Fun Slot Machines and Transformations of Words Avoiding Factors

Marcella Anselmo
Dipartimento di Informatica, University of Salerno, Fisciano (SA), Italy

Manuela Flores
Dipartimento di Informatica, University of Salerno, Fisciano (SA), Italy

Maria Madonia
Dipartimento di Matematica e Informatica, University of Catania, Italy

Abstract

Fun Slot Machines are a variant of the classical ones. Pulling a lever, the player generates a sequence of symbols which are placed on the reels. The machine pays when a given pattern appears in the sequence. The variant consists in trying to transform a losing sequence of symbols in another one, in such a way that the winning pattern does not appear in any intermediate step. The choice of the winning pattern can be crucial; there are "good" and "bad" sequences. The game results in a combinatorial problem on transformations of words avoiding a given pattern as a factor. We investigate "good" and "bad" sequences on a k-ary alphabet and the pairs of words that witness that a word is "bad". A main result is an algorithm to decide whether a word is "bad" or not and to provide a pair of witnesses of minimal length when the word is "bad". It runs in $O(n)$ time with a preprocessing of $O(n)$ time and space to construct an enhanced suffix tree of the word.

2012 ACM Subject Classification Mathematics of computing → Combinatorics on words; Theory of computation → Design and analysis of algorithms; Theory of computation → Formal languages and automata theory

Keywords and phrases Isometric words, Words avoiding factors, Index of a word, Overlap, Lee distance

Digital Object Identifier 10.4230/LIPIcs.FUN.2022.4

Funding Partially Supported by INdAM-GNCS Project 2021, FARB Project ORSA218107 of University of Salerno and TEAMS Project of University of Catania.

1 Introduction

Everybody knows what a slot machine is, some more, some less. In a simple model, there is a screen where a certain number of symbols appears in a line. Every symbol is placed on some reel that "spins" when the game is activated. The machine pays out according to the pattern of symbols displayed when the reels stop spinning, e.g., whether or not it contains a given factor. One of the first model was composed of three spinning reels containing a total of five symbols each: horseshoes, diamonds, spades, hearts and a Liberty Bell; the bell gave the machine its name. Three consecutive bells in a row produced the biggest payoff, ten nickels. Later on, fruit symbols were placed on the reels besides the original bell to refer to the fruit-flavoured gums offered.

Recently, the designers of the *MMM company*, leader in the field of fun machines, have designed a new machine, called *Fun Slot Machine*, and they are evaluating the product. The machine offers a variant to the usual game. After the player has pulled the lever twice without having found the winning pattern, she/he can take the last two sequences of symbols that have appeared - both of which do not contain the winning pattern - and try to transform the first into the second, so that the pattern does not appear this time either in any intermediate step. When the player succeeds, the slot machine pays a consolation prize. The game has


© Marcella Anselmo, Manuela Flores, and Maria Madonia;
licensed under Creative Commons License CC-BY 4.0

rules to follow. Only the symbols where the two sequences differ can be exchanged and the exchange must be done step by step, following the sequence of symbols on the reels. The MMM's designers claim that this game is not just a gamble. There are "good" and "bad" patterns, "good" and "bad" numbers of reels. They propose the following examples to support their claim.

Suppose that the fun slot machine has 6 reels, each with 4 symbols on, A, C, T, G, which stand for the favourite fruit flavours, *Apricot*, *Cherry*, *Tamarind*, and *Grape*. The symbols are placed on each reel in this cyclical order A, C, T, G. Consider the case that the winning pattern is $AGAC$ and the displayed sequence is $AGAAAC$ the first time and $AGATAC$ the second time. The player has lost both times, because $AGAC$ has not appeared. Hence, the player can try the variant to the game, that is, to transform $AGAAAC$ in $AGATAC$ so that $AGAC$ does not appear either. The sequences differ only in their fourth position, where the first sequence contains A, while the second one T. In the first step, symbol A can be swapped either in C or in G, its neighbourhoods on the reel. Unfortunately, for the player (but not for the owner), in both cases $AGAC$ will be displayed. There is no way to win in this case.

The situation would have been different if the number of reels was 5, and not 6. If the displayed sequences were $AGAAA$ and $AGATA$, the exchange of A in the fourth position to C, would have yield $AGAC$, but there is a winning transformation, $AGAAA$ in $AGAGA$ and then in $AGATA$.

Consider now a different scenario. The symbols on the reels are as before, but the winning pattern is AAA. Suppose there are 5 reels and the losing sequences are $ACAGA$ and $AGATA$. The player wants to transform $ACAGA$ in $AGATA$, which differ in their positions 2 and 4. The swap of G into T in the fourth position is safe, no AAA is displayed. Then, there are two possibilities to swap C into G in the second position; either through A or through T. While the first choice would display AAA, the second one would be winning; the transformation of $ACAGA$ in $ACATA$, then in $ATATA$ and finally in $AGATA$, never let AAA appear, and the player will gain the consolation prize. One can show (and we will prove it in the sequel) that when the pattern is AAA, for any number of reels and any pairs of losing sequences, the player has always a possibility to gain.

In some sense, the pattern AAA is "good", because it always leaves a chance of winning, while $AGAC$ is "bad", since, in some situations, it leaves no chance of winning. Now, the question of the MMM's designers is: how should the number of reels, the number and the order of symbols on, and the winning pattern be chosen, whether I am a player or the owner of the machine?

Bad and good sequences of symbols have been investigated in the literature. They were introduced in the binary case, that is when only two symbols are available. A binary word (i.e., a sequence of symbols in a finite set of two symbols) f is called *d-good* if for any pair of words u and v of length d which do not contain the factor f, u can be transformed in v by exchanging one by one the bits on which they differ and generating only words which do not contain f. It is *good* if it is d-good for all d. A binary word f is *bad* if it is not good. The *index* of a binary bad word is the threshold d from which the word is no longer d-good, and a pair of words (u, v) showing that the word is not good is called a pair of *witnesses* for the bad word. Recently, good and bad words have been considered in the case of larger sets of symbols where they are referred to as isometric and non-isometric words; see [1], and [2] on quaternary words. Also, binary bad words have been considered in the two-dimensional setting, and bad pictures have been investigated [3].

The fun slot machine problem thus concerns transformations of words that avoid a given factor. Actually, it is not only a problem in combinatorics of words. It can be stated as a problem on some graphs, called k-ary n-cubes, introduced in [12], and their isometric sub-graphs. More specifically, it concerns a problem on k-ary n-cube avoiding a k-ary word f [2].

Let us mention such framework in this introduction, while it will be no more considered in the rest of the paper. The k-ary n-cube, Q_n^k, is a graph with k^n vertices, each associated to a word of length n over a k-ary alphabet identified with $\mathbb{Z}_k = \{0, 1, \ldots, k-1\}$. Two vertices in Q_n^k are adjacent whenever their associated words differ in exactly one position, and the mismatch is given by two symbols x and y, with $x = (y \pm 1) \bmod k$. Special cases include rings (when $n = 1$), hypercubes Q_n (when $k = 2$) and tori. They have been introduced in [12], in the context of interconnection networks. The binary case ($k = 2$) has been extensively investigated [6]. In order to obtain some variants of hypercubes such that the number of vertices increases slower than in a hypercube, Hsu introduced Fibonacci cubes [7]. They received a lot of attention afterwards (see [9] for a survey). These notions have been then extended to define the generalized Fibonacci cube $Q_n(f)$ [8]. It is the subgraph of the hypercube Q_n obtained by considering only vertices associated to binary words that do not contain a given word f as a factor. In this framework, a binary word f is good when, for any $n \geq 1$, $Q_n(f)$ can be isometrically embedded into Q_n, and bad, otherwise [10]. More generally, given a k-ary word f, the *k-ary n-cube avoiding f*, $Q_n^k(f)$, is obtained from Q_n^k by elimination of the vertices containing f as a factor. Good and bad words are investigated in this more general setting in [2], where they are referred to as isometric and non-isometric words.

Coming back to the fun slot machines, f represents the winning pattern, u and v the displayed losing sequences, and their length is the number of reels. The goal is to find a transformation of u in v that changes only the positions where u and v differ and f is never displayed. If f is good then there is always a chance of winning. If it is bad, in the situation where the number of reels is greater than or equal to the index, and u and v are witnesses for f, then there is no chance of winning!

The main result in this paper is an algorithm to test whether a word is good or not. Further, in case the word is bad, the algorithm provides its index and a pair of witnesses of length equal to the index. Note that, when $k > 4$, no good word exists and any bad word has index equal to its length [1]. Therefore, the computation of the index is given for $k = 2, 3, 4$. It is based on the construction of the pairs of witnesses for f given in [1, 2] to show Theorem 4. The construction generalizes to k-ary words the one given for binary words in [13]. We will revisit it, while highlighting some valuable features and adding some more results. We will show that the index is the minimal length of the above constructed witnesses. Moreover, the index of a quaternary word can be directly computed from the word, without going through its binary representation, as it was in [2].

The algorithm presented in this paper runs in linear time and space, for k-ary words (with $k = 2, 3, 4$). This complexity can be achieved thanks to a preprocessing of linear time and space for computing the suffix tree of the word and enhancing it in order to answer Lowest Common Ancestor (LCA) queries in constant time. An example of execution of the algorithm is provided. The first part of the algorithm follows the one provided in a very recent paper [4] to efficiently check whether a word is isometric. This algorithm is based on the characterization in [2] and applies some methods of the pattern matching with mismatches. Note that the algorithm in this paper not only checks whether a word is isometric, but it also provides its index and a pair of witnesses of minimal length, while keeping the same linear

complexity. A cubic time algorithm for the computation of the index and some related words was given in [13] for binary words. The algorithm presented here works for k-ary words, with $k = 2, 3, 4$, while improving the time complexity.

To conclude, observe that in the previously mentioned examples, $AGAC$ is a bad word, 6 is its index, and $AGAAAC$ and $AGATAC$ are witnesses for $AGAC$; that's why there is no possibility of winning the game. On the other hand, AAA, or more generally a sequence of (three) equal symbols, is good. Could it have been for this reason that, more than a century ago, the Liberty Bell machine paid the maximum when three bells in a row were displayed?

2 Fun Slot Machines and Isometric Words

A *Fun Slot Machine* is composed of d reels that can spin in both directions. Each reel carries k symbols, s_1, s_2, ... , s_k. Let Σ be the set of all symbols, and s_1, s_2, ... , s_k be the order in which symbols follow in each reel. This means that when a reel is spinning in a clockwise direction, s_i appears after s_{i-1}, for $i = 2, ..., k$, and s_1 after s_k; vice versa in the opposite direction. The unique winning pattern is f of length $n < d$. The player inserts the coin and then pulls the lever. The displayed sequence u of d symbols is called a word or string over Σ of length d. The problem is whether the shorter word f is a factor of u or not. If f is not a factor of u, we say that u is f-free or that u avoids the factor f.

Let us formalize the problem with the terminology of the combinatorics of words. First, let us recall some preliminary notions.

Let Σ be an alphabet and $|\Sigma| = k$. Throughout the paper, Σ will be identified with $\mathbb{Z}_k = \{0, 1, \ldots, k-1\}$, the ring of integers modulo k. A word (or string) $f \in \Sigma^*$ of length n is $f = x_1 x_2 \cdots x_n$, where $x_1, x_2, \ldots, x_n$ are symbols in Σ. The set of words over Σ of length n is denoted Σ^n. Let $f[i]$ denote the symbol of f in position i, i.e. $f[i] = x_i$. Then, $f[i..j] = x_i \cdots x_j$, for $1 \leq i \leq j \leq n$, is a factor of f. A word $s \in \Sigma^*$ is said f-free if it does not contain f as a factor. The prefix of f of length l is $pre_l(f) = f[1..l]$; while the suffix of f of length l is $suf_l(f) = f[n-l+1..n]$. When $pre_l(f) = suf_l(f)$ then $pre_l(f)$ is referred to as an overlap of f of length l.

Let $u, v \in \Sigma^*$ be two words of the same length. Then, the *Hamming distance* $dist_H(u, v)$ between u and v is the number of positions at which u and v differ. The *Lee distance* between two words $u, v \in \mathbb{Z}_k^n$, $u = x_1 \cdots x_n$ and $v = y_1 \cdots y_n$ is

$$dist_L(u, v) = \sum_{i=1}^{n} min(|x_i - y_i|, k - |x_i - y_i|).$$

In the sequel, Σ will denote a generic alphabet of cardinality k, while Δ to denote the quaternary alphabet $\Delta = \{A, C, T, G\}$, referred to as the *genetic alphabet*. Symbols A and T (C and G, resp.) will be called *complementary symbols*, in analogy to the Watson-Crick complementary bases they represent. The alphabet Δ will be identified with $\mathbb{Z}_4$, in such a way that A, C, T, and G will be identified with 0, 1, 2, and 3, respectively. Therefore, pairs of complementary symbols have Lee distance 2, whereas pairs of distinct non-complementary symbols have Lee distance 1.

Let us now define what we have called "good" and "bad" word in the Introduction. To be more precise and to avoid ambiguity with similar definitions in other papers, from now on, "good" and "bad" words will be referred to as "Lee-isometric" and "Lee-non-isometric" words, respectively. The definitions are based on the process of transforming a word in another one of equal length, changing one symbol at a time.

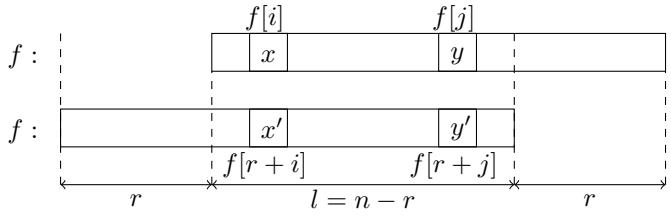

Figure 1 The word f and its 2-error overlap of length l.

Let f be a word in Σ^n. A *Lee-transformation* of length h from u to v is a sequence of words $w_0, w_1, \ldots, w_h$ such that $w_0 = u$, $w_h = v$, and for any $i = 0, 1, \cdots, h-1$, $dist_L(w_i, w_{i+1}) = 1$. The Lee-transformation is *f-free* if for any $i = 0, 1, \cdots, h$, the word w_i is f-free. The word $f \in \Sigma^n$ is *Lee-isometric* if for all $d \geq n$, and f-free words u, $v \in \Sigma^d$, there is an f-free Lee-transformation from u to v of length equal to $dist_L(u, v)$. A word is *Lee-non-isometric* if it is not Lee-isometric. Note that there exists an f-free Lee-transformation from u to v if and only if there exists an f-free Lee-transformation from v to u.

A pair (u, v) of words u, $v \in \Sigma^d$ is referred to as a pair of *Lee-witnesses* for f, if u and v are f-free words and there does not exist an f-free Lee-transformation from u to v of length equal to $dist_L(u, v)$. If f is Lee-non-isometric then its *Lee-index*, denoted by $I_L(f)$, is the shortest length d of u, v such that (u, v) is a pair of Lee-witnesses for f. The *Lee-index* of a Lee-isometric word is defined ∞.

▶ **Example 1.** Let Δ be the quaternary genetic alphabet, $f = ACT$, $u = ACCCT$, and $v = ACGCT$. Observe that $dist_L(u, v) = 2$, since they differ in their third position only and $dist_L(C, G) = 2$. The sequences $ACCCT, ACACT, ACGCT$ and $ACCCT, ACTCT, ACGCT$ are two Lee-transformations from u to v of length equal to $dist_L(u, v) = 2$; they are not f-free. Actually, no f-free Lee-transformation exists from u to v. This shows that ACT is Lee-non-isometric and that (u, v) is a pair of Lee-witnesses for ACT.

Let us state the following definition (see Figure 1).

▶ **Definition 2.** *Let Σ be a k-ary alphabet, $f \in \Sigma^n$, and q be an integer $1 \leq q \leq n$. The word f has a* q-Lee-error overlap *of length l, if $dist_L(pre_l(f), suf_l(f)) = q$. Its* shift *is $r = n - l$; its* error positions *are the m positions where $pre_l(f)$ differs from $suf_l(f)$.*

▶ Remark 3. Using the notations in the previous definition, if f has a q-Lee-error overlap of length l, then $m \leq l, q$. In particular, when $k = 4$ and $q = 2$, then $m = 1$ or $m = 2$. The case $m = 1$ holds if $pre_l(f)$ and $suf_l(f)$ differ in exactly one position and the error is given by a pair of complementary symbols. For example, $f = AGAC \in \Delta^4$ has a 2-Lee-error overlap of length $l = 2$. Indeed, $m = 1$ and $dist_L(AG, AC) = 2$.
If $m = 2$ then $pre_l(f)$ and $suf_l(f)$ differ in two different positions i and j and the errors are given by pairs of non-complementary symbols.

Next theorem, proved in [1, 2], provides a characterization of Lee-isometric words, which is fundamental to test whether a word is Lee-isometric or not. Furthermore, it allows us to restrict the investigation to k-ary alphabets with $k = 2, 3, 4$.

▶ **Theorem 4** ([1, 2])**.** *Let Σ be a k-ary alphabet and $f \in \Sigma^*$.*
- *f is Lee-isometric if and only if it has no 2-Lee-error overlap, when $k = 2, 3, 4$*
- *f is never Lee-isometric, when $k > 4$.*

The proof of Theorem 4 is constructive. In the case that $k = 2, 3, 4$ and f has a 2-Lee-error overlap, the proof provides a pair of Lee-witnesses showing that f is Lee-non-isometric. Applying Theorem 4, one can show that, when $k = 2, 3, 4$, the Lee-index of a Lee-non-isometric word f satisfies $n + 1 \leq I_L(f) \leq 2n - 1$ [2].

3 Computing the Lee-index of a Lee-non-isometric Word

The Lee-index of a "bad" word has been introduced as a threshold on the length of words that witness the "badness" of the word. In this section, we are going to show how to compute the Lee-index of a "bad" - actually, Lee-non-isometric - word. The results prove the correctness of the algorithm in the next section.

Let $\Sigma = \{0, 1, \ldots, k-1\}$, $f = f_1 f_2 \cdots f_n$ be a word over Σ, $u, v \in \Sigma^d$ be f-free words, and $h, j \in \Sigma$. The *reverse* of f is $f^R = f_n \cdots f_2 f_1$. The *h-shift* of j is $j^{S(h)} = (j + h) \bmod k$, while the h-shift of f is $f^{S(h)} = f_1^{S(h)} f_2^{S(h)} \cdots f_n^{S(h)}$. When $k = 2$, the 1-shift of f is its complement. Next result allows us to restrict the domain of strings to be studied.

▶ **Lemma 5.** *Let Σ be a k-ary alphabet and $f \in \Sigma^*$. Then*
- *f is Lee-isometric if and only if f^R is Lee- isometric*
- *for any $h \in \Sigma$, f is Lee-isometric if and only if $f^{S(h)}$ is Lee-isometric.*

Proof. Suppose that f is Lee-isometric and let u, $v \in \Sigma^d$ be f^R-free words, for some $d \geq n$. Clearly, u^R, $v^R \in \Sigma^d$ are f-free words and $dist_L(u^R, v^R) = dist_L(u, v)$. Since f is Lee-isometric, then there is an f-free Lee-transformation from u^R to v^R of length equal to $dist_L(u^R, v^R)$, say $w_0 = u^R, w_1, \ldots, w_h = v^R$. Since w_i is f-free, for $1 \leq i \leq h$, then w_i^R is f^R-free, for $1 \leq i \leq h$ and $w_0^R, w_1^R, \ldots, w_h^R$ is an f^R-free Lee-transformation from u to v of length equal to $dist_L(u, v)$. The same reasoning applied to f^R shows that the converse is true, since $(f^R)^R = f$. The second claim can be proved by a similar reasoning, noting that, for any $u, v \in \Sigma^*$, $dist_L(u, v) = dist_L(u^{S(h)}, v^{S(h)})$ and that u is f-free if and only if $u^{S(h)}$ is $f^{S(h)}$-free. ◀

Let us show how to compute the Lee-index of a Lee-non-isometric k-ary word f. Recall that, when $k > 4$, any word is Lee-non-isometric and its Lee-index is equal to its length [1]. Therefore, from now on, all considerations are done only for alphabets of cardinality $k = 2, 3, 4$. The computation of the Lee-index is based on the construction of the pairs of witnesses for f given in [1, 2] to show Theorem 4. The construction generalizes to k-ary words the one given for binary words in [13]. It will turn out that the Lee-index is the minimal length of such witnesses. Let us revisit this construction of pairs of witnesses, while highlighting some valuable features and adding some more results. First, let us state the following notation. Refer to Figures 2 and 3 for the construction of α_r, β_r, η_r, and γ_r.

Notation. Let Σ be a k-ary alphabet and $f \in \Sigma^n$ have a 2-Lee-error overlap of length l and shift $r = n - l$. Let i, j, with $1 \leq i \leq j \leq l$, be error positions in f (possibly, $i = j$). Then,
- $f^{(i)}$ is the word obtained from f replacing $f[i]$ by $f[r + i]$
- if r is even, $t = j + r/2$, and $i \neq j$ then $f^{(j,t)}$ is the word obtained from f replacing $f[j]$ with $f[r + j]$, and $f[t]$ by $f[i]$
- $f^{(i,-)}$ is the word obtained from f replacing $f[i]$ with $(f[i] - 1) \bmod k$
- $f^{(i,+)}$ is the word obtained from f replacing $f[i]$ with $(f[i] + 1) \bmod k$
- $\alpha_r(f) = pre_r(f) f^{(i)}$ and $\beta_r(f) = pre_r(f) f^{(j)}$
- $\alpha'_r(f) = pre_r(f) f^{(i,-)}$ and $\beta'_r(f) = pre_r(f) f^{(i,+)}$
- if r is even, $\eta_r(f) = pre_r(f) f^{(i)} suf_{r/2}(f)$
- if r is even, $\gamma_r(f) = pre_r(f) f^{(j,t)} suf_{r/2}(f)$.

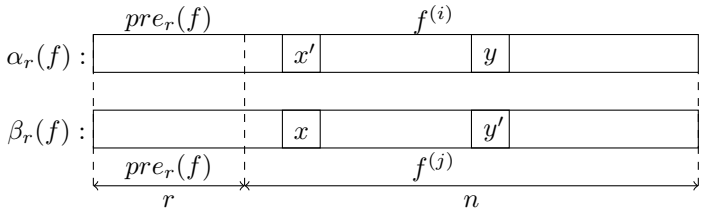

Figure 2 The words $\alpha_r(f)$ and $\beta_r(f)$.

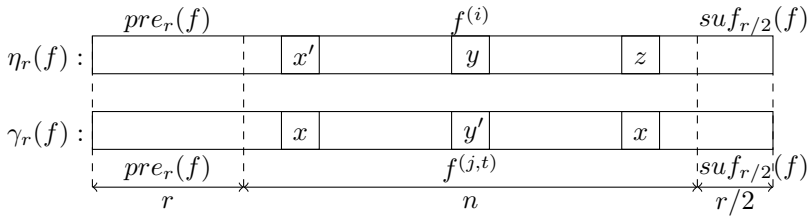

Figure 3 The words $\eta_r(f)$ and $\gamma_r(f)$.

The words introduced in the previous notation will constitute the pairs of witnesses for f. Note that such words are constructed in such a way to be f-free, unless for $\beta_r(f)$. When $\beta_r(f)$ is f-free then $(\alpha_r(f), \beta_r(f))$ is a pair of witnesses for f. Otherwise, it becomes necessary to consider the other words as above introduced. It turns out that $\beta_r(f)$ contains f as a factor if and only if the following *Condition*$^+$ holds [2].

▶ **Definition 6.** *Let Σ be a k-ary alphabet and $f \in \Sigma^n$ have a 2-Lee-error overlap of length l, shift $r = n - l$ and error positions i, j, with $1 \leq i < j \leq l$. We say that the 2-Lee-error overlap satisfies Condition$^+$ if r is even, $j - i = r/2$, $f[r + i] = f[r + j]$, and $f[i..i + r/2 - 1] = f[j..j + r/2 - 1]$.*

Let us start considering the case $k = 2, 3$. In this case, the Lee distance of two words coincides with the number of positions where they differ. Then, a 2-Lee-error overlap is given by two distinct error positions. This is no more true when $k = 4$; the quaternary case will be treated later.

Consider a Lee-non-isometric word $f \in \Sigma^n$, with $|\Sigma| = k$ and $k = 2, 3$. Then, f has a 2-Lee-error overlap. Actually, it may have more than one 2-Lee-error overlap. For any 2-Lee-error overlap, it is possible to construct a pair of witnesses for f as follows.

▶ **Proposition 7** ([2])**.** *Let f be a Lee-non-isometric k-ary word, $k = 2, 3$. Consider a 2-Lee-error overlap of f, of length l, shift $r = n - l$, and error positions i, j, with $1 \leq i < j \leq l$.*

1. *If the 2-Lee-error overlap does not satisfy Condition$^+$ then $(\alpha_r(f), \beta_r(f))$ is a pair of witnesses for f*
2. *If the 2-Lee-error overlap satisfies Condition$^+$ and $1 \leq i \leq r/2$ then $(\eta_r(f), \gamma_r(f))$ is a pair of witnesses for f*
3. *If the 2-Lee-error overlap satisfies Condition$^+$ and $i > r/2$ then f has a 2-Lee-error overlap of shift $r' > r$, which does not satisfy Condition$^+$ and $(\alpha_{r'}(f), \beta_{r'}(f))$ is a pair of witnesses for f.*

Let us show some examples of the construction in Proposition 7 for a ternary and a binary alphabet, respectively.

▶ **Example 8.** Consider the ternary alphabet $\Sigma = \{0, 1, 2\}$. Let $f = 021 \in \Sigma^3$. The word f has a 2-Lee-error overlap of length 2; here $n = 3$, $r = 1$, and $n - r = 2$. Hence, $f = 021$ is Lee-non-isometric from Theorem 4. Following the proof of the same theorem, let us exhibit a pair (α, β) of Lee-witnesses for f. We have that $pre_2(f)$ disagrees from $suf_2(f)$ in positions $i = 1$ and $j = 2$, and $f[i] = 0$, $f[j] = 2$, $f[r + i] = 2$ and $f[r + j] = 1$. Then, $\alpha_r(f) = pre_r(f)f^{(i)} = 0221$ and $\beta_r(f) = pre_r(f)f^{(j)} = 0011$. The words $\alpha_r(f)$ and $\beta_r(f)$ are f-free words and $dist_L(\alpha_r(f), \beta_r(f)) = 2$. Moreover, there is no f-free Lee-transformation from $\alpha_r(f)$ to $\beta_r(f)$ of length $dist_L(\alpha_r(f), \beta_r(f)) = 2$. Indeed, if we replace $\alpha_r(f)[2]$ with $\beta_r(f)[2] = 0$ then f occurs at position 2 of $\alpha_r(f)$; if we replace $\alpha_r(f)[3]$ with $\beta_r(f)[3] = 1$ then f occurs at position 1.

▶ **Example 9.** Consider the binary alphabet $B = \{0, 1\}$. Let $f = 0011 \in B^4$. The word f has a 2-Lee-error overlap of length 2. Hence, $f = 0011$ is Lee-non-isometric from Theorem 4. Note that the 2-Lee-error overlap of f satisfies $Condition^+$. Indeed, we have $i = 1$, $j = 2$, $r = 2$, $f[r + i] = f[3] = 1 = f[4] = f[r + j]$ and $f[i] = 0 = f[j]$. Therefore, following the proof of Theorem 4 in the case that the 2-Lee-error overlap of f satisfies $Condition^+$ and $1 \leq i \leq r/2$, let us set $t = (r/2) + j = 3$ and let us consider the two words $\eta_r(f) = pre_r(f)f^{(i)}suf_{r/2}(f) = 0010111$ and $\gamma_r(f) = pre_r(f)f^{(j,t)}suf_{r/2}(f) = 0001011$. The words $\eta_r(f)$ and $\gamma_r(f)$ are f-free words and $dist_L(\eta_r(f), \gamma_r(f)) = 3$. Moreover, there is no f-free Lee-transformation from $\eta_r(f)$ to $\gamma_r(f)$ of length $dist_L(\eta_r(f), \gamma_r(f)) = 3$. Indeed, if we replace $\eta_r(f)[3]$ with $\gamma_r(f)[3] = 0$ then f occurs at position 3 of $\eta_r(f)$; if we replace $\eta_r(f)[4]$ with $\gamma_r(f)[4] = 1$ then f occurs at position 1 and if we replace $\eta_r(f)[5]$ with $\gamma_r(f)[5] = 0$ then f occurs at position 4.

Consider now the case of the quaternary alphabet $\Delta = \{A, C, T, G\}$.

The construction of a pair of witnesses for a Lee-non-isometric quaternary word f is obtained in [2] referring to the binary representation of f. Actually, there is an isomorphism between quaternary words and binary words of even length. It is given by the map which associates to A, C, T, G the binary words 00, 01, 11, 10, respectively. Hence, a word $f \in \Delta^n$ will be possibly denoted as $(f)_4$ to stress its belonging to the quaternary alphabet, while its binary representation in $\{0, 1\}^{2n}$ will be denoted as $(f)_2$. The correspondence preserves the Lee distance.

Let $(f)_4 \in \Delta^n$ be a Lee-non-isometric word, r be the shift of a 2-Lee-error overlap of $(f)_4$ and m the number of its error positions. Recall that $m = 1$ or $m = 2$; see Remark 3. Note that $(f)_2$ has 2-Lee-error overlap of shift $2r$. A pair of witnesses for $(f)_4$ is obtained in [1, 2] considering the pair of witnesses for $(f)_2$, constructed as in Proposition 7, and representing it in quaternary. Recall that $(A)_2 = 00$, $(C)_2 = 01$, $(T)_2 = 11$, and $(G)_2 = 10$. For example, if the 2-Lee-error overlap of shift $2r$ of $(f)_2$ fills case 1 of Proposition 7 then $(\alpha_{2r}((f)_2), \beta_{2r}((f)_2))$ is a pair of witnesses for $(f)_2$. Thus, the quaternary representation of this pair of witnesses for $(f)_2$, that is $((\alpha_{2r}((f)_2))_4, (\beta_{2r}((f)_2))_4)$, is a pair of witnesses for $(f)_4$.

Next proposition shows that, indeed, the pair of witnesses for $(f)_4$ as just obtained from a 2-Lee-error overlap can be directly constructed from $(f)_4$, without going through its binary representation. Therefore, let us simply denote by f the quaternary word. Example 11 shows both constructions.

▶ **Proposition 10.** *Let f be a Lee-non-isometric k-ary word, $k = 4$. Consider a 2-Lee-error overlap of f, of length l, shift $r = n - l$, and error positions i, j, with $1 \leq i \leq j \leq l$ ($i = j$ if $m = 1$).*

1. *If $m = 1$, then $(\alpha'_r(f), \beta'_r(f))$ is a pair of Lee-witnesses for f.*

2. *If $m = 2$ then*
   - **a.** *if the 2-Lee-error overlap does not satisfy Condition$^+$, then $(\alpha_r(f), \beta_r(f))$ is a pair of Lee-witnesses for f.*
   - **b.** *If the 2-Lee-error overlap satisfies Condition$^+$ and $i \leq r/2$, then $(\eta_r(f), \gamma_r(f))$ is a pair of Lee-witnesses for f.*
   - **c.** *If the 2-Lee-error overlap satisfies Condition$^+$ and $i > r/2$, then f has a 2-Lee-error overlap of shift $r' > r$, which does not satisfy Condition$^+$ and $(\alpha_{r'}(f), \beta_{r'}(f))$ is a pair of Lee-witnesses for f.*

Proof. Consider $(f)_2$, the binary representation of f. Since f has a 2-Lee-error overlap of shift r, then $(f)_2$ has a 2-Lee-error overlap of shift $2r$; let i, j, with $i < j$, be its error positions.

In the case $m = 1$, we have that $i = j$ and that $f[i]$ and $f[r + i]$ are two complementary symbols. Moreover, we have $s = 2i-1$ and $z = s+1 = 2i$. Suppose $f[i] = A$ and $f[r+i] = T$; the other cases can be treated analogously. We have $(f)_2[s] = 0$, $(f)_2[2r+s] = 1$, $(f)_2[z] = 0$, $(f)_2[2r + z] = 1$. Therefore, the quaternary representation of $(f)_2^{(s)}$ ($(f)_2^{(z)}$, resp.) coincides with the word obtained from f by replacing the symbol A in position i with the symbol G (C, resp.), that is $((f)_2^{(s)})_4 = f^{(i,-)}$ ($((f)_2^{(z)})_4 = f^{(i,+)}$), resp.). Moreover, $pre_{2r}((f)_2)$, represented in the quaternary alphabet, coincides with $pre_r(f)$. Hence, $(\alpha_{2r}((f)_2))_4 = pre_r(f)f^{(i,-)}$ and $(\beta_{2r}((f)_2))_4 = pre_r(f)f^{(i,+)}$ In [2], it is proved that the quaternary representation of $(\alpha_{2r}((f)_2), \beta_{2r}((f)_2))$ is a pair of Lee-witnesses for f. Hence the thesis follows.

In the case $m = 2$, we have that $i < j$, $f[i]$ and $f[r + i]$ ($f[j]$ and $f[r + j]$, resp.) differ because of two non-complementary symbols.

- **a.** If the 2-Lee-error overlap does not satisfy *Condition*$^+$ then the quaternary representation of $(\alpha_{2r}((f)_2)$, $\beta_{2r}((f)_2))$ is a pair of Lee-witnesses for f; see [1]. Suppose $f[i] = T$, $f[r + i] = C$, $f[j] = G$ and $f[r + j] = A$; the other cases can be treated analogously. We have $s = 2i - 1$ and $z = 2j - 1$, $(f)_2[s] = 1$, $(f)_2[2r + s] = 0$, $(f)_2[z] = 1$, $(f)_2[2r + z] = 0$. Therefore, $(f)_2^{(s)}$ ($(f)_2^{(z)}$, resp.), represented in the quaternary alphabet, coincides with the word obtained from f by replacing T (G, resp.) in position i (j, resp.) with C (A, resp.), that is $((f)_2^{(s)})_4 = f^{(i)}$ ($((f)_2^{(z)})_4 = f^{(j)}$, resp.). Moreover, $pre_{2r}((f)_2)$, represented in the quaternary alphabet, coincides with $pre_r(f)$ and the thesis follows.
- **b.** If the 2-Lee-error overlap of f of shift r satisfies *Condition*$^+$ and $i \leq r/2$ then the 2-Lee-error overlap of $(f)_2$ of shift $2r$ satisfies *Condition*$^+$. The proof is given for $f[i] = f[j] = A$ and $f[r + i] = f[r + j] = C$; the other cases can be treated analogously. Then, $s = 2i$ and $z = 2j$. Since the 2-Lee-error overlap of f of shift r satisfies *Condition*$^+$, then r is even, $j - i = r/2$. Therefore, $z - s = 2(j - i) = r$ is even and it is equal to the half of $2r$, the shift of the 2-Lee-error overlap of $(f)_2$. Moreover, from $f[r+i] = f[r+j]$ and $f[i..i+r/2-1] = f[j..j + r/2 - 1]$, it follows $f[2r + s] = f[2r + z]$ and $f[s..s + r - 1] = f[z..z + r - 1]$. At last, $i \leq r/2$, implies $s \leq r$. In this case, from [1], $\eta_{2r}((f)_2)$, $\gamma_{2r}((f)_2)$, represented in the quaternary alphabet, is a pair of Lee-witnesses for f. Recall that $\eta_{2r}((f)_2) = pre_{2r}((f)_2)(f_2)^{(s)}suf_{r/2}((f)_2)$ and $\gamma_{2r}((f)_2) = pre_{2r}((f)_2)(f_2)^{(z,t)}suf_{r/2}((f)_2)$. Then, $((f)_2)^{(s)}$ will be obtained from $(f)_2$ by replacing $(f)_2[s] = 0$ with $(f)_2[2r + s] = 1$. Therefore, its quaternary representation coincides with the word obtained from f by replacing A in position i with C, that is $((f)_2^{(s)})_4 = f^{(i)}$. Analogous considerations show that $(\eta_{2r}((f)_2))_4 = pre_r(f)f^{(i)}suf_{r/2}(f)$ and $(\gamma_{2r}((f)_2))_4 = pre_r(f)f^{(j,t)}suf_{r/2}(f)$ and the thesis follows.
- **c.** This case can be treated as case **a**. ◀

▶ **Example 11.** Let $(f)_4 = AGCT \in \Delta^*$ and, hence, $(f)_2 = 00100111 \in (B^2)^*$. Then, $(f)_4$ has a 2-Lee-error overlap of length $l = 1$ and shift $r = 3$. In this case $m = 1$ and $i = 1$ is the unique error position with $(f)_4[1] = A$ and $(f)_4[4] = T$, that is the error is caused by two complementary symbols. A pair of witnesses for $(f)_4$ can be constructed from a pair of witnesses for $(f)_2$, as follows. Consider the 2-Lee-error overlap of $(f)_2$ of even length $2l = 2$ and shift $2r = 6$. It is caused by two different error positions, $s = 1$ and $z = 2$, since $(f)_2[1] = (f)_2[2] = 0$, while $(f)_2[7] = (f)_2[8] = 1$. Starting from this 2-Lee-error overlap of $(f)_2$, the pair $(\alpha_6((f)_2), \beta_6((f)_2))$ of Lee-witnesses for $(f)_2$ can be obtained. The pair is composed of $\alpha_6((f)_2) = 00100110100111$ and $\beta_6((f)_2) = 00100101100111$. In this case, the 2-Lee-error overlap of $(f)_2$ does not satisfy $Condition^+$, since $z - s = 1$ is different from $r = 3$. Therefore, coming back to the quaternary representation, $(\alpha_3(AGCT), \beta_3(AGCT)) = (AGCGGCT, AGCCGCT)$ is a pair of Lee-witnesses for $(f)_4$.
On the other hand, this pair of witnesses for $AGCT$ can be directly constructed from $AGCT$, without going through the binary representation. Following Proposition 10 in the case $m = 1$, $\alpha_3(f) = pre_3(f)f^{(1,-)}$, $\beta_3(f) = pre_3(f)f^{(1,+)}$. Observe that $f^{(1,-)} = GGCT$ and $f^{(1,+)} = CGCT$. Hence, $\alpha_3(f) = pre_3(f)f^{(1,-)} = AGCGGCT$ and $\beta_3(f) = pre_3(f)f^{(1,+)} = AGCCGCT$. Finally, $(\alpha_3(AGCT), \beta_3(AGCT)) = (AGCGGCT, AGCCGCT)$ is the same pair of Lee-witnesses for $(f)_4$, as previously computed using the binary representation.

Let us come back to the computation of the Lee-index in the general case of a k-ary word with $k = 2, 3, 4$. Proposition 14 proves that the Lee-index is the minimal length of the Lee-witnesses as constructed in Propositions 7 and 10.

▶ Remark 12. Let $f \in \Sigma^*$ and let $u, v \in \Sigma^*$ be two f-free words. Consider any f-free Lee-transformation from u to v of length equal to $dist_L(u, v)$. Then, only symbols in the positions where u and v differ are modified in this transformation. Moreover, at each step of the Lee-transformation, a symbol x can be replaced by y only if $dist_L(x, y) = 1$. Hence, each position i such that $dist_L(u[i], v[i]) = d$ is replaced exactly d times.

▶ Remark 13. Let $f \in \Sigma^*$ be a Lee-non-isometric word and (u, v) be a pair of Lee-witnesses for f with $u, v \in \Sigma^d$ be f-free words. Let $h = dist_L(u, v)$ and suppose h to be minimal. Let $V = \{i_1, i_2, \ldots, i_m\}$, with $1 \leq i_1 < i_2 < \cdots < i_m \leq d$, be the set of all the positions where u and v differ; $m \leq h$. Consider a Lee-transformation of length h, $u = w_0, w_1, \ldots, w_h = v$. Then, for any $j = 1, 2, \ldots, h-1$, w_j has f as a factor. Moreover, for any error position $i \in V$, and any Lee-transformation of u in v of length h which starts changing $u[i]$ (cfr. Remark 12), the word obtained after this first replacement contains an occurrence of f including position i.

▶ **Proposition 14.** *Let f be a Lee-non-isometric k-ary word, $k = 2, 3, 4$. Then, the Lee-index of f, $I_L(f)$, is the minimum length of words $\alpha_r(f)$, $\eta_r(f)$, or $\alpha'_r(f)$, as appropriate, where r is taken over the shifts of all 2-Lee-error overlaps of f.*

Proof. Let f be a Lee-non-isometric k-ary word and $|f| = n$. Let (u, v) be the pair of witnesses of minimal distance $d_L(u, v)$ among all witnesses of minimal length. Note that (u, v) is also of minimal distance. Then, $I_L(f) = |u|$. Let V be the set of all error positions. The minimality of the distance of u and v implies that, when changing in u the symbol in any position $i \in V$, as in a Lee-transformation from u to v, then an occurrence f_i of f appears as a factor; see Remark 13. This f_i covers i and another error position (the same, if $|V| = 1$). The minimality of the length of u implies that the occurrences of f_i for all $i \in V$ completely cover u. Hence, let f_{i_1} be the occurrence that covers position 1 and f_{i_2} be the occurrence that covers position n.

If f_{i_1} and f_{i_2} contain both positions i_1 and i_2, then f has a 2-Lee-error overlap of shift $r = |u| - n$ and error positions i_1 and i_2. Then, u will eventually be $\alpha_r(f)$, $\alpha'_r(f)$, $\beta_r(f)$, or $\beta'_r(f)$, and the claim is proved.

Otherwise, there is another error position $i_3 \in V$ and a corresponding intermediate occurrence f_{i_3} of f. Each occurrence of f must contain two of these error positions. Observe that if an occurrence of f contains the first and the last error position (in non-decreasing order) then it also contains the second one. Then, there are two occurrences of f, say f_i and f_j which contains both error positions i and j; note that i and j cannot be the first and the last error position in this case. Then f has a 2-Lee-error overlap with error positions i and j; let r be its shift. Consider the pair $(\alpha_r(f), \beta_r(f))$, if $i \neq j$, or $(\alpha'_r(f), \beta'_r(f))$, if $i = j$. The length of such words is strictly less than $|u|$. Then, this pair cannot be a pair of witnesses for f, because of the minimality of the length of u. This means that $\beta_r(f)$ ($\beta'_r(f)$, resp.) is not f-free. Hence, $Condition^+$ holds for the 2-Lee-error overlap with shift r and f occurs at position $r/2$ in $\beta_r(f)$ ($\beta'_r(f)$, resp.). Further, the shortest pair of witnesses that can be constructed in such situation with three occurrences of f covering u is $(\eta_r(f), \gamma_r(f))$ and finally $u = \eta_r(f)$ or $u = \gamma_r(f)$. ◀

Proposition 14 suggests to compute the Lee-index of a word considering all its 2-Lee-error overlaps, obtaining for each 2-Lee-error overlap the corresponding pair of witnesses as in Proposition 7 (if $k = 2, 3$) or in Proposition 10 (if $k = 4$), and then computing the minimal length of a so obtained witness. The algorithm in next section will analyse the possible 2-Lee-error overlaps in increasing order of their shift. Nevertheless, note that it is not possible to stop at the first found 2-Lee-error overlap. The 2-Lee-error overlap that corresponds to the witnesses of minimal length can be a subsequent one, as shown in Example 15.

▶ **Example 15.** Consider the ternary alphabet $\Sigma = \{0, 1, 2\}$. For any $h \geq 0$, let $w = 2^h$ and $f = 0w0w1w1$, with $|f| = n = 3h + 4$. It can be observed that, for any $h \geq 0$, f has two 2-Lee-error overlaps, one of length $h + 2$, for which $d_L(0w0, 1w1) = 2$ and one of length $h + 1$, for which $d_L(w0, 1w) = 2$. Then, for any $h \geq 0$, f is Lee-non-isometric applying Theorem 4. The first pair of Lee-witnesses for f can be constructed starting from the 2-Lee-error overlap of length $l = h+2$ and shift $r = n-l = 2h+2$, following the proof of Theorem 4. This overlap satisfies $Condition^+$ and the corresponding pair of Lee-witnesses for f is $(\eta_r(f), \gamma_r(f))$, with $\eta_r(f) = 0w0w(1w0w1w1)w1$ and $\gamma_r(f) = 0w0w(0w1w0w1)w1$. The length of this pair of Lee-witnesses is $|\eta_r(f)| = |\gamma_r(f)| = 6h + 7$. The second pair of Lee-witnesses for f can be constructed starting from the 2-Lee-error overlap of length $h + 1$, following the proof of Theorem 4 again. This overlap does not satisfies $Condition^+$ and the corresponding pair of Lee-witnesses for f is $(\alpha_r(f), \beta_r(f))$, with $\alpha_r(f) = 0w0w1(2w0w1w1)$ and $\beta_r(f) = 0w0w1(02^{h-1}10w1w1)$. The length of this pair of Lee-witnesses is $|\alpha_r(f)| = |\beta_r(f)| = 5h + 7$. Thus, the 2-Lee-error overlap that corresponds to the witnesses of minimal length is the second one and then it provides the Lee-index $I_L(f) = 5h + 7$.

4 The Algorithm

In this section, the results provided in Section 3 are applied to design an algorithm that computes the Lee-index of a word and yields a pair of Lee-witnesses of minimal length. It is assumed that Σ is a k-ary alphabet, with $k \leq 4$ and $f \in \Sigma^*$ is a finite sequence $f[0]f[1]\cdots f[n-1]$ of symbols in Σ, where n is the length of f and $f[i]$'s are its symbols. Note that now the indices of f start from 0, and not 1, in view of an implementation of the algorithm in the main programming languages. Recall that if f is Lee-isometric then its

Lee-index is ∞, else it is the minimal length of two words u, v, such that (u, v) is a pair of Lee-witnesses for f. Observe that an algorithm to compute the Lee-index and its witnesses is given for binary alphabet in [13]; it runs in $O(n^3)$ time. The algorithm designed in this section runs in $O(n)$ time and space. Finally, note that this algorithm can be easily modified, without changing its complexity, to compute the index and related witnesses of a word, referring to Hamming distance, as considered, for example, in [13], or to other distances, instead of Lee one.

Let us sketch an algorithm which inputs a k-ary non-empty word f of length n, with $k = 2, 3, 4$, and outputs an integer I, that is the Lee-index of f, and a pair (u, v) of Lee-witnesses for f of length I. Its pseudo-code is given by Algorithm 1 and an example follows. The algorithm starts looking for all 2-Lee-error overlaps of f and saving them into a list. This is done by function `TwoErrorOverlaps` which can be computed in time and space $O(n)$, thanks to a preprocessing step which uses an enhanced suffix tree to answer Lowest Common Ancestor (LCA) queries in constant time. If there are not 2-Lee-error overlaps, then the algorithm sets the Lee-index I to ∞. Otherwise, for each 2-Lee-error overlap, it constructs a pair of Lee-witnesses calling the function `WitnessesConstructor`. According to Propositions 7 and 10, the construction depends on whether the $Condition^+$ is satisfied; function `CondPlus` checks this. Then, it outputs the Lee-index I as the minimal length of all these pairs of Lee-witnesses, following Proposition 14. It also outputs a pair (u, v) of Lee-witnesses of length I. Note that the Lee-index of f is upper bounded by $I_L(f) \leq 2n - 1$ in [2]; then I can be initialized as $I = 2n$ (in Line 6). Since the 2-Lee-error overlaps are at most $n - 1$, there are $O(n)$ calls to `WitnessesConstructor`, each running in time $O(1)$. The overall time and space complexity of Algorithm 1 is thus $O(n)$.

▶ **Proposition 16.** *Let Σ be a k-ary alphabet, with $k \leq 4$ and $f \in \Sigma^n$ be a non-empty word of length n. The Lee-index of f and a pair of Lee-witnesses of minimal length for f can be computed in time $O(n)$ with additional $O(n)$ space.*

Proof. Let us analyse the main functions in Algorithm 1.

- `TwoErrorOverlaps` inputs the word f and outputs all lengths of its 2-Lee-error overlaps in the list $2eolens$ and all corresponding error positions in the list $allerrpos$. It is similar to Algorithm 3 in [4], with the difference that Algorithm 3 in [4] checks only if a word f has at least one 2-Lee error overlap, while this function finds all 2-Lee error overlaps of f and stores their lengths in the list $2eolens$. Further, it stores all corresponding error positions in the list $allerrpos$, which is, therefore, a list of lists. It is based on a technique called the Kangaroo method [5, 11], used in designing efficient pattern matching with mismatches algorithms. It computes the number of mistakes in a given alignment by "jumping" from one error to the next. It allows to check, for a given position i in f, whether f has a 2-Lee-error overlap of length $n - i$ in time $O(1)$. To do this, it first computes in time and space $O(n)$ the suffix tree of f enhanced to answer to Lowest Common Ancestor (LCA) queries in time $O(1)$. A call to LCA(i, j) returns the length of the longest common prefix between the suffix of f starting from position i and the one starting from position j. The function `TwoErrorOverlaps` checks whether the suffix starting at i has two mismatches with its prefix of the same length. It uses a variable l which gives the length of the current overlap and a variable d which contains the current Lee distance. They are increased when a mismatch has been found. Since there are at most two LCA queries for a given i, this can be done in $O(1)$ time. Thus, the time and space complexity of `TwoErrorOverlaps` is $O(n)$.

Algorithm 1 Computing the Lee-index and Lee-witnesses for f.

Input: a k-ary non-empty word f of length n, with $k \leq 4$
Output: an integer I, Lee-index of f, and a pair of words (u, v), Lee-witnesses for f of length I

```
(2eolens, allerrpos) ← TwoErrorOverlaps(f);
(u, v) ← (empty, empty);
if 2eolens is empty then
    I ← ∞;
else
    I ← 2(len(f));
    for i ← 0 to len(2eolens) − 1 do
        (utmp, vtmp) ← WitnessesConstructor(f, 2eolens[i], allerrpos[i]);
        if len(u) < I then
            I ← len(u);
            (u, v) ← (utmp, vtmp);
return I, (u, v);

function TwoErrorOverlaps(f):
    (2eolens, allerrpos, n) ← ([], [], len(f));
    for i ← 1 to n − 1 do
        (l, d, allerrpostmp) ← (0, 0, []);
        while d ≤ 2 do
            l ← l + LCA(l, i + l);
            if l < n − i then
                allerrpostmp.append(l + 1);
            if d = 2 and l = n − i then
                2eolens.append(l);
                allerrpos.append(allerrpostmp);
            if d < 2 and l < n − i then
                l ← l + 1;
                d ← d + d_L(f[l], f[i + l]);
            else
                BREAK
    return (2eolens, allerrpos);
end function

function WitnessesConstructor(f, l, errpos):
    (n, r, i) ← (len(f), n − l, errpos[0]);
    if len(errpos) = 1 then
        (falfa1, fbeta1) ← (f, f);
        alfa1[i] = (falfa1[i] − 1) mod 4;
        beta1[i] = (fbeta1[i] + 1) mod 4;
        u ← pre_r(f) + falfa1;
        v ← pre_r(f) + fbeta1;
    else
        j ← errpos[1];
        cplus ← CondPlus(f, r, errpos[0], errpos[1]);
        if cplus = False then
            (falfa, fbeta) ← (f, f);
            falfa[i] = f[r + i];
            fbeta[j] = f[r + j];
            u ← pre_r(f) + falfa;
            v ← pre_r(f) + fbeta;
        else
            if i ≤ r/2 then
                (feta, fgamma) ← (f, f);
                feta[i] = f[r + i];
                fgamma[j] = f[r + j];
                fgamma[r/2 + j] = f[i];
                u ← pre_r(f) + feta;
                v ← pre_r(f) + fgamma;
    return (u, v);
end function

function CondPlus(f, r, i, j):
    (cond1, cond2, cond3) ← (False, False, False);
    if r mod 2 = 0 then
        if j − i = r/2 then
            cond1 ← True;
            if f[r + i] = f[r + j] then
                cond2 ← True;
                if LCA(i, j) ≥ r/2 then
                    cond3 ← True;
    return (cond1 and cond2 and cond3);
end function
```

- `WitnessesConstructor` inputs the word f, the length l of a 2-Lee error overlap of f, its corresponding error positions in the list *errpos* and outputs a pair (u, v) of Lee-witnesses for f. This function constructs the pair (u, v), according to Proposition 10, in two different ways, following that the 2-Lee-error overlap is caused by two complementary symbols (i.e., $m = 1$ and the list *errpos* has only one element) or by two non-complementary symbols (i.e., $m = 2$ and the list *errpos* has two elements). Note that the first case may occur only if the alphabet cardinality is $k = 4$. In this case, the function constructs u by appending to the prefix of f of length $r = n - l$ the word $f^{(i,-)}$ as defined in the list of Notation given in Section 3. Similarly, it constructs v, this time appending $f^{(i,+)}$. The second case follows case $m = 2$ in Proposition 10. It has other two subcases following that the function `CondPlus` returns $False$ or $True$. In the first subcase, the pair (u, v) may be constructed according to case 2.**a** of Proposition 10. In the second case, if $i \leq r/2$, it may be constructed as case 2.**b**. Otherwise, it is case 2.**c**, and there is nothing else to do because it is proved that f always has another 2-Lee-error overlap of (even) length smaller than l, and thus the pair (u, v) will be constructed as in case 2.**a** in a subsequent call of the function. Since `CondPlus` runs in $O(1)$ time, all these instructions can be executed in constant time. Thus, the time complexity of `WitnessesConstructor` is $O(1)$.
- `CondPlus` inputs the word f, the integers r, i and j, where i and j are the error positions in the 2-Lee-error overlap of shift r. This function outputs $True$ if $Condition^+$ is verified, $False$, otherwise. Recall that $Condition^+$ is defined in the list of Notation in Section 3. Note that $Condition^+$ may be $True$ only if r is even. If r is even, then the function checks the other conditions in *cond*1, *cond*2, and *cond*3. In particular, *cond*3 is $True$ iff $f[i..i + r/2 - 1] = f[j..j + r/2 - 1]$. This check is done in $O(1)$ time testing if $LCA(i, j) \geq r/2$, rather than in $O(r)$ time comparing all the symbols. Thus, all the instructions of `CondPlus` can be done in $O(1)$ time.

In summary, the overall time complexity of Algorithm 1 can be obtained as the sum of the cost of `TwoErrorOverlaps` and at most $n-1$ times the cost of `WitnessesConstructor`. Thus, it is $O(n) + O(n) = O(n)$. The space complexity of Algorithm 1 is due to `TwoErrorOverlaps`, thus it is $O(n)$. ◀

▶ **Example 17.** Let us run Algorithm 1 to compute the Lee-index and a pair of Lee-witnesses for $f = f[0]f[1] \cdots f[5] = AGATAC$. It starts calling the function `TwoErrorOverlaps` with input $f = AGATAC$. This function finds two 2-Lee-error overlaps. The first one is of length 4, where the errors are in positions $1, 3$ and are caused by non-complementary symbols; in fact, $f[1] = G \neq f[3] = T$ and $f[3] = T \neq f[5] = C$. The second one is of length 2, and has a unique error position 1; in fact, $f[1] = G \neq f[5] = C$ and further $dist_L(G, C) = 2$, since G and C are complementary symbols. Thus, `TwoErrorOverlaps` outputs $2eolens = [4, 2]$ and $allerrpos = [[1, 3], [1]]$.

Then, coming back to Line 3 of the main algorithm, because $2eolens$ is not empty, the algorithm initializes the output variable $I = 12$. For $i = 0$ to 1 the algorithm calls twice the function `WitnessesConstructor`.

The first call takes as input $(AGATAC, 4, [1, 3])$ and sets $n = 6, r = 2, i = 1$. Because $len(errpos) = 2$, then $j = 3$ and $cplus = False$ after calling `CondPlus`$(AGATAC, 2, 1, 3)$. Thus, the function `WitnessesConstructor` computes and outputs $u = AGATATAC$ and $v = AGAGACAC$, obtained as $\alpha_r(f)$ and $\beta_r(f)$; they have two error positions containing non-complementary symbols. Since $len(u) = 8 < I = 12$, the algorithm updates $I = 8$ and $(u, v) = (AGATATAC, AGAGACAC)$.

The second call to `WitnessesConstructor` takes as input $(AGATAC, 2, [1])$ and sets $n = 6, r = 4, i = 1$. Because $len(errpos) = 1$, then the function outputs $u = AGATAAATAC$ and $v = AGATATATAC$, obtained as $\alpha_r(f)$ and $\beta_r(f)$; they have one error position containing complementary symbols. Since $len(u) = 10$ is greater then $I = 8$ the algorithm does not update neither I nor (u, v).

Therefore, the main algorithm outputs the Lee-index $I = 8$ and the pair of Lee-witnesses $(u, v) = (AGATATAC, AGAGACAC)$.

References

1 Marcella Anselmo, Manuela Flores, and Maria Madonia. On k-ary n-cubes and isometric words. *Preprint*, 2021. URL: `https://docenti.unisa.it/uploads/rescue/385/8179/afm-k-aryisometricwords.pdf`.

2 Marcella Anselmo, Manuela Flores, and Maria Madonia. Quaternary n-cubes and isometric words. In Thierry Lecroq and Svetlana Puzynina, editors, *Combinatorics on Words*, pages 27–39, Cham, 2021. Springer International Publishing.

3 Marcella Anselmo, Dora Giammarresi, Maria Madonia, and Carla Selmi. Bad pictures: Some structural properties related to overlaps. In Galina Jirásková and Giovanni Pighizzini, editors, *DCFS 2020*, volume 12442 of *Lect. Notes Comput. Sci.*, pages 13–25. Springer, 2020. `doi:10.1007/978-3-030-62536-8_2`.

4 Marie-Pierre Béal and Maxime Crochemore. Checking whether a word is hamming-isometric in linear time. *arXiv preprint arXiv:2106.10541*, 2021.

5 Zvi Galil and Raffaele Giancarlo. Improved string matching with k mismatches. *ACM SIGACT News*, 17(4):52–54, 1986.

6 Frank Harary, John P. Hayes, and Horng-Jyh Wu. A survey of the theory of hypercube graphs. *Computers & Mathematics with Applications*, 15(4):277–289, 1988. `doi:10.1016/0898-1221(88)90213-1`.

7 W. . Hsu. Fibonacci cubes-a new interconnection topology. *IEEE Transactions on Parallel and Distributed Systems*, 4(1):3–12, 1993. `doi:10.1109/71.205649`.

8 Aleksandar Ilić, Sandi Klavžar, and Yoomi Rho. Generalized fibonacci cubes. *Discrete Mathematics*, 312(1):2–11, 2012. `doi:10.1016/j.disc.2011.02.015`.

9 Sandi Klavžar. Structure of fibonacci cubes: A survey. *Journal of Combinatorial Optimization*, 25, May 2013. `doi:10.1007/s10878-011-9433-z`.

10 Sandi Klavžar and Sergey V. Shpectorov. Asymptotic number of isometric generalized Fibonacci cubes. *Eur. J. Comb.*, 33(2):220–226, 2012.

11 Gad M. Landau and Uzi Vishkin. Efficient string matching in the presence of errors. In *26th Annual Symposium on Foundations of Computer Science*, pages 126–136. IEEE, 1985.

12 Weizhen Mao and David M. Nicol. On k-ary n-cubes: theory and applications. *Discrete Applied Mathematics*, 129(1):171–193, 2003. `doi:10.1016/S0166-218X(02)00238-X`.

13 Jianxin Wei. The structures of bad words. *Eur. J. Comb.*, 59:204–214, 2017.