Journal of Integer Sequences, Vol. 10 (2007),
Article 07.6.5

The Minimal Density of a Letter in an Infinite Ternary Square-Free Word is $\frac{883}{3215}$

Andrey Khalyavin
Mech. & Math. Department
Moscow State University
119992 Moscow
Russia
halyavin@land.ru

Abstract

The problem of determining the minimal density of a letter in an infinite ternary square-free word was investigated by Tarannikov and Ochem. In this paper we solve this problem, and prove that the minimal density is equal to $\frac{883}{3215}$.

1 Introduction

A word is called *square-free* if it cannot be written in the form $axxb$ for two words a, b and a nonempty word x. In this paper we study the minimal density of a letter a in an infinite square-free word over the alphabet $\{a, b, c\}$. The *density* of a letter a in a finite word w is $\frac{n(w)}{|w|}$ where $n(w)$ is the number of letters a in w and $|w|$ is the length of the word. The *density* of a letter a in an infinite word is $\liminf\limits_{l\to\infty} \frac{n(w_l)}{|w_l|}$ where w_l is the prefix of the word w of length l. Tarannikov [1] found the minimal density of a letter a with up to 4 tight digits after the decimal point. Ochem [2] improved this result, proving that the minimal density of a letter a belongs to the interval $[\frac{1000}{3641}, \frac{883}{3215}]$. In this article we completely solve this problem and prove that the minimal density of a letter a is equal to $\frac{883}{3215}$. We use huge computer calculations in our proof. You can find sources of the programs at `http://mech.math.msu.su/department/dm/dmmc/PUBL/words.zip`.

2 Main result

Our aim is to prove that the minimal density is at least $\rho = \frac{883}{3215}$. Let us construct 4 special words $b_1, \ldots, b_4$ of length 3215 that contain 883 letters a each (they are probably the same words used by Ochem [2] in order to construct an infinite word with density ρ). We call these words the *basic words*.The files with these basic words can be found at the URL address above. We proved the next theorem by a computation using the technique of backtracking.

Theorem 1. *In any square-free word of length 90000 there exist either*

1) *a prefix with density greater than or equal than ρ,*

2) *two adjacent basic words (a subword b_ib_j).*

So, either an infinite square-free word can be split into words with densities greater than or equal than ρ or there exist two basic words one after another. In the first case the density of an infinite word is greater than or equal than ρ because the lengths of the words in the splitting are bounded by 90000. So we can assume that an infinite square-free word begins with a word b_ib_j.

When our program that proves Theorem 1 finds two adjacent basic words in the current word w, it saves the pair $(n(w), |w|)$ and backtracks. Let us denote by W the set of all such current words. Then we delete equal pairs $(n(w), |w|)$ from the list and obtain 80 distinct pairs $(n(w), |w|)$. Let us denote by L the set of these pairs. Then for all $l < 200000$ we calculate the maximal number $p(l)$ such that after the concatenation of any words $w \in W$ and v such that $n(v) \geq p(l), |v| = l$, the word vw has the density greater than or equal than ρ. So, $p(l) = \max_{(n(w),|w|)\in L}[\rho(l + |w|)] - n(w) + 1$. Our second program proves the next theorem.

Theorem 2. *Let u be an infinite square-free word that begins with a word b_ib_j. Then u contains one of the following prefixes:*

1) *a prefix w such that $2 \cdot 3215 < |w| < 200000$ and $n(w) \geq p(|w|)$,*

2) *a prefix w such that $4 \cdot 3215 \leq |w| < 200000, n(w) \geq \rho|w|$, that ends with two adjacent basic words.*

Now, we are prepared to prove

Theorem 3. *An infinite square-free word that begins with a word b_ib_j has density not less than ρ.*

Proof. Let us consider an infinite square-free word u that begins with a word b_ib_j. Let us show that u must either have density greater than or equal than ρ or start with one of the following prefixes:

1) the prefix $ww_1w_2 \ldots w_nv$ such that $|w| < 200000, |w_i| < 200000, 2 \cdot 3215 < |v| < 200000, n(w_i) \geq \rho|w_i|, n(wv) \geq \rho|wv|$, which ends with two adjacent basic words.

2) the prefix w such that $4 \cdot 3215 \leq |w| < 200000, n(w) \geq \rho|w|$, which ends with two adjacent basic words.

Let us apply Theorem 2 to our infinite square-free word. If the second case of Theorem 2 holds, we have the second case in the above sentence. Otherwise we apply Theorem 1 to the word u without the prefix w. We obtain either A) a subword v that ends with two adjacent basic words or B) a subword w_1 with the density greater than or equal than ρ. In the case A) we have the second case in our sentence since $n(w) \geq p(|w|)$. In the case B) we apply Theorem 1 to the word u without the prefix ww_1, and so on. If the second case of Theorem 1 never occurs then $u = ww_1w_2\dots$ where lengths of words w and w_i are bounded. Therefore the density of u is greater than or equal than ρ. If the second case of Theorem 1 occurs on the nth step, we obtain the prefix $ww_1\dots w_{n-1}v$ such that $n(w) \geq p(|w|), v \in W$ and so $n(wv) \geq \rho|wv|$.

Therefore our word u can be split into subwords w and $ww_1w_2\dots w_nv$ with densities greater than or equal than ρ and bounded lengths of words w,w_i and v. Therefore the word u also has the density greater than or equal than ρ. □

From Theorem 1 and Theorem 3 we obtain immediately our main Theorem:

Theorem 4. *The density of a letter a in an infinite square-free word is greater than or equal than ρ.*

Using the result from Ochem [2] and Theorem 4 we obtain that the minimal density of an infinite square-free word is equal to ρ.

3 Optimization of calculations

We have spent about 12 hours of calculations (on a 2GHz CPU) in order to prove Theorems 1 and 2. In order to speed up calculations we used a special algorithm for checking the square-free property.

The program proving the theorems above uses a backtracking algorithm. So we must check on every step if the current word S ends with a word of the form ww. Although the algorithm described below uses $O(n^2)$ time in the worst case for checking the square-free property, on average it uses sublinear time. First, we choose m. In our experiments the optimal m was about 100. Then we calculate the exact and inexact hash for all positions from the mth character to the end of the current string. Let us denote by S_i the number corresponding to the ith symbol ($S_i = 0$ for symbol a, $S_i = 1$ for symbol b, $S_i = 2$ for symbol c). Let us define the *inexact hash* for the jth position as $h_j = S_j + S_{j-1}p + S_{j-2}p^2 + \dots + S_{j-m+1}p^{m-1} (\mod P)$ where $p = 5$ and P is the size of the hash table (we use $P = 100003$ for Theorem 1 and $P = 200003$ for Theorem 2). If there do not exist positions before the jth with the exact hash h_j then we define the exact hash of the jth position equal to the inexact hash. If such positions exist, we check the last m symbols before the jth position. If the last m symbols before the jth position coincide with the last m symbols before kth position with the exact hash h_j, we define the exact hash for the jth position equal to h_j. In the opposite case we check all positions that have the exact hash equal to $h_j + 1$ and repeat the previous procedure. Then we check positions with the exact hash equal to $h_j + 2$ and so on. We continue to increase the hash value until we have found a position with the same last m characters or hash value that does not correspond to any position in the string. We define the exact hash of the jth position equal to the last checked hash value. In order to check hash values quickly we use the table where for each hash value the last position with the same exact hash is written. Also for each position in the string we store the nearest position before it with the same exact hash. Thus, all positions with the same exact hash are linked to the list.

Hence, the exact hash of the position uniquely defines the last m symbols before it and we can quickly update it after deleting or inserting a symbol to the end of the current string.

In order to check if the current word S ends with a word of the form ww, we check possible lengths (of word w) less than m directly. Then we search all positions with exact hash equal to the exact hash of last position. Then we compare strings for these positions. We speed up this procedure by m times using exact hash values.

4 Acknowledgments

The author is grateful to his scientific supervisor Prof. Yuriy Tarannikov, Dr. Roman Kolpakov and the anonymous referee for the attention to this work and helpful remarks.

References

[1] Y. Tarannikov. The minimal density of a letter in an infinite ternary square-free word is 0.2746... J. Integer Sequences 5(2): Article 02.2.2 (2002).

[2] P. Ochem. Letter frequency in infinite repetition-free words. Proceedings of Words 2005, Montreal, Canada, September 13-17 (2005), 335-340.

2000 *Mathematics Subject Classification*: Primary 11B05.
Keywords: combinatorics on words; square-free word; factorial languages; minimal density.

(Concerned with sequence A006156.)

Received June 19 2006; revised versions received May 5 2007; June 12 2007. Published in *Journal of Integer Sequences*, June 14 2007.

Return to Journal of Integer Sequences home page.