Deciding the Confusability of Words under Tandem Repeats in Linear Time

YEOW MENG CHEE, National University of Singapore, Singapore
JOHAN CHRISNATA, HAN MAO KIAH, and TUAN THANH NGUYEN,
Nanyang Technological University, Singapore

Tandem duplication in DNA 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. (2016) proposed the study of codes that correct tandem duplications to improve the reliability of data storage. We investigate algorithms associated with the study of these codes.

Two words are said to be $\leqslant k$-confusable if there exists a sequence of tandem duplications for each word, where each duplication is of length at most k, such that the resulting two words after duplications are equal. For $k = 3$, we demonstrate that the problem of deciding whether two words is $\leqslant 3$-confusable is linear-time solvable through a characterisation that can be checked efficiently. Combining with previous results, the decision problem is linear-time solvable for $k \leqslant 3$. We conjecture that this problem is undecidable for $k > 3$.

Using insights gained from the algorithm, we study the size of tandem-duplication codes. We improve the previous known upper bound and then construct codes with larger sizes as compared to the previous constructions. We determine the sizes of optimal tandem-duplication codes for lengths up to 20, develop recursive methods to construct tandem-duplication codes for all word lengths, and compute explicit lower bounds for the size of optimal tandem-duplication codes for lengths from 21 to 30.

CCS Concepts: • **Mathematics of computing** → **Combinatorics on words**; • **Applied computing** → **Bioinformatics**;

Additional Key Words and Phrases: Tandem duplications, DNA-based data storage

ACM Reference format:
Yeow Meng Chee, Johan Chrisnata, Han Mao Kiah, and Tuan Thanh Nguyen. 2019. Deciding the Confusability of Words under Tandem Repeats in Linear Time. *ACM Trans. Algorithms* 15, 3, Article 42 (July 2019), 22 pages.
https://doi.org/10.1145/3338514

1 INTRODUCTION

At the beginning of the millenium, Lander et al. [9] published a draft sequence of the human genome and reported that more than 50% of the human genome consists of repeated substrings [9]. There are two types of common repeats: *interspersed* repeats and *tandem* repeats. Interspersed

This work is supported in part by the Singapore Ministry of Education under Grant No.: MOE2015-T2-2-086.
Authors' addresses: Y. M. Chee, National University of Singapore, Department of Industrial Systems Engineering and Management, 1 Engineering Drive 2, Singapore 117576; email: pvocym@nus.edu.sg; J. Chrisnata, H. M. Kiah, and T. T. Nguyen, Nanyang Technological University, School of Physical and Mathematical Sciences, 21 Nanyang Link, Singapore 637371; emails: {johanchr001, hmkiah, nguyentu001}@ntu.edu.sg.

1549-6325/2019/07-ART42 $15.00
https://doi.org/10.1145/3338514

repeats are caused by transposons when a segment of DNA is copied and pasted into new positions of the genome. In contrast, tandem repeats are caused by slipped-strand mispairings [13], and they occur when a pattern of one or more nucleotides is repeated and the repetitions are adjacent to each other. For example, consider the 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. Tandem repeats are believed to be the cause of several genetic disorders [3, 17, 18] and this motivated the study of tandem duplication mechanisms in a variety of contexts.

- **Formal languages**: Leopold et al. [10, 11] defined the *unbounded duplication language* and the *k-bounded duplication language* to be the set of words generated by seed word under tandem duplications of unbounded length and tandem duplications of length up to k, respectively. They investigated certain decidability problems involving unbounded duplication languages and showed that all k-bounded duplication languages are context free. Furthermore, they showed that k-bounded duplication language is always regular for any binary seed and $k \geqslant 1$. However, the k-bounded duplication language is not regular for any *square-free* seed word over an alphabet of at least three letters and $k \geqslant 4$. More recently, Jain et al. [6] completed this characterization and proved that k-bounded duplication languages are regular for $k \leqslant 3$.
- **Information theory**: Farnoud et al. [2] introduced the concept of *capacity* to determine average information content of a k-bounded duplication language. Later, Jain et al. [6] introduced the notion of *expressiveness* to measure a language's capability to generate words that contain certain desired substrings. A complete characterization of fully expressive k-bounded duplication languages was provided by Jain et al. for all alphabet sizes and all k.
- **Codes correcting tandem duplications**: Motivated by applications that store data in living organisms [1, 4, 12, 15], Jain et al. [7] proposed the study of codes that correct tandem duplications to improve the reliability of data storage. They investigated various types of tandem duplications and provided optimal code construction for uniform tandem duplication channel and in the case where tandem duplication length is at most two.

In this article, we study the last problem and investigate algorithms associated with these codes. In particular, given two words $\mathbf{x}$ and $\mathbf{y}$, we look for efficient algorithms that answer the following question:

> When are the words $\mathbf{x}$ and $\mathbf{y}$ *confusable* under tandem repeats? In other words, are there two sequences of tandem duplications such that the resulting words $\mathbf{x}'$ and $\mathbf{y}'$ are equal?

Interestingly, as we demonstrate, even for small duplication lengths, the solutions to this question are nontrivial. This is surprising as efficient algorithms are well known for analogous questions in other problems in string matching [14]. Before we give an account of these results, we introduce some necessary notations and provide a formal statement of our problem.

2 PRELIMINARIES

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

We state the *tandem duplication* rules. For nonnegative integers $k \leqslant n$ and $i \leqslant n-k$, we define $T_{i,k} : \Sigma_q^n \to \Sigma_q^{n+k}$ such that

$$T_{i,k}(\boldsymbol{x}) = \boldsymbol{uvvw}, \text{ where } \boldsymbol{x} = \boldsymbol{uvw}, \ |\boldsymbol{u}| = i, \ |\boldsymbol{v}| = k.$$

If a finite sequence of tandem duplications of length k is performed to obtain $\boldsymbol{y}$ from $\boldsymbol{x}$, then we say that $\boldsymbol{y}$ is a *k-descendant* of $\boldsymbol{x}$, or $\boldsymbol{x}$ is a *k-ancestor* of $\boldsymbol{y}$, and denote this relation by $\boldsymbol{x} \overset{*}{\underset{k}{\Rightarrow}} \boldsymbol{y}$. Formally, $\boldsymbol{x} \overset{*}{\underset{k}{\Rightarrow}} \boldsymbol{y}$ means that for some positive t, there exist t non-negative integers $i_1, i_2, \ldots i_t$ such that $\boldsymbol{y} = T_{i_t,k} \circ T_{i_{t-1},k} \circ \cdots \circ T_{i_1,k}(\boldsymbol{x})$.

Given a sequence $\boldsymbol{x}$ and integer k, we consider the set of words that may be obtained from $\boldsymbol{x}$ via a finite number of tandem duplications of length k. We define the *k-descendant cone* of $\boldsymbol{x}$ to be the set of all k-descendants of $\boldsymbol{x}$ and denote this cone by $D_k^*(\boldsymbol{x})$. In other words,

$$D_k^*(\boldsymbol{x}) \triangleq \left\{ \boldsymbol{y} \in \Sigma_q^* \mid \boldsymbol{x} \overset{*}{\underset{k}{\Rightarrow}} \boldsymbol{y} \right\}.$$

Our work studies tandem duplications whose lengths are upper bounded by an integer k, and we extend the previous definitions. Formally, if a finite sequence of tandem duplications of length up to k is performed to obtain $\boldsymbol{y}$ from $\boldsymbol{x}$, then we say that $\boldsymbol{y}$ is a *$\leqslant k$-descendant* of $\boldsymbol{x}$ and denote this relation by $\boldsymbol{x} \overset{*}{\underset{\leqslant k}{\Longrightarrow}} y$. We have analogous definitions of *$\leqslant k$-descendant*, *$\leqslant k$-ancestor*, and *$\leqslant k$-descendant cone* $D^*_{\leqslant k}(\boldsymbol{x})$.

Example 2.1. Consider $\boldsymbol{x} = 01210$ over Σ_3. Since $T_{1,3}(\boldsymbol{x}) = 01211210$ and $T_{0,2}(01211210) = 0101211210$, we write $01210 \overset{*}{\underset{3}{\Rightarrow}} 01211210$ and $01210 \overset{*}{\underset{\leqslant 3}{\Longrightarrow}} 0101211210$. Alternatively, we have that $01211210 \in D_3^*(\boldsymbol{x})$ and $0101211210 \in D^*_{\leqslant 3}(\boldsymbol{x})$.

2.1 Problem Formulation

Motivated by applications that store data on living organisms, Jain et al. [7] looked at the $\leqslant k$-descendant cones of a pair of words and asked whether the two cones have a nontrivial intersection. Specifically, we introduce the notion of confusability.

Definition 2.2 (Confusability). Two words $\boldsymbol{x}$ and $\boldsymbol{y}$, are said to be *k-confusable* if $D_k^*(\boldsymbol{x}) \cap D_k^*(\boldsymbol{y}) \neq \varnothing$. Similarly, they are said to be *$\leqslant k$-confusable* if $D^*_{\leqslant k}(\boldsymbol{x}) \cap D^*_{\leqslant k}(\boldsymbol{y}) \neq \varnothing$.

To design error-correcting codes that store information in the DNA of living organisms, Jain et al. then proposed the use of codewords that are not pairwise confusable.

Definition 2.3 ($\leqslant k$-Tandem-Duplication Codes). A subset $C \subset \Sigma_q^n$ is a *$\leqslant k$-tandem-duplication code* if for all $\boldsymbol{x}, \boldsymbol{y} \in C$ and $\boldsymbol{x} \neq \boldsymbol{y}$, we have that $\boldsymbol{x}$ and $\boldsymbol{y}$ are not $\leqslant k$-confusable. We say that C is an $(n, \leqslant k; q)$-tandem-duplication code or $(n, \leqslant k; q)$-TD code.

Therefore, to determine if a set of words is a tandem-duplication code, we need to verify that all pairs of distinct words are not confusable. Hence, we state our problem of interest.

Confusability Problem
Instance: Two words $\boldsymbol{x}$ and $\boldsymbol{y}$ over Σ_q and an integer k
Question: Are $\boldsymbol{x}$ and $\boldsymbol{y}$ $\leqslant k$-confusable?

While the confusability problem is a natural question, efficient algorithms are only known for the case where $k \in \{1, 2\}$. We review these results in the next subsection.

2.2 Previous Work

Confusability. We summarize known efficient algorithms that determine whether two words are k-confusable for all k, and whether they are $\leqslant k$-confusable for $k \in \{1, 2\}$. We then highlight why these methods cannot be extended for the case $k \geqslant 3$. Crucial to the algorithms is the concept of irreducible words and roots.

Definition 2.4. A word $\boldsymbol{x}$ is said to be *k-irreducible* if $\boldsymbol{x}$ cannot be deduplicated into shorter words with deduplication of length k. In other words, if $\boldsymbol{y} \overset{*}{\underset{k}{\Rightarrow}} \boldsymbol{x}$, then $\boldsymbol{y} = \boldsymbol{x}$. The set of k-irreducible q-ary words is denoted by $\mathrm{Irr}_k(q)$ and the set of k-irreducible words of length n is denoted by $\mathrm{Irr}_k(n, q)$. The k-ancestors of $\boldsymbol{x} \in \Sigma_q^*$ that are k-irreducible words are called the *k-roots* of $\boldsymbol{x}$. The set of k-roots of $\boldsymbol{x}$ is denoted by $R_k(\boldsymbol{x})$. In a similar fashion, we have the definitions of *$\leqslant k$-irreducible* words and $\leqslant k$-roots of $\boldsymbol{x}$, and the notation $\mathrm{Irr}_{\leqslant k}(q)$, $\mathrm{Irr}_{\leqslant k}(n, q)$, and $R_{\leqslant k}(\boldsymbol{x})$.

Example 2.5. For alphabet Σ_3, we have $R_{\leqslant 3}(01012012) = \{012\}$, and $R_{\leqslant 2}(012012) = \{012012\}$. However, $R_{\leqslant 4}(012101212) = \{012, 0121012\}$.

As we see, it is possible for a word to have more than one root. Jain et al. [7] determined when a tandem duplication system has only one root, irrespective of the word, and provided efficient algorithms to compute this unique root.

Proposition 2.6 (Jain et al. [7]). *For any $\boldsymbol{x} \in \Sigma_q^*$, we have that $|R_k(\boldsymbol{x})| = 1$ for all k, and $|R_{\leqslant k}(\boldsymbol{x})| = 1$ for all $k \in \{1, 2, 3\}$. Furthermore, there exist linear-time algorithms to compute $R_k(\boldsymbol{x})$ for all k.*

One may easily derive linear-time algorithms to compute $R_{\leqslant k}(\boldsymbol{x})$ for all $k \in \{1, 2, 3\}$. For completeness, we formally describe these algorithms in Appendix A.

As it turns out, for certain cases, determining the confusability of two words is equivalent to computing the roots for the words.

Proposition 2.7 (Jain et al. [7]). *For all $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^*$, we have that*

(i) $\boldsymbol{x}$ and $\boldsymbol{y}$ are k-confusable if and only if $R_k(\boldsymbol{x}) = R_k(\boldsymbol{y})$ for all k;
(ii) $\boldsymbol{x}$ and $\boldsymbol{y}$ are $\leqslant k$-confusable if and only if $R_{\leqslant k}(\boldsymbol{x}) = R_{\leqslant k}(\boldsymbol{y})$ for $k \in \{1, 2\}$.
(iii) If $R_{\leqslant 3}(\boldsymbol{x}) \neq R_{\leqslant 3}(\boldsymbol{y})$, then $\boldsymbol{x}$ and $\boldsymbol{y}$ are not $\leqslant 3$-confusable.

In other words, when $k \in \{1, 2\}$, to determine $\boldsymbol{x}$ and $\boldsymbol{y}$ are $\leqslant k$-confusable, it is both necessary and sufficient to compute the $\leqslant k$-roots of $\boldsymbol{x}$ and $\boldsymbol{y}$. Therefore, applying Proposition 2.6 and the algorithms in Appendix A, we are able to determine whether two words are $\leqslant k$-confusable in linear time for $k \in \{1, 2\}$.

Unfortunately, when $k = 3$, it is no longer sufficient to compute the $\leqslant 3$-roots of $\boldsymbol{x}$ and $\boldsymbol{y}$. Specifically, even though $R_{\leqslant 3}(\boldsymbol{x}) = R_{\leqslant 3}(\mathbf{y})$, it is possible that $\boldsymbol{x}$ and $\boldsymbol{y}$ are not $\leqslant 3$-confusable. We illustrate this in the next example.

Example 2.8. Consider $\boldsymbol{x} = 012012$ and $\boldsymbol{y} = 011112$ over Σ_3. The words have the same root as $R_{\leqslant 3}(\boldsymbol{x}) = R_{\leqslant 3}(\mathbf{y}) = \{012\}$. However, $\boldsymbol{x}$ and $\boldsymbol{y}$ are not $\leqslant 3$-confusable, because any $\leqslant 3$-descendant of $\boldsymbol{x}$ has a 2 to the left of a 0, whereas any $\leqslant 3$-descendant of $\boldsymbol{y}$ does not.

Therefore, the next smallest open case is where $k = 3$. A polynomial-time algorithm is implied by the results of Leupold et al. [11] and Jain et al. [6].

Proposition 2.9 (Leupold et al. [11], Jain et al. [6]). *Let $k \in \{2, 3\}$. Let $\boldsymbol{x} \in \Sigma_q^*$. Then $D^*_{\leqslant k}(\boldsymbol{x})$ is a regular language. Furthermore, if $|\boldsymbol{x}| = m$, then the deterministic finite automaton that generates $D^*_{\leqslant k}(\boldsymbol{x})$ has $O(m)$ vertices and $O(m)$ edges. The number of vertices and edges is independent of the alphabet size.*

Therefore, for two words $\boldsymbol{x}$ and $\boldsymbol{y}$ with $|\boldsymbol{x}| = m$ and $|\boldsymbol{y}| = n$, the language $D^*_{\leqslant 3}(\boldsymbol{x}) \cap D^*_{\leqslant 3}(\boldsymbol{y})$ is regular and the corresponding finite automaton has $O(mn)$ vertices and $O(mn)$ edges (see for example, Sipser [16, Footnote 3, Theorem 1.25]). Hence, to determine if $\boldsymbol{x}$ and $\boldsymbol{y}$ are confusable, it suffices to determine if the language $D^*_{\leqslant 3}(\boldsymbol{x}) \cap D^*_{\leqslant 3}(\boldsymbol{y})$ is nonempty. The latter can be done in $O(mn)$ time. We improve this running time by providing an algorithm that runs in $O(\max\{m, n\})$ time.

Code Construction. We are interested in determining the maximum possible size of an $(n, \leqslant k; q)$-TD code. An $(n, \leqslant k; q)$-TD code that achieves this maximum is said to be *optimal*.

Motivated by Proposition 2.7, Jain et al. used irreducible words to construct tandem-duplication codes.

Construction 1. *For $k \in \{2, 3\}$ and $n \geqslant k$. An $(n, \leqslant k; q)$-TD code $C_I(n, \leqslant k; q)$ is given by*

$$C_I(n, \leqslant k; 3) = \bigcup_{i=1}^{n} \left\{ \xi_{n-i}(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathrm{Irr}_{\leqslant k}(i, q) \right\}.$$

Here, $\xi_i(\boldsymbol{x}) = \boldsymbol{x}z^i$, where z is the last symbol of $\boldsymbol{x}$. In other words, $\xi_i(\boldsymbol{x})$ is the sequence $\boldsymbol{x}$ with its last symbol z repeated i more times. Furthermore, the size of $C_I(n, \leqslant k; q)$ is $\sum_{i=1}^{n} \left|\mathrm{Irr}_{\leqslant k}(i, q)\right|$.

It then follows from Proposition 2.7 that $C_I(n, \leqslant 2; q)$ is an optimal $(n, \leqslant 2; q)$-TD code. Unfortunately, the TD code $C_I(n, \leqslant 3; q)$ is not an optimal for $n \geqslant 6$, and we illustrate this via the following example.

Example 2.10. Consider $n = 6$, $q = 3$. The code $C_I(6, \leqslant 3; 3)$ from Construction 1 is given by

$$\begin{aligned} C_I(6, \leqslant 3; q) = {} & \{aaaaaa \mid a \in \Sigma_3\} \cup \{abbbbb \mid a, b \in \Sigma_3, a \neq b\} \\ & \cup \{abaaaa, abcccc \mid a, b, c \in \Sigma_3, a \neq b, b \neq c, a \neq c\} \\ & \cup \{abaccc, abcaaa, abcbbb \mid a, b, c \in \Sigma_3, a \neq b, b \neq c, a \neq c\} \\ & \cup \{abacaa, abacbb, abcabb, abcacc, abcbaa \mid a, b, c \in \Sigma_3, a \neq b, b \neq c, a \neq c\} \\ & \cup \{abacab, abacba, abacbc, abcaba, abcacb, abcbab, abcbac \mid a, b, c \in \Sigma_3, a \neq b, b \neq c, a \neq c\}. \end{aligned}$$

Hence, $|C_I(6, \leqslant 3; 3)| = 111$. However, we may remove from $C_I(6, \leqslant 3; 3)$ the six codewords $\{abcccc \mid a, b, c \in \Sigma_3, a \neq b, b \neq c, a \neq c\}$ and augment the code with twelve more codewords $\{abcabc, abbbbc \mid a, b, c \in \Sigma_3, a \neq b, b \neq c, a \neq c\}$. Then, we can check that the new code has size 117, and we later verify using Proposition 2.11 that the new code is indeed optimal. For lengths at least six, we construct TD codes with strictly larger sizes.

Upper Bound. We also study upper bounds on the size of an optimal $(n, \leqslant 3; q)$-TD code. By definition, an $(n, \leqslant 3; q)$-TD code is also an $(n, \leqslant 2; q)$-TD code. Since an optimal $(n, \leqslant 2; q)$-TD code is provided by Construction 1, we have the following upper bound on the size of an optimal $(n, \leqslant 3; q)$-TD code.

Proposition 2.11. *The size of an $(n, \leqslant 3; q)$-TD code is at most $\sum_{i=1}^{n} \left|\mathrm{Irr}_{\leqslant 2}(i, q)\right|$.*

Proposition 2.11 implies that Construction 1 is tight for $k = 3$ and $n \leqslant 5$. Using a combinatorial characterization implied by our algorithm, we improve this upper bound for longer lengths in Section 4.

2.3 Our Contributions

- **Confusability**. In Section 3, we present sufficient and necessary conditions for two words to be $\leqslant 3$-confusable and propose a linear-time algorithm to solve the $\leqslant 3$-confusability problem.
- **Estimates on code sizes**. Using insights gained from Section 3, we study the size of tandem-duplication codes in Section 4. We first improve the upper bound given by Proposition 2.11 and then construct codes with larger sizes as compared to those given by Construction 1. We also provide certain explicit constructions and recursive constructions for tandem-duplication codes. Furthermore, we determine the sizes of optimal $\leqslant 3$-tandem-duplication codes for lengths up to 20.

3 DETERMINING $\leqslant 3$-CONFUSABILITY

We derive a linear-time algorithm to determine the $\leqslant 3$-confusability of two words $\boldsymbol{x}$ and $\boldsymbol{y}$. For the sake of simplicity, we omit the use of "$\leqslant 3$" and assume confusable, descendant, and root to mean $\leqslant 3$-confusable, $\leqslant 3$-descendant, and $\leqslant 3$-root, respectively.

Prior to stating the algorithm, we make some technical observations on tandem duplications of length at most three. The first lemma states that given a sequence of tandem duplications, we may reorder the tandem duplications such that the lengths of the repeats are nonincreasing.

LEMMA 3.1. *Let $k_1 < k_2 \leqslant 3$ and $\boldsymbol{x} \in \Sigma_q^*$. Suppose that $T_{i_2,k_2} \circ T_{i_1,k_1}(\boldsymbol{x}) = \boldsymbol{x}'$ for some i_1, i_2. Then there exist integers $3 \geqslant \ell_1 \geqslant \ell_2 \geqslant \cdots \geqslant \ell_t$ and $j_1, j_2, \ldots, j_t$ such that*

$$T_{j_t,\ell_t} \circ T_{j_{t-1},\ell_{t-1}} \circ \cdots \circ T_{j_1,\ell_1}(\boldsymbol{x}) = \boldsymbol{x}'.$$

PROOF. Consider a sequence of two tandem duplications $T_{i_2,k_2} \circ T_{i_1,k_1}$ with $k_1 < k_2 \leqslant 3$. We replace the sequence of duplications according to the following rules.

- Suppose that $k_1 = 1$ and $k_2 = 3$.
  (a) If $i_2 \leqslant i_1 - 2$, then substitute with $T_{i_1+3,1} \circ T_{i_2,3}$.
  (b) If $i_2 = i_1 - 1$, then substitute with $T_{i_1+3,1} \circ T_{i_1,1} \circ T_{i_1-1,2}$.
  (c) If $i_2 = i_1$, then substitute with $T_{i_1+3,1} \circ T_{i_1,1} \circ T_{i_1,2}$.
  (d) If $i_2 \geqslant i_1 + 1$, then substitute with $T_{i_1,1} \circ T_{i_2-1,3}$.
- Suppose that $k_1 = 2$ and $k_2 = 3$.
  (a) If $i_2 \leqslant i_1 - 1$, then substitute with $T_{i_1+3,2} \circ T_{i_2,3}$.
  (b) If $i_2 = i_1$, then substitute with $T_{i_2,1} \circ T_{i_1,2} \circ T_{i_1,2}$.
  (c) If $i_2 = i_1 + 1$, then substitute with $T_{i_1+3,1} \circ T_{i_1,2} \circ T_{i_1,2}$.
  (d) If $i_2 \geqslant i_1 + 2$, then substitute with $T_{i_1,2} \circ T_{i_2-2,3}$.
- Suppose that $k_1 = 1$ and $k_2 = 2$.
  (a) If $i_2 \leqslant i_1 - 1$, then substitute with $T_{i_1+2,1} \circ T_{i_2,2}$.
  (b) If $i_2 = i_1$, then substitute with $T_{i_1,1} \circ T_{i_1,1} \circ T_{i_1,1}$.
  (c) If $i_2 \geqslant i_1 + 1$, then substitute with $T_{i_1,1} \circ T_{i_2-1,2}$.

Thus, given a sequence of tandem duplications, we may reorder the tandem duplications such that the lengths of the repeats are non-increasing. ■

The next lemma states that we may assume the duplications of length two and three are performed on segments whose symbols are distinct. This is because if a duplication is performed on a segment whose symbols are not distinct, then we may find an equivalent sequence of duplications of strictly shorter lengths.

LEMMA 3.2. *Let $k \in \{2, 3\}$. Suppose $T_{i,k}(\boldsymbol{x}) = \boldsymbol{uvvw}$, where $\boldsymbol{x} = \boldsymbol{uvw}$, $|\boldsymbol{u}| = i$ and $|\boldsymbol{v}| = k$. If the symbols in $\boldsymbol{v}$ are not pairwise distinct, then there exist integers $k > \ell_1 \geqslant \ell_2$ and j_1, j_2 such that*

$$T_{j_2,\ell_2} \circ T_{j_1,\ell_1}(\boldsymbol{x}) = T_{i,k}(\boldsymbol{x}).$$

PROOF. When $k = 3$, we consider the following three cases.

- Duplication of *aba* to *abaaba* is equivalent to $T_{2,1} \circ T_{1,2}(aba) = T_{2,1}(ababa) = abaaba$.
- Duplication of *aab* to *aabaab* is equivalent to $T_{3,1} \circ T_{1,2}(aab) = T_{3,1}(aabab) = aabaab$.
- Duplication of *abb* to *abbabb* is equivalent to $T_{1,1} \circ T_{0,2}(abb) = T_{1,1}(ababb) = abbaab$.
- Duplication of *aaa* to *aaaaaa* is equivalent to $T_{0,1} \circ T_{0,2}(aaa) = T_{0,1}(aaaaa) = aaaaaa$.

When $k = 2$, we consider the duplication of length two on aa to obtain $aaaa$. This is equivalent to $T_{0,1} \circ T_{0,1}(aa) = T_{0,1}(aaa) = aaaa$. Therefore, if a duplication is performed on a segment whose symbols are not distinct, then we may find an equivalent sequence of duplications of strictly shorter lengths. ■

Henceforth, we use the notation $\boldsymbol{x} \overset{*}{\underset{k_d}{\Longrightarrow}} \boldsymbol{y}$ to denote that there is a sequence of tandem duplications of length k of *distinct* symbols from $\boldsymbol{x}$ to $\boldsymbol{y}$. It follows from definition that $\boldsymbol{x} \overset{*}{\underset{1}{\Rightarrow}} \boldsymbol{y}$ is equivalent to $\boldsymbol{x} \overset{*}{\underset{1_d}{\Longrightarrow}} \boldsymbol{y}$. The next lemma is immediate from Lemmas 3.1 and 3.2.

Lemma 3.3. *Suppose that* $\boldsymbol{x} \overset{*}{\underset{\leqslant 3}{\Longrightarrow}} \boldsymbol{x}'$. *Then there exist* $\boldsymbol{x}_1$ *and* $\boldsymbol{x}_2$ *such that* $\boldsymbol{x} \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{x}_2 \overset{*}{\underset{2_d}{\Longrightarrow}} \boldsymbol{x}_1 \overset{*}{\underset{1_d}{\Longrightarrow}} \boldsymbol{x}'$. *Furthermore,* $\boldsymbol{x}_1$ *and* $\boldsymbol{x}_2$ *are uniquely determined by* $\boldsymbol{x}_1 = R_1(\boldsymbol{x}')$ *and* $\boldsymbol{x}_2 = R_{\leqslant 2}(\boldsymbol{x}')$.

Equipped with this lemma, we state the following theorem that provides a necessary and sufficiency condition for two words to be confusable.

Theorem 3.4. *Two words* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *are* $\leqslant 3$*-confusable if and only if there exist* $\boldsymbol{x}'$ *and* $\boldsymbol{y}'$ *such that*

$$\boldsymbol{x} \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{x}', \ \boldsymbol{y} \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{y}' \text{ and } R_{\leqslant 2}(\boldsymbol{x}') = R_{\leqslant 2}(\boldsymbol{y}').$$

Proof. Suppose that $\boldsymbol{x}$ and $\boldsymbol{y}$ are confusable. Then there exists z such that $\boldsymbol{x} \overset{*}{\underset{\leqslant 3}{\Longrightarrow}} z$ and $\boldsymbol{y} \overset{*}{\underset{\leqslant 3}{\Longrightarrow}} z$. By Lemma 3.3, there exist $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{y}_1, \boldsymbol{y}_2$ such that $\boldsymbol{x} \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{x}_2 \overset{*}{\underset{2_d}{\Longrightarrow}} \boldsymbol{x}_1 \overset{*}{\underset{1_d}{\Longrightarrow}} z$ and $\boldsymbol{y} \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{y}_2 \overset{*}{\underset{2_d}{\Longrightarrow}} \boldsymbol{y}_1 \overset{*}{\underset{1_d}{\Longrightarrow}} z$. Hence, we set $\boldsymbol{x}' = \boldsymbol{x}_2$ and $\boldsymbol{y}' = \boldsymbol{y}_2$. Note that by Proposition 2.7, $R_{\leqslant 2}(\boldsymbol{x}') = R_{\leqslant 2}(\boldsymbol{y}')$, since $\boldsymbol{x}'$ and $\boldsymbol{y}'$ are $\leqslant 2$-confusable.

Conversely, suppose that there exist $\boldsymbol{x}'$ and $\boldsymbol{y}'$ such that $\boldsymbol{x} \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{x}'$, $\boldsymbol{y} \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{y}'$, and $R_{\leqslant 2}(\boldsymbol{x}') = R_{\leqslant 2}(\boldsymbol{y}')$. Since $\boldsymbol{x}'$ and $\boldsymbol{y}'$ have a common $\leqslant 2$-root, it follows from Proposition 2.7 that $\boldsymbol{x}'$ and $\boldsymbol{y}'$ are $\leqslant 2$-confusable. Thus, $\boldsymbol{x}$ and $\boldsymbol{y}$ are confusable. ■

In conclusion, for two words $\boldsymbol{x}$ and $\boldsymbol{y}$ to be confusable, it is equivalent to checking if there are tandem duplications of length three that make their descendants $\boldsymbol{x}'$ and $\boldsymbol{y}'$ have the same $\leqslant 2$-root. We make use of this important fact in the next section.

3.1 Strategy Behind Algorithm

We outline our strategy to determine the confusability of two words $\boldsymbol{x}$ and $\boldsymbol{y}$. Recall from Proposition 2.7(iii), if $\boldsymbol{x}$ and $\boldsymbol{y}$ have different roots, then they are not confusable. Hence, our first step is to determine $R_{\leqslant 3}(\boldsymbol{x})$ and $R_{\leqslant 3}(\boldsymbol{y})$ and see if the roots are equal. We can do this in linear time using Algorithm 2. If the roots are not equal, then we immediately conclude that $\boldsymbol{x}$ and $\boldsymbol{y}$ are not confusable. Therefore, the nontrivial task is to determine confusability when $R_{\leqslant 3}(\boldsymbol{x}) = R_{\leqslant 3}(\boldsymbol{y})$. Henceforth, we assume that the *common root of* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *is* $\boldsymbol{r}$.

Suppose that $\boldsymbol{r}$ contains at most two distinct symbols. Then from Lemma 3.3, we have that $\boldsymbol{r} \overset{*}{\underset{\leqslant 2}{\Longrightarrow}} \boldsymbol{x}$ and $\boldsymbol{r} \overset{*}{\underset{\leqslant 2}{\Longrightarrow}} \boldsymbol{y}$. In other words, $\boldsymbol{r}$ is also the $\leqslant 2$-root of both $\boldsymbol{x}$ and $\boldsymbol{y}$. Applying Proposition 2.7, we have that $\boldsymbol{x}$ and $\boldsymbol{y}$ are $\leqslant 2$-confusable and so, $\leqslant 3$-confusable. Therefore, it remains to consider the case where $\boldsymbol{r}$ contains *at least three distinct* symbols.

Given $\boldsymbol{r}$, $\boldsymbol{x}$ and $\boldsymbol{y}$, our next step is to compute certain proper prefixes $*\text{Pref}(\boldsymbol{r}, \boldsymbol{x})$ and $*\text{Pref}(\boldsymbol{r}, \boldsymbol{y})$ of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, such that *under certain conditions* the following holds:

($\boldsymbol{x}$ and $\boldsymbol{y}$ are confusable) if and only if ($\boldsymbol{x} \setminus *\text{Pref}(\boldsymbol{r}, \boldsymbol{x})$ and $\boldsymbol{y} \setminus *\text{Pref}(\boldsymbol{r}, \boldsymbol{y})$ are confusable).

Here, $\boldsymbol{x} \setminus \boldsymbol{z}$ denotes the word we obtain by removing the prefix $\boldsymbol{z}$ from $\boldsymbol{x}$. Since the words on the righthand side are strictly shorter than the words on the lefthand side, we may repeat this process for the shorter words until the common root of the words has less than three distinct symbols.

Before we formally define the prefixes $*\text{Pref}(\boldsymbol{r}, \boldsymbol{x})$ and $*\text{Pref}(\boldsymbol{r}, \boldsymbol{y})$, we provide certain intuition behind our technical definitions via an example.

Example 3.5. Let $\boldsymbol{x} = 01102021020120111$ and $\boldsymbol{y} = 00101002110200120201$. Since both $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same $\leqslant 3$-root $\boldsymbol{r} \triangleq 010201$, it is possible that $\boldsymbol{x}$ and $\boldsymbol{y}$ are $\leqslant$ 3-confusable.

To do so, we make some observations on the root $\boldsymbol{r}$ and look at how we obtain $\boldsymbol{x}$ and $\boldsymbol{y}$ from $\boldsymbol{r}$. Now, $\boldsymbol{r}$ has two 3-substrings that contain three distinct symbols: namely, 102 and 201, and so, we divide $\boldsymbol{r}$ two *overlapping* substrings $\boldsymbol{r}_1 = 0102$ and $\boldsymbol{r}_2 = 0201$.

We can also similarly divide each of $\boldsymbol{x}$ and $\boldsymbol{y}$ into two overlapping substrings. Namely,

$$\boldsymbol{x}_1 = 0110202102, \qquad \boldsymbol{x}_2 = 020120111,$$
$$\boldsymbol{y}_1 = 001010021102, \qquad \boldsymbol{y}_2 = 0200120201.$$

We can easily check that $\boldsymbol{r}_1 \overset{*}{\underset{\leqslant 3}{\Longrightarrow}} \boldsymbol{x}_1, \boldsymbol{r}_1 \overset{*}{\underset{\leqslant 3}{\Longrightarrow}} \boldsymbol{y}_1, \boldsymbol{r}_2 \overset{*}{\underset{\leqslant 3}{\Longrightarrow}} \boldsymbol{x}_2$, and $\boldsymbol{r}_2 \overset{*}{\underset{\leqslant 3}{\Longrightarrow}} \boldsymbol{y}_2$. In what follows, we prove that $\boldsymbol{x}$ and $\boldsymbol{y}$ are $\leqslant 3$-confusable if and only if $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ are $\leqslant 3$-confusable for $i \in \{1, 2\}$. Furthermore, for $i \in \{1, 2\}$, we define a combinatorial property of $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ that allows one to easily determine that $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ are $\leqslant$ 3-confusable.

Before this, we describe rules on how to break $\boldsymbol{r}$ into the *regions*: $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$.

We now provide the formal definitions. Assume that $\boldsymbol{r}$ contains at least three distinct symbols. Since $\boldsymbol{r} \in \text{Irr}_{\leqslant 3}(q)$, we easily check that either $r_1r_2r_3$ or $r_2r_3r_4$ is a substring with three distinct symbols. Define $\text{main}(\boldsymbol{r})$ to be the first substring in $\boldsymbol{r}$ with three distinct symbols and we define the *first region of* $\boldsymbol{r}$, denoted by $\text{Reg}(\boldsymbol{r})$, to be a certain prefix of $\boldsymbol{r}$ according to the following rule.

If $r_1 = r_3$, then we set

$$\text{main}(\boldsymbol{r}) \triangleq r_2r_3r_4, \text{ and } \text{Reg}(\boldsymbol{r}) \triangleq \begin{cases} r_1r_2r_3r_4 = r_1r_2r_1r_4, & \text{if } r_2 \neq r_5, \\ r_1r_2r_3r_4r_5 = r_1r_2r_1r_4r_2, & \text{if } r_2 = r_5, r_3 \neq r_6, \\ r_1r_2r_3r_4r_5r_6 = r_1r_2r_1r_4r_2r_1, & \text{otherwise.} \end{cases}$$

If $r_1 \neq r_3$, then we set

$$\text{main}(\boldsymbol{r}) \triangleq r_1r_2r_3, \text{ and } \text{Reg}(\boldsymbol{r}) \triangleq \begin{cases} r_1r_2r_3, & \text{if } r_1 \neq r_4, \\ r_1r_2r_3r_4 = r_1r_2r_3r_1, & \text{if } r_1 = r_4, r_2 \neq r_5, \\ r_1r_2r_3r_4r_5 = r_1r_2r_3r_1r_2, & \text{otherwise.} \end{cases}$$

Intuitively, the first region $\text{Reg}(\boldsymbol{r})$ of $\boldsymbol{r}$ is defined as above so that the following properties of the region, or Lemma 3.6 hold.

Lemma 3.6. *Suppose the root* $\boldsymbol{r}$ *has three distinct symbols. Define* $\text{Reg}(\boldsymbol{r})$ *as above. Then* $\text{Reg}(\boldsymbol{r})$ *can be written as* $\boldsymbol{w}(abc)^\ell ab$*, where* a, b, c *are distinct symbols,* $\ell \in \{0, 1\}$ *and* $\boldsymbol{w}$ *is prefix of length at most three over the alphabet* $\{a, b, c\}$*. Furthermore, if* $\text{Reg}(\boldsymbol{r}) \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{z}$*, then* $\boldsymbol{z}$ *can be written as* $\boldsymbol{w}(abc)^m ab$ *with* $m \geqslant \ell$*. Also, the symbol in* $\boldsymbol{r}$ *after the prefix* $\text{Reg}(\boldsymbol{r})$ *is not* c*.*

Example 3.7. Suppose that $\boldsymbol{r} = 010201$. Then $\text{main}(\boldsymbol{r}) = 102$ and $\text{Reg}(\boldsymbol{r}) = 0102$. Indeed, we can write $\text{Reg}(\boldsymbol{r})$ as $\boldsymbol{w}(abc)^\ell ab$ with $\boldsymbol{w} = 01$, $\ell = 0$, and $abc = 021$. Note that abc is not necessarily $\text{main}(\boldsymbol{r})$. We also check that the next symbol after $\text{Reg}(\boldsymbol{r})$ in $\boldsymbol{r}$ is 0, which is not $c = 1$.

Remark 1. Given a root $\boldsymbol{r}$, we provide the rule to determine the *first region*. The subsequent regions of $\boldsymbol{r}$ are then recursively defined by looking at the "first region" of a certain suffix of $\boldsymbol{r}$. Details are provided in Algorithm 1.

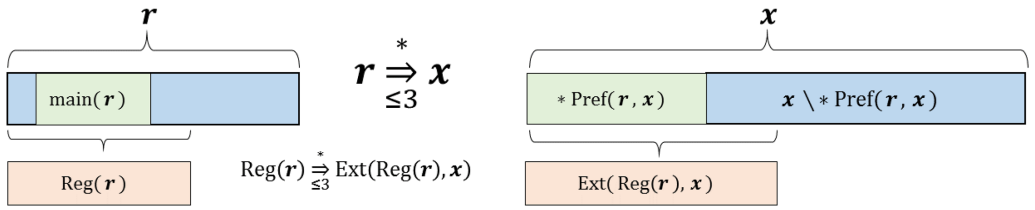

Fig. 1. Prefixes used to determine confusability.

Given $\mathrm{Reg}(\boldsymbol{r}) = \boldsymbol{w}(abc)^{\ell}ab$, we consider the longest prefix of $\boldsymbol{x}$ that is generated from $\mathrm{Reg}(\boldsymbol{r})$ through a sequence of tandem duplications of length at most three. We call this prefix the *first extended region of* $\boldsymbol{r}$ *in* $\boldsymbol{x}$ and denote it with $\mathrm{Ext}(\mathrm{Reg}(\boldsymbol{r}), \boldsymbol{x})$. This prefix can be computed deterministically using the finite automaton that defines $D^{*}_{\leqslant 3}(\mathrm{Reg}(\boldsymbol{r}))$ (see Subsection 3.2 for details).

Finally, we define our special prefix $*\mathrm{Pref}(\boldsymbol{r}, \boldsymbol{x})$. Suppose that $\mathrm{Ext}(\mathrm{Reg}(\boldsymbol{r}), \boldsymbol{x}) = p_1p_2 \cdots p_t = \boldsymbol{p}$ and p_i is the last appearance of a in $\boldsymbol{p}$, where a is the second last symbol of $\mathrm{Reg}(\boldsymbol{r})$. The *special prefix* $*\mathrm{Pref}(\boldsymbol{r}, \boldsymbol{x})$ is given by $p_1p_2 \cdots p_{i-1}$.

Example 3.8. Let $\boldsymbol{r} = 010201$ and $\boldsymbol{x} = 01102021020120111$. Recall that $\mathrm{Reg}(\boldsymbol{r}) = 0102$ and $a = 0$. Then,

$$\mathrm{Ext}(0102, \boldsymbol{x}) = 0110202102 \text{ and } *\mathrm{Pref}(\boldsymbol{r}, \boldsymbol{x}) = 01102021.$$

Remark 2. Consider a word $\boldsymbol{x}$ whose root is $\boldsymbol{r}$. Similar to the definition of the first region of $\boldsymbol{r}$, we only provide the rule to determine the *first extended region of* $\boldsymbol{r}$ *in* $\boldsymbol{x}$. The subsequent extended regions of $\boldsymbol{r}$ in $\boldsymbol{x}$ are then recursively defined by looking at the "first extended region" of a certain suffix of $\boldsymbol{r}$ in a certain suffix of $\boldsymbol{x}$. As before, details are provided in Algorithm 1.

We summarize the relationships of $\boldsymbol{r}$, $\boldsymbol{x}$, $\mathrm{Reg}(\boldsymbol{r})$, $\mathrm{Ext}(\mathrm{Reg}(\boldsymbol{r}), \boldsymbol{x})$, and $*\mathrm{Pref}(\boldsymbol{r}, \boldsymbol{x})$ in Figure 1, and we are ready to state the main characterization theorem.

Theorem 3.9. *Suppose that* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *are two words such that* $\boldsymbol{r} = R_{\leqslant 3}(\boldsymbol{x}) = R_{\leqslant 3}(\boldsymbol{y})$ *and* $\boldsymbol{r}$ *contains at least three distinct symbols. Set*

$$\boldsymbol{p} = \mathrm{Ext}(\mathrm{Reg}(\boldsymbol{r}), \boldsymbol{x}), \qquad \boldsymbol{q} = \mathrm{Ext}(\mathrm{Reg}(\boldsymbol{r}), \boldsymbol{y}),$$
$$\boldsymbol{u} = \boldsymbol{x} \setminus *\mathrm{Pref}(\boldsymbol{r}, \boldsymbol{x}), \qquad \boldsymbol{v} = \boldsymbol{y} \setminus *\mathrm{Pref}(\boldsymbol{r}, \boldsymbol{y}).$$

Then $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *are confusable if and only if* $\boldsymbol{p}$ *and* $\boldsymbol{q}$ *are confusable, and* $\boldsymbol{u}$ *and* $\boldsymbol{v}$ *are confusable.*

Proof. From Lemma 3.6, we may set $\mathrm{Reg}(\boldsymbol{r}) = \boldsymbol{w}(abc)^{\ell}ab$. Since $\mathrm{Reg}(\boldsymbol{r}) \overset{*}{\underset{\leqslant 3}{\Longrightarrow}} \boldsymbol{p}$, Lemma 3.3 implies that $\mathrm{Reg}(\boldsymbol{r}) \overset{*}{\underset{3_d}{\Longrightarrow}} R_{\leqslant 2}(\boldsymbol{p})$. So, $R_{\leqslant 2}(\boldsymbol{p}) = \boldsymbol{w}(abc)^{m}ab$ for some $m \geqslant \ell$, and $R_{\leqslant 2}(\boldsymbol{q}) = \boldsymbol{w}(abc)^{j}ab$ for some $j \geqslant \ell$.

Suppose that $\boldsymbol{x}$ and $\boldsymbol{y}$ are confusable. By Theorem 3.4, there exist 3_d-descendants of $\boldsymbol{x}$ and $\boldsymbol{y}$ namely x' and y', such that $R_{\leqslant 2}(\boldsymbol{x}') = R_{\leqslant 2}(\boldsymbol{y}')$. Set

$$\boldsymbol{p}' = \mathrm{Ext}(\mathrm{Reg}(\boldsymbol{r}), \boldsymbol{x}'), \qquad \boldsymbol{q}' = \mathrm{Ext}(\mathrm{Reg}(\boldsymbol{r}), \boldsymbol{y}'),$$
$$\boldsymbol{u}' = \boldsymbol{x}' \setminus *\mathrm{Pref}(\boldsymbol{r}, \boldsymbol{x}'), \qquad \boldsymbol{v}' = \boldsymbol{y}' \setminus *\mathrm{Pref}(\boldsymbol{r}, \boldsymbol{y}').$$

Since the overlapping substring between $\boldsymbol{p}$ and $\boldsymbol{u}$ must be of the form $ab \cdots b$, each duplication of distinct triplets from $\boldsymbol{x}$ to $\boldsymbol{x}'$ must be either entirely in $\boldsymbol{p}$ or in $\boldsymbol{u}$. In other words, $\boldsymbol{p} \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{p}'$ and $\boldsymbol{u} \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{u}'$, and likewise $\boldsymbol{q} \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{q}'$ and $\boldsymbol{v} \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{v}'$. Looking at the overlapping substring between $\boldsymbol{p}'$

and $\boldsymbol{u}'$, we can observe that $R_{\leqslant 2}(\boldsymbol{x}') = R_{\leqslant 2}(\boldsymbol{p}'\boldsymbol{u}')$. Then $\mathrm{Reg}(\boldsymbol{r}) \overset{*}{\underset{\leqslant 3}{\Longrightarrow}} \boldsymbol{p}'$ and Lemma 3.3 imply that $\mathrm{Reg}(\boldsymbol{r}) \overset{*}{\underset{3_d}{\Longrightarrow}} R_{\leqslant 2}(\boldsymbol{p})$ and so, $R_{\leqslant 2}(\boldsymbol{p}') = \boldsymbol{w}(abc)^i ab$ for some $i \geqslant m$.

Next, $\boldsymbol{w}(abc)^i R_{\leqslant 2}(\boldsymbol{u}')$ is a $\leqslant 2$-irreducible word, because $R_{\leqslant 2}(\boldsymbol{u}')$ starts with ab and by definition, $\boldsymbol{w}(abc)^i$ ends with c but not cac. Therefore, $\boldsymbol{w}(abc)^i R_{\leqslant 2}(\boldsymbol{u}')$ is the $\leqslant 2$-root of $\boldsymbol{x}'$, or, $R_{\leqslant 2}(\boldsymbol{x}') = \boldsymbol{w}(abc)^i R_{\leqslant 2}(\boldsymbol{u}')$ and likewise $R_{\leqslant 2}(\boldsymbol{y}') = \boldsymbol{w}(abc)^h R_{\leqslant 2}(\boldsymbol{v}')$ for some $h \geqslant j$.

Since the first symbol of $R_{\leqslant 2}(\boldsymbol{x}' \setminus \boldsymbol{p}')$ and $R_{\leqslant 2}(\boldsymbol{y}' \setminus \boldsymbol{q}')$ cannot be c, neither $R_{\leqslant 2}(\boldsymbol{u}')$ nor $R_{\leqslant 2}(\boldsymbol{v}')$ start with abc. Therefore, since $R_{\leqslant 2}(\boldsymbol{x}') = R_{\leqslant 2}(\boldsymbol{y}')$, we can deduce that $h = i$ and $R_{\leqslant 2}(\boldsymbol{u}') = R_{\leqslant 2}(\boldsymbol{v}')$. Since $h = i$, we have $R_{\leqslant 2}(\boldsymbol{p}') = R_{\leqslant 2}(\boldsymbol{q}')$. Together with $\boldsymbol{p} \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{p}'$, $\boldsymbol{q} \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{q}'$, by Theorem 3.4, we can have $\boldsymbol{p}$ and $\boldsymbol{q}$ are confusable. Since $\boldsymbol{u} \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{u}'$, $\boldsymbol{v} \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{v}'$, and $R_{\leqslant 2}(\boldsymbol{u}') = R_{\leqslant 2}(\boldsymbol{v}')$, we can also conclude by Theorem 3.4 that $\boldsymbol{u}$ and $\boldsymbol{v}$ are confusable.

Suppose that $\boldsymbol{p}$ and $\boldsymbol{q}$ are confusable, and $\boldsymbol{u}$ and $\boldsymbol{v}$ are confusable. Hence, $\boldsymbol{pu}$ and $\boldsymbol{qv}$ are confusable. Note that $\boldsymbol{pu}$ is a descendant of $\boldsymbol{x}$ and $\boldsymbol{qv}$ is a descendant of $\boldsymbol{y}$, hence $\boldsymbol{x}$ and $\boldsymbol{y}$ are also confusable. ■

We can efficiently determine whether the prefixes $\boldsymbol{p}$ and $\boldsymbol{q}$ are confusable. When the substring $\mathrm{main}(\boldsymbol{r})$ appears the same number of times in $R_{\leqslant 2}(\boldsymbol{p})$ as in $R_{\leqslant 2}(\boldsymbol{q})$, we may assume that $R_{\leqslant 2}(\boldsymbol{p}) = \boldsymbol{w}(abc)^m ab = R_{\leqslant 2}(\boldsymbol{q})$. Hence, $\boldsymbol{p}$ and $\boldsymbol{q}$ are confusable by Proposition 2.7. In particular, we have Corollary 3.10.

Consider $\boldsymbol{z} = abc$, where a, b, c are three distinct symbols. Let $\mathrm{Count}(\boldsymbol{z}, \boldsymbol{x})$ denote the number of times $\boldsymbol{z}$ appears as a substring in $\boldsymbol{x}$, and $\mathrm{Count}(\mathrm{rot}(\boldsymbol{z}), \boldsymbol{x})$ denote the number of times abc, bca, or cab appears as a substring in $\boldsymbol{x}$.

Corollary 3.10. *Suppose that $\boldsymbol{x}$ and $\boldsymbol{y}$ are two words such that $\boldsymbol{r} = R_{\leqslant 3}(\boldsymbol{x}) = R_{\leqslant 3}(\boldsymbol{y})$ and $\boldsymbol{r}$ contains at least three distinct symbols. Set $\boldsymbol{p} = \mathrm{Ext}(\mathrm{Reg}(\boldsymbol{r}), \boldsymbol{x})$ and $\boldsymbol{q} = \mathrm{Ext}(\mathrm{Reg}(\boldsymbol{r}), \boldsymbol{y})$. Suppose further that $\mathrm{Count}(\mathrm{main}(\boldsymbol{r}), R_{\leqslant 2}(\boldsymbol{p})) = \mathrm{Count}(\mathrm{main}(\boldsymbol{r}), R_{\leqslant 2}(\boldsymbol{q}))$. Then $\boldsymbol{x}$ and $\boldsymbol{y}$ are confusable if and only if $\boldsymbol{x} \setminus *\mathrm{Pref}(\boldsymbol{r}, \boldsymbol{x})$ and $\boldsymbol{y} \setminus *\mathrm{Pref}(\boldsymbol{r}, \boldsymbol{y})$ are confusable.*

Example 3.11. Let $\boldsymbol{r} = 010201$, $\boldsymbol{x} = 01102021020120111$, and $\boldsymbol{y} = 00101002110200120201$. Note that $R_{\leqslant 3}(\boldsymbol{x}) = R_{\leqslant 3}(\boldsymbol{y}) = \boldsymbol{r}$. Recall that $\mathrm{main}(\boldsymbol{r}) = 102, \mathrm{Reg}(\boldsymbol{r}) = 0102$ and $a = 0$. Then,

$$\boldsymbol{p} = \mathrm{Ext}(0102, \boldsymbol{x}) = 0110202102, \quad *\mathrm{Pref}(\boldsymbol{r}, \boldsymbol{x}) = 01102021, \quad \boldsymbol{u} = \boldsymbol{x} \setminus *\mathrm{Pref}(\boldsymbol{r}, \boldsymbol{x}) = 020120111,$$
$$\boldsymbol{q} = \mathrm{Ext}(0102, \boldsymbol{y}) = 001010021102, \quad *\mathrm{Pref}(\boldsymbol{r}, \boldsymbol{y}) = 0010100211, \quad \boldsymbol{v} = \boldsymbol{y} \setminus *\mathrm{Pref}(\boldsymbol{r}, \boldsymbol{y}) = 0200120201.$$

Note that $R_{\leqslant 2}(\boldsymbol{p}) = 0102102 = R_{\leqslant 2}(\boldsymbol{q})$, and $\mathrm{Count}(\mathrm{main}(\boldsymbol{r}), R_{\leqslant 2}(\boldsymbol{p})) = \mathrm{Count}(\mathrm{main}(\boldsymbol{r}), R_{\leqslant 2}(\boldsymbol{q})) = \mathrm{Count}(102, 0102102) = 2$. Therefore, by Corollary 3.10, we have $\boldsymbol{x}$ and $\boldsymbol{y}$ are confusable if and only if $\boldsymbol{u}$ and $\boldsymbol{v}$ are confusable. Since $R_{\leqslant 2}(\boldsymbol{u}) = 0201201 = R_{\leqslant 2}(\boldsymbol{v})$, Proposition 2.7 implies that $\boldsymbol{u}$ and $\boldsymbol{v}$ are confusable. Therefore, $\boldsymbol{x}$ and $\boldsymbol{y}$ are confusable.

However, when the number of appearances is not equal, we may modify the proof to obtain the following corollary.

Corollary 3.12. *Suppose that $\boldsymbol{x}$ and $\boldsymbol{y}$ are two words such that $\boldsymbol{r} = R_{\leqslant 3}(\boldsymbol{x}) = R_{\leqslant 3}(\boldsymbol{y})$ and $\boldsymbol{r}$ contains at least three distinct symbols. Set $\boldsymbol{p} = \mathrm{Ext}(\mathrm{Reg}(\boldsymbol{r}), \boldsymbol{x})$ and $\boldsymbol{q} = \mathrm{Ext}(\mathrm{Reg}(\boldsymbol{r}), \boldsymbol{y})$. Suppose further that $\mathrm{Count}(\mathrm{main}(\boldsymbol{r}), R_{\leqslant 2}(\boldsymbol{p})) < \mathrm{Count}(\mathrm{main}(\boldsymbol{r}), R_{\leqslant 2}(\boldsymbol{q}))$. Then $\boldsymbol{x}$ and $\boldsymbol{y}$ are confusable if and only if $\mathrm{Count}(\mathrm{rot}(\mathrm{main}(\boldsymbol{r})), \boldsymbol{p}) > 0$ and $\boldsymbol{x} \setminus *\mathrm{Pref}(\boldsymbol{r}, \boldsymbol{x})$ and $\boldsymbol{y} \setminus *\mathrm{Pref}(\boldsymbol{r}, \boldsymbol{y})$ are confusable.*

Proof. By Theorem 3.9, it is sufficient to show that $\mathrm{Count}(\mathrm{rot}(\mathrm{main}(\boldsymbol{r})), \boldsymbol{p}) > 0$ iff $\boldsymbol{p}$ and $\boldsymbol{q}$ are confusable. Let $R_{\leqslant 2}(\boldsymbol{p}) = \boldsymbol{w}(abc)^k ab$ and $R_{\leqslant 2}(\boldsymbol{q}) = \boldsymbol{w}(abc)^j ab$ for some $k < j$.

ALGORITHM 1: CONFUSE($\boldsymbol{x}, \boldsymbol{y}$)

Input: $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^*$
Output: `true` if $\boldsymbol{x}$ and $\boldsymbol{y}$ are confusable, `false` otherwise.

```
1: r ← R_{⩽3}(x)
2: r' ← R_{⩽3}(y)
3: if r ≠ r' then
4:     return false
5: else
6:     return Confuse-Recursion(x, y, r)

7: procedure Confuse-Recursion(x,y,r)
8:     if r has at most two distinct symbols then
9:         return true
10:    else
11:        Compute main(r) and Reg(r)
12:        p ← Ext(Reg(r), x),                    q ← Ext(Reg(r), y)
13:        u ← x \ ∗Pref(r, x),                   v ← y \ ∗Pref(r, y)
14:        C_p ← Count(main(r), R_{⩽2}(, p)),   C_q ← Count(main(r), R_{⩽2}(q))
15:        r* ← r \ w(abc)^ℓ, where Reg(r) = w(abc)^ℓ ab
16:        if C_p = C_q then
17:            return Confuse-Recursion(u, v, r*)
18:        else if C_p < C_q and Count(rot(main(r)), p) > 0 then
19:            return Confuse-Recursion(u, v, r*)
20:        else if C_p < C_q and Count(rot(main(r)), q) > 0 then
21:            return Confuse-Recursion(u, v, r*)
22:        else
23:            return false
```

Suppose there exists at least one distinct triplet in $\boldsymbol{p}$, we set $\boldsymbol{p}'$ to be the result of duplication of the distinct triplet in $\boldsymbol{p}$ as many as $j-k$ times and $\boldsymbol{q}'$ to be $\boldsymbol{q}$. Then $R_{\leqslant 2}(\boldsymbol{p}') = \boldsymbol{w}(abc)^j ab = R_{\leqslant 2}(\boldsymbol{q}')$, and hence by Theorem 3.4, $\boldsymbol{p}$ and $\boldsymbol{q}$ are confusable.

Suppose, however, $\boldsymbol{p}$ and $\boldsymbol{q}$ are confusable. By Theorem 3.4, there exist $\boldsymbol{p}'$ and $\boldsymbol{q}'$ such that $\boldsymbol{p} \overset{*}{\underset{3d}{\Longrightarrow}} \boldsymbol{p}'$ and $\boldsymbol{q} \overset{*}{\underset{3d}{\Longrightarrow}} \boldsymbol{q}'$, where $R_{\leqslant 2}(\boldsymbol{p}') = R_{\leqslant 2}(\boldsymbol{q}') = \boldsymbol{w}(abc)^g ab$ for some $g \geqslant j > k$. Then there has to be at least one substring of length three with distinct symbols, or *distinct triplet*, in $\boldsymbol{p}$. Since $\boldsymbol{p} = \text{Ext}(\text{Reg}(\boldsymbol{r}), \boldsymbol{x})$, then by definition, the only distinct triplet possible is from the set $\text{rot}(\text{main}(\mathbf{r}))$. Therefore, $\text{Count}(\text{rot}(\text{main}(\mathbf{r})), \boldsymbol{p}) > 0$. ■

Example 3.13. Let $\boldsymbol{r}$ and $\boldsymbol{x}$ be as defined in Example 3.11. We consider another word $\boldsymbol{y}' = 0010100200120201$. As before, note that $R_{\leqslant 3}(\boldsymbol{x}) = R_{\leqslant 3}(\boldsymbol{y}') = \boldsymbol{r}$. Recall that $\text{main}(\boldsymbol{r}) = 102, \text{Reg}(\boldsymbol{r}) = 0102$, and $a = 0$. Then $\boldsymbol{p}$, $*\text{Pref}(\boldsymbol{r}, \boldsymbol{x})$ and $\boldsymbol{u}$ are computed as before. In contrast, for $\boldsymbol{y}'$, we have

$$\boldsymbol{q}' = \text{Ext}(0102, \boldsymbol{y}') = 00101002, \quad *\text{Pref}(\boldsymbol{r}, \boldsymbol{y}') = 001010, \quad \boldsymbol{v}' = \boldsymbol{y}' \setminus *\text{Pref}(\boldsymbol{r}, \boldsymbol{y}') = 0200120201.$$

Now, $R_{\leqslant 2}(\boldsymbol{q}') = 0102$, and so, $\text{Count}(\text{main}(\boldsymbol{r}), R_{\leqslant 2}(\boldsymbol{q}')) = 1$. Hence, $\text{Count}(\text{main}(\boldsymbol{r}), R_{\leqslant 2}(\boldsymbol{p})) \neq \text{Count}(\text{main}(\boldsymbol{r}), R_{\leqslant 2}(\boldsymbol{q}'))$. Furthermore, $\text{Count}(\text{rot}(\text{main}(\boldsymbol{r})), \boldsymbol{q}) = \text{Count}(\text{rot}(102), 00101002) = 0$. Therefore, by Corollary 3.12, we have $\boldsymbol{x}$ and $\boldsymbol{y}$ are not confusable.

3.2 Confusability Algorithm

Based on the ideas in the previous subsection, we formally describe our algorithm Confuse in Algorithm 1 and demonstrate that the running time is linear. In the following analysis, we assume the definitions in Algorithm 1.

- Main($\boldsymbol{r}$) and Reg($\boldsymbol{r}$) can be computed in constant time. This is clear from definition.
- The prefixes $\boldsymbol{p}$ and $\boldsymbol{q}$ can be computed in time linear in $|\boldsymbol{p}|$ and $|\boldsymbol{q}|$, respectively. From Proposition 2.9, we have that $D^*_{\leqslant 3}(\mathrm{Reg}(\boldsymbol{r}))$ is a regular language. Since the length of Reg($\boldsymbol{r}$) is at most six, the finite automaton generating the language has a constant number of vertices. Hence, the longest prefixes of $\boldsymbol{x}$ and $\boldsymbol{y}$ that are accepted by the finite automaton are precisely $\boldsymbol{p}$ and $\boldsymbol{q}$, respectively. In Appendix B, we show that the sum of the lengths of all prefixes that are computed in all recursive calls is at most $3(|\boldsymbol{x}| + |\boldsymbol{y}|)$. Therefore, the computation of all prefixes takes linear time.
- In the recursive calls, the root $\boldsymbol{r}^*$ can be computed in constant time. As mentioned earlier, the first step is to compute the roots of $\boldsymbol{x}$ and $\boldsymbol{y}$ in linear time. When the roots are equal to $\boldsymbol{r}$ and certain conditions are met, we make the recursive call Confuse-Recursion. If the roots are recomputed for the shorter words $\boldsymbol{u}$ and $\boldsymbol{v}$, then the overall running time for Confuse is quadratic. To avoid this, we observe that the roots for the shorter words are always equal and in fact, the common root $\boldsymbol{r}^*$ of $\boldsymbol{u}$ and $\boldsymbol{v}$ may be easily inferred from $\boldsymbol{r}$. More precisely, if $\mathrm{Reg}(\boldsymbol{r}) = \boldsymbol{w}(abc)^\ell ab$, then $\boldsymbol{r}^* = \boldsymbol{r} \setminus \boldsymbol{w}(abc)^\ell$.
- In the recursive calls, determining $\boldsymbol{r}$ has at most two symbols can be done in constant time. If $\boldsymbol{r}$ has length at least four, then $\boldsymbol{r}$ necessarily has at least three distinct symbols. Otherwise, it contradicts the fact that $\boldsymbol{r}$ is irreducible. Therefore, it suffices to check if the first four symbols $\boldsymbol{r}$ contain three distinct symbols.

In summary, from the above observations, we conclude that the running time is $O(\max\{|\boldsymbol{x}|, |\boldsymbol{y}|\})$.

4 $\leqslant$3-TANDEM-DUPLICATION CODES

We use insights gained from the previous section to construct $(n, \leqslant 3; q)$-TD codes. Motivated by the concept of roots, we consider a $\leqslant$3-irreducible word $\boldsymbol{r}$, and we say that a $(n, \leqslant 3; q)$-TD code C is an $(n, \leqslant 3; \boldsymbol{r})$-TD code if all words in C belong to $D^*_{\leqslant 3}(\boldsymbol{r})$.

For $\boldsymbol{r} \in \mathrm{Irr}_{\leqslant 3}(i, q)$ with $i \leqslant n$, suppose that $C(n, \boldsymbol{r})$ is an $(n, \leqslant 3; \boldsymbol{r})$-TD code. Then Proposition 2.7 implies that $\bigcup_{\boldsymbol{r} \in \mathrm{Irr}_{\leqslant 3}(i,q),\ i \leqslant n} C(n, \boldsymbol{r})$ is an $(n, \leqslant 3; q)$-TD code. Trivially, $\{\xi_{n-i}(\boldsymbol{r})\}$ is an $(n, \leqslant 3; \boldsymbol{r})$-TD code for all $\boldsymbol{x} \in \mathrm{Irr}_{\leqslant 3}(i, q)$. Taking the union of these codes, we recover Construction 1.

Therefore, in the rest of this section, our objective is to provide estimates on the size of an optimal $(n, \leqslant 3; \boldsymbol{r})$-TD code for fixed $\leqslant$3-irreducible word $\boldsymbol{r}$. To simplify our discussion, we focus on the case where $q = 3$ and let $T(n)$ and $T(n, \boldsymbol{r})$ to denote the sizes of an optimal $(n, \leqslant 3; 3)$-TD code and an optimal $(n, \leqslant 3; \boldsymbol{r})$-TD code, respectively.

Pick $\boldsymbol{x} \in D^*_{\leqslant 3}(\boldsymbol{r})$. Using concepts from Section 3, we define the following combinatorial characterisation of $\boldsymbol{x}$. Set $\boldsymbol{x}_1 = \boldsymbol{x}$ and $\boldsymbol{r}_1 = \boldsymbol{r}$. For $i \geqslant 2$, set $\boldsymbol{x}_i = \boldsymbol{x}_{i-1} \setminus *\mathrm{Pref}(\boldsymbol{r}_{i-1}, \boldsymbol{x}_{i-1})$ and $\boldsymbol{r}_i = R_{\leqslant 3}(\boldsymbol{x}_i)$. We terminate this recursion when $\boldsymbol{r}_{m+1}$ has less than three distinct symbols. If $\boldsymbol{r}_m$ is the last root to have three distinct symbols, then we say that $\boldsymbol{r}$ has m *regions* and for $1 \leqslant i \leqslant m$, define

$$\boldsymbol{p}_i = \mathrm{Ext}(\mathrm{Reg}(\boldsymbol{r}_i), \boldsymbol{x}_i),\ \boldsymbol{t}_i = \mathrm{main}(\boldsymbol{r}_i),\ c_i = \mathrm{Count}(\boldsymbol{t}_i, R_{\leqslant 2}(\boldsymbol{p}_i)),\ \text{and } \delta_i = \begin{cases} +, & \text{if } \mathrm{Count}(\mathrm{rot}(\boldsymbol{t}_i), \boldsymbol{p}_i) > 0, \\ -, & \text{otherwise.} \end{cases}$$

For the word $\boldsymbol{x}$, we define its *label* by $\mathrm{Label}(\boldsymbol{x}) = (\boldsymbol{r}, (c_1, \delta_1), (c_2, \delta_2), \ldots, (c_m, \delta_m))$.

Example 4.1. Consider $\boldsymbol{r} = 01210$. Then $\boldsymbol{r}$ has two regions with $\boldsymbol{t}_1 = 012$ and $\boldsymbol{t}_2 = 210$. Consider also the words 01210210, 01201210, and 01112110 that belongs to $D^*_{\leqslant 3}(01210)$. Then their labels

are as follows:

$$\begin{aligned}\text{Label}(01210210) &= (01210, (1,+), (2,+)),\\ \text{Label}(01201210) &= (01210, (2,+), (1,+)),\\ \text{Label}(01112110) &= (01210, (1,-), (1,-)).\end{aligned}$$

Then it follows from Theorem 3.9 and Corollaries 3.10 and 3.12 that 01210210 and 01201210 are confusable, while 01210210 and 01112110 are not confusable.

More generally, the next proposition is immediate from Theorem 3.9 and Corollaries 3.10 and 3.12.

Proposition 4.2. *Let $\boldsymbol{x}$ and $\boldsymbol{y}$ belong to $D^*_{\leqslant 3}(\boldsymbol{r})$. Suppose that*

$$\begin{aligned}\text{Label}(\boldsymbol{x}) &\triangleq \left(\boldsymbol{r}, \left(c_1^{(\boldsymbol{x})}, \delta_1^{(\boldsymbol{x})}\right), \left(c_2^{(\boldsymbol{x})}, \delta_2^{(\boldsymbol{x})}\right), \ldots, \left(c_m^{(\boldsymbol{x})}, \delta_m^{(\boldsymbol{x})}\right)\right),\\ \text{Label}(\boldsymbol{y}) &\triangleq \left(\boldsymbol{r}, \left(c_1^{(\boldsymbol{y})}, \delta_1^{(\boldsymbol{y})}\right), \left(c_2^{(\boldsymbol{y})}, \delta_2^{(\boldsymbol{y})}\right), \ldots, \left(c_m^{(\boldsymbol{y})}, \delta_m^{(\boldsymbol{y})}\right)\right).\end{aligned}$$

Then $\boldsymbol{x}$ and $\boldsymbol{y}$ are not $\leqslant 3$-confusable if and only if for some $1 \leqslant i \leqslant m$,

$$\left(c_i^{(\boldsymbol{x})} < c_i^{(\boldsymbol{y})} \text{ and } \delta_i^{(\boldsymbol{x})} = -\right) \text{ or } \left(c_i^{(\boldsymbol{x})} > c_i^{(\boldsymbol{y})} \text{ and } \delta_i^{(\boldsymbol{y})} = -\right).$$

Immediate from Proposition 4.2 are certain sufficient conditions for two words to be confusable.

Corollary 4.3. *Let $\boldsymbol{x}, \boldsymbol{y} \in D^*_{\leqslant 3}(\boldsymbol{r})$ and their labels* Label($\boldsymbol{x}$) *and* Label($\boldsymbol{y}$) *be as defined in Proposition 4.2.*

(i) If $c_i^{(\boldsymbol{x})} = c_i^{(\boldsymbol{y})}$ for all i, then $\boldsymbol{x}$ and $\boldsymbol{y}$ are $\leqslant 3$-confusable.
(ii) If $\delta_i^{(\boldsymbol{x})} = \delta_i^{(\boldsymbol{y})} = +$ for all i, then $\boldsymbol{x}$ and $\boldsymbol{y}$ are $\leqslant 3$-confusable.

From Corollary 4.3, we use the number of integer solutions to certain equations as an upper bound for $T(n, \boldsymbol{r})$. As this combinatorial argument is fairly technical, we state the upper bound and defer the proof to Appendix C.

Proposition 4.4. *Let $i \leqslant n$. Suppose that $\boldsymbol{r} \in \text{Irr}_{\leqslant 3}(i, 3)$ has m regions. Then,*

$$T(n, \boldsymbol{r}) \leqslant U(n, i, m) \triangleq \begin{cases} \binom{(n-i)/3+m}{m} - \binom{(n-i)/3+m-1}{m-1} + 1, & \text{if 3 divides } n-i,\\ \binom{\lfloor (n-i)/3 \rfloor + m}{m}, & \text{otherwise.}\end{cases}$$

4.1 Estimates of $T(|\boldsymbol{r}| + t, \boldsymbol{r})$ for $0 \leqslant t \leqslant 5$

Immediate from Proposition 4.4 is that $T(|\boldsymbol{r}|, \boldsymbol{r}) = T(|\boldsymbol{r}| + 1, \boldsymbol{r}) = T(|\boldsymbol{r}| + 2, \boldsymbol{r}) = 1$. Next, we provide lower bounds for $T(n, \boldsymbol{r})$ by constructing $(n, \leqslant 3; \boldsymbol{r})$-TD codes. We first show that there exists an $(n, \leqslant 3; \boldsymbol{r})$-TD code of size two.

Construction 2. *Let $\boldsymbol{r} \in \text{Irr}_{\leqslant 3}(i, 3)$ for some $i \geqslant 4$. Set*

$$C_2 \triangleq \begin{cases} \{r_1 r_2 r_3 r_1 r_2 r_3 r_4 \cdots r_i, \quad r_1 r_2 r_2 r_3 r_3 r_4 r_4 \cdots r_i\}, & \text{if } r_1 \neq r_3,\\ \{r_1 r_2 r_1 r_4 r_2 r_1 r_4 r_5 \cdots r_i, \ r_1 r_2 r_1 r_1 r_4 r_4 r_5 r_5 \cdots r_i\}, & \text{if } r_1 = r_3 \text{ and } i \geqslant 5,\\ \{r_1 r_2 r_1 r_4 r_2 r_1 r_4, \quad r_1 r_2 r_1 r_1 r_4 r_4 r_4\}, & \text{if } r_1 = r_3 \text{ and } i = 4.\end{cases}$$

Then C_2 is an $(i+3, \leqslant 3; \boldsymbol{r})$-TD code of size two. By Proposition 4.4, C_2 is optimal.

Proof. Let $C_2 = \{\boldsymbol{x}, \boldsymbol{y}\}$ and the labels of the words be as given in Proposition 4.2. Observe that in all cases, $(c_1^{(\boldsymbol{x})}, \delta_1^{(x)}) = (2, +)$ and $(c_1^{(\boldsymbol{y})}, \delta_1^{(y)}) = (1, -)$. Proposition 4.2 then implies that $\boldsymbol{x}$ and $\boldsymbol{y}$ are not confusable. ∎

Following the above construction, we have an improvement to Construction 1.

Corollary 4.5. *For $n \geqslant 6$, we have that*

$$T(n) \geqslant \sum_{\substack{1 \leqslant i \leqslant 3, \text{ or} \\ n-2 \leqslant i \leqslant n}} \mathrm{Irr}_{\leqslant 3}(i,3) + 2\sum_{i=4}^{n-3} \mathrm{Irr}_{\leqslant 3}(i,3).$$

Construction 2 also implies that $T(|\boldsymbol{r}|+3, \boldsymbol{r}) = 2$ for all irreducible words $\boldsymbol{r}$. When $n \in \{|\boldsymbol{r}|+4, |\boldsymbol{r}|+5\}$, Proposition 4.4 states that $T(n, \boldsymbol{r}) \leqslant m+1$, where m is the number of regions for $\boldsymbol{r}$. However, the upper bound can be reduced to a constant independent of m and we do this by analysing the possible labels of words in a tandem-duplication code.

Proposition 4.6. $T(|\boldsymbol{r}|+4, \boldsymbol{r}) \leqslant 4$.

Proof. Let $T(|\boldsymbol{r}|+4, \boldsymbol{r}) = b$, and suppose that $C(|\boldsymbol{r}|+4, \boldsymbol{r})$ is the $(|\boldsymbol{r}|+4, \leqslant 3; \boldsymbol{r})$-TD code with size b. Let $C(|\boldsymbol{r}|+4, \boldsymbol{r}) = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_b\}$. Let

$$\mathrm{Label}(\boldsymbol{x}_i) \triangleq \left(\boldsymbol{r}, \left(c_1^{(\boldsymbol{x}_i)}, \delta_1^{(\boldsymbol{x}_i)}\right), \left(c_2^{(\boldsymbol{x}_i)}, \delta_2^{(\boldsymbol{x}_i)}\right), \ldots, \left(c_m^{(\boldsymbol{x}_i)}, \delta_m^{(\boldsymbol{x}_i)}\right)\right).$$

Note that there can only be at most one word, $\boldsymbol{x}_i \in C(|\boldsymbol{r}|+4, \boldsymbol{r})$, such that $c_t^{(\boldsymbol{x}_i)} = 1$ for all $1 \leqslant t \leqslant m$. We also know that $c_t^{(\boldsymbol{x}_i)} \leqslant 2$ for all $1 \leqslant i \leqslant b, 1 \leqslant t \leqslant m$. Otherwise, the length of $\boldsymbol{x}_i$ is be at least $|\boldsymbol{r}|+6$. Furthermore, for each $\boldsymbol{x}_i \in C(|\boldsymbol{r}|+4, \boldsymbol{r})$, there can only be at most one $1 \leqslant t \leqslant m$, such that $c_t^{(\boldsymbol{x}_i)} = 2$. So, we can define $C'(|\boldsymbol{r}|+4, \boldsymbol{r}) = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{b'}\}$, where $b' \leqslant b$, to be the subset of $C(|\boldsymbol{r}|+4, \boldsymbol{r})$ such that for all $1 \leqslant i \leqslant b'$, $c_t^{(\boldsymbol{x}_i)} = 2$ for exactly one $t \leqslant m$, and conclude that $b \leqslant 1 + b'$.

We say that $\boldsymbol{x}_i$ *loses to* $\boldsymbol{x}_j$ if there exists $1 \leqslant t \leqslant m$, such that $c_t^{(\boldsymbol{x}_i)} < c_t^{(\boldsymbol{x}_j)}$ and $\delta_t^{(\boldsymbol{x}_i)} = -$. Note that in $C'(|\boldsymbol{r}|+4, \boldsymbol{r})$, for any distinct pair $1 \leqslant i < j \leqslant b'$, either $\boldsymbol{x}_i$ *loses to* $\boldsymbol{x}_j$ or $\boldsymbol{x}_j$ *loses to* $\boldsymbol{x}_i$. Hence, in total, there is at least $b'(b'-1)/2$ losses in $C'(|\boldsymbol{r}|+4, \boldsymbol{r})$. Suppose that for some $1 \leqslant t \leqslant m$, we have $c_t^{(\boldsymbol{x}_i)} = 2$, then there can only be at most one $1 \leqslant t' \leqslant m, t' \neq t$, such that $\delta_{t'}^{(\boldsymbol{x}_i)} = -$. Otherwise, the length of $\boldsymbol{x}_i$ is be at least $|\boldsymbol{r}|+5$. And since for distinct $i, j \leqslant b'$, we can never have $c_t^{(\boldsymbol{x}_i)} = c_t^{(\boldsymbol{x}_j)} = 2$. This implies that for each word $\boldsymbol{x}_i \in C'(|\boldsymbol{r}|+4, \boldsymbol{r})$, the word $\boldsymbol{x}_i$ can only lose at most once to another word in $C'(|\boldsymbol{r}|+4, \boldsymbol{r})$. Therefore, there are at most b' losses in $C'(|\boldsymbol{r}|+4, \boldsymbol{r})$. Hence, we have the inequality $b'(b'-1)/2 \leqslant b'$, which implies that $b' \leqslant 3$, and therefore $b \leqslant 4$.

We conclude the proof with the labels of a possible set of four words in $C(|\boldsymbol{r}|+4, \boldsymbol{r})$:

$$\begin{aligned}
\mathrm{Label}(\boldsymbol{x}_1) &= (\boldsymbol{r}, (2,+), (1,-), (1,+)),\\
\mathrm{Label}(\boldsymbol{x}_2) &= (\boldsymbol{r}, (1,+), (2,+), (1,-)),\\
\mathrm{Label}(\boldsymbol{x}_3) &= (\boldsymbol{r}, (1,-), (1,+)), (2,+),\\
\mathrm{Label}(\boldsymbol{x}_4) &= (\boldsymbol{r}, (1,-), (1,-), (1,-)).
\end{aligned}$$

■

Corollary 4.7. $T(|\boldsymbol{r}|+5, \boldsymbol{r}) \leqslant 6$.

Proof. Following the proof of Proposition 4.6, there are at least $b'(b'-1)/2$ losses in $C'(|\boldsymbol{r}|+5, \boldsymbol{r})$. And for each word in $C'(|\boldsymbol{r}|+5, \boldsymbol{r})$, it can have at most two losses. Hence, we have $b'(b'-1)/2 \leqslant 2b'$, and so $T(|\boldsymbol{r}|+5, \boldsymbol{r}) = b \leqslant 1 + b' \leqslant 1 + 5 = 6$.

As before, we provide the labels of a possible set of six words in $C(|\boldsymbol{r}|+5,\boldsymbol{r})$:

$$\begin{aligned}
\mathrm{Label}(\boldsymbol{x}_1) &= (\boldsymbol{r},(2,+),(1,-),(1,-),(1,+),(1,+)),\\
\mathrm{Label}(\boldsymbol{x}_2) &= (\boldsymbol{r},(1,+),(2,+),(1,-),(1,-),(1,+)),\\
\mathrm{Label}(\boldsymbol{x}_3) &= (\boldsymbol{r},(1,+),(1,+)),(2,+),(1,-),(1,-),\\
\mathrm{Label}(\boldsymbol{x}_4) &= (\boldsymbol{r},(1,-),(1,+),(1,+),(2,+),(1,-)),\\
\mathrm{Label}(\boldsymbol{x}_5) &= (\boldsymbol{r},(1,-),(1,-),(1,+),(1,+),(2,+)),\\
\mathrm{Label}(\boldsymbol{x}_6) &= (\boldsymbol{r},(1,-),(1,-),(1,-),(1,-),(1,-)).
\end{aligned}$$

■

4.2 Exact Value of $T(n,r)$ When the Number of Regions is Exactly One

Proposition 4.2 suggests that we examine code constructions according to the number of regions of $\boldsymbol{r}$. When $|\boldsymbol{r}|\leqslant 2$, the number of regions is zero and hence $T(n,\boldsymbol{r})=1$. We consider the smallest nontrivial case where $\boldsymbol{r}$ has exactly one region. Without loss of generality, assume that $\boldsymbol{r}\in R\triangleq\{012,0120,01201,1012,10120,101201,\ 0121,01202,012010,10121,101202,1012010\}$, and so, $\boldsymbol{t}_1=012$.

Construction 3. *For $\boldsymbol{r}\in R$ and $\ell\geqslant 1$, define the words $\boldsymbol{x}(\boldsymbol{r},\ell)$ and $\boldsymbol{z}(\boldsymbol{r},\ell)$ with the following rule.*

$\boldsymbol{r}$	$\boldsymbol{x}(\boldsymbol{r},\ell)$	$\boldsymbol{z}(\boldsymbol{r},\ell)$	$\boldsymbol{r}$	$\boldsymbol{x}(\boldsymbol{r},\ell)$	$\boldsymbol{z}(\boldsymbol{r},\ell)$
012	$0(112200)^{\ell-1}112$	$(012)^{\ell}$	1012	$10(112200)^{\ell-1}112$	$1(012)^{\ell}$
0120	$0(112200)^{\ell-1}11220$	$(012)^{\ell}0$	10120	$10(112200)^{\ell-1}11220$	$1(012)^{\ell}0$
01201	$0(112200)^{\ell-1}1122001$	$(012)^{\ell}01$	101201	$10(112200)^{\ell-1}1122001$	$1(012)^{\ell}01$
0121	$0(112200)^{\ell-1}1121$	$(012)^{\ell}1$	10121	$10(112200)^{\ell-1}1121$	$1(012)^{\ell}1$
01202	$0(112200)^{\ell-1}112202$	$(012)^{\ell}02$	101202	$10(112200)^{\ell-1}112202$	$1(012)^{\ell}02$
012010	$0(112200)^{\ell-1}11220010$	$(012)^{\ell}010$	1012010	$10(112200)^{\ell-1}11220010$	$1(012)^{\ell}010$

For $\boldsymbol{r}\in R$ and $n\geqslant|\boldsymbol{x}(\boldsymbol{r},2)|$, set $\ell_z=\lfloor(n-|\boldsymbol{r}|)/3\rfloor+1$ and

$$C_{\boldsymbol{r}}(n)\triangleq\left\{\xi_{n-|\boldsymbol{x}(\boldsymbol{r},\ell)|}(\boldsymbol{x}(\boldsymbol{r},\ell)):|\boldsymbol{x}(\boldsymbol{r},\ell)|\leqslant n\right\}\cup\left\{\xi_{n-|\boldsymbol{z}(\boldsymbol{r},\ell_z)|}(\boldsymbol{z}(\boldsymbol{r},\ell_z))\right\}.$$

Then $C_{\boldsymbol{r}}(n)$ is an $(n,\leqslant 3;\boldsymbol{r})$-TD code. Furthermore, $C_{\boldsymbol{r}}(n)$ in optimal. Therefore, if we set $n_2=|\boldsymbol{x}(r,2)|$, then we have

$$T(n,\boldsymbol{r})=\begin{cases}\left\lfloor\frac{n-n_2}{6}\right\rfloor+3, & \text{if}\quad n\geqslant n_2,\\ 2, & \text{if}\quad n_2>n\geqslant|\boldsymbol{r}|+3,\\ 1, & \text{if}\ |\boldsymbol{r}|+3>n\geqslant|\boldsymbol{r}|,\\ 0, & \text{otherwise.}\end{cases}$$

Proof. Let $\mathcal{X}=\left\{\xi_{n-|\boldsymbol{x}(\boldsymbol{r},\ell)|}(\boldsymbol{x}(\boldsymbol{r},\ell)):|\boldsymbol{x}(\boldsymbol{r},\ell)|\leqslant n\right\}$ and $M\geqslant 2$ is the biggest integer that satisfies $|\boldsymbol{x}(\boldsymbol{r},M))|\leqslant n$. Observe that for all $\ell\geqslant 1$, we have $\mathrm{Label}(\boldsymbol{x}(\boldsymbol{r},\ell))=(\boldsymbol{r},(\ell,-))$ and $\mathrm{Label}(\boldsymbol{z}(\boldsymbol{r},\ell))=(\boldsymbol{r},(\ell,+))$. Furthermore, the length of $\boldsymbol{x}(\boldsymbol{r},\ell)$ is at least $6(\ell-1)+|\boldsymbol{r}|+1$. Therefore, we have $6(M-1)+|\boldsymbol{r}|+1\leqslant|\boldsymbol{x}(\boldsymbol{r},M)|\leqslant n$. So, $\ell_z=\lfloor(n-|\boldsymbol{r}|)/3\rfloor+1\geqslant\lfloor(6M-5)/3\rfloor+1\geqslant M+1$ for $M\geqslant 2$.

Hence, for any word $\boldsymbol{x}$ in $\mathcal{X}$, we have that $\mathrm{Label}(\boldsymbol{x})=(\boldsymbol{r},(\ell',-))$ with $1\leqslant\ell'\leqslant M<\ell_z$. Therefore, Proposition 4.2 implies that $\boldsymbol{x}$ and $\boldsymbol{z}(\boldsymbol{r},\ell_z)$ are not confusable. Also, Corollary 4.3(ii) implies that $\boldsymbol{x}$ and $\boldsymbol{x}'$ are not confusable for $\boldsymbol{x},\boldsymbol{x}'\in\mathcal{X}$. Therefore, $C_{\boldsymbol{r}}(n)$ is an $(n,\leqslant 3;\boldsymbol{r})$-TD code.

To demonstrate optimality, we observe that for $\boldsymbol{y}\in D^*_{\leqslant 3}(\boldsymbol{r})$, if $\mathrm{Label}(\boldsymbol{y})=(\ell,-)$, then $|\boldsymbol{y}|\geqslant|\boldsymbol{x}(\boldsymbol{r},\ell)|$. Similarly, if $\mathrm{Label}(\boldsymbol{y})=(\ell,+)$, then $|\boldsymbol{y}|\geqslant|\boldsymbol{z}(\boldsymbol{r},\ell)|$. Suppose that there is an $(n,\leqslant 3;\boldsymbol{r})$-TD code C' with size $M+2$. We know from Corollary 4.3(ii) that there is at most one codeword

in C' whose label δ_1 is +. So, there are at least $M+1$ codewords in C' whose label δ_1 are −. Furthermore, Corollary 4.3(i), implies that no two codewords whose label δ_1 are −, have the same label c_1. Therefore there exists a $\boldsymbol{y} \in C'$ whose label is $(c_1, -)$ with $c_1 \geqslant M+1$. This implies that $n = |\boldsymbol{y}| \geqslant |\boldsymbol{x}(\boldsymbol{r}, M+1)|$, which contradicts M being the biggest integer that satisfies $|\boldsymbol{x}(\boldsymbol{r}, M))| \leqslant n$.
Finally, the values of $T(n, \boldsymbol{r})$ follow from straightforward computations. ■

Hence, the value of $T(n, \boldsymbol{r})$ is completely determined whenever $\boldsymbol{r}$ has at most one region. Combining this result with Proposition 4.4, we have the following theorem.

Theorem 4.8. *Let $I(i, m)$ denote the number of irreducible words in $\mathrm{Irr}_{\leqslant 3}(i, 3)$ with exactly m regions[1], and $U(n, i, m)$ be as defined in Proposition 4.4. Then,*

$$T(n) \leqslant \sum_{\boldsymbol{r} \in R} T(n, \boldsymbol{r}) + \sum_{i=5}^{n} \sum_{m=2}^{i} I(i, m) U(n, i, m). \tag{1}$$

4.3 Estimates of $T(n)$ for $n \leqslant 30$

In addition to the above constructions, we construct tandem-duplication codes for small lengths by searching for them exhaustively. Specifically, fix $\boldsymbol{r} \in \mathrm{Irr}_{\leqslant 3}(i, 3)$ and $n \geqslant i$, we construct the graph $\mathcal{G}(n, \boldsymbol{r})$, whose vertices correspond to the set of all labels of descendants of $\boldsymbol{r}$ of length n, or $\{\boldsymbol{L} : \boldsymbol{L} = \mathrm{Label}(\boldsymbol{x})$ and $\boldsymbol{x} \in \Sigma_3^n \cap D^*_{\leqslant 3}(\boldsymbol{r})\}$. Two vertices or labels $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ are connected in $\mathcal{G}(n, \boldsymbol{r})$ if and only if the words whose labels are $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ are not $\leqslant$3-confusable. Hence, a clique of size M in the graph $\mathcal{G}(n, \boldsymbol{r})$ correspond to an $(n, \leqslant 3; \boldsymbol{r})$-TD code of size M and $T(n, \boldsymbol{r})$ is the maximum size of a clique in $\mathcal{G}(n, \boldsymbol{r})$.

We use the exact algorithm `MaxCliqueDyn` [8] to determine the maximum size of the clique in these graph $\mathcal{G}(n, \boldsymbol{r})$ for $n \leqslant 20$. Since $T(n) = \sum_{\boldsymbol{r} \in \mathrm{Irr}_{\leqslant 3}(i,3),\ i \leqslant n} T(n, \boldsymbol{r})$, we tabulate the results $T(n)$ in Table 1.

Remark 3.

- Such a method to compute $T(n, \boldsymbol{r})$ is only possible because we have developed the algorithm to determine confusability in Section 3. Prior to this work, the necessary conditions for $\leqslant$3-confusability was not known and hence, methods to compute $T(n, \boldsymbol{r})$ were not available.
- For a fixed length and root, even though the set of descendants is huge, the set of all labels are significantly smaller. Hence, the task of computing maximum cliques remains feasible. More concretely, when $n = 20$, the order of the largest graph $\mathcal{G}(20, \boldsymbol{r})$ is 366, despite the fact that the average size[2] of $\Sigma_3^{20} \cap D^*_{\leqslant 3}(\boldsymbol{r})$ is $3^{20}/27{,}687 \approx 125{,}935$.

Finally, we develop recursive constructions in the following proposition.

Proposition 4.9. *Let $\boldsymbol{r} = r_1 r_2 \cdots r_i \in \mathrm{Irr}_{\leqslant 3}(i, 3)$. Then the following holds:*

$$T(n, \boldsymbol{r}) \geqslant \begin{cases} T(n-1, \boldsymbol{r} \setminus r_1), & \text{if } r_1 = r_3, \\ \max\{2T(n-4, \boldsymbol{r} \setminus r_1), 3T(n-8, \boldsymbol{r} \setminus r_1)\}, & \text{if } r_1 \neq r_3,\ r_1 \neq r_4, \\ \max\{2T(n-5, \boldsymbol{r} \setminus r_1 r_2), 3T(n-10, \boldsymbol{r} \setminus r_1 r_2)\}, & \text{if } r_1 \neq r_3,\ r_1 = r_4,\ r_2 \neq r_5, \\ \max\{2T(n-6, \boldsymbol{r} \setminus r_1 r_2 r_3), 3T(n-12, \boldsymbol{r} \setminus r_1 r_2 r_3)\}, & \text{if } r_1 \neq r_3,\ r_1 = r_4,\ r_2 = r_5. \end{cases}$$

Furthermore, $T(n, \boldsymbol{r}) \geqslant T(n-1, \boldsymbol{r})$ and $T(n, \boldsymbol{r}) = T(n, \boldsymbol{r}^R)$, where $\boldsymbol{z}^R$ denotes the reverse of word $\boldsymbol{z}$.

Proof. First consider $r_1 = r_3$. Suppose that $\mathcal{D}$ is an $(n-1, \leqslant 3; \boldsymbol{r} \setminus r_1)$-TD code. To construct a code of length n, we simply prepend the prefix r_1 to all words in $\mathcal{D}$. For convenience, given a set of

[1] We provide a formula in Appendix D.
[2] The number of $\leqslant$3-irreducible ternary words of length at most 20 is 27,687.

Table 1. Lower, Upper Bounds and Exact Values for $T(n)$

Length	Constr. 1	This article	Equation (1)	Prop. 2.11	Length	Constr. 1	This article	Equation (1)	Prop. 2.11
1	**3**	**3**	**3**	**3**	16	5,985	**10,641**	12,267	15,495
2	**9**	**9**	**9**	**9**	17	8,781	**16,287**	19,479	25,077
3	**21**	**21**	**21**	**21**	18	12,879	**25,005**	30,957	40,581
4	**39**	**39**	**39**	**39**	19	18,885	**38,223**	49,245	65,667
5	**69**	**69**	**69**	**69**	20	27,687	**57,957**	78,417	106,257
6	111	**117**	**117**	**117**	21	40,587	83,619	125,001	171,933
7	171	**195**	**195**	**195**	22	59,493	116,145	199,467	278,199
8	261	**315**	**315**	321	23	87,201	166,761	318,621	450,141
9	393	**495**	**495**	525	24	127,809	249,159	509,457	728,349
10	585	**777**	**777**	855	25	187,323	375,129	815,361	1,178,499
11	867	**1,221**	1,227	1,389	26	274,545	558,573	1,306,107	1,906,857
12	1,281	**1,887**	1,941	2,253	27	402,375	813,771	2,093,967	3,085,365
13	1,887	**2,913**	3,075	3,651	28	589,719	1,164,309	3,359,685	4,992,231
14	2,775	**4,527**	4,875	5,913	29	864,285	1,675,935	5,394,369	8,077,605
15	4,077	**6,969**	7,731	9,573	30	1,266,681	2,464,419	8,667,075	13,069,845

The columns labelled Construction 1 and Proposition 2.11 correspond to the previous known lower and upper bounds for $T(n)$, respectively. The column labelled "This article" summarizes the best lower bounds that arise from this work, while the column labelled Equation (1) give the upper bound that result from Theorem 4.8. Optimal values of $T(n)$ are highlighted in **bold**.

words $\mathcal{X}$ and a word $\boldsymbol{p}$, we use $\boldsymbol{p}\mathcal{X}$ to denote the set $\{\boldsymbol{p}\boldsymbol{x} : \boldsymbol{x} \in \mathcal{X}\}$. When $\boldsymbol{p} = p$ is of length one, we simply write $p\mathcal{X}$. Using this notation, we set $C = r_1\mathcal{D}$. We then apply Proposition 4.2 and verify that C is an $(n, \leqslant 3; \boldsymbol{r})$-TD code whose size is given by $|\mathcal{D}|$.

Next, consider $r_1 \neq r_3$ and $r_1 \neq r_4$. Set $\boldsymbol{r}' = \boldsymbol{r} \setminus r_1$ and suppose that $\mathcal{D}$ is an $(n-4, \leqslant 3; \boldsymbol{r}')$-TD code. Let $\mathcal{D}_1 = r_1r_2r_2r_2\mathcal{D}$ and $\mathcal{D}_2 = r_1r_2r_3r_1\mathcal{D}$, and so, $\mathcal{D}_1 \cup \mathcal{D}_2 \subseteq D^*_{\leqslant 3}(\boldsymbol{r}) \cap \Sigma_3^n$.

For $\boldsymbol{x} \in \mathcal{D}$, let $\mathrm{Label}(\boldsymbol{x}) = (\boldsymbol{r}', (c_1, \delta_1), \ldots, (c_m, \delta_m))$. Then, we have that

$$\mathrm{Label}(r_1r_2r_2r_2\boldsymbol{x}) = (\boldsymbol{r}, (1, -), (c_1, \delta_1), \ldots, (c_m, \delta_m)),$$
$$\mathrm{Label}(r_1r_2r_3r_1\boldsymbol{x}) = (\boldsymbol{r}, (2, +), (c_1, \delta_1), \ldots, (c_m, \delta_m)).$$

Proposition 4.2 implies that $r_1r_2r_2r_2\boldsymbol{x}$ and $r_1r_2r_3r_1\boldsymbol{x}$ are not confusable. We can similarly check that any pair of distinct words in $\mathcal{D}_1 \cup \mathcal{D}_2$ are not confusable.

For the remaining cases, we choose the short code $\mathcal{D}$ and prepend $\mathcal{D}$ according to the rules below.

Conditions for $\boldsymbol{r}$	Short Code $\mathcal{D}$	$(n, \leqslant 3; \boldsymbol{r})$-TD code
$r_1 = r_3$	$(n-1, \leqslant 3; \boldsymbol{r} \setminus r_1)$-TD code	r_1
$r_1 \neq r_3, r_1 \neq r_4$	$(n-4, \leqslant 3; \boldsymbol{r} \setminus r_1)$-TD code	$r_1r_2r_2r_2\mathcal{D}$ $\cup\, r_1r_2r_3r_1\mathcal{D}$
$r_1 \neq r_3, r_1 \neq r_4$	$(n-8, \leqslant 3; \boldsymbol{r} \setminus r_1)$-TD code	$r_1r_2r_2r_2r_2r_2r_2r_2\mathcal{D}$ $\cup\, r_1r_2r_2r_3r_3r_1r_1r_2\mathcal{D}$ $\cup\, r_1r_2r_3r_1r_2r_3r_1r_2\mathcal{D}$
$r_1 \neq r_3, r_1 = r_4, r_2 \neq r_5$	$(n-5, \leqslant 3; \boldsymbol{r} \setminus r_1r_2)$-TD code	$r_1r_2r_2r_2r_3\mathcal{D}$ $\cup\, r_1r_2r_3r_1r_2\mathcal{D}$

Conditions for $\boldsymbol{r}$	Short Code $\mathcal{D}$	$(n, \leqslant 3; \boldsymbol{r})$-TD code
$r_1 \neq r_3, r_1 = r_4, r_2 \neq r_5$	$(n-10, \leqslant 3; \boldsymbol{r} \setminus r_1r_2)$-TD code	$r_1r_2r_2r_3r_3r_3r_3r_3r_3\mathcal{D}$ $\cup\, r_1r_2r_2r_3r_3r_1r_1r_2r_2r_3\mathcal{D}$ $\cup\, r_1r_2r_3r_1r_2r_3r_1r_2r_3r_3\mathcal{D}$
$r_1 \neq r_3, r_1 = r_4, r_2 = r_5$	$(n-6, \leqslant 3; \boldsymbol{r} \setminus r_1r_2r_3)$-TD code	$r_1r_2r_2r_3r_3r_1\mathcal{D}$ $\cup\, r_1r_2r_3r_1r_2r_3\mathcal{D}$
$r_1 \neq r_3, r_1 = r_4, r_2 = r_5$	$(n-12, \leqslant 3; \boldsymbol{r} \setminus r_1r_2r_3)$-TD code	$r_1r_2r_2r_3r_3r_1r_1r_1r_1r_1r_1\mathcal{D}$ $\cup\, r_1r_2r_2r_3r_3r_1r_1r_2r_2r_3r_3r_1\mathcal{D}$ $\cup\, r_1r_2r_3r_1r_2r_3r_1r_2r_3r_1r_1r_1\mathcal{D}$

To show that $T(n, \boldsymbol{r}) \geqslant T(n-1, \boldsymbol{r})$, let $\mathcal{D}$ be an $(n-1, \leqslant 3; \boldsymbol{r})$-TD code. Then $C = \{\xi_1(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{D}\}$ is an $(n, \leqslant 3; \boldsymbol{r})$-TD code.

To show that $T(n, \boldsymbol{r}) = T(n, \boldsymbol{r}^R)$, let $\mathcal{D}$ be an $(n, \leqslant 3; \boldsymbol{r})$-TD code. Then $C = \{\boldsymbol{x}^R : \boldsymbol{x} \in \mathcal{D}\}$ is an $(n, \leqslant 3; \boldsymbol{r}^R)$-TD code. ■

Using Proposition 4.9 with Constructions 2 and 3 and the values computed by `MaxCliqueDyn`, we derive lower bounds for $T(n)$ for $21 \leqslant n \leqslant 30$. The results are summarized in Table 1. In addition to the lower bounds for the code size $T(n)$, we also compare the upper bounds in Proposition 2.11 and (1). Observe that (1) is tight up to lengths at most ten and the constructions in this article improve the rates[3] for Construction 1 by as much as 6.74%.

5 DISCUSSION

We studied the problem of determining the $\leqslant k$-confusability of two words. Combining the results of this article, we have linear-time algorithms to solve the confusability problem for $k \in \{1, 2, 3\}$.

It remains open whether there exist efficient algorithms to determine $\leqslant 4$-confusability. One key obstacle is the fact that Proposition 2.7(iii) does not hold for $k = 4$. In particular, there exists $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $R_{\leqslant 4}(\boldsymbol{x}) \neq R_{\leqslant 4}(\boldsymbol{y})$, but $\boldsymbol{x}$ and $\boldsymbol{y}$ are $\leqslant 4$-confusable. An example is provided by Jain et al. [7], where $\boldsymbol{x} = 012$ and $\boldsymbol{y} = 0121012$ belongs to $\mathrm{Irr}_{\leqslant 4}(3)$, but 012101212 is a common descendant of $\boldsymbol{x}$ and $\boldsymbol{y}$.

Another approach is to consider the intersection of the descendant cones $D^*_{\leqslant 4}(\boldsymbol{x}) \cap D_{\leqslant 4}(\boldsymbol{y})$ as formal languages. Unfortunately, Leupold et al. [11] demonstrated that while the language $D^*_{\leqslant 4}(\boldsymbol{x})$ is context free for all $\boldsymbol{x}$, the language is not regular in general. However, given two context free languages L_1 and L_2, determining whether $L_1 \cap L_2$ is empty is an undecidable problem (see, for example, Sipser [16, Exercise 5.32]). While this does not imply that the confusability problem is undecidable, we nevertheless conjecture that it is undecidable for $k \geqslant 4$.

APPENDICES

A COMPUTING THE ROOT IN LINEAR TIME

We formally describe linear-time algorithms to compute all $\leqslant k$-roots for a word for $k \in \{2, 3\}$. Given any $\boldsymbol{x} \in \Sigma_q^*$, since the root set of $\boldsymbol{x}$ has size exactly one by Proposition 2.6, it suffices to find one $\boldsymbol{r} \in \mathrm{Irr}_{\leqslant k}(q)$ so that $\boldsymbol{r} \overset{*}{\underset{\leqslant k}{\Longrightarrow}} \boldsymbol{x}$. Lemma 3.3 then implies that we may reorder the duplications and assume that the tandem duplications are performed in decreasing lengths.

[3]The rate of a code C of length n is given by $(\log_3 |C|)/n$.

Therefore, to compute $\boldsymbol{r}$, we simply remove duplicates in increasing length. Formally, given a word of the form $\boldsymbol{uvvw}$ where $|\boldsymbol{v}| = k'$, we say that $\boldsymbol{v}$ is a *k'-duplicate. Removing the k'-duplicate* $\boldsymbol{v}$ yields the word $\boldsymbol{uvw}$. Therefore, to compute $\leqslant$2-root, we simply remove all 1-duplicates from $\boldsymbol{x}$ and then all 2-duplicates from the result. If we want to compute $\leqslant$3-root, then we remove all 3-duplicates from the $\leqslant$2-root.

A formal description of the algorithm in Algorithm 2. Since removing all k'-duplicates may be performed in linear time, the algorithms run in linear-time.

ALGORITHM 2: Finding the $\leqslant k$-root

procedure FIND-$\leqslant$2-ROOT($\boldsymbol{x}$)
Input: $\boldsymbol{x} \in \Sigma_q^*$
Output: $\boldsymbol{r}$, where $R_{\leqslant 2}(\boldsymbol{x}) = \{\boldsymbol{r}\}$.
    $\boldsymbol{r} \leftarrow \boldsymbol{x}$
    Remove all 1-duplicates in $\boldsymbol{r}$ using REMOVEDUPLICATES($\boldsymbol{r}$, 1)
    Remove all 2-duplicates in $\boldsymbol{r}$ using REMOVEDUPLICATES($\boldsymbol{r}$, 2)

procedure FIND-$\leqslant$3-ROOT($\boldsymbol{x}$)
Input: $\boldsymbol{x} \in \Sigma_q^*$
Output: $\boldsymbol{r}$, where $R_{\leqslant 2}(\boldsymbol{x}) = \{\boldsymbol{r}\}$.
    $\boldsymbol{r} \leftarrow \boldsymbol{x}$
    Remove all 1-duplicates in $\boldsymbol{r}$ using REMOVEDUPLICATES($\boldsymbol{r}$, 1)
    Remove all 2-duplicates in $\boldsymbol{r}$ using REMOVEDUPLICATES($\boldsymbol{r}$, 2)
    Remove all 3-duplicates in $\boldsymbol{r}$ using REMOVEDUPLICATES($\boldsymbol{r}$, 3)

procedure REMOVEDUPLICATES($\boldsymbol{r}$,k)
    while $i + 2k - 1 \leqslant |\boldsymbol{r}|$ **do**
        if the substring of length k starting at index i is equal to the substring of length k starting at index $i + k$ **then**
            Remove the substring of length k starting at index i
        else
            $i \leftarrow i + 1$

B ON THE SUM OF LENGTHS OF ALL PREFIXES

Recall in Section 3.2, the running time of Algorithm 1 is linear in the sum of the lengths of all prefixes that are computed in all recursive calls. Therefore, if this sum is linear in the lengths of the original words, then Algorithm 1 runs in linear time. Specifically, we establish the following proposition.

PROPOSITION B.1. *Let $\boldsymbol{x} \in \Sigma_q^m$. Let $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_s$ be all extended regions of $\boldsymbol{r}$ in $\boldsymbol{x}$ computed in Line 12 of Algorithm 1. Then $\sum_{i=1}^{s} |\boldsymbol{p}_i| \leqslant 3m$.*

PROOF. Let $\boldsymbol{x} = x_1 x_2 \cdots x_m$. To establish the proposition, it suffices to show that each symbol x_j with $1 \leqslant j \leqslant m$ appears[4] in at most three extended regions $\boldsymbol{p}_i, \boldsymbol{p}_{i+1}, \boldsymbol{p}_{i+2}$ for some $1 \leqslant i \leqslant s - 2$.

Let i be the smallest index such that x_j appears in the extended region $\boldsymbol{p}_i$ and let $\boldsymbol{u}_i$ be the suffix of $\boldsymbol{x}$ computed in Line 13 of Algorithm 1 in the same recursive call. Note that $\boldsymbol{p}_{i+1}$ is a prefix of $\boldsymbol{u}_i$ and $\boldsymbol{u}_{i+1}$ is a proper suffix of $\boldsymbol{u}_i$. If x_j is still in $\boldsymbol{p}_{i+1}$, which is a prefix of $\boldsymbol{u}_i$, then x_j must belong

[4]In this proof, when we refer to a symbol x_j, we refer specifically to the symbol at index j. More formally, we may rewrite the word $\boldsymbol{x}$ as $(x_1, 1)(x_2, 2) \cdots (x_m, m)$.

to the overlapping substring of $\boldsymbol{p}_i$ and $\boldsymbol{u}_i$ (see Figure 1). Suppose that $\boldsymbol{r}_i$ is the current root and $\mathrm{Reg}(\boldsymbol{r}_i) = \boldsymbol{w}(abc)^\ell ab$. Then the overlapping substring is of the form ab^t for some $t \geqslant 1$, and we have the following cases.

- $x_j = a$. Then x_j is necessarily the first symbol of $\boldsymbol{u}_i$, and hence the first symbol of $\boldsymbol{p}_{i+1}$. Therefore, in the next recursive call, x_j is no longer in $\boldsymbol{u}_{i+1}$,which is a proper suffix of $\boldsymbol{u}_i$ and hence not in $\boldsymbol{p}_{i+2}$. In other words, x_j appears only in $\boldsymbol{p}_i$ and $\boldsymbol{p}_{i+1}$.
- $x_j = b$ for some b in the overlapping substring. To get $\boldsymbol{u}_{i+1}$, at least ab^{t-1} is removed from the prefix of $\boldsymbol{u}_i$. If exactly ab^{t-1} is removed from the prefix of $\boldsymbol{u}_i$ to get $\boldsymbol{u}_{i+1}$, and x_j is the last b in the overlapping substring of $\boldsymbol{p}_i$ and $\boldsymbol{u}_i$, then x_j still appears in $\boldsymbol{u}_{i+1}$ and hence still appears in $\boldsymbol{p}_{i+2}$, but not in $\boldsymbol{u}_{i+2}$. In this case, x_j appears in $\boldsymbol{p}_i$, $\boldsymbol{p}_{i+1}$ and $\boldsymbol{p}_{i+2}$, but not $\boldsymbol{p}_{i+3}$. Otherwise, if x_j is not the last b in the overlapping substring of $\boldsymbol{p}_i$ and $\boldsymbol{u}_i$, x_j is no longer in $\boldsymbol{u}_{i+1}$, and hence x_j does not appear in $\boldsymbol{p}_{i+2}$. Hence, in this case, x_j appears in $\boldsymbol{p}_i, \boldsymbol{p}_{i+1}$, but not $\boldsymbol{p}_{i+2}$.

Therefore, from the two cases above, x_j appears in at most three consecutive extended regions. ■

C PROOF OF PROPOSITION 4.4

Let $\boldsymbol{x}$ be a word of length n whose root is $\boldsymbol{r}$. Suppose that $\boldsymbol{r}$ has length i and m regions and we set $\mathrm{Label}(\boldsymbol{x}) = (\boldsymbol{r}, (c_1, \delta_1), (c_2, \delta_2), \ldots, (c_m, \delta_m))$. To prove Proposition 4.4, we first show certain properties of the label of $\boldsymbol{x}$.

Lemma C.1. *We have that*

$$i + 3(c_1 + c_2 + \cdots + c_m) - m \leqslant n. \tag{2}$$

Proof. Since $\boldsymbol{r} \overset{*}{\underset{\leqslant 3}{\Longrightarrow}} \boldsymbol{x}$, then by Theorem 3.4, there exists $\boldsymbol{x}'$ such that $\boldsymbol{r} \overset{*}{\underset{3_d}{\Longrightarrow}} \boldsymbol{x}'$, and $\boldsymbol{x}' = R_{\leqslant 2}(\boldsymbol{x})$. Let $1 \leqslant j \leqslant m$. Since there are c_j distinct triplets in the jth region of $R_{\leqslant 2}(\boldsymbol{x})$, we have to duplicate the distinct triplet in the jth region of r by $c_j - 1$ times. Therefore, we have $|\boldsymbol{x}'| = |\boldsymbol{r}| + \sum_{j=1}^{m} 3(c_j - 1) = i + 3(c_1 + c_2 + \cdots c_m) - m$. Note that since $\boldsymbol{x}' = R_{\leqslant 2}(\boldsymbol{x})$, we have $|\boldsymbol{x}| \geqslant |\boldsymbol{x}'|$, and hence it yields Equation (2). ■

Lemma C.2. *We have that $\delta_1 = \delta_2 = \cdots = \delta_m = +$, whenever*

$$i + 3(c_1 + c_2 + \cdots + c_m) - m = n. \tag{3}$$

Proof. Since we know that $i + 3(c_1 + c_2 + \cdots c_m) - m$ is the length of $\boldsymbol{x}' = R_{\leqslant 2}(\boldsymbol{x})$, and the equality holds, that means there is no tandem duplication of length at most two in $\boldsymbol{x}$. Hence, a distinct triplet must remain in each of the extended regions of $\boldsymbol{r}$ in $\boldsymbol{x}$ and so, $\delta_j = +$ for all $1 \leqslant j \leqslant m$. ■

Finally, we complete the proof of Proposition 4.4.

Proposition 15. Let $i \leqslant n$. Suppose that $\boldsymbol{r} \in \mathrm{Irr}_{\leqslant 3}(i, 3)$ has m regions. Then,

$$T(n, \boldsymbol{r}) \leqslant U(n, i, m) \triangleq \begin{cases} \binom{(n-i)/3+m}{m} - \binom{(n-i)/3+m-1}{m-1} + 1, & \text{if 3 divides } n-i, \\ \binom{\lfloor (n-i)/3 \rfloor + m}{m}, & \text{otherwise.} \end{cases}$$

Proof. Let C be an $(n \leqslant 3; \boldsymbol{r})$-TD code. For $\boldsymbol{x} \in C$, set $\mathrm{Label}(\boldsymbol{x}) = (\boldsymbol{r}, (c_1, \delta_1), (c_2, \delta_2), \ldots, (c_m, \delta_m))$ and so, $(c_1, c_2, \ldots, c_m)$ is an integer solution to Equation (2) whose entries are all positive. Corollary 4.3(i) implies that these integer solutions are distinct for different $\boldsymbol{x}$ chosen from C. Hence, the number of integer solutions to Equation (2) is an upper bound to the size of C. This number is well known (see Heubach and Mansour [5]) and is given by $\binom{\lfloor (n-i)/3 \rfloor + m}{m}$.

When $n - i$ is divisible by three, we improve this upper bound. Observe that the integer solutions to Equation (3) is a proper subset of Equation (2). Corollary 4.3(ii) then implies that there is at most one $\boldsymbol{x}$ in C whose label yields a positive integer solution to Equation (3). Since the number of positive integer solutions to Equation (3) is $\binom{(n-i)/3+m-1}{m-1}$, we obtain the desired upper bound. ■

D NUMBER OF IRREDUCIBLE WORDS WITH CERTAIN NUMBER OF REGIONS

Recall that $I(i, m)$ denote the number of irreducible words in $\mathrm{Irr}_{\leqslant 3}(i, 3)$ with exactly m regions. In this Appendix, we derive a recursive formula for $I(i, m)$ that allows us to efficiently compute Equation (1).

Let $\mathrm{Irr}(aba, i, m)$ and $\mathrm{Irr}(abc, i, m)$ denote the set of irreducible words of length i with exactly m regions that have two and three distinct symbols, respectively, in their prefixes of length three. Let $I(aba, i, m)$ and $I(abc, i, m)$ denote the sizes of $\mathrm{Irr}(aba, i, m)$ and $\mathrm{Irr}(abc, i, m)$, respectively.

We consider the maps,

$$\begin{aligned}
\Phi_1 &: \mathrm{Irr}(aba, i, m) \to \mathrm{Irr}(abc, i-1, m),\\
\Phi_2 &: \mathrm{Irr}(abc, i-1, m) \to \mathrm{Irr}(aba, i, m),\\
\Psi_1 &: \mathrm{Irr}(abc, i, m) \to \mathrm{Irr}(aba, i-1, m-1) \cup \mathrm{Irr}(aba, i-2, m-1) \cup \mathrm{Irr}(aba, i-3, m-1),\\
\Psi_2 &: \mathrm{Irr}(aba, i-1, m-1) \cup \mathrm{Irr}(aba, i-2, m-1) \cup \mathrm{Irr}(aba, i-3, m-1) \to \mathrm{Irr}(abc, i, m),
\end{aligned}$$

defined via the following rules. In what follows, we set $\boldsymbol{r} = r_1 r_2 \cdots r_{|\boldsymbol{r}|}$ and for distinct elements r_1, r_2, set r^* to be the unique symbol distinct from r_1 and r_2:

$$\Phi_1(\boldsymbol{r}) = \boldsymbol{r} \setminus r_1, \qquad \Phi_2(\boldsymbol{r}) = r_2 \boldsymbol{r},$$

$$\Psi_1(\boldsymbol{r}) = \begin{cases} \boldsymbol{r} \setminus r_1, & \text{if } r_1 \neq r_4,\\ \boldsymbol{r} \setminus r_1 r_2, & \text{if } r_1 = r_4, r_2 \neq r_5,\\ \boldsymbol{r} \setminus r_1 r_2 r_3, & \text{if } r_1 = r_4, r_2 = r_5, \end{cases} \qquad \Psi_2(\boldsymbol{r}) = \begin{cases} r^* \boldsymbol{r}, & \text{if } |\boldsymbol{r}| = i-1,\\ r_2 r^* \boldsymbol{r}, & \text{if } |\boldsymbol{r}| = i-2,\\ r_1 r_2 r^* \boldsymbol{r}, & \text{if } |\boldsymbol{r}| = i-3. \end{cases}$$

Then we check that the maps Φ_1, Φ_2, Ψ_1, and Ψ_2 are well-defined and $\Phi_1 \circ \Phi_2$, $\Phi_2 \circ \Phi_1$, $\Psi_1 \circ \Psi_2$, and $\Psi_2 \circ \Psi_1$ are identity maps on their respective domains. Therefore, all four maps are bijections, and we establish the following recursion. For $m \geqslant 0$ and $i \geqslant 3$,

$$I(aba, i, m) = \begin{cases} I(abc, i-1, m), & i > 3,\\ 0, & \text{if } i = 3, m > 0,\\ 6, & \text{if } i = 3, m = 0. \end{cases}$$

$$I(abc, i, m) = \begin{cases} I(aba, i-1, m-1) + I(aba, i-2, m-1) + I(aba, i-3, m-1), & \text{if } i \geqslant 6, m \geqslant 1,\\ 0, & \text{if } i = 5, m > 2,\\ 6, & \text{if } i = 5, m = 2,\\ 12, & \text{if } i = 5, m = 1,\\ 0, & \text{if } i = 4, m > 1,\\ 12, & \text{if } i = 4, m = 1,\\ 0, & \text{if } i = 3, m > 1,\\ 6, & \text{if } i = 3, m = 1,\\ 0, & \text{if } m = 0. \end{cases}$$

Finally, to compute $I(i, m)$, we have that $I(i, m) = I(aba, i, m) + I(abc, i, m)$.

ACKNOWLEDGMENT

We are grateful for an anonymous reviewer who proposed the definition of label given in Section 4. This definition has led to a more succinct presentation and a spectrum of new results, including Proposition 4.4 and the computation of the size of optimal codes of lengths up to 20.

REFERENCES

[1] Masanori Arita and Yoshiaki Ohashi. 2004. Secret signatures inside genomic DNA. *Biotechnol. Progr.* 20, 5 (2004), 1605–1607.
[2] Farzad Farnoud, Moshe Schwartz, and Jehoshua Bruck. 2016. The capacity of string-duplication systems. *IEEE Trans. Info. Theory* 62, 2 (2016), 811–824.
[3] John W. Fondon and Harold R. Garner. 2004. Molecular origins of rapid and continuous morphological evolution. *Proc. Natl. Acad. Sci. U.S.A.* 101, 52 (2004), 18058–18063.
[4] Dominik Heider and Angelika Barnekow. 2007. DNA-based watermarks using the DNA-Crypt algorithm. *BMC Bioinform.* 8, 1 (2007), 176.
[5] Silvia Heubach and Toufik Mansour. 2009. *Combinatorics of Compositions and Words*. CRC Press.
[6] Siddharth Jain, Farzad Farnoud Hassanzadeh, and Jehoshua Bruck. 2017. Capacity and expressiveness of genomic tandem duplication. *IEEE Trans. Info. Theory* 63, 10 (2017), 6129–6138.
[7] Siddharth Jain, Farzad Farnoud, Moshe Schwartz, and Jehoshua Bruck. 2017. Duplication-correcting codes for data storage in the DNA of living organisms. *IEEE Trans. Info. Theory* 63, 8 (2017), 4996–5010.
[8] Janez Konc and Dušanka Janezic. 2007. An improved branch and bound algorithm for the maximum clique problem. *Proteins* 4, 5 (2007). Retrieved from http://insilab.org/maxclique/.
[9] Eric S. Lander, Lauren M. Linton, Bruce Birren, Chad Nusbaum, Michael C. Zody, Jennifer Baldwin, Keri Devon, Ken Dewar, Michael Doyle, William FitzHugh et al. 2001. Initial sequencing and analysis of the human genome. *Nature* 409, 6822 (2001), 860–921.
[10] Peter Leupold, Carlos Martin-Vide, and Victor Mitrana. 2005. Uniformly bounded duplication languages. *Discrete Appl. Math.* 146, 3 (2005), 301–310.
[11] Peter Leupold, Victor Mitrana, and José M. Sempere. 2003. Formal languages arising from gene repeated duplication. In *Aspects of Molecular Computing*. Springer, 297–308.
[12] Michael Liss, Daniela Daubert, Kathrin Brunner, Kristina Kliche, Ulrich Hammes, Andreas Leiherer, and Ralf Wagner. 2012. Embedding permanent watermarks in synthetic genes. *PloS One* 7, 8 (2012), e42465.
[13] Nicholas I. Mundy and Andreas J. Helbig. 2004. Origin and evolution of tandem repeats in the mitochondrial DNA control region of shrikes (Lanius spp.). *J. Molec. Evol.* 59, 2 (2004), 250–257.
[14] Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM Comput. Surveys* 33, 1 (2001), 31–88.
[15] Seth L. Shipman, Jeff Nivala, Jeffrey D. Macklis, and George M. Church. 2017. CRISPR–Cas encoding of a digital movie into the genomes of a population of living bacteria. *Nature* 547, 7663 (2017), 345.
[16] Michael Sipser. 2006. *Introduction to the Theory of Computation*, vol. 2. Thomson Course Technology, Boston.
[17] Grant R. Sutherland and Robert I. Richards. 1995. Simple tandem DNA repeats and human genetic disease. *Proc. Natl. Acad. Sci. U.S.A.* 92, 9 (1995), 3636–3641.
[18] Karen Usdin. 2008. The biological effects of simple tandem repeats: Lessons from the repeat expansion diseases. *Genome Res.* 18, 7 (2008), 1011–1019.

Received April 2018; revised May 2019; accepted May 2019