Schutzenberger's Guessing Game on a Finite Kolakoski Subsequence

Marlo S. de Guzman
Institute of Mathematics, UPDiliman
C.P. Garcia Ave., Diliman
Quezon City 1101
msdeguzman2@up.edu.ph

Joseph M. Pasia *
Institute of Mathematics, UPDiliman
C.P. Garcia Ave., Diliman
Quezon City 1101
jmpasia@up.edu.ph

Henry N. Adorna †
Department of Computer Science, UPDiliman
Velasquez Ave., Diliman
Quezon City 1101
hnadorna@up.edu.ph

ABSTRACT

Significant number of researchers focus on the reconstruction of word from a given set of subwords. Solution to this problem is directly applicable to biological sequences. Recently, an algorithm called Schutzenberger Guessing Game was introduced to obtain a linear run time complexity to reconstruct a word from a given set of subwords. In this paper, we apply this linear running algorithm to realize a smallest set of subwords to reconstruct a finite proper prefix of A006928 or the well-known Kolakoski sequence.

Keywords

Subwords, Kolakoski sequence, Schutzenberger Guessing game

*J. Pasia is funded by the UP Chicago Club Professorial Chair, College of Science, UPDil..

†H. Adorna is partially supported by a FRIA grant from the College of Engineering, UPDil..

1. DEFINITIONS AND PRELIMINARIES

DEFINITION 1.

- *A finite alphabet A is a finite non-empty set. A word w drawn from A, say $w = a_1a_2\cdots a_n$, defines a map $w : [n] \to A$ from [n] into A where $[n]$ is the index set $\{0, 1, 2, \ldots n-1\}$.*
- *For any word w, let $||w||$ denote its length and $w_{[j]}$ its restriction to the subset [j] of [n] where $j \in [n]$*
- *Let $A^{[n]}$ be the collection of words over A of exactly length n.*
- *Given any set S, $\sharp S$ gives its cardinality.*
- *For any word $w \in A^{[n]}$ and any letter $a \in A$, we define the function w^{-1} as*

  $$w^{-1}(a) := \{i \in [n] | w(i) = a\}$$

  and $||w||_a := \sharp w^{-1}(a)$.
- *Given any $n \in N$, any alphabet A, and any word $w \in A^{[n]}$, define the subset $\binom{w}{m}$ of $A^{[m]}$ for every $m \in [n]$ by $\binom{w}{m} := \{a_{i_1}a_{i_2}\cdots a_{i_m} | i_1, i_2 \cdots, i_m \in [n], 1 \leq i_1 < \cdots < i_m \leq n\}$.*

  Any word v in $\binom{w}{m}$ is called a subword of w .

  $\forall w \in A^{[m]}, \sharp \binom{w}{m} =_n C_m = \frac{n!}{(n-m)!m!}$
- *A prefix of length $m \in \mathbb{N}$ is a subword v of a word w where $\|v\| = m$ having consecutive indices beginning at 0 and ending at $m-1$.*

For the rest of the paper, we let A be the alphabet and $w \in A^{[}n]$, where $n \in \mathbb{N}$ is the length of w. We also denote $m \in \mathbb{N}$ as the length of the subwords of w.

For the rest of the paper, all standard definitions on formmal language theory follow [2, 6]. Also, we agree that $w \in A^{[n]}$ and $\sharp A = 2$.

1.1 Schutzenberger's Guessing Game

The *Schutzenberger Guessing Game* (SGG) is an algorithm that can systematically reconstruct a word $w \in A^{[n]}$ of length n from answers to queries about its subword of fixed length m. It follows from the result of Schutzenberger and Simon[9] (as cited in [1]) and Levenstein[7, 8] (as cited in [1]) that one can always reconstruct a word w from answers to sufficiently many queries when $2m > n$. The shortest proof of that can be found in the *Bible of Formal Language Theory* by [5]. SGG algorithm in [1] considers the following three types of queries:

(i.) What is $\|w : m\| := \max \left(\|v\|_a : v \in \binom{w}{m} \right)$?

(ii.) What is $\bar{j}_a(w\,|m|\,k) :=$

$$\max \left(\min \left(v^{-1}(a) : v \in \binom{w}{m}, ||v||_a \geq k \right) \right)?$$

(iii.) What is $\underline{j}_a(w\,|m|\,k) :=$

$$\min \left(\max(v^{-1}(a)) : v \in \binom{w}{m}, ||v||_a \geq k \right)?$$

Since the condition $\|w\|_a < m$ must hold for all but at most one letter $a \in A$ when $2m > n$, the following theorem holds:

THEOREM 1. *[1] Given an alphabet A and two integers $n, m \in \mathbb{N}$ with $2m > n$, any word $w \in A^{[n]}$ can be reconstructed from answers to $\sharp A$ queries of type (i), and $\left\lceil n\left(1 - \frac{1}{\sharp A}\right)\right\rceil$ queries of type (ii) and (iii).*

Note that knowing the length of the word w formed from A and the corresponding indices of each element of A is equivalent to knowing the word w.

An elaboration dealing with alphabets and sequences with reverse complementation have been recently developed in [3].

1.2 Kolakoski Sequence

The Kolakoski sequence (Sloane's A006928) is a self-describing sequence consisting of "blocks" of single and double 1s and 2s, where a "block" is a single digit or pair of digits that is different from the digit (or pair of digits) in the preceding block.

To construct the sequence, start with the single digit 1 (the first "block"). Here, the single 1 means that block of length one follows the first block. Therefore, require that the next block is 2, giving the sequence 12. Now, the 2 means that the next (third) block will have length two, so append 11 and obtain the sequence 1211. We have added two 1s, so the fourth and fifth blocks each has length of one, giving 12112 and then 121121 respectively. As a result of adding 21, we obtain 121121221. As a result of adding 221, we obtain 12112122122112, and so on, giving the sequence 1, 2, 1, 1, 2, 1, 2, 2, 1, 2, 2, 1, 1, 2, ... The sequence after successive iterations is given by 1, 12, 1211, 121121, 121121221, ..., and the lengths of this sequence after steps n=1, 2, ... are given by 1, 2, 4, 6, 9, 14, 22, $\cdots$.

Other ways of defining the Kolakoski Sequence can be found in [4]

In this paper, we also refer to the Kolakowski Sequence as Kol and we define the set W_{Kol} as the set containing the prefixes of Kol i.e.,

$$W_{Kol} = \{w | w \text{ is a prefix of } Kol, \|w\| = n,\ \forall n \in \mathbb{N}\}\ .$$

2. ALGORITHM FOR FINDING THE SUBWORDS

In this section, we describe our algorithm that finds the set with the smallest number of subwords that could reconstruct w using the Schutzenberger's Guessing Game. We first propose a method of classifying or grouping all subwords of fixed length. This classification is used in our algorithm.

2.1 Classification of Subwords

Let

$$\binom{w}{m}_0 = \{w_0, w_1, \ldots, w_{n-m}\}$$

where,

$$w_0 = a_0 a_1 a_2 \ldots a_{n-m+1}$$

$$\vdots$$

$$w_k = a_k a_{k+1} a_{k+1} \ldots a_{n-m+k+1}$$

From $\binom{w}{m}_0$ we define,

$$\binom{w}{m}_k = \bigcup_{i=0}^{i=n-m} V_i$$

$$k = \{1, \ldots, m\}$$

such that,

$$V_i = \{x \in \binom{w}{m} | Ham(x, w_i) = k\},$$

$$i = \{0, \ldots, n-m\}$$

where $Ham(x, w_i)$ is the number of places where x differs from w_i. We call $\binom{w}{m}_k$ the *set of order k*.

We list the following properties of $\binom{w}{m}_k, k \in \{0, \ldots, m\}$

- $\sharp \binom{w}{m}_k = \sum_{i=1}^{n-m} {}_{n-m-i+k}C_k$
- $\forall k, l, k \neq l, \binom{w}{m}_k \bigcap \binom{w}{m}_l = \emptyset$
- $\bigcup_{k=0}^{k=m} \binom{w}{m}_k = \binom{w}{m}$
- $\binom{w}{m}_k, \forall k = \{0, \cdots, m\}$, is a partition of $\binom{w}{m}$

EXAMPLE 1. *Let* $w = 12211212$ *and* $m = 5$. *The sets below display the subwords of* 12211212 *and how are they classified according to our method of classification. Let* $w = 1_0 2_1 2_2 1_3 1_4 2_5 1_6 2_7$

$$\binom{w}{m}_0 = \{1_0 2_1 2_2 1_3 1_4, 2_1 2_2 1_3 1_4 2_5, 2_2 1_3 1_4 2_5 1_6, 1_3 1_4 2_5 1_6 2_7\}$$

$$\binom{w}{m}_1 = \{1_0 2_1 2_2 1_3 2_5, 1_0 2_1 2_2 1_3 1_6, 1_0 2_1 2_2 1_3 2_7, 2_1 2_2 1_3 1_4 1_6, 2_1 2_2 1_3 1_4 2_7, 2_2 1_3 1_4 2_5 2_7\}$$

$$\binom{w}{m}_2 = \{1_0 2_1 2_2 1_4 2_5, 1_0 2_1 2_2 1_4 1_6, 1_0 2_1 2_2 1_4 2_7, 1_0 2_1 2_2 2_5 1_6, 1_0 2_1 2_2 2_5 2_7, 1_0 2_1 2_2 1_6 2_7, 2_1 2_2 1_3 2_5 1_6, 2_1 2_2 1_3 2_5 2_7, 2_1 2_2 1_3 1_6 2_7, 2_2 1_3 1_4 1_6 2_7\}$$

$$\binom{w}{m}_3 = \{1_0 2_1 1_3 1_4 2_5, 1_0 2_1 1_3 1_4 1_6, 1_0 2_1 1_3 1_4 2_7, 1_0 2_1 1_3 2_5 1_6, 1_0 2_1 1_3 2_5 2_7, 1_0 2_1 1_3 1_6 2_7, 1_0 2_1 1_4 2_5 1_6, 1_0 2_1 1_4 2_5 2_7, 1_0 2_1 1_4 1_6 2_7, 1_0 2_1 2_5 1_6 2_7, 2_1 2_2 1_4 2_5 1_6, 2_1 2_2 1_4 2_5 2_7, 2_0 2_2 1_4 1_6 2_7, 2_1 2_2 2_5 1_6 2_7, 2_2 1_3 2_5 1_6 2_7\}$$

$$\binom{w}{m}_4 = \{1_0 2_2 1_3 1_4 2_5, 1_0 2_2 1_3 1_4 1_6, 1_0 2_2 1_3 1_4 2_7, 1_0 2_2 1_3 2_5 1_6, 1_0 2_2 1_3 2_5 2_7, 1_0 2_2 1_3 1_6 2_7, 1_0 2_2 1_4 2_5 1_6, 1_0 2_2 1_4 2_5 2_7, 1_0 2_2 1_4 1_6 2_7, 1_0 2_2 2_5 1_6 2_7, 1_0 1_3 1_4 2_5 1_6, 1_0 1_3 1_4 2_5 2_7, 1_0 1_3 1_4 1_6 2_7, 1_0 1_3 2_5 1_6 2_7, 1_0 1_4 2_5 1_6 2_7, 2_1 1_3 1_4 2_5 1_6, 2_1 1_3 1_4 2_5 2_7, 2_1 1_3 1_4 1_6 2_7, 2_1 1_3 2_5 1_6 2_7, 2_1 1_4 2_5 1_6 2_7, 2_2 1_4 2_5 1_6 2_7\}$$

2.2 The Algorithm

The general idea of our algorithm is to remove subwords $\binom{w}{m}$ until we can find the smallest possible collection of subwords that can reconstruct w.

We start by exhausting all the possible combinations of the $\binom{w}{m}_k$, $k \in \{0, \ldots, m-1\}$ i.e. if m is the number of groups, we get

$$\{{}_mC_1, {}_mC_2, \ldots, {}_mC_{m-1}\} .$$

We start from ${}_mC_1$ up to ${}_mC_{m-1}$. If we find j where $1 < j < m-1$ such that j is a collection of sets with order k that can reconstruct w, we terminate.
We then collect those combinations of groups that can reconstruct w in a set and call it $\mathbb{P}$. Take note that for each $p \in \mathbb{P}$, $p \subset \binom{w}{m}$. For all $p \in \mathbb{P}$, we delete the subwords inside p, one by one, and apply the SGG.
If the SGG reconstructs w then we permanently remove that subword and obtain a new subword.
If SGG fails to reconstruct w, we replace the subword with another subword.
For each p, we obtain the smallest possible number of subwords that can reconstruct w after exhausting all the subwords.

Below is the algorithm:

1. *Construct* $\mathbb{P}$ *using SGG*
2. *If this is the initial run, we get a new p from* $\mathbb{P}$ *else we return p to* $\mathbb{P}$ *before we get a new p. Terminate, if all elements of* $\mathbb{P}$ *has been exhausted.*
3. *Initialize k to* -1.
4. *Let* $k = k + 1$
5. *If* $k < \|p\|$ *then remove* v_{p_k} *from p, where* v_{p_k} *is a subword, else go back to* (2)
6. *Let* $p = p - \{v_{p_k}\}$
7. *Apply the SGG on p*

8. *If SGG reconstructs w, then go to* (4) *else return* v_{p_k} *to* p.

9. *Let* $p = p + v_{p_k}$ *and go to* (4)

After running the algorithm, we are left with the smallest number of subwords for each $p \in \mathbb{P}$ needed to reconstruct w.

This is easily obtained since our method is especially exhaustive.

EXAMPLE 2. *Let* $w = 1_0 2_1 2_2 1_3 1_4 2_5 1_6 2_7$ *and* $m = 5$ *and we generate* $\binom{w}{m}$ *where* $\sharp \binom{w}{m} = 56$. *We have already classified its subwords into its respective orders in example 1. We now apply the algorithm to those orders. We first find* $\mathbb{P}$,

$$\mathbb{P} = \{ \binom{w}{m}_3 \}$$

Our $\mathbb{P}$ *is composed of only a single element that is the set* $\binom{w}{m}_3$ *which is of order* 3. *From there, we start processing the subwords inside* $\binom{w}{m}_3$ *according to our algorithm. We obtain a smaller set* p *as follows,*

$$p = \{1_0 2_1 1_3 1_4 1_6, 1_0 2_1 1_3 2_5 2_7, 2_1 2_2 1_4 2_5 1_6, 2_1 2_2 2_5 1_6 2_7, 2_2 1_3 2_5 1_6 2_7\}$$

Applying the SGG on p *still yields* $w = 12211212$.

Hence, the algorithm has found a set p, $\sharp p < \sharp \binom{w}{m}_3$, *such that* p *could still reconstruct* w.

Note that $\sharp \binom{w}{m}_3 = 56$.

2.3 Some Properties of the Algorithm

DEFINITION 2.

- *Let* j_a *be the collection of all the answers to queries two and three.*

- $f_a(w : m) : \binom{w}{m} \Rightarrow [n]$ *where* $f_a(w : m)(v) =$ a *for some* $v \in \binom{w}{m}$, $a \in [n]$

- $\bar{f}_a(w|m|k) : \binom{w}{m} \Rightarrow [n]$ *index set of* w *where* $\bar{f}_a(w|m|k)(v) = \min\left(v^{-1}(a)\right)$ *where* $v \in \binom{w}{m}$ *and* $\|v\|_a \geq k$

- $\underline{f}_a(w|m|k) : \binom{w}{m} \Rightarrow [n]$ *index set of* w *where* $\underline{f}_a(w|m|k)(v) = \max\left(v^{-1}(a)\right)$ *where* $v \in \binom{w}{m}$ *and* $\|v\|_a \geq k$

We also define the following set mapping for the sets of subwords of w.

DEFINITION 3.

- $\|w : m\|_a : \left\{ \binom{w}{m} \right\} \Rightarrow [n]$ *where* $\|w : m\|_a(v) =$ $\max(f_a(w : m)(v))$ *where* $v \in \binom{w}{m}$.

- $\bar{j}_a(w|m|k) : \binom{w}{m} \Rightarrow [n]$ *index set of* w *where* $\bar{j}_a(w|m|k)(v) = \min\left(v^{-1}(a)\right)$ *where* $v \in \binom{w}{m}$ *and* $\|v\|_a \geq k$

- $\underline{j}_a(w|m|k) : \binom{w}{m} \Rightarrow [n]$ *index set of* w *where* $\underline{j}_a(w|m|k)(v) = \max\left(v^{-1}(a)\right)$ *where* $v \in \binom{w}{m}$ *and* $\|v\|_a \geq k$

REMARK 1. *The SGG fails to reconstruct a word using the subwords on* V *if the following two cases occur:*

1. *The sum of the answers to query (i) is less than the length of the word i.e.* $\sum_{a \in A} ||w : m||_a < ||w||$.
2. *The total number of elements in* $\{j_a\}$ *is less than the length of the word i.e.,* $||\{j_a\}|| < ||w||$.

It is important to note that cases 1 and 2 happen if the set V lacks a subword v that uniquely answers any of the three queries.

PROPOSITION 1. *Let* $w \in A^{[n]}, \forall a \in A$ *and* $m \in \mathbb{N}$. *The mappings* f_a, $\bar{f}_a$ *and* $\underline{f}_a$ *are not necessarily one to one.*

PROOF. Firstly, the Schutzenberger Algorithm on $\binom{w}{m}$ guarantees the existence of subwords u, v and w where $f_a(w:m)(u) = n$, $\bar{f}_a(w|m|k)(v) = y$ or $\underline{f}_a(w|m|k)(w) = z$ such that n is a natural number and y and z are the answers to queries (ii) and (iii) for a given $k \in \mathbb{N}$. We know that $\| \binom{w}{m} \| = {}_nC_m$ and $\|[n]\| = \|w\| = n$. Since the cardinality of the domain of the mappings above is greater than the cardinality of its codomain, the pigeonhole principle says that more than one subword will be sent to at least one index of the index set. Therefore, the mappings described above are not necessarily one to one. □

PROPOSITION 2. *Let* $V \subseteq \binom{w}{m}$ *such that* $\bar{j}_a(w|m|k)(V) = y$, *for some* $y \in [n]$ *of* w *and for some* $k \in \{1, \cdots, \|w\|\}$ *i.e.,* $\exists v \in V$ *s.t.* $\forall v_i \in V - \{v\}$, $\bar{f}_a(w|m|k)(v) = y$ *and* $\bar{f}_a(w|m|k)(v) \geq \bar{f}_a(w|m|k)(v_i)$.

Take $\{v'\}$, $v' \in \binom{w}{m} - V$ *such that* $\bar{f}_a(w|m|k)(v') = z$ *where* $z \in [n]$ *of* w *and consider* $V \cup \{v'\}$. *The following statements hold:*

a. if $y \geq z$ *then* $\bar{j}_a(w|m|k)(V \cup \{v'\}) = y$

b. if $y \leq z$ *then* $\bar{j}_a(w|m|k)(V \cup \{v'\}) = z$

PROOF. a.) Since $\bar{f}_a(w|m|k)(v') = z < y = \bar{f}_a(w|m|k)(v)$. This implies that for the set $V \cup \{v'\}$, $v \in V$ still has the maximum $\bar{f}_a(w|m|k)$. Therefore, $\bar{j}_a(w|m|k)(V \cup \{v'\}) = y$.

b.) Since $\bar{f}_a(w|m|k)(v') = z > y = \bar{f}_a(w|m|k)(v)$ and the image of v is the maximum for V, then $\forall v_i \in V$, $\bar{f}_a(w|m|k)(v') = z > \bar{f}_a(w|m|k)(v_i)$. Hence, the image of v' now is the maximum for $V \cup \{v'\}$. Therefore, $\bar{j}_a(w|m|k)(V \cup \{v'\}) = z$. □

PROPOSITION 3. *Let* $V \subseteq \binom{w}{m}$ *such that* $\underline{j}_a(w|m|k)(V) = y$, *for some* $y \in [n]$ *of* w *and for some* $k \in \{1, \cdots, \|w\|\}$ *i.e.,* $\exists v \in V$ *s.t.* $\forall v_i \in V - \{v\}$, $\bar{f}_a(w|m|k)(v) \leq \underline{f}_a(w|m|k)(v_i)$.

Take $\{v_p\}$, $v_p \in \binom{w}{m}$ *such that* $\underline{f}_a(w|m|k)(v_p) = z$ *where* $z \in [n]$ *of* w. *and consider* $V \cup \{v_p\}$. *The following statements hold:*

a. If $y \leq z$ *then* $\underline{j}_a(w|m|k)(V \cup \{v_p\}) = y$

b. If $y \geq z$ *then* $\underline{j}_a(w|m|k)(V \cup \{v_p\}) = z$.

Observe that Proposition 3 is similar to Proposition 2. Hence, its proof follows the proof of Proposition 2.

REMARK 2. *The concepts described in Proposition 1 and Proposition 2 can also be applied when deleting a subword from the set. If there exists a subword whose image is the same as the subword to be deleted, then it would have no effect on the Algorithm. On the other hand, if the subword and its image is unique, then the deletion of the subword from the set would follow the properties defined by the two previous propositions.*

Moreover, the concepts in Proposition 1 can also be applied to $f_a(w:m)$ *but in a much simpler case as we are now dealing with just the maximum of natural numbers.*

3. RESULTS AND DISCUSSION

In this study, we reconstruct words derived from a finite proper prefix of Kolakoski or A006928 sequence. The lengths of the words ranges from 8 to 20. We performed all our computations on a personal computer and encoded the algorithm in C.

The following results were observed when performing the different rules of removing subwords.

We first tested the ability of the SGG to reconstruct the word just by using a proper subset of $\binom{w}{m}$. We randomly remove subwords and applied the SGG on the ones that are left behind. We do this until the SGG fails to reconstruct w. The following are the results,

Table 1: Randomly Removing Subwords

$\|\|w\|\|$	$\|\|v\|\|$	$\bigcup v$	Subwords Left	Percentage
10	6	210	15	7.14 %
12	8	495	77	15.56 %
16	12	1820	67	3.68 %

Table 1 shows that, given a word w and its subword length m, the SGG can still properly reconstruct a w using only 3% of original number of subwords. This shows that, using SGG, it is still possible to reconstruct a w without finding all of its subwords $\binom{w}{m}$.

Table 2 shows the result of our algorithm applied to the varying lengths of w. The smallest subset of $\binom{w}{m}$ we can obtain for a fixed n and m of w is 5 at the very least. Table 2 also says that our algorithm can reduce the number of subwords to as much as less than 1 %. This also implies that the SGG only needs that much to reconstruct a subword.

Table 2: Prefixes of the Kolakoski Sequence

Length	Subword Lengths and Count		
		$\lceil \frac{n}{2} + 1 \rceil$	$\lceil \frac{3n}{4} \rceil$
	All Subwords	Minimum	Minimum
8	56	5	3
9	126	5	4
10	210	5	5
11	462	7	6
12	792	8	7
13	1716	9	7
14	3003	9	8
15	6435	10	7
16	11440	10	8
17	24310	10	8
18	43758	11	9
19	92378	15	12
20	167960		10

We note that the set of subwords $\binom{w}{m}_0$ is not enough for the SGG to reconstruct w but when $m = \lceil \frac{n}{2} + 1 \rceil$, there are instances where there exists k such that $\binom{w}{m}_k$ can reconstruct w using SGG. When $m = \lceil \frac{3n}{4} \rceil$, we cannot find k such that $\binom{w}{m}_k$ can reconstruct w using SGG.

Our algorithm shows that not all subwords of a certain length of the prefix w of Kol is needed by the SGG. Moreover, by the method of exhaustion, our algorithm can find a small set of subwords that can still reconstruct w using SGG. Our algorithm is a direct result of the propositions discussed in section 2.3 which guarantees that the removal of subwords does not alter the results of SGG given that the subwords removed does not uniquely answer any of the queries of SGG. Giving emphasis on the classification of subwords by order as defined from section 2.1, our algorithm shows that for the case where $m = \lceil \frac{n}{2} + 1 \rceil$, the SGG can reconstruct w using only a single set with order j. For the case where $m = \lceil \frac{3n}{4} + 1 \rceil$, the SGG needs at least two sets with orders k and l to reconstruct w. Note that the higher the order of the set, the more subwords it contains. Interestingly enough, the set with the highest order alone cannot reconstruct w using SGG. It might be interesting to study in-depth the structures of the subwords that belong to the output of our algorithm and create a way to generate those subwords using an automata.

Acknowledgement

We thank the Algorithms and Complexity Lab of the Department of Computer Science UPDiliman in facilitating this work.

4. REFERENCES

[1] Dress, A.W.M., Erdös, P.L.. Reconstructing Words from Subwords in Linear Time, *Annals of Combinatorics* **8**. (2004). pp 457-462.

[2] Hopcroft, J.E., Ullman, J.D.*Introduction to Automata Theory, Languages and Computation,* Addison-Wesley, (1979).

[3] Erdös, P.L., Ligeti, P., Sziklai, P., Torney, D.C. Subwords in reverse-complement order, preprint, (2004).

[4] Shallit, J., Allouche, J.P.. *Automatic Sequences: Theory, Applications, Generalizations.* Cambridge University Press, (2003).

[5] Lothaire, M., *Combinatorics of Words,* Addison-Wesley, (1983).

[6] Sipser, M. *Introduction to the Theory of Computation.* PWS Publishing Company, 1997.

[7] V.I. Levenshtein. On perfect codes in deletion and insertion metric, *Discrete Math. Appl. 2* (1992) pp 241-258.

[8] V.I. Levenshtein, Efficient reconstruction of sequences from their subsequences or supersequences, *J.Combin. Theory Ser. A 93* (2001), pp 310-332.

[9] I. Simon, Piecewise testable events, *In H. Brakhage (ed).: Automata Theory and Formal Languages* , LNCS **33** Springer Verlag, (1975), pp. 214-222.