Efficient Encoding/Decoding of Irreducible Words for Codes Correcting Tandem Duplications

Yeow Meng Chee, Johan Chrisnata, Han Mao Kiah, and Tuan Thanh Nguyen
School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
email: {ymchee, jchrisnata, hmkiah, nguyentu001}@ntu.edu.sg

***Abstract*—Tandem duplication is the process of inserting a copy of a segment of DNA adjacent to the original position. Motivated by applications that store data in living organisms, Jain *et al.* (2017) proposed the study of codes that correct tandem duplications. All code constructions are based on *irreducible words*.**

We study efficient encoding/decoding methods for irreducible words. First, we describe an (ℓ, m)-finite state encoder and show that when $m = \Theta(1/\epsilon)$ and $\ell = \Theta(1/\epsilon)$, the encoder has rate that is ϵ away from the optimal. Next, we provide ranking/unranking algorithms for irreducible words and modify the algorithms to reduce the space requirements for the finite state encoder.

I. Introduction

Advances in synthesis and sequencing technologies have made DNA macromolecules an attractive medium for digital information storage. Besides being biochemically robust, DNA strands offer ultrahigh storage densities of 10^{15}-10^{20} bytes per gram of DNA, as demonstrated in recent experiments (see [1, Table 1]).

These synthetic DNA strands may be stored *ex vivo* or *in vivo*. When the DNA strands are stored *ex vivo* or in a non-biological environment, code design takes into account the synthesising and sequencing platforms being used (see [2] for a survey of the various coding problems). In contrast, when the DNA strands are stored *in vivo* or recombined with the DNA of a living organism, we design codes to correct errors due to the biological mutations.

This work looks at the latter case, and specifically, examines codes that correct errors due to *tandem duplications*. Tandem duplications or repeats is one of the two common repeats found in the human genome [3] and they are caused by slipped-strand mispairings [4]. They occur in DNA when a pattern of one or more nucleotides is repeated and the repetitions are directly adjacent to each other. For example, consider the string or word `AGTAGTCTGC`. The substring `AGTAGT` is a tandem repeat, and we say that `AGTAGTCTGC` is generated from `AGTCTGC` by a *tandem duplication* of length three.

Jain *et al.* [5] first proposed the study of codes that correct errors due to tandem duplications. In the same paper, Jain *et al.* used *irreducible* words (see Section I-A for definition) to construct a family of codes that correct tandem duplications of lengths at most k, where $k \in \{2, 3\}$. While these codes are optimal in size for the case $k = 2$, these codes are not optimal for $k = 3$, and in fact, Chee *et al.* [6] constructed a family of codes with strictly larger size. Recently, Jain *et al.* [7] looked at other error mechanisms, and studied the capacity of these tandem-duplication systems in the presence of point-mutation noise (substitution errors).

In this paper, we look at encoding/decoding methods for irreducible words. In particular, we provide polynomial-time algorithms that encodes either exactly the rates of irreducible words or close to the asymptotic rates of irreducible words. While the encoding/decoding algorithms are standard in constrained coding [8] and combinatorics literature [9], our contribution is a detailed analysis of the space and time complexities of the respective algorithms. Before we state the main results of the paper, we go through certain notations.

A. Notation and Terminology

Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. Let $\Sigma_q = \{0, 1, \cdots q-1\}$ be an alphabet of $q \geqslant 2$ symbols. For a positive integer n, let Σ_q^n denote the set of all words of length n over Σ_q, and let Σ_q^* denote the set of all words over Σ_q with finite length. Given two words $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^*$, we denote their concatenation by $\boldsymbol{xy}$.

We state the *tandem duplication* rules. For integers $k \leqslant n$ and $i \leqslant n - k$, we define $T_{i,k} : \Sigma_q^n \to \Sigma_q^{n+k}$ such that $T_{i,k}(\boldsymbol{x}) = \boldsymbol{uvvw}$, where $\boldsymbol{x} = \boldsymbol{uvw}$, $|\boldsymbol{u}| = i$, $|\boldsymbol{v}| = k$.

If a finite sequence of tandem duplications of length at most k is performed to obtain $\boldsymbol{y}$ from $\boldsymbol{x}$, then we say that $\boldsymbol{y}$ is a $\leqslant k$-*descendant* of $\boldsymbol{x}$, or $\boldsymbol{x}$ is a $\leqslant k$-*ancestor* of $\boldsymbol{y}$. Given a word $\boldsymbol{x}$, we define the $\leqslant k$-*descendant cone* of $\boldsymbol{x}$ is the set of all $\leqslant k$-descendants of $\boldsymbol{x}$ and denote this cone by $D^*_{\leqslant k}(\boldsymbol{x})$.

Example 1. Consider $\boldsymbol{x} = 01210$ over Σ_3. We have $T_{1,3}(\boldsymbol{x}) = 01211210$ and $T_{0,2}(01211210) = 0101211210$. So, $0101211210 \in D^*_{\leqslant 3}(\boldsymbol{x})$.

Definition 1 ($\leqslant k$-Tandem-Duplication Codes)**.** A subset $\mathcal{C} \subseteq \Sigma_q^n$ is a $\leqslant k$-*tandem-duplication code* if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$ and $\boldsymbol{x} \neq \boldsymbol{y}$, we have that $D^*_{\leqslant k}(\boldsymbol{x}) \cap D^*_{\leqslant k}(\boldsymbol{y}) = \varnothing$. We say that $\mathcal{C}$ is an $(n, \leqslant k; q)$-TD code.

The *size* of $\mathcal{C}$ refers to $|\mathcal{C}|$, while the *rate* of $\mathcal{C}$ is given by $(1/n)\log_q|\mathcal{C}|$. Given an infinite family $\{\mathcal{C}_n : \mathcal{C}_n \text{ is of length } n\}_{n=1}^{\infty}$, its *asymptotic rate* is given by $\lim_{n\to\infty}(1/n)\log_q|\mathcal{C}_n|$.

B. Irreducible Words

Of interest is a family of tandem-duplication codes constructed by Jain *et al.* [5]. Crucial to the code construction is the concept of irreducible words and roots.

Definition 2. A word is $\leqslant k$-*irreducible* if it cannot be deduplicated into shorter words with deduplications of length at most k. We use $\mathrm{Irr}_{\leqslant k}(n, q)$ to denote the set of all $\leqslant k$-irreducible words of length n over Σ_q. The $\leqslant k$-ancestors of $\boldsymbol{x} \in \Sigma_q^*$ that are $\leqslant k$-irreducible words are called the $\leqslant k$-*roots* of $\boldsymbol{x}$.

Construction 1 (Jain *et al.* [5])**.** *For $k \in \{1, 2, 3\}$ and $n \geqslant k$. An $(n, \leqslant k; q)$-TD-code $\mathcal{C}(n, \leqslant k; q)$ is given by*

$$\mathcal{C}(n, \leqslant k; q) \triangleq \bigcup_{i=1}^{n} \{\xi_{n-i}(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathrm{Irr}_{\leqslant k}(i, q)\}.$$

Here, $\xi_i(\boldsymbol{x}) = \boldsymbol{x}z^i$, where z is the last symbol of $\boldsymbol{x}$.

978-1-5386-4780-6/18/$31.00©2018 IEEE

We point out certain advantages of Construction 1.

(a) *Almost optimal rates*. Jain *et al.* demonstrated that Construction 1 is optimal for $k \in \{1, 2\}$. However, when $k = 3$, Chee *et al.* [6] provided constructions that achieve almost twice the size in Construction 1 (see [6, Table I]). Unfortunately, the asymptotic rate of the latter is the same as Construction 1. Therefore, the set of irreducible words gives the best known asymptotic rates for $k = 3$.
Furthermore, for $q \geqslant 5$ and $k = 3$, the asymptotic rates of Construction 1 differs from a theoretical upper bound (see [6, Proposition 4] and Table I) by at most 0.01. In other words, Construction 1 is almost optimal in terms of rates.

(b) *Linear-time decoding*. Consider $\boldsymbol{x} \in \mathcal{C}(n, \leqslant k; q)$ and we read $\boldsymbol{y} \in D^*_{\leqslant k}(\boldsymbol{x})$. To retrieve the codeword $\boldsymbol{x}$, we simply compute the $\leqslant k$-root of $\boldsymbol{y}$ and extend the root if the root is shorter than n. Jain *et al.* showed that there is at most one root when $k \in \{1, 2, 3\}$, while Chee *et al.* provided algorithms to compute these roots in linear time [6].

In view of these points, we study other practical aspects of Construction 1. Specifically, we look at *efficient encoding* of messages in Σ_q^ℓ to codewords in $\boldsymbol{x} \in \mathcal{C}(n, \leqslant k; q)$ for some $\ell < n$.

To this end, we look at the rates of $\mathcal{C}(n, \leqslant k; q)$. Let $I_{\leqslant k}(n, q) \triangleq |\mathrm{Irr}_{\leqslant k}(n, q)|$. Then the size of $\mathcal{C}(n, \leqslant k; q)$ is given by $\sum_{i=1}^n I_{\leqslant k}(i, q)$. Let $\mathrm{rate}_{\leqslant k}(n, q)$ and $\mathrm{rate}_{\leqslant k}(q)$ denote the rate and asymptotic rate of $\mathcal{C}(n, \leqslant k; q)$, respectively. In other words, $\mathrm{rate}_{\leqslant k}(n, q) \triangleq (1/n) \log_q |\mathcal{C}(n, \leqslant k; q)|$ and $\mathrm{rate}_{\leqslant k}(q) \triangleq \lim_{n\to\infty} \mathrm{rate}_{\leqslant k}(n, q)$. Jain *et al.* observed that $\bigcup_{n=1}^\infty \mathrm{Irr}_{\leqslant k}(n, q)$ is a regular language and hence,

$$\mathrm{rate}_{\leqslant k}(q) = \lim_{n\to\infty} \frac{\log_q I_{\leqslant k}(n, q)}{n}. \tag{1}$$

Furthermore, using Perron-Frobenius theory (see [8]), Jain *et al.* computed $\mathrm{rate}_{\leqslant 3}(3)$ to be approximately 0.347934. In view of (1), we look at encoding of the words in $\mathrm{Irr}_{\leqslant k}(n)$ instead and the extension of our encoding methods to $\mathcal{C}(n, \leqslant k; q)$ is straightforward.

In this paper, we focus on the case $k \in \{2, 3\}$ as the results for $k = 1$ is well known. Specifically, the size of $\mathrm{Irr}_{\leqslant 1}(n, q)$ is given by $q(q-1)^{n-1}$ and linear-time encoding methods can be obtained via differential coding (see for example, [8]).

C. Our Contributions

We first develop a recursive formula for $I_{\leqslant k}(n, q)$ and hence, provide a formula for the asymptotic rate for $\mathcal{C}(n, \leqslant k; q)$. We then provide two efficient encoding methods and use combinatorial insights provided by the recursive formula to analyse the space and time complexities.

Specifically, our main contributions are as follows.

(A) We compute $\mathrm{rate}_{\leqslant k}(q)$ for all q and $k \in \{2, 3\}$ in Section II.

(B) In Section III, we propose an (ℓ, m)-finite state encoder with rate ℓ/m. Furthermore, we show that we can choose the lengths ℓ and m to be small and yet come close to the asymptotic rate. In particular, if we choose $m = \Theta(1/\epsilon)$ and $\ell = \Theta(1/\epsilon)$, we showed that the rate is at least $\mathrm{rate}_{\leqslant k}(q)$. Here, the running time for the encoder is linear in codeword length n for constant ϵ .

(C) Using bijections developed Section II, we provide a ranking/unranking algorithm that encodes with rate equal to $(1/n) \log_q(\mathrm{Irr}_{\leqslant k}(n, q))$ in Section IV. This algorithm runs in $O(n^2)$ time using $O(n^2)$ space. Furthermore, this ranking/unranking technique can be modified to reduce the space requirement to $O(m^2)$ in the (ℓ, m)-finite state encoder.

Due to space constraints, we present proofs and illustrate examples for the case $k = 2$ and simply state the relevant results for $k = 3$. The detailed proofs are deferred to the full paper.

II. Enumerating Irreducible Words

In this section, we compute $\mathrm{rate}_{\leqslant k}(q)$ for all q and $k \in \{2, 3\}$ by obtaining a recursive formula for $I_{\leqslant k}(n, q)$. While the Perron-Frobenius theory (see [8]) is sufficient to determine the asymptotic rates, the recursive formula is useful in the analysis of the finite state encoder in Section III and the development of the ranking/unranking methods in Section IV.

To this end, we partition the set of irreducible words into two classes and provide bijections from irreducible words of shorter lengths into them. Specifically, notice that the suffix of an irreducible word is of the form either aba or abc, where a, b, c are distinct symbols. Hence, we let $\mathrm{Irr}^{(s)}_{\leqslant k}(2, n, q)$ and $\mathrm{Irr}^{(s)}_{\leqslant k}(3, n, q)$ denote the set of irreducible words with length-three suffixes that have two and three distinct symbols, respectively.

In the case $k = 2$, we consider the following maps for $n \geqslant 4$,

$$\phi : \mathrm{Irr}_{\leqslant 2}(n-1) \times [q-2] \to \mathrm{Irr}^{(s)}_{\leqslant 2}(3, n, q),$$
$$\psi : \mathrm{Irr}_{\leqslant 2}(n-2) \times [q-2] \to \mathrm{Irr}^{(s)}_{\leqslant 2}(2, n, q).$$

We first define ϕ. If $\boldsymbol{x} = x_1 x_2 \ldots x_{n-1} \in \mathrm{Irr}_{\leqslant 2}(n-1)$ and $i \in [q-2]$, set σ to be the ith element in $\Sigma_q \setminus \{x_{n-2}, x_{n-1}\}$. Then set $\phi(\boldsymbol{x}, i) = x_1 x_2 \ldots x_{n-1}\sigma$.

For ψ, let $\boldsymbol{x} = x_1 x_2 \ldots x_{n-2} \in \mathrm{Irr}_{\leqslant 2}(n-2)$ and $i \in [q-2]$ and set σ to be the ith element in $\Sigma_q \setminus \{x_{n-3}, x_{n-2}\}$. Then set $\psi(\boldsymbol{x}, i) = x_1 x_2 \ldots x_{n-2}\sigma x_{n-2}$.

Proposition 1. *The maps ϕ and ψ are bijections.*

Proof. We construct the inverse map for ϕ. Specifically, we set $\phi^{-1} : \mathrm{Irr}^{(s)}_{\leqslant 2}(3, n, q) \to \mathrm{Irr}_{\leqslant 2}(n-1) \times [q-2]$ such that $\phi^{-1}(\boldsymbol{x}) = (x_1 \ldots x_{n-1}, i)$, where i is the index of x_n in $\Sigma_q \setminus \{x_{n-2}, x_{n-1}\}$. It can be verified that $\phi \circ \phi^{-1}$ and $\phi^{-1} \circ \phi$ are identity maps on their respective sets. Similarly, the inverse map for ψ is given by $\psi^{-1} : \mathrm{Irr}^{(s)}_{\leqslant 2}(2, n, q) \to \mathrm{Irr}_{\leqslant 2}(n-2) \times [q-2]$ such that $\psi^{-1}(\boldsymbol{x}) = (x_1 \ldots x_{n-2}, i)$, where i is the index of x_{n-1} in $\Sigma_q \setminus \{x_{n-3}, x_{n-2}\}$. ■

The following corollary is then immediate.

Corollary 1. *We have that $I_{\leqslant 2}(2, q) = q(q-1)$, $I_{\leqslant 2}(3, q) = q(q-1)^2$, and*

$$I_{\leqslant 2}(n, q) = (q-2)I_{\leqslant 2}(n-1, q) + (q-2)I_{\leqslant 2}(n-2, q) \tag{2}$$

for $n \geqslant 4$. Therefore, the asymptotic rate is $\mathrm{rate}_{\leqslant 2}(q) = \log_q \lambda_2$, where $\lambda_2 = (q - 2 + \sqrt{q^2 - 4})/2$.

In the next section, we are interested in irreducible words with certain prefixes or suffixes. Specifically, let $\boldsymbol{p}$ be a word of length $\ell < n$. Then we denote the set of irreducible words of length n

with prefix $\boldsymbol{p}$ by $\mathrm{Irr}^{(p)}_{\leqslant k}(\boldsymbol{p},n,q)$. The set of irreducible words of length n with suffix $\boldsymbol{p}$ is denoted by $\mathrm{Irr}^{(s)}_{\leqslant k}(\boldsymbol{p},n,q)$.

Fix $\boldsymbol{p}$. Notice that the maps ϕ and ψ simply appends one and two symbols to words in their domains. Hence, if we apply the maps to a word with prefix $\boldsymbol{p}$, the image also has the same prefix $\boldsymbol{p}$. Therefore, both ϕ and ψ remain as bijections when we restrict the domains and codomains to the irreducible words with prefix $\boldsymbol{p}$. In other words, we obtain a similar recursion for $\mathrm{Irr}^{(p)}_{\leqslant 2}(\boldsymbol{p},n,q)$.

Corollary 2. *Let $\boldsymbol{p}\in\Sigma_q^\ell$ For $n\geqslant \ell+2$,*

$$\left|\mathrm{Irr}^{(p)}_{\leqslant 2}(\boldsymbol{p},n,q)\right| = (q-2)\left|\mathrm{Irr}^{(p)}_{\leqslant 2}(\boldsymbol{p},n-1,q)\right| + (q-2)\left|\mathrm{Irr}^{(p)}_{\leqslant 2}(\boldsymbol{p},n-2,q)\right|. \quad (3)$$

We conclude this section with the recursion for $\mathrm{Irr}_{\leqslant 3}(n,q)$.

Proposition 2. *We have that $I_{\leqslant 3}(3,q)=q(q-1)^2$, $I_{\leqslant 3}(4,q)=q^2(q-1)(q-2)$, $I_{\leqslant 3}(5,q)=q(q-1)(q-2)(q^2-q-1)$ and*

$$I_{\leqslant 3}(n,q)=(q-2)I_{\leqslant 3}(n-1,q)+(q-3)I_{\leqslant 3}(n-2,q) + (q-2)I_{\leqslant 3}(n-3,q) \quad (4)$$

for $n\geqslant 6$. Therefore, $\mathrm{rate}_{\leqslant 3}(q)=\log_q\lambda_3$, where λ_3 is the largest real root of equation $x^3-(q-2)x^2-(q-3)x-(q-2)=0$.

We compute the values of $\mathrm{rate}_{\leqslant k}(q)$ for $k\in\{2,3\}$ in Table I. Let $T(n,q)$ be the largest size of an $(n,\leqslant 3;q)$-TD code and define $\tau(q)\triangleq(1/n)\limsup_{n\to\infty}\log_q T(n,q)$. From [5], [6], we have that that $\mathrm{rate}_{\leqslant 3}(q)\leqslant\tau(q)\leqslant\mathrm{rate}_{\leqslant 2}(q)$. Therefore, Table I demonstrates that $\mathcal{C}(n,\leqslant 3;q)$ is *almost* optimal for $q\geqslant 5$.

q	3	4	5	6	7	8
$\mathrm{rate}_{\leqslant 2}(q)$	0.4380	0.7249	0.8280	0.8788	0.9081	0.9269
$\mathrm{rate}_{\leqslant 3}(q)$	0.3479	0.7054	0.8208	0.8753	0.9062	0.9258

TABLE I: The asymptotic information rates for k-irreducible words for $k\in\{2,3\}$

III. Finite State Encoder

For integers $\ell<m$, an (ℓ,m)-*finite state encoder* is triple $(\mathcal{S},\mathcal{E},\mathcal{L})$, where $\mathcal{S}$ is a set of *states*, $\mathcal{E}\subset\mathcal{S}\times\mathcal{S}$ is a set of *directed edges*, and $\mathcal{L}:\mathcal{E}\to\Sigma_q^\ell\times\Sigma_q^m$ is an *edge labeling*.

To encode irreducible words, we choose $m\geqslant 2k-1$, and set

$$\mathcal{S}\triangleq\mathrm{Irr}_{\leqslant k}(m,q) \text{ and } \mathcal{E}\triangleq\{(\boldsymbol{x},\boldsymbol{x}'):\boldsymbol{x}\boldsymbol{x}'\in\mathrm{Irr}_{\leqslant k}(2m,q)\}.$$

For $\boldsymbol{x}\in\mathcal{S}$, we define the *neighbours* of $\boldsymbol{x}$ to be $N(\boldsymbol{x})\triangleq\{\boldsymbol{x}':(\boldsymbol{x},\boldsymbol{x}')\in\mathcal{E}\}$. We also consider the quantity $\Delta_{\leqslant k}(m,q)\triangleq\min\{|N(\boldsymbol{x})|:\boldsymbol{x}\in\mathcal{S}\}$ and choose ℓ such that

$$\Delta_{\leqslant k}(m,q)\geqslant q^\ell. \quad (5)$$

We now define the edge labelling $\mathcal{L}$ using this choice of ℓ. For $\boldsymbol{x}\in\mathcal{S}$, since $|N(\boldsymbol{x})|\geqslant q^\ell$, we may use the set Σ^ℓ to index the first q^ℓ words in $N(\boldsymbol{x})$. Hence, for $\boldsymbol{x}'\in S$, if $\boldsymbol{x}'$ is one of the first q^ℓ words, we let $\boldsymbol{y}_{\boldsymbol{x}'}\in\Sigma^\ell$ denote the index. Otherwise, we simply set $\boldsymbol{y}_{\boldsymbol{x}'}=-$. Therefore, for $(\boldsymbol{x},\boldsymbol{x}')\in\mathcal{E}$, we set $\mathcal{L}(\boldsymbol{x},\boldsymbol{x}')=(\boldsymbol{y}_{\boldsymbol{x}'},\boldsymbol{x}')$. Finally, we call this triple an (ℓ,m)- *finite state encoder for irreducible words*.

Example 2. Let $k=2$, $q=3$, $m=3$. Then $\mathcal{S}=\{010,012,020,021,101,102,120,121,201,202,210,212\}$, and

$$N(010)=\{201,210,212\},$$
$$N(012)=\{010,012,021,101,102\}.$$

We verify that $\Delta_{\leqslant 2}(3,3)=3$ and so, we choose $\ell=1$. So, we can set $\mathcal{L}$ to map the edges exiting the state 010 as follow:

$$(010,201)\mapsto(0,201),\ (010,210)\mapsto(1,210),\ (010,212)\mapsto(2,212).$$

We represent the mapping $\mathcal{L}$ using the following lookup table.

$\boldsymbol{x}$	$N(\boldsymbol{x})$				
	0	1	2	–	–
010	201	210	212	–	–
012	010	012	021	101	102
020	102	120	121	–	–
021	012	020	021	201	202
101	201	202	210	–	–
102	010	012	101	102	120
120	102	120	121	210	212
121	012	020	021	–	–
201	020	021	201	202	210
202	101	102	120	–	–
210	120	121	201	210	212
212	010	012	021	–	–

Here, to determine $\mathcal{L}(\boldsymbol{x},\boldsymbol{x}')$, we look at the row corresponding to $\boldsymbol{x}$ and look at the column corresponding to $\boldsymbol{x}'$. If the column is $\boldsymbol{y}_{\boldsymbol{x}'}$, then $\mathcal{L}(\boldsymbol{x},\boldsymbol{x}')=(\boldsymbol{y}_{\boldsymbol{x}'},\boldsymbol{x}')$. So, $\mathcal{L}(012,010)=(0,010)$.

A. Encoding

Let s be a positive integer and set $n=s\ell$. Suppose the message $\boldsymbol{y}=\boldsymbol{y}_1\boldsymbol{y}_2\ldots\boldsymbol{y}_s\in\Sigma^{s\ell}$.

To encode $\boldsymbol{y}$ using an (ℓ,m)-finite state encoder for irreducible words, we do the following:

(I) Set $\boldsymbol{x}_0$ to the first word in $\mathcal{S}=\mathrm{Irr}_{\leqslant k}(m,q)$.
(II) For $i\in[s]$, set $\boldsymbol{x}_i$ to be the unique word such that $\mathcal{L}(\boldsymbol{x}_{i-1},\boldsymbol{x}_i)=(\boldsymbol{y}_i,\boldsymbol{x}_i)$.
(III) The encoded irreducible word is $\boldsymbol{x}=\boldsymbol{x}_1\boldsymbol{x}_2\ldots\boldsymbol{x}_s$.

Example 3 (Example 2 continued)**.** Let $s=3$ and consider the message $\boldsymbol{y}=012$. First, we set $\boldsymbol{x}_0=010$. Then $\boldsymbol{x}_1=201$ since $\mathcal{L}(010,201)=(0,201)$. Similarly, $\boldsymbol{x}_2=021$ and $\boldsymbol{x}_3=021$.

Therefore, the encoded word $\boldsymbol{x}$ is 201021021.

Since the encoded word has length sm, the (ℓ,m)-finite state encoder for irreducible words has rate ℓ/m. In the next subsection, we see that ℓ and m can be chosen in such a way that the rate ℓ/m approaches $\mathrm{rate}_{\leqslant k}(q)$ *quickly*.

B. Approaching the Asymptotic Information Rate

Pick $\epsilon>0$. We find suitable values for ℓ and m so that the encoding rate satisfies

$$\ell/m\geqslant\mathrm{rate}_{\leqslant k}(q)-\epsilon. \quad (6)$$

In particular, we show that $\ell=\Theta(1/\epsilon)$ and $m=\Theta(1/\epsilon)$ suffice to guarantee (6).

Recall that ℓ and m are required to satisfy (5). Hence, we determine $\Delta_{\leqslant k}(m,q)$. Surprisingly, these values have the same recursive structure as $I_{\leqslant k}(m,q)$ and therefore, have the same growth rate.

Proposition 3. *We have that* $\Delta_{\leqslant 2}(3,q) = q(q-2)^2$, $\Delta_{\leqslant 2}(4,q) = (q-2)^2(q^2-q-1)$, *and for* $m \geqslant 5$,

$$\Delta_{\leqslant 2}(m,q) = (q-2)\Delta_{\leqslant 2}(m-1,q) + (q-2)\Delta_{\leqslant 2}(m-2,q). \quad (7)$$

Proof. Observe that by symmetry, we have $|N(\boldsymbol{x})| = |N(\boldsymbol{x}')|$ for $\boldsymbol{x}, \boldsymbol{x}' \in \mathrm{Irr}^{(s)}_{\leqslant 2}(2,m,q)$. Similarly, $|N(\boldsymbol{y})| = |N(\boldsymbol{y}')|$ for $\boldsymbol{y}, \boldsymbol{y}' \in \mathrm{Irr}^{(s)}_{\leqslant 2}(3,m,q)$.

We first show that $|N(\boldsymbol{x})| \leqslant |N(\boldsymbol{y})|$ for $\boldsymbol{x} \in \mathrm{Irr}^{(s)}_{\leqslant 2}(2,m,q)$ and $\boldsymbol{y} \in \mathrm{Irr}^{(s)}_{\leqslant 2}(3,m,q)$. Without loss of generality, we assume $\boldsymbol{x} \in \mathrm{Irr}^{(s)}_{\leqslant 2}(010,m,q)$ and $\boldsymbol{y} \in \mathrm{Irr}^{(s)}_{\leqslant 2}(210,m,q)$. Then the neighbours of $\boldsymbol{x}$ and $\boldsymbol{y}$ are given by

$$N(\boldsymbol{x}) = \left\{ \boldsymbol{x}' : 10\boldsymbol{x}' \in \bigcup_{\sigma \notin \{0,1\}} \mathrm{Irr}^{(p)}_{\leqslant 2}(10\sigma, m+2, q) \right\}, \quad (8)$$

$$N(\boldsymbol{y}) = \left\{ \boldsymbol{y}' : 10\boldsymbol{y}' \in \bigcup_{\sigma \neq 0} \mathrm{Irr}^{(p)}_{\leqslant 2}(10\sigma, m+2, q) \right\}. \quad (9)$$

Since $N(\boldsymbol{x}) \subseteq N(\boldsymbol{y})$, the inequality $|N(\boldsymbol{x})| \leqslant |N(\boldsymbol{y})|$ follows. Hence, $\Delta_{\leqslant 2}(m,q) = |N(\boldsymbol{x})|$ where $\boldsymbol{x} \in \mathrm{Irr}^{(s)}_{\leqslant 2}(010,m,q)$.

Since $\Delta_{\leqslant 2}(m,q) = \sum_{\sigma \notin \{0,1\}} \left| \mathrm{Irr}^{(p)}_{\leqslant 2}(10\sigma, m+2, q) \right|$, the recursive equation (7) follows from Corollary 2. ■

For $k=3$, we state the recursive equation without proof.

Proposition 4. *We have that*

$$\Delta_{\leqslant 3}(5,q) = (q-2)(q^2-2q-1)^2,$$
$$\Delta_{\leqslant 3}(6,q) = (q-1)(q^5-6q^4+9q^3+4q^2-8q-9),$$
$$\Delta_{\leqslant 3}(7,q) = (q-2)(q^6-6q^4+9q^3+4q^2-8q-10q+3),$$

and for $m \geqslant 8$,

$$\Delta_{\leqslant 3}(m,q) = (q-2)\Delta_{\leqslant 3}(m-1,q) + (q-3)\Delta_{\leqslant 3}(m-2,q) + (q-2)\Delta_{\leqslant 3}(m-3,q). \quad (10)$$

Recall that λ_2 and λ_3 are roots of the equations $x^2-(q-2)x-(q-2)=0$ and $x^3-(q-2)x^2-(q-3)x-(q-2)=0$, respectively.

Set κ_2 such that $\Delta_{\leqslant 2}(m,q) \geqslant \kappa_2\lambda_2^m$ for $m \in \{3,4\}$. Similarly, set κ_3 so that $\Delta_{\leqslant 3}(m,q) \geqslant \kappa_3\lambda_3^m$ for $m \in \{5,6,7\}$. Then it follows from an inductive argument, (7) and (10) that

$$\Delta_{\leqslant k}(m,q) \geqslant \kappa_k\lambda_k^m \text{ for all } m. \quad (11)$$

We are now ready to present the main theorem of this section.

Theorem 1. *Let* $k \in \{2,3\}$. *Set* $c_k = \mathrm{rate}_{\leqslant k}(q) = \log_q \lambda_k$. *For* $\epsilon > 0$, *if we choose* m *and* ℓ *such that*

$$\ell = \left\lceil \frac{(c_k-\epsilon)(c_k - \log_q \kappa_k)}{\epsilon} \right\rceil, \quad (12)$$

$$m = \left\lceil \frac{\ell - \log_q \kappa_k}{c_k} \right\rceil, \quad (13)$$

then the (ℓ,m)*-finite state encoder has rate at least* $\mathrm{rate}_{\leqslant k}(q) - \epsilon$.

Proof. We have to verify that (5) and (6) hold for the choice of ℓ and m. Now, (12) implies that $\epsilon\ell \geqslant (c_k-\epsilon)(c_k-\log_q\kappa_k)$, and equivalently, $\epsilon\ell \geqslant c_k^2 - c_k\log_q\kappa_k - \epsilon c_k + \epsilon\log_q\kappa_k$. It implies that $c_k\ell \geqslant (c_k^2 - c_k\log_q\kappa_k + c_k\ell) - \epsilon(c_k - \log_q\kappa_k + \ell) = (c_k-\epsilon)(c_k - \log_q\kappa_k + \ell)$. Thus, $c_k\ell/(\ell - \log_q\kappa_k + c_k) \geqslant c_k - \epsilon$. Therefore,

$$\frac{\ell}{m} \geqslant \frac{\ell}{1 + (\ell - \log_q\kappa_k)/c_k} = \frac{c_k\ell}{\ell - \log_q\kappa_k + c_k} \geqslant c_k - \epsilon.$$

Thus, we verify (6). Next, from (11) and (13), we have that

$$\Delta_{\leqslant k}(m,q) \geqslant \kappa_k\lambda_k^{(\ell-\log_q\kappa_k)/\log_q\lambda_k} = q^\ell.$$

Hence, we verify (5) and complete the proof. ■

Therefore, to achieve encoding rates at least $\mathrm{rate}_{\leqslant k}(q) - \epsilon$, we only require $\ell = \Theta(1/\epsilon)$ and $m = \Theta(1/\epsilon)$. If we naively use a lookup table to represent $(\mathcal{S}, \mathcal{E}, \mathcal{L})$, we require $q^{\Theta(1/\epsilon)}$ space. Furthermore, using binary search, the (ℓ,m)-finite state encoder for irreducible words encodes in $O(n/\epsilon)$ time. In the next section, we use combinatorial insights from (2) and (4) to reduce the space requirement to $O(1/\epsilon^2)$.

IV. Ranking/Unranking algorithm

A *ranking function* for a finite set S of cardinality N is a bijection $\mathrm{rank} : S \to [N]$. Associated with the function rank is a unique *unranking function* $\mathrm{unrank} : [N] \to S$, such that $\mathrm{rank}(s) = j$ if and only if $\mathrm{unrank}(j) = s$ for all $s \in S$ and $j \in [N]$. In this section, we present an algorithm for ranking and unranking $\mathrm{Irr}_{\leqslant k}(n,q)$. For ease of exposition, we focus on the case where $k=2$ and defer the case $k=3$ to the full paper.

The basis of our ranking and unranking algorithms is the bijections ϕ and ψ defined in Section II. As implied by the codomains of ϕ and ψ, for $n \geqslant 4$, we order the words in $\mathrm{Irr}_{\leqslant 2}(n,q)$ such that words in $\mathrm{Irr}^{(s)}_{\leqslant 2}(3,n,q)$ are ordered before words in $\mathrm{Irr}^{(s)}_{\leqslant 2}(2,n,q)$. For words in $\mathrm{Irr}_{\leqslant 2}(2,q)$ and $\mathrm{Irr}_{\leqslant 2}(3,q)$, we simply order them lexicographically. We illustrate the idea behind the unranking algorithm through an example.

Example 4. Let $n=6$ and $q=3$. Then the values of $I_{\leqslant 2}(m,q)$ are as follow.

m	2	3	4	5	6
$I_{\leqslant 2}(m,q)$	6	12	18	30	48

Suppose we want to compute $\mathrm{unrank}(40)$. Proposition 1 gives

$$\mathrm{Irr}_{\leqslant 2}(6,3) = \phi(\mathrm{Irr}_{\leqslant 2}(5,3) \times [1]) \cup \psi(\mathrm{Irr}_{\leqslant 2}(4,3) \times [1]).$$

Now, we are interested in the 40th word of $\mathrm{Irr}_{\leqslant 2}(6,3)$. Since $40 > I_{\leqslant 2}(5,3) = 30$, the 40th word of $\mathrm{Irr}_{\leqslant 2}(6,3)$ is the image of the $40-30=10$-th word in $\mathrm{Irr}_{\leqslant 2}(4,3)$ under ψ. Recursing tells us that the 10-th word in $\mathrm{Irr}_{\leqslant 2}(4,3)$ is the 10-th element in $\phi(\mathrm{Irr}_{\leqslant 2}(3,3) \times [1])$. The 10-th element of $\mathrm{Irr}_{\leqslant 2}(3,3)$ is 202. This gives

$$\mathrm{unrank}(40) = \psi(\phi(202,1),1) = \psi(202\textcolor{red}{1},1) = 2021\textcolor{red}{01}.$$

The formal unranking algorithm is described in Algorithm 1.

The corresponding ranking algorithm for $\mathrm{Irr}_{\leqslant 2}(n,q)$ has a similar recursive structure and is described in Algorithm 2.

Algorithm 1 $\texttt{unrank}(n, q, j)$

Input: Integers $n \geq 2$, $q \geqslant 3$, $1 \leq j \leq I_{\leqslant 2}(n, q)$
Output: $\boldsymbol{x}$, where $\boldsymbol{x}$ is the codeword of rank j in $\mathrm{Irr}_{\leqslant 2}(n, q)$

```
if n ≤ 3 then
    return j-th codeword in Irr_{⩽2}(n, q)
if j ⩽ (q − 2)I_{⩽2}(n − 1) then
    j' ← 1 + ⌊(j − 1)/(q − 2)⌋
    i ← (j − 1) (mod q − 2) + 1
    return φ(unrank(n − 1, q, j'), i)
else
    j' ← 1 + ⌊(j − (q − 2)I_{⩽2}(n − 1) − 1)/(q − 2)⌋
    i ← (j − (q − 2)I_{⩽2}(n − 1) − 1) (mod q − 2) + 1
    return ψ(unrank(n − 2, 3, j'), i)
```

Example 5. Let $n = 6$ and $q = 3$ as before. Suppose we want to compute $\texttt{rank}(202101)$. Since $202101 \in \mathrm{Irr}^{(s)}_{\leqslant 2}(2, 6, 3)$, we have that 202101 is obtained from applying ψ to $2021 \in \mathrm{Irr}_{\leqslant 2}(4, 3)$. Again, since $2021 \in \mathrm{Irr}^{(s)}_{\leqslant 2}(3, 6, 3)$, we have that 202 is obtained from applying ϕ to $202 \in \mathrm{Irr}_{\leqslant 2}(3, 3)$. Therefore,

$$\begin{aligned}\mathrm{rank}(202101) &= \mathrm{rank}(2021) + I_{\leqslant 2}(5, 3)\\ &= \mathrm{rank}(202) + I_{\leqslant 2}(5, 3)\\ &= 10 + 30 = 40\end{aligned}$$

Algorithm 2 $\texttt{rank}(n, q, \boldsymbol{x})$

Input: $n \geq 2$, $q \geqslant 3$ and irreducible word $\boldsymbol{x}$ of length n
Output: j, where $1 \leq j \leq I_{\leqslant 2}(n, q)$, the rank of $\boldsymbol{x}$ in $\mathrm{Irr}_{\leqslant 2}(n, q)$

```
if n ≤ 3 then
    return rank(x) in Irr_{⩽2}(n, q)
if x_n ≠ x_{n−2} then
    x' ← x_1 x_2 ... x_{n−1}
    i ← the index of x_n in Σ_q \ {x_{n−2}, x_{n−1}}
    return (rank(n − 1, q, x') − 1)(q − 2) + i
else
    x' ← x_1 x_2 ... x_{n−2}
    i ← the index of x_{n−1} in Σ_q \ {x_{n−3}, x_{n−2}}
    return (rank(n − 2, q, x') − 1)(q − 2) + i + (q − 2)I_{⩽2}(n − 1, q)
```

The set of values of $\{I_{\leqslant 2}(m, q) : m \leqslant n\}$ required in Algorithms 1 and 2 can be precomputed based on the recurrence (2). Since the numbers $I_{\leqslant 2}(m, q)$ grow exponentially, these n stored values require $O(n^2)$ space.

Next, Algorithms 1 and 2 involve $O(n)$ iterations and each iteration involves a constant number of arithmetic operations. Therefore, Algorithms 1 and 2 involve $O(n)$ arithmetics operations and have time complexity $O(n^2)$.

A. Reducing the Space Requirement for the Finite State Encoder

As discussed earlier, a naive implementation of the (ℓ, m)-finite state encoder in Section III requires $q^{\Theta(m)}$ space (assuming $\ell = \Theta(m)$). Here, we modify our unranking algorithm to reduce the space requirement $O(m)$ integers or $O(m^2)$ bits.

Recall the notation in Section III. In particular, let $\boldsymbol{x}_{i-1} \in \mathrm{Irr}_{\leqslant 2}(m, q)$ and $\boldsymbol{y}_i \in \Sigma_q^\ell$. Our encoding task is to determine the irreducible word $\boldsymbol{x}_i$ in $N(\boldsymbol{x}_i)$ whose index corresponds to $\boldsymbol{y}_i$. Equivalently, if j is the rank of $\boldsymbol{y}_i \in \Sigma_q^\ell$, then our task is to find $\boldsymbol{x}_i$ such that its rank in $N(\boldsymbol{x}_{i-1})$ is j. Since $\boldsymbol{x}_{i-1}$ is irreducible and using symmetry, we assume that $\boldsymbol{x}_{i-1} \in \mathrm{Irr}^{(s)}_{\leqslant 2}(010, m, q)$ or $\boldsymbol{x}_{i-1} \in \mathrm{Irr}^{(s)}_{\leqslant 2}(210, m, q)$. Furthermore, (8) and (9) imply that $N(\boldsymbol{x}_{i_1})$ corresponds to a union of $\leqslant$2-irreducible words with prefixes of the form 10σ. Therefore, it suffices to provide ranking/unranking algorithms for $\mathrm{Irr}^{(p)}_{\leqslant 2}(10\sigma, m, q)$.

Since (3) implies that $\mathrm{Irr}^{(p)}_{\leqslant 2}(10\sigma, m, q)$ has the same recursive structure as $\mathrm{Irr}_{\leqslant 2}(m, q)$, we can modify Algorithms 1 and 2 to unrank and rank $\mathrm{Irr}^{(p)}_{\leqslant 2}(10\sigma, m, q)$.

To rank/unrank $\mathrm{Irr}^{(p)}_{\leqslant 2}(10\sigma, m, q)$ require $O(m)$ precomputed integers. Assuming q is constant, we require only $O(m)$ integers or $O(m^2)$ bits. Hence, the running time is increased to $O(m^2)$.

V. Conclusion

For $k \in \{2, 3\}$ and all q, we provided an explicit recursive formula for $\mathrm{Irr}_{\leqslant k}(n, q)$ and hence, derived the expressions for $\mathrm{rate}_{\leqslant k}(q)$.

We design efficient encoders/decoders for $\mathrm{Irr}_{\leqslant k}(n, q)$.

(i) We provide an (ℓ, m)-finite state encoder and showe that for all $\epsilon > 0$, if we choose $m = \Theta(1/\epsilon)$ and $\ell = \Theta(1/\epsilon)$, the encoder achieves rate that is at least $\mathrm{rate}_{\leqslant k}(q) - \epsilon$. The implementation of the finite state encoder with a lookup table runs in $O(n/\epsilon)$ time and requires $q^{\Theta(1/\epsilon)}$ space. However, if we use the ranking/unranking method in Section IV, the encoder runs in $O(n/\epsilon^2)$ time and requires $O(1/\epsilon)$ space.

(ii) We provide an unranking algorithm for irreducible words whose encoding rate is $(1/n)\log_q(\mathrm{Irr}_{\leqslant k}(n, q)) \geqslant \mathrm{rate}_{\leqslant k}(q)$. The encoder runs in $O(n^2)$ time and requires $O(n^2)$ space.

Acknowledgment

Research of Y. M. Chee is partially funded by the Singapore Ministry of Education under grants MOE2017-T3-1-007 and MOE2015-T2-2-086. Research of J. Chrisnanta and H. M. Kiah is partially funded by the Singapore Ministry of Education under grants MOE2015-T2-2-086 and MOE2016-T1-001-156.

References

[1] S. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage", *Scientific Reports*, no. 5011, vol. 7, 2017.

[2] S. Yazdi, H. M. Kiah, E. R. Garcia, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: Trends and methods," *IEEE Trans. Molecular, Biological, Multi-Scale Commun.*, vol. 1, no. 3, pp. 230–248, 2015.

[3] E. S. Lander, L. M. Linton, B. Birren, C. Nusbaum, M. C. Zody, J. Baldwin, K. Devon, K. Dewar, M. Doyle, W. FitzHugh et al., "Initial sequencing and analysis of the human genome", *Nature*, vol. 409, no. 6822, pp. 860–921, 2001.

[4] N. I. Mundy and A. J. Helbig, "Origin and evolution of tandem repeats in the mitochondrial DNA control region of shrikes (lanius spp.)," *Journal of Molecular Evolution*, vol. 59, no. 2, pp. 250–257, 2004.

[5] S. Jain, F. Farnoud, M. Schwartz and J. Bruck, "Duplication-Correcting Codes for Data Storage in the DNA of Living Organisms," *IEEE Trans. Inform. Theory*, vol. 63, no. 8, pp. 4996–5010, 2017.

[6] Y. M. Chee, J. Chrisnata, H. M. Kiah, and T. T. Nguyen, "Deciding the confusability of words under tandem repeats," *preprint arXiv:1707.03956*, 2017.

[7] S. Jain, F. Farnoud, M. Schwartz and J. Bruck, "Noise and Uncertainty in String-Duplication Systems," in *Proc. 2017 IEEE Intl. Symp. Inform. Theory, Aachen, Germany*, Jun. 2017, pp. 3120–3124.

[8] B. H. Marcus, R. M. Roth, and P. H. Siegel, "An Introduction to Coding for Constrained System", Oct 2001.

[9] A. Nijenhuis, and H. S. Wilf, *Combinatorial Algorithms: for Computers and Calculators*. Elsevier, 2014.