A fast algorithm for the Kolakoski sequence

Richard P. Brent

Australian National University
and University of Newcastle

13 December 2016
(updated 30 Dec. 2016)

Joint work with Judy-anne Osborn

Abstract

The *Kolakoski sequence* $122112122122112112\ldots$ is the unique countable $\{1,2\}$-sequence $k_1k_2k_3\ldots$ such that $k_1 = 1$, whose j-th block has length k_j. We shall briefly survey what is known and conjectured about this sequence.
Define the *Kolakoski discrepancy function* $\delta(n) := \sum_{1\le j\le n}(-1)^{k_j}$.
It is an open question whether $\delta(n) = o(n)$, i.e. whether the density of 1's in k is $\frac{1}{2}$.
The obvious algorithm to compute $\delta(n)$ takes linear time and space. Nilsson (2012) gave an algorithm for computing $k_1\ldots k_n$ (and hence $\delta(n)$) in time $O(n)$ and space $O(\log n)$. We shall give an algorithm that computes $\delta(n)$ faster, using a space-time tradeoff. It is conjectured that the algorithm runs in time $O(n^\alpha)$ and space $O(n^\alpha)$, where $\alpha = \log(2)/\log(3) \approx 0.631$.
Using our algorithm, we have computed $\delta(n)$ for various $n \le 5\times10^{17}$. The results provide numerical evidence for the conjecture that $\delta(n) = \widetilde{O}(n^{1/2})$.

Notation

$\widetilde{O}(f(n))$ means $O(f(n)(\log n)^c)$ for some $c \geq 0$.

We consider finite or countably infinite sequences over an alphabet Σ of two symbols. In the exposition we take $\Sigma := \{1, 2\}$.
In programs it is often more convenient to use $\Sigma := \{0, 1\}$.

We use *sequence*, *string* and *word* interchangeably
(in the literature some distinctions are made – e.g. strings are sometimes assumed to be finite).

If $\sigma = \sigma_1 \ldots \sigma_n$ is a finite sequence, the *length* of σ is $\lambda(\sigma) := n$.

More notation

$\mathbb{N}$ is the set of positive integers.

The *empty string* (of length zero) is written as ε.

Σ^n is the set of 2^n strings $\sigma_1 \ldots \sigma_n$ of length exactly $n \geq 0$.

$\Sigma^* = \cup_{n \geq 0} \Sigma^n$ is the set of finite strings over Σ.

$\Sigma^{\mathbb{N}}$ is the set of countably infinite strings $\sigma_1\sigma_2\sigma_3\ldots$ over Σ.

σ^j is a string of j consecutive σ s ($\sigma^0 = \varepsilon$, $\sigma^{j+1} = \sigma^j\sigma$).

If $\sigma = \sigma_1^{j_1}\sigma_2^{j_2}\sigma_3^{j_3}\ldots$ where $\sigma_i \neq \sigma_{i+1}$ and $j_i > 0$, then $\sigma_n^{j_n}$ is the n-th *run* in σ and its length is j_n.

Run-length decoding

We can define a "run-length decoding" function $T : \Sigma^* \to \Sigma^*$ by

$$T(\sigma_1\sigma_2\ldots) = 1^{\sigma_1}2^{\sigma_2}1^{\sigma_3}2^{\sigma_4}\ldots$$

For example, if $\sigma = 12221121$, then $T(\sigma) = 122112212112$.

σ is called the *mother* of $T(\sigma)$.

Strings can be motherless. In our example, σ contains a run of length three, so has no mother in $\{1, 2\}^*$.

The Kolakoski sequence k

The *Kolakoski sequence* k is the (unique) fixed point of the map $\sigma \in \Sigma^{\mathbb{N}} \mapsto T(\sigma)$. In other words, $k = T(k)$, and this defines $k \in \Sigma^{\mathbb{N}}$ uniquely.

Proof of uniqueness (sketch). Suppose $k = k_1 k_2 \ldots$ satisfies $k = T(k)$. Then $k_1 k_2 k_3 \ldots = 1^{k_1} 2^{k_2} 1^{k_3} \ldots$, where $1 \le k_j \le 2$. Comparing the first symbols on left and right shows that $k_1 = 1$. Thus $1 k_2 k_3 \ldots = 12^{k_2} \ldots$. Thus $k_2 = 2$ and $k_3 = 2$. Proceeding in this way, we see that, for $j \ge 3$, k_j on the left is defined by $k_1, k_2, \ldots, k_m$ on the right, for some $m < j$. This shows that the solution is unique (and also gives an algorithm for computing it, thus showing existence). $\square$

A small generalisation

We can define

$$T_1(\sigma_1\sigma_2\ldots) = 1^{\sigma_1}2^{\sigma_2}1^{\sigma_3}2^{\sigma_4}\ldots$$

and

$$T_2(\sigma_1\sigma_2\ldots) = 2^{\sigma_1}1^{\sigma_2}2^{\sigma_3}1^{\sigma_4}\ldots$$

(so T_1 is the same as T, and T_2 differs only in that its output string starts with 2 instead of 1).

We say that $\sigma = \sigma_1\sigma_2\ldots$ is the *mother* of both $T_1(\sigma)$ and $T_2(\sigma)$.

Thus, starting from any $u \in \Sigma^*\backslash\{\varepsilon\}$ as the root, we obtain an (infinite) binary tree where the left child of σ is $T_1(\sigma)$ and the right child is $T_2(\sigma)$.

Remarks on fixed points

The equation $T_2(\widehat{k}) = \widehat{k}$ has a unique nonempty solution

$$\widehat{k} = 22112122122\ldots \in \Sigma^{\mathbb{N}},$$

and this solution is the same as the Kolakoski sequence k with the first symbol deleted, i.e.

$$k = 1\widehat{k}.$$

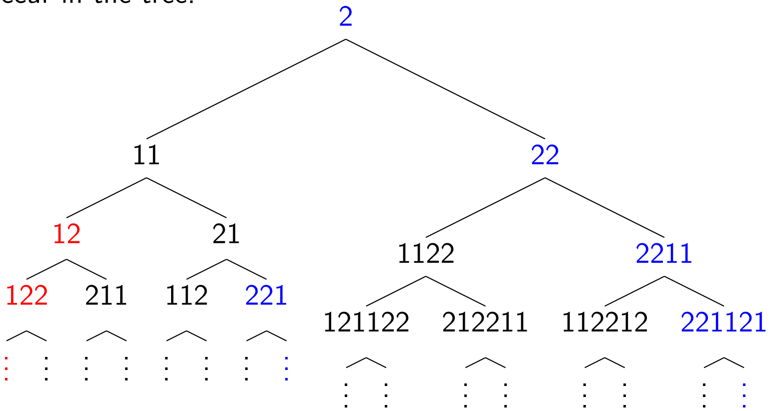
Sometimes the Kolakoski sequence is defined to be $\widehat{k}$ instead of k. We follow the definition in Sloane's OEIS (sequence A000002).

If we allow finite sequences, i.e. $\sigma \in \Sigma^*$, then there are some "trivial" fixed points:

$$T_1(\varepsilon) = \varepsilon, \quad T_2(\varepsilon) = \varepsilon \text{ and } T_1(1) = 1.$$

Example of a tree generated by T_1 and T_2

Starting from $\sigma = 2 \in \Sigma^*$, we obtain the following (infinite) tree whose nodes are finite $\{1, 2\}$-sequences. Note that finite segments of the Kolakoski sequences $k = 122112\ldots$ and $\widehat{k} = 221121\ldots$ occur in the tree.



Note the sequence $(b_j)_{j\geq 1} = (1, 2, 4, 6, \ldots)$ of lengths of nodes on the rightmost path: we will consider this later.

Some history

The "Kolakowski" sequence was first defined (but not named) in a paper by Rufus Oldenburger (1939).

In 1965, William Kolakoski (in the problem section of the *American Mathematical Monthly*) gave the first 41 terms of the sequence and asked

1. What is a simple rule for constructing the sequence?
2. What is [a formula for] the n-th term?
3. Is the sequence periodic?

In 1966, Necdet Üçoluk answered Kolokoski's questions 1 and 3 (answer to 3: the sequence is not periodic). Kolakoski and Üçoluk were apparently unaware of Oldenburger's paper.

History continued

Michael Keane (1991) and others conjectured that the density of 1s in k is $\frac{1}{2}$. This is still open. Even the existence of a density is open. Vašek Chvátal (1993) proved an upper density of 0.501 (and a corresponding lower density of 0.499).

Michaël Rao (2012) improved these bounds to 0.5001 and 0.4999.

Arturo Carpi (1994) proved that the Kolakoski sequence is cube-free, and contains only squares of length $2, 4, 6, 18$ and 54. Here a *square* is a nonempty substring of the form xx (with length defined to be $\lambda(xx) = 2\lambda(x)$), and a *cube* is a nonempty substring of the form xxx.

For other historical references, see the OEIS entry A000002.

Some open questions

There are many open questions/conjectures about the properties of the Kolakoski sequence k. Here is a subset from a report by Dekking (1995).

- What is the *subword complexity* of k, i.e. the cardinality $P(n)$ of the set of words of length n that occur in k? Dekking (1981) showed that $P(n)$ has polynomial growth.
- If a word $x = x_1 \ldots x_n$ occurs in k, does it occur infinitely often? Does it occur with bounded gaps? Does the *reverse* $x_n \ldots x_1$ occur? Does the *complement* $x_1' \ldots x_n'$ occur? (Here $1' = 2, 2' = 1$.)
- *Equidistribution*: Does the frequency of 1 in k exist, and is it equal to $1/2$?
- *Subword equidistribution*: For each word w in k, does the frequency of w in k exist?

Equidistribution

For a string $\sigma = \sigma_1 \ldots \sigma_n \in \Sigma^*$, recall that $\lambda(\sigma) = n$ denotes the length of σ. For $x \in \Sigma$, define $\lambda_x(\sigma)$ to be the number of occurrences of x in σ. For example, $\lambda_1(1121) = 3$, $\lambda_2(1121) = 1$.

By an abuse of notation, we abbreviate $\lambda_x(k_1 \ldots k_n)$ by $\lambda_x(n)$.

k is equidistributed if

$$\lim_{n\to\infty} \frac{\lambda_1(n)}{n} = \lim_{n\to\infty} \frac{\lambda_2(n)}{n} = \frac{1}{2}\,.$$

Since $\lambda_1(n) + \lambda_2(n) = n$, an equivalent condition is

$$\lim_{n\to\infty} \frac{\lambda_1(n)}{n} = \frac{1}{2}\,.$$

Equivalent conditions

Since

$$\sum_{j=1}^{n} k_j = \lambda_1(n) + 2\lambda_2(n),$$

an equivalent condition is

$$\lim_{n\to\infty} \frac{1}{n}\sum_{j=1}^{n} k_j = \frac{3}{2}\,.$$

In other words, the "expansion factor" in going from $k_1 \ldots k_n$ to its child $1^{k_1}2^{k_2}1^{k_3}\cdots(\cdots)^{k_n}$ is asymptotically equal to $3/2$.

Another equivalent condition (stated in the abstract) is $\delta(n) = o(n)$, where

$$\delta(n) := \sum_{j=1}^{n}(-1)^{k_j} = \lambda_2(n) - \lambda_1(n) = n - 2\lambda_1(n).$$

Algorithms

Since there is no known theoretical approach to proving equidistribution of k, there is an interest in computing $\delta(n)$ for large n to see if we can obtain numerical evidence for/against equidistribution.

A linear time and space algorithm

It is straightforward to generate $k_1 k_2 \ldots k_n$ in linear fashion, using the fact that k is a fixed point of the “run-length decoding” function $T_1(\sigma_1\sigma_2\ldots) = 1^{\sigma_1}2^{\sigma_2}1^{\sigma_3}2^{\sigma_4}\ldots$

Denote the complement of $x \in \Sigma$ by x'. Thus $1' = 2,\ 2' = 1$, but we can represent 2 by 0 if desired (to save space).

We use an array A of $n+1$ bits $A_1 \ldots A_{n+1}$, and indices i, j.

Algorithm 1:

```
$(A_1, A_2, i, j) \leftarrow (1, 1', 2, 2)$;
while $i \le n$ do
  if $A_j = 1$ then $(A_i, i, j) \leftarrow (A'_{i-1}, i+1, j+1)$
    else      $(A_i, A_{i+1}, i, j) \leftarrow (A'_{i-1}, A'_{i-1}, i+2, j+1)$.
```

Now $A_1 \ldots A_n = k_1 \ldots k_n$.

The algorithm uses time $O(n)$ and space $O(n)$.

Illustration of Algorithm 1

The algorithm can be illustrated by the following table. As the indices i and j increase, the third and fourth columns both represent initial segments of the Kolakoski sequence.

i increases by 2 when $A_j = 2$ (skipped values in parentheses).

i	j	A_i	A_j
1	1	1	1
2	2	2	2
(3)		2	
4	3	1	2
(5)		1	
6	4	2	1
7	5	1	1
8	6	2	2
(9)		2	
10	7	1	1

Saving space via recursion

Nilsson (2012) improved Algorithm 1 by reducing the space required to generate k_n (or $\delta(n)$) from $O(n)$ to $O(\log n)$. The time required is still $O(n)$.

The idea is to generate the Kolakoski sequence by a recursive procedure, where the reference to A_j in Algorithm 1 is replaced by a recursive call to the same procedure, unless $j \le 2$.

This can be illustrated by the table on the next slide.

Recursive generation of the Kolakoski sequence

Each column $A, B, C, \ldots$ gives the Kolakoski sequence. Column B generates column A, column C generates column B, etc. The depth of recursion increases at each blue row (at indices $b_j + 2$).

index	A	B	C	D	E	F	G	...
1	1	1	1	1	1	1	1	...
2	2	2	2	2	2	2	2	...
3	2							
4	1	2						
5	1							
6	2	1	2					
7	1	1						
8	2	2	1	2				
9	2							
10	1	1	1					
11	2	2	2	1	2			
12	2							
13	1	2						
14	1							
15	2	1	1	1				
16	1	2	2	2	1	2		
17	1							
18	2	2						
19	2							
20	1	1	2					
21	2	1						
22	1	2	1	1	1			
23	1							
24	2	1	2	2	2	1	2	
...								

A sublinear algorithm (Algorithm 2)

Nilsson's algorithm takes time of order n to generate k_n or $\delta(n)$. This is the best that we can do if all of $k_1, \ldots, k_n$ are required.

However, it is possible to generate a single k_n or $\delta(n)$ value (or a sparse sequence of such values) in time $\ll n$.

The first step is to convert Nilsson's recursive algorithm into an iterative algorithm that generates the table on the previous slide row by row. This gives a non-recursive algorithm with essentially the same time and space requirements as Nilsson's algorithm.

Next, we observe that a row in the table determines some of the following rows, as illustrated on the following slide.

One row determines several following rows

index	A	B	C	D	E	F
1–10	...					
11	2	2	2	1	2	
12	2					
13	1	2				
14	1					
15	2	1	1	1		
16	1	2	2	2	1	2
...						

Row 11 determines rows 12–15 as the number of columns (5) in row 11 is at least the number of columns in rows 12–15.

However, row 11 does not determine row 16, since row 16 has 6 columns, and the column labelled "F" could have an entry 1 or 2.

Using table lookup

For each d, $1 \le d \le d_{max}$, where d_{max} is determined by the amount of random-access memory available, we can construct a table of 2^d entries indexed by d-bit binary integers (the *keys*). Each such integer m corresponds to a row (say row r) with d symbols from Σ, which we can map to $\{0, 1\}$.

The table tells us how far we can skip ahead, say ℓ rows, and sufficient information to construct row $r + \ell$ from row r. It should also contain $\sum_{r<j\le r+\ell} (-1)^{k_j}$ so there is enough information to determine $\delta(n)$ when we reach row n.

In fact, it is sufficient to consider rows ending in 2, since rows ending in 1 have $\ell = 0$.

If a row has length $> d_{max}$, or if the table lookup would skip over the desired index n, we revert to the "slow but sure" iterative version of Nilsson's algorithm.

Space-time tradeoff

The mean value of ℓ over all 2^d d-bit keys is $(3/2)^d$. On the assumption that each possible key is equally likely to occur, we expect a speedup of order $(3/2)^{d_{max}}$, ignoring logarithmic factors. This is confirmed by numerical experiments.

Note: the keys are *not* substrings of the Kolakoski sequence. The latter can not include strings such as 111 or 222, so the number of length-d substrings that occur in the Kolakoski sequence is (much) less than 2^d. (It is polynomial in d.)

We have to initialise the lookup tables. This takes time $\widetilde{O}(2^{d_{max}})$ if done recursively, starting from small d.

Thus, the total time to compute $\delta(n)$ is [conjectured to be]

$$\widetilde{O}((2/3)^{d_{max}} n + 2^{d_{max}}).$$

Choosing d_{max}

Choosing $d_{max} = \lfloor \log(n)/\log(3) \rfloor$ gives time $\widetilde{O}(n^{\alpha})$ and space $O(n^{\alpha})$, where $\alpha = \log(2)/\log(3) \approx 0.631$.

For the values of n that we are interested in, say $n \approx 10^{18}$, this would give $d_{max} \approx 37$. Our program uses about $24 \times 2^{d_{max}}$ bytes. Thus, in practice, we are often limited by the availability of memory and have to choose d_{max} smaller than the theoretically optimal value. For $d_{max} \le \log(n)/\log(3)$, the runtime is $\widetilde{O}((2/3)^{d_{max}}\, n)$.

One way to interpret this analysis is as follows. Each time that we double the amount of memory available for lookup tables, we obtain a speedup by a factor of about $3/2$ (for sufficiently large n).

Computing other quantities

It is possible to compute other functions related to $\delta(n)$ with only a constant factor slowdown by variants on the algorithm described for $\delta(n)$.

For example, we can compute

$$\max_{a \le n \le b} \delta(n) \quad \text{and} \quad \min_{a \le n \le b} \delta(n), \text{ and hence } \max_{a \le n \le b} |\delta(n)|,$$

at the expense of a factor $5/3$ in memory for lookup tables, and a small constant factor increase in running time.

Computation of $\delta(n)$

The following table gives some values of $\delta(n)$ and scaled values of $\delta(n)$ and $\Delta(n)$, where $\Delta(n) := \max_{1 \le j \le n} |\delta(j)|$.

n	$\delta(n)$	$\delta(n)/n^{1/2}$	$\Delta(n)/n^{1/2}$
10^3	-4	-0.1265	0.1897
10^6	+28	+0.0280	0.0660
10^9	-2,446	-0.0773	0.1560
10^{12}	-101,402	-0.1014	0.1515
10^{15}	-1,954,842	-0.0618	0.1390
5×10^{17}	+40,997,288	+0.0580	0.0921

Our computation took about 3.5 hours per block of 10^{15} on a 2GHz Intel Xeon running Linux, using 80GB memory ($d_{max} = 31$).

Conjecture

From our $\Delta(n)$ computations we can conclude that

$$|\delta(n)| < n^{1/2}/4 \text{ for } 2000 \le n \le 5 \times 10^{17}.$$

Thus, it is natural to conjecture that $\delta(n) = \widetilde{O}(n^{1/2})$.

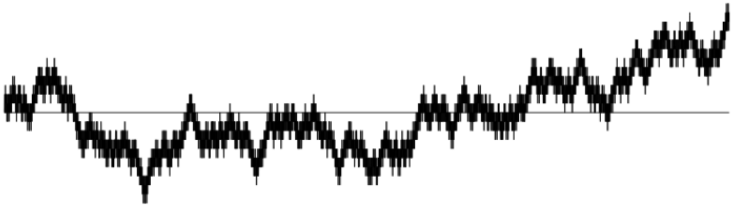
For a random walk with i.i.d. random variables we would get $\delta(n) \ll \sqrt{n \log\log n}$ almost surely (this is Khinchin's *law of the iterated logarithm*).

The function $\sqrt{\log\log n}$ grows far too slowly to make a significant difference to the numerical results. Thus, we only conjecture $\widetilde{O}(n^{1/2})$, although the numerical results suggest $O(n^{1/2})$.

Fractal nature of the Kolakoski sequence

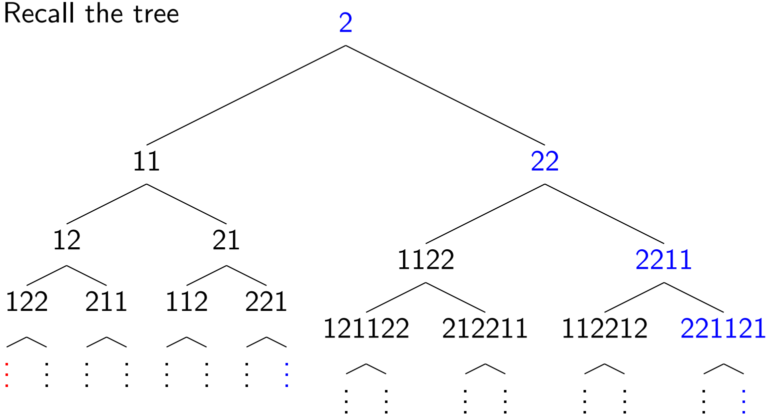
A table lookup algorithm can speed up computation of $\delta(n)$ because the Kolakoski sequence exhibits some self-similarity, i.e. in some sense it is a *fractal* (I won't try to define this).

The graph shows $\delta(n)$ for $n \leq 8000$. If you look closely you can see evidence of self-similarity.



The sequence $1, 2, 4, 6, 9, 14, 22, 33, \ldots$

Recall the tree



OEIS sequence A042942 *"From substitutional generation of Kolakoski sequence"* is the sequence $(b_j)_{j\geq 1} = (1, 2, 4, 6, \ldots)$ of lengths of the sequence $2, 22, 2211, 221121, \ldots$ obtained by starting at the root (2) and always taking the rightmost branch. The nodes are initial subsequences of the Kolakoski sequence $\widehat{k}$.

Computation of the sequence of lengths

If $b_1, b_2, b_3, b_4, \ldots = 1, 2, 4, 6, \ldots$ and at depth d of the tree we have

$$b_d = \lambda(k_2 k_3 \ldots k_n) = n - 1,$$

then

$$b_{d+1} = \lambda_1(n) + 2\lambda_2(n) - 1 = \frac{3n + \delta(n)}{2} - 1.$$

Thus

$$b_{d+1} = \frac{3b_d + \delta(b_d + 1) + 1}{2}$$

Thus, the conjecture $\delta(n) = o(n)$ implies that $b_{d+1} \sim 3b_d/2$.

Starting from $b_1 = 1$, our program for the function δ can compute $b_2, b_3, \ldots, b_{d+1}$ in [conjectured] time $\widetilde{O}((b_d)^\alpha)$, where $\alpha \approx 0.631$.

Computational results for $b_1, b_2, \ldots$

Benoit Cloitre conjectured that $b_d \sim c(3/2)^d$ where $c \approx 1.3094$.

The OEIS entry A042942 gives $b_1, \ldots, b_{69} = 1850766589061$, computed by David Spies. Our program verifies these values in 161 seconds using 384MB of memory (speedup about 160).

We have extended the table of Spies to

$$b_{100} = 532329637135234861$$

by a computation that took 33.5 days, using 96GB of memory on a 2Ghz Intel Xeon ($d_{max} = 32$). To store b_{100} bits would require 6.65×10^{16} bytes of memory, but our program does not store (or even explicitly generate) all these bits. We find that

$$b_{100}/b_{99} = 1.49999999997275\ldots$$

and estimate

$$c \approx \frac{b_{100}}{(3/2)^{100}} \approx 1.309346948 \pm 1\,\text{ulp}$$

(the error is at most 1.2×10^{-9} if $|\delta(n)| \leq n^{1/2}/4$ for large n).

Rao's computation

Michaël Rao (2012) computed $\lambda_1(n)$[1] by an algorithm based on composition of finite state transducers, for various $n \le 10^{18}$. We have confirmed the values that he gives where they overlap our computations (up to $n = 5 \times 10^{17}$).

Rao does not give any information about running times or storage requirements, so we can not compare the efficiency of his algorithm with that of our Algorithm 2.

So far as we know, Rao did not compute $\Delta(n)$ or the A042942 sequence (b_n).

[1] Recall that $\delta(n) = n - 2\lambda_1(n)$, so computing $\lambda_1(n)$ is essentially equivalent to computing $\delta(n)$.

References

A. Carpi, On repeated factors in C^∞-words, *Information Processing Letters* **52** (1994), 289–294.

V. Chvátal, *Notes on the Kolakoski sequence*, DIMACS Tech. Report 93-84, Dec. 1993. `http://dimacs.rutgers.edu/TechnicalReports/abstracts/1993/93-84.html`

F. M. Dekking, *What is the long range order in the Kolakoski sequence?*, Report 95–100, TU-Delft, 1995. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.6839`

W. Kolakoski, Self generating runs, Problem 5304, *Amer. Math. Monthly* **72** (1965), 674. Partial solution: Ü. Necdet, *ibid* **73** (1966), 681–682.

J. Nilsson, A space-efficient algorithm for calculating the digit distribution in the Kolakoski sequence, *J. of Integer Sequences* **15**, article 12.6.7 (2012).

References cont.

R. Oldenburger, Exponent trajectories in symbolic dynamics, *Trans. Amer. Math. Soc.* **46** (1939), 453–466.

OEIS, Sequence A000002, Kolakoski sequence, http://oeis.org/A000002.

OEIS, Sequence A042942, from substitutional generation of the Kolakoski sequence A000002, http://oeis.org/A042942.

OEIS, Sequence A088568, $3*n-2*$(partial sums of Oldenburger-Kolakoski sequence A000002), http://oeis.org/A088568.

M. Rao, *Trucs et bidules sur la séquence de Kolakoski*, Oct. 1, 2012. http://www.arthy.org/kola/kola.php.

Wikipedia, *Kolakoski sequence*, https://en.wikipedia.org/wiki/Kolakoski_sequence.