**IMFM**
Institute of Mathematics, Physics and Mechanics
Jadranska 19, 1000 Ljubljana, Slovenia

**Preprint series**
**Vol. 49 (2011), 1161**
ISSN 2232-2094

# THE INDEX OF A BINARY WORD

Aleksandar Ilić Sandi Klavžar
Yoomi Rho

Ljubljana, September 27, 2011

Preprint series, IMFM, ISSN 2232-2094, no. 1161, September 27, 2011

# The index of a binary word

Aleksandar Ilić
Faculty of Sciences and Mathematics
University of Niš, Serbia
e-mail: aleksandari@gmail.com

Sandi Klavžar
Faculty of Mathematics and Physics
University of Ljubljana, Slovenia
and
Faculty of Natural Sciences and Mathematics
University of Maribor, Slovenia
e-mail: sandi.klavzar@fmf.uni-lj.si

Yoomi Rho
Department of Mathematics
University of Incheon, Korea
e-mail: rho@incheon.ac.kr

**Abstract**

A binary word $u$ is $f$-free if it does not contain $f$ as a factor. A word $f$ is $d$-good if for any $f$-free words $u$ and $v$ of length $d$, $v$ can be obtained from $u$ by complementing one by one the bits of $u$ on which $u$ and $v$ differ, such that all intermediate words are $f$-free. We say that $f$ is good if it is $d$-good for any $d \geq 1$. A word is bad if it is not good. The index $\beta(f)$ of $f$ is the smallest integer $d$ such that $f$ is not $d$-good, so that $\beta(f) < \infty$ if and only if $f$ is bad.

It is proved that $\beta(f) < |f|^2$ holds for any bad word $f$. In addition, $\beta(f) < 2|f|$ holds for almost all bad words $f$ and it is conjectured that the same holds for all bad words. An infinite family of words such that each member of it is bad, but 2-good, is constructed. It is conjectured that the words of this family are all the words that are bad and 2-good among those with exactly two 1s. These conjectures are supported by computer experiments.

**Keywords:** binary words, combinatorics on words, good words, index of a word, algorithm, generalized Fibonacci cube.

**AMS Subject Classification (2010)**: 68R15, 68W32.

# 1 Introduction

Let $f$ be a finite binary word. Then a binary word $u$ is called *$f$-free* if it does not contain $f$ as a factor. For instance, 110100110 is 111-free but not 1001-free.

Let $d$ be a positive integer. Then $f$ is called *$d$-good* if for any $f$-free words $u$ and $v$ of length $d$, the following holds: $u$ can be transformed into $v$ by complementing one by one all the bits on which $u$ differs from $v$, such that all of the new words we obtain in this process are $f$-free. Such a transformation will be called an *$f$-free transformation of $u$ to $v$*. Clearly, if there is an $f$-free transformation of $u$ to $v$, there is also an $f$-free transformation of $v$ to $u$. Now, we say that $f$ is *good* if it is $d$-good for any $d \geq 1$. The word $f$ is *bad* if it is not good, that is, if there exist words $u$ and $v$ (of the same length) for which no $f$-free transformation of $u$ to $v$ exists.

A motivation for the present study comes from isometric embeddings of graphs, as we will describe below, but the concepts and problems are of general nature which we follow here. Good and bad words were introduced in [7] as follows. For a finite binary word $f$, the *generalized Fibonacci cube*, $Q_n(f)$, is the graph obtained from $Q_n$ by removing all vertices that contain $f$ as a factor [5]. The classical Fibonacci cubes $\Gamma_n$ [4, 6] can be thus defined with $\Gamma_n = Q_n(11)$, and the subclass $Q_n(1^s)$ of generalized Fibonacci cubes was studied in [8, 11] (also under the name generalized Fibonacci cubes). Now, it is easy to see that a binary word $f$ is good if and only if $Q_d(f)$ is an isometric subgraph of $Q_d$ for any $d \geq 1$.

To test (say, using a computer) if a given word $f$ is good or bad, it would be utmost useful to know whether there is a function $\beta$ such that $f$ is good as soon as $f$ is $d$-good for $d < \beta(f)$. We therefore introduce *the index of a word $f$*, denoted $\beta(f)$, as the smallest integer $d$ for which $f$ is not $d$-good. If no such integer exists we set $\beta(f) = \infty$. Clearly, $\beta(f) < \infty$ if and only if $f$ is bad.

To the best of our knowledge, these concepts and problems were not studied earlier, except in [7]. However, numerous other operations on (binary) words have been investigated. One such operation is a prefix reversal, see [3], where it has been in particular proved that the prefix reversal distance between two arbitrary binary strings is NP-hard. Another example is the paper [2] where operations are presented that preserve primitivity of words. For the general theory on combinatorics on words and their applications, see the books [9] and [10], respectively.

We proceed as follows. In the rest of this section remaining necessary definitions are given. In the next section we first prove that the index of any bad word $f$ is smaller than $|f|^2$. Then we demonstrate that the index of **almost all** bad words $f$ is smaller than $2|f|$ and conjecture that this is eventually true for **all** bad word. Then, in Section 3, we consider the words that are 2-good but are not good. An infinite family of such words is constructed. Each of these words contains exactly two 1s and we conjecture that among such words, the constructed are the only words that are bad and 2-good. Computer support for the two conjectures is also provided.

Let $B = \{0, 1\}$ and call elements of $B$ *bits*. An element of $B^d$ is called a *binary word* (or simply a *word*) of length $d$. A word $u \in B^d$ will be written in the coordinate form as $u = u_1u_2 \dots u_d$. The $i$-th unit word, that is, the word with 1 in coordinate $i$ and 0 elsewhere, will be denoted with $e^{(i)}$. We will use the product notation for words meaning concatenation, for example, $1^d$ means $11 \dots 1$, the word of length $d$. A word $f$ is a *factor* of a word $x$ if $f$ appears as a sequence of $|f|$ consecutive bits of $x$. For a word $f$, $b_k(f)$ denotes the prefix of $f$ of length $k$ and $e_k(f)$ its suffix of the same length $k$.

## 2 Bounding the index of a word

As announced, we first prove that the index of a word can be bounded by the square of its length:

**Theorem 2.1** *Let $f$ be a bad word. Then $\beta(f) < |f|^2$.*

**Proof.** Let $d = \beta(f)$ and let $u$ and $v$ be words of length $d$ such that there is no $f$-free transformation of $u$ to $v$. We may assume that $u$ and $v$ are different in the smallest number, say $r$, of bits among all such pairs of words.

Consider the following directed graph $D_f = (V(D_f), A(D_f))$:

$$V(D_f) = \{f + e^{(i)} \mid i = 1, \ldots, |f|\}$$

and

$$A(D_f) = \{(f', f'') \mid e_k(f') = b_k(f'') \text{ for some } k \geq 1\}.$$

As an example, Fig. 1 shown for the digraph $D_{1100}$. For instance, since 1110 ends with 110 which is at the same time the beginning of 1101, there is an arc from 1110 to 1101.

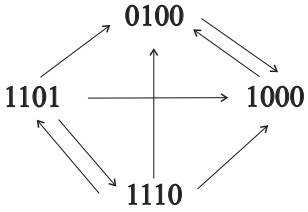


Figure 1: Digraph $D_{1100}$

Let $i_1, \ldots, i_r$ be the coordinates in which $u$ and $v$ differ. The word $u + e^{(i_j)}$ contains $f$ as a factor for $j = 1, \ldots, r$. Indeed, otherwise $u + e^{(i_j)} \in Q_d(f)$ and $d_{Q_d(f)}(u + e^{(i_j)}, v) > d_{Q_d}(u + e^{(i_j)}, v) = r - 1$, a contradiction to the minimality of $r$.

For $j = 1, \ldots, r$, let $f^{(i_j)}$ be a copy of $f$ that appears as a factor in $u + e^{(i_j)}$ and let $\widehat{f^{(i_j)}}$ be the subword of $u$ from which $f^{(i_j)}$ is obtained by complementing the $i$-th bit.

Note that each $f^{(i_j)}$ has a common coordinate with at least one $f^{(i_{j'})}$, where $j' \neq j$, because otherwise $v$ would contain $f$ as a factor. Observe also that $u$ is covered with $\cup_{j=1}^{r} \widehat{f^{(i_j)}}$, that is, in each coordinate $u$ intersects with at least one of the $\widehat{f^{(i_j)}}$. Indeed, otherwise $d$ would not be the index of $f$.

Consider now the subdigraph $X$ of $D_f$ induced by vertices $\widehat{f^{(i_j)}}$, $j = 1, \ldots, r$. For example, for $f = 1100$ and vertices $u = 1110100$ and $v = 1101000$, the subdigraph is induced by vertices 1110, 1101, and 0100.

Suppose first that the words $\widehat{f^{(i_j)}}$, $j = 1, \ldots, r$, are pairwise different. Then $r \leq |f|$ and since $u$ is covered with the $\widehat{f^{(i_j)}}$'s, $d < r \cdot |f| \leq |f|^2$.

Assume next that two among the words $\widehat{f^{(i_j)}}$ are equal. Then $X$ contains a directed cycle, say $C = f_1 \to f_2 \to \cdots \to f_s \to f_1$. Construct new vertices $u'$ and $v'$ with smaller length by removing the path $f_2 \to \cdots \to f_s \to f_1$. Clearly, $u'$ and $v'$ differ in less bits than $u$ and $v$, which is a contradiction. Therefore, $X$ does not contain directed cycles and $d < r \cdot |f| \leq |f|^2$. □

We could define also the edge weights in digraph: the weight of the directed edge $(f', f'')$ is the largest number $k$ such that $e_k(f') = b_k(f'')$. Since each vertex is represented by the path in digraph $D_f$, for each bad word $f$ its index corresponds to the directed path lengths in $D_f$. We also remark that we could further refine the $\beta(f) < |f|^2$ bound by considering the intersection of every two consecutive vertices from $D$, but it would be still quadratic. On the other hand, we can do much better with high probability:

**Theorem 2.2** *For almost all bad words, $\beta(f) < 2|f|$.*

**Proof.** If $b_k(f)$ and $e_k(f)$ agree in all but $r$ positions, then $f$ has an *$r$-error overlap of length $k$*. If $f$ has an $r$-error overlap for some length $k$ then we simply say that $f$ has an *$r$-error overlap*. We also say that $f$ is a *stutter* if $f$ has an $r$-error overlap of length $k$, where $r \leq 2$ and $k \geq \frac{n}{2}$. In [7] it was proved that the proportion of stutters among all words of length $n$ tends to zero when $n \to \infty$. Moreover, asymptotically close to 92% of all words are bad. It follows from these two facts that the proportion of stutters among all bad words of length $n$ also tends to zero when $n \to \infty$. Hence, the theorem will be proved if we show that the index of any bad word $f$ that is not a stutter is less that $2|f|$.

Suppose therefore that $f$ is a bad word but not a stutter. Since $f$ is bad, a theorem of [7] guarantees that $f$ has a 2-error overlap. Let $k$ be the length of a 2-error overlap and let $b_k(f)$ disagree from $e_k(f)$ in positions $i$ and $j$ of $b_k(f)$, where $i < j$, see Fig. 2.

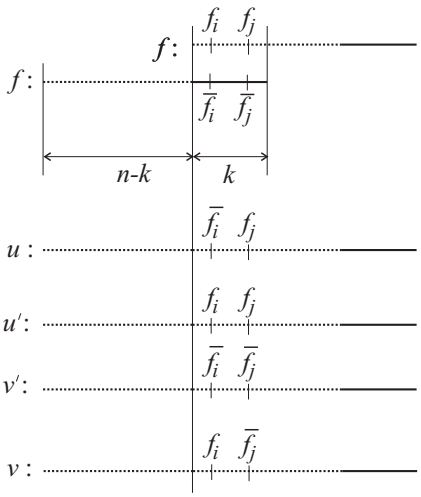


Figure 2: 2-error overlap when $f$ is not a stutter

Set $d = 2n - k$ and define words $u, v \in B^d$ as follows. Let $u$ be the concatenation of $b_{n-k}(f)$ and the word $f + e^{(i)}$, and let $v$ be the concatenation of $b_{n-k}(f)$ with $f + e^{(j)}$, see Fig. 2 again. Clearly, $u$ and $v$ disagree in positions $n - k + i$ and $n - k + j$. Now consider the words $u' = u + e^{(n-k+i)}$ and $v' = u + e^{(n-k+j)}$. Observe that $u'$ contains $f$ as its suffix of length $n$ and that $v'$ contains $f$ as its prefix of length $n$. Because $f$ is not a stutter and $k < n/2$ we conclude that none of $u'$ and $v'$ is $f$-free. Hence $v$ cannot be obtained from $u$ by an $f$-free transformation. Since $d < 2n$, we conclude that $\beta(f) < 2n$. □

We note that Theorem 2.2 is implicit in [7]. Based on this theorem and computer experiments we close the section with:

**Conjecture 2.3** *For any bad word $f$, $\beta(f) < 2|f|$.*

## 3 On words that are bad and 2-good

Another look at the proof of Theorem 2.2 reveals that in the case when $f$ is not a stutter, there exist words $w$ and $w'$ of length $d = 2|f| - k$ (where $k$ is the length of a 2-error overlap) demonstrating that $f$ is not 2-good. In other words, any bad word that is not a stutter is not even 2-good. In view of Conjecture 2.3 one might be tempted that this is the case for all bad words, that is, as soon as a word is bad, it is not 2-good. That this is not the case is demonstrated with the following result:

**Theorem 3.1** *Let $r \geq 0$. Then*

$$f = 0^{2^{r+1}}10^{2^r-1}10^{2^r-1}$$

*is a 2-good, bad word.*

**Proof.** We first show that $f$ is not 3-good. Set $d = 7(2^r - 1)$ and consider the words

$$u = (00^{2^r-1})^2 00^{2^r-1}10^{2^r-1}00^{2^r-1}(10^{2^r-1})^2$$

and

$$v = (00^{2^r-1})^2 10^{2^r-1}00^{2^r-1}10^{2^r-1}(10^{2^r-1})^2\,.$$

Note that both $u$ and $v$ are $f$-free and that they differ in three bits. The three words obtained from $u$ by complementing the bits in which $u$ differs from $v$ are

$$(00^{2^r-1})^2 10^{2^r-1}10^{2^r-1}00^{2^r-1}(10^{2^r-1})^2\,,$$

$$(00^{2^r-1})^2 00^{2^r-1}00^{2^r-1}00^{2^r-1}(10^{2^r-1})^2\,,$$

and

$$(00^{2^r-1})^2 00^{2^r-1}10^{2^r-1}10^{2^r-1}(10^{2^r-1})^2\,.$$

None of these three words is $f$-free, hence $f$ is not 3-good. So $f$ is bad.

To complete the proof we need to show that $f$ is 2-good. Assume on the contrary that there exist two $f$-free words $u$ and $v$ of length at least $|f| = 2^{r+2}$ that differ in two bits, but there is no $f$-transformation of $u$ to $v$. Let $i$ and $j$ be the positions in which $u$ and $v$ differ, where $i < j$. It follows that both words $u' = u + e^{(i)}$ and $u'' = u + e^{(j)}$ contain $f$ as a factor. Denote the factor $f$ of $u'$ with $f'$ and the factor $f$ of $v'$ with $f''$. Let factors $f'$ and $f''$ start from positions $k'$ and $k''$, respectively. Assume that $k' < k''$, the case when $k'' < k'$ is treated analogously. Note that $f'$ and $f''$ must have some common indices because otherwise $v$ would contain $f$ as a factor. In other words, $k' < k'' \leq k' + 2^{r+2}$. Note that the common indices of $f'$ and $f''$ are from the segment $S = [k'', k' + 2^{r+2} - 1]$.

Both indices $i$ and $j$ belong to the segment $S$. Indeed, if $i$ is not from $S$, then $v$ would contain $f'' = f$ as a factor. Similarly, $u$ would contain $f' = f$ as a factor if $j$ would not be from $S$.

Consider the word $f = 0^{2^{r+1}}10^{2^r-1}10^{2^r-1}$; its first half is composed of 0s and its two 1s are on the positions $\frac{|f|}{2} + 1$ and $\frac{3|f|}{4} + 1$. Since $f' = f$ and $f'' = f$ differ in exactly two positions from the segment $S$, namely in positions $i$ and $j$, this is possible only when $k''$, the first bit of $f''$, is under the position $\frac{|f|}{2} + 1$ of $f'$. Here is an example for $r = 2$:

```
0000000010001000
        0000000010001000
```

But now $u$ contains the factor $0^{2^{r+1}}00^{2^{r-1}}10^{2^r-1}10^{2^r-1}10^{2^r-1}$, a contradiction since we assumed that $u$ is $f$-free. □

For the special case $r = 0$ (that is, $f = 0011$) of Theorem 3.1 it was earlier [5] proved that 0011 is a bad word.

The 2-good (and bad) words from Theorem 3.1 contain precisely two 1s. On the other hand, many such words are not 2-good:

**Proposition 3.2** *Let $r, s, t \geq 0$ and $t \geq r + s + 3$. Then the word $0^r10^s10^t$ in not 2-good.*

**Proof.** Let $k = r + 1$. Then $k \leq t - s - 2$ since we have assumed that $t \geq r + s + 3$. Let $d \geq 2r + 2s + t + 5$ and consider the words

$$u = 0^{d-r-2s-t-k-4}0^r10^s10^k00^s10^t$$

and

$$v = 0^{d-r-2s-t-k-4}0^r10^s10^k10^s00^t\,.$$

Note first $u$ and $v$ differ in two bits. In addition, we claim that they are $f$-free. Indeed, If $u$ would contain $f$ as a factor, then the factor must contain the first two 1s, but this is impossible as $k \leq t - s - 2$. Similarly, suppose $v$ contains $f$ as a factor. As we already know that $u$ does not contain $f$ as a factor, the factor $f$ in $v$ cannot contain the first two 1s. But the factor also cannot contain the last two 1s since $k \neq s$. This proves the claims.

The words that differ from $u$ in the two bits in which $u$ differs from $v$ are

$$w = 0^{d-r-2s-t-k-4}0^r10^s10^k00^s00^t$$

and

$$w' = 0^{d-r-2s-t-k-4}0^r10^s10^k10^s10^t\,.$$

Clearly, $w$ contains $f$. Moreover, the same also holds for $w'$ because $k \geq r$. (Actually $w'$ is not $f$-free if and only if $k \geq r$ because if $w'$ contains $f$ as a factor, then the factor contains the last two bits of 1 of $w'$, which occurs exactly when $k \geq r$.) Hence $f$ is not 2-good. □

By the symmetry, the same conclusion can also be made for words $f = 0^r10^s10^t$ with $r \geq s + t + 3$.

Recall that by Theorem 3.1, the word

$$00001010$$

is 2-good. On the other hand, Proposition 3.2 and the above remark imply that the word

$$000001010$$

(obtained by $s = t = 1$, $r = 5$) is not 2-good. These two words show that there is a very thin line between being 2-good and not being 2-good.

We also conclude this section with a conjecture. It is motivated by Theorem 3.1 and computer experiments.

**Conjecture 3.3** *Let $f$ be a bad word that contains exactly two 1s. Then $f$ is 2-good if and only if $f = 0^{2^{r+1}}10^{2^r-1}10^{2^r-1}$ (or its reverse) for some $r \geq 0$.*

## 4 Computational results

For each length $3 \leq r \leq 10$ we generated all binary words of length $r$ (reverse words and complements are excluded) and calculated the index of these words by considering all possible words of lengths $d$, $r \leq d \leq 20$. For each word $f$ and dimension $d$ we constructed generalized Fibonacci cube $Q_d(f)$ and then ran breadth first search algorithm from each vertex of a cube in order to determine the distance matrix and check the embeddability of $Q_d(f)$ in the hypercube $Q_d$. In Table 1 the computational results for words with exactly two 1s are presented.

The table needs some comments. The words that were recognized as bad and for which the index was computed, clearly support Conjecture 2.3. Among them, only the word 0011 attains the conjectured upper bound: let $f = 0011$, then $\beta(f) = 7 = 2|f| - 1$. It was proved in [5] that for any $s \geq 2$, the word $1^s01^s0$ is good, and that for any $s \geq 1$, the word $(10)^s$ is good. These two results cover all the good words from the table except the four words with "(?)" attached to them. These are the words 0001001, 0001010, 000010010, and 000010001. Each of them is a stutter, so we cannot use the proof of Theorem 2.2 to conclude that they are good. However, it follows from our computations that either each of them is good or Conjecture 2.3 is false. Note also that the obtained results support Conjecture 3.3.

We also designed an $O(|f|^4)$ algorithm for checking whether a given binary word $f$ is 2-good. If $f$ is not 2-good, then there exist two words $v$ and $u$ which differ on two places $i$ and $j$, such that $v + e^{(i)}$ contains $f$ as a prefix and $v + e^{(j)}$ contains $f$ as a suffix. This means that we can try to overlap two copies of $f$ such that $b_k(f)$ and $e_k(f)$ differ on exactly zero or two places (see Fig. 2), where $k$ is the number of bits in the intersection of two copies of $f$. That is, in order to construct possible words $u$ and $v$ to demonstrate that $f$ is not 2-good, we will consider cases when $b_k(f)$ and $e_k(f)$ differ on exactly zero or two places.

If the number of differences is two, we can construct the words $u$ and $v$ by changing the bits on $i$-th and $j$-th position, respectively. Finally, if $v$ and $u$ do not contain $f$ as a subword, it follows that $f$ is not 2-good. If the number of differences is zero, we need to traverse all pairs $(i, j)$, $1 \leq i < j \leq k$, construct the words $u$ and $v$ as above and check whether $u$ and $v$ contain $f$ as a subword.

For searching the occurrences of $f$ in the words $u$ and $w$ we use the Knuth-Morris-Pratt string searching algorithm [1]. KMP algorithm searches for occurrences of a word $w$ within a main text string $s$ by employing the observation that when a mismatch occurs, the word $w$ itself embodies sufficient information to determine where the next match in $s$ could begin. The running time of this algorithm is linear $O(|w| + |s|)$, which is optimal in the worst case sense. Pseudo code of this approach is shown below as Algorithm 1.

| Length | The index | Words |
|---|---|---|
| 2 | $\infty$ | 11 |
| 3 | 4 | 101 |
| | $\infty$ | 110 |
| 4 | 5 | 0110 |
| | 7 | 0011 |
| | $\infty$ | 1010 |
| 5 | 6 | 00110, 10001 |
| | 7 | 01001 |
| | 8 | 00011, 01010 |
| | $\infty$ | 00101 |
| 6 | 7 | 000110, 001100, 100001 |
| | 8 | 010001, 010010 |
| | 9 | 000011, 000101, 001010 |
| | $\infty$ | 001001 |
| 7 | 8 | 0000110, 0001100, 1000001 |
| | 9 | 0010100, 0100001, 0100010 |
| | 10 | 0000011, 0000101, 0010001 |
| | 11 | 0010010 |
| | $\infty$ | 0001001(?), 0001010(?) |
| 8 | 9 | 00000110, 00001100, 00011000, 10000001 |
| | 10 | 00010100, 01000001, 01000010 |
| | 11 | 00000011, 00000101, 00100001, 00100010 |
| | 12 | 00001001, 00010010, 00100100 |
| | 14 | 00001010 |
| | $\infty$ | 00010001 |
| 9 | 10 | 000000110, 000001100, 000011000, 100000001 |
| | 11 | 000010100, 000101000, 010000001, 010000010 |
| | 12 | 000000011, 000000101, 001000001, 001000010, 001000100 |
| | 13 | 000001001, 000100001, 000100100 |
| | 14 | 000001010, 000100010 |
| | $\infty$ | 000010001(?), 000010010(?) |
| 10 | 11 | 0000000110, 0000001100, 0000011000, 0000110000, 1000000001 |
| | 12 | 0000010100, 0000101000, 0100000001, 0100000010 |
| | 13 | 0000000011, 0000000101, 0001001000, 0010000001, 0010000010, 0010000100 |
| | 14 | 0000001001, 0001000001, 0001000010 |
| | 15 | 0000001010, 0000010001, 0000010010, 0000100010, 0000100100, 0001000100 |
| | $\infty$ | 0000100001 |

Table 1: The index of words with two 1s.

```
Input: Binary word f
Output: True if f is 2-good, false otherwise
n = |f|;
for k = 2 to n − 1 do
    diff = 0;
    i = −1;
    j = −1;
    for s = 1 to k do
        if f[s] ≠ f[n − k + s] then
            diff = diff + 1;
            if diff > 2 then
                break;
            end
            if i = −1 then
                i = s;
            else
                j = s;
            end
        end
    end
    if diff = 2 then
        v = f + substring(f, k + 1, n);
        u = substring(f, 1, k) + f;
        v[k + i] = 1 − v[k + i];
        u[k + j] = 1 − u[k + j];
        if KMP(v, f) = false and KMP(u, f) = false then
            return false;
        end
    end
    if diff = 0 then
        v = f + substring(f, k + 1, n);
        u = substring(f, 1, k) + f;
        for i = 1 to k − 1 do
            v[k + i] = 1 − v[k + i];
            for j = i + 1 to k do
                u[k + j] = 1 − u[k + j];
                if KMP(v, f) = false and KMP(u, f) = false then
                    return false;
                end
                u[k + j] = 1 − u[k + j];
            end
            v[k + i] = 1 − v[k + i];
        end
    end
end
return true;
```

**Algorithm 1**: Determining whether a word $f$ is 2-good.

## Acknowledgements

This work was supported by Research Grants 174010 and 174033 of Serbian Ministry of Science, by the Research Grant P1-0297 of Ministry of Higher Education, Science and Technology Slovenia, and by Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology grant 2011-0025319. The work was partially done during a visit of S.K. at the University of Incheon, Korea, whose support is gratefully acknowledged.

## References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, Second Edition, MIT Press, 2001.

[2] J. Dassow, G. M. Martín, F. J. Vico, Some operations preserving primitivity of words, Theoret. Comput. Sci. 410 (2009) 2910–2919.

[3] C. Hurkens, L. van Iersel, J. Keijsper, S. Kelk, L. Stougie, J. Tromp, Prefix reversals on binary and ternary strings, SIAM J. Discrete Math. 21 (2007) 592–611.

[4] W.-J. Hsu, Fibonacci cubes—a new interconnection technology, IEEE Trans. Parallel Distrib. Syst. 4 (1993) 3–12.

[5] A. Ilić, S. Klavžar, Y. Rho, Generalized Fibonacci cubes, Discrete Math., to appear. doi:10.1016/j.disc.2011.02.015.

[6] S. Klavžar, Structure of Fibonacci cubes: a survey, J. Comb. Optim., to appear.

[7] S. Klavžar, S. Shpectorov, Asymptotic number of isometric generalized Fibonacci cubes, manuscript, 2010.

[8] J. Liu, W.-J. Hsu, M. J. Chung, Generalized Fibonacci cubes are mostly Hamiltonian, J. Graph Theory 18 (1994) 817–829.

[9] M. Lothaire, Algebraic Combinatorics on Words, Cambridge University Press, Cambridge, 2002.

[10] M. Lothaire, Applied Combinatorics on Words, Cambridge University Press, Cambridge, 2005.

[11] N. Zagaglia Salvi, On the existence of cycles of every even length on generalized Fibonacci cubes, Matematiche (Catania) 51 (1996) 241–251.