ISSN 0001-4346, Mathematical Notes, 2014, Vol. 96, No. 1, pp. 43–58. © Pleiades Publishing, Ltd., 2014.
Original Russian Text © Yu. S. Medvedeva, 2014, published in Matematicheskie Zametki, 2014, Vol. 96, No. 1, pp. 51–69.

{FirstPage:x0

Fast Enumeration of Words Generated by Dik Grammars

Yu. S. Medvedeva*

Institute of Computational Technologies, Russian Academy of Sciences, Novosibirsk, Russia

Received August 15, 2013; in final form, December 25, 2013

Abstract—The problem of enumerating and denumerating words generated by Dik grammars arises in the work of compilers for high-level programming languages and a number of other applications. The present paper proposes an algorithm for the fast enumeration and denumeration of words of Dik languages; the complexity of this algorithm per one symbol of enumerated words is $O(\log^3 n \log\log n)$ bit operations, provided that the Schönhage–Strassen multiplication and division algorithm is used. The well-known methods applied earlier possess complexity $O(n)$ per one symbol of enumerated words. The construction of the proposed algorithm is based on the Ryabko method.

DOI: 10.1134/S0001434614070049

Keywords: *fast enumeration and denumeration of words, Dik language, Schönhage–Strassen multiplication and division algorithm, bracketing, Fürer fast multiplication algorithm.*

1. INTRODUCTION

{ssec1}

The problem of enumerating words from an ordered set W is as follows: to each word from W we assign its number, i.e., a unique number from the range $[0; |W| - 1]$. In the denumeration problem, we search for the solution of the inverse problem: from a number in the range $[0; |W| - 1]$ we must find the corresponding word from W. Usually we consider the lexicographic order on W, while the number is expressed in a binary system.

The problems of enumerating combinatorial objects of different form have attracted the attention of many authors. Among these problems, we note the problem of enumerating words with a given number of 0's and 1's and the problem of enumerating words with a bound on the number of consecutive identical symbols [1], [2].

In general form, the problem of enumeration and denumeration was considered in Cover's paper [3], where a general method for any given set of words was proposed. Ryabko [4] was first to propose a fast general algorithm of enumerating words of a given set which, for sets of many combinatorial objects, has a rate exponentially greater than that in the Cover method.

In the present paper, we consider the problem of enumerating and denumerating words of Dik languages [5], or of "regular" sequence of brackets. Words of the Dik language over a $2m$-letter alphabet are sequences of regularly embedded brackets of m types. As an example, consider all words of length $n = 4$ of the Dik language over six letters, i.e., sequences of length 4 of regularly embedded brackets of three types. There are 18 such words (see the table). To them we assign numbers in binary form of the length $\lceil \log_2 18 \rceil = 5$. All such words are placed in the first column, while their numbers in binary form are written in the second column.

To any word belonging to the Dik language over a $2m$-letter alphabet of length $2n$ the enumeration algorithm assigns a sequence of 0's and 1's, i.e., its number. For example, for the set of words of the Dik language over the 6-letter alphabet of length 4 located in the order given by the table, for the given word ( ) { } the algorithm must find its number 00101.

The necessity for the fast enumeration and the denumeration of words of Dik languages arises in the work of compilers of high-level languages related to the compression of regular sequences of brackets and the random generation of regular sequences of brackets [6]–[8].

*E-mail: mjulja@gmail.com

Table

The word	Its number	The word	Its number
(( ))	00000	[ ] [ ]	01001
( ) ( )	00001	[{ }]	01010
([ ])	00010	[ ] { }	01011
( ) [ ]	00011	{ ( ) }	01100
({ })	00100	{ } ( )	01101
( ) { }	00101	{ [ ] }	01110
[( )]	00110	{ } [ ]	01111
[ ] ( )	00111	{ { } }	10000
[[ ]]	01000	{ } { }	10001

The enumeration algorithm of words of length $2n$ belonging to Dik languages over a $2m$-letter alphabet based on the Cover method [3] has complexity $O(n^2)$ of bit operations per one enumerated word or $O(n)$ of bit operations per one symbol of enumerated words.

The method for enumerating words of length $2n$ belonging to Dik languages over a $2m$-letter alphabet proposed in this paper is based on the approach from [4] and has complexity $O(\log nM(n\log n)/n)$ of bit operations per one symbol of enumerated words, where $M(k)$ is the time needed for the multiplication or division of words of length k. While the Schönhage–Strassen method [9] has complexity $O(k\log k\log\log k)$, for multiplication or division of words of length k, the complexity of the method under consideration is $O(\log^3 n\log\log n)$ per one symbol of enumerated words. While the Fürer method [10] has complexity $O(k\log k2^{O(\log^* k)})$ for multiplication or division of words of length k, the complexity of the method under consideration is $O(\log^3 n2^{O(\log^* n)})$ per one symbol of enumerated words.

2. FAST ENUMERATION ALGORITHM FOR WORDS BELONGING TO DIK LANGUAGES {ssec2}

Denote by D_{2n}^{2m} the set of words of the Dik language over a $2m$-letter alphabet of length $2n$. We must find the number of the word w from the set D_{2n}^{2m} among all the words in this set.

Let us describe the enumeration algorithm by giving an example of the search for the number of the word $w = (\,)\,[\,]\,(\,[\,]\,)$ among the words in the set D_8^4, i.e., among all the words of length 8 representing correct bracketings of two types.

Let us replace the opening brackets of all kinds by 0's and the closing brackets of all kinds by 1's. It is easy to see that the resulting word w' will be a word of the Dik language over the alphabet $\{0,1\}$, i.e., it will correspond to a correct bracketing of one kind. In this case, we obtain the word $w' = 01010011$. Denote by S_{2n} the set of all words of n, 0's and n, 1's corresponding to correct bracketings. The cardinality of S_{2n} is equal to the nth Catalan number [11],

$$|S_{2n}| = C_n = C_{2n}^n - C_{2n}^{n-1}.$$

(Here and elsewhere, C_n^m denotes the binomial coefficient $\binom{n}{m}$.) To describe the method, we consider the auxiliary set A_m^n of words of length n belonging to the alphabet $A_m = \{a_0, a_1, \ldots, a_{m-1}\}$. Now, to the enumerated word of the set D_{2n}^{2m} we assign a word w'' belonging to the set A_m^n. To do this, we replace all the opening brackets of the first type by the symbol a_0, all the opening brackets of the second type by the symbol a_1, etc. and eliminate the closing brackets. In this case, the word $(\,)\,[\,]\,(\,[\,]\,)$ will be assigned to the word $w'' = a_0a_1a_0a_1$. It can be seen that the order of types of closing brackets is uniquely determined by the order of types of opening brackets.

Thus, each word of the set D_{2n}^{2m} can be uniquely assigned to a pair of words (w', w''), one belonging to the set S_{2n} and the other to the set A_m^n. And conversely, each pair of words, one from the set S_{2n} and the other from the set A_m^n, uniquely determines the corresponding words of the set D_{2n}^{2m}. Thus,

$$|D_{2n}^{2m}| = |A_m^n| \cdot |S_{2n}| = m^n \cdot (C_{2n}^n - C_{2n}^{n-1}).$$

In the example under consideration,

$$|D_8^4| = |A_2^4| \cdot |S_8| = 2^4 \cdot (C_8^4 - C_8^3) = 2^4 \cdot (70 - 56) = 224.$$

To describe the algorithm, we shall use the following ordering of words from the set D_{2n}^{2m}: first, we put them in the lexicographic order of the corresponding words of the set A_m^n and then in the lexicographic order of the corresponding words of the set S_{2n}.

Denote the number of the word w in the set D_{2n}^{2m} thus ordered by $N(w)$, the number of the word w' in the lexicographically ordered set S_{2n} by $N'(w')$, and the number of the word w'' in the lexicographically ordered set A_m^n by $N''(w'')$. Then it is easy to see that

$$N(w) = N''(w'') \cdot |S_{2n}| + N'(w'). \tag{2.1}$$ {eq2.1}

It is convenient to begin the description of our method with the description of how to find the number $N'(w')$ by using the Cover method [3]. By this method, the number of a word in the lexicographically ordered set S_{2n} can be obtained from the formula

$$N'(x_1x_2\dots x_{2n}) = \sum_{i=1}^{2n} \sum_{\chi<x_i} N_{S_{2n}}(x_1x_2\dots x_{i-1}\chi), \tag{2.2}$$ {eq2.2}

where $N_{S_{2n}}(x_1x_2\dots x_{i-1}\chi)$ is the number of words from S_{2n}, beginning with $x_1x_2\dots x_{i-1}\chi$.

Using $N_{S_8}(01)$ as an example, let us find this number, i.e., the number of words beginning with 01, belonging to the set of words of length 8 composed of 0's and 1's, and corresponding to correct bracketings. The words of the set S_8 beginning with 01 will be the words composed of four 0's and of four 1's beginning with 01, except those not corresponding to correct bracketings. The number of all words composed of four 0's and four 1's beginning with 01 is easy to find; it is equal to the number of all words composed of three 0's and three 1's, i.e., $C_6^3 = 20$.

The words beginning with 01 composed of four 0's and four 1's and not corresponding to correct bracketings are words of four 0's and four 1's for which there exists a j, $2 < j \le 8$ such that the number of 1's in the sequence $x_1x_2\dots x_j$ exceeds the number of 0's in this sequence. There exists a one-to-one correspondence between such words and all words composed of three 0's and five 1's and beginning with 01. Such a correspondence can be defined as follows. For a word not corresponding to correct bracketings, there exists a j, $2 < j \le 8$, such that the number of 1's in the sequence $x_1x_2\dots x_j$ exceeds the number of 0's in this sequence. For each such word, we can find the minimal j. We can see that, for such a j, the number of 1's in the sequence $x_1x_2\dots x_j$ exceeds the number of 0's by one symbol. In this word, let us now replace all the symbols after the jth by the opposite ones. We obtain a word composed of three 0's and five 1's, beginning with 01. Since this mapping is bijective, the number of words beginning with 01, composed of four 0's and four 1's, and not corresponding to correct bracketings is equal to the number of all words beginning with 01 and composed of three 0's and five 1's. The number of such words is the same as the number of words composed of two 0's and four 1's, i.e., $C_6^2 = 15$. Thus,

$$N_{S_8}(01) = C_6^3 - C_6^2 = 5.$$

In general form,

$$\begin{aligned} N_{S_{2n}}(x_1x_2\dots x_i) &= C_{2n-i}^{n-z} - C_{2n-i}^{n-z-1} \\ &= \frac{(2n-i)!}{(n-z)!\,(n-i+z)!} - \frac{(2n-i)!}{(n-z-1)!\,(n-i+z+1)!} \\ &= \frac{(2n-i)!(2z-i+1)}{(n-z)!\,(n-i+z+1)!}, \end{aligned} \tag{2.3}$$ {eq2.3}

where z is the number of 0's in $x_1x_2\dots x_i$, provided that $x_1x_2\dots x_i$ can be the beginning of the word corresponding to the correct bracketing of length $2n$. If $x_1x_2\dots x_i$ cannot be the beginning of the words corresponding to the correct bracketing of length $2n$, then, obviously, $N_{S_{2n}}(x_1\dots x_i)=0$.

We see that $N_{S_{2n}}(x_1\dots x_i)$ depends only on $0<i\le 2n$, n, and z, the number of 0's in $x_1x_2\dots x_i$ $(0<z\le i, 0<z\le n)$. Thus, for a given n, there exists at most $3n^2/2$ different numbers $N_{S_{2n}}(x_1\dots x_i)$. In addition, we see that the values of $N_{S_{2n}}(x_1\dots x_i)$ do not exceed 2^{2n-i}, i.e., are of length not greater than $2n-i$.

We shall use an auxiliary table in which each pair of values of i, $0<i<2n$, and z, $0\le z\le \min(i,n)$, is assigned the value of

$$N_{S_{2n}}(x_1\dots x_i)=\frac{(2n-i)!\,(2z-i+1)}{(n-z)!\,(n-i+z+1)!},$$

identical for all words of length i with z, 0's. The size of the table is $O(n^3)$.

To obtain the number of the word w', we use (2.2) to find the value of $N_{S_{2n}}(x_1x_2\dots x_{i-1}0)$ for i such that $x_i=1$ and then add them together. For each such i, these values are found as follows. We find the value of z, equal to the number of 0's in the word $x_1\dots x_{i-1}0$. Then, using the table, we find the value of $N_{S_{2n}}(x_1\dots x_i)$ corresponding to the pair i, z.

In our example,

$$\begin{aligned}N'(01010011)&=N_{S_8}(00)+N_{S_8}(0100)+N_{S_8}(0101000)\\&\quad+N_{S_8}(01010010)=9+3+0+0=12.\end{aligned}\tag{2.4}$$ {eq2.4}

Here the values of $N_{S_8}(00)$ and $N_{S_8}(0100)$ are taken from the table. For the values of i equal to 2, 4, 6, and 8, we have $x_i=1$. For $i=2$, the word $x_{i-1}0$ is 00, and hence $z=2$; therefore, from the table we find the value of $N_{S_8}(00)$ corresponding to the pair $(2,2)$ and equal to $(6!\,3)/(2!\,5!)=9$. For $i=4$, the word $x_1\dots x_{i-1}0$ is 0100, and hence $z=3$; therefore, from the table we find the value of $N_{S_8}(0100)$, corresponding to the pair $(4,3)$ and equal to $(4!\,3)/(1!\,4!)=3$. The values of $N_{S_8}(0101000)$ and $N_{S_8}(01010010)$ are zero, because there are no words in the set S_8 beginning with 0101000 or 01010010.

We see that, for such a computation, we need to perform a maximum of n operations of addition of words of lengths from 1 to $2n$. Thus, if the auxiliary table is used, then the complexity of the computation of the number of a word by the Cover method is $O(n^2)$ or $O(n)$ per one symbol of enumerated words.

Let us now pass to the description of the proposed method for finding $N'(w')$.

Let us determine the quantities $P(x_i|x_1\dots x_{i-1})$, $q(x_i|x_1\dots x_{i-1})$ for $0<i\le 2n$ as follows:

$$\begin{aligned}P(x_1)&=\frac{N_{S_{2n}}(x_1)}{|S_{2n}|},&\qquad P(x_i|x_1x_2\dots x_{i-1})&=\frac{N_{S_{2n}}(x_1x_2\dots x_i)}{N_{S_{2n}}(x_1x_2\dots x_{i-1})},\\ q(x_1)&=\sum_{\chi<x_1}P(\chi),&\qquad q(x_i|x_1\dots x_{i-1})&=\sum_{\chi<x_i}P(\chi|x_1\dots x_{i-1}).\end{aligned}\tag{2.5}$$ {eq2.5}

We can see that, by (2.2),

$$N'(x_1\dots x_{2n})=|S_{2n}|\big(q(x_1)+q(x_2|x_1)P(x_1)+q(x_3|x_1x_2)P(x_2|x_1)P(x_1)+\cdots\big).\tag{2.6}$$ {eq2.6}

The idea of the method is to perform bracketing in this expression in such a way that, in order to compute the number of a word, the majority of operations is performed over short numbers. Such a bracketing is as follows:

$$\begin{aligned}N'(x_1\dots x_{2n})=|S_{2n}|\big(&(q(x_1)+q(x_2|x_1)P(x_1))\\&+((q(x_3|x_1x_2)+q(x_4|x_1\dots x_3)P(x_3|x_1x_2))P(x_2|x_1)P(x_1))+\cdots\big).\end{aligned}\tag{2.7}$$ {eq2.7}

For $0\le a\le\log(2n)$, $1\le b\le 2n/2^a$, we determine the quantities ρ_b^a, λ_b^a as follows:

$$\begin{aligned}\rho_b^0&=P(x_b|x_1\dots x_{b-1}),&\qquad \lambda_b^0&=q(x_b|x_1\dots x_{b-1}),\\ \rho_b^a&=\rho_{2b-1}^{a-1}\rho_{2b}^{a-1},&\qquad \lambda_b^a&=\lambda_{2b-1}^{a-1}+\rho_{2b-1}^{a-1}\lambda_{2b}^{a-1}.\end{aligned}\tag{2.8}$$ {eq2.8}

Then

$$\lambda^{\log(2n)} = \big((q(x_1) + q(x_2|x_1)P(x_1)) + ((q(x_3|x_1x_2) + q(x_4|x_1\dots x_3)P(x_3|x_1x_2)) \cdot P(x_2|x_1)P(x_1)) + \cdots\big).$$

Combining this with (2.7), we obtain

$$N'(x_1x_2\dots x_{2n}) = \lambda_1^{\log(2n)}|S_{2n}| = \lambda_1^{\log(2n)}(C_{2n}^n - C_{2n}^{n-1}). \tag{2.9}$$ {eq2.9}

We can see that, in the case $0 < i \le 2n$, for $x_i = 0$,

$$\begin{aligned} P(x_i|x_1x_2\dots x_{i-1}) &= \frac{(2n-i)!\,(2z-i+1)}{(n-z)!\,(n-i+z+1)!} \Big/ \frac{(2n-i+1)!\,(2z-i)}{(n-z+1)!\,(n-i+z+1)!} \\ &= \frac{(2z-i+1)(n-z+1)}{(2z-i)(2n-i+1)} \end{aligned} \tag{2.10}$$ {eq2.10}

and, for $x_i = 1$,

$$\begin{aligned} P(x_i|x_1x_2\dots x_{i-1}) &= \frac{(2n-i)!\,(2z-i+1)}{(n-z)!\,(n-i+z+1)!} \Big/ \frac{(2n-i+1)!\,(2z-i+2)}{(n-z)!\,(n-i+z+2)!} \\ &= \frac{(2z-i+1)(n-i+z+2)}{(2n-i+1)(2z-i+2)}. \end{aligned} \tag{2.11}$$ {eq2.11}

Let us now pass to computations serving as an illustration of the algorithm. By formulas (2.10) and (2.11), we obtain

$$P(x_1) = \rho_1^0, \quad P(x_2|x_1) = \rho_2^0, \quad \dots, \quad P(x_8|x_1x_2\dots x_7) = \rho_8^0,$$
$$q(x_1) = \lambda_1^0, \quad q(x_2) = \lambda_2^0, \quad \dots, \quad q(x_8) = \lambda_8^0,$$

and

$$\begin{gathered} P(x_1) = 1, \qquad P(x_2|x_1) = P(1|0) = \frac{5}{14}, \qquad P(x_3|x_1x_2) = P(0|01) = \frac{6}{6}, \\ P(x_4|x_1x_2x_3) = P(1|010) = \frac{4}{10}, \qquad P(x_5|x_1\dots x_4) = P(0|0101) = \frac{4}{4}, \\ P(x_6|x_1\dots x_5) = P(0|01010) = \frac{3}{6}, \qquad P(x_7|x_1\dots x_6) = P(1|010100) = \frac{6}{6}, \\ P(x_8|x_1\dots x_7) = P(1|0101001) = \frac{2}{2}, \end{gathered} \tag{2.12}$$ {eq2.12}

$$\begin{gathered} q(x_1) = q(0) = 0, \qquad q(x_2|x_1) = q(1|0) = \frac{9}{14}, \qquad q(x_3|x_1x_2) = q(0|01) = 0, \\ q(x_4|x_1x_2x_3) = q(1|010) = \frac{6}{10}, \qquad q(x_5|x_1\dots x_4) = q(0|0101) = 0, \\ q(x_6|x_1\dots x_5) = q(0|01010) = 0, \qquad q(x_7|x_1\dots x_6) = q(1|010100) = 0, \\ q(x_8|x_1\dots x_7) = q(1|0101001) = 0. \end{gathered}$$

Accordingly,

$$\begin{gathered} \rho_1^0 = 1, \quad \rho_2^0 = \frac{5}{14}, \quad \rho_3^0 = 1, \quad \rho_4^0 = \frac{2}{5}, \quad \rho_5^0 = 1, \quad \rho_6^0 = \frac{1}{2}, \quad \rho_7^0 = 1, \quad \rho_8^0 = 1, \\ \lambda_1^0 = 0, \quad \lambda_2^0 = \frac{9}{14}, \quad \lambda_3^0 = 0, \quad \lambda_4^0 = \frac{3}{5}, \quad \lambda_5^0 = 0, \quad \lambda_6^0 = 0, \quad \lambda_7^0 = 0, \quad \lambda_8^0 = 0. \end{gathered} \tag{2.13}$$ {eq2.13}

Further, from (2.8), we obtain

$$\begin{gathered} \rho_1^1 = \frac{5}{14}, \quad \rho_2^1 = \frac{2}{5}, \quad \rho_3^1 = \frac{1}{2}, \quad \rho_4^1 = 1, \qquad \lambda_1^1 = \frac{9}{14}, \quad \lambda_2^0 = \frac{3}{5}, \quad \lambda_3^0 = 0, \quad \lambda_4^0 = 0, \\ \rho_1^2 = \frac{1}{7}, \quad \rho_2^1 = \frac{1}{2}, \qquad \lambda_1^2 = \frac{6}{7}, \quad \lambda_2^2 = 0, \qquad \rho_1^3 = \frac{1}{14}, \quad \lambda_1^3 = \frac{6}{7}. \end{gathered} \tag{2.14}$$ {eq2.14}

By (2.9), we have

$$N'(01010011) = \lambda_1^3 \cdot |S_8| = 6/7 \cdot (C_8^4 - C_8^3) = 12.$$

Thus, we have obtained $N'(w')$, the number of the word 01010011 belonging to the set S_8.

The search for the number of a word w'' in the set A_m^n involves expressing an m-adic number in binary form and representing a letter from the alphabet A_m by digits of the m-adic system: α_0 is the digit corresponding to 0 in this system, α_1 is the digit corresponding to 1, etc. To find the binary representation of an m-adic word w'', we use the fast transformation algorithm [4] from the m-adic system to the binary system, based on the principle "divide and rule".

Denote the binary representation of the m-adic word $x_1 \dots x_i$ by $(x_1 \dots x_i)_2$. At the first step, we find the binary representations of the digits comprising the word w'': $(\alpha_0)_2 = 0$, $(\alpha_1)_2 = 1$, $(\alpha_2)_2 = 10$, etc. Knowing the binary representations of two m-adic words of length i, $x_1x_2 \dots x_i$ and $x_{i+1}x_{i+2} \dots x_{2i}$, we can find the binary representation of the m-adic word $x_1x_2 \dots x_{2i}$ of length $2i$ composed of these two words from the formula

$$(x_1x_2 \dots x_{2i})_2 = (x_1x_2 \dots x_i)_2 \cdot m^i + (x_{i+1}x_{i+2} \dots x_{2i})_2.$$

Using this formula, from n binary representations of the letters comprising w'', we obtain the binary representations of $n/2$ subwords of length 2 comprising w''; further, from these subwords, by the same formula, we find the binary representation of $n/4$ subwords of length 4 comprising w'', and continue these computations until we obtain the binary representation of w'', i.e., $N''(w'')$.

Obviously, in our example, there is no need for such a transformation, because the system corresponding to the alphabet A_2, is already binary:

$$N''(\alpha_0\alpha_1\alpha_0\alpha_1) = (0101)_2 = 5.$$

Using (2.1), we obtain the number of the word ( )[ ] ([ ]):

$$N((\,)\,[\,]\,(\,[\,])) = N''(\alpha_0\alpha_1\alpha_0\alpha_1) \cdot 14 + N'(01010011) = 5 \cdot 14 + 12 = 82.$$

The following theorem describes the properties of the proposed method.

{th1}

Theorem 1. *The memory capacity required for encoding a word of length $2n$ of the Dik language over a $2m$-letter alphabet is $O(n \log n)$ bits. The encoding rate of a word of length $2n$ of the Dik language over a $2m$-letter alphabet (i.e., the time needed for encoding one letter) is $O(\log n M(n \log n)/n)$ bit operations, where $M(k)$ is the time needed for the multiplication of two words of length k.*

{cor1}

Corollary 1. *If the Schönhage–Strassen fast multiplication algorithm with*

$$M(k) = O(k \log k \log\log k)$$

is used, then the enumeration rate is $O(\log^3 n \log\log n)$.

{cor2}

Corollary 2. *If the Fürer fast multiplication algorithm with*

$$M(k) = O(k \log k 2^{O(\log^* k)})$$

is used, then the enumeration rate is $O(\log^3 n 2^{O(\log^* n)})$.

The computation time for $N(w)$ consists of the computation time for $N'(w')$, the computation time for $N''(w'')$, and the computation time for $N(w)$ (the latter is obtained from the computed values of $N'(w')$ and $N''(w'')$).

Let us find the computation time for $N'(w')$. It consists of the computation times for ρ_b^a and λ_b^a, where $0 \le a \le \log(2n)$, $1 \le b \le 2n/2^a$, and the computation time for the product $\lambda^{\log(2n)}|S_{2n}|$. To compute ρ_b^0 and λ_b^0, $1 \le b \le 2n$, it is required to compute $2n$ values of $P(x_b|x_1 \dots x_{b-1})$ and $2n$ values of $P(\chi|x_1 \dots x_{b-1})$, where $\chi < x_b$. To compute each of these values from formulas (2.10) or (2.11), we need to perform one multiplication of numbers of length $\log(2n)$ to determine the numerator and one multiplication of numbers of length $\log(2n)$ to determine the denominator.

Denote by $M(n)$ the time needed for the multiplication of two numbers of length n.

For $1 \le b \le 2n$, the computation time for ρ_b^0 and λ_b^0 is $8nM(\log(2n))$.

It follows from (2.10) and (2.11) that, for $1 \le b \le 2n$, the numerators and the denominators of the fractions $P(x_b|x_1 \dots x_{b-1})$, (and, obviously, of the fractions $P(\chi|x_1 \dots x_{b-1})$, $\chi < x_b$) do not exceed $4n^2$. To record the numerator, just as the denominator of these fractions, at most $2\log n + 2$ bits are required. Therefore, it follows from (2.8) that, for $1 \le b \le 2n$, in order to record the fractions λ_b^0 and ρ_b^0, it suffices $2\log n + 2$ bits for the numerator and the same amount for the denominator. It follows from (2.8) that, for $1 \le b \le 2n$, the computation of the quantities ρ_b^1 or λ_b^1 requires, respectively, two or three operations of multiplication of numbers whose length is at most $2\log n + 2$ bits and the total number of operations of multiplication needed to compute all the λ_b^1, ρ_b^1 for $1 \le b \le n$ is $5n$. To compute λ_b^1, it suffices to apply the usual equality $a/b + c/d = (ad + bc)/(bd)$ requiring three multiplications. This yields fractions that require at most $2(2\log n + 2)$ bits for recording the numerator and the same amount for the denominator. Similarly, for $1 \le b \le n/2$, the computation of ρ_b^2 and λ_b^2 requires $5n/2$ operations of multiplication over numbers of length $2 \cdot 2\log n + 4$ bits, and so on; for $1 \le a \le \log(2n)$, $1 \le b \le 2n/2^a$, the computation of ρ_b^a and λ_b^a requires $5n/2^{a-1}$ operations of multiplication over numbers of length $2^{a-1}(2\log n + 2)$ bits.

Therefore, for $1 \le a \le \log(2n)$, $1 \le b \le 2n/2^a$, the total computation time for λ_b^a and ρ_b^a can be expressed as

$$5nM(2\log n + 2) + \dots + \frac{5n}{2^{a-1}} M(2^{a-1}(2\log n + 2)) + \dots + 5M(n(2\log n + 2)). \tag{2.15}$$ {eq2.15}

Denote by $M^*(n)$ the time needed for the multiplication of two numbers of length n divided by the length of these numbers: $M^*(n) = M(n)/n$. Then by (2.15), the total computation time for λ_b^a and ρ_b^a is

$$10n\log nM^*(2\log n) + 10n\log nM^*(4\log n) + \dots + 10n\log nM^*(2n\log n). \tag{2.16}$$ {eq2.16}

In this sum, there are $\log(2n)$ summands and each of them is not greater than $10n\log nM^*(2n\log n)$.

Thus, the computation time for the fractions λ_b^a and ρ_b^a is

$$O((\log n + 1)10n\log nM^*(n2\log n)) = O\left(n\log^2 n\frac{M(n2\log n)}{n(2\log n)}\right) = O(\log nM(n2\log n)). \tag{2.17}$$ {eq2.17}

The computation time for the product $\lambda^{\log(2n)}|S_{2n}|$ consists of the computation time for the product of the numerator of $\lambda^{\log(2n)}$ and $|S_{2n}|$ and the computation time for the division of the resulting number by the denominator of $\lambda^{\log(2n)}$. The number of symbols required for recording the numerator of $\lambda^{\log(2n)}$ does not exceed $2n \cdot 2\log n$. The number of symbols required for recording $|S_{2n}|$, does not exceed $2n$, because the number of binary sequences of length $2n$ corresponding to the correct bracketing of $(|S_{2n}|)$ is less than the number of all binary sequences of length $2n$. Thus, the time needed for the multiplication of the numerator of $\lambda^{\log(2n)}$ and $|S_{2n}|$ is $M(2n\,2\log n)$. The length of the resulting number does not exceed $4n(2\log n)$. The length of the denominator of $\lambda^{\log(2n)}$ does not exceed $2n\,2\log n$. Since the time needed for the division of two numbers of length a is equal to the time needed for the multiplication of two numbers of length a, it follows from [7] that the time needed for the division of the resulting number by the denominator of $\lambda^{\log(2n)}$ is $M(4n\,2\log n)$.

Thus, the computation time for $N'(w')$ is equal to the sum of the computation time for ρ_b^0 and λ_b^0 for $1 \le b \le 2n$, i.e., $8nM(\log(2n))$, of the computation time for λ_b^a and ρ_b^a for

$$1 \le a \le \log(2n), \qquad 1 \le b \le \frac{2n}{2^a},$$

i.e., $O(\log nM(n\,2\log n))$, and of the computation time for $\lambda^{\log(2n)}|S_{2n}|$, i.e.,

$$M(2n\,2\log n) + M(4n\,2\log n),$$

giving

$$\begin{aligned} &8nM(\log(2n)) + O(\log nM(n\,2\log n)) + M(2n\,2\log n) + M(4n\,2\log n) \\ &\quad = 8n\log(2n)M^*(\log(2n)) + O(\log n\,n(2\log n)M^*(n\,2\log n)) + O(M(4n\,2\log n) \end{aligned}$$

$$= O(8n\log(2n)M^*(\log(2n)) + \log n\, n(2\log n)M^*(n\,2\log n) + M(4n\,2\log n)$$
$$= O(n\log n(2\log n)M^*(4n\,2\log n)) = O(\log nM(n\log n)). \tag{2.18}$$ {eq2.18}

Let us now determine the computation time for $N''(w'')$. The computation of each of the elements $(x_1x_2)_2,\dots,(x_{n-1}x_n)_2$ requires one multiplication of the numbers m and x_i for i equal to $1,3,\dots,n-1$ and one addition of the resulting products and the numbers x_{i+1}. The number of symbols required for recording the number m, just as for recording x_i, does not exceed $\log m$. Therefore, the time needed for their multiplication is $M(\log m) = O(1)$. The length of the resulting product does not exceed $2\log m$. The length x_{i+1} does not exceed $\log m$. Therefore, the time needed for their addition does not exceed $2\log m = O(1)$. Thus, the computation of all elements $(x_1x_2)_2,\dots,(x_{n-1}x_n)_2$ requires n operations taking the time $O(1)$. The total time is $O(n)$. Similarly, the computation of $(x_1x_2x_3x_4)_2$, …, $(x_{n-3}x_{n-2}x_{n-1}x_n)_2$ requires $n/4$ operations taking the time $M(2\log m) = O(1)$, and $n/4$ operations taking the time $4\log m = O(1)$. The total time is $O(n)$, etc.; the computation $(x_1\dots x_{2^k})_2,\dots,(x_{n-2^k-1}\dots x_n)_2$ for $1 \le k \le \log n$ requires $n/2^k$ operations taking the time $M(2^{k-1}\log m$, and $n/2^k$ operations, taking the time $2^k\log m$. The total time is

$$\frac{n}{2^k}\cdot M(2^{k-1}\log m) + 2^k\log m.$$

The total computation time for $N''(w'')$ is

$$\frac{n}{2}(M(\log m) + 2\log m) + \frac{n}{4}(M(2\log m) + 4\log m) + \dots + \left(M\left(\frac{n}{2}\log m\right) + n\log m\right)$$
$$= \frac{n}{2}\cdot(\log mM^*(\log m) + 2\log m) + \frac{n}{4}\cdot(2\log mM^*(2\log m) + 4\log m) + \dots$$
$$\dots + \left(\frac{n}{2}\log mM^*\left(\frac{n}{2}\log m\right) + n\log m\right). \tag{2.19}$$ {eq2.19}

This expression contains $\log n$ summands, each of which does not exceed

$$\frac{n}{2}\log mM^*\left(\frac{n}{2}\log m\right) + n\log m;$$

thus, the time needed to compute $N''(w'')$, is

$$\log n\left(\frac{n}{2}\log mM^*\left(\frac{n}{2}\log m\right) + n\log m\right) = O(\log nM(n)). \tag{2.20}$$ {eq2.20}

Let us determine the time needed to compute $N(w)$ from $N'(w')$ and $N''(w'')$ from (2.1). The number of symbols required for recording $N''(w'')$ is at most $n\log m$. The number of symbols required for recording the number $|S_{2n}|$, is at most $2n$. The multiplication of these numbers requires the time $M(n\log m) = O(M(n))$. The number of symbols required for recording the resulting number is at most $n\log m$. The number of symbols required for recording $N'(w')$ is at most $2n$. The addition of the resulting number and $N'(w')$ requires the time $n\log m = O(n)$. Thus, the total time needed to compute $N(w)$ from $N'(w')$ and $N''(w'')$, is

$$O(M(n)) + O(n) = O(M(n)). \tag{2.21}$$ {eq2.21}

It follows from (2.18), (2.20), (2.21) that the total time needed for determining the number w is

$$O(\log nM(n\log n)) + O(\log nM(n)) + O(M(n)) = O(\log nM(n\log n)).$$

The complexity of the computation of the number of the word w per one symbol is $O(\log n/nM(n\log n))$.

Let us estimate the memory capacity needed for the enumeration. To determine $N'(w')$ in the process of the computation of λ_b^a and ρ_b^a for $1 \le a \le \log(2n)$, $1 \le b \le 2n/2^a$, we use only the quantities λ_b^{a-1} and ρ_b^{a-1}, $1 < a \le \log n$, $1 \le b \le 2n/2^a$. Therefore, for the enumeration, it suffices to have memory capacity for storing the collections of λ_b^a, ρ_b^a, $1 \le a \le \log(2n)$, $1 \le b \le 2n/2^a$, and λ_b^{a+1}, ρ_b^{a+1}, $1 < a \le \log n$,

$1 \le b \le 2n/2^a$. The length of each fraction λ_b^a and ρ_b^a is at most $2^{a+1}(2\log n + 2)$. Hence the memory capacity needed to determine the number of the word w' is at most $O(n \log n)$.

To determine $N''(w'')$ in computing

$$(x_1 \dots x_{2^k})_2, \dots, (x_{n-2^k+1} \dots x_n)_2, \qquad 1 \le k \le \log n,$$

only the quantities $(x_1 \dots x_{2^{k-1}})_2, \dots, (x_{n-2^{k-1}+1} \dots x_n)_2$ are used. Therefore, for the enumeration, it suffices to have memory capacity for storing the collections

$$(x_1 \dots x_{2^k})_2, \dots, (x_{n-2^k+1} \dots x_n)_2, \quad (x_1 \dots x_{2^{i-1}})_2, \dots, (x_{n-2^{k-1}+1} \dots x_n)_2, \qquad 1 \le k \le \log n.$$

The length of each number

$$(x_1 \dots x_{2^k})_2, \dots, (x_{n-2^k+1} \dots x_n)_2$$

is $2^k \log m$; there are $n/2^k$ such numbers in all. The length of each number

$$(x_1 \dots x_{2^{k-1}})_2, \dots, (x_{n-2^{k-1}+1} \dots x_n)_2$$

is $2^{k-1} \log m$; there are $n/2^{k-1}$ such numbers in all. Hence the memory capacity required to compute $N''(w'')$ is at most $O(n \log m) = O(n)$.

The total memory capacity needed to compute $N(w)$ is $O(n \log n) + O(n) = O(n \log n)$. The theorem is proved. □

3. THE DENUMERATION ALGORITHM {ssec3}

Let us describe the decoding algorithm. As an example, we consider the search for a word from the set D_8^4, provided its number $N = 82$ is known.

From the number N, we find $N'(w')$ and $N''(w'')$:

$$N'(w') = N \bmod |S|, \qquad N''(w'') = \lfloor N/|S| \rfloor.$$

In our example, $N'(w') = 82 \bmod 14 = 12$ and $N''(w'') = \lfloor 82/14 \rfloor = 5$.

Given $N'(w')$, let us find the word w' from the set S_{2n}.

To describe the algorithm, we introduce auxiliary functions for the upper and lower bounds for λ_b^a. Let p/q be a rational number expressed as the pair of positive integers p, q, $p \le q$, and let $t > 1$ be an integer. Set $l = \lfloor \log q \rfloor$. Let $(q_l q_{l-1} \dots q_0)$ and $(p_l p_{l-1} \dots p_0)$ be the binary representations of the numbers q and p. Then we define $\phi_t^+(p/q)$ and $\phi_t^-(p/q)$ as follows:

$$\phi_t^+\left(\frac{p}{q}\right) = \frac{\sum_{i=l-t}^{l} p_i 2^i + 2^{l-t}}{\sum_{i=l-t}^{l} q_i 2^i}, \qquad \phi_t^-\left(\frac{p}{q}\right) = \frac{\sum_{i=l-t}^{l} p_i 2^i}{\sum_{i=l-t}^{l} q_i 2^i + 2^{l-t}}. \tag{3.1}$$ {eq3.1}

If $l - t < 0$, then we multiply the numerator and the denominator of the resulting fraction by $2^{-(l-t)}$.

For example,

$$\phi_3^+\left(\frac{9}{17}\right) = \frac{5}{8}, \qquad \phi_3^-\left(\frac{9}{17}\right) = \frac{4}{9}.$$

Let $q_{\max}$ be equal to the maximal denominator of the numbers

$$\frac{N_{S_{2n}}(x_1 \dots x_{i+1})}{N_{S_{2n}}(x_1 \dots x_i)}, \qquad x_1 \dots x_{2n} \in S_{2n}, \quad 1 \le i \le 2n-1.$$

It follows from (2.10) and (2.11) that

$$q_{\max} = 4n^2. \tag{3.2}$$ {eq3.2}

From (2.8), we find that the denominators of the rational fractions λ_b^a and ρ_b^a do not exceed $q_{\max}^{2^a}$ for all $1 \le b \le 2n/2^a$ and, therefore,

$$\rho_b^a \ge \frac{1}{q_{\max}^{2^a}}. \tag{3.3}$$ {eq3.3}

Given the number $N'(w')$, the first step in the search for the word w' from the set S_{2n} consists in computing the estimates $\lambda^+(\log(2n),1)$, $\lambda^-(\log(2n),1)$ using the formulas

$$\begin{aligned}\lambda^+(\log(2n),1)&=\phi^+_{8n\lceil\log q_{\max}\rceil+4}\left(\frac{N'(w')}{|S_{2n}|}\right),\\ \lambda^-(\log(2n),1)&=\phi^-_{8n\lceil\log q_{\max}\rceil+4}\left(\frac{N'(w')}{|S_{2n}|}\right).\end{aligned}\tag{3.4}$$

{eq3.4}

Given the prefix $x_1\dots x_{2^a(b-1)}$ and estimates $\lambda^+(a,b)$, $\lambda^-(a,b)$ $(0<a\le\log(2n),\ 1\le b\le 2n/2^a)$, let us describe a recursive procedure for finding the subword $x_{2^a(b-1)+1}\dots x_{2^ab}$ and the quantities λ^a_b, ρ^a_b or a pair of words such that one of them is $x_{2^a(b-1)+1}\dots x_{2^ab}$, and the corresponding pair of the conjectured values of λ^a_b and ρ^a_b. Further, if $b=2n/2^a$, then by this procedure, we can uniquely determine the subword $x_{2n-2^a+1}\dots x_{2n}$. (In this case, we do not need to find λ^a_b, ρ^a_b, because, given the prefix $x_1\dots x_{2n-2^a}$ and the subword $x_{2n-2^a+1}\dots x_{2n}$, we can find the unknown word $x_1\dots x_{2n}$.)

Let us compute the estimates $\lambda^+(a-1,2b-1)$, $\lambda^-(a-1,2b-1)$ using the formulas

$$\begin{aligned}\lambda^+(a-1,2b-1)&=\phi^+_{2^{a+1}\lceil\log q_{\max}\rceil+4}(\lambda^+(a,b)),\\ \lambda^-(a-1,2b-1)&=\phi^-_{2^{a+1}\lceil\log q_{\max}\rceil+4}(\lambda^-(a,b)).\end{aligned}\tag{3.5}$$

{eq3.5}

If $a-1>0$, then, given the prefix $x_1\dots x_{2^{a-1}(2b-2)}$ and these estimates, we carry out the recursive procedure for finding the subword $x_{2^{a-1}(2b-2)+1}\dots x_{2^{a-1}(2b-1)}$ and the quantities λ^{a-1}_{2b-1}, ρ^{a-1}_{2b-1} or the pair of words, one of which is $x_{2^{a-1}(2b-2)+1}\dots x_{2^{a-1}(2b-1)}$, and the corresponding pair of the conjectured values of λ^{a-1}_{2b-1}, ρ^{a-1}_{2b-1}. Denote the lexicographically smaller word from the pair of words by $x'_{2^{a-1}(2b-2)+1}\dots x'_{2^{a-1}(2b-1)}$, the greater word by $x''_{2^{a-1}2(b-2)+1}\dots x''_{2^{a-1}(2b-1)}$, and the corresponding conjectured values of λ^{a-1}_{2b-1}, ρ^{a-1}_{2b-1} by λ'^{a-1}_{2b-1}, λ''^{a-1}_{2b-1}, ρ'^{a-1}_{2b-1}, and ρ''^{a-1}_{2b-1}.

If $a-1=0$, then we obtain letters χ, $\chi\in\{0,1\}$, for which the following conditions simultaneously hold:

$$\begin{aligned}P(\chi|x_1\dots x_{2b-2})&>0,\\ \lambda^+(0,2b-1)&\ge q(\chi|x_1\dots x_{2b-2}),\\ \lambda^-(0,2b-1)&<q(\chi|x_1\dots x_{2b-2})+P(\chi|x_1\dots x_{2b-2}).\end{aligned}\tag{3.6}$$

{eq3.6}

If there exists one such χ, then $x_{2b-1}=\chi$. If there exists two such χ's, we assume that x'_{2b-1} is equal to the smallest of the two, i.e., zero, while x''_{2b-1} is taken as the greatest of the two such χ's, i.e., as 1. In the first case, we compute the values of λ^0_{2b-1} and ρ^0_{2b-1} from formulas (2.8), while, in the second case, we compute the values of λ'^0_{2b-1}, ρ'^0_{2b-1}, λ''^0_{2b-1}, and ρ''^0_{2b-1} using the following formulas:

$$\begin{aligned}\lambda'^0_{2b-1}&=q(x'_{2b-1}|x_1\dots x_{2b-2}),&\qquad\lambda''^0_{2b-1}&=q(x''_{2b-1}|x_1\dots x_{2b-2}),\\ \rho'^0_{2b-1}&=P(x'_{2b-1}|x_1\dots x_{2b-2}),&\qquad\rho''^0_{2b-1}&=P(x''_{2b-1}|x_1\dots x_{2b-2}).\end{aligned}\tag{3.7}$$

{eq3.7}

Thus, at this stage of the procedure, we obtain the subword $x_1\dots x_{2^{a-1}(2b-2)}$ and the values of λ^{a-1}_{2b-1} and ρ^{a-1}_{2b-1} or the words $x'_1\dots x'_{2^{a-1}(2b-2)}$, $x''_1\dots x''_{2^{a-1}(2b-2)}$ and the values of λ'^{a-1}_{2b-1}, ρ'^{a-1}_{2b-1}, λ''^{a-1}_{2b-1}, and ρ''^{a-1}_{2b-1}. In the second case, we verify whether the following inequalities hold:

$$\begin{aligned}\lambda'^{a-1}_{2b-1}+\rho'^{a-1}_{2b-1}&\le\lambda^-(a,b),\\ \lambda''^{a-1}_{2b-1}&>\lambda^+(a,b).\end{aligned}\tag{3.8}$$

{eq3.8}

If the first inequality holds, then

$$x_1\dots x_{2^{a-1}(2b-2)}=x''_1\dots x''_{2^{a-1}(2b-2)},\qquad\lambda^{a-1}_{2b-1}=\lambda''^{a-1}_{2b-1},\qquad\rho^{a-1}_{2b-1}=\rho''^{a-1}_{2b-1}.$$

If the second inequality holds, then

$$x_1\dots x_{2^{a-1}(2b-2)}=x'_1\dots x'_{2^{a-1}(2b-2)},\qquad \lambda^{a-1}_{2b-1}=\lambda'^{a-1}_{2b-1},\qquad \rho^{a-1}_{2b-1}=\rho'^{a-1}_{2b-1}.$$

If, at this stage of computations, the exact values of λ^{a-1}_{2b-1} and ρ^{a-1}_{2b-1} are known (from the use of the comparisons (3.8) or without them), we compute the estimates $\lambda^+(a-1,2b)$, $\lambda^-(a-1,2b)$ using the formulas

$$\begin{aligned}\lambda^+(a-1,2b)&=\phi^+_{2^{a+1}\lceil\log q_{\max}\rceil+4}\biggl(\frac{\lambda^+(a,b)-\lambda^{a-1}_{2b-1}}{\rho^{a-1}_{2b-1}}\biggr),\\ \lambda^-(a-1,2b)&=\phi^-_{2^{a+1}\lceil\log q_{\max}\rceil+4}\biggl(\frac{\lambda^-(a,b)-\lambda^{a-1}_{2b-1}}{\rho^{a-1}_{2b-1}}\biggr),\end{aligned}\tag{3.9}$$

{eq3.9}

Let us apply to them the recursive procedure if $a-1>0$, obtaining the subword $x_{2^{a-1}(2b-1)+1}\dots x_{2^{a-1}2b}$ and the values of λ^{a-1}_{2b}, ρ^{a-1}_{2b} (or, in the case $b=2n/2^a$, only the subword $x_{2^{a-1}(2b-1)+1}\dots x_{2^{a-1}2b}$) or a pair of subwords such that one word from this pair is $x_{2^{a-1}(2b-1)+1}\dots x_{2^{a-1}2b}$, and the corresponding pairs of the conjectured values of λ^{a-1}_{2b}, ρ^{a-1}_{2b}. But if $a-1=0$, then the resulting estimates are subjected to actions similar to those performed with the estimates λ^0_{2b-1}, ρ^0_{2b-1}; as a result, we obtain x_{2b}, λ^0_{2b}, ρ^0_{2b} or the pairs of letters x'_{2b} and x''_{2b}, $(x'_{2b}<x''_{2b})$ and the pairs of values of λ'^0_{2b}, ρ'^0_{2b}, λ''^0_{2b}, ρ''^0_{2b}. But if $b=2n/2^a$, i.e., $a=0$, $b=2n$ and we have found the letters x'_{2n} and x''_{2n}, then this implies that $x_{2n}=x''_{2n}$.

If neither of the inequalities (3.8) holds, then, for $b=2n/2^a$, this means that $x_{2^a(b-1)+1}\dots x_{2^{a-1}}$ has been obtained and is equal to $x''_{2^a(b-1)+1}\dots x''_{2^{a-1}}$. We then successively find the letters

$$x_i,\qquad 2^{a-1}(2b-1)+1\le i\le 2^{a-1}2b,$$

such that

$$\begin{aligned}q(x_i|x_1\dots x_{i-1})&=0,\\ P(x_i|x_1\dots x_{i-1})&>0.\end{aligned}\tag{3.10}$$

{eq3.10}

If neither of the inequalities (3.8) holds and $b\ne 2n/2^a$, then we successively obtain the letters

$$x'_i,\qquad 2^{a-1}(2b-1)+1\le i\le 2^{a-1}2b$$

such that

$$\begin{aligned}q(x'_i|x_1\dots x_{2^a(b-1)}x'_{2^a(b-1)+1}\dots x'_{i-1})+P(x'_i|x_1\dots x_{2^a(b-1)}x'_{2^a(b-1)+1}\dots x'_{i-1})&=1,\\ P(x'_i|x_1\dots x_{2^a(b-1)}x'_{2^a(b-1)+1}\dots x'_{i-1})&>0,\end{aligned}\tag{3.11}$$

{eq3.11}

and successively obtain the letters

$$x''_i,\qquad 2^{a-1}(2b-1)+1\le i\le 2^{a-1}2b$$

such that

$$\begin{aligned}q(x''_i|x_1\dots x_{2^a(b-1)}x''_{2^a(b-1)+1}\dots x''_{i-1})&=0,\\ P(x''_i|x_1\dots x_{2^a(b-1)}x''_{2^a(b-1)+1}\dots x''_{i-1})&>0.\end{aligned}\tag{3.12}$$

{eq3.12}

Using formulas similar to (2.8), we recursively obtain the pair λ'^{a-1}_{2b}, ρ'^{a-1}_{2b} (in the formulas, the prefix $x_1\dots x_{2^a(b-1)}x'_{2^a(b-1)+1}\dots x'_{2^ab}$ is used) and the pair of numbers λ''^{a-1}_{2b}, ρ''^{a-1}_{2b} (in the formulas, the prefix $x_1\dots x_{2^a(b-1)}x''_{2^a(b-1)+1}\dots x''_{2^ab}$ is used).

Thus, at this stage of the procedure, we have several possible cases.

In the first case, we know the subwords

$$x_{2^{a-1}(2b-2)+1}\dots x_{2^{a-1}(2b-1)},\qquad x_{2^{a-1}(2b-1)+1}\dots x_{2^{a-1}2b},$$

which means that the subword $x_{2^a(b-1)+1}\dots x_{2^ab}$ is known. If $b\neq 2n/2^a$, then we compute the values of λ_b^a and ρ_b^a from the formulas

$$\lambda_b^a=\lambda_{2b-1}^{a-1}+\rho_{2b-1}^{a-1}\cdot\lambda_{2b}^{a-1},\qquad \rho_b^a=\rho_{2b-1}^{a-1}\cdot\rho_{2b}^{a-1}. \tag{3.13}$$ {eq3.13}

In the second case, we know the subword $x_{2^{a-1}(2b-2)+1}\dots x_{2^{a-1}(2b-1)}$ and the pair of words

$$x'_{2^{a-1}(2b-1)+1}\dots x'_{2^{a-1}2b},\qquad x''_{2^{a-1}(2b-1)+1}\dots x''_{2^{a-1}2b}.$$

Then the word $x'_{2^a(b-1)+1}\dots x'_{2^ab}$ is assumed equal to the concatenation of

$$x_{2^{a-1}(2b-2)+1}\dots x_{2^{a-1}(2b-1)}\qquad\text{and}\qquad x'_{2^{a-1}(2b-1)+1}\dots x'_{2^{a-1}2b};$$

the word $x''_{2^a(b-1)+1}\dots x''_{2^ab}$ is assumed equal to the concatenation of

$$x_{2^{a-1}(2b-2)+1}\dots x_{2^{a-1}(2b-1)}\qquad\text{and}\qquad x''_{2^{a-1}(2b-1)+1}\dots x''_{2^{a-1}2b}.$$

Let us compute the values of $\lambda_b'^a$, $\rho_b'^a$, $\lambda_b''^a$, $\rho_b''^a$ from the formulas

$$\begin{aligned}\lambda_b'^a&=\lambda_{2b-1}^{a-1}+\rho_{2b-1}^{a-1}\cdot\lambda_{2b}'^{a-1},\qquad \rho_b'^a=\rho_{2b-1}^{a-1}\cdot\rho_{2b}'^{a-1},\\ \lambda_b''^a&=\lambda_{2b-1}^{a-1}+\rho_{2b-1}^{a-1}\cdot\lambda_{2b}''^{a-1},\qquad \rho_b''^a=\rho_{2b-1}^{a-1}\cdot\rho_{2b}''^{a-1}.\end{aligned} \tag{3.14}$$ {eq3.14}

In the third case, we know the pairs of the following words:

$$\begin{gathered}x'_{2^{a-1}(2b-2)+1}\dots x'_{2^{a-1}(2b-1)},\qquad x''_{2^{a-1}(2b-2)+1}\dots x''_{2^{a-1}(2b-1)},\\ x'_{2^{a-1}(2b-1)+1}\dots x'_{2^{a-1}2b},\qquad x''_{2^{a-1}(2b-1)+1}\dots x''_{2^{a-1}2b}.\end{gathered}$$

The word $x'_{2^a(b-1)+1}\dots x'_{2^ab}$ is assumed equal to the concatenation of

$$x'_{2^{a-1}(2b-2)+1}\dots x'_{2^{a-1}(2b-1)}\quad\text{and}\quad x'_{2^{a-1}(2b-1)+1}\dots x'_{2^{a-1}2b},$$

and the word $x''_{2^a(b-1)+1}\dots x''_{2^ab}$ is assumed equal to the concatenation of

$$x''_{2^{a-1}(2b-2)+1}\dots x''_{2^{a-1}(2b-1)}\quad\text{and}\quad x''_{2^{a-1}(2b-1)+1}\dots x''_{2^{a-1}2b}.$$

Let us compute the values of $\lambda_b'^a$, $\rho_b'^a$, $\lambda_b''^a$, $\rho_b''^a$ from the formulas

$$\begin{aligned}\lambda_b'^a&=\lambda_{2b-1}'^{a-1}+\rho_{2b-1}'^{a-1}\cdot\lambda_{2b}'^{a-1},\qquad \rho_b'^a=\rho_{2b-1}'^{a-1}\cdot\rho_{2b}'^{a-1},\\ \lambda_b''^a&=\lambda_{2b-1}''^{a-1}+\rho_{2b-1}''^{a-1}\cdot\lambda_{2b}''^{a-1},\qquad \rho_b''^a=\rho_{2b-1}''^{a-1}\cdot\rho_{2b}''^{a-1}.\end{aligned} \tag{3.15}$$ {eq3.15}

This concludes the description of the recursive procedure.

Applying the procedure described above for $a=\log(2n)$, $b=1$ and the estimates $\lambda^+(\log(2n),1)$, $\lambda^-(\log(2n),1)$, we obtain the word $x_1\dots x_{2n}$. (Here we assume that the prefix of zero length is known.)

Consider an example. Given the number $N'(w')=12$, let us find the word w' of length $n=8$. We obtain $q_{\max}=4n^2=64$. We find $\lambda^+(3,1)$, $\lambda^-(3,1)$ using formulas (3.4):

$$\lambda^+(3,1)=\phi_{196}^+\frac{12}{14}=\frac{12\cdot 2^{193}+1}{14\cdot 2^{193}},\qquad \lambda^-(3,1)=\frac{12\cdot 2^{193}}{14\cdot 2^{193}+1}.$$

We obtain $\lambda^+(2,1)$, $\lambda^-(2,1)$, and then $\lambda^+(1,1)$, $\lambda^-(1,1)$, $\lambda^+(0,1)$, $\lambda^-(0,1)$ using formulas (3.5):

$$\begin{aligned}\lambda^+(2,1)&=\phi_{100}^+\frac{12\cdot 2^{193}+1}{14\cdot 2^{193}}=\frac{12\cdot 2^{97}+1}{14\cdot 2^{97}},\qquad \lambda^-(2,1)=\frac{12\cdot 2^{97}}{14\cdot 2^{97}+1},\\ \lambda^+(1,1)&=\phi_{52}^+\frac{12\cdot 2^{97}+1}{14\cdot 2^{97}}=\frac{12\cdot 2^{49}+1}{14\cdot 2^{49}},\qquad \lambda^-(1,1)=\frac{12\cdot 2^{49}}{14\cdot 2^{49}+1},\\ \lambda^+(0,1)&=\phi_{28}^+\frac{12\cdot 2^{49}+1}{14\cdot 2^{49}}=\frac{12\cdot 2^{25}+1}{14\cdot 2^{25}},\qquad \lambda^-(0,1)=\frac{12\cdot 2^{25}}{14\cdot 2^{25}+1}.\end{aligned}$$

We obtain χ's for which inequalities (3.6) hold, i.e.,

$$\lambda^+(0,1) \geq q(\chi), \quad \lambda^-(0,1) < q(\chi) + P(\chi), \qquad P(\chi) > 0.$$

The unique χ satisfying these inequalities is 0; we can see that

$$\lambda^+(0,1) > q(0) = 0, \quad \lambda^-(0,1) < q(0) + P(0) = 1, \qquad P(0) = 1 > 0,$$

at the same time, for 1, these inequalities do not hold, because $P(1) = 0$. Therefore, $x_1 = 0$.

We can find λ_1^0 and ρ_1^0 from formulas (2.8):

$$\lambda_1^0 = q(0) = 0, \qquad \rho_1^0 = P(0) = 1.$$

We obtain $\lambda^+(0,2)$, $\lambda^-(0,2)$ from formulas (3.9):

$$\lambda^+(0,2) = \phi_{28}^+ \frac{(12 \cdot 2^{49} + 1)/(14 \cdot 2^{49}) - 0}{1} = \frac{12 \cdot 2^{25} + 1}{14 \cdot 2^{25}}, \qquad \lambda^-(0,2) = \frac{12 \cdot 2^{25}}{14 \cdot 2^{25} + 1}.$$

We obtain χ's for which inequalities (3.6) hold, i.e.,

$$\lambda^+(0,2) \geq q(\chi|0), \quad \lambda^-(0,2) < q(\chi|0) + P(\chi|0), \qquad P(\chi|0) > 0.$$

These inequalities hold for $\chi = 1$, because

$$\lambda^+(0,2) > q(1|0) = \frac{5}{14}, \quad \lambda^-(0,2) < q(1|0) + P(1|0) = 1, \qquad P(1|0) = \frac{5}{14} > 0,$$

while the inequalities do not hold for $\chi = 0$, because

$$\lambda^-(0,2) > q(0|0) + P(0|0) = \frac{9}{14}.$$

Therefore, $x_2 = 1$.

We obtain λ_2^0 and ρ_2^0, and then λ_1^1 and ρ_1^1 by using formulas (2.8):

$$\lambda_2^0 = q(1|0) = \frac{9}{14}, \quad \rho_2^0 = P(1|0) = \frac{5}{14}, \qquad \lambda_1^1 = \lambda_1^0 + \lambda_2^0 \cdot \rho_1^0 = \frac{9}{14}, \quad \rho_1^1 = \rho_1^0 \cdot \rho_2^0 = \frac{5}{14}.$$

We obtain $\lambda^+(1,2)$, $\lambda^-(1,2)$ from formulas (3.9):

$$\lambda^+(1,2) = \frac{3 \cdot 2^{50} + 1}{5 \cdot 2^{50}}, \qquad \lambda^-(1,2) = \frac{3 \cdot 2^{50}}{5 \cdot 2^{50} + 1}.$$

We obtain $\lambda^+(0,3)$, $\lambda^-(0,3)$, from formulas (3.5):

$$\lambda^+(0,3) = \frac{3 \cdot 2^{26} + 1}{5 \cdot 2^{26}}, \qquad \lambda^-(0,3) = \frac{3 \cdot 2^{26}}{5 \cdot 2^{26} + 1}.$$

We obtain χ's for which inequalities (3.6) hold, i.e.,

$$\lambda^+(0,3) \geq q(\chi|01), \quad \lambda^-(0,3) < q(\chi|01) + P(\chi|01), \qquad P(\chi|01) > 0.$$

The inequalities hold for $\chi = 0$, because

$$\lambda^+(0,3) > q(0|01) = 0, \quad \lambda^-(0,3) < q(0|01) + P(0|01) = 1, \qquad P(0|01) = 1 > 0;$$

the inequalities do not hold for $\chi = 1$, because $P(1|01) = 0$. Therefore, $x_3 = 0$.

We obtain λ_3^0 and ρ_3^0 from formulas (2.8):

$$\lambda_3^0 = q(0|01) = 0, \qquad \rho_3^0 = P(0|01) = 1.$$

We obtain $\lambda^+(0,4)$, $\lambda^-(0,4)$ from formulas (3.9):

$$\lambda^+(0,4) = \frac{3 \cdot 2^{26} + 1}{5 \cdot 2^{26}}, \lambda^-(0,4) = \frac{3 \cdot 2^{26}}{5 \cdot 2^{26} + 1}.$$

We obtain χ's for which inequalities (3.6) hold, i.e.,

$$\lambda^+(0,4) \geq q(\chi|010), \quad \lambda^-(0,4) < q(\chi|010) + P(\chi|010), \qquad P(\chi|010) > 0.$$

The inequalities hold for $\chi = 0$, because

$$\lambda^+(0,4) > q(0|010) = 0, \quad \lambda^-(0,4) < q(0|010) + P(0|010) = \frac{3}{5}, \qquad P(0|010) = \frac{3}{5} > 0;$$

the inequalities also hold for $\chi = 1$, because

$$\lambda^+(0,4) > q(1|010) = \frac{3}{5}, \quad \lambda^-(0,4) < q(1|010) + P(1|010) = 1, \qquad P(1|010) = \frac{2}{5} > 0.$$

Thus, $x'_4 = 0$, $x''_4 = 1$. We obtain λ'^0_4, ρ'^0_4, λ''^0_4, ρ''^0_4 from formulas (3.7):

$$\lambda'^0_4 = q(0|010) = 0, \quad \rho'^0_4 = P(0|010) = \frac{3}{5}, \qquad \lambda''^0_4 = q(1|010) = \frac{3}{5}, \quad \rho'^0_4 = P(1|010) = \frac{2}{5}.$$

We obtain λ'^1_2, ρ'^1_2, λ''^1_2, ρ''^1_2, and then λ'^2_1, ρ'^2_1, λ''^2_1, ρ''^2_1 from formulas (3.14):

$$\lambda'^1_2 = 0, \quad \rho'^1_2 = \frac{3}{5}, \qquad \lambda''^1_2 = \frac{3}{5}, \quad \rho''^1_2 = \frac{2}{5},$$
$$\lambda'^2_1 = \frac{9}{14}, \quad \rho'^2_1 = \frac{3}{14}, \qquad \lambda''^2_1 = \frac{12}{14}, \quad \rho''^2_1 = \frac{2}{14}.$$

Let us use the comparisons (3.8). We see that the following two inequalities simultaneously hold:

$$\lambda'^2_1 + \rho'^2_1 > \lambda^-(3,1), \qquad \lambda''^2_1 \leq \lambda^+(3,1).$$

Further, the equality $1 = (2n)/2^3$ holds, i.e., $b = 2n/2^a$. Therefore, $x_4 = x''_4 = 1$, and we successively find the letters $x_5, \ldots, x_8$. Then we find a letter x_5 such that the conditions

$$q(x_5|0101) = 0, \qquad P(x_5|0101) > 0$$

hold. These conditions hold for $x_5 = 0$. We find an x_6 such that the conditions

$$q(x_6|01010) = 0, \qquad P(x_6|01010) > 0$$

hold. These conditions hold for $x_6 = 0$. We find an x_7 such that the conditions

$$q(x_7|010100) = 0, \qquad P(x_7|010100) > 0$$

hold. These conditions hold for $x_7 = 1$. We find a letter x_8 such that the conditions

$$q(x_8|0101001) = 0, \qquad P(x_8|0101001) > 0$$

hold. These conditions hold for $x_8 = 1$. Thus, the unknown word is 01010011.

Now let us transform the number $N''(w'')$ of the word w'' from the binary system to the m-adic system with the symbols of the alphabet A_m as digits of the system. The algorithm is based on the idea that, given the binary representation of a number in an m-adic system, we can find the binary representations for its halves:

$$N(x_1x_2\ldots x_{n/2}) = \left\lfloor \frac{N(x_1\ldots x_n)}{(m^{n/2})} \right\rfloor, \qquad N(x_{n/2+1}\ldots x_n) = N(x_1\ldots x_n) \mod m^{n/2},$$
$$N(x_1x_2\ldots x_{n/4}) = \left\lfloor \frac{N(x_1\ldots x_{n/2})}{(m^{n/4})} \right\rfloor, \qquad N(x_{n/4+1}\ldots x_{n/2}) = N(x_1\ldots x_{n/2}) \mod m^{n/4},$$
$$N(x_{n/2+1}\ldots x_{3n/4}) = \left\lfloor \frac{N(x_{n/2+1}\ldots x_n)}{(m^{n/4})} \right\rfloor, \quad N(x_{3n/4+1}\ldots x_n) = N(x_{n/2+1}\ldots x_n) \mod m^{n/4}, \tag{3.16}$$

{eq3.16}

etc. This action is repeated $\log n$ times; as a result, we obtain all the symbols of the number $w'' = (x_1x_2\ldots x_n)_m$, which are replaced by the corresponding letters of the alphabet A_m. If the result obtained has less than n symbols, the symbols corresponding to 0's are put on the left. This results in

the word w''. Since, in our example, $m = 2$, it follows that there is no need for such a transformation. We obtain

$$N''(w'') = (101)_2, \qquad w'' = \alpha_0\alpha_1\alpha_0\alpha_1. \tag{3.17}$$ {eq3.17}

Now the 0's in the word w' are replaced by opening brackets of m types in the order given by the word w''. The 1's are replaced by closing brackets of m types so that the resulting word represents a correct bracketing of m types. This method is uniquely determined by the types of opening brackets. In our case, we obtain ( ) [ ] ([ ]). The denumerated word w is obtained.

The properties of the denumeration algorithm are described by the following theorem.

{th2}

Theorem 2. *The memory capacity required for the denumeration of a word of length $2n$ is $O(n\log^2 n)$.*

The denumeration complexity, i.e., the time needed to find one letter of the denumerated word is equal to $O(\log nM(n\log n)/n)$, where $M(n\log n)$ is the time needed for the multiplication or the division of two words of length $n\log n$.

{cor3}

Corollary 3. *If the Schönhage–Strassen fast multiplication algorithm with complexity $M(n) = n\log n\log\log n$ is used, then the denumeration rate is $O(\log^3 n\log\log n)$.*

{cor4}

Corollary 4. *If the Fürer fast multiplication algorithm with complexity $M(n) = n\log n2^{O(\log^* n)}$ is used, then the denumeration rate is*

$$O(\log^3 n2^{O(\log^* n)}).$$

In the computation of the word w' from its number, the left half $x_1, x_2, \dots, x_n$ and the right half $x_{n+1}, \dots, x_{2n}$ are decoded independently and the same memory can be used. The same applies to the computation of the word w'' from its number. It is easy to see that the memory capacity used for the encoding and decoding of the word w is asymptotically the same.

The total memory capacity needed to compute the word w from its number is $O(n\log n)$.

The time needed to determine the word w from its number $N(w)$ is comprised of the time needed for determining $N'(w')$ and $N''(w'')$ from $N(w)$, of the time needed for determining the word w' from $N'(w')$, and of the time needed for determining the word w'' from $N''(w'')$.

Let us estimate the time needed to compute $N'(w')$ and $N''(w'')$ from $N(w)$. To compute $N''(w'')$, it is required to perform one operation of division over the numbers $N''(w'')$ and $|S_{2n}|$, i.e., over numbers of lengths $n\log m$ and $2n$. This requires the time $M(n\log m) = O(M(n))$. To compute $N'(w') = N(w) - |S_{2n}|\cdot N''(w'')$, it is required to perform one operation of multiplication of numbers of lengths $n\log m = O(n)$ and $2n = O(n)$ and one operation of subtraction of numbers of lengths $n(\log m) = O(n)$. To do this, the time $O(M(n)) + O(n) = O(M(n))$ is needed.

Let us estimate the time needed to compute the word w' from $N'(w')$. To compute $\lambda^+(a,b)$ and $\lambda^-(a,b)$ from formulas (3.9), it is required to perform two operations of division, with the length of the numerator and the denominator of the dividend not exceeding

$$2^{a+3}\lceil\log q_{\max}\rceil + 4 = 2^{a+3}(2\log n + 2) + 4 \quad \text{bits}$$

and, for the divisor,

$$2^{a+1}\lceil\log q_{\max}\rceil = 2^{a+1}(2\log n + 2) \quad \text{bits}.$$

The lengths of the dividend and the divisor are proportional to the lengths of the factors used to compute λ_b^a from (2.8). Therefore, the computation time for $\lambda^+(a,b)$ and $\lambda^-(a,b)$ will be proportional to the computation time for λ_b^a in the enumeration. In addition, in the denumeration, just as in the enumeration, the values of the quantities λ_b^a and ρ_b^a or the pairs of quantities of the same length $\lambda_b'^a$, $\lambda_b''^a$, and $\rho_b'^a$, $\rho_b''^a$ are computed in the same way.

In addition, in order to compare $\lambda_b'^a + \rho_b'^a$ with $\lambda^+(a+1,(b-1)/2)$ and $\lambda_b''^a$ with $\lambda^-(a+1,(b-1)/2)$, it is required to compare numbers whose lengths do not exceed $2^{a+2}\lceil\log q_{\max}\rceil + 4$. The comparison time depends linearly on the length of the compared numbers, which is asymptotically less than the

time needed for their multiplication and division. Therefore, the time needed for the operations of computing the estimates $\lambda^+(a,b)$, $\lambda^-(a,b)$, the quantities λ_b^a and ρ_b^a, (or the pairs $\lambda_b'^a$, $\lambda_b''^a$, $\rho_b'^a$, $\rho_b''^a$) and of comparing $\lambda_b'^a + \rho_b'^a$ with $\lambda^+(a+1,(b-1)/2)$ and $\lambda_b''^a$ with $\lambda^-(a+1,(b-1)/2)$, where $0 \le a \le \log(2n)$, $1 \le b \le 2n/2^a$, is proportional to the time of encoding w':

$$O(\log nM(2n\log n)) = O(\log nM(n\log n)).$$

Let us estimate the time needed to determine the letter x_i or the pair x_i', x_i'', $1 \le i \le 2n$. To find one letter or one pair of letters, we need at most two comparisons of words whose length does not exceed $2^2\lceil \log q_{\max}\rceil + 4 = 8\log n + 10$ bits. Hence we see that the total number of operations over the bit words needed at this stage is $O(n\log n)$.

Thus, we find that the time needed to find the word w' from its number $N(w')$ is

$$O(\log nM(n\log n)) + O(n\log n) = O(\log nM(n\log n)).$$

We can note that the time needed for the denumeration of the word w'' is proportional to the time of its enumeration, because the time $M(n)$ needed for the multiplication of two words of length n is equal to the time needed for the division of two words of length n, i.e., the time needed for the denumeration of w'' is $O(\log nM(n))$.

Thus, the total time needed to find the word w from its number, is

$$O(M(n)) + O(\log nM(n\log n)) + O(\log nM(n)) = O(\log nM(n\log n)).$$

The rate of the computation of the word w from its number, i.e., the time needed to compute one letters of the word w from its number, is $O(\log nM(n\log n)/n)$. The theorem is proved. □

REFERENCES

1. T. J. Lunch, "Sequence time coding for data compression," Proc. IEEE **54** (10), 1490–1491 (1966).
2. Yu. S. Medvedeva and B. Ya. Ryabko, "A fast algorithm for the enumeration of words with given constraints on the run lengths of ones," Problemy Peredachi Informatsii **46** (4), 130–139 (2010) [Probl. Inform. Transm. **46** (4), 390–399 (2010)].
3. T. M. Cover, "Enumerative Source Encoding," IEEE Trans. Information Theory **IT-19** (1), 73–77 (1973).
4. B. Ya. Ryabko, "Fast enumeration of combinatorial objects," Diskret. Mat. **10** (2), 101–119 (1998) [Discrete Math. Appl. **8** (2), 163–182 (1998)].
5. A. E. Pentus and M. R. Pentus, *Theory of Formal Languages*, A manual (Izd. TsPI, Mekh.-Mat. Moskov. Univ., Moscow, 2004) [in Russian].
6. E. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms*: *Theory and Practice* (Prentice–Hall, Englewood Cliffs, NJ, 1977; Mir, Moscow, 1980).
   Compilers: Principles, Techniques, and Tools[1] is a computer science textbook by Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman about compiler construction
7. A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, *Compilers*: *Principles, Techniques, and Tools* (Pearson Education, Inc., 2006; Williams, Moscow, 2008).
8. R. E. Krichevskii, *Information Compression and Search* (Radio i Svyaz, Moscow, 1989) [in Russian].
9. A. Schönhage and V. Strassen, "Schnelle Multiplikation grosser Zahlen," Computing **7** (3-4), 281–292 (1971).
10. M. Fürer, "Faster integer multiplication," in *Proceedings of the 39th Annual ACM Symposium on Theory of Computing* (ACM, New York, 2007), pp. 57–66.
11. A. Shen', *Programming: Theorems and Problems* (MTsNMO, Moscow, 2004) [in Russian].