Аннотация

Предлагается алгоритм быстрого нумерационного кодирования для основных задач теории информации, таких как:

1) кодирование двоичных слов заданной длины с заданным количеством единиц и частный случай этой задачи, когда количество единиц в слове равно количеству нулей;

2) кодирование слов с ограничением на количество подряд идущих одинаковых символов. Эта задача имеет приложение в магнитной записи и некоторых других областях;

3) кодирование элементов грассманиана и кодирование слов языков Дика.

Для решения этих задач применяется метод быстрой нумерации и денумерации комбинаторных объектов, предложенный Б. Рябко [21]. При этом метод денумерации был существенно модифицирован. Предлагаемый нами алгоритм имеет меньшую вычислительную сложность, чем другие известные алгоритмы.

1 Введение

В теории информации есть несколько задач создания быстрых алгоритмов кодирования и декодирования некоторых особых множеств слов.

Первая из таких задач — задача создания быстрого нумерационного кода для двоичных слов заданной длины с заданным количеством единиц.

Обозначим через S множество всех двоичных слов заданной длины, имеющих заданное количество единиц. Алгоритм нумерационного кодирования позволяет по данному слову из множества S находить его кодовое слово или номер, т. е. целое число из промежутка $[0, |S| - 1]$. Алгоритм нумерационного декодирования позволяет по кодовому слову, т. е. целому числу из промежутка $[0, |S| - 1]$, находить соответствующее ему слово из множества S. Особый интерес имеет частный случай этой задачи, когда количество единиц равно половине длины слова [15].

Следующая задача, привлекающая внимание многих исследователей — задача создания быстрых нумерационных кодов для слов с заданным ограничением на количество подряд идущих одинаковых символов ($dklr$-последовательностей). Эта задача имеет важное приложение в магнитной и оптической записи, так же как в оптической передаче данных и была рассмотрена во множестве работ, таких как [14], [1], [3], [9] и др.

Третья задача — быстрые нумерационные коды для элементов грассманиана. Эта задача имеет приложение в сетевом кодировании [5], [13], [7], [12], [12].

Мы также рассматриваем быстрый нумерационный код для слов языков Дика. Кодируемое множество слов в этом случае — это множество сбалансированных последовательностей длины $2n$ открывающих и закрывающих скобок k типов. Необходимость быстрой нумерации и денумерации слов языков Дика возникает при работе трансляторов языков высокого уровня, для сжатия правильных последовательностей скобок и случайной генерации правильных последовательностей скобок [20], [16], [17].

В данной работе описывается модификация метода из работы [21] для быстрых алгоритмов нумерации и денумерации и предлагается основывающийся на данном методе алгоритм для названных задач. Алгоритм описывается на примере нумераци слов языков Дика. Некоторые результаты, связанные с кодированием последовательностей с ограничением на количество подряд идущих одинаковых символов опубликованы в работе автора [19]. Результаты, связанные с кодированием элементов грассманиана, опубликованы в работе [18].

2 Нумерационное кодирование слов языков Дика

Словами языка Дика над алфавитом, состоящим из $2m$ букв являются последовательности правильно вложенных скобок m типов. Рассмотрим в качестве примера все слова длины $n = 4$ языка Дика над шестью буквами, т. е. последовательности длины 4 правильно вложенных скобок трёх типов. Всего таких слов 18, см. таблицу 1. Сопоставим им номера, записанные в двоичном виде, длины $\lceil \log_2 18 \rceil = 5$. В первый столбец запишем все такие слова, а во второй запишем их номера в двоичном виде.

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

Таблица 1.

Алгоритм нумерации ставит слову, принадлежащему языку Дика над алфавитом, состоящим из $2m$ букв, длины n последовательность из нулей и единиц, т. е. его номер. Например, для множества слов языка Дика над алфавитом 6 длины 4, расположенных в порядке, указанном в таблице, по данному слову ( ) { } алгоритм должен находить его номер 00101.

В данной работе мы рассмотрим быстрый алгоритм нумерации и денумерации на примере множества слов языков Дика над алфавитом $\{0, 1\}$.

Алгоритм нумерации слов длины n языков Дика над алфавитом $2m$, основанный на методе Ковера [2], имеет сложность $O(n^2)$ битовых операций на одно нумеруемое слово, или $O(n)$ битовых операций на один символ нумеруемого слова.

Метод нумерации слов длины n языков Дика над алфавитом $2m$, предлагаемый в данной работе и основанный на подходе из работы [21], имеет сложность $O(\log n/n\, M(n \log n))$ битовых операций на один символ нумеруемого слова, где $M(n \log n)$ время умножения или деления слов длины $n \log n$. Если использовать метод Шёнхаге-Штрассена [10], сложность которого $n \log n \log\log n$ при умножении или делении слов длины n, то сложность рассматриваемого метода $O(\log^3 n \log\log n)$ на один символ нумеруемого слова. Если использовать метод Фюрера [4], сложность которого $n \log n 2^{O(\log^* n)}$ при умножении или делении слов длины n, то сложность рассматриваемого метода $O(\log^3 n 2^{O(\log^* n)})$ на один символ нумеруемого слова.

2.1 Нумерация слов языков Дика над алфавитом $\{0, 1\}$

Обозначим D_n^{2m} множество слов языка Дика над алфавитом, состоящим из $2m$ букв, длины n.

Покажем, как применяется быстрый алгоритм нумерации, предлагаемый в данной статье, для нумерации слов языка Дика длины n над алфавитом $\{0, 1\}$, т. е. слов множества D_n^2.

В качестве примера будем искать номер слова $w = 01010011 \in D_8^2$.

Описание будет удобно начать с описания нахождения номера слова w среди множества S, где S — произвольное множество слов длины n, который мы обозначим через $code_S(w)$ с помощью метода Ковера из [2].

Согласно [2], номер слова $w = x_1 \dots x_n$ из заданного множества слов S длины n, упорядоченного лексикографически, можно найти по формуле

$$code_S(w) = \sum_{i=1}^{n} \sum_{\chi < x_i} N_S(x_1 x_2 ... x_{i-1} \chi), \qquad (1)$$

где $N_S(x_1 x_2 ... x_{i-1} \chi)$ — количество слов множества S, начинающихся с $x_1 x_2 ... x_{i-1} \chi$.

Для того, чтобы использовать эту формулу для нумерации слов множества D_n^2, нужно опеределить, чему равны значения $N_{D_n^2}(x_1 \dots x_i)$, $0 < i \leq n$.

Найдем, чему равно $N_{D_8^2}(01)$, т. е. сколько слов множества D_8^2 начинаются на 01. Словами множества D_8^2, начинающимися на 01 будут являться слова, состоящие из четырех нулей и четырех единиц, которые начинаются на 01, кроме тех, которые не соответствуют правильным расстановкам скобок. Количество всех слов, состоящих из четырех нулей и четырех единиц, начинающихся на 01 легко найти, оно равно количеству всех слов, состоящих из трех нулей и трех единиц, т. е. $C_6^3 = 20$.

Слова, начинающиеся на 01, состоящие из четырех нулей и четырех единиц и не соответствующие правильным расстановкам скобок, это такие слова из четырех нулей и четырех единиц, для которых существует такое j, $2 < j \leq 8$, что количество единиц в последовательности $x_1x_2...x_j$ превышает количество нулей в этой последовательности. Существует взаимоднозначное соответствие таких слов и всех слов, состоящих из трёх нулей и пяти единиц и начинающихся на 01. Это отображение осуществляется следующим образом. Для слова, не соответствующего правильной расстановке скобок, есть такие j, $2 < j \leq 8$, что количество единиц в последовательности $x_1x_2...x_j$ превышает количество нулей в этой последовательности. Для каждого такого слова можно найти минимальное среди всех j. Можно видеть, что для такого j количество единиц в последовательности $x_1x_2...x_j$ превышает количество нулей на один символ. Заменим теперь в слове все символы после j-го на противоположные. Получаем слово из трёх нулей и пяти единиц, начинающееся на 01. Т. к. это отображение взаимоднозначное, количество слов, начинающихся на 01, состоящих из четырех нулей и четырех единиц и не соответствующих правильным расстановкам скобок, равно количеству всех слов, начинающихся на 01 и состоящих из трёх нулей и пять единиц. Таких слов столько же, сколько слов, состоящих из двух нулей и четырех единиц, т. е. $C_6^2 = 15$. Таким образом, $N_{D_8^2}(01) = C_6^3 - C_6^2 = 5$.

В общем виде, $C_{n-i}^{n/2-z} - C_{n-i}^{n/2-z-1} = (n-i)!/((n/2-z)!\,(n/2-i+z)!) - (n-i)!/((n/2-z-1)!\,(n/2-i+z+1)!) = (n-i)!(2z-i+1)/((n/2-z)!\,(n/2-i+z+1)!)$, т. е.

$$N_{D_n^2}(x_1x_2...x_i) = \frac{(n-i)!(2z-i+1)}{(n/2-z)!\,(n/2-i+z+1)!}, \quad (2)$$

где z — количество нулей в $x_1x_2...x_i$, при том, что $x_1x_2...x_i$ может быть началом слова, соответствующего правильной расстановке скобок длины n. Если $x_1x_2...x_i$ не может быть началом слов, соответствующих правильной расстановке скобок длины n, то очевидно $N_{D_n^2}(x_1...x_i) = 0$.

При применении метода Ковера используется вспомогательная таблица, в которой хранятся все возможные значения $N_S(x_1x_2...x_{i-1}\chi)$ или все возможные значения $\sum \chi < x_i N_S(x_1x_2...x_{i-1}\chi)$, $0 < i \leq n$. Эта таблица строится один раз и затем используется для всех последующих поисков номера слов множества S.

В случае множества D_n^2 достаточно таблицы, в которой каждой паре i, $0 < i \leq n$, и z, $0 \leq z \leq i$ сопоставляется значение $N_{D_n^2}(x_1...x_i) = \frac{(n-i)!(2z-i+1)}{(n/2-z)!\,(n/2-i+z+1)!}$ (2). Размер такой таблицы равен $O(n^3)$.

Для получения номера слова $w \in D_n^2$ согласно (1) для каждого i, $0 < i \leq n$, такого, что $x_i = 1$, находится значение z, равное количеству нулей в слове $x_1...x_{i-1}0$, затем находится с помощью таблицы находится соответствующее паре i и z значение $N_{D_n^2}(x_1...x_i)$, затем складываются все найденные значения $N_{D_n^2}(x_1...x_i)$.

В данном примере: $code_{D_8^2}(01010011) = N_{D_8^2}(00) + N_{D_8^2}(0100) + N_{D_8^2}(0101000) + N_{D_8^2}(01010010) = 9 + 3 + 0 + 0 = 12$.

Значения $N_{D_8^2}(00)$, $N_{D_8^2}(0100)$ при этом берутся из заранее построенной таблицы. При значениях i, равных 2, 4, 6, 8, выполняется $x_i = 1$. При $i = 2$ слово $x_{i-1}0$ равно 00, следовательно $z = 2$, поэтому из таблицы берется значение $N_{D_8^2}(00)$, соответствующее паре $i = 2\,z = 2$) и равное $(6!\,3)/(2!\,5!) = 9$. При $i = 4$ слово $x_1 \ldots x_{i-1}0$ равно 0100, следовательно $z = 3$, поэтому по таблице находится значение $N_{D_8^2}(0100)$, соответствующее паре $i = 4$, $z = 3$) и равное $(4!\,3)/(1!\,4!) = 3$. Значения же $N_{D_8^2}(0101000)$ и $N_{D_8^2}(01010010)$ равны нулю, т. к. не существует слов множества D_8^2, начинающихся на 0101000 или 01010010.

Мы видим, что для такого вычисления требуется совершить максимум n операций сложения слов длин от 1 до n. Т. о., если использовать вспомогательную таблицу, то сложность вычисления номера слова по методу Ковера равно $O(n^2)$ или $O(n)$ на один символ нумеруемого слова.

Перейдем теперь к описанию нумерации слова $w = x_1 \ldots x_n$ заданного множества S слов длины n предлагаемым быстрым алгоритмом нумерации, затем покажем как применяется этот алгоритм для нумерации слов множества D_n^2.

Определим величины $P(x_i|x_1 \ldots x_{i-1})$,

$q(x_i|x_1 \dots x_{i-1})$ при $0 < i \le n$

$$\begin{gathered} P(x_1) = \frac{N_S(x_1)}{|S|}, \\ P(x_i|x_1x_2\dots x_{i-1}) = \frac{N_S(x_1x_2\dots x_i)}{N_S(x_1x_2\dots x_{i-1})}, \\ q(x_1) = \sum_{\chi<x_1} P(\chi), \\ q(x_i|x_1\dots x_{i-1}) = \sum_{\chi<x_i} P(\chi|x_1\dots x_{i-1}). \end{gathered} \tag{3}$$

Можно видеть, что по (1) $code_S(x_1 \dots x_n) = |S|(q(x_1) + q(x_2|x_1)P(x_1) + q(x_3|x_1x_2)P(x_2|x_1)P(x_1) + \dots)$. Идея метода заключается в расстановке скобок в этом выражении таким образом, что при вычислении номера большинство операций производится над короткими числами. Такой расстановкой скобок будет являться:

$$\begin{gathered} code_S(x_1\dots x_n) = \\ = |S|((q(x_1) + q(x_2|x_1)P(x_1)) + ((q(x_3|x_1x_2)+ \\ +q(x_4|x_1\dots x_3)P(x_3|x_1x_2))P(x_2|x_1)P(x_1)) + \dots) \end{gathered} \tag{4}$$

Определим величины ρ_b^a, λ_b^a при $0 \le a \le \log n$, $1 \le b \le n/2^a$ следующим образом:

$$\begin{gathered} \rho_b^0 = P(x_b|x_1\dots x_{b-1}),\ \lambda_b^0 = q(x_b|x_1\dots x_{b-1}), \\ \rho_b^a = \rho_{2b-1}^{a-1}\rho_{2b}^{a-1},\ \lambda_b^a = \lambda_{2b-1}^{a-1} + \rho_{2b-1}^{a-1}\lambda_{2b}^{a-1}. \end{gathered} \tag{5}$$

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

Отсюда и (4):

$$code_S(x_1x_2\dots x_n) = \lambda_1^{\log n}\,|S| \tag{6}$$

Алгоритм заключается в том, что сначала находятся значения $P(x_i|x_1\dots x_{i-1})$, $q(x_i|x_1\dots x_{i-1})$ при $0 < i \le n$, определенные в (3)), затем находится $\lambda_1^{\log n}$ последовательным вычислением значений ρ_b^a, λ_b^a при $0 \le a \le \log n$, $1 \le b \le n/2^a$ формулам (5), затем находится искомый номер $code_S(w)$ по формуле (6), при этом значение $|S|$ находится до начала нумерации и используется затем при всех последующих нумерациях.

Покажем, как применить быстрый алгоритм нумерации для нумерации слов множества D_n^2.

До начала нумерации необходимо вычислить мощность D_n^2. Мощность такого множества равна $n/2$-ому числу Каталана [22],

$$|D_n^2| = C_n = C_n^{n/2} - C_n^{n/2-1}. \tag{7}$$

В нашем примере $|D_8^2| = C_8^4 - C_8^3 = 14$.

Можно видеть из (2) и определения (3), что для множества D_n^2 значения $P(x_i|x_1\dots x_{i-1})$ $(0 < i \le n)$ находятся следующим образом:

$$\begin{gathered} P(x_i|x_1x_2\dots x_{i-1}) = \\ = \frac{(n-i)!(2z-i+1)}{(n/2-z)!\,(n/2-i+z+1)!} : \\ : \frac{(n-i+1)!(2z-i)}{(n/2-z+1)!\,(n/2-i+z+1)!} = \\ = \frac{(2z-i+1)(n/2-z+1)}{(2z-i)(n-i+1)} \end{gathered} \tag{8}$$

при $x_i = 0$,

$$\begin{gathered} P(x_i|x_1x_2\dots x_{i-1}) = \\ = \frac{(n-i)!(2z-i+1)}{(n/2-z)!\,(n/2-i+z+1)!} : \\ : \frac{(n-i+1)!(2z-i+2)}{(n/2-z)!\,(n/2-i+z+2)!} = \\ = \frac{(2z-i+1)(n/2-i+z+2)}{(n-i+1)(2z-i+2)} \end{gathered} \tag{9}$$

при $x_i = 1$.

Таким образом, для нашего примера находим по формулам (8) и (9) значения $P(x_1) = \rho_1^0$, $P(x_2|x_1) = \rho_2^0$, $\dots$, $P(x_8|x_1x_2\dots x_7) = \rho_8^0$, $q(x_1) = \lambda_1^0$, $q(x_2) = \lambda_2^0$, $\dots$, $q(x_8) = \lambda_8^0$, $P(x_1) = 1$, $P(x_2|x_1) = P(1|0) = 5/14, P(x_3|x_1x_2) = P(0|01) = 6/6$, $P(x_4|x_1x_2x_3) = P(1|010) = 4/10, P(x_5|x_1\dots x_4) = P(0|0101) = 4/4$, $P(x_6|x_1\dots x_5) = P(0|01010) = 3/6, P(x_7|x_1\dots x_6) = P(1|010100) = 6/6$, $P(x_8|x_1\dots x_7) = P(1|0101001) =$

$2/2, q(x_1) = q(0) = 0, q(x_2|x_1) = q(1|0) = 9/14, q(x_3|x_1x_2) = q(0|01) = 0, q(x_4|x_1x_2x_3) = q(1|010) = 6/10, q(x_5|x_1 \dots x_4) = q(0|0101) = 0$, $q(x_6|x_1 \dots x_5) = q(0|01010) = 0, q(x_7|x_1 \dots x_6) = q(1|010100) = 0$, $q(x_8|x_1 \dots x_7) = q(1|0101001) = 0$. Соответственно, по (5) $\rho_1^0 = 1$, $\rho_2^0 = 5/14$, $\rho_3^0 = 1$, $\rho_4^0 = 2/5$, $\rho_5^0 = 1$, $\rho_6^0 = 1/2$, $\rho_7^0 = 1$, $\rho_8^0 = 1, \lambda_1^0 = 0$, $\lambda_2^0 = 9/14$, $\lambda_3^0 = 0$, $\lambda_4^0 = 3/5, \lambda_5^0 = 0$, $\lambda_6^0 = 0$, $\lambda_7^0 = 0$, $\lambda_8^0 = 0$. Затем по (5) вычисляем $\rho_1^1 = 5/14$, $\rho_2^1 = 2/5$, $\rho_3^1 = 1/2$, $\rho_4^1 = 1$, $\lambda_1^1 = 9/14$, $\lambda_2^0 = 3/5$, $\lambda_3^0 = 0$, $\lambda_4^0 = 0, \rho_1^2 = 1/7$, $\rho_2^1 = 1/2$, $\lambda_1^2 = 6/7$, $\lambda_2^2 = 0$, $\rho_1^3 = 1/14$, $\lambda_1^3 = 6/7$.

По (6) $code_{D_8^2}(01010011) = \lambda_1^3 \cdot |D_8^2| = \frac{6}{7} \cdot 14 = 12$. Таким образом, мы получили $code_{D_8^2}(w)$, номер слова $01010011 \in D_8^2$.

Для исследования свойств алгоритма нумерации слов, принадлежащих множеству S, мы определим несколько величин.

Обозначим через q_{max} максимальный знаменатель дробей $N_S(x_1x_2 \dots x_i)/N_S(x_1x_2 \dots x_{i-1})$, $x_1 \dots x_n \in S$, $i = 1, \dots, n$. Из (8) и (9) следует, что для слов множества D_n^2

$$q_{max} = n^2. \tag{10}$$

Свойства предложенного метода нумерации слов длины n, принадлежащих множеству S, характеризует следующая теорема.

Теорема 1. *1) Скорость кодирования (то есть время кодирования на букву, измеряемое числом операций над однобитовыми словами) слов множества S длины n равна*

$$O(\log n M(n \log q_{max})/n), \tag{11}$$

где $M(n)$ время умножения двух слов длины n.

Следствие 1. При использовании алгоритма быстрого умножения Шёнхаге-Штрассена, для которого $M(n) = O(n \log n \log \log n)$, скорость нумерации равна $O(\log n \log q_{max} \log(n \log q_{max}) \log \log(n \log q_{max}))$.

Следствие 2. При использовании алгоритма быстрого умножения Фюрера, для которого $M(n) = O(n \log n 2^{O(\log^* n)})$, скорость нумерации равна $O(\log n \log(n \log q_{max}) 2^{O(\log^*(n \log q_{max}))})$.

2) Объем памяти в битах, используемый при кодировании слов множества S длины n, не превосходит

$$O((n \log q_{max}) \log n). \tag{12}$$

Отсюда и из (10) следуют свойства нумерации слов длины n языков Дика над алфавитом мощности 2.

Теорема 2. *1)Скорость кодирования слова длины n языка Дика над алфавитом мощности 2, т. е. время, требуемое для кодирования одной буквы, равна $O(\log n M(n \log n)/n)$ битовых операций, где $M(n)$ время умножения двух слов длины n.*

2)Объем памяти, требуемый для кодирования слова длины n языка Дика над алфавитом мощности 2 равен $O(n \log n)$ бит.

Следствие 1. При использовании алгоритма быстрого умножения ШёнхагеШтрассена, для которого $M(n) = O(n \log n \log \log n)$, скорость нумерации равна $O(\log^3 n \log \log n)$.

Следствие 2. При использовании алгоритма быстрого умножения Фюрера, для которого $M(n) = O(n \log n 2^{O(\log^* n)})$, скорость нумерации равна $O(\log^3 n 2^{O(\log^* n)})$.

Список литературы

[1] Beenker G. F. M., Immink K. A. S. A generalized method for encoding and decoding run-length-limited binary sequences // IEEE Transactions on Information Theory. 1983. V. 29, № 3. P. 751–754.

[2] Cover T. M. Enumerative source encoding // IEEE Transactions on Information Theory. 1973. V. IT-19, №. 1. P. 73–77.

[3] Datta S. and McLaughlin S. W. An enumerative method for runlength-limited codes: permutation codes // IEEE Transactions on Information Theory. 1999. V. IT-45, № 6. P. 2199–2204.

[4] Fürer M. Faster integer multiplication // Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing (STOC

2007). San Diego, California, USA. 2007. P. 57–66.

[5] Gadouleau M. and Yan Z. Constant-rank codes and their connection to constant-dimension codes // IEEE Transactions on Information Theory. 2010. V. IT-56. P. 3207–3216.

[6] Kautz W. Fibonacci codes for synchronization control // IEEE Transactions on Information Theory. 1965. V. 11, № 2. P. 284–292.

[7] Koetter R. and Kshcischang F. R. Coding for errors and erasures in random network coding // IEEE Transactions on Information Theory. 2008. V. 54, № 8. P. 3579–3591.

[8] Lint J. H. van and Wilson R. M. A Course in Combinatorics. Cambridge University Press, 2001 (second edition).

[9] Milenkovic O. and Vasic B. Permutation $(d;k)$-codes: efficient enumerative coding and phrase length distribution shaping // IEEE Transactions on Information Theory. 2000. V. IT-46, № 7. P. 2671–2675.

[10] Schönhage A. and Strassen V. Schnelle Multiplikation grosser Zahlen // Computing. 1971. V. 7. P. 281–292.

[11] Silberstein N. and Etzion T. Enumerative coding for Grassmannian space // IEEE Transactions on Information Theory. 2011. V. 57, № 1. P. 365–374.

[12] Silberstein N. and Etzion T. Error-correcting codes in projective space via rank-metric codes and Ferrers diagrams // IEEE Transactions on Information Theory. 2009. V. IT-55. P. 2909–2919.

[13] Skachek V. Recursive code construction for random networks // IEEE Transactions on Information Theory. 2010. V. IT-56. P. 1378–1382.

[14] Tang D. T., Bahl L. R. Block codes for a class of constrained noiseless channels // Information and Control. 1970. V. 17, № 5. P. 436–461.

[15] Weber, J. H., Schouhamer Immink. Knuth's balanced codes revisited // IEEE Transactions on Information Theory. 2010. V. 56. P. 1673–1679.

[16] Ахо А., Лам М., Сети Р., Ульман Дж. Компиляторы. Принципы, технологии и инструментарий. М.: Вильямс, 2008.

[17] Кричевский Р. Е. Сжатие и поиск информации. М.: Радио и связь, 1989.

[18] Медведева Ю. С. Быстрая нумерация элементов грассманиана // Вычислительные технологии. 2013. Т. 18, № 3. С. 22–33.

[19] Медведева Ю. С., Рябко Б. Я. Быстрый алгоритм нумерации слов с заданными ограничениями на длины серий единиц. // Проблемы передачи информ. 2010. Т. 46, № 4. С. 130–139.

[20] Рейнольд Э., Нивергельт Ю., Део Н. Комбинаторные алгоритмы. Теория и практика. М.: Мир, 1980.

[21] Рябко Б. Я. Быстрая нумерация комбинаторных объектов. // Дискретная математика. 1998. Т. 10. № 2. С. 101–119.

[22] Шень А. Программирование: теоремы и задачи. М.: МЦНМО, 2004.