom{k-1}{m-1}$ entries are given by $((w - m + 1)x_\alpha)_{\alpha \in J(k-1,m-1)}$ and the next $\binom{k-1}{m}$ entries are given by $((n - k + m - w + 1)x_\alpha)_{\alpha \in J(k-1,m)}$. Then (4.7) and (4.8) imply that
\begin{equation}
H^{(k,m)}(x_\beta)_{\beta \in J(k,m)} = z. \tag{4.9}
\end{equation}

Example 4.3. Let $k = 3$ and $m = 2$. Then
\begin{equation*}
H^{(3,2)} = \begin{pmatrix}
2 & 1 & 0 \\
0 & 1 & 2 \\
1 & 1 & 1
\end{pmatrix}.
\end{equation*}

Consider $x = 110011$ and so, $n = 6$, $w = 4$. Also,
\begin{equation*}
(x_\beta)^T_{\beta \in J(3,2)} = (2, 8, 2),
\end{equation*}
\begin{equation*}
z^T = (3 \cdot 4, 3 \cdot 4, 2 \cdot 6) = (12, 12, 12).
\end{equation*}
We verify that (4.9) holds.

The following lemma then characterizes the $(k,m)$-equivalence of two sequences when they share the same $(k-1)$-deck.

Lemma 4.14. Let $H^{(k,m)}$ be as defined above. If $x \sim_{k-1} y$, then $(x_\beta - y_\beta)_{\beta \in J(k,m)}$ belongs to the nullspace of $H^{(k,m)}$.

Proof. Since $x \sim_{k-1} y$, we have that $H^{(k,m)}(x_\beta)_{\beta \in J(k,m)} = H^{(k,m)}(y_\beta)_{\beta \in J(k,m)}$. Hence, we have that $H^{(k,m)}(x_\beta - y_\beta)_{\beta \in J(k,m)} = 0$ and the lemma follows. \qed

Therefore, it remains to provide an upper bound on the nullity of $H^{(k,m)}$.

Proposition 4.15. The nullity of $H^{(k,m)}$ is at most $\binom{k-2}{m-1}$.

Proof. We write $H = H^{(k,m)}$ for short. Recall that the columns of $H$ are indexed by $J(k,m)$ and we arrange the columns in an increasing lexicographic order as in
4.3. Upper Bounds on $\text{Deck}_k(n)$

To this end, we demonstrate that the nullity of $H$ is at most $\binom{k-2}{m-1}$, or equivalently, that the rank of $H$ is at least $\binom{k}{m} - \binom{k-2}{m-1}$.

Recall that the leading coefficient of a row in a matrix is the first non-zero entry of that row. Hence, a column with a leading coefficient is a column that has a non-zero entry which is a leading coefficient of a row. Hence, to provide the desired lower bound for the rank of $H$, it suffices to exhibit $\binom{k}{m} - \binom{k-2}{m-1}$ columns with leading coefficients.

We have the following cases.

- Let $\beta = \beta_1 \beta_2 \cdots \beta_k \in J(k,m)$ with $\beta_1 = 0$. Then consider the row $\alpha \triangleq \beta_2 \cdots \beta_k \in J(k-1,m)$ and clearly, $H_{\alpha,\beta} \geq 1$. Suppose $\beta' \in J(k,m)$ and $\beta'_\alpha \geq 1$. Then $\beta$ is necessarily lexicographically smaller than or equal to $\beta'$. In other words, $H_{\alpha,\beta'} = 0$ for all sequences $\beta'$ that are lexicographically smaller than $\beta$. Therefore, $H_{\alpha,\beta}$ is the leading coefficient of row $\alpha$.

- Let $\beta = \beta_1 \beta_2 \cdots \beta_k \in J(k,m)$ with $\beta_k = 1$. Then consider the row $\alpha \triangleq \beta_1 \beta_2 \cdots \beta_{k-1} \in J(k-1,m-1)$ and as before, $H_{\alpha,\beta} \geq 1$. Proceeding as before, we observe that $H_{\alpha,\beta''} = 0$ for all sequences $\beta'' \in J(k,m)$ that are lexicographically smaller than $\beta$. Therefore, $H_{\alpha,\beta}$ is the leading coefficient of row $\alpha$.

Hence, the columns with possibly no leading coefficients start with a one and end with a zero. Therefore, there are $\binom{k-2}{m-1}$ such columns and the proposition follows.

Finally, we state the main theorem for this section and provide an upper bound on $\text{Deck}_k(n)$. Recall that $x \sim_{(k,m)} y$ if and only if $(x_\beta)_{\beta \in J(k,m)} = (y_\beta)_{\beta \in J(k,m)}$.

**Theorem 4.16.** The number of $(k,m)$-equivalence classes for sequences of length $n$ with the same $(k-1)$-deck is $O\left(n^{k\binom{k-2}{m-1}}\right)$.

Therefore, the number of distinct $k$-decks with the same $(k-1)$-deck is $O\left(n^{k(k-2)}\right)$ and hence, $\text{Deck}_k(n) = O\left(n^{(k-1)2^{k-1}+1}\right)$.

**Proof.** Fix $x$ to be of length $n$. Suppose that $y \sim_{k-1} x$. Then Lemma 4.14 states that $(y_\beta - x_\beta)_{\beta \in J(k,m)}$ belongs to the nullspace of $H^{(k,m)}$. Since the nullity of $H^{(k,m)}$ is
at most \((\binom{k-2}{m-1})\) and every entry of \((y_\beta)_{\beta \in J(k,m)}\) is at most \(\binom{n}{k} = O(n^k)\), the number of choices for \((y_\beta)_{\beta \in J(k,m)}\) is \(O\left(n^{k\binom{k-2}{m-1}}\right)\).

Therefore, the number of distinct \(k\)-decks with the same \((k-1)\)-deck is

\[
O\left(\prod_{m=1}^{k-1} n^{k\binom{k-2}{m-1}}\right) = O\left(n^{k\sum_{m=1}^{k-1} \binom{k-2}{m-1}}\right) = O\left(n^{k2^{k-2}}\right).
\]

Finally, it follows from simple induction that

\[
\text{Deck}_k(n) = \text{Deck}_{k-1}(n) \cdot O\left(n^{k2^{k-2}}\right) = O\left(n^{(k-1)2^{k-1}+1}\right).
\]

\[\Box\]

**Corollary 4.17.** \(\text{Deck}_3(n) = O(n^9)\).

In the next section, we will provide a general asymptotic lower bound on \(\text{Deck}_k(n)\) and show that \(\text{Deck}_k(n) = \Omega(n^k)\). We also look at the specific case for \(k = 3\) and improve the lower bound to \(\text{Deck}_3(n) = \Omega(n^6)\). In comparison, we note that (4.6) states that \(\text{Deck}_3(n) = O(n^9)\).

### 4.4 Lower Bounds

Recall that \(\text{Deck}_k(n)\) is the number of distinct equivalence classes among the binary sequences of length \(n\) formed by the equivalence relation \(\sim_k\). Here, we also define the following

**Definition 4.18.** Let \(\text{Deck}_{k,w}(n)\) to be the number of distinct equivalence classes among the binary sequences of length \(n\) and weight \(w\) formed by the equivalence relation \(\sim_k\).

As a consequence, we have \(\text{Deck}_k(n) = \sum_{w=0}^{n} \text{Deck}_{k,w}(n)\), since binary sequences with different weights belong to distinct equivalence classes.

In this section, we first derive lower bounds on the quantity \(\text{Deck}_k(n)\) for general \(k\). Then in the special case where \(k = 3\), we improve the lower bound of \(\text{Deck}_3(n)\) from \(\Omega(n^3)\) to \(\Omega(n^6)\). However, there remains a gap to the upper bound \(O(n^9)\) obtained
from Corollary 4.17. As such, in the rest of the section, we study lower bounds on the quantity of Deck\textsubscript{k,w}(n) with the hope that it sheds light on Deck\textsubscript{k}(n). In particular, we show that Deck\textsubscript{4,3}(n) = Θ(n^3).

Following [49], we consider the notion of zero-vectors. Let \( x \in J(n, w) \). The zero-vector of \( x \), denoted by \( u_x = (u_0, u_1, \ldots, u_w) \), is the vector of length \( w + 1 \), where \( u_0 \) is the number of zeroes in front of the first one, \( u_w \) is the number of zeroes after the last one, and \( u_j \) is the number of zeroes between the \( j \)th one and the \((j+1)\)th one for any \( 1 \leq j \leq w - 1 \). In other words, if \( x = 0^{u_0}1^{u_1} \cdots 10^{u_w} \), then \( u_x = (u_0, u_1, \ldots, u_w) \).

Recall that \( x_α \) is the number of occurrences of \( α \) as a subsequence of \( x \). To this end, we have the following lemma from [49].

**Lemma 4.19** (Manvel et al.[49, Lemma 14]). For \( k \geq 1 \), define the following \( k \times (w + 1) \)-integer-valued matrix:

\[
M^{(k,w)} = \begin{pmatrix}
\binom{w}{k-1} & \binom{w-1}{k-1} & \binom{w-2}{k-1} & \cdots & 1 & 0 & 0 & \cdots & 0 \\
0 & \binom{w-1}{k-2} & 2\binom{w-2}{k-2} & \cdots & (w-k+1)\binom{k-1}{k-2} & (w-k+2) & 0 & \cdots & 0 \\
0 & 0 & \binom{w-2}{k-3} & \cdots & (w-k+1)\binom{k-1}{k-3} & (w-k+2) & \binom{k-2}{k-3} & \binom{k-2}{k-2} & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & (w-k+1) & (w-k+3) & \binom{k-2}{k-1} & \binom{k-3}{k-1} & \cdots & \binom{w}{k-1}
\end{pmatrix}
\]

If \( x \) is a binary sequence of weight \( w \), then \( M^{(k,w)}u_x = (x_α)_{α \in J(k,k-1)} \).

**Remark 4.2.** The leftmost non-zero entry of the bottom row starts from \( \binom{k-1}{k-1} \) and ends with \( \binom{w}{k-1} \). However, in order to align with the non-zero entries from the previous rows, we fill in the entry of the \((w - k + 2)\)-th column to be \( \binom{w-k+1}{k-1} \).

**Example 4.4.** Let \( x = 110011 \) and \( k = 3 \). So, \( u_x = (0, 0, 2, 0, 0) \) and the weight \( w = 4 \). Note that

\[
M^{(3,4)} = \begin{pmatrix}
6 & 3 & 1 & 0 & 0 \\
0 & 3 & 4 & 3 & 0 \\
0 & 0 & 1 & 3 & 6
\end{pmatrix}
\]

We verify that \( (x_α)_{α \in J(3,2)}^T \) is indeed given by \( M^{(3,4)}u_x = (2, 8, 2)^T \).

Let \( u = (u_0, u_1, \ldots, u_w) \) be a vector of length \( w + 1 \), where \( u_i \) is a non-negative integer for all \( 0 \leq i \leq w \). We define the following...
Definition 4.20. Let $W(u)$ be the binary sequence of weight $w$ whose zero-vector is $u$, namely $W(u) = 0^w1^01^1 \cdots 0^{w-1}1^{w}$. 

In the following subsection, we are going to present lower bounds for $\text{Deck}_k(n)$ for general $k$.

4.4.1 Lower Bounds on $\text{Deck}_k(n)$

Consider 

$$C^{(k,w)}(n) \triangleq \left\{ (u_0, u_1, \ldots, u_w) : \sum_{i=0}^{k-1} u_i = n - w, u_j = 0 \text{ for all } j \geq k \right\}.$$ 

Then we have the following lemma.

Lemma 4.21. For any distinct zero-vector $v_1$ and $v_2$ in $C^{(k,w)}(n)$, we have that $W(v_1) \sim_{(k,k-1)} W(v_2)$.

Proof. Recall from Lemma 4.19 that $M^{(k,w)}u_x = (x_a)_{a \in J(k,k-1)}$. Observe that the first $k$ columns of $M^{(k,w)}$ are linearly independent. Thus, any distinct linear combination of the first $k$ columns of $M^{(k,w)}$ yields different vector. Therefore if $v_1$ and $v_2$ are distinct zero-vectors in $C^{(k,w)}(n)$, $M^{(k,w)}v_1 \neq M^{(k,w)}v_2$, and hence $W(v_1) \sim_{(k,k-1)} W(v_2)$. 

Furthermore, for a fixed length $n$, binary sequences with different weights have different 1-decks, and therefore have different $k$-decks ($k \geq 1$). Thus the binary sequences in 

$$\left\{ W(u) : u \in \bigcup_{w=0}^{n} C^{(k,w)}(n) \right\}$$

correspond to binary sequences of length $n$ with distinct $k$-decks. Since the total number of sequences is $\sum_{w=0}^{n} |C^{(k,w)}(n)| = \sum_{w=0}^{n} \binom{n-w+k-1}{k-1} = \binom{n-k}{k}$, we have the following theorem.

Theorem 4.22. $\text{Deck}_k(n) \geq \binom{n-k}{k}$.

Corollary 4.23. In particular, we have $\text{Deck}_3(n) = \Omega(n^3)$.

In the following subsection, we focus on the case $k = 3$ and improve the lower bound to $\text{Deck}_3(n) = \Omega(n^6)$. 

Technion - Computer Science Department - Ph.D. Thesis PHD-2021-16 - 2021
4.4.2 Lower Bounds on Deck$_3(n)$

As with Section 4.3, we consider the sequences with the same $(k - 1)$-deck, or 2-deck, and determine the number of 3-equivalence classes amongst these sequences. Let $x$ be a binary sequence of length $n$. Set $x_{01} = t$ and $x_1 = w$. Our objective is to estimate the possible values of $x_{011}$.

**Lemma 4.24.** Set $\Gamma$ to be the following set of vectors of length $w + 1$:

$$\Gamma \triangleq \{(1, -2, 1, 0, 0, \ldots, 0), (0, 1, -2, 1, 0, \ldots, 0), (0, 0, 1, -2, 1, \ldots, 0), \ldots, (0, \ldots, 1, -2, 1)\}.$$ 

If $x$ and $x'$ are binary sequences of weight $w$ such that $u_{x'} = u_x + c$ for some $c \in \Gamma$, then $x_1 = x'_1$, $x_{01} = x'_{01}$ and $x'_{011} = x_{011} + 1$.

**Proof.** It follows from the definition of zero-vectors that $x_1 = x'_1$. For the other two equalities, we have that $M^{(k,w)}u_{x'} = M^{(k,w)}u_x + M^{(k,w)}c$. Applying Lemma 4.19 for $k \in \{2, 3\}$ and considering the first coordinate, we have

$$x'_{01} = x_{01} + (M^{(2,w)}c)_{1,1} \quad \text{and} \quad x'_{011} = x_{011} + (M^{(3,w)}c)_{1,1},$$

where $(M)_{i,j}$ is the the entry on the $i$-th row and $j$-th column of matrix $M$. Then $(M^{(2,w)}c)_{1,1} = 0$ because $a - 2(a + 1) + (a + 2) = 0$ for all $a$. On the other hand, $(M^{(3,w)}c)_{1,1} = 1$ because $\binom{a}{2} - 2\binom{a+1}{2} + \binom{a+2}{2} = 1$ for all $a$. The lemma is then immediate. 

**Example 4.5.** As before, let $x = 110011$ and $k = 3$. Further set $x' = 101101$ and so, $u_{x'} = (0, 1, 0, 1, 0)$ and $u_x = u_x + (0, 1, -2, 1, 0)$. We verify that $x_1 = x'_1 = 4$, $x_{01} = x'_{01} = 4$, and $x'_{011} = x_{011} + 1 = 3$.

Next, we have the following proposition.

**Proposition 4.25.** Fix $n$, $0 \leq w \leq n$ and $0 \leq t \leq w(n - w)$. Then there exist $x$ and $y$ of length $n$ such that the following hold:

1. $x_1 = y_1 = w$ and $x_{01} = y_{01} = t$;
2. $x_{011} \leq (n - w)\binom{q+1}{2}$ where $q = \lceil t/(n - w) \rceil$;
3. \( y_{011} \geq \frac{w-1}{2} (t - \binom{w}{2}) \);

4. for any \( x_{011} \leq s \leq y_{011} \), there exists \( z \) such that \( z_1 = w \), \( z_{01} = t \) and \( z_{011} = s \).

**Proof.** Write that \( t = q(n - w) + r \). Set \( x \) to be the sequence whose zero vector is \((A_0, A_1, \ldots, A_w)\), where \( A_{w-q-1} = r \), \( A_{w-q} = n - w - r \) and all others are 0. Recall that \( x_{01} = \sum_{i=0}^{w-1} u_i (w - i) \) and indeed, \( x_{01} = r(q + 1) + (n - w - r)q = t \).

Furthermore, \( x_{011} = r\left(\frac{q+1}{2}\right) + (n - w - r)\left(\frac{q}{2}\right) \leq (n - w)\left(\frac{q+1}{2}\right) \).

To construct \( y \), we iteratively add some \( c \) from \( \Gamma \) to \( u_x \). Specifically, we start with \( x(0) = x_0 \) and \( u(0) = u_x \) and suppose that we have \( u(0), u(1), \ldots, u(i) \). If \( u(i) \) has a component with value at least two at index \( j \) with \( 1 \leq j \leq w - 1 \), we choose \( c(i) = (0, 0, \ldots, 0, 1, \ldots, 0) \) with the minus two at index \( j \). Then we set \( u(i+1) = u(i) + c(i) \) and \( x(i+1) \) to be the corresponding sequence. It follows from Lemma 4.24 that the two-deck of \( x(i+1) \) is the same as \( x(i) \) and the number of 011 in \( x(i) \) is given by \( x_{011} + i \).

Hence, we terminate the process when the components of \( u(i) \) are at most one on the indices from 1 to \( w - 1 \). Let \( (B_0, B_1, \ldots, B_w) \) be the final zero-vector and \( y \) be the corresponding sequence. It then remains to show Proposition 4.25(iii).

Again, we have \( 0 \leq B_i \leq 1 \) for all \( 1 \leq i \leq w - 1 \). Furthermore, we have \( y_{01} = x_{01} = t \), which means

\[
t = wB_0 + \sum_{i=1}^{w-1} (w - i)B_i \leq wB_0 + \sum_{i=1}^{w-1} (w - i) \leq wB_0 + \binom{w}{2}.
\]

Therefore, \( B_0 \geq \frac{t - \binom{w}{2}}{w} \) and so, \( y_{011} \geq B_0 \binom{w}{2} \geq \frac{w-1}{2} (t - \binom{w}{2}) \).

Following this proposition, for purposes of brevity, we write

\[
s' \triangleq (n - w)\left(\frac{q+1}{2}\right),
\]
\[
s'' \triangleq \frac{w-1}{2} \left(t - \binom{w}{2}\right).
\]

Therefore, a lower bound for \( \text{Deck}_3(n) \) is

\[
\sum_{w=0}^{n} \sum_{t=0}^{w(n-w)} \max \{0, s'' - s'\}. \tag{4.10}
\]
4.4. Lower Bounds

We estimate the terms of (4.10). Note that if \( t = 0 \) then \( s'' < 0 \), then we can consider

\[
\sum_{w=0}^{n} \sum_{t=1}^{w(n-w)} s'' = \sum_{w=0}^{n} \sum_{t=1}^{w(n-w)} \frac{w-1}{2} \left( t - \left( \frac{w}{2} \right) \right)
\]

\[
= \sum_{w=0}^{n} \frac{w-1}{2} \sum_{t=1}^{w(n-w)} t - \left( \frac{w}{2} \right)
\]

\[
\geq \sum_{w=0}^{n} \frac{w-1}{2} \left( \frac{1}{2} w^2(n-w)^2 - \left( \frac{w}{2} \right) w(n-w) \right)
\]

\[
\geq \sum_{w=0}^{n} \frac{w-1}{4} \left( w^2(n-w)^2 - w^3(n-w) \right)
\]

\[
= \sum_{w=0}^{n} \frac{w-1}{4} \left( w^2(n-w)(n-2w) \right)
\]

\[
= \left( \sum_{w=0}^{n} \frac{1}{4} w^3(n-w)(n-2w) \right) - O\left( n^5 \right), \quad (4.11)
\]

since \( \sum_{w=0}^{n} w^2(n-w)(n-2w) \) is a polynomial of \( n \) of degree at most five. Here, the big-O notation is evaluated with respect to \( n \). On the other hand,

\[
\sum_{w=0}^{n} \sum_{t=0}^{w(n-w)} s' = \sum_{w=0}^{n} \sum_{t=0}^{w(n-w)} (n-w) \left( q + 1 \right) \left( \frac{w}{2} \right)
\]

\[
= \sum_{w=0}^{n} \sum_{t=0}^{w(n-w)} \frac{1}{2}(n-w)q^2 + \frac{1}{2}(n-w)q
\]

\[
= \sum_{w=0}^{n} \frac{1}{2}(n-w) \left( \sum_{t=0}^{w(n-w)} [t/(n-w)]^2 + [t/(n-w)] \right)
\]

\[
= \sum_{w=0}^{n} \frac{1}{2}(n-w) \left( \frac{1}{6}(n-w)(w-1)w(2w-1) + w^2 + \frac{1}{2}(n-w)(w-1)w + w \right)
\]

\[
= \sum_{w=0}^{n} \frac{1}{6}(n-w)^2w^3 + O\left( (n-w)^2w^2 \right) = \left( \sum_{w=0}^{n} \frac{1}{6}(n-w)^2w^3 \right) + O\left( n^5 \right). \quad (4.12)
\]

Again, the final big-O estimate is valid since \( \sum_{w=0}^{n} w^2(n-w)^2 \) is a polynomial of \( n \) of degree at most five. Combining (4.11) and (4.12) into (4.10), we have that the
number of 3-decks is at least
\[
\left( \sum_{w=0}^{n} \frac{1}{4} w^3(n-w)(n-2w) - \frac{1}{6} (n-w)^2 w^3 \right) + O(n^5)
\]
\[
= \left( \sum_{w=0}^{n} w^3(n-w) \left( \frac{1}{4} n - \frac{1}{2} \frac{1}{w} n + \frac{1}{6} w \right) \right) + O(n^5)
\]
\[
= \left( \sum_{w=0}^{n} \frac{1}{12} w^3(n-w)(n-4w) \right) + O(n^5) = \Omega(n^6). \tag*{(4.13)}
\]

We summarize our discussion in the following theorem.

**Theorem 4.26.** $\text{Deck}_3(n) = \Omega(n^6)$.

**Remark 4.3.** The statement in Theorem 4.26 can be made more precise. We have demonstrated that the number of $(3, 2)$-equivalence classes amongst all sequences of length $n$ is $\Omega(n^6)$. It then follows from Theorem 4.16 that this estimate is tight.

**Remark 4.4.** Implicit in the proof of Proposition 4.25 is an efficient method that encodes messages into sequences with distinct 3-decks. Specifically, let the message set be
\[
\mathcal{M} \triangleq \{(w, t, s) : 0 \leq w \leq n, \ 0 \leq t \leq w(n-w), \ s' \leq s \leq s''\}.
\]

Given any triple $(w, t, s) \in \mathcal{M}$, we can construct $x$ in linear time such that $x_1 = w$, $x_{01} = t$ and $x_{011} = s'$.

Following the procedure described in the proof of Proposition 4.25, we choose a sequence of $c^{(0)}, c^{(1)}, \ldots \in \Gamma$ to add to $u_x$. Since $s' \leq s \leq s''$, there is a sequence such that the resulting zero-vector corresponds to $y$ and $y_{011} = s$. Therefore, $y$ is the codeword encoding the message $(w, t, s)$ and $y$ can be computed in $O(n^3)$ time.

### 4.4.3 Lower Bounds on $\text{Deck}_{3,3}(n)$

We first review known results on $\text{Deck}_{k,w}(n)$. The following corollary is immediate from Proposition 4.2.

**Corollary 4.27.** For $n \geq w$, $\text{Deck}_{2,w}(n) = w(n-w) + 1$.

Furthermore, Manvel et al. have also proven the following results.
4.4. Lower Bounds

**Theorem 4.28** (Manvel et al.[49, Lemma 7][49, Lemma 10]).

1. For \( w \leq k - 1 \), \( \text{Deck}_{k,w}(n) = \binom{n}{w} \).
2. For \( n < 2^{k-1} + k \), \( \text{Deck}_{k,k}(n) = \binom{n}{k} \).

In other words, all binary sequences of length \( n \) and weight \( w \leq k-1 \) have distinct \( k \)-decks. However, for \( n \geq 2^{k-1} + k \), there are distinct binary sequences with length \( n \) and weight \( w \) that have the same \( k \)-deck, and hence the exact values for \( \text{Deck}_{k,k}(n) \) is still unknown for \( n \geq 2^{k-1} + k \). For example, when \( k = 3 \), and \( n = 2^2 + 3 = 7 \), observe that 1000110 and 0110001 have the same 3-deck. Therefore, the smallest open problem is to determine \( \text{Deck}_{3,3}(n) \) for \( n \geq 2^{k-1} + k \).

Suppose that \( x \) and \( y \) are two binary sequences with length \( n \geq 3 \) and weight \( w = 3 \) such that \( x \sim (3,2) y \), then we know from Lemma 4.19 that \( u_x - u_y \) is in the null-space of \( M^{(3,3)} \). Observe that the null space of \( M^{(3,3)} = \begin{pmatrix} 3 & 1 & 0 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 0 & 1 & 3 \end{pmatrix} \) is spanned by \((1,-3,3,-1)\). Hence \( u_x - u_y \) has to be an integer multiple of \((1,-3,3,-1)\). Let \( u_x = (u_0, u_1, u_2, u_3) \) and \( u_y = (u_0 + t, u_1 - 3t, u_2 + 3t, u_3 - t) \). Then we have

\[
y_{001} - x_{001} = \left( \frac{u_0 + t}{2} \right) + \left( \frac{u_0 + u_1 - 2t}{2} \right) + \left( \frac{u_0 + u_1 + u_2}{2} \right) - \\
\left( \frac{u_0}{2} \right) - \left( \frac{u_0 + u_1}{2} \right) - \left( \frac{u_0 + u_1 + u_2}{2} \right) = u_0 - 2(u_0 + u_1)t + (u_0 + u_1 + u_2)t + 3t^2 = u_0(1-t) + u_1(-t) + u_2(t) + 3t^2.
\]

Thus, we have the following corollary.

**Lemma 4.29.** Let \( x \) and \( y \) be binary sequences of length \( n \) and weight \( w = 3 \), such that \( u_x = (u_0, u_1, u_2, u_3) \) and \( u_y = (u_0 + t, u_1 - 3t, u_2 + 3t, u_3 - t) \), where \( t \geq 1 \). If \( u_3 \geq t \) and \( u_1 > u_2 + 3t \), then \( x \sim (3,2) y \) and \( y_{001} < x_{001} \).

**Proof.** Note that \( u_x - u_y = (t, -3t, 3t, -t) \), which is in the nullspace of \( M^{(3,3)} \), hence by Lemma 4.19, we have \( x \sim (3,2) y \). Furthermore, \( y_{001} - x_{001} = u_0(1-t) + u_1(-t) + u_2(t) + 3t^2 \leq u_1(-t) + u_2(t) + 3t^2 = t(u_2 - u_1 + 3t) < 0. \)
Similar to the construction of $C^{(3,3)}(n)$, we define
\[
P^{(3,3)}(n) = \{(u_0, u_1, u_2, u_3) : u_0 + u_1 + u_3 = n - 3, u_2 = 0\}.
\]

Since the first, second, and fourth columns of $M^{(3,3)}$ are linearly independent, the result of Lemma 4.21 still follows for $P^{(3,3)}(n)$, namely that the binary sequences in $\{W(u) : u \in P^{(3,3)}(n)\}$ correspond to binary sequences of weight three with distinct 3-decks, which is summarized in the following corollary.

**Corollary 4.30.** For any distinct pair of zero-vectors $v_1$ and $v_2$ in $P^{(3,3)}(n)$, we have that $W(v_1) \sim_{(3,2)} W(v_2)$.

For every $u = (u_0, u_1, u_2, u_3) \in P^{(3,3)}(n)$, we define $u(t) \triangleq (u_0 + t, u_1 - 3t, u_2 + 3t, u_3 - t)$. Let $T(u) \triangleq \{u(t) : t \geq 0, u_1 > u_2 + 6t, u_3 > 2t\}$.

**Lemma 4.31.** For every $u \in P^{(3,3)}(n)$, for any distinct $v_1, v_2 \in T(u)$, we have $W(v_1) \sim_{(3,2)} W(v_2)$ but $W(v_1)_{001} > W(v_2)_{001}$.

**Proof.** Suppose $v_1 = u(\ell)$ and $v_2 = u(m)$, for some non-negative integers $\ell, m$. Without loss of generality, we assume that $\ell < m$. We want to use Lemma 4.29 to show that $W(v_1) \sim_{(3,2)} W(v_2)$ and $W(v_1)_{001} > W(v_2)_{001}$. By substituting $u_x = (u_0, u_1, u_2, u_3)$ with $u(\ell) = (u_0 + \ell, u_1 - 3\ell, u_2 + 3\ell, u_3 - \ell)$ and $t$ with $m - \ell$ in Lemma 4.29, we need to check if $u_3 - \ell \geq m - \ell$ and $u_1 - 3\ell > u_2 + 3\ell + (m - \ell) = u_2 + 3m$. The first inequality is true due to the fact that $u(m)$ is in $T(u)$, and hence $u_3 \geq m$. Since $u(m), u(\ell) \in T(u)$, we have $6m < u_1 - u_2$, and also $6\ell < u_1 - u_2$, and hence $u_1 - u_2 > 3m + 3\ell$ which implies the second inequality. \hfill \Box

**Theorem 4.32.** For any fixed length $n$,
\[
\text{Deck}_{3,3}(n) \geq \sum_{u \in P^{(3,3)}(n)} |T(u)| = \sum_{u_0 + u_1 + u_3 = n - 3} \min \left\{ \left\lceil \frac{u_1}{6} \right\rceil, u_3 + 1 \right\}.
\]

**Proof.** For a fixed length $n$, for a fixed $u \in P^{(3,3)}(n)$, by Lemma 4.31, for any two distinct zero-vectors $v_1$ and $v_2$ in $T(u)$ we have that $W(v_1) \sim_{(3,1)} W(v_2)$. Moreover, together with Corollary 4.30, we also have that for any distinct $u_1$ and $u_2$ in $P^{(3,3)}(n)$, for any $v_1 \in T(u_1), v_2 \in T(u_2)$, we have that $W(v_1) \sim_{(3,2)} W(u_1) \sim_{(3,2)} W(u_2) \sim_{(3,2)} W(v_2)$. Hence every zero-vector in $\bigcup_{u \in P^{(3,3)}(n)} T(u)$ corresponds to a binary sequence with weight three with distinct 3-decks.
Furthermore, based on the definition of $T(u)$, we have

$$u(t) \in T(u) \iff 0 \leq t < \frac{u_1 - u_2}{6} \text{ and } t \leq u_3$$

$$\iff 0 \leq t \leq \min \left\{ \left\lceil \frac{u_1 - u_2}{6} \right\rceil - 1, u_3 \right\}.$$ 

Since $u_2 = 0$ in the definition of $P^{(3,3)}(n)$, we have $|T(u)| = \min \left\{ \left\lceil \frac{u_1}{6} \right\rceil, u_3 + 1 \right\}$, as desired.

**Corollary 4.33.** $\text{Deck}_{3,3}(n) = \Theta(n^3)$.

**Proof.** Set $u_0 = a, u_1 = \left\lfloor \frac{n-3}{3.3} \right\rfloor + b, u_3 = \left\lfloor \frac{n-3}{21} \right\rfloor + c$, where $a, b, c$ are non-negative integers. From Theorem 4.32, we have

$$\text{Deck}_{3,3}(n) \geq \sum_{u_0+u_1+u_3=n-3} \min \left\{ \left\lceil \frac{u_1}{6} \right\rceil, u_3 + 1 \right\}$$

$$\geq \sum_{u_0+u_1+u_3=n-3} \min \left\{ \frac{u_1}{6}, u_3 + 1 \right\}$$

$$\geq \sum_{a+b+c=\left\lceil \frac{2}{3}(n-3) \right\rceil} \frac{n-3}{21} - 6$$

$$\geq \frac{n-129}{21} \left( \left\lceil \frac{2}{3}(n-3) \right\rceil + 2 \right)$$

$$= \Omega(n^3).$$

Moreover, there are only $\binom{n}{3}$ binary sequences of length $n$ and weight three, which implies that $\text{Deck}_{3,3}(n) = O(n^3)$. Hence the corollary follows.

In fact, we make the following conjecture regarding $\text{Deck}_{3,3}(n)$.

**Conjecture 2.** Consider the generating function $D(z) = \sum_{n \geq 0} \text{Deck}_{3,3}(n)z^n$. We have that

$$D(z) = \frac{z^3}{(1-z)^4} - \frac{z^7}{(1-z)^2(1-z^2)(1-z^4)}.$$ 

Using a modified version of Algorithm 4, we have verified that the first 20 coefficients of $D(z)$ coincide with the exact values of $\text{Deck}_{3,3}(n)$.
4.5 Conclusion

We provide an efficient trellis-based method to compute the number of distinct $k$-decks and determine the exact value of $\text{Deck}_k(n)$ for $k \in \{3, 4, 5, 6\}$ and $k \leq n \leq 30$. As an interesting consequence, we show that $S(6) = 30$.

We also establish asymptotic upper bounds and lower bounds on $\text{Deck}_k(n)$ for general $k$, and further improve the lower bounds for $\text{Deck}_3(n)$ and $\text{Deck}_{3,3}(n)$. In summary, we have $\text{Deck}_k(n) = O(n^{(k-1)2^{k-1}+1})$, $\text{Deck}_k(n) = \Omega(n^k)$, $\text{Deck}_3(n) = O(n^3)$, $\text{Deck}_3(n) = \Omega(n^6)$ and $\text{Deck}_{3,3}(n) = \Theta(n^3)$. It remains open to determine tighter bounds on the asymptotic growth rate of $\text{Deck}_3(n)$. Finally, let us briefly discuss certain possible extensions of the current work.

- For general alphabet of size $q \geq 2$, we can similarly define $\text{Deck}_k^{(q)}(n)$ to be the number of distinct $k$-decks for $q$-ary sequences of length $n$. This quantity was studied by Lejeune, Rigo, and Rosenfeld in [46]. In particular, for all $q$, Lejeune, Rigo, and Rosenfeld showed that $\text{Deck}_2^{(q)}(n) = \Theta\left(n^{q^2-1}\right)$ and $\text{Deck}_k^{(q)}(n) = O\left(n^{k^2q^k}\right)$ for $k \geq 3$. When $q = 2$, our Theorem 4.16 improves the upper bound. It is not clear whether the analysis in Section 4.3 can be extended to provide tighter upper bounds for large alphabets. Such an analysis would entail studying the $k$-decks of sequences with the same composition (or, equivalently, 1-deck) and setting up a corresponding system of linear equations.
Chapter 5

Correcting Deletions with Multiple Distinct Reads

This chapter is dedicated for the analysis and properties of \( t \)-deletion balls, of special interests are the single-deletion and two-deletion balls. More specifically, we characterize the conditions when two single-deletion balls have intersection size two (i.e. when two different codewords result in two noisy reads through the single-deletion channel). In this same spirit, we determine when two single-deletion balls have intersection size one. Using this characterization, we show that the same reconstruction code for the single-deletion channel can be used to uniquely reconstruct codewords with approximately half the number of reads (as compared to the case with no coding) for the \( t \)-deletions channel with \( t \geq 2 \). For two-deletion channel, we also provide an explicit code construction that uniquely reconstructs a codeword from any five of its length-\((n - 2)\) subsequences, using only \( 2\log n + o(\log n) \) redundant bits. Similar to the case of single-deletion, the construction is based on the characterization of the intersection of two-deletion balls.

Part of this chapter is already presented in the IEICE International Symposium on Information Theory and Its Applications (ISITA) 2020 [81] and also in the IEEE International Symposium on Information Theory (ISIT) 2021 [86].
5.1 Read Coverage and Deletion Balls

Consider a data storage scenario described by an error-ball function. Formally, given an input space $X$ and an output space $Y$, an error-ball function $B$ maps a sequence $x \in X$ to a subset of noisy reads $B(x) \subseteq Y$. We then define the read coverage of a code $C$ as follows.

**Definition 5.1.** Given a code $C \subseteq X$, we define the read coverage of $C$, denoted by $\nu(C; B)$, to be the quantity

$$
\nu(C; B) \triangleq \max \left\{|B(x) \cap B(y)| : x, y \in C, x \neq y\right\}.
$$

This means that $\nu(C; B)$ is the maximum intersection between the error-balls of any two codewords in $C$. The quantity $\nu(C; B)$ was introduced by Levenshtein [59], where he showed that the number of reads\(^1\) required to reconstruct a codeword from $C$ is at least $\nu(C; B) + 1$. Therefore, to construct a code $C$ that can uniquely reconstruct the original codeword, it all boils down to finding the size of the maximum intersection among the error-balls. The problem to determine $\nu(C; B)$ is referred to as the *sequence reconstruction problem*.

In [60], the authors propose the study of *code design* when the number of noisy reads is at most $\nu(\{0, 1\}^n; B)$. Therefore, we define reconstruction code as follows

**Definition 5.2.** A code $C$ is an $(n, N; B)$-reconstruction code if $C \subseteq \{0, 1\}^n$ and $\nu(C; B) < N$.

This gives rise to a *new quantity of interest* that measures the *trade-off between codebook redundancy and read coverage*. Specifically, given $N$ and an error-ball $B$, we study the quantity

$$
\rho(n, N; B) \triangleq \min \left\{n - \log |C| : C \subseteq \{0, 1\}^n, \nu(C; B) < N\right\}.
$$

In this chapter, we focus on channels that introduce *deletions only*. For convenience, we will introduce the following notations.

\(^1\)In the original paper, Levenshtein used the term “channels”, instead of reads. Here, we used the term “reads” to reflect the data storage scenario.
Let $D_t(x)$ be the deletion ball of $x$ with exactly $t$ deletions. In other words, $D_t(x)$ is the set of all $(n-t)$-subsequences of $x$. Let $D_t(n)$ denote the maximum deletion ball size of words of length $n$, that is,

$$D_t(n) = \max\{|D_t(x)| : x \in \{0,1\}^n\}.$$ 

Levenshtein also provided a summation form of $D_t(n)$ as described in the theorem below.

**Theorem 5.3** (Levenshtein [66]).

$$D_t(n) = \sum_{i=0}^{t} \binom{n-t}{i} = \frac{n^t}{t!} + O(n^{t-1}), \text{ for } 0 \leq t \leq n. \quad (5.1)$$

**Remark 5.1.** For convenience, we assign $D_t(n) = 0$ when $t < 0$ or $t > n$.

For purposes of brevity, we let $\nu_t(n)$ denote $\nu(\{0,1\}^n; D_t)$, the read coverage of $\{0,1\}^n$. We also have the following landmark result of Levenshtein.

**Theorem 5.4** (Levenshtein [66]).

$$\nu_t(n) = 2D_{t-1}(n-2) = \frac{2n^{t-1}}{(t-1)!} + O(n^{t-2}). \quad (5.2)$$

This implies that $\rho(n, N; D_t) = 0$ for $N \geq \nu_t(n) + 1$.

In the next few sections, we are going to discuss the code constructions for $(n, N; D_t)$-reconstruction code for $t = 1$, $t = 2$, and $t \geq 3$, where $N \leq \nu_t(n)$.

## 5.2 Single-deletion Reconstruction Code

In this section, we are going to discuss the past results regarding $(n, N; D_1)$-reconstruction code for $N = 1, 2$ and 3. In 1966, Levenshtein constructed a single-deletion correcting code based on the following VT syndrome.

**Definition 5.5** (VT syndrome). Let $x$ be a binary sequence of length $n$. Then we define the VT syndrome of $x$ as $\text{VT}(x) \triangleq \sum_{i=1}^{n} ix_i$. 

5.2. Single-deletion Reconstruction Code

Following which, he proposed a construction the well-known asymptotically optimal single-deletion correcting code called VT-code as follows

**Construction A** (VT-code). Let \( \Sigma = \{0, 1\} \). Given a positive integer \( n \) and an integer \( 0 \leq a \leq n \). Then we define \( \text{VT}_a(n) = \{ x \in \Sigma^n : \text{VT}(x) = a \mod n + 1 \} \).

**Theorem 5.6** ([67]). For a given positive integer \( n \), the optimal redundancy of a \( t \)-deletion correcting code is between \( t \log_2 n + \Theta(1) \) and \( 2t \log_2 n + \Theta(1) \). Furthermore, \( \rho(n, 1; \mathcal{D}_1) \geq \log_2 n + \Theta(1) \). Furthermore, the code \( \text{VT}_0(n) \) from Construction A is asymptotically optimal with redundancy \( \log_2(n + 1) + \Theta(1) \).

This theorem implies that \( \rho(n, 1; \mathcal{D}_1) = \log_2(n) + \Theta(1) \). Moreover, by plugging in \( t = 1 \) to equation (5.2), we get \( \nu_1(n) = 2D_0(n - 2) = 2 \). This means that even if we use the whole codebook \( \{0, 1\}^n \), the maximum intersection between any single-deletion balls is of size 2. Therefore, we have the following corollary from Theorem 5.4.

**Corollary 5.7.** The minimum redundancy for a single-deletion channel with \( N = 3 \) distinct reads is 0, and hence \( \rho(n, N; \mathcal{D}_1) = 0 \) for \( N \geq 3 \).

From Theorem 5.6 and Corollary 5.7, we are now left to determine the value of \( \rho(n, N; \mathcal{D}_1) \) for \( N = 2 \).

Now, the first construction of a \( (n, 2; \mathcal{D}_1) \)-reconstruction code was proposed in [58] for the design of codes in racetrack memory. The codebook uses \( \log_2 \log_2 n + O(1) \) redundant bits and in [60], Kiah et al. modified the construction to obtain codebooks that uniquely reconstruct codewords for the single-edit channel and its variants. The construction can be seen as a generalization of the classical VT-code in Construction A and the shifted VT codes proposed by Schoeny et al. [68].

Before, we present the construction, we need to introduce several definitions as follows.

**Definition 5.8.** A sequence \( x = x_1x_2 \cdots x_n \) is 2-periodic if \( x_k = x_{k+2} \) for all \( k \).
We say that \( x \) is alternating if it is 2-periodic and \( x_1 \neq x_2 \). For convenience, length-one sequences are alternating.

**Definition 5.9.** Let \( x \) be a binary sequence. A 2-periodic run of \( x \) is a contiguous substring \( x_i \cdots x_j \) of \( x \) that is 2-periodic.
Chapter 5. Correcting Deletions with Multiple Distinct Reads

Construction B (Constrained Shifted VT Codes [58, 60]). For \( n \geq P > 0 \) and \( P \) even, let \( c \in \mathbb{Z}_{1+P/2} \) and \( d \in \mathbb{Z}_2 \). The constrained shifted VT code \( \mathcal{C}_{CSVT}(n, P; c, d) \) is defined to be the set of all words \( x = x_1x_2\cdots x_n \) such that the following holds.

1. \( VT(x) = c \pmod{1 + P/2} \).
2. \( \sum_{i=1}^{n} x_i = d \pmod{2} \).
3. The longest 2-periodic run in \( x \) is of length at most \( P \).

Remark 5.2. When \( P = 2n \) and we remove Condition 2, we recover the classical VT code that corrects a single deletion. This is because when \( P = 2n \), then any 2-periodic run is at most \( n < P \), hence, Condition 3 is always true. On the other hand, when we remove the Condition 3, we recover the shifted VT code that is used in the correction of a single burst of deletions [68].

It was recently demonstrated that the Constrained Shifted VT-code or CSVT in short, enables unique reconstruction whenever we have two distinct noisy reads as summarized in the following theorem.

Theorem 5.10 ([58, 60]). For all choices of \( c \) and \( d \), we have that \( \mathcal{C}_{CSVT}(n, P; c, d) \) is an \( (n, 2; \mathcal{D}_1) \)-reconstruction code. Furthermore, if we set \( P = \lceil \log_2 n \rceil + 2 \), the code \( \mathcal{C}_{CSVT}(n, P; c, d) \) has redundancy \( 1 + \log_2(\lceil \log_2 n \rceil + 4) = \log_2 \log_2 n + O(1) \) for some choice of \( c \) and \( d \). Thus, \( \rho(n, 2; \mathcal{D}_1) \leq \log_2 \log_2 n + O(1) \).

Therefore, we are now interested to know what is the lower bound of \( \rho(n, 2; \mathcal{D}_1) \). Surprisingly, it turns out that the codes in Theorem 5.10 are already asymptotically optimal as summarized in the following main theorem of this section.

Theorem 5.11. The value \( \rho(n, N; \mathcal{D}_1) \) satisfies

\[
\rho(n, N; \mathcal{D}_1) = \begin{cases} 
\log_2 n + \Theta(1), & \text{when } N = 1, \\
\log_2 \log_2 n + \Theta(1), & \text{when } N = 2, \\
0, & \text{when } N \geq 3.
\end{cases}
\]

To show the lower bound of \( \rho(n, 2; \mathcal{D}_1) \) and therefore proving Theorem 5.11, we first observe that \( \nu_1(n) = 2 \) and thus, our strategy is to find the characterization of pairs of sequences whose single-deletion balls have intersection size exactly two. To do so, we present the following definition of confusability.
5.2. Single-deletion Reconstruction Code

**Definition 5.12.** Two sequences $x$ and $y$ are Type-A-confusable if

$$x = uav, \text{ and } y = uav,$$

for some subsequences $a$, $u$, and $v$ such that $|a| \geq 2$, $\bar{a}$ is the complement of $a$, and $a = a_1a_2\ldots a_j$ is an alternating sequence.

The following characterization was demonstrated in [60].

**Lemma 5.13 (Type-A-confusability [60]).** Let $x$ and $y$ be binary words. We have that $|D_1(x) \cap D_1(y)| = 2$ if and only if $x$ and $y$ are Type-A-confusable.

Note that we want to avoid having two Type-A confusable sequences in an $(n, 2; D_1)$ reconstruction code. Therefore, the next step of our strategy is to find a collection of binary sequences, such that any two sequences in the collection are Type-A confusable. This means that we must remove all binary sequences in the collection but for one sequence.

To this end, we borrow graph theoretic tools and consider the graph $G(n)$ whose vertices correspond to $\{0, 1\}^n$. The vertices $x$ and $y$ are adjacent if and only if $|D_1(x) \cap D_1(y)| = 2$, or equivalently, $x$ and $y$ are Type-A-confusable.

Hence, $C$ is an $(n, 2; D_1)$-reconstruction code if and only if the corresponding set of vertices are independent in $G(n)$.

**Definition 5.14.** A collection $Q$ of cliques is a clique cover of $G$ if every vertex in $G$ belongs to some clique in $Q$.

We have the following fact from graph theory (see for example, [70]).

**Theorem 5.15.** If $Q$ is a clique cover, then the size of any independent set is at most $|Q|$.

Hence, our objective is to construct a clique cover for $G(n)$. To this end, we consider another parameter $\ell$, and set $m = \lfloor n/(2\ell) \rfloor$ and $r = n - 2\ell m$. We divide each word of length $n$ into $m$ blocks of length $2\ell$ and one block of length $r$. Set

$$\Lambda = \{(01)^j(10)^{\ell-j} : j \in [\ell]\} \cup \{(10)^j(01)^{\ell-j} : j \in [\ell]\}$$
Chapter 5. Correcting Deletions with Multiple Distinct Reads

where \((01)^0 = (10)^0\) is the empty word, and \(\bar{\Lambda} = \{0, 1\}^{2\ell} \setminus \Lambda\). So, \(|\Lambda| = 2\ell\) and \(|\bar{\Lambda}| = 2^{2\ell} - 2\ell\). To construct our clique cover \(Q(n, \ell)\), we consider two types of cliques. The first type of cliques are singletons of the form

\[ S_x = \{x\}, \text{ where } x \in \bar{\Lambda}^m \times \{0, 1\}^r. \]

The second type of cliques are cliques of size \(\ell\). Here, we define

\[ \Gamma = \left\{ (u, w, i) : u \in \bar{\Lambda}^{i-1}, w \in \{0, 1\}^{2\ell(m-i)+r}, i \in [m] \right\}, \]

where the \(\bar{\Lambda}^0 = \{0, 1\}^0\) is the set containing an empty word. For each \(z = (u, w, i)\), we define two sets of vertices (which we later show to be cliques of size \(\ell\)):

\[ Q^{(0)}_z = \{u(01)^j(10)^{\ell-j}w : j \in [\ell]\}, \]
\[ Q^{(1)}_z = \{u(10)^j(01)^{\ell-j}w : j \in [\ell]\}. \]

We then define

\[ Q(n, \ell) = \left\{ S_x : x \in \bar{\Lambda}^m \times \{0, 1\}^r \right\} \cup \{Q^{(0)}_z, Q^{(1)}_z : z \in \Gamma\}. \]

**Lemma 5.16.** \(Q(n, \ell)\) is a clique cover for \(\mathcal{G}(n)\).

**Proof.** Clearly, all singletons are cliques. Next, we show that the \(\ell\)-set \(Q^{(\mu)}_z\) is a clique for all \(z \in \Gamma\) and \(\mu \in \{0, 1\}\). We assume \(\mu = 0\) and the proof for \(\mu = 1\) is similar.

Let \(x = u(01)^i(10)^{\ell-i}w\) and \(y = u(01)^j(10)^{\ell-j}w\) be two words in \(Q^{(0)}_z\). Without loss of generality, let \(i < j\). Then we can rewrite \(x\) and \(y\) as

\[ x = u(01)^i(10)^{j-i}(10)^{\ell-j}w, \]
\[ y = u(01)^i(01)^{j-i}(10)^{\ell-j}w. \]

Thus, \(x\) and \(y\) are Type-A-confusable and so, \(x\) and \(y\) are adjacent in \(\mathcal{G}(n)\). Therefore, \(Q^{(0)}_z\) is a clique.

It remains to show that any word \(x \in \{0, 1\}^n\) belongs to some clique in \(Q(n, \ell)\). If \(x \in \bar{\Lambda}^m \times \{0, 1\}^r\), then \(x \in S_x\). Otherwise, \(x \notin \bar{\Lambda}^m \times \{0, 1\}^r\) and one of the \(m\) subblocks of \(x\) belongs to \(\Lambda\). Let the \(i\)th subblock be the first subblock from the
left that belongs to $\Lambda$. Hence, this subblock is either of the form $(01)^j(10)^{\ell-j}$ or $(10)^j(01)^{\ell-j}$ for some $j \in [\ell]$. In the first case, $x$ belongs to $Q^{(0)}_{(u,w,i)}$ where $u$ is the first $(i-1)$ subblocks and $w$ is the last $(m-i+1)$ subblocks. In the second case, $x$ belongs to $Q^{(1)}_{(u,w,i)}$ where $u$ and $w$ are similarly defined.

**Example 5.1.** Set $\ell = 2$ and so, $\Lambda = \{0110, 0101, 1001, 1010\}$. When $n = 12$ and $m = 3$, a possible element $z$ in $\Gamma$ is the triple $(0000, 1000, 2)$ and the cliques corresponding to $z$ are

$Q^{(0)}_z = \{000001101000, 00001011000\}$,

$Q^{(1)}_z = \{00010011000, 00010101000\}$.

For $n = 2m\ell$, since $|\tilde{\Lambda}| = 12$, the number of singletons is $12^m$. Furthermore, the number of $\ell$-cliques is $2|\Gamma|$. Since the size of $\Gamma$ is given by $\sum_{i=1}^{m} 12^{i-1} 2^{n-2(1-(3/4)^m)} = 2^{n-2}(1-(3/4)^m)$, we have that the size of the clique cover $\Omega(n,2)$ is

$2 \cdot \left(2^{n-2}(1-(3/4)^m)\right) + 12^m = 2^{n-1}(1+o(1))$.

So, $\log |\Omega(n,2)| = n - 1 + o(1)$ and an $(n,2;D_1)$-reconstruction code requires at least one redundant bit asymptotically.

To obtain the lower bound of $\log_2 \log_2 n - o(1)$ redundant bits, we refine our analysis by allowing $\ell$ to grow with $n$.

Now, we write $\lambda = |\tilde{\Lambda}| = 2^{2\ell} - 2\ell$. Similar to the analysis in Example 5.1, we have the following lemma.

**Lemma 5.17.** The size of $\Omega(n,\ell)$ is given by

$$2^n \left\{ \left(1 - \frac{2\ell}{2^{2\ell}} \right) \left\lfloor \frac{\lambda}{\ell} \right\rfloor + \frac{1}{\ell} \left(1 - \left(1 - \frac{2\ell}{2^{2\ell}} \right) \left\lfloor \frac{\lambda}{\ell} \right\rfloor \right) \right\}.$$

**Proof.** Recall that $n = 2m\ell + r$, where $0 \leq r < 2\ell$. The number of singletons is $2^r\lambda^m$, while the number of $\ell$-cliques is $2|\Gamma|$, where $|\Gamma| = \sum_{i=1}^{m} \lambda^{i-1} 2^{2\ell(m-i)+r}$. Hence, the size of $\Omega(n,\ell)$ is

$$2^r\lambda^m + 2 \sum_{i=1}^{m} \lambda^{i-1} 2^{2\ell(m-i)+r} = 2^r\lambda^m + 2^{n-2\ell+1} \left(\frac{\lambda}{2\ell}\right)^m - 1.$$

Straightforward manipulations then yield the lemma.
We set $\ell = \lceil \frac{1}{2} (1 - \epsilon) \log_2 n \rceil$ where $0 < \epsilon < 1$ and write $f(n) = (1 - \frac{2\ell}{2\ell})^{n/(2\ell)}$.

Hence,

$$\log_2 |Q(n, \ell)| = n - \log_2 \ell + \log_2 (1 + (\ell - 1)f(n))$$

$$\leq n - \log_2 \ell + \log_2 (1 + \ell f(n)).$$

Since $\log_2 \ell \geq \log_2 \log_2 n - O(1)$, it suffices to show that $\log_2 (1 + \ell f(n)) = o(1)$.

**Lemma 5.18.** We have that $\lim_{n \to \infty} \ell f(n) = 0$, or equivalently, $\lim_{n \to \infty} \ln(\ell f(n)) = -\infty$.

**Proof.** First, we show that

$$\lim_{n \to \infty} \frac{\ln \ell}{\ln f(n)} = 0. \quad (5.3)$$

Note that for $0 < x < 1$, we have $|\ln(1 - x)| \geq x$. Therefore,

$$\lim_{n \to \infty} \left| \frac{\ln \ell}{\ln f(n)} \right| = \lim_{n \to \infty} \left| \frac{\ln \ell}{\frac{n}{2^{2\ell}} \ln \left(1 - \frac{2\ell}{2\ell}\right)} \right|$$

$$\leq \lim_{n \to \infty} \left| \frac{\ln \ell}{\frac{n - 1}{2^{2\ell}}} \right|$$

$$= \lim_{n \to \infty} \left| \frac{2^{2\ell} \ln \ell}{n - 2\ell} \right|$$

$$\leq \lim_{n \to \infty} \left| \frac{2^{2\ell + (1 - \epsilon) \log_2 n \ln \ell}}{n - 2\ell} \right|$$

$$= 4 \lim_{n \to \infty} \left| \frac{n \ln \ell}{n^\epsilon (n - 2\ell)} \right| = 0$$

which implies (5.3). Note that since $\lim_{n \to \infty} \ln \ell = \infty$, and $f(n) < 1$ for sufficiently large $n$, combined with (5.3), this implies that $\lim_{n \to \infty} \ln f(n) = -\infty$. Therefore, together with (5.3), we have the following:

$$\lim_{n \to \infty} \ln(\ell f(n)) = \lim_{n \to \infty} (\ln \ell + \ln f(n))$$

$$= \lim_{n \to \infty} \ln(f(n)) \left( 1 + \frac{\ln \ell}{\ln f(n)} \right)$$

$$= \lim_{n \to \infty} \ln f(n) \lim_{n \to \infty} \left( 1 + \frac{\ln \ell}{\ln f(n)} \right) = -\infty. \quad \square$$

Therefore, the results in this section can be summarized in following theorem.
Theorem 5.19. Let \( C \) be an \((n, 2; D_1)\)-reconstruction code. For \( \epsilon > 0 \), we have that
\[
n - \log_2 |C| \geq \log_2 \log_2 n - 1 + \log_2 (1 - \epsilon) - o(1).
\]
(5.4)

Therefore, if we let \( \epsilon \to 0 \), we have \( \rho(n, 2; D_1) \geq \log_2 \log_2 n - 1 - o(1) \). Combining with Theorem 5.10, we have that \( \rho(n, 2; D_1) = \log_2 \log_2 n + \Theta(1) \), and therefore proven Theorem 5.11.

In the arXiv version of [60], Kiah et al. generalize the construction in Theorem 5.10 for the nonbinary alphabet. A similar analysis can also demonstrate these nonbinary reconstruction codes are also asymptotically optimal.

5.3 Two-deletion Reconstruction Code

In this section, we are going to discuss the past results regarding the best known \((n, N; D_2)\) reconstruction code for \( N = 1, N = 7, \) and \( N = 2n - 3 \) where each is going to be presented in their respective subsections. Furthermore, we are going to show our contribution regarding \((n, N; D_2)\) reconstruction code for \( N = n + 1, \) and \( N = 5 \). All the results for \( \rho(n, N; D_2) \) for all these subsections are summarized in Table 5.1.

5.3.1 \( N = 1 \)

The most basic case of an \((n, N; D_2)\)-reconstruction code is when \( N = 1 \), i.e. the classical two-deletion correcting code. From Theorem 5.6, we know that an optimal two-deletion correcting code must have redundancy between \( 2 \log_2 n + O(1) \) and \( 4 \log_2 n + O(1) \).

However, only recently did Guruswami and Håstad [75] propose the first known explicit construction for two-deletion correcting code that is within Levenshtein’s existential bound with redundancy \( 4 \log n + O(1) \). This is the best known two-deletion correcting code so far. As a summary, here are some techniques that are used in [75] as well as previous results pertaining to a two-deletion correcting code.
Chapter 5. Correcting Deletions with Multiple Distinct Reads

### Work Reads Redundancy Coding Techniques

<table>
<thead>
<tr>
<th>Work</th>
<th>Reads</th>
<th>Redundancy</th>
<th>Coding Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guruswami and Håstad, 2020 [75]</td>
<td>$N \geq 1$</td>
<td>$4 \log n + O(\log \log n)$</td>
<td>• First- and second-order VT syndromes on $x$ and “runs of $x$”&lt;br&gt;• Additional coding to handle deletions in small intervals</td>
</tr>
<tr>
<td>Chrisnata and Kiah, 2021 [86]</td>
<td>$N \geq 5$</td>
<td>$2 \log n + O(\log \log n)$</td>
<td>• First- and second-order VT syndromes on $x$&lt;br&gt;• Shifted VT syndrome and limited lengths for 2-periodic runs</td>
</tr>
<tr>
<td>Gabrys and Yaakobi, 2018 [61]</td>
<td>$N \geq 7$</td>
<td>$\log n + O(1)$</td>
<td>• First-order VT syndrome on $x$&lt;br&gt;• Any single-deletion correcting code has desired reconstruction capability.</td>
</tr>
<tr>
<td>Chrisnata, Kiah and Yaakobi, 2020 [81]</td>
<td>$N \geq n + 1$</td>
<td>$\log n + O(1)$</td>
<td>• Shifted VT syndrome and limited lengths for 2-periodic runs&lt;br&gt;• Any $(n, 2; D_1)$-reconstruction code has desired reconstruction capability.</td>
</tr>
<tr>
<td>Levenshtein, 2001 [59]</td>
<td>$N \geq 2n - 3$</td>
<td>0</td>
<td>• No coding</td>
</tr>
</tbody>
</table>

**Table 5.1:** Best known reconstruction codes for two-deletion channels

- **Higher order VT syndromes.** The VT syndrome of a binary sequence $x = x_1 x_2 \cdots x_n$ is given by $\text{VT}(x) \triangleq \sum_{i=1}^{n} c_i x_i$, where $c_i = i$. Levenshtein showed that binary sequences with the same VT syndrome modulo $n + 1$ form a single-deletion correcting code [67]. To correct more deletions, it is natural to generalize the VT syndrome so that $c_i$ is a higher-degree polynomial in $i$ (see Definition 5.31). However, a straightforward application of higher order VT syndromes does not work and the authors in [75, 78] considered the second-order VT syndrome of a vector related to $x$ and used these syndromes with other constraints to design two-deletion correcting codes.

- **Shifted VT syndromes.** Another variant of VT syndrome is the shifted VT syndrome introduced by Schoeny et al. to correct a burst of deletions [68]. Here, instead of taking the VT syndrome modulo $n + 1$, the authors considered the VT syndrome modulo $P$ where $P$ is much smaller than $n$. The objective is to reduce the number of redundant bits incurred by the syndrome. However, in order to correct errors, other side information are required and typically, techniques from constrained coding are introduced.

- **Runlength constraints.** In addition to satisfying certain VT syndrome conditions and parity checks, previous authors also imposed certain runlength constraints on the codewords. For example, in [68], to correct a single burst of deletions, the...
authors restricted the runlengths of consecutive zeroes or ones. Later, in [58, 60], in addition to runs of consecutive zeroes or ones, the authors restricted the lengths of alternating substrings and designed reconstruction codes that correct a single deletion with two reads.

**Theorem 5.20** ([75]). There exists an \((n, 1; D_2)\)-reconstruction code with \(4 \log_2 n + O(1)\) redundancy and hence \(\rho(n, 2; D_2) \leq 4 \log_2 n + O(1)\).

### 5.3.2 \(N = 2n - 3\)

Now, we focus on finding \((n, N; D_2)\), for \(N \geq 2\). When \(t = 2\), it follows from (5.2) that \(\nu_2(n) = 2D_1(n - 2) = 2n - 4\). Therefore we have the following corollary from Theorem 5.4.

**Corollary 5.21.** The minimum redundancy for a two-deletion channel with \(N = 2n - 3\) reads is 0, and hence \(\rho(n, N; D_2) = 0\) for \(N \geq 2n - 3\).

This means that if we transmit any binary sequence \(x\) (i.e. no coding), then \(2n - 3\) reads are sufficient to reconstruct \(x\).

### 5.3.3 \(N = 7\)

Recently, Gabrys and Yaakobi [61] showed that seven reads suffice to reconstruct the original codeword from a single-deletion-correcting code with \(\log n\) redundant bits.

**Theorem 5.22** ([61]). Let \(x\) and \(y\) be two words of length \(n \geq 7\). If \(D_1(x) \cap D_1(y) = \emptyset\), then \(|D_t(x) \cap D_t(y)| \leq N_t^{(1)}(n)\) for \(t \geq 2\), where

\[
N_t^{(1)}(n) = 2D_{t-2}(n - 4) + 2D_{t-2}(n - 5) + 2D_{t-2}(n - 7) + D_{t-3}(n - 6) + D_{t-3}(n - 7)
\]

\[
= \frac{6n^{t-2}}{(t-2)!} + O(n^{t-3}).
\]

Therefore, if \(C\) is an \((n, 1; D_1)\)-reconstruction code, then \(\nu(C; D_t) \leq N_t^{(1)}(n)\) and so, \(C\) is also an \((n, N_t^{(1)}(n) + 1; D_t)\)-reconstruction code for \(t \geq 2\) and \(n \geq 7\). Furthermore, this implies that \(\rho\left(n, N_t^{(1)}(n) + 1; D_t\right) \leq \log_2 n + O(1)\).
In particular, if we plug in \( t = 2 \) we have \( N_2^{(1)}(n) = 6 \), thus the following corollary follows.

**Corollary 5.23.** If \( C \) is an \( (n, 1; D_1) \)-reconstruction code then \( C \) is also an \( (n, 7; D_2) \)-reconstruction code for \( n \geq 7 \), and thus \( \rho(n, 7; D_2) \leq \log_2 n + O(1) \).

Therefore, the results from Theorem 5.20, Corollary 5.7 and 5.23 can be summarized in Table 5.1. Our contribution in this dissertation can also be seen in Table 5.1, namely for \( N = 5 \) and \( N = n + 1 \) reads, which will be exposed in the following subsections.

### 5.3.4 \( N = n + 1 \)

In this subsection, we study the sequence reconstruction problem when the codebook \( C \) is an \( (n, 2; D_1) \)-reconstruction code. Specifically, we will show that if every channel introduces \( t \) deletions, then it is possible to uniquely reconstruct codewords from \( C \) with approximately \( \nu_t(n)/2 \) reads.

In this section, we consider two words \( x \) and \( y \) whose single-deletion balls have intersection size one and study the size of intersection of their corresponding \( t \)-deletion balls for \( t \geq 2 \). In order to characterize pairs of sequences whose single-deletion balls have intersection size one, we need to define the following notion of confusability.

**Definition 5.24.** Two sequences \( x \) and \( y \) are Type-B-confusable if

\[
x = u\alpha\alpha v\omega w \quad \text{and} \quad y = u\alpha v\alpha b\omega w,
\]

or vice versa, for some subsequences \( u, v \) and \( w \), and \( a, b \in \{0, 1\} \).

And now we present the following main theorem of this subsection.

**Theorem 5.25.** Let \( x \) and \( y \) be binary words of length \( n \geq 7 \) and \( t \geq 2 \). If \( |D_1(x) \cap D_1(y)| = 1 \), then we have that

\[
|D_t(x) \cap D_t(y)| \leq D_{t-1}(n-3) + D_{t-2}(n-3) + 2D_{t-2}(n-5) \leq D_{t-1}(n-1) + \nu_{t-1}(n-3) = \frac{n^{t-1}}{(t-1)!} + O(n^{t-2}) \quad \text{for fixed values of } t.
\]
5.3. Two-deletion Reconstruction Code

Furthermore, the equality is achieved for even $n$, when $x = 1010(10)^m01$ and $y = 010(10)^m010$, where $m \geq 1$, and for odd $n$, when $x = 101(01)^m10$ and $y = 01(01)^m101$, where $m \geq 1$.

Since the proof of Theorem 5.25 is very technical and tedious, we put the detail proof in the Appendix B. Instead, we look at the implication of this theorem. Suppose that we have an $(n, 2; D_1)$-reconstruction code $C$. Then the intersection size of the single-deletion balls of any two codewords in $C$ is at most one. Applying Theorems 5.22 and 5.25, we have that the read coverage $\nu(C; D_t)$ is $N'_t(n) = \max \left\{ N_t^{(1)}(n), N_t^{(2)}(n) \right\}$, where $N_t^{(2)}(n) = D_{t-1}(n-3) + D_{t-2}(n-3) + 2D_{t-2}(n-5)$. Hence, $C$ is an $(n, N'_t(n) + 1; D_t)$-reconstruction code. For fixed values of $t$, we have that $N'_t(n) = N_t^{(2)}(n)$ for sufficiently large $n$ and also observe that $N_t^{(2)}(n) \sim \nu_t(n)/2$, or, $\lim_{n \to \infty} N_t^{(2)}(n)/\nu_t(n) = 1/2$. Therefore, by sacrificing $\log_2 \log_2 n + O(1)$ bits of information, the codes in Theorem 5.10 are able to uniquely reconstruct codewords with half the number of noisy reads (as compared to no coding). Note also that by Theorem 5.22, if the redundancy is roughly $\log_2 n$, then the number of noisy reads has to be $\frac{6n^{t-2}}{(t-2)!} + O(n^{t-3})$. We summarize our discussion with the following theorem.

**Theorem 5.26.** Let $n \geq 7$ and $t \geq 2$. Set $N_t^{(2)}(n) = D_{t-1}(n-3) + D_{t-2}(n-3) + 2D_{t-2}(n-5)$ and $N'_t(n) = \max \left\{ N_t^{(1)}(n), N_t^{(2)}(n) \right\}$. If $C$ is an $(n, 2; D_1)$-reconstruction code, then $C$ is also an $(n, N'_t(n) + 1; D_t)$-reconstruction code. Furthermore, when the value of $t$ is fixed, this implies that $\rho \left( n, N_t^{(2)}(n) + 1; D_t \right) \leq \log_2 \log_2 n + O(1)$.

Plugging in $t = 2$ to Theorem 5.26, we have that $N_2^{(2)}(n) = n + 1$, and thus using constrained SVT codes in Construction B, we are able to uniquely reconstruct the original sequence using $N = n + 2$ distinct noisy reads.

In what follows, we focus on this special case and show that constrained SVT codes in Construction B are able to uniquely reconstruct codewords with only $n + 1$ noisy reads. To do so, we require a stronger version of Theorem 5.25 for the case $t = 2$. The proof is also moved to Appendix B since it is related to the proof of Theorem 5.25.
Chapter 5. Correcting Deletions with Multiple Distinct Reads

**Theorem 5.27.** Let \( x \) and \( y \) be sequences of length \( n \geq 4 \) that are Type-B-confusable. If \( D_1(x) \cap D_1(y) = \{ z \} \) and then we have that

\[
\left| \left( D_2(x) \cap D_2(y) \right) \setminus D_1(z) \right| \leq 2.
\] (5.7)

Thus \( |D_2(x) \cap D_2(y)| \leq 2 + D_1(z) = 2 + n - 2 = n \). Therefore if \( \mathcal{C} \) is an \((n, 2; D_1)\)-reconstruction code, then \( \mathcal{C} \) is also an \((n, n + 1; D_2)\)-reconstruction code. Thus, \( \rho(n, n + 1; D_2) \leq \log_2 \log_2 n + O(1) \).

To conclude this section, we focus on the case \( t = 2 \) and show that by controlling the parameter \( P \) in Construction B, we are able to bound the number of noisy reads required to reconstruct a codeword. To do so, we make the following simple observation.

**Lemma 5.28.** Let \( z \) be a word of length \( n \). If the length of any alternating run in a word \( z \) is at most \( P \), then the number of runs in \( z \) is at most \( n - \lceil n/P \rceil + 1 \).

**Proof.** Let \( S = \{ i \in \mathbb{Z} : z_i = z_{i+1} \} \). We order the elements of \( S \) and call them \( s_1, s_2, ..., s_{|S|} \) from smallest to biggest. We want to show that \( |S| \geq \lceil n/P \rceil - 1 \). Note that \( s_{i+1} - s_i \leq P \) for all \( i \geq 1 \), \( s_1 \leq P \) and \( s_{|S|} \geq n - P \), since otherwise there would be an alternating run of length more than \( P \).

Suppose on the contrary that \( |S| < \lceil n/P \rceil - 1 \), this implies that \( s_{|S|} = s_1 + \sum_{i=1}^{S-1} s_{i+1} - s_i \leq |S|P \leq ([n/P] - 2)P < (n/P + 1 - 2)P = n - P \), which contradicts that \( s_{|S|} \geq n - P \). Therefore \( |S| \geq \lceil n/P \rceil - 1 \), and hence the number of runs in \( z \) is at most \( n - \lceil n/P \rceil + 1 \). \( \square \)

Recall that by design, the length of any alternating run of any codeword \( x \) in a constrained SVT code is at most \( P \). Hence, the same property holds for any word \( z \) in the single-deletion ball of \( x \). So, we can apply Lemma 5.28 and provide a tighter bound on the size of \( D_1(z) \).

**Proposition 5.29.** For any \( c \in \mathbb{Z}_{1+P/2} \) and \( d \in \mathbb{Z}_2 \), the constrained SVT code \( \mathcal{C}_{CSVT}(n, P; c, d) \) is an \((n, nP; D_2)\)-reconstruction code where \( N_P = \max\{n - \lceil n/P \rceil + 4, 7\} \).

**Proof.** Let \( x \) and \( y \) be distinct codewords. Then \( |D_1(x) \cap D_1(y)| \leq 1 \) and it remains to show that \( |D_2(x) \cap D_2(y)| < N_P \).
If $|\mathcal{D}_1(x) \cap \mathcal{D}_1(y)| = 0$, then Theorem 5.22 states that $|\mathcal{D}_2(x) \cap \mathcal{D}_2(y)| \leq 6 < N_P$.

If $|\mathcal{D}_1(x) \cap \mathcal{D}_1(y)| = 1$, then let $\{z\} = \mathcal{D}_1(x) \cap \mathcal{D}_1(y)$. Since the length of any alternating run of $x$ is at most $P$, by Lemma 5.28, the number of runs in $x$ is at most $n - \lceil n/P \rceil + 1$. Since $z$ is a subword of $x$, the number of runs in $z$ is at most the number of runs in $x$, which is $n - \lceil n/P \rceil + 1$. Applying (5.7), we have that $|\mathcal{D}_2(x) \cap \mathcal{D}_2(y)| \leq |\mathcal{D}_1(z)| + 2 \leq n - \lceil n/P \rceil + 3 < N_P$, as required.

Let $P \geq 4$. It is well-known (see for example, [58]) that the number of length-$n$ words whose 2-periodic run is at most $P$ is $4F_{P-1}(n-2)$, where

$$F_\ell(n) = \begin{cases} 
2^n, & \text{if } 0 \leq n \leq \ell - 1, \\
\sum_{i=1}^\ell F_\ell(n-i), & \text{otherwise.}
\end{cases}$$

Hence, we have the following lower bound on the size of a reconstruction code.

**Corollary 5.30.** For $P \geq 4$, set $N_P = \max\{n - \lceil n/P \rceil + 4, 7\}$. Then there exists an $(n, N_P; \mathcal{D}_2)$-reconstruction code of size at least $4F_{P-1}(n-2)/(P+2)$.

To end this section, for codelengths $n \in \{127, 255, 1023\}$, we vary the parameter $P$ in the constrained SVT codes and compute the corresponding values of $N_P$ and redundancy. The numerical results are given in Table 5.2. As expected, as we decrease the value of $P$, the number of required reads also decreases. However, the number of redundant bits also increases significantly and in this case (where $P$ is small), the VT code uses significantly less redundant bits. For completeness, we list the values of read-coverage and redundancy of a VT code of length $n$ and the space $\{0, 1\}^n$ (corresponding to the uncoded case).

### 5.3.5 $N = 5$

Before presenting our main contribution for this work: a class of $(n, 5; \mathcal{D}_2)$-reconstruction codes with $2\log n + 2\log\log n + O(1)$ redundant bits, we introduce higher order VT-syndromes as follows.

**Definition 5.31.** Fix $k \geq 1$. Let $x = x_1x_2\cdots x_n \in \{0, 1\}^n$. We define its $k$-th order VT syndrome is given by $VT^{(k)}(x) \triangleq \sum_{i \geq 1} \binom{k-1+i}{k} x_i$. 

---

Technion - Computer Science Department - Ph.D. Thesis PHD-2021-16 - 2021
Table 5.2: List of constrained SVT codes and their read coverage and redundancy

<table>
<thead>
<tr>
<th>$n$</th>
<th>$P$</th>
<th>Read Coverage</th>
<th>Redundancy</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>127</td>
<td>–</td>
<td>6</td>
<td>7.00</td>
<td>VT code</td>
</tr>
<tr>
<td>127</td>
<td>6</td>
<td>108</td>
<td>6.016</td>
<td>–</td>
</tr>
<tr>
<td>127</td>
<td>8</td>
<td>114</td>
<td>4.018</td>
<td>–</td>
</tr>
<tr>
<td>127</td>
<td>10</td>
<td>117</td>
<td>3.753</td>
<td>–</td>
</tr>
<tr>
<td>127</td>
<td>–</td>
<td>250</td>
<td>0.00</td>
<td>{0, 1}^n</td>
</tr>
<tr>
<td>255</td>
<td>–</td>
<td>6</td>
<td>8.00</td>
<td>VT code</td>
</tr>
<tr>
<td>255</td>
<td>8</td>
<td>226</td>
<td>4.762</td>
<td>–</td>
</tr>
<tr>
<td>255</td>
<td>10</td>
<td>232</td>
<td>3.935</td>
<td>–</td>
</tr>
<tr>
<td>255</td>
<td>12</td>
<td>236</td>
<td>3.894</td>
<td>–</td>
</tr>
<tr>
<td>255</td>
<td>–</td>
<td>506</td>
<td>0.00</td>
<td>{0, 1}^n</td>
</tr>
<tr>
<td>1023</td>
<td>–</td>
<td>6</td>
<td>10.00</td>
<td>VT code</td>
</tr>
<tr>
<td>1023</td>
<td>8</td>
<td>898</td>
<td>9.22</td>
<td>–</td>
</tr>
<tr>
<td>1023</td>
<td>10</td>
<td>923</td>
<td>5.03</td>
<td>–</td>
</tr>
<tr>
<td>1023</td>
<td>12</td>
<td>940</td>
<td>4.17</td>
<td>–</td>
</tr>
<tr>
<td>1023</td>
<td>14</td>
<td>952</td>
<td>4.09</td>
<td>–</td>
</tr>
<tr>
<td>1023</td>
<td>–</td>
<td>2042</td>
<td>0.00</td>
<td>{0, 1}^n</td>
</tr>
</tbody>
</table>

Remark 5.3. Note that when $k = 1$, we recover the standard VT syndrome.
When $k = 2$, then $VT^{(2)}(x) = 1 \cdot x_1 + (1 + 2) \cdot x_2 + (1 + 2 + 3) \cdot x_3 + \cdots + (1 + 2 + \cdots + n) \cdot x_n$.

Now, we are ready to present our construction for $(n, 5; D_2)$-reconstruction code.

Construction C. For $n \geq P > 0$ and $P$ even, let $c_1 \in \mathbb{Z}_{nP+1}$ and $c_2 \in \mathbb{Z}_{nP+P^2+1}$.
Let $\mathcal{C}(n, P; c_1, c_2)$ be the set of all sequences $x = x_1 x_2 \cdots x_n$ such that the following holds.

1. $VT^{(1)}(x) = c_1 \pmod{nP+1}$.
2. $VT^{(2)}(x) = c_2 \pmod{nP+P^2+1}$.
3. Each 2-periodic run in $x$ has length at most $P$.

To show that $\mathcal{C}(n, P; c_1, c_2)$ has the desired reconstruction capabilities, we need the following theorem.
Theorem 5.32. For any two distinct codewords $x, y \in \mathcal{C} \triangleq \mathcal{C}(n, P; c_1, c_2)$, we have that $|D_2(x) \cap D_2(y)| \leq 4$. Furthermore, if we set $P = \lceil \log_2 n \rceil + 2$, the code $\mathcal{C}$ has redundancy $2 \log n + 2 \log \log n + O(1)$ for some choice of $c_1$ and $c_2$. Thus, $\rho(n, 5; D_2) \leq 2 \log n + 2 \log \log n + O(1) = 2 \log n + o(\log n)$.

However, due to the high technicality of the proof of Theorem 5.32, we move the proof to Appendix C.

5.4 General $t$-deletion Reconstruction Code

Finally for general $t \geq 3$, the current best known $t$-deletion correcting code is proposed by Sima and Gabrys, 2020 [77] with redundancy $4t \log_2 n + o(\log_2 n)$. Therefore we know that $\rho(n, 1; D_t) \leq 4 \log_2 n + o(\log_2 n)$.

However, for $N \geq 2$, not much is known about $t$-deletion reconstruction code. The trivial codes that we know would be $\mathcal{C} = \{0, 1\}^n$ and hence by Theorem 5.4, we know that $\nu_t(n) = 2D_{t-1}(n-2)$. Thus, we can conclude that $\rho(n, N; D_t) = 0$ for $N \geq 2D_{t-1}(n-2) + 1$.

Lastly, quoting Theorem 5.26, we know that for fixed $t$, $\rho\left(n, N_t^{(2)}(n) + 1; D_t\right) \leq \log_2 \log_2 n + O(1)$ for $n \geq 7$.

5.5 Conclusion

We discuss about a variant of the sequence reconstruction problem where the number of noisy reads $N$ that the decoder receives is fixed. In that case, we provide some bounds on the value of $\rho(n, N; D_t)$ for various values of $N$ and $t$.

Of particular interest is when $t = 1$, namely the single-deletion reconstruction code, where we show that the code in Construction B is asymptotically optimal for $N = 2$ noisy reads. Using graph theoretic approach, namely clique cover of a graph, we show that the minimum redundancy required to reconstruct the original sequence in a single-deletion channel with two noisy reads is at least $\log_2 \log_2 n + \Theta(1)$. This revelation gives a complete picture of the optimal redundancy of reconstruction code for single deletion channel, which is summarized in Theorem 5.11.
In the case of $t = 2$, namely the two-deletion reconstruction code, we know that $\nu_2(n) = 2n - 4$, and therefore there are more options for the number $N$ of distinct noisy reads to consider. As far as this chapter is concerned, we focus on the case where $N = n + 1$, and hence the number of noisy reads is about half of $\nu_2(n)$. We found from Theorem 5.26 that $\rho(n, n + 1; D_2) \leq \log_2 \log_2 n + O(1)$. Furthermore, using codes in Construction C, we can reconstruct the original sequence using 5 noisy reads. Therefore, we conclude that $\rho(n, 5; D_2) \leq 2 \log_2 n + o(\log_2 n)$. Lastly, we also provide some upper bounds for the redundancy of the general $t$-deletion reconstruction codes.
Chapter 6

Concluding Remarks

Binary sequences are widely used in the exchange of information through a communication channel. Thus, it is of utmost importance to investigate the properties of sequences to provide a more reliable and error-free communication channels. In this thesis we have given a detailed but concise introduction to some properties of sequences, namely the linear complexity of sequences and sequence reconstruction problem for deletion channels in Chapter 2.

One property of sequence that is of interest is the linear complexity of sequences. In Chapter 3, we propose a direct-generalization to the Games-Chan algorithm, namely the Powers of Irreducible Polynomial (PIP) algorithm that can find linear complexity of binary sequences. Furthermore, using this PIP algorithm, we provide a general method to find the linear complexity of binary sequences with certain periods in linear time. Our algorithm also provides the minimal polynomial that generates the binary sequence. However, our algorithm may not be efficient (may not take linear time) for sequences of certain periods in general. In particular, our algorithm may not be efficient for binary sequences of period $N$, where $z^N - 1$ has a lot of irreducible polynomial factors. Thus, further investigations need to be done to cater to these types of sequences.

The second property that we discuss in this dissertation is the reconstruction capability of sequences from their subsequences. In Chapter 4, we assume that in the deletion channel, the decoder receives all possible subsequences of a certain length $k$ of the original sequence. This multiset of all subsequences of length $k$ is also called the $k$-deck of a sequence. We provide the trellis-based method to
compute the number of distinct $k$-decks of binary sequences of length $n$. From this number, we know the upper bound of the size of a code in a deletion channel that provides all subsequences of length $k$ as noisy reads. However, tight upper and lower bounds of the number of distinct $k$-decks are hard to enumerate for higher $k \geq 4$ and general $n$. Therefore, other enumeration tools can be explored for higher values of $k \geq 4$.

Finally, Chapter 5 discusses a variant of the sequence reconstruction problem, in which we assume that the decoder receives a fixed number of noisy reads. We analyze the properties of $t$-deletion balls and the intersections between them. Based on the result of these investigations, we show that an already existing code construction that can correct one deletion with two noisy reads is asymptotically optimal. Furthermore, using higher order VT syndromes, we provide a new code construction that can correct two deletions with five distinct noisy reads with only $2 \log n + o(\log n)$ redundancy. However, investigations of the intersections of $t$-deletion balls for $t \geq 3$ are complicated and very technical. Therefore, other tools and further research are needed to construct a good $(n, N; D_t)$-reconstruction code for $t \geq 3$ and constant $N$. 
Appendix A

Proof of Theorem 3.18

This appendix is for Chapter 3, Section 3.6, Theorem 3.18.

A.1 Proof of Theorem 3.18 part 1

Firstly, we consider the number of bit operations required to compute the linear complexity of binary sequences whose period is $3 \cdot 2^n$, which is stated in Theorem 3.18 part 1. In order to show part 1 we need the following few theorems.

Theorem A.1. Let $y$ be a binary cyclic sequence of the form $[X + Y, Y + Z, Z + X]$ for some binary sequences $X, Y, Z$ of length $2^{n-k+1}$. If $x = [X, Y, Z]$ is the binary input sequence, then the following properties hold.

(P1) There exist binary sequences $X', Y', Z'$ of length $2^{n-k}$, such that $(E^2 + E + 1)^{2^{n-k}} y = [X' + Y', Y' + Z', Z' + X']$,

(P2) Obtaining the binary sequences $X', Y', Z'$ from (P1) requires at most $3 \cdot 2^{n-k}$ bit operations,

(P3) If $(E^2 + E + 1)^{2^{n-k}} y = 0$, then there exist binary sequences $X'', Y'', Z''$ of length $2^{n-k}$, such that $y = [X'' + Y'', Y'' + Z'', Z'' + X'']$,

(P4) Obtaining the binary sequences $X'', Y'', Z''$ from (P3) requires no computation.
Proof. Suppose that $X, Y, Z$ are binary sequences of length $2^{n-k+1}$, and $y = [X + Y, Y + Z, Z + X]$ is a binary sequence of length $3 \cdot 2^{n-k+1}$. Let $X = [X_1X_2], Y = [Y_1Y_2], Z = [Z_1Z_2]$ where $X_1, X_2, Y_1, Y_2, Z_1, Z_2$ are of length $2^{n-k}$. Then we have the following equalities for $(E^2 + E + 1)^{2^{n-k}}y = (E^{2^{n-k+1}} + E^{2^{n-k}} + 1)y$ (where variables on the same column in the related matrices are supposed to be added together),

$$(E^2 + E + 1)^{2^{n-k}}y = \begin{bmatrix} X_1 + Y_1 & X_2 + Y_2 & Y_1 + Z_1 & Y_2 + Z_2 & Z_1 + X_1 & Z_2 + X_2 \\ X_2 + Y_2 & Y_1 + Z_1 & Y_2 + Z_2 & Z_1 + X_1 & Z_2 + X_2 & X_1 + Y_1 \\ Y_1 + Z_1 & Y_2 + Z_2 & Z_1 + X_1 & Z_2 + X_2 & X_1 + Y_1 & X_2 + Y_2 \\ Z_1 + X_1 & Z_2 + X_2 & X_1 + Y_1 & X_2 + Y_2 & Y_1 + Z_1 & Y_2 + Z_2 \\ Z_2 + X_2 & X_1 + Y_1 & X_2 + Y_2 & Z_1 + X_1 & Z_2 + X_2 & X_1 + Y_1 \\ X_2 + Y_2 & Y_1 + Z_1 & Y_2 + Z_2 & Z_1 + X_1 & Z_2 + X_2 & X_1 + Y_1 \end{bmatrix}.$$ 

Now, if we let

$$X' = Y_2 + X_1,$$
$$Y' = X_2 + Z_1,$$
$$Z' = Y_1 + Z_2,$$

then $(E^2 + E + 1)^{2^{n-k}}s = [X' + Y', Y' + Z', Z' + X']$, where $X', Y', Z'$ are binary sequences of length $2^{n-k}$, which implies (P1).

Furthermore, $X', Y', Z'$ can be computed using $3 \cdot 2^{n-k}$ bit operations since $X_1, X_2, Y_1, Y_2, Z_1, Z_2$ are given as inputs of length $2^{n-k}$, which implies (P2).

If $(E^2 + E + 1)^{2^{n-k}}y = 0$, then by (P1) we have that

$$(E^2 + E + 1)^{2^{n-k}}y = \begin{bmatrix} Z_1 + X_1 & Z_2 + X_2 & X_1 + Y_1 \\ X_2 + Y_2 & Y_1 + Z_1 & Y_2 + Z_2 \end{bmatrix} = 0.$$
which implies that
\[
Z_1 + X_1 = X_2 + Y_2, \\
Z_2 + X_2 = Y_1 + Z_1, \\
X_1 + Y_1 = Y_2 + Z_2.
\]

Therefore,
\[
y = \left[ X + Y \ Y + Z \ Z + X \right] \\
= \left[ X_1 + Y_1 \ X_2 + Y_2 \ Y_1 + Z_1 \ Y_2 + Z_2 \ Z_1 + X_1 \ Z_2 + X_2 \right] \\
= \left[ X_1 + Y_1 \ X_1 + Z_1 \ Y_1 + Z_1 \right] \\
= \left[ X'' + Y'' \ Y'' + Z'' \ Z'' + X'' \right],
\]
where \( X'' = Y_1, Y'' = X_1, Z'' = Z_1 \) are all given inputs which require no computation and hence (P3) and (P4) are implied.

**Theorem A.2.** Let \( x \) be a binary cyclic sequence of length \( 3 \cdot 2^n \) for some non-negative integer \( n \). The computation of the linear complexity of \( x \) requires at most \( 7 \cdot 2^n + 2^n \) bit operations.

**Proof.** Since \( x \) is a binary cyclic sequence of length \( 3 \cdot 2^n \), we have that \((E^{3 \cdot 2^n} + 1)x = 0\). The polynomial \( E^{3 \cdot 2^n} + 1 \) is factorized into irreducible polynomials, i.e., \( E^{3 \cdot 2^n} + 1 = (E^2 + E + 1)^{2^n}(E + 1)^{2^n} \). To compute the linear complexity of \( x \), it is required to find the smallest \( i \) and \( j \), such that \((E^2 + E + 1)^i(E + 1)^jx = 0\).

Computing the linear complexity using our algorithm can be done in two steps:

(A) Find the smallest \( i \) such that \((E^2 + E + 1)^i(E + 1)^{2^n}x = 0\).

(B) Find the smallest \( j \) such that \((E + 1)^j(E^2 + E + 1)^{2^n}x = 0\).

Let \( x = \left[ X \ Y \ Z \right] \) be the input sequence and consider first the computations for step (A). Let \( y = (E + 1)^{2^n}x = \left[ X + Y \ Y + Z \ Z + X \right] \). First, we initialize \( x' \) to be \( y \), and at the \( k \)-th iteration of PIP algorithm, we compute \((E^2 + E + 1)^{2^n-k}x'\). But instead of computing it, from Theorem A.1 (P1), we can just compute \( X', Y', Z' \) such that \((E^2 + E + 1)^{2^n-k}x' = [X' + Y', Y' + Z', Z' + X'] \), which requires \( 3 \cdot 2^{n-k} \) bit operations.
Now, to verify whether \[X' + Y' + Z' + X'] = 0, we can observe that \[X' + Y' + Z' + Z' + X'] = 0 if and only if \[X' + Y' + Z'] = 0. Therefore only \(2 \cdot 2^{n-k}\) bit operations are required to compute \(X' + Y' + Z' + X'\). If the computation leads to a nonzero result, then we proceed to the next iteration having \[X' + Y' + Z' + Z' + X']\) as the new \(x'\). Otherwise, if the computation leads to a zero result, then by (P3) of Theorem A.1, we set \[X'' + Y'' + Z'' + X''\] as the new \(x'\) which requires no further computation. In total, for the \(k\)-th iteration, the algorithm requires at most \(5 \cdot 2^{n-k}\) bit operations. Therefore, to perform step (A), the algorithm (together with the integer additions) requires at most \(5 \cdot 2^n + n\) bit operations.

Next, we consider the number of bit operations required to perform step (B). Note that \((E^2 + E + 1)^2^n x = X + Y + Z\), which is a binary sequence of length \(2^n\). We will proceed with The Games-Chan Algorithm, where in the first iteration we compute \((Y_2 + X_1) + (X_2 + Z_1) + (Y_1 + Z_2) = X' + Y' + Z'\). Since \[X' + Y'\] and \(Z'\) are already computed in step (A) we only need \(2^{n-1}\) bit operations for the first iteration. If the computation leads to 0, then before the second iteration, we compute the first \(2^{n-1}\) bits of \(X + Y + Z\), namely \(X_1 + Y_1 + Z_1\), which requires \(2 \cdot 2^{n-1}\) bit operations. Otherwise, if the computation leads to nonzero, we proceed to the second iteration with \[X' + Y' + Z'\] of length \(2^{n-1}\) which is already computed. Hence, the first iteration requires at most \(3 \cdot 2^{n-1}\) bit operations and at most \(2^{n-2} + 2^{n-3} + \cdots + 1 \leq 2^{n-1}\) bit operations are required from the second iteration to the last iteration. Therefore, step (B) of the algorithm (together with the integer additions) requires at most \(2 \cdot 2^n + n\) bit operations.

Thus, in total, the algorithm requires at most \(7 \cdot 2^n + 2n\) bit operations. \(\square\)

## A.2 Proof of Theorem 3.18 part 2 and part 3

Since Theorem 3.18 part 2 is covered in Theorem 3.18 part 3, we will just show the more general part, which is part 3.

Let \(p\) be a prime such that 2 is a generator in the multiplicative group modulo \(p\), where \(p = 4\ell + 1\), for some positive integer \(\ell\). Let \(x\) be the binary cyclic input
sequence of length $p \cdot 2^n$, which implies that $(E^{p \cdot 2^n} + 1)x = 0$. The polynomial $E^{p \cdot 2^n} + 1$ is factorized into irreducible polynomials, i.e.,

$$E^{p \cdot 2^n} + 1 = (E^{p-1} + \cdots + E^2 + E + 1)^{2^n}(E + 1)^{2^n}.$$ 

To compute the linear complexity of $x$, it is required to find the smallest $i$ and $j$, such that

$$(E^{p-1} + \cdots + E^2 + E + 1)^i(E + 1)^j x = 0.$$ 

Computing the linear complexity using our algorithm can be done in two steps:

(A) Find the smallest $i$ such that $(E^{p-1} + \cdots + E^2 + E + 1)^i(E + 1)^{2^n} x = 0$,

(B) Find the smallest $j$ such that $(1 + E)^j(E^{p-1} + \cdots + E^2 + E + 1)^{2^n} x = 0$.

Similar to theorem A.1 one can prove the following two theorems.

**Theorem A.3.** Let $x$ be a binary cyclic sequence of length $p \cdot 2^k$, where $p$ is a prime such that $2$ is a generator in the multiplicative group modulo $p$, and $k \geq 0$. If $x$ is the binary input sequence, then the following properties hold:

1. $(E^{p-1} + \cdots + E^2 + E + 1)^{2^k-1}(E + 1)^{2^k} x = (E + 1)^{2^k-1} x'$, for some binary sequence $x'$ of length $p \cdot 2^{k-1}$;

2. the computation of $x'$ requires $p \cdot 2^{k-1}$ bit operations;

3. $(E + 1)^{2^{k-1}} x' = 0$ if and only if the first $(p - 1) \cdot 2^{k-1}$ bits of $(E + 1)^{2^{k-1}} x'$ equal to $0$. Hence, only $(p - 1) \cdot 2^{k-1}$ bit operations are required to check if $(E + 1)^{2^{k-1}} x' = 0$.

**Theorem A.4.** Let $x$ be a binary sequence of length $p \cdot 2^k$, where $p$ is a prime such that $2$ is a generator in the multiplicative group modulo $p$, $p = 4\ell + 1$ for some positive integer $\ell$, and $k \geq 0$. Suppose that $(E^{p-1} + \cdots + E^2 + E + 1)^{2^{k-1}}(E + 1)^{2^k} x = 0$, then given $x$ as input, the following things hold,

1. If $y = (E + 1)^{2^k} x$, then the first half of $y$ is the same as the second half of $y$, which means that $y$ is a cyclic sequence of period at most $p \cdot 2^{k-1}$.

2. $(E^{p-1} + \cdots + E^2 + E + 1)^{2^{k-2}}(E + 1)^{2^k} x$ can be computed using $(3p + 1) \cdot 2^{k-2}$ bit operations.
Theorem A.5. Step (A) requires at most $\frac{p^2-+7p-1}{4} \cdot 2^n$ bit operations.

Proof. Let $y = (E + 1)^2 x$. To perform step (A) we will follow our PIP algorithm on the sequence $y$.

(1) The first iteration of our PIP algorithm computes $(E^{p-1} + \cdots + E^2 + E + 1)^{2n-1} y$ to check if it is equal to the zero sequence. However, it follows from Theorem A.3, that when $k = n$ the computation of $(E^{p-1} + \cdots + E^2 + E + 1)^{2n-1} y$ is not required. Instead, we compute a binary sequence $x'$ of length $p \cdot 2n-1$, where $(E^{p-1} + \cdots + E^2 + E + 1)^{2n-1} y = (E + 1)^{2n-1} x'$, and the first $(p - 1) \cdot 2^n-1$ bits of $(E + 1)^{2n-1} x'$, which require $(2p - 1) \cdot 2^n-1$ bit operations for the first iteration. If $(E + 1)^{2n-1} x'$ is computed to be 0, then go to step (2); otherwise, go to step (3).

(2) The second iteration of our algorithm computes $(E^{p-1} + \cdots + E^2 + E + 1)^{2n-2} y$. It follows from Theorem A.4, that when $k = n$, it requires $(3p + 1) \cdot 2^n-2$ bit operations. If $(E^{p-1} + \cdots + E^2 + E + 1)^{2n-2} y$ is equal to 0, then go to step (4); otherwise, go to step (5).

(3) The second iteration of our algorithm computes $(E^{p-1} + \cdots + E^2 + E + 1)^{2n-2} (E + 1)^{2n-1} x'$. However, it follows again from Theorem A.3, that when $k = n - 1$, the second iteration of the algorithm can be done by computing a binary sequence $x''$ of length $p \cdot 2n-2$, where $(E^{p-1} + \cdots + E^2 + E + 1)^{2n-2} (E + 1)^{2n-1} x' = (1 + E)^{2n-2} x''$, and the first $(p - 1) \cdot 2^n-2$ bits of $(E + 1)^{2n-2} x''$, which require $(2p - 1) \cdot 2^n-2$ bit operations in total for the second iteration. If $(E + 1)^{2n-2} x''$ is computed to be 0, then go to step (6); otherwise, go to step (7).

(4) Now we have $(E^{p-1} + \cdots + E^2 + E + 1)^{2n-1} r = 0$ from step (1) and $(E^{p-1} + \cdots + E^2 + E + 1)^{2n-2} y = 0$ from step (2). It implies by Theorem A.4 that $y$ is a sequence of period at most $p \cdot 2^n-1$. Furthermore, note that $(E^{p-1} + \cdots + E^2 + E + 1)^{2n-2} y = (E^{p-1} + \cdots + E^2 + E + 1)^{2n-2} (E^{p-1} + \cdots + E^2 + E + 1)^{2n-1} y = (E + 1)^{2^n-2} 0 = 0$. Since $y$ is a sequence of period at most $p \cdot 2^n-1$, then $(E^{p-1} + \cdots + E^2 + E + 1)^{2n-2} y = 0$ implies that $y$ is of period at most $p \cdot 2^n-2$. Let $z$ be the first $p \cdot 2^n-2$ bits of $y$, which takes $p \cdot 2^n-2$ bit operations to compute. Go to step (8).

(5) Let $z$ be the first $p \cdot 2^n-2$ bits of $(E^{p-1} + \cdots + E^2 + E + 1)^{2n-2} y$ which is already computed in step (2). Go to step (8).
Appendix A. Proof of Theorem 3.18

(6) From step (3), we have \((E^{p-1} + \cdots + E^2 + E + 1)^{2^{n-2}}(E + 1)^{2^{n-1}}x' = 0\). Then it follows from Theorem A.4, that when \(k = n - 1\), \((E + 1)^{2^{n-1}}x'\) is a binary sequence of period at most \(p \cdot 2^{n-2}\). Let \(z\) be the first \(p \cdot 2^{n-2}\) bits of \((E + 1)^{2^{n-1}}x'\) whose computation requires \(p \cdot 2^{n-2}\) bit operations, and go to step (8).

(7) Let \(z\) be the first \(p \cdot 2^{n-2}\) bits of \((E + 1)^{2^{n-2}}x''\). Since the first \((p - 1) \cdot 2^{n-2}\) bits of \((1 + E)^{2^{n-2}}x''\) has been computed in step (3), we just need to compute the last \(2^{n-2}\) which requires \(2^{n-2}\) bit operations. Go to step (8).

(8) For any \(m \geq 3\), the \(m\)-th iteration of our algorithm starts with a binary sequence \(z\) of length \(p \cdot 2^{n-m+1}\), and compute the first \(p \cdot 2^{n-m}\) bits of \((E^{p-1} + \cdots + E^2 + E + 1)^{2^{n-m}}z\) which requires \((p - 1) \cdot p \cdot 2^{n-m}\) bit operations. If it is nonzero then we replace \(z\) with the computed sequence, otherwise we replace \(z\) with the first \(p \cdot 2^{n-m}\) bits of \(z\). Repeat step (8).

To perform the first and the second iteration, namely step (1) through step (7), it requires at most \((2p - 1) \cdot 2^{n-1} + (3p + 1) \cdot 2^{n-2} + p \cdot 2^{n-2} = (8p - 1) \cdot 2^{n-2}\) bit operations.

To perform the third iteration onwards, namely step (8), it requires at most \((p - 1) \cdot p \cdot 2^{n-2}\) bit operations. Therefore, in total, we require at most \(\frac{p^2 + 7p - 1}{4} \cdot 2^n\) bit operations to perform step (A).

\(\square\)

Theorem A.6. Step (B) requires at most \(2^{n+1}\) bit operations.

Proof. To perform step (B), our algorithm first computes \(y = (E^{p-1} + \cdots + E^2 + E + 1)^{2^n}x\), which requires \(2^n\) bit operations. Then, we perform The Games-Chan algorithm on the sequence \(y\) to find its complexity, which requires \(2^n\) bit operations. Therefore, in total, at most \(2 \cdot 2^n\) bit operations are required to perform step (B).

\(\square\)

Corollary A.7. Let \(x\) be a binary cyclic sequence of length \(N = p \cdot 2^n\), where \(p = 4\ell + 1\), and \(p\) is a generator in the multiplicative group modulo \(p\). The algorithm requires \(\frac{p^2 + 7p + 1}{4} \cdot 2^n\) bit operations to compute the complexity of \(x\).

Corollary A.8.

1. If \(p = 5\), the algorithm requires \(16 \frac{3}{4} \cdot 2^n = 3.35N\) bit operations,
2. if $p = 13$, the algorithm requires $66 \frac{3}{4} \cdot 2^n = 5.14N$ bit operations,

3. if $p = 29$, the algorithm requires $262 \frac{3}{4} \cdot 2^n = 9.1N$ bit operations.
Appendix B

Proof of Theorem 5.25 and Theorem 5.27

As the proof of Theorem 5.25 is fairly technical, we outline our proof strategy.

- First, in Section B.1, we provide a characterization lemma similar to Lemma 5.13. Specifically, we describe the necessary conditions for a pair of sequences to have single-deletion balls intersecting at exactly one output sequence.

- Applying the characterization lemma, we consider pairs of sequences with certain properties. In Section B.2, we analyse the intersection size of certain \( t \)-deletion balls under certain scenarios.

- Finally, in Section B.3, we use an inductive argument to complete the proof.

We also provide the proof of Theorem 5.27 in Section B.4.

B.1 Type-B-Confusability

Recall that

**Definition B.1.** Two sequences \( x \) and \( y \) are Type-B-confusable if

\[
x = uvbw \quad \text{and} \quad y = uwbvbw.
\]
Suppose that

Next, we borrow certain notation from [61]. Let \( S \) be a set of binary sequences and \( a, b \in \{0, 1\} \). We define \( S^a \) to be the set of all sequences in \( S \) that start with \( a \) and \( S_b \) to be the set of all sequences in \( S \) that end with \( b \). We also combine both notations and let \( S^a_b \) be the set of all sequences in \( S \) that start with \( a \) and end with \( b \). If \( x \) is a sequence, we define \( S \circ x \) (or \( x \circ S \)) to be set of all sequences obtained by appending (or prepending) \( x \) to every sequence in \( S \).

**Lemma B.2.** Let \( x \) and \( y \) be two binary sequences. If \( |D_1(x) \cap D_1(y)| = 1 \), then either the Hamming distance of \( x \) and \( y \) is one or \( x \) and \( y \) are Type-B-confusable.

**Proof.** Suppose that \( x \) and \( y \) have Hamming distance at least two. Then \( x \) and \( y \) must be of the form

\[
uaebw \text{ and } uaebw
\]

for subsequences \( u, w, d, \) and \( e \), where \( |d| = |e| \), and \( a, b \in \{0, 1\} \).

Without loss of generality, suppose that \( x = uadbw \) and \( y = uaebw \). If \( d \) is empty, then \( x = uabw \) and \( y = uawb \). If \( a = b \), then their weight differ by two and hence \( |D_1(x) \cap D_1(y)| = 0 \) which contradicts our assumption. Else, if \( a \neq b \), then by definition we have that \( x \) and \( y \) are Type-A-confusable, and by Lemma 5.13, we have \( |D_1(x) \cap D_1(y)| = 2 \) which also contradicts our assumption. Therefore \( d \) is nonempty.

Let \( \{z\} = D_1(x) \cap D_1(y) \). Note that the following intersection of 1-deletion balls are empty:

\[
D_1(ua) \circ dbw \cap D_1(ua) \circ ebw = \emptyset
\]
\[
ua \circ D_1(dbw) \cap uae \circ D_1(ebw) = \emptyset
\]
\[
D_1(uad) \circ bw \cap D_1(uae) \circ bwb = \emptyset,
\]
\[
ua \circ D_1(bw) \cap uae \circ D_1(bwb) = \emptyset
\]

Hence \( z \) can only be in \( D_1(ua) \circ dbw \cap uae \circ D_1(bwb) \) or \( uad \circ D_1(bw) \cap D_1(ua) \circ ebw \).

Without loss of generality, we assume that \( z \in D_1(ua) \circ dbw \cap uae \circ D_1(bwb) \). Matching positions implies that \( u \in D_1(ua), db = ae \) and \( w \in D_1(bwb) \). Furthermore it implies that \( z = uadbw \). Let \( d = \bar{a}v \) and \( e = rb \) for some subsequences \( v \) and \( r \). Since \( db = \bar{a}e \), we have \( \bar{a}vb = \bar{ar}b \), and hence \( v = r \).
Therefore we have shown that $x = uadbw = uaαωbw$ and $y = uαωbw = uαωbw$. 

\[\square\]

## B.2 Special Cases

Following Lemma B.2, we study the intersection size of $t$-deletion balls for two special cases. In the first case, we assume that the two sequences differ at exactly one coordinate. In the second case, we assume that the sequences are Type-B-confusable with $u$ and $w$ being empty strings.

In our proofs, we appeal to the following technical results on deletion balls.

**Lemma B.3** ([61, 66, 69]). Let $1 \leq t \leq n$ and $a \in \{0, 1\}$. Suppose that $u, v, x, y$ are binary sequences.

1. In addition to (5.1) and (5.2), we have that
   \[D_t(n) = D_t(n - 1) + D_{t-1}(n - 2),\]
   \[ν_t(n) = ν_t(n - 1) + ν_{t-1}(n - 2).\]

2. $D_t(n) \geq D_{t-i}(n - i)$ and $ν_t(n) \geq ν_{t-i}(n - i)$ for $i \leq t$.

3. $D_t(ax)^a = a \circ D_t(x)$ and $D_t(αx)^a = D_{t-1}(x)^a$.

4. $|D_t(x)^a| \leq D_t(|x| - 1)$.

5. Suppose further that $t < n/2$. Then $|D_t(x)| = D_t(n)$ if and only if $x$ is an alternating sequence.

**Proof.** (1) and (2) are from Levenshtein’s work [66], while (3) is derived in [61].

We prove (4) here. If $a$ does not appear in $x$, then the inequality is trivial. Now suppose that $x = α^m ax^*$, for some $m \geq 0$, and subsequence $x^*$. Then we have
\[|D_t(x)^a| = |D_t(α^m ax^*)^a| = |D_{t-m}(x^*)| \leq D_{t-m}(|x^*|) = D_{t-m}(|x| - m - 1) \leq D_t(|x| - 1),\]
where the last inequality follows from Lemma B.3(ii).

Next, we prove (5). When $x$ is alternating, it is straightforward to verify that $|D_t(x)| = D_t(n)$. To show the converse, we suppose that $x$ is not alternating. Then
Lemma B.3(i) states that $|\mathcal{D}_t(x)| \leq D_t(n - 2) + D_{t-1}(n - 2) + D_{t-2}(n - 4)$. Applying Lemma B.3(i), we have that $|\mathcal{D}_t(x)| < D_t(n)$ if $D_{t-2}(n - 4) < D_{t-1}(n - 3)$. Now, since the difference $D_{t-1}(n - 3) - D_{t-2}(n - 4) = \binom{n-t-2}{t-1}$, we have a strict inequality when $n - t - 2 \geq t - 1$, or, $t < n/2$.

We proceed to study the first special case where $x$ and $y$ have Hamming distance one. And the result will be a corollary of the following theorem.

**Theorem B.4.** Let $x = ucv$ and $y = udv$, where $u, c, d$ and $v$ are binary sequences. Then

\[
\mathcal{D}_t(x) \cap \mathcal{D}_s(y) = \bigcup_{a+b \leq \min(t, s)} \mathcal{D}_a(u) \circ (\mathcal{D}_{t-a-b}(c) \cap \mathcal{D}_{s-a-b}(d)) \circ \mathcal{D}_b(v).
\]

**Proof.** It is clear that $\mathcal{D}_t(x) \cap \mathcal{D}_s(y) \supseteq \bigcup_{a+b \leq \min(t, s)} \mathcal{D}_a(u) \circ (\mathcal{D}_{t-a-b}(c) \cap \mathcal{D}_{s-a-b}(d)) \circ \mathcal{D}_b(v)$. Now we want to show the other direction. Let $z \in \mathcal{D}_t(x) \cap \mathcal{D}_s(y)$. Suppose that $z = u'z'v'$, where $u'$ is the longest prefix of $z$ that is a subsequence of $u$, and $v'$ is the longest suffix of $z$ that is a subsequence of $v$. This implies that $z'$ must be a subsequence of $c$ and also a subsequence of $d$. Suppose that $|u| - |u'| = a$ and $|v| - |v'| = b$, then $z' \in \mathcal{D}_{t-a-b}(c) \cap \mathcal{D}_{s-a-b}(d)$. Thus $z \in \mathcal{D}_a(u) \circ (\mathcal{D}_{t-a-b}(c) \cap \mathcal{D}_{s-a-b}(d)) \circ \mathcal{D}_b(v)$. \hfill \Box

**Lemma B.5.** Let $x$ and $y$ be sequences with Hamming distance one. That is, $x = u|v$ and $y = u\emptyset v$ for subsequences $u$ and $v$. Then $\mathcal{D}_t(x) \cap \mathcal{D}_t(y) = \mathcal{D}_{t-1}(uv)$ for any $t \geq 1$.

**Proof.** Using Theorem C.4, we have

\[
\mathcal{D}_t(x) \cap \mathcal{D}_t(y) = \bigcup_{a+b \leq t} \mathcal{D}_a(u) \circ (\mathcal{D}_{t-a-b}(0) \cap \mathcal{D}_{t-a-b}(1)) \circ \mathcal{D}_b(v)
\]

\[
= \bigcup_{a+b = t-1} \mathcal{D}_a(u) \circ \mathcal{D}_b(v) = \mathcal{D}_{t-1}(uv).
\]

\hfill \Box

Next, we consider the case where the sequences are Type-B confusable with the subsequences $u$ and $w$ being empty. Before that we need to have several lemmas that will help us.
Lemma B.6. Suppose that $aav$ is a binary sequence of length $n \geq 3$, where $a \in \{0,1\}$ and some subsequence $v$. Then $|D_t(aav)| \leq D_t(n - 1) + D_{t-2}(n - 3)$.

Proof. $|D_t(aav)| = |D_t(aav)^a| + |D_t(aav)^\bar{a}| = |D_t(\bar{a}v) + |D_{t-2}(v)| \leq D_t(n - 1) + D_{t-2}(n - 3)$, where the last inequality comes from Lemma B.3.

Lemma B.7. Suppose $avb$ is binary sequence of length $n \geq 4$ that is not alternating, for $a, b \in \{0,1\}$ and some subsequence $v$. Then $|D_t(avb)| + |D_{t-1}(v)| \leq D_t(n - 1) + D_{t-1}(n - 1)$.

Proof. Let $avb = v_1v_2\cdots v_n$, where $v_1 = a, v_n = b$ and $v = v_2v_3\cdots v_{n-1}$. Since $avb$ is not an alternating binary sequence, let $i \geq 1$ be the smallest index such that $v_i = v_{i+1}$. We are going to show that the lemma is true by induction on $i$.

The base cases are when $i = 1$ and $i = 2$. For $i = 1$, it means that $v_1 = v_2 = a$, and therefore $|D_t(avb)| = |D_t(aav_3\cdots v_n)| \leq D_t(n - 1) + D_{t-2}(n - 3)$, where the last inequality comes from Lemma B.6. Hence $|D_t(avb)| + |D_{t-1}(v)| \leq D_t(n - 1) + D_{t-2}(n - 3) + D_{t-1}(n - 2) = D_t(n - 1) + D_{t-1}(n - 1)$, where the last equality comes from Lemma B.3.

For $i = 2$, it means that $v_2 = \bar{a}$ and $v_3 = \bar{a}$. This implies that $|D_t(abv)| = |D_t(aav_4\cdots v_n)| = |D_t(aav_4\cdots v_n)^a| + |D_t(aav_4\cdots v_n)^\bar{a}| = |D_t(aav_4\cdots v_n)| + |D_{t-1}(v_4\cdots v_{n-1})| \leq |D_t(aav_4\cdots v_n)| + |D_{t-1}(v_4\cdots v_{n-1})| \leq D_t(n - 2) + D_{t-2}(n - 4) + D_{t-1}(n - 2), where the last inequality comes from Lemma B.6. Note that $|D_{t-1}(v)| = |D_{t-1}(aav_4\cdots v_{n-1})| \leq D_{t-1}(n - 3) + D_{t-3}(n - 5)$, from Lemma B.6. Hence $|D_t(abv)| + |D_{t-1}(v)| \leq D_t(n - 2) + D_{t-2}(n - 4) + D_{t-1}(n - 2) + D_{t-1}(n - 3) + D_{t-3}(n - 5) = D_t(n - 1) + D_{t-1}(n - 1)$, where the last equality comes from several applications of Lemma B.3.

Suppose the statement is true for all $i \leq k$, where $k \geq 2$, we want to show that the statement is true for $i = k + 1$. Suppose that $x_{k+1} = x_{k+2}$, where $k + 1 \geq 3$, and $x_1 \cdots x_k$ is an alternating binary subsequence. By induction hypothesis, we have
B.2. Special Cases

that

$$|D_t(avb)| + |D_{t-1}(v)|$$

$$= |D_t(avb)^a| + |D_{t-1}(v)^a| + |D_t(avb)^b| + |D_{t-1}(v)^b|$$

$$= |D_t(v_2 \cdots v_n)| + |D_{t-1}(v_3 \cdots v_{n-1})|$$

$$+ |D_{t-1}(v_3 \cdots v_n)| + |D_{t-2}(v_4 \cdots v_{n-1})|$$

$$\leq D_t(n - 2) + D_{t-1}(n - 2) + D_{t-1}(n - 3) + D_{t-2}(n - 3)$$

$$= D_t(n - 1) + D_{t-1}(n - 1),$$

where the last inequality comes from Lemma B.3.

Next, we make the following observation on the sequence \( z \) that lies in the intersection of the single-deletion balls.

**Lemma B.8.** If \( x \) and \( y \) are Type-B-confusable and \( D_1(x) \cap D_1(y) = \{z\} \), then \( z \) is not alternating.

**Proof.** Suppose \( x = ua\overline{ab}vw \) and \( y = u\overline{ab}bw \), for some subsequences \( u, v \) and \( w \), where \( a, b \in \{0, 1\} \). Suppose that \( z = u\overline{ab}bw \) is an alternating sequence. This means \( \overline{ab} \) is an alternating subsequence, and hence \( a\overline{ab}b \) (and \( \overline{ab}bw \)), which is a subsequences of \( x \) (and \( y \), respectively) is an alternating sequence as well. This implies that \( x = uaw \) and \( y = u\overline{a}w \) are Type-A-confusable, where \( a = a\overline{ab}b \), and hence \( |D_1(x) \cap D_1(y)| = 2 \) by Lemma 5.13, which contradicts our assumption.

**Lemma B.9.** Let \( x \) and \( y \) be binary sequences of the form

\[ x = a\overline{ab}b \quad \text{and} \quad y = \overline{ab}b, \]

or vice versa, for some subsequence \( v \) of length \( n - 3 \), where \( n \geq 5 \) and \( a, b \in \{0, 1\} \).

If \( D_1(x) \cap D_1(y) = \{z\} \), then \( |D_t(x) \cap D_t(y)| \leq D_{t-1}(n - 3) + D_{t-2}(n - 3) + 2D_{t-2}(n - 5) \) for \( t \geq 2 \). Furthermore, for \( t = 2 \), we have that \( |(D_2(x) \cap D_2(y)) \setminus D_1(z)| \leq 2. \)

**Proof.** Let \( S = D_t(x) \cap D_t(y) \). Note that \( z = \overline{ab}b \). We split this into three cases.
Appendix B. Proof of Theorem 5.25 and Theorem 5.27

1. If \( v = \overline{v}v^* \), for some subsequence \( v^* \)

We consider the size of the following two disjoint subsets:

\[
|S^\overline{v}| \leq |\mathcal{D}_t(x)^\overline{v}| = |\mathcal{D}_t(a\overline{v}vb)^\overline{v}| = |\mathcal{D}_{t-1}(\overline{v}vb)^\overline{v}| \quad \text{(B.1)}
\]

\[
= |\mathcal{D}_{t-1}(\overline{v}v^*b)| \leq D_{t-1}(n-2),
\]

\[
|S^a| \leq |\mathcal{D}_t(y)^a| = |\mathcal{D}_t(\overline{a}\overline{v}v^*b\overline{b})^a|
\]

\[
= |\mathcal{D}_{t-2}(v^*b\overline{b})^a| \leq D_{t-2}(n-3), \quad \text{(B.2)}
\]

where the last inequality comes from Lemma B.3. Hence by several applications of Lemma B.3, we have

\[
|S| = |S^\overline{v}| + |S^a| \leq D_{t-1}(n-2) + D_{t-2}(n-3) = D_{t-1}(n-3) + D_{t-2}(n-4) + D_{t-2}(n-3) = D_{t-1}(n-3) + D_{t-2}(n-5) + D_{t-2}(n-6) + D_{t-2}(n-3) \leq D_{t-1}(n-3) + 2D_{t-2}(n-5) + D_{t-2}(n-3)
\]

as desired.

Furthermore, it can be seen from (B.1) and (B.2), that when \( t = 2 \), \( |S^\overline{v}| + |S^a| \leq |\mathcal{D}_1(z)| + D_0(n-3) \leq |\mathcal{D}_1(z)| + 1.\)

2. If \( v = v^*b \), for some subsequence \( v^* \)

We consider the size of the following two disjoint subsets:

\[
|S_b| \leq |\mathcal{D}_t(y)_b| = |\mathcal{D}_t(a\overline{v}vb\overline{b})_b| = |\mathcal{D}_{t-1}(\overline{v}vb)_b| \quad \text{(B.3)}
\]

\[
= |\mathcal{D}_{t-1}(\overline{v}v^*b)| \leq D_{t-1}(n-2),
\]

\[
|S_b| \leq |\mathcal{D}_t(x)_b| = |\mathcal{D}_t(\overline{a}\overline{v}v^*b\overline{b})_b|
\]

\[
= |\mathcal{D}_{t-2}(a\overline{v}v^*)_b| \leq D_{t-2}(n-3), \quad \text{(B.4)}
\]

where the last inequality comes from Lemma B.3. Hence similar to Case (i), we have

\[
|S| = |S^\overline{v}| + |S_b| \leq D_{t-1}(n-2) + D_{t-2}(n-3) \leq D_{t-1}(n-3) + 2D_{t-2}(n-5) + D_{t-2}(n-3)
\]

as desired.

Furthermore, it can be seen from (B.3) and (B.4), that when \( t = 2 \), \( |S^\overline{v}| + |S^a| \leq |\mathcal{D}_1(z)| + D_0(n-3) \leq |\mathcal{D}_1(z)| + 1.\)

3. If \( v = a\overline{v}v^*b \), for some subsequence \( v^* \)

Note that \( z = \overline{v}vb = \overline{a}v^*b\overline{b} \). We consider the size of the following two
disjoint subsets:

\[ S_a \subseteq D_t(x) = D_t(aavb) = D_{t-1}(avb) \]
\[ = \overline{a} \circ D_{t-1}(av^*bb), \]

\[ S_b^a \subseteq D_t(y)^a = D_t(\overline{a}v\overline{b}\overline{b})^a = D_{t-1}(\overline{a}vb)^a \]
\[ = D_{t-1}(\overline{a}av^*bb)^a = a \circ D_{t-2}(v^*b) \circ b. \] \hspace{1cm} (B.5)

By Lemma B.8, we know that \( z = \overline{a}av^*bb \) is not an alternating binary sequence. This implies that \( av^*bb \) is also not alternating, and hence together with Lemma B.7, we have

\[ |S^a| + |S_b^a| \leq D_{t-1}(n - 3) + D_{t-2}(n - 3). \] \hspace{1cm} (B.7)

Lastly, consider the size of the last disjoint subset:

\[ |S_b^a| = |D_t(aav^*bb) \cap D_t(\overline{a}av^*bb)| \]
\[ = |D_{t-1}(\overline{a}a)^{n-3}|. \]

Now we want to show that \( \overline{a}av^* \) and \( v^*bb \) are distinct binary sequences of length \( n - 3 \). If \( b = a \), it can be shown that \( \overline{a}av^* \) is equal to \( v^*bb \) if and only if \( v^* = (\overline{a})^m \) for \( m \geq 0 \), in which case \( x = a\overline{a}(\overline{a})^m\overline{a}a \) and \( y = \overline{a}a(\overline{a})^m\overline{a}a \) would be Type-A-confusable and by Lemma 5.13 contradicts our assumption. If \( b = \overline{a} \), it can be shown that \( \overline{a}av^* \) is equal to \( v^*bb \) if and only if \( v^* = (a\overline{a})^m \) for \( m \geq 0 \), in which case \( x = a\overline{a}(a\overline{a})^m\overline{a}a \) and \( y = \overline{a}a(a\overline{a})^m\overline{a}a \) would be Type-A-confusable, and by Lemma 5.13 contradicts our assumption. Therefore \( \overline{a}av^* \) and \( v^*bb \) are distinct binary sequences of length \( n - 3 \), and therefore

\[ |S_b^a| \leq \nu_{t-1}(n - 3) = 2D_{t-2}(n - 5). \] \hspace{1cm} (B.8)

Combining this with (B.7), we have \( |S| = |S^a| + |S_b^a| + |S_b^a| \leq D_{t-1}(n - 3) + D_{t-2}(n - 3) + 2D_{t-2}(n - 5) \), which gives the desired result.

Furthermore, it can be seen from (B.5) and (B.6), that when \( t = 2 \), \( S^a \cup S_b^a \subseteq D_1(z) \), and also from (B.8) that \( |S_b^a| \leq \nu_{t}(n - 3) = 2 \). Hence for this case, we have \( |S| \leq |D_1(z)| + 2 \).
B.3 Proof of Theorem 5.25

Before we prove the main result, we need the following claim that can be proven by identities in Lemma B.3.

Lemma B.10. For \( t \geq 2 \) and \( n \geq 6 \), we have the following

1. \( D_{t-1}(n-1) \leq D_{t-1}(n-3) + D_{t-2}(n-3) + 2D_{t-2}(n-5) \).

2. \( D_{t-2}(n-3) + D_{t-1}(n-4) + D_{t-2}(n-4) + 2D_{t-2}(n-6) \leq D_{t-1}(n-3) + D_{t-2}(n-3) + 2D_{t-2}(n-5) \).

Finally, we prove the main result of this section.

Proof of Theorem 5.25. From Lemma B.2, we know that if \( \mathcal{D}_1(x) \cap \mathcal{D}_1(y) = \{ z \} \), then there are two possibilities. First possibility is when \( x \) and \( y \) have Hamming distance one, which by Lemma B.5, implies that \( |\mathcal{D}_t(x) \cap \mathcal{D}_t(y)| = |\mathcal{D}_{t-1}(z)| \leq D_{t-1}(n-1) \leq D_{t-1}(n-3) + D_{t-2}(n-3) + 2D_{t-2}(n-5) \) from Lemma B.10.

Second possibility is when \( x \) and \( y \) are Type-B-confusable. Without loss of generality, let \( x = u \alpha \nu \nu \) and \( y = u \alpha \nu \nu \nu \). Let \( S = \mathcal{D}_t(x) \cap \mathcal{D}_t(y) \). Note that \( z = u \alpha \nu \nu \).

We are going to show the result by induction on \( n \). The base case is when \( u \) and \( w \) are empty. In this case, from Lemma B.9, we have the desired result. Now, suppose the statement is true for length \( n \leq k - 1 \), we want to show for \( n = k \). Now, we want to consider several cases for the prefix \( u \). Suppose \( u \) is a nonempty subsequence, i.e. \( u = cu^* \), for some subsequence \( u^* \) and \( c \in \{0,1\} \).
Consider the following,
\[
|S^c| = |\mathcal{D}_t(\mathcal{u}^a\mathcal{a}v\mathcal{b}w)^c \cap \mathcal{D}_t(\mathcal{u}^a\mathcal{b}v\mathcal{b}w)^c| \\
= |\mathcal{D}_{t-1}(\mathcal{u}^a\mathcal{a}v\mathcal{b}w)^c \cap \mathcal{D}_{t-1}(\mathcal{u}^a\mathcal{b}v\mathcal{b}w)^c|, 
\]
where the last inequality comes from our induction hypothesis. Now, consider the following cases

**Case 1:** If \( \bar{c} \) does not appear in \( \mathcal{u}^* \) and \( \bar{c} = \bar{a} \).

From (B.9), we have \( |S^c| \leq |\mathcal{D}_{t-1}(\mathcal{u}^a\mathcal{a}v\mathcal{b}w)^c| \leq |\mathcal{D}_{t-2-|\mathcal{a}|}(\mathcal{v}\mathcal{b}w)| \leq D_{t-2-|\mathcal{a}|}(n - |\mathcal{u}^*| - 3) \leq D_{t-2}(n - 3) \), where the last inequality follows from Lemma B.3. Combined with (B.10), we have \( |S| = |S^c| + |S^2| \leq D_{t-2}(n - 3) + D_{t-1}(n - 4) + D_{t-2}(n - 4) + 2D_{t-2}(n - 6) \leq D_{t-1}(n - 3) + D_{t-2}(n - 3) + 2D_{t-2}(n - 5) \), where the last inequality follows from Lemma B.10.

**Case 2:** If \( \bar{c} \) does not appear in \( \mathcal{u}^* \) and \( \bar{c} = \bar{a} \).

From (B.9), we have \( |S^c| \leq |\mathcal{D}_{t-1}(\mathcal{u}^a\mathcal{b}v\mathcal{b}w)^c| \leq |\mathcal{D}_{t-2-|\mathcal{a}|}(\mathcal{v}\mathcal{b}w)|^{\prime} \leq D_{t-2-|\mathcal{a}|}(|\mathcal{v}\mathcal{b}w| - 1) \leq D_{t-2-|\mathcal{a}|}(n - |\mathcal{u}^*| - 3) \leq D_{t-2}(n - 3) \), where the third and last inequalities come from Lemma B.3. Thus exactly the same as Case 1, combining this with (B.10), we have the desired result.

**Case 3:** If \( \bar{c} \) appears in \( \mathcal{u}^* \) i.e. \( \mathcal{u}^* = c^m\mathcal{u}' \), for some subsequence \( \mathcal{u}' \) and \( m \geq 0 \).

From (B.9), we have \( |S^2| = |\mathcal{D}_{t-m-1}(\mathcal{u}'a\mathcal{a}v\mathcal{b}w) \cap \mathcal{D}_{t-m-1}(\mathcal{u}'a\mathcal{b}v\mathcal{b}w)| \leq D_{t-m-2}(n - m - 5) + D_{t-m-3}(n - m - 5) + 2D_{t-m-3}(n - m - 7) \leq D_{t-2}(n - 5) + D_{t-3}(n - 5) + 2D_{t-3}(n - 7) \), where the first inequality holds because \( \mathcal{u}'a\mathcal{a}v\mathcal{b}w \) and \( \mathcal{u}'a\mathcal{b}v\mathcal{b}w \) are *Type-B confusables* and hence we can use our induction hypothesis, and the last inequality follows from Lemma B.3. Combined with (B.10), we have \( |S| = |S^c| + |S^2| \leq D_{t-1}(n - 4) + D_{t-2}(n - 4) + 2D_{t-2}(n - 6) + D_{t-2}(n - 5) + D_{t-3}(n - 5) + 2D_{t-3}(n - 7) = D_{t-1}(n - 3) + D_{t-2}(n - 3) + 2D_{t-2}(n - 5) \), where the last equality follows from Lemma B.3.

In all cases, we have shown that the statement is true. Now suppose \( \mathcal{w} \) is a nonempty subsequence, i.e. \( \mathcal{w} = \mathcal{w}^*c \), for some binary sequence \( \mathcal{w}^* \) and \( c \in \{0, 1\} \),
then similarly to the above, by considering \(|S_{c}|\) and \(|S_{r}|\), we can also show that the statement is true.

Therefore we are left with the case when \(u\) and \(w\) are both empty strings, which is already covered in the base case. Hence we have shown that the inequality (5.5) is true for all \(n \geq 7\) and \(t \geq 2\).

To show (5.6), we combine (5.5) with Lemma B.3; it is easy to see that \(|D_t(x) \cap D_t(y)| \leq D_{t-1}(n-3) + D_{t-2}(n-3) + 2D_{t-2}(n-5) = D_{t-1}(n-3) + D_{t-2}(n-3) + \nu_{t-1}(n-3) = D_{t-1}(n-2) - D_{t-2}(n-4) + D_{t-2}(n-3) + \nu_{t-1}(n-3) \leq D_{t-1}(n-1) + \nu_{t-1}(n-3) = n^{t-1} + O(n^{t-2})\) for fixed values of \(t\).

The inequality (5.5) is tight by considering the following examples. For even length \(n\), consider \(x = 1010(10)^m01\) and \(y = 010(10)^m010\), where \(m \geq 1\). Note that \(D_1(x) \cap D_1(y) = \{z\} = \{010(10)^m01\}\). Let \(S = D_t(x) \cap D_t(y)\).

Consider the following subsets:

\[
|S^0| = |D_{t-1}(10(10)^m01) \cap D_t(10(10)^m010)|
= |D_{t-1}(10(10)^m01)|
= |D_{t-1}((10)^m1001)_0| + |D_{t-1}((10)^m1001)_1|
= |D_{t-2}((10)^m10)| + |D_{t-1}((10)^m100)|
= D_{t-2}(n-4) + |D_{t-1}((10)^m100)_0| + |D_{t-1}((10)^m100)_1|
= D_{t-2}(n-4) + |D_{t-1}((10)^m10)| + |D_{t-2}(10^m)|
= D_{t-2}(n-4) + D_{t-1}(n-4) + D_{t-3}(n-6).
\]

\[
|S^1| = |D_t(010(10)^m0) \cap D_{t-2}(0(10)^m0)|
= |D_{t-2}(0(10)^m0)|
= |D_{t-2}(0(10)^m0)_0| + |D_{t-2}(0(10)^m0)_1|
= |D_{t-2}(0(10)^m)| + |D_{t-4}(0(10)^{m-1})|
= D_{t-2}(n-5) + D_{t-4}(n-7).
\]
\[ |S_0| = |D_{t-1}(010(10)^m) \cap D_{t-1}(0(10)^m01)| \]
\[ = |D_{t-1}(010(10)^m) \cap D_{t-1}(0(10)^m01)| \]
\[ = |D_{t-1}(010(10)^m0) \cap D_{t-1}(0(10)^m01)0| \]
\[ + |D_{t-1}(010(10)^m1) \cap D_{t-1}(0(10)^m01)1| \]
\[ = |D_{t-1}(010(10)^m1) \cap D_{t-2}(0(10)^m)| \]
\[ + |D_{t-2}(010(10)^m) \cap D_{t-1}(0(10)^m0)| \]
\[ = |D_{t-2}(010(10)^m)| + |D_{t-2}(0(10)^m)| \]
\[ = 2D_{t-2}(n - 5). \]

Hence \( |S| = D_{t-2}(n - 4) + D_{t-1}(n - 4) + D_{t-3}(n - 6) + D_{t-2}(n - 5) + D_{t-4}(n - 7) + 2D_{t-2}(n - 5) = D_{t-1}(n - 3) + D_{t-2}(n - 3) + 2D_{t-2}(n - 5). \) And similarly for odd length \( n \), it can be shown that the equality holds when \( x = 101(01)^m10 \) and \( y = 01(01)^m101 \), for \( m \geq 1 \).

\[ \square \]

**B.4 Proof of Theorem 5.27**

*Proof.* We are going to show the result by induction on \( n \). The base case is when \( u \) and \( w \) are empty subwords, which is obtained from Lemma B.9. Suppose the statement is true for length \( n \leq k - 1 \), we want to show for \( n = k \). Now, we want to consider several cases for the prefix \( u \). Suppose \( u \) is a nonempty prefix, i.e. \( u = cu^* \), for some subword \( u^* \) and \( c \in \{0, 1\} \).

Consider the following

\[ S^\pi = D_2(cu^*a\overline{a}bvw)^\pi \cap D_2(cu^*a\overline{a}b\overline{a}w)^\pi \]
\[ = D_1(u^*a\overline{a}bvw)^\pi \cap D_1(u^*a\overline{a}b\overline{a}w)^\pi \]
\[ = D_0(u^*a\overline{a}bvw)^\pi = D_1(cu^*a\overline{a}bvw)^\pi = D_1(z)^\pi, \tag{B.11} \]
where the third equality holds because $|D_1(x) \cap D_1(y)| = 1$, and

$$ |S \setminus D_1(z)^c| = |D_2(cu^*a\alpha\omega b\omega)^c \setminus D_2(cu^*\alpha\omega b\omega)^c \setminus D_1(z)^c| \\
= |(c \circ D_2(u^*a\alpha\omega b\omega) \setminus D_2(u^*\alpha\omega b\omega)) \setminus D_1(z)^c| \\
= |c \circ ((D_2(u^*a\alpha\omega b\omega) \setminus D_2(u^*\alpha\omega b\omega)) \setminus D_1(u^*\alpha\omega b\omega))| \\
\leq 2 $$

(B.12)

where the inequality follows from our induction hypothesis. Combining equations (B.11) and (B.12), we have that $|S \setminus D_1(z)| = |S \setminus D_1(z)^c| + |S \setminus D_1(z)^c| \leq 2$. Hence, our induction on $n$ is complete.

Therefore $|D_2(x) \cap D_2(y)| \leq 2 + D_1(z) = 2 + n - 2 = n$, where the second last equality comes from Theorem B.8. Therefore if $C$ is an $(n, 2; D_1)$-reconstruction code, then $C$ is also an $(n, n + 1; D_2)$-reconstruction code. Thus, $\rho(n, n + 1; D_2) \leq \log_2 \log_2 n + O(1)$. \qed
Appendix C

Proof of Theorem 5.32

To demonstrate Theorem 5.32, we follow the general strategy in [60]. Specifically, in Section C.1, we first analyze the necessary conditions for which \( x \) and \( y \) are confusable, and define the following notion of confusability.

**Definition C.1.** Two binary sequences \( x \) and \( y \) are said to be Type-C confusable if \(|D_2(x) \cap D_2(y)| \geq 5\) and \(|D_1(x) \cap D_1(y)| = \emptyset\).

This analysis then culminates in our characterization theorem, Theorem C.2, where we provide the explicit forms of \( x \) and \( y \) when they are Type-C confusable. Then in Section C.2, we show that whenever \( x \) and \( y \) satisfy the characterization, they cannot meet the design conditions given in Construction C. This therefore contradicts the assumption that \( x \) and \( y \) are Type-C confusable, proving the reconstruction capabilities of the code \(\mathcal{C}(n, P; c_1, c_2)\).

C.1 Characteristics of Type-C confusable Sequences

Unless otherwise stated, **boldface** symbols represent binary sequences while letters in regular font represent indices or bit values. Let \( x \) be a binary sequence, we denote the complement of \( x \) by \( \bar{x} \). Also, recall for brevity, we say that two sequences \( x \) and \( y \) are Type-C confusable if \(|D_2(x) \cap D_2(y)| \geq 5\) and \(D_1(x) \cap D_1(y) = \emptyset\). In this section, we demonstrate the following theorem.

\(^1\)Observe that Condition 1 in Construction C ensures that \(D_1(x) \cap D_1(y) = \emptyset\).
Theorem C.2 (Characterization Theorem). If \( x \) and \( y \) are Type-C confusable, then they have to be in one of the following forms.

1. \( x = u\alpha\omega\beta v \) and \( y = u\alpha\omega\beta v \), where \( \alpha \) and \( \beta \) are alternating sequences of length at least two.

2. \( x = u\alpha\gamma\beta v \) and \( y = u\alpha\gamma\beta v \), where \( \alpha \), \( \beta \) and \( \gamma \) are alternating sequences and \( a, b \in \{0, 1\} \). Here, \( \alpha \) is of length at least two and ends with \( a \), \( \beta \) is of length at least two and starts with \( b \), and \( \gamma \) starts with \( \alpha \) and ends with \( \beta \).

Our proof of this characterization theorem is fairly technical. We first analyze the properties of intersection of two-deletion balls in Section C.1.1, and subsequently, provide the proof of this characterization in Section C.1.2.

C.1.1 Intersection of Two-Deletion Balls

Let \( S \) be a set of binary sequences. Following [61], we partition \( S \) according to its first or last (or both) bits. Specifically, for \( a, b \in \{0, 1\} \), we define \( S^a, S \), and \( S^b \) as follows.

\[
S^a \triangleq \{ \text{sequences in } S \text{ starting with } a \},
\]
\[
S_b \triangleq \{ \text{sequences in } S \text{ ending with } b \},
\]
\[
S^a_b \triangleq \{ \text{sequences in } S \text{ starting with } a \text{ and ending with } b \}.
\]

Let \( T \) be another set of binary sequences, we use \( S \circ T \) to denote the set of sequences obtained by concatenating sequences in \( S \) with sequences in \( T \). When \( T = \{a\} \), we simply write \( a \circ S \) or \( S \circ a \). The following lemma was demonstrated in [61] and is used throughout the thesis.

Lemma C.3 ([61]). Let \( 1 \leq t \leq n \) and \( a \in \{0, 1\} \). Then \( \mathcal{D}_t(ax)^a = a \circ \mathcal{D}_t(x) \) and \( \mathcal{D}_t(ax)^b = \mathcal{D}_{t-1}(x)^a \).

Let \( x \) and \( y \) be binary sequences. Let \( u \) and \( v \) be a common prefix and suffix, respectively, of \( x \) and \( y \). In other words, \( x = ucv \) and \( y = udv \). In the subsequent results, we show that to study the intersection of \( t \)-deletion balls of \( x \) and \( y \), we can look at the \( t \)-deletion balls of \( c \) and \( d \) under some conditions.
Theorem C.4. Suppose that $x = ucv$ and $y = udv$. Then the intersection of the $s$- and $t$-deletion balls $D_t(x) \cap D_s(y)$ is

$$
\bigcup_{a+b \leq \min(t,s)} D_a(u) \circ (D_{t-a-b}(c) \cap D_{s-a-b}(d)) \circ D_b(v).
$$

Proof. It is clear that $D_t(x) \cap D_s(y)$ contains

$$
\bigcup_{a+b \leq \min(t,s)} D_a(u) \circ (D_{t-a-b}(c) \cap D_{s-a-b}(d)) \circ D_b(v).
$$

Now we want to show the other direction. Let $z \in D_t(x) \cap D_s(y)$. Suppose that $z = u'z'v'$, where $u'$ is the longest prefix of $z$ that is a subsequence of $u$, and $v'$ is the longest suffix of $z$ that is a subsequence of $v$. This implies that $z'$ must be a subsequence of $c$ and also a subsequence of $d$. Suppose that $|u| - |u'| = a$ and $|v| - |v'| = b$, then $z' \in D_{t-a-b}(c) \cap D_{s-a-b}(d)$. Thus $z \in D_a(u) \circ (D_{t-a-b}(c) \cap D_{s-a-b}(d)) \circ D_b(v)$. □

Lemma C.5. Let $x = ucv$ and $y = udv$ with $|x| \geq t$. If $D_{t-1}(x) \cap D_{t-1}(y) = \emptyset$, then $D_t(x) \cap D_t(y) = u \circ (D_t(c) \cap D_t(d)) \circ v$.

Proof. Suppose that $|c| \leq t - 1$. Then we can delete $c$ from $x$ and $d$ from $y$, and hence $D_{t-1}(x) \cap D_{t-1}(y)$ is non-empty, which is a contradiction. Therefore $|c| \geq t$.

Since $D_{t-1}(x) \cap D_{t-1}(y)$ is empty, it implies that $D_{t-1}(c) \cap D_{t-1}(d)$ is also empty. Hence, we have $D_s(c) \cap D_s(d)$ is empty for all $s \leq t - 1$. Applying Theorem C.4, we know that $D_t(x) \cap D_t(y) = \bigcup_{a+b \leq t} D_a(u) \circ (D_{t-a-b}(c) \cap D_{t-a-b}(d)) \circ D_b(v) = D_0(u) \circ (D_t(c) \cap D_t(d)) \circ D_0(v) = u \circ (D_t(c) \cap D_t(d)) \circ v$. □

Suppose that $x$ and $y$ are binary sequences of the same length with Hamming distance at least two, then we can write $x = uirjv$ and $y = uisjv$. By setting $t = 2$ to Lemma C.5, we have the corollary.

Corollary C.6. Suppose that $x = uirjv$ and $y = uisjv$. If $D_1(x) \cap D_1(y) = \emptyset$, then $D_2(x) \cap D_2(y) = u \circ (D_2(irj) \cap D_2(isj)) \circ v$.

Next, we assume that $D_1(x) \cap D_1(y) = \emptyset$, which implies that $x$ and $y$ must have Hamming distance at least two. Recall our objective is to analyze $D_2(x) \cap D_2(y)$. In light of Corollary C.6, it suffices to consider the case when $x = irj$ and $y = isj$. 


Let $S \triangleq \mathcal{D}_2(x) \cap \mathcal{D}_2(y)$. In [61], the authors showed that $|S| \leq 6$. Here, for exposition purposes, we provide a proof and additional observations on the size of $S$. We consider the disjoint sets:

\[
\begin{align*}
S'_j &= (i \circ \mathcal{D}_2(r) \circ j) \cap \mathcal{D}_0(s)'_j, \\
S''_j &= (i \circ \mathcal{D}_2(r) \circ j) \cap (\mathcal{D}_1(s)' \circ j), \\
S''''_j &= (\mathcal{D}_1(r) \circ j) \cap (i \circ \mathcal{D}_1(s)'_j), \\
S''''''_j &= \mathcal{D}_0((r_s)_j) \cap (i \circ \mathcal{D}_2(s) \circ j). 
\end{align*}
\]

**Lemma C.7.** Suppose that $x = irj$ and $y = is'j$. If we set $S = \mathcal{D}_2(x) \cap \mathcal{D}_2(y)$, then the following holds.

1. $|S'_j| \leq 1$, $|S''_j| \leq 2$, $|S''''_j| \leq 2$, and $|S''''''_j| \leq 1$.

2. If the first and second bit of $x$ are equal, i.e. $r = ir'$ then $|S| \leq 4$. Hence by symmetry, if $r = ir'$ or $r = r'j$ or $s = is'$ or $s = s'j$ for some subsequences $r'$ and $s'$, then $|S| \leq 4$.

3. If $|S| \geq 5$, then $x = i\bar{r}j$ and $y = is'j$, for some subsequences $r'$ and $s'$.

**Proof.** 1. Note for any $x$ and $y$, we have $|S'_j| \leq |\mathcal{D}_0(s)'_j| \leq 1$. Likewise, we also have $|S''_j| \leq |\mathcal{D}_0(r)''_j| \leq 1$. Now we show that $|S''''_j| \leq 2$ by considering the following cases. If $r = ir'$ or $s = s'j$, then $|S''''_j| \leq |\mathcal{D}_1(r)''| = |\mathcal{D}_0(r')''| \leq 1$ or $|S''''_j| \leq |\mathcal{D}_1(s)'_j| = |\mathcal{D}_0(s'')_j| \leq 1$, respectively. Otherwise, suppose that $r = i\bar{r}'$ and $s = s'j$, i.e. $x = i\bar{r}'j$ and $y = is'j$. This implies that $r' \neq s'$, since otherwise $i\bar{r}'j \in \mathcal{D}_1(x) \cap \mathcal{D}_1(y)$, which contradicts our assumption that their intersection is empty. Therefore, $|S''''_j| = |\mathcal{D}_1(r') \cap \mathcal{D}_1(s')| \leq 2$. Similarly, we can show that $|S''''''_j| \leq 2$.

2. Without loss of generality, suppose that $r = ir'$. Then $|S''''''_j| \leq |\mathcal{D}_1(r)''| = |\mathcal{D}_0(r')''| \leq 1$. Furthermore, $|S''''''_j| \leq |\mathcal{D}_0(r)''_j| = |\mathcal{D}_0(i\bar{r})''_j| = 0$. Hence, we know that $|S| \leq 4$.

3. This is a direct corollary of part (ii). \qed

**C.1.2 Proof of Theorem C.2**

For convenience, we introduce notations for certain alternating sequences. Specifically, $a'_j(n)$ denotes the alternating sequence of length $n$ that starts with $i$ and
ends with $j$. For example $a^1_i(5) = 10101$. Note that $i = j$ if and only if $n$ is odd integer. Observe too that $a^i(n) = a^j(n)$. If $i$, $j$ or $n$ do not matter or is clear from the context, then we omit the symbol and just use $a_j(n)$, $a^i(n)$, or $a^j_i$, respectively. So, for example, $a^0_0(4) = 1010 = a^0_0(4)$. Here, we use $a^i$ to denote an alternating binary sequence of any length that ends with $i$ and $a(n)$ to denote any alternating binary sequence of length $n$. Using this notation, we restate Theorem C.2 in a more convenient form for our proof.

**Theorem C.8** (Restatement of Theorem C.2). If $x$ and $y$ are Type-C confusable, then $x$ and $y$ have to be one of the following forms.

1. $x = ua_{(m_1)}w a_{(m_2)}v$ and $y = ua_{(m_1)}wa_{(m_2)}v$ with $m_1, m_2 \geq 2$.

2. $x = ua_{(m_1)}a_{(m_2)}a_{(m_3)}b v$ and $y = ua_{(m_1)}a_{(m_2+2)}a_{(m_3)}v$, with $m_1, m_3 \geq 2$.

The figure below is an example of the Form 1 from Theorem C.2.

![Figure C.1: Example of two Type-C confusable sequences of the Form 1 in Theorem C.8](image)

To prove Theorem C.8, we need several lemmas to aid our analysis. As before, we assume that $D_1(x) \cap D_1(y) = \emptyset$ and $x = irj$ and $y = jsj$. Set $S = D_2(x) \cap D_2(y)$ and we consider the case $|S| \geq 5$. Following Lemma C.7(iii), we have that both $x$ and $y$ have alternating prefixes and suffixes of lengths at least two. Formally, let $a^i(m_1)$ and $a^j(m_2)$ be the longest alternating sequences such that $x$ and $y$ can be written as $a^i(m_1)ra^j(m_2)$ and $a^i(m_1)sa^j(m_2)$, respectively. From Lemma C.7(iii), we know that $m_1, m_2 \geq 2$. Suppose that $a^i(m_1)$ ends with $a$ and $a^j(m_2)$ starts with $b$. And so we replace $a^i(m_1)$ with $a^i_a(m_1)$ and $a^j(m_2)$ with $a^j_b(m_2)$. Now, we write $|S_j|$, $|S_j|$, $|S_j|$ and $|S_j|$ in terms of deletion balls involving $a$, $b$, $r$ and $s$. 
Lemma C.9. Let $x = a_i^h(m_1)ra_j^b(m_2)$ and $y = \overline{a_i^h(m_1)sa_j^b(m_2)}$. If $S = D_2(x) \cap D_2(y)$, then we have

$$
|S_j^x| = |D_2(ar) \cap D_0(s)|, \quad |S_j^y| = |D_1(ar) \cap D_1(s\overline{b})|,
$$

$$
|S_i^x| = |D_1(rb) \cap D_1(\overline{as})|, \quad |S_i^y| = |D_0(r) \cap D_2(\overline{as\overline{b}})|.
$$

Proof. Note that

$$
|S_j^y| = |D_2(a_i^h(m_1)ra_j^b(m_2)) \cap D_2(\overline{a_i^h(m_1)sa_j^b(m_2)})| = |D_2(a_i^h(m_1 - 1)ra_j^b(m_2 - 1)) \cap D_0(a_i^h(m_1 - 2)sa_j^b(m_2 - 2))| = |D_2(ar) \cap D_0(s)|,
$$

where the last equality comes from Theorem C.4. The other three equations can be derived similarly.

Next, we obtain more conditions where $|S|$ is at most four.

Lemma C.10. Let $x = a_i^h(m_1)ra_j^b(m_2)$ and $y = \overline{a_i^h(m_1)sa_j^b(m_2)}$. If $S = D_2(x) \cap D_2(y)$, then the following are true.

1. If $r = s$ are empty sequences, then $|S| \leq 4$.
2. If $r$ starts with $a$ and $s$ starts with $\overline{a}$, then $|S| \leq 4$.
3. If $r$ ends with $b$ and $s$ ends with $\overline{b}$, then $|S| \leq 4$.
4. If $|r| = |s| = 1$, then $|S| \leq 4$.

Proof. Let $e$ denote the empty sequence.

1. If $r = s = e$, then by Lemma C.9, we have that $|S_j^x| = |D_1(a) \cap D_1(\overline{b})| = |\{e\}| = 1$, and $|S_j^y| = |D_1(b) \cap D_1(\overline{a})| = |\{e\}| = 1$. Furthermore from Lemma C.7, we know that $|S_j^x| \leq 1$ and $|S_j^y| \leq 1$. These imply that $|S| \leq 4$. 

2. Let \( r = ar' \) and \( s = \overline{a}s' \). By Lemma C.9, we have that
\[
|S_j^i| = |D_1(aar') \cap D_1(\overline{a}s'b)|
\]
\[
= |D_1(aar')^a \cap D_1(\overline{a}s'b)^a| + |D_1(aar')^b \cap D_1(\overline{a}s'b)^b|
\]
\[
= |D_1(aar')^a \cap D_0(s'b)^a| + |D_0(ar')^b \cap D_1(\overline{a}s'b)^b|
\]
\[
\leq |D_0(s'b)^a| + |D_0(ar')^b| \leq 1 + 0 = 1.
\]

Similarly, we have that \( |S_j^\overline{a}| \leq 1 \). Again, from Lemma C.7, we have that \( |S_j^1| \leq 1 \) and \( |S_j^2| \leq 1 \), and so, \( |S| \leq 4 \).

3. The proof is analogous to the proof of part (ii).

4. Suppose that \( |r| = |s| = 1 \). If \( r = a \) and \( s = \overline{a} \) then from part (ii), we know that \( |S| \leq 4 \). If \( r = \overline{a} \) and \( s = a \) then it contradicts the assumption that \( a^i_1(m_1) \) and \( b_j^2(m_2) \) are the longest alternating sequences such that \( x \) and \( y \) can be written as \( a^i_1(m_1)rb_j^2(m_2) \) and \( a^i_1(m_1)sb_j^2(m_2) \), respectively. Hence the only possibilities are \( r = s = a \) or \( r = s = \overline{a} \).

Note that if \( ar = sb \), then \( a^i_1(m_1 - 1)sa^i_2(m_2) = a^i_1(m_1)ra^i_2(m_2 - 1) \in D_1(x) \cap D_1(y) \), which contradicts \( D_1(x) \cap D_1(y) \) is non-empty. For the case when \( r = s = a \), this means that \( a = b \), since otherwise \( ar = sb \). Thus by Lemma C.9, \( |S_j^1| = |D_1(aa) \cap D_1(a\overline{a})| = 1 \) and \( |S_j^\overline{a}| = |D_1(aa) \cap D_1(\overline{a}a)| = 1 \). Furthermore with Lemma C.7, we have \( |S_j^1| \leq 1 \) and \( |S_j^2| \leq 1 \). Therefore \( |S| \leq 4 \).

Similarly note that if \( rb = \overline{a}s \) then \( a^i_2(m_1 - 1)ra^i_2(m_2) = a^i_2(m_1)sa^i_2(m_2 - 1) \in D_1(x) \cap D_1(y) \), which contradicts \( D_1(x) \cap D_1(y) \) is non-empty. For the case when \( r = s = \overline{a} \), this means that \( a = b \), since otherwise \( rb = \overline{a}s \). Thus by Lemma C.9, \( |S_j^\overline{a}| = |D_1(a\overline{a}) \cap D_1(\overline{a}a)| = 1 \) and \( |S_j^1| = |D_1(\overline{a}a) \cap D_1(\overline{a}a)| = 1 \). Applying with Lemma C.7, we have \( |S_j^1| \leq 1 \), \( |S_j^2| \leq 1 \) and so, \( |S| \leq 4 \). \( \square \)

Now, observe in Lemma C.10, the sets \( S_j^1 \) and \( S_j^2 \) are the intersections of certain single-deletion balls. The following characterization lemma for single-deletion balls was proven in [60].

**Lemma C.11 ([60]).** If \( |D_1(x) \cap D_1(y)| = 2 \), then \( x \) and \( y \) can be rewritten as \( x = ucv \) and \( y = uc\overline{v} \) where \( c \) is an alternating sequence of length at least two.

Now we are ready to prove Theorem C.8 – our main characterization theorem.
Proof of Theorem C.8. Suppose that $\mathcal{D}_1(x) \cap \mathcal{D}_1(y) = \emptyset$ and $|\mathcal{D}_2(x) \cap \mathcal{D}_2(y)| \geq 5$. Let $a^*_a(m_1)$ and $a^*_b(m_2)$ to be the longest alternating sequences such that $x$ and $y$ can be written as $a^*_a(m_1)ra^*_b(m_2)$ and $a^*_a(m_1)sas^*_b(m_2)$, respectively. Furthermore, from Lemma C.7(iii), we know that $m_1, m_2 \geq 2$.

Now, if $r = s$, we have that $x$ and $y$ are in form (A) and we are done. Hence, we assume that $r \neq s$. From Lemma C.10, we know that $r$ and $s$ have to start with the same bit, end with the same bit, and $|r| = |s| \geq 2$ in order for $|S| \geq 5$. Similar to the proof of Lemma C.10(iv), we also know that $ar \neq \overline{s}b$ and $rb \neq \overline{as}$. To analyze $|S|$, we have four cases.

1. Suppose that $r = \overline{ac\overline{b}}$ and $s = \overline{ad\overline{b}}$ with $c \neq d$.

By Lemma C.9, we have that $|S^*_j| = |\mathcal{D}_2(a\overline{ac\overline{b}}b) \cap \mathcal{D}_0(\overline{ad\overline{b}})| = 0$, and therefore, $|S^*_j| = |\mathcal{D}_1(ar) \cap \mathcal{D}_1(sb)| = 2$ and $|S^*_j| = |\mathcal{D}_1(rb) \cap \mathcal{D}_1(\overline{as})| = 2$. Applying Lemma C.11 to the pair $ar = \overline{ac\overline{b}}$ and $s\overline{b} = \overline{ad\overline{b}}$, we see that both sequences have complementary alternating prefixes and so, $d$ must start with $a$. Similarly, by considering the other pair $rb = \overline{ac\overline{b}}$ and $\overline{as} = \overline{ad\overline{b}}$, we have that $d$ must end with $b$. Therefore, we write $d = ad'b$. We then have two subcases.

- If $c$ starts with $a$, then applying Lemma C.11 to $rb = \overline{ac\overline{b}}$ and $\overline{as} = \overline{ad\overline{b}}$ implies that $c\overline{b}b$ is alternating with $\overline{c\overline{b}} = \overline{ad\overline{b}}$. Therefore, $c = d$, a contradiction.

- If $c$ starts with $\overline{a}$, then applying Lemma C.11 to $ar = \overline{a\overline{ac\overline{b}}b}$ and $s\overline{b} = \overline{ad\overline{b}}b\overline{b}$ implies that $c = d'\overline{b}$. Furthermore, applying Lemma C.11 to $rb = \overline{a\overline{c\overline{b}}b} = \overline{ad'\overline{b}b}b$ and $\overline{as} = \overline{ad\overline{b}}b\overline{b}$ implies that $d'\overline{b}b = \overline{ad'} = c$. This can only happen if $c = a^*_a$ is alternating and hence $d = \overline{c}$. Thus, $x$ and $y$ are in form (B).

2. When $r = ac\overline{b}$ and $s = ad\overline{b}$ with $c \neq d$, the proof is analogous to case (i).

3. Suppose that $r = ac\overline{b}$ and $s = ad\overline{b}$ with $c \neq d$.

Since $|S| \geq 5$, we have that either the case $|S^*_j| = 2$, or the case $|S^*_j| = 1$ and $|S^*_j| = 2$.

- Suppose that $|S^*_j| = 2$. Then applying Lemma C.11, we have that $rb = ac\overline{b}b$ and $\overline{as} = \overline{ad\overline{b}}$ are both alternating and complementary. Hence $r = s = a^*_a$, a contradiction.
• Suppose that \(|S_j| = 1\). Then the above argument shows that \(cbb = ad\).
  Again, we apply Lemma C.11 to the pair \(ar = aacb\) and \(sb = adb\).
  Proceeding as before, we have \(r = s\), a contradiction.

4. When \(r = acb\) and \(s = adb\) with \(c \neq d\), the proof is analogous to case (iii). \(\square\)

### C.2 Proof of Theorem 5.32

Let \(\mathcal{C} = \mathcal{C}(n, P; c_1, c_2)\) as defined in Construction C. In this section, we show that \(\nu(\mathcal{C}, D_2) = 4\). In other words, for the two-deletion channel and any sequence \(x \in \mathcal{C}\), we are able to uniquely reconstruct \(x\) from any five distinct noisy reads.

To establish Theorem 5.32, we frequently apply the following lemma. The proofs follow from straightforward algebraic manipulations and are hence omitted.

**Lemma C.12.** Suppose that \(x = ua^0(n)v\) and \(y = ua^1(n)v\). Let \(i\) denote the starting index of \(a^0(n)\), i.e. \(i = |u| + 1\). Then we have the following.

1. If \(n = 2t\), then \(VT^{(1)}(x) - VT^{(1)}(y) = t\).
2. If \(n = 2t + 1\), then \(VT^{(1)}(x) - VT^{(1)}(y) = -i - t\).
3. If \(n = 2t\), then \(VT^{(2)}(x) - VT^{(2)}(y) = t^2 + ti\).
4. If \(n = 2t + 1\), then \(VT^{(2)}(x) - VT^{(2)}(y) = t^2 + ti - \left(\frac{i + 2t + 1}{2}\right)\).

We now proceed with the proof of Theorem 5.32.

Let \(x\) and \(y\) be two distinct sequences of \(\mathcal{C}\). From Condition (C1), we have that \(\mathcal{C}\) is a single-deletion correcting code and so, \(D_1(x) \cap D_1(y) = \emptyset\). Suppose that \(|D_2(x) \cap D_2(y)| \geq 5\). Then Theorem C.8 states that \(x\) and \(y\) must be one of the two forms listed.

**Form (A):** Suppose that \(x = ua(m_1)wa(m_2)v\) and \(y = u\overline{a(m_1)}\overline{wa(m_2)}v\) with \(m_1, m_2 \geq 2\). Let \(i\) and \(j\) be the starting indices of \(a(m_1)\) and \(a(m_2)\), respectively. So, \(i = |u| + 1 = i\) and \(j = |ua(m_1)w| + 1 = j > i\). Without loss of generality, we assume that \(a(m_1)\) starts with a zero, and so, \(a(m_1) = a^0(m_1)\). Let \(a(m_2)\) start with the bit \(a\) and so, \(a(m_2) = a^a(m_2)\).
Hence, we need to consider eight cases based on the value of $a$, the parity of $m_1$, and the parity of $m_2$ as follows. Note that all the computations of $V^{(r)}(x) - V^{(r)}(y)$ for $r \in \{1, 2\}$ of the following cases use Lemma C.12.

(i) If $m_1 = 2t_1$, $m_2 = 2t_2$, and $a = 0$, then $V^{(1)}(x) - V^{(1)}(y) = t_1 + t_2 = 0 \pmod{nP + 1}$. However this is impossible, as $0 < t_1 + t_2 \leq n < nP + 1$.

(ii) If $m_1 = 2t_1$, $m_2 = 2t_2$, and $a = 1$, then $V^{(1)}(x) - V^{(1)}(y) = t_1 - t_2 = 0 \pmod{nP + 1}$. Since $|t_1 - t_2| \leq n < nP + 1$, we have $t_1 = t_2$. Furthermore using $t_1 = t_2$, we can have $V^{(2)}(x) - V^{(2)}(y) = t_1^2 + t_1i - t_2^2 - t_2j = t_1(i - j) = 0 \pmod{2nP + P^2 + 1}$. Note that the length of a 2-periodic run in $x$ and $y$ is at most $P$, this means that $2t_1, 2t_2 \leq P$. However, $0 < |t_1(i - j)| \leq Pn < 2nP + P^2 + 1$ which is a contradiction.

(iii) If $m_1 = 2t_1$, $m_2 = 2t_2 + 1$, and $a = 0$, then $V^{(1)}(x) - V^{(1)}(y) = t_1 - t_2 - j = 0 \pmod{nP + 1}$. However this is impossible, as $0 < |t_1 - t_2 - j| < nP + 1$.

(iv) If $m_1 = 2t_1$, $m_2 = 2t_2 + 1$, and $a = 1$, then $V^{(1)}(x) - V^{(1)}(y) = t_1 + t_2 + j = 0 \pmod{nP + 1}$. Again, since $0 < |t_1 + t_2 + j| \leq 2n < nP + 1$, we have a contradiction.

(v) If $m_1 = 2t_1 + 1$, $m_2 = 2t_2$, and $a = 0$, then $V^{(1)}(x) - V^{(1)}(y) = -i - t_1 + t_2 = 0 \pmod{nP + 1}$. Since $| -i - t_1 + t_2| \leq n < nP + 1$, this means that $t_2 = t_1 + i$. Furthermore using $t_2 = t_1 + i$ and $t_1, t_2 \leq \frac{1}{2}P$, we can have $2nP + P^2 + 1 > V^{(2)}(x) - V^{(2)}(y) = i^2 + t_1i - \left(i + 2t_1 + 1\right) + t_2^2 + t_2j \geq t_2^2 + t_1 + i + t_2i - \left(i + 2t_1 + 1\right) \geq 2t_2^2 - \left(\frac{2t_2}{2}\right) > 0$. This contradicts the assumption that $V^{(2)}(x) - V^{(2)}(y) = 0 \pmod{2nP + P^2 + 1}$.

(vi) If $m_1 = 2t_1 + 1$, $m_2 = 2t_2$, and $a = 1$, then $V^{(1)}(x) - V^{(1)}(y) = -i - t_1 + t_2 = 0 \pmod{nP + 1}$. However this is impossible, as $0 < | -i - t_1 + t_2| \leq 2n < nP + 1$.

(vii) If $m_1 = 2t_1 + 1$, $m_2 = 2t_2 + 1$, and $a = 0$, then $V^{(1)}(x) - V^{(1)}(y) = -i - t_1 - j - t_2 = 0 \pmod{nP + 1}$. However this is impossible, as $0 < | -i - t_1 - j - t_2| \leq 2n < nP + 1$.

(viii) If $m_1 = 2t_1 + 1$, $m_2 = 2t_2 + 1$, and $a = 1$, then $V^{(1)}(x) - V^{(1)}(y) = -i - t_1 + j + t_2 = 0 \pmod{nP + 1}$. However this is impossible, as $0 < | -i - t_1 + j + t_2| \leq n < nP + 1$.

Form (B): Suppose that $x = u a_{m_1} (\alpha a_{m_3}^2 (m_3) \beta a_{m_2}^b (m_2)) v$ and $y = u a_{m_1} (\alpha a_{m_3}^2 (m_3 + 2) a_{m_2}^l (m_2)) v$ with $m_1, m_2 \geq 2$. Since the length of a 2-periodic run in $x$ and $y$ is at most $P$, we have that $m_1, m_2, m_3 \leq P$.

Without loss of generality, we assume that $a = 1$. As before, let $i$, $j$, $k$ denote the starting indices of $a(m_1)$, $a(m_3)$, and $a(m_2)$, respectively. So, $|u| + 1 = i,$
Appendix C. Proof of Theorem 5.32

|u| + m_1 + 2 = j and |u| + m_1 + m_3 + 3 = k. Observe that once the parity of m_1 and m_3 are fixed, and the value of b is fixed, then the parity of m_2 can be uniquely determined. Hence, we need to consider eight cases based on the value of b, the parity of m_1, and the parity of m_3. For each case, we then compute VT^{(1)}(x) - VT^{(1)}(y) and we can check that 0 < |VT^{(1)}(x) - VT^{(1)}(y)| ≤ n < nP + 1, leading to a contradiction as shown in the following computations.

(i) If m_1 = 2t_1, m_3 = 2t_3, and b = 0
   Then we have m_2 = 2t_2, and VT^{(1)}(x) - VT^{(1)}(y) = t_1 + t_2 + t_3 = 0 (mod nP + 1).

(ii) If m_1 = 2t_1, m_3 = 2t_3, and b = 1
   Then we have m_2 = 2t_2 + 1, and VT^{(1)}(x) - VT^{(1)}(y) = t_1 - j - t_2 - t_3 = 0 (mod nP + 1).

(iii) If m_1 = 2t_1, m_3 = 2t_3 + 1, and b = 0
   Then we have m_2 = 2t_2, and VT^{(1)}(x) - VT^{(1)}(y) = t_1 + t_2 - k - t_3 = 0 (mod nP + 1).

(iv) If m_1 = 2t_1, m_3 = 2t_3 + 1, and b = 1
   Then we have m_2 = 2t_2, and VT^{(1)}(x) - VT^{(1)}(y) = t_1 - j - t_2 + k + t_3 = 0 (mod nP + 1).

(v) If m_1 = 2t_1 + 1, m_3 = 2t_3, and b = 0
   Then we have m_2 = 2t_2, and VT^{(1)}(x) - VT^{(1)}(y) = i + t_1 + t_2 + t_3 = 0 (mod nP + 1).

(vi) If m_1 = 2t_1 + 1, m_3 = 2t_3, and b = 1
   Then we have m_2 = 2t_2, and VT^{(1)}(x) - VT^{(1)}(y) = i + t_1 - j - t_2 - t_3 = 0 (mod nP + 1).

(vii) If m_1 = 2t_1 + 1, m_3 = 2t_3 + 1, and b = 0
    Then we have m_2 = 2t_2, and VT^{(1)}(x) - VT^{(1)}(y) = i + t_1 + t_2 - k - t_3 = 0 (mod nP + 1).

(viii) If m_1 = 2t_1 + 1, m_3 = 2t_3 + 1, and b = 1
     Then we have m_2 = 2t_2, and VT^{(1)}(x) - VT^{(1)}(y) = i + t_1 - j - t_2 + k + t_3 = 0 (mod nP + 1).
Hence we have shown that it is impossible for two codewords $x$ and $y$ to satisfy any of the two forms in Theorem C.8. Therefore, $|D_2(x) \cap D_2(y)| \leq 4$ and so, $\nu(C, D_2) \leq 4$, as required.

Remark C.1. We have demonstrated that for any codeword $x \in C(n, P; c_1, c_2)$, we can uniquely determine $x$ from any set of reads $R \subseteq D_2(x)$ with $|R| \geq 5$. It turns out that we can find $x$ in $O(n^2)$ time. We do the decoding as follows. First, we assume that one of the deleted bit is a 0(or 1), and attempt to insert 0(or 1) in any possible locations ($O(n)$ locations). Next we can correct the second deletion using the decoding algorithm of the well-known VT code ($O(n)$ time). Finally, we check the correctness of the resulting sequence by checking if its second order VT in $(\mod n)P + P^2 + 1$ is equal to $c_2$ and if its 2-periodic run has length at most $P$ as described in Construction C. All of these steps take $O(n^2)$ time in total.
List of Author’s Awards, Patents, and Publications\textsuperscript{1}

Award

- ISITA 2020 Best Student Paper Award

\textsuperscript{1}The superscript * indicates joint first authors
Bibliography


סדרות ושימושיהן

הبيب ע"ע מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר

דוקטור לפילוסופיה

ג'רמן קריסנטה

הנהלת הקמפוס הטכנולוגי - מרכז טכנולוגיה לים הדר

מחזור 2021

דצמבר