УДК 519.163
DOI: 10.25559/SITITO.019.202303.564-574

Оригинальная статья

# Метод сжатия данных журналов событий на основе теории комбинаторной генерации с применением структур деревьев И/ИЛИ

**Ю. В. Шабля**
ФГБОУ ВО «Томский государственный университет систем управления и радиоэлектроники», г. Томск, Российская Федерация
Адрес: 634034, Российская Федерация, г. Томск, пр. Ленина, д. 40
shablya-yv@mail.ru

**Аннотация**
Экспоненциальный рост объема производимой современным обществом цифровой информации влечет за собой проблему хранения большого объема данных, в том числе архивных данных. Под архивными данными понимается категория «холодных» данных (это такие данные, которые требуют хранения, но при этом редко используются). Наглядным примером такого рода архивных данных являются данные журналов событий, содержащих краткое описание произошедших в информационной системе событий в хронологическом порядке. Учитывая большой объем архивных данных и редкое их использование, актуальным становится хранение таких данных в сжатом виде. В данной статье рассматривается задача разработки метода сжатия архивных данных на примере данных журналов событий за счет применения алгоритмов комбинаторной генерации. В частности, если зафиксировать некоторое текущее состояние журнала событий, то множество его записей может быть рассмотрено как комбинаторное множество. Тогда, используя алгоритм ранжирования элементов комбинаторного множества, каждую запись журнала событий можно закодировать одним числом, для хранения которого потребуется меньше памяти. На базе данной идеи предложен метод сжатия данных журналов событий на основе теории комбинаторной генерации с применением структур деревьев И/ИЛИ. Для оценки эффективности предложенного метода рассмотрен пример сжатия данных журналов событий, генерируемых внутри электронных курсов системы Moodle. Результаты экспериментального исследования подтвердили эффективность предложенного метода, а именно: суммарный объем памяти, требуемой для хранения журнала события электронного курса системы Moodle в сжатом виде, имеет меньшее значение по сравнению с существующими методами сжатия текстовых файлов.

**Ключевые слова:** комбинаторная генерация, журнал событий, сжатие данных, дерево И/ИЛИ, алгоритм ранжирования

**Финансирование:** Исследование выполнено за счет гранта Российского научного фонда (проект № 22-71-10052 «Разработка математического, алгоритмического и программного обеспечения комбинаторной генерации для решения задач хранения и обработки больших объёмов данных»).

**Конфликт интересов:** автор заявляет об отсутствии конфликта интересов.

**Для цитирования:** Шабля Ю. В. Метод сжатия данных журналов событий на основе теории комбинаторной генерации с применением структур деревьев И/ИЛИ // Современные информационные технологии и ИТ-образование. 2023. Т. 19, № 3. С. 564-574. https://doi.org/10.25559/SITITO.019.202303.564-574

Original article

# A Method for Compressing Event Log Data Based on Combinatorial Generation Using AND/OR Tree Structures

**Y. V. Shablya**
Tomsk State University of Control Systems and Radioelectronics, Tomsk, Russian Federation
Address: 40 Lenin Ave., Tomsk 634034, Russian Federation
shablya-yv@mail.ru

## Abstract

The exponential growth in the volume of digital information produced by modern society entails the problem of storing large amounts of data, including archival data. Archival data refers to the category of "cold" data (data that requires storage, but is rarely used). A clear example of this type of archival data is data from event logs, which contain a brief description of events that occurred in the information system in chronological order. Due to the large amount of archival data and its rare use, it is relevant to store such data in compressed form. This article discusses the problem of developing a method for compressing archival data using the example of event log data by applying combinatorial generation algorithms. In particular, if we fix some current state of the event log, then the set of its entries can be considered as a combinatorial set. Then, using an algorithm for ranking elements of the combinatorial set, each event log entry can be encoded with a single number, which will require less memory to store. Based on this idea, a method for compressing event log data based on combinatorial generation using AND/OR tree structures is proposed. To evaluate the effectiveness of the proposed method, an example of compressing event log data generated within Moodle electronic courses is considered. The results of the experimental study confirmed the effectiveness of the proposed method: the total amount of memory required to store the event log of a Moodle electronic course in the compressed form is less compared to the existing methods for compressing text files.

**Keywords:** combinatorial generation, event log, data compression, AND/OR tree, ranking algorithm

**Funding:** The research was supported by a grant from the Russian Science Foundation (project No. 22-71-10052 "Development of Mathematical, Algorithmic and Combinatorial Generation Software for Solving Problems of Storing and Processing Large Volumes of Data").

**Conflict of interests:** The author declares no conflict of interests.

**For citation:** Shablya Y.V. A Method for Compressing Event Log Data Based on Combinatorial Generation Using AND/OR Tree Structures. *Modern Information Technologies and IT-Education.* 2023;19(3):564-574. https://doi.org/10.25559/SITITO.019.202303.564-574

## 1. Введение

Современное информационное общество характеризуется экспоненциальным ростом объема производимой информации. Использование цифровых технологий в различных аспектах жизни вошло в норму жизнедеятельности общества. Опубликованный отчет Digital 2023: Global overview report[1] подтверждает тот факт, что число пользователей различных цифровых устройств, социальных сетей и в целом сети Интернет неуклонно растет. Соответственно, растет и объем цифровых данных, производимых такими пользователями.

По подсчетам International Data Corporation[2], в 2018 году общий объем созданной цифровой информации в мире составил 33 зеттабайта, что эквивалентно 33 триллионам гигабайт. При этом, согласно составленным прогнозам, к 2025 году общий объем созданной цифровой информации в мире достигнет 175 зеттабайт, что также подтверждает экспоненциальный рост объема производимой информации. Кроме того, генерируемая информация требует временного либо постоянного ее хранения на различных носителях. Причем большая часть длительно хранимых данных относится к архивным, т. е. не требующих постоянного доступа к ним.

Таким образом, возникает проблема хранения большого объема архивных данных, которая с каждым годом становится все более актуальной [1]. В частности, потребность в хранении больших объемов архивных данных наблюдается при проведении различных исследований (например, результаты проводимых научных экспериментов и сформированные на их основе наборы данных) [2].

С точки зрения частоты доступа к хранимым данным выделяют следующие три категории [3]:

1) «горячие» данные – данные, которые используются постоянно;

2) «теплые» данные – данные, к которым требуется регулярный доступ;

3) «холодные» данные – данные, которые редко используются.

В данной работе под архивными данными понимается категория «холодных» данных. Проблема, связанная с хранением архивных данных, может быть рассмотрена с точки зрения организации процессов хранения и доступа к данным. Тогда одним из актуальных направлений является организация облачных хранилищ архивных данных[3] [4]. Однако, учитывая большой объем архивных данных и редкое их использование, имеет смысл хранить такие данные в сжатом виде [5-7]. В таком случае архивные данные предварительно обрабатываются и хранятся уже в сжатом виде, а при наступлении момента, когда их нужно использовать, данные восстанавливаются в исходный вид.

Таким образом, возникает потребность в разработке новых методов сжатия архивных данных, которые будут эффективны с точки зрения занимаемого объема памяти для хранения. Целью данной работы является разработка метода сжатия архивных данных на примере данных журналов событий с использованием теории комбинаторной генерации.

## 2. Постановка задачи

Журналирование событий является типовым процессом, реализуемым в различного рода информационных системах. Результатом работы такого процесса является создание журналов событий, содержащих краткое описание произошедших в системе событий в хронологическом порядке. При этом обращение к содержимому журналов событий чаще всего совершается лишь изредка, например, при наступлении какого-либо сбоя в системе [8]. Таким образом, данные журналов событий можно считать архивными данными, и их можно хранить в сжатом виде [9-10].

С точки зрения внутреннего содержимого журналы событий имеют фиксированную структуру и описывают журналируемые события с помощью шаблонных текстовых структур. С точки зрения организации способа хранения журнал событий представляет собой обычный текстовый файл, где каждая строка описывает произошедшее в системе событие. В таком случае файл журнала событий может занимать большой объем памяти для его хранения. Также можно встретить реализацию журнала событий в формате файла базы данных.

Для решения задачи сжатия данных журналов событий предлагается воспользоваться методами комбинаторной генерации [11-12][4]. В частности, если зафиксировать некоторое текущее состояние журнала событий без возможности добавления в него новых записей (например, журнал событий за определенный прошедший период времени), то множество записей журнала событий может быть рассмотрено как комбинаторное множество. Тогда, используя алгоритмы ранжирования элементов комбинаторных множеств, можно кодировать каждую запись журнала событий одним числом (рангом), представляющим собой порядковый номер этой записи среди некоторого заданного упорядочивания конечного множества всех возможных записей. Восстановление исходного вида записей журнала событий возможно за счет применения обратного алгоритма генерации элементов комбинаторного множества по их рангам.

Существует подход [13], позволяющий представить множество кортежей реляционной базы данных в форме структуры дерева И/ИЛИ с последующим применением для них соответствующей методологии комбинаторной генерации [14-16][5].

[1] Kemp S. Digital 2023: Global overview report [Электронный ресурс] // DataReportal, 2023. URL: https://datareportal.com/reports/digital-2023-global-overview-report (дата обращения: 01.09.2023).

[2] Reinsel D., Gantz J., Rydning J. The Digitization of the World: From Edge to Core [Электронный ресурс] // International Data Corporation, 2018. URL: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf (дата обращения: 01.09.2023).

[3] Коробейникова К. В. Облачное хранилище как модель хранения архивных данных // Управление информацией и документацией в цифровой среде : материалы Международной научно-практической конференции. Донецк : ДонНу, 2022. С. 63-67. EDN: ZGYUEC

[4] Ruskey F. Combinatorial Generation. Working version (1j-CSC 425/520). 2003. 311 p. [Электронный ресурс]. URL: https://page.math.tu-berlin.de/~felsner/SemWS17-18/Ruskey-Comb-Gen.pdf (дата обращения: 01.09.2023).

[5] Кручинин В. В. Методы построения алгоритмов генерации и нумерации комбинаторных объектов на основе деревьев И/ИЛИ : монография. Томск : В-Спектр, 2007. 199 с. EDN: QJTHIZ

Дерево И/ИЛИ является древовидной структурой, в которой внутренние узлы могут быть двух типов: И-узел (аналог операции декартова произведения множеств) и ИЛИ-узел (аналог операции объединения непересекающихся множеств) [17-18]. В таком случае структура кортежа реляционной базы данных представляется И-узлом дерева И/ИЛИ, потомками которого являются узлы, количество которых определяется количеством содержащихся в кортеже атрибутов. При этом каждый атрибут кортежа представляется ИЛИ-узлом дерева И/ИЛИ, потомками которого являются листья, количество которых определяется количеством уникальных значений соответствующего атрибута. Тогда одно из возможных содержательных наполнений кортежа может быть представлено вариантом дерева И/ИЛИ. Вариантом дерева И/ИЛИ называется дерево, которое получается из заданного путем отсечения у всех ИЛИ-узлов всех ребер, кроме одного. Количество вариантов дерева И/ИЛИ определяется типом корневого узла и количеством вариантов в поддеревьях его потомков: если это И-узел, то необходимо перемножить количество вариантов в поддеревьях его потомков; если это ИЛИ-узел, то необходимо сложить количество вариантов в поддеревьях его потомков. Для краткого обозначения количества вариантов в поддереве некоторого узла $s$ дерева И/ИЛИ используется следующее обозначение: $w(s)$.

Формальное описание структуры кортежей атрибутов:

$n$ – количество атрибутов, содержащихся в кортеже;

$A_i$ – множество возможных значений $i$-го атрибута ($i = 1, ..., n$);

$m_i = |A_i|$ – количество возможных значений $i$-го атрибута;

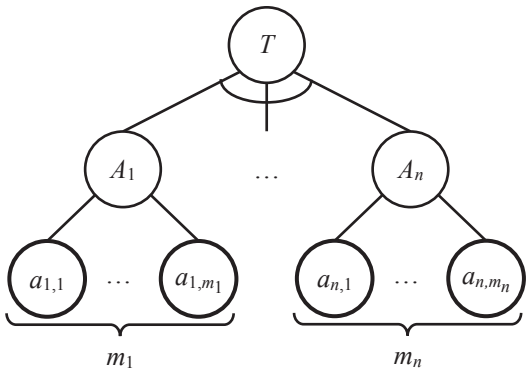
$a_{i,j}$ – $j$-е возможное значение $i$-го атрибута ($j = 1, ..., m_i$), т. е. $A_i = \{a_{i,1}, ..., a_{i,mi}\}$;

$a_i$ – конкретное выбранное значение $i$-го атрибута, т. е. $a_i \in A_i$;

$t = < a_1, ..., a_n >$ – кортеж выбранных значений $n$ атрибутов, т. е. $t \in A_1 \times A_2 \times ... \times A_n$;

$T = A_1 \times A_2 \times ... \times A_n$ – множество возможных кортежей $n$ атрибутов.

На рис. 1 изображен пример представления множества возможных кортежей $n$ атрибутов в виде структуры дерева И/ИЛИ.




Р и с. 1. Представление множества возможных кортежей $n$ атрибутов в виде структуры дерева И/ИЛИ

F i g. 1. Representing the set of possible tuples of $n$ attributes as an AND/OR tree structure

*Источник:* здесь и далее в статье все рисунки составлены автором.

*Source:* Hereinafter in this article all figures were drawn up by the author.

Каждая запись журнала событий также может быть рассмотрена как кортеж значений атрибутов. Внутреннее содержимое журнала событий имеет фиксированную структуру, следовательно, каждая его запись состоит из значений одних и тех же атрибутов. Кроме того, если рассматривается сжатие некоторого зафиксированного состояния журнала событий без возможности добавления в него новых записей, то множество возможных значений каждого атрибута является конечным. В результате получаем, что структура записей журнала событий аналогичным образом может быть представлена в виде структуры дерева И/ИЛИ. Тогда, чтобы сжать записи журнала событий, необходимо каждую из них представить в виде варианта соответствующей структуры дерева И/ИЛИ и вычислить ранг полученного варианта с помощью алгоритма ранжирования.

Далее представлено подробное описание предлагаемого метода сжатия данных журналов событий на основе теории комбинаторной генерации с применением структур деревьев И/ИЛИ и рекомендации по его применению.

## 3. Метод сжатия данных журналов событий

Предлагаемый метод сжатия данных журналов событий на основе теории комбинаторной генерации состоит из выполнения следующих шагов:

**Шаг 1. Определение кодируемых атрибутов журнала событий**

В ходе данного шага требуется сначала определить, из каких $n$ атрибутов состоит каждая запись исследуемого журнала событий. Таким образом, каждая запись журнала событий представляется как кортеж значений $n$ атрибутов $t = < a_1, ..., a_n >$. Далее необходимо указать, какие именно $k$ атрибутов ($k = 1, ..., n$) будут подвержены сжатию за счет применения методов комбинаторной генерации, при этом оставшиеся атрибуты будут записаны в исходном виде без сжатия. Тогда кортеж значений $n$ атрибутов $< a_1, ..., a_n >$ может быть представлен следующим образом: $< c_1, ..., c_{n-k}, b_1, ..., b_k >$, где $b_i$ – значение $i$-го атрибута ($i = 1, ..., k$) среди подверженных дальнейшему сжатию $k$ атрибутов, $c_i$ – значение $i$-го атрибута ($i = 1, ..., n - k$) среди оставшихся $n - k$ атрибутов. При этом допускается изменение порядка атрибутов в кортеже относительно исходного расположения с сохранением инструкций по восстановлению исходного упорядочивания, т. е. возможна ситуация, когда выполняется неравенство $< a_1, ..., a_n > \neq < c_1, ..., c_{n-k}, b_1, ..., b_k >$, но при этом кортежи состоят из одних и тех же элементов: $\{ a_1, ..., a_n \} = \{ c_1, ..., c_{n-k}, b_1, ..., b_k \}$. Рекомендацией для того, чтобы оставить какой-либо атрибут без его последующего сжатия, является отсутствие значительного дублирования в его значениях, так как сжатие происходит за счет удаления подобного рода избыточности в исходных данных.

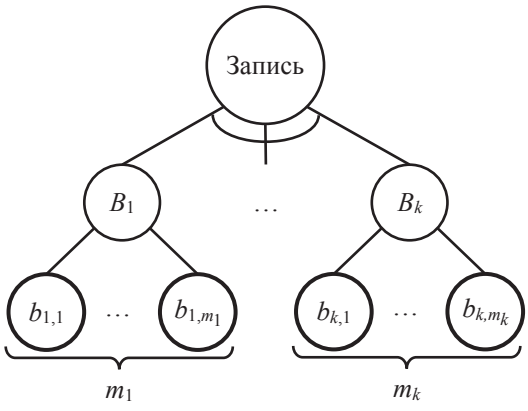
**Шаг 2. Составление справочников уникальных значений атрибутов журнала событий**

В ходе данного шага требуется для каждого подверженного дальнейшему сжатию атрибута определить множество его уникальных значений. В результате чего формируются $k$ множеств $B_i$ ($i = 1, ..., k$), каждое из которых представляет-

ся в виде файла-справочника. Каждая запись (строка) такого файла-справочника представляет собой одно из уникальных значений атрибута, а порядковый номер записи (строки) является ее идентификатором. При этом возможны различные варианты упорядочивания записей внутри файла-справочника: например, если расположить записи в лексикографическом порядке, то это ускорит процесс поиска записей внутри файла-справочника и, следовательно, ускорит работу алгоритмов комбинаторной генерации.

**Шаг 3. Построение структуры дерева И/ИЛИ для кодируемых атрибутов журнала событий**

В ходе данного шага требуется построить структуру дерева И/ИЛИ, варианты которого будут описывать кортежи значений, подверженных дальнейшему сжатию $k$ атрибутов $< b_1, ..., b_k >$. Структура такого дерева идентична структуре дерева И/ИЛИ, представленного на рис. 1. Следовательно, корнем дерева будет являться И-узел, количество потомков которого определяется количеством кодируемых атрибутов, и каждый из них имеет метку $B_i$ ($i$ = 1, ..., $k$). В свою очередь, каждый узел с меткой $B_i$ представляется ИЛИ-узлом, количество потомков (листов) которого определяется количеством уникальных значений соответствующего атрибута, т. е. равно $|B_i|$. На рис. 2 представлен итоговый вид полученной структуры дерева И/ИЛИ для кодируемых атрибутов журнала событий.




Р и с. 2. Структура дерева И/ИЛИ для кодируемых атрибутов журнала событий

F i g. 2. The AND/OR tree structure for encoded event log attributes

Количество вариантов в поддеревьях полученной структуры дерева И/ИЛИ:

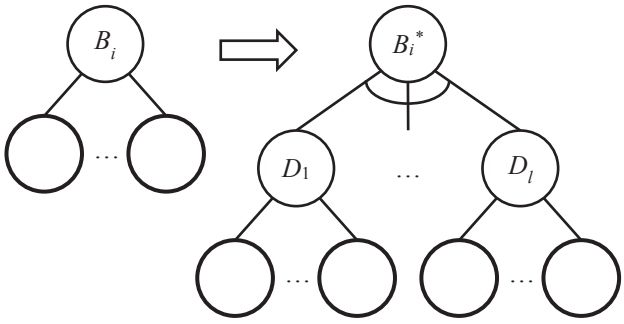
- для конечных узлов (листов): $w(b_{i,j}) = 1$;
- для ИЛИ-узлов: $w(B_i) = w(b_{i,1}) + ... + w(b_{i,mi}) = |B_i| = m_i$;
- для И-узла: $w(\text{Запись}) = w(B_1) \cdot ... \cdot w(B_k) = |B_1| \cdot ... \cdot |B_k| = m_1 \cdot ... \cdot m_k$.

Таким образом, максимальное возможное значение ранга для вариантов дерева И/ИЛИ, представленного на рис. 2, равно $w(\text{Запись}) - 1$ (так как нумерация вариантов начинается со значения 0). Следовательно, для хранения значения ранга для каждого варианта данного дерева И/ИЛИ требуется $\log_2(w(\text{Запись}) - 1)$ бит памяти.

**Шаг 4. Декомпозиция значений атрибутов журнала событий**

В ходе данного шага требуется рассмотреть возможность усложнения построенной структуры дерева И/ИЛИ для кодируемых атрибутов журнала событий за счет декомпозиции значений каких-либо атрибутов с целью получения выигрыша в количестве хранимой информации. В качестве примера рассмотрим следующие две типовые ситуации:

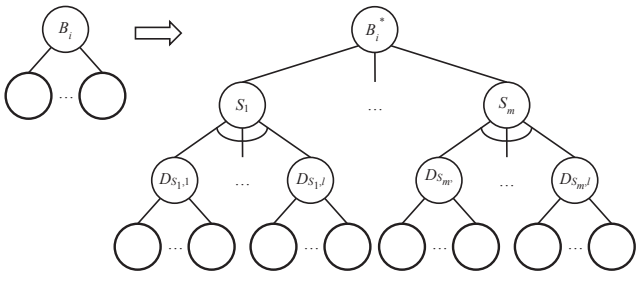
Случай 1. Каждое значение $b_i$ для $i$-го кодируемого атрибута представляет собой конкатенацию значений $d_1 ... d_l$ некоторых других атрибутов, уникальные значения которых определяются множествами $D_1, ..., D_l$, т. е. значению $b_i$ можно сопоставить кортеж $< d_1 ... d_l >$. Тогда узел с меткой $B_i$ преобразуется из ИЛИ-узла в И-узел. Результат данного преобразования представлен на рис. 3.




Р и с. 3. Пример изменения структуры узла для одного из атрибутов (Случай 1)

F i g. 3. An example of changing the structure of the node for one of the attributes (Case 1)

Случай 2. Каждое значение $b_i$ для $i$-го кодируемого атрибута представляет собой один из возможных вариантов конкатенации значений некоторых других атрибутов, т. е. наблюдается ситуация, описанная выше в Случае 1, но допускается несколько шаблонных вариантов кортежей для сопоставления со значением $b_i$. Тогда узел с меткой $B_i$ преобразуется в ИЛИ-узел, количество потомков которого определяется количеством шаблонных вариантов кортежей, и каждый из них имеет метку $S_j$ ($j$ = 1, ..., $m$). При этом каждый узел с меткой $S_j$ представляется И-узлом аналогично представлению узла с меткой $B_i$ в Случае 1. Результат данного преобразования представлен на рис. 4.




Р и с. 4. Пример изменения структуры узла для одного из атрибутов (Случай 2)

F i g. 4. An example of changing the structure of the node for one of the attributes (Case 2)

Так как в результате данных преобразований имевшийся атрибут заменяется на новый составной атрибут, то необходимо заменить файл-справочник уникальных значений данного атрибута на набор файлов-справочников уникальных значений для новых атрибутов. Это приводит к сокращению избыточности информации в имеющихся файлах-справочниках уникальных значений атрибутов, а также к уменьшению объема этих файлов. Однако данные преобразования приводят к увеличению количества вариантов дерева И/ИЛИ и, следовательно, увеличению количества бит памяти, требуемой для хранения значения ранга для каждого варианта дерева И/ИЛИ. Разумность применения декомпозиции значения атрибута журнала событий проверяется путем выполнения следующего неравенства:
где:

$$\left\lceil \log_2\left(w(\text{Запись}^*)-1\right)\right\rceil \cdot N + \sum_i C_i^* < \left\lceil \log_2\left(w(\text{Запись})-1\right)\right\rceil \cdot N + \sum_i C_i,$$

- $N$ – количество строк записей в рассматриваемом журнале событий;
- $w$(Запись) – количество вариантов в исходной версии дерева И/ИЛИ;
- $w$(Запись*) – количество вариантов в преобразованной версии дерева И/ИЛИ;
- $C_i$ – количество бит памяти, требуемой для хранения $i$-го файла-справочника уникальных значений атрибутов в исходной версии дерева И/ИЛИ;
- $C_i^*$ – количество бит памяти, требуемой для хранения $i$-го файла-справочника уникальных значений атрибутов в преобразованной версии дерева И/ИЛИ.

Таким образом, если по итогу изменения структуры дерева И/ИЛИ для хранения сжатого журнала событий и всех требуемых файлов-справочников в сумме необходимо меньше памяти, чем при исходной версии дерева И/ИЛИ, то такое преобразование считается разумным.

**Шаг 5. Нормализация записей журнала событий**

В ходе данного шага требуется представить каждый кортеж подверженных дальнейшему сжатию $k$ атрибутов < $b_1$, ..., $b_k$ > в виде кортежа идентификаторов значений атрибутов, полученных на основе сформированных файлов-справочников. Таким образом, каждому кортежу < $b_1$, ..., $b_k$ > сопоставляется кортеж вида < ID$b_1$, ..., ID$b_k$ >, где ID$b_i$ – это порядковый номер (начиная с 0) выбранного значения $b_i$ среди упорядоченных элементов множества $B_i$. Выполненные преобразования также необходимо учесть в структуре дерева И/ИЛИ, т. е. для каждого ИЛИ-узла с меткой $B_i$ меняются метки потомков (листов): вместо уникального значения атрибута записывается его идентификатор.

**Шаг 6. Ранжирование записей журнала событий**

В ходе данного шага требуется для каждого кортежа идентификаторов значений атрибутов < ID$b_1$, ..., ID$b_k$ > вычислить значение ранга соответствующего варианта дерева И/ИЛИ. Например, если дерево И/ИЛИ имеет структуру, представленную на рис. 2, то ранг варианта дерева И/ИЛИ, которому соответствует кортеж < ID$b_1$, ..., ID$b_k$ >, вычисляется по формуле:

$$\text{rank}(<\text{ID}b_1,\text{ID}b_2,\ldots,\text{ID}b_k>) = \text{ID}b_1 + w(B_1)\cdot\left(\text{ID}b_2 + w(B_2)\cdot\left(\ldots\left(\text{ID}b_{k-1} + w(B_{k-1})\cdot\text{ID}b_k\right)\ldots\right)\right) =$$
$$= \text{ID}b_1 + |B_1|\cdot\left(\text{ID}b_2 + |B_2|\cdot\left(\ldots\left(\text{ID}b_{k-1} + |B_{k-1}|\cdot\text{ID}b_k\right)\ldots\right)\right).$$

Если структура дерева И/ИЛИ была преобразована за счет декомпозиции значений атрибутов, то данные изменения также необходимо учесть в правилах вычисления рангов для вариантов дерева И/ИЛИ.

## 4. Экспериментальное исследование эффективности предложенного метода

Для оценки эффективности предложенного метода рассмотрим пример сжатия данных журналов событий, генерируемых внутри электронных курсов системы Moodle [19]. С использованием данных системы дистанционного обучения Томского государственного университета систем управления и радиоэлектроники[6], был выгружен журнал событий для электронного курса, который:
- содержит информацию о деятельности внутри электронного курса для одного потока студентов в течение одного семестра в рамках обучения по дисциплине «Теория вероятностей и математическая статистика»;
- состоит из строки-заголовка и 116 245 строк записей о журналируемых событиях электронного курса;
- объем файла журнала событий в формате CSV равен 44 324 589 байт, или 43 286 Кб.

Далее подробно представлена реализация основных шагов предложенного метода и полученные результаты для указанного журнала событий электронного курса.

**Шаг 1. Определение кодируемых атрибутов журнала событий**

Первая строка-заголовок рассматриваемого журнала событий показывает название атрибутов, которые содержит каждая запись журнала событий. Далее представлены полные названия каждого атрибута и присвоенные им краткие цифровые коды [20-21]:
- Атрибут 0: «Время» – точные дата и время журналируемого события электронного курса;
- Атрибут 1: «Полное имя пользователя» – фамилия, имя и отчество пользователя, сгенерировавшего событие электронного курса;
- Атрибут 2: «Затронутый пользователь» – фамилия, имя и отчество пользователя, над которым совершено действие в электронном курсе;
- Атрибут 3: «Контекст события» – название элемента электронного курса, с помощью которого совершено действие в электронном курсе;
- Атрибут 4: «Компонент» – тип элемента электронного курса, с помощью которого совершено действие в электронном курсе;
- Атрибут 5: «Название события» – название журналируемого события электронного курса;
- Атрибут 6: «Описание» – подробное описание журналируемого события электронного курса;

[6] Система управления обучением ТУСУР [Электронный ресурс] // ТУСУР, 2023. URL: https://sdo.tusur.ru (дата обращения: 01.09.2023).

- Атрибут 7: «Источник» – тип источника журналируемого события электронного курса;
- Атрибут 8: «IP-адрес» – IP-адрес, с которого совершено действие в электронном курсе.

Каждая запись рассматриваемого журнала событий представляет собой одну текстовую строку, полученную конкатенацией текстовых строк с конкретными значениями Атрибутов 1–8 с использованием разделителя в виде символа запятой.

С целью упрощения дальнейшей первичной обработки сжатых данных журнала событий электронного курса, Атрибут 0 не будет участвовать в формировании структуры дерева И/ИЛИ, т. е. Атрибут 0 будет храниться в исходном виде без какого-либо сжатия. Объем файла журнала событий в формате CSV, в котором от каждой записи оставлена только подстрока с значением Атрибута 0, равен 2 171 581 байт, или 2 121 Кб (4,9 % от объема исходного файла журнала событий). Тогда итоговый вид журнала событий после его сжатия будет представлять собой набор измененных записей, каждая из которых будет являться кортежем, содержащим значение Атрибута 0 и значение ранга варианта дерева И/ИЛИ, соответствующего набору значений Атрибутов 1–8 в исходной неизмененной записи.

**Шаг 2. Составление справочников уникальных значений атрибутов журнала событий**

Исследование содержимого записей рассматриваемого журнала событий показало, что в его Атрибутах 1–8 содержится следующее количество уникальных значений:
- Атрибут 1: 182 уникальных значения;
- Атрибут 2: 208 уникальных значений;
- Атрибут 3: 50 уникальных значений;
- Атрибут 4: 19 уникальных значений;
- Атрибут 5: 87 уникальных значений;
- Атрибут 6: 36 513 уникальных значений;
- Атрибут 7: 4 уникальных значения;
- Атрибут 8: 3 016 уникальных значений.

Так как рассматриваемый журнал событий содержит всего 116 245 строк записей, то из полученной информации о количестве уникальных значений Атрибутов 1–8 очевиден вывод о значительном дублировании хранимых данных. Следовательно, более эффективной с точки зрения объема хранимых данных будет организация хранения записей рассматриваемого журнала событий в виде кортежей идентификаторов уникальных значений атрибутов, а сами уникальные значения атрибутов будут храниться в виде отдельных справочников. При этом каждый такой справочник представляет собой текстовый файл, каждая строка которого является текстовой строкой с одним из уникальных значений атрибута, а порядковый номер строки является ее идентификатором.

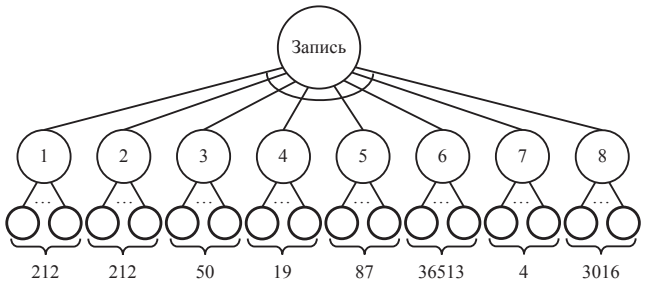
Также, учитывая тот факт, что Атрибуты 1–2 охватывают практически одно и то же множество пользователей, было реализовано объединение множеств уникальных значений данных атрибутов. В результате было получено одно множество, состоящее из 212 уникальных значений Атрибутов 1–2.

Объем файлов сформированных справочников в формате CSV равен:
- Атрибуты 1–2: 10 768 байт;
- Атрибут 3: 2 703 байт;
- Атрибут 4: 516 байт;
- Атрибут 5: 4 574 байт;
- Атрибут 6: 4 553 086 байт;
- Атрибут 7: 26 байт;
- Атрибут 8: 46 456 байт.

Общий объем файлов справочников равен 4 618 129 байт, или 4 510 Кб (10,4 % от объема исходного файла журнала событий).

**Шаг 3. Построение структуры дерева и/или для кодируемых атрибутов журнала событий**

Основываясь на выбранных для кодирования Атрибутах 1–8, а также основываясь на информации о количестве уникальных значений данных атрибутов, строится структура дерева И/ИЛИ, представленная на рис. 5.




Р и с. 5. Структура дерева И/ИЛИ для атрибутов рассматриваемого журнала событий

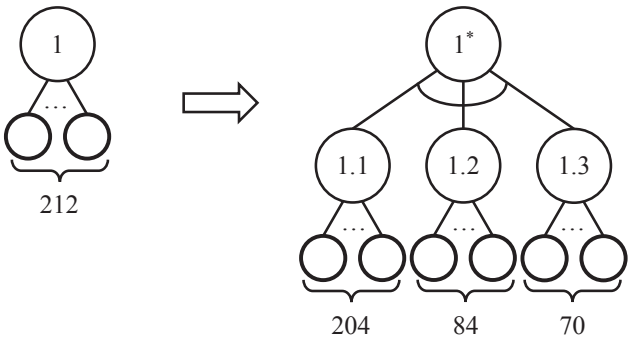
F i g. 5. The AND/OR tree structure for the attributes of the considered course logs

Построенная структура дерева И/ИЛИ представляет собой корневой узел, помеченный текстом «Запись» и являющийся И-узлом. Данный И-узел содержит 8 потомков, помеченных числами от 1 до 8, что соответствует цифровым кодам Атрибутов 1–8. Каждый такой узел является ИЛИ-узлом и в качестве потомков содержит листы, количество которых определяется количеством уникальных значений соответствующего атрибута среди всех записей рассматриваемого журнала событий. Таким образом, каждая запись журнала событий (точнее, ее часть, являющаяся подстрокой с значениями Атрибутов 1–8) может быть представлена одним вариантом построенной структуры дерева И/ИЛИ.

**Шаг 4. Декомпозиция значений атрибутов журнала событий**

Одним из возможных вариантов декомпозиции значений атрибутов рассматриваемого журнала событий является представление Атрибута 1 в виде комбинации следующих трех атрибутов:
- Атрибут 1.1: «Фамилия пользователя»;
- Атрибут 1.2: «Имя пользователя»;
- Атрибут 1.3: «Отчество пользователя».

Изменение структуры соответствующего ИЛИ-узла дерева И/ИЛИ представлена на рис. 6.




Р и с. 6. Изменение структуры ИЛИ-узла для Атрибута 1
F i g. 6. Changing the structure of the OR node for Attribute 1

В таком случае будут получены следующие новые справочники:
- Атрибут 1.1: 204 уникальных значения, объем файла равен 3 414 байт;
- Атрибут 1.2: 84 уникальных значений, объем файла равен 1 158 байт;
- Атрибут 1.3: 70 уникальных значений, объем файла равен 1 495 байт.

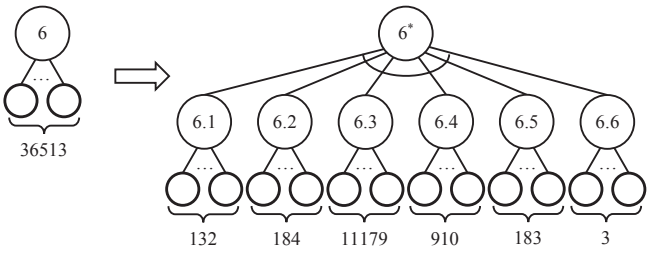
Общий объем полученных файлов справочников равен 6 067 байт, что составляет 56,3 % от объема исходного файла справочника для Атрибута 1, т. е. меньше на 10 768 – 6 067 = 4 701 байт. При этом изменение структуры ИЛИ-узла приводит к увеличению количества вариантов в поддереве данного узла:
- поддерево узла, помеченного «1», имеет 212 вариантов, т. е. максимальное значение ранга варианта равно 211, что требует 8 бит для хранения;
- поддерево узла, помеченного «1*», имеет 204 · 84 · 70 = 1 199 520 вариантов, т. е. для хранения максимального значения ранга варианта требуется 21 бит.

Следовательно, на каждую запись журнала событий дополнительно потребуется 21 – 8 = 13 бит для хранения, т. е. для всех имеющихся записей понадобится дополнительно 116 245 · 13 = 1 511 185 бит, или 188 898,125 байт. В результате получаем, что предложенное изменение структуры ИЛИ-узла приводит к увеличению объема хранимых данных примерно на 180 Кб (188 898,125 – 4 701 = 184 197,125 байт). Кроме того, аналогичные преобразования нужно выполнить и с ИЛИ-узлом для Атрибута 2, так как для них используется общий справочник, что также приведет к увеличению объема хранимых данных.

Аналогичный отрицательный результат будет получен, если Атрибут 8, значения которого содержат информацию об IP-адресах, представить в виде комбинации четырех значений составляющих его байтов. В таком случае ИЛИ-узел для Атрибута 8 станет И-узлом с четырьмя потомками, каждый из которых содержит 256 вариантов. Это приведет к увеличению общего количества вариантов с 3016 до $256^4$ = 4 294 967 296 и, соответственно, к увеличению объема хранимых данных для хранения рангов вариантов дерева И/ИЛИ. Однако можно оптимизировать объем файла справочника для Атрибута 8 за счет хранения IP-адресов не в виде текстового файла, содержащего 3016 строк общим объемом 46 456 байт, а в виде бинарного файла, описывающего каждый IP-адрес четырьмя байтами, т. е. файл общим объемом 3 016 · 4 = 12 064 байт, что составляет 26 % от объема исходного файла справочника для Атрибута 8.

Наибольшим по количеству уникальных значений и по объему файла справочника является Атрибут 6. Анализ значений данного атрибута показал, что в нем содержатся шаблонные предложения с подробным описанием журналируемых событий электронного курса, в которые встроены идентификаторы затронутых элементов электронного курса[7] [22]. Если не учитывать значения идентификаторов, то все 36 513 уникальных значений Атрибута 6 могут быть описаны всего лишь с использованием 132 шаблонов. Каждый такой шаблон содержит до пяти идентификаторов затронутых элементов электронного курса, при этом количество уникальных значений самих идентификаторов равно 183 для первого по счету идентификатора, 11 178 для второго, 909 для третьего, 182 для четвертого и 2 для пятого. Так как шаблоны могут содержать разное количество идентификаторов, то дополнительно введем пустой идентификатор, выбор которого будет означать выбор шаблона с меньшим количеством идентификаторов. Таким образом, можно предложить декомпозицию значений Атрибута 6 в виде комбинации следующих атрибутов:
- Атрибут 6.1: «Шаблон»;
- Атрибуты 6.2-6.6: «Идентификатор».

Изменение структуры соответствующего ИЛИ-узла дерева И/ИЛИ представлена на рис. 7.




Р и с. 7. Изменение структуры ИЛИ-узла для Атрибута 6
F i g. 7. Changing the structure of the OR node for Attribute 6

В таком случае будут получены следующие новые справочники:
- Атрибут 6.1: 132 уникальных значения, объем файла равен 9 806 байт;
- Атрибут 6.2: 184 уникальных значения, объем файла равен 1 263 байт;
- Атрибут 6.3: 11 179 уникальных значений, объем файла равен 100 486 байт;
- Атрибут 6.4: 910 уникальных значений, объем файла равен 6 997 байт;
- Атрибут 6.5: 183 уникальных значения, объем файла равен 4 039 байт;

[7] Репьюк Н. С., Кручинин Д. В. Программное обеспечение для декомпозиции значений атрибута «Описание» журнала событий Moodle // Перспективы развития фундаментальных наук : сб. науч. тр. XX Международной конференции студентов, аспирантов и молодых ученых. Т. 7. Томск : ТПУ, 2023. С. 124-126. EDN: GRNDME

- Атрибут 6.6: 3 уникальных значения, объем файла равен 41 байт.

Общий объем полученных файлов справочников равен 122 632 байт, что составляет 2,7 % от объема исходного файла справочника для Атрибута 6, т. е. меньше на 4 553 086 – 122 632 = 4 430 454 байт. При этом изменение структуры ИЛИ-узла приводит к увеличению количества вариантов в поддереве данного узла:

- поддерево узла, помеченного «6», имеет 36 513 вариантов, т. е. максимальное значение ранга варианта равно 36 512, что требует 16 бит для хранения;
- поддерево узла, помеченного «6*», имеет $132 \cdot 184 \cdot 11\,179 \cdot 910 \cdot 183 \cdot 3 \approx 1{,}4 \cdot 10^{14}$ вариантов, т. е. для хранения максимального значения ранга варианта требуется 47 бит.

Следовательно, на каждую запись журнала событий дополнительно потребуется 47 – 16 = 31 бит для хранения, т. е. для всех имеющихся записей понадобится дополнительно $116\,245 \cdot 31 = 3\,603\,595$ бит, или 450 449,375 байт. В результате получаем, что предложенное изменение структуры ИЛИ-узла приводит к уменьшению объема хранимых данных примерно на 3 887 Кб (4 430 454 – 450 449,375 = 3 980 004,625 байт).

Таким образом, положительный эффект с точки зрения сжатия данных рассматриваемого журнала событий показывает только декомпозиция Атрибута 6. Дополнительное улучшение данного эффекта может быть достигнуто за счет более детальной декомпозиции шаблонов по количеству используемых идентификаторов.

**Шаг 5. Нормализация записей журнала событий**

Следующим этапом необходимо представить каждую запись рассматриваемого журнала событий в виде кортежа идентификаторов значений атрибутов, полученных на основе сформированных после декомпозиции справочников. В таблице 1 представлены требования к объему памяти, необходимому для хранения идентификаторов значений атрибутов. Таким образом, для хранения каждой нормализованной записи требуется в сумме 98 бит для Атрибутов 1–8. Тогда для всех имеющихся записей понадобится $116\,245 \cdot 98 = 11\,392\,010$ бит, или 1 424 001,25 байт (примерно 1 391 Кб).

Таблица 1. **Требования к объему памяти, необходимому для хранения идентификаторов**
Table 1. **Requirements for the memory required to store identifiers**

| Атрибут | Количество уникальных значений | Значения идентификатора | Количество бит для хранения идентификатора | Количество байт для хранения файла справочника |
|---|---|---|---|---|
| Атрибут 1 | 212 | 0, ..., 211 | 8 | 10 768 |
| Атрибут 2 | 212 | 0, ..., 211 | 8 | |
| Атрибут 3 | 50 | 0, ..., 49 | 6 | 2 703 |
| Атрибут 4 | 19 | 0, ..., 18 | 5 | 516 |
| Атрибут 5 | 87 | 0, ..., 86 | 7 | 4 574 |
| Атрибут 6.1 | 132 | 0, ..., 131 | 8 | 9 806 |
| Атрибут 6.2 | 184 | 0, ..., 183 | 8 | 1 263 |
| Атрибут 6.3 | 11 179 | 0, ..., 11 178 | 14 | 100 486 |
| Атрибут 6.4 | 910 | 0, ..., 909 | 10 | 6 997 |
| Атрибут 6.5 | 183 | 0, ..., 182 | 8 | 4 039 |
| Атрибут 6.6 | 3 | 0, ..., 2 | 2 | 41 |
| Атрибут 7 | 4 | 0, ..., 3 | 2 | 26 |
| Атрибут 8 | 3 016 | 0, ..., 3 015 | 12 | 46 456 |
| Сумма: | | | 98 | 187 675 |

*Источник*: составлено автором.
*Source*: Compiled by the author.

Если ограничиться текущим этапом, то итоговый суммарный объем сжатого путем нормализации записей журнала событий будет состоять из:

- общего объема файлов справочников, равного 187 675 байт;
- объема текстового файла, равного 2 171 581 байт, с значениями Атрибута 0 для каждой записи исходного журнала событий;
- объема бинарного файла, равного 1 424 001,25 байт, с набором идентификаторов значений Атрибутов 1–8 для каждой записи исходного журнала событий.

Суммарно получаем 3 783 257,25 байт или примерно 3 695 Кб, что составляет 8,54 % от объема исходного файла журнала событий.

**Шаг 6. Ранжирование записей журнала событий**

Последним этапом необходимо для каждой записи рассматриваемого журнала событий, представленной в виде кортежа идентификаторов значений атрибутов, вычислить значение ранга соответствующего варианта дерева И/ИЛИ. Построенная структура дерева И/ИЛИ для атрибутов рассматриваемого журнала событий, представленная на рис. 5, с учетом внесенных изменений структуры ИЛИ-узла для Атрибута 6, представленных на рис. 7, содержит в себе следующее количество вариантов:

$212 \cdot 212 \cdot 50 \cdot 19 \cdot 87 \cdot (132 \cdot 184 \cdot 11\,179 \cdot 910 \cdot 183 \cdot 3) \cdot 4 \cdot 3\,016 \approx 6{,}1 \cdot 10^{27}$.

Таким образом, для хранения каждой ранжированной записи требуется 93 бита для Атрибутов 1–8. Тогда для всех имеющихся записей понадобится 116 245 · 93 = 10 810 785 бит, или 1 351 348,125 байт (примерно 1 320 Кб), что на 5,1 % меньше по сравнению с требуемым объемом памяти для хранения аналогичной информации в формате нормализованных записей. Итоговый суммарный объем сжатого путем ранжирования записей журнала событий будет состоять из:
- общего объема файлов справочников, равного 187 675 байт;
- объема текстового файла, равного 2 171 581 байт, с значениями Атрибута 0 для каждой записи исходного журнала событий;
- объема бинарного файла, равного 1 351 348,125 байт, с рангом соответствующего значениям Атрибутов 1–8 варианта дерева И/ИЛИ для каждой записи исходного журнала событий.
Суммарно получаем 3 710 604,125 байт, или примерно 3 624 Кб, что составляет 8,37 % от объема исходного файла журнала событий.

## 5. Заключение

Фиксирование некоторого текущего состояния журнала событий позволяет рассматривать множество его записей в качестве комбинаторного множества. В свою очередь, применение алгоритмов ранжирования и генерации по рангу элементов комбинаторных множеств делает возможным кодирование записей журнала событий в числа (ранги), для хранения которых требуется меньше памяти. Таким образом, в качестве основного результата исследования предложен метод сжатия данных журналов событий на основе теории комбинаторной генерации с применением структур деревьев И/ИЛИ.

Рассмотренный вариант апробации предложенного метода на примере сжатия данных журналов событий, генерируемых внутри электронных курсов системы Moodle, подтвердил его эффективность. Результатом стало сжатие исходного файла журнала событий с 43 286 Кб до 3 624 Кб (уменьшение на 91,63 %). На сегодняшний день существует большое количество готовых программных продуктов, специализирующихся на сжатии файлов без потерь [23-25]. Например, популярны следующие форматы сжатых файлов: ZIP, RAR, 7z. Если сжать исходный файл журнала событий с помощью соответствующих программ-архиваторов с максимальным уровнем сжатия, то получим сжатие до 1 798 Кб в формате ZIP (уменьшение на 95,85 %), до 786 Кб в формате RAR (уменьшение на 98,18 %) и до 893 Кб в формате 7z (уменьшение на 97,94 %). Данные результаты являются лучшими по сравнению с результатами апробации предложенного метода. Однако необходимо учесть, что файлы, которые были получены в ходе применения предложенного метода, также могут быть дополнительно сжаты с помощью соответствующих программ-архиваторов. Совокупный результат показал сжатие до 702 Кб в формате ZIP (уменьшение на 98,38 %), до 650 Кб в формате RAR (уменьшение на 98,50 %) и до 556 Кб в формате 7z (уменьшение на 98,72 %). Таким образом, получаем сжатие без использования предложенного метода в лучшем случае до 786 Кб против сжатия в лучшем случае до 556 Кб с его использованием (уменьшение на 29,2 6%). Следовательно, суммарный объем памяти, требуемой для хранения журнала события электронного курса системы Moodle в сжатом с помощью предложенного метода виде, имеет меньшее значение по сравнению с существующими методами сжатия текстовых файлов.

## References

[1] Larin M.V., Surovtseva N.G. Some theoretical issues of archival storage of electronic documents. *Herald of an archivist*. 2019;(3):809-824. (In Russ., abstract in Eng.) https://doi.org/10.28995/2073-0101-2019-3-809-824
[2] Memishi B., Appuswamy R., Paradies M. Cold storage data archives: More than just a bunch of tapes. In: Proceedings of the 15th International Workshop on Data Management on New Hardware (DaMoN'19). New York, NY, USA: Association for Computing Machinery; 2019. Article number: 1. https://doi.org/10.1145/3329785.3329921
[3] Pernet C., Svarer C., Blair R., van Horn J.D., Poldrack R.A. On the long-term archiving of research data. *Neuroinformatics*. 2023;21:243-246. https://doi.org/10.1007/s12021-023-09621-x
[4] Liu A., Yu T. Overview of Cloud Storage And Architecture. *International Journal of Scientific & Technology Research*. Available at: https://ssrn.com/abstract=3649074 (accessed 01.09.2023).
[5] Jayasankar U., Thirumal V., Ponnurangam D. A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. *Journal of King Saud University – Computer and Information Sciences*. 2021;33(2):119-140. https://doi.org/10.1016/j.jksuci.2018.05.006
[6] Gupta A., Bansal A., Khanduja V. Modern lossless compression techniques: Review, comparison and analysis. In: 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT). Coimbatore, India: IEEE Computer Society; 2017. p. 1-8. https://doi.org/10.1109/ICECCT.2017.8117850
[7] Bakulina M. Efficient lossless compression of large information arrays. *Problems of Informatics*. 2022;(4):63-69. (In Russ., abstract in Eng.) https://doi.org/10.24412/2073-0667-2022-4-63-69
[8] Ko J., Comuzzi M. A Systematic Review of Anomaly Detection for Business Process Event Logs. *Business & Information Systems Engineering*. 2023;65(7):441-462. https://doi.org/10.1007/s12599-023-00794-y
[9] Yao K., Sayagh M., Shang W., Hassan A.E. Improving State-of-the-Art Compression Techniques for Log Management Tools. *IEEE Transactions on Software Engineering*. 2022;48(8):2748-2760. https://doi.org/10.1109/TSE.2021.3069958
[10] Balakrishnan R., Sahoo R. Lossless compression for large scale cluster logs. In: Proceedings 20th IEEE International Parallel & Distributed Processing Symposium. Rhodes, Greece: IEEE Computer Society; 2006. p. 7. https://doi.org/10.1109/IPDPS.2006.1639692

[11] Grebennik I.V., Lytvynenko O.S. Generating combinatorial sets with given properties. *Cybernetics and Systems Analysis*. 2012;48:890-898. https://doi.org/10.1007/s10559-012-9469-9
[12] Hartung E., Hoang H.P., Mutze T., Williams A. Combinatorial generation via permutation languages. I. Fundamentals. *Transactions of the American Mathematical Society*. 2020;375:2255-2291. https://doi.org/10.1090/tran/8199
[13] Kruchinin V.V., Titkov A.V., Khomich S.L. Approach to development of database based on the generation algorithms and tuple identification. *Bulletin of Tomsk Polytechnic University*. 2006;309(8):28-31. (In Russ., abstract in Eng.) EDN: HYZVQV
[14] Shablya Y., Kruchinin D., Kruchinin V. Method for developing combinatorial generation algorithms based on AND/OR trees and its application. *Mathematics*. 2020;8(6):962. https://doi.org/10.3390/math8060962
[15] Shablya Y.V., Kruchinin D.V. Modification of the algorithm development method for combinatorial generation based on the application of the generating functions theory. *Proceedings of TUSUR University*. 2019;22(3):55-60. (In Russ., abstract in Eng.) https://doi.org/10.21293/1818-0442-2019-22-3-55-60
[16] Kruchinin D.V. Modification of the method for developing combinatorial generation algorithms based on the use of multivariate generating functions and approximations. *Proceedings of TUSUR University*. 2022;25(1):55-60. (In Russ., abstract in Eng.) https://doi.org/10.21293/1818-0442-2021-25-1-55-60
[17] Kruchinin V.V. Presentation of set by means of tree AND/OR. *Proceedings of TUSUR University*. 2008;(1):107-112. (In Russ., abstract in Eng.) EDN: KUUJLT
[18] Kruchinin V. V., Lukschin B. A. Method of coding of information objects on the basis of trees And-Or. *Proceedings of TUSUR University*. 2010;(1):170-172. (In Russ., abstract in Eng.) EDN: MPWDAR
[19] Bojiah J. Effectiveness of Moodle in teaching and learning. *Journal of Hunan University Natural Sciences*. 2022;49(12):320-328. https://doi.org/10.55463/issn.1674-2974.49.12.33
[20] Parise P. A preliminary look at online learner behavior what can the Moodle logs tell us? *Bulletin of Kanagawa Prefectural Institute of Language and Culture Studies*. 2017;6:15-31. https://doi.org/10.20686/academiakiyou.6.0_15
[21] Rotelli D., Monreale A. Processing and understanding Moodle log data and their temporal dimension. *Journal of Learning Analytics*. 2023;10(2):126-141. https://doi.org/10.18608/jla.2023.7867
[22] Athaya H., Nadir R.D.A., Indra Sensuse D., Kautsarina K., Suryono R.R. Moodle Implementation for E-Learning: A Systematic Review. In: Proceedings of the 6th International Conference on Sustainable Information Engineering and Technology (SIET '21). New York, NY, USA: Association for Computing Machinery; 2021. p. 106-112. https://doi.org/10.1145/3479645.3479646
[23] Jacob N., Somvanshi P., Tornekar R. Comparative analysis of lossless text compression techniques. *International Journal of Computer Applications*. 2012;56(3):17-21. https://doi.org/10.5120/8871-2850
[24] Tanjung A. S., Nasution S. D. Comparison analysis with Huffman algorithm and Goldbach codes algorithm in file compression text using the method exponential comparison. *International Journal of Informatics and Computer Science*. 2020;4(1):29-34. http://dx.doi.org/10.30865/ijics.v4i1.1387
[25] Kotb A., Hassan S., Hassan H. A Comparative Study Among Various Algorithms for Lossless Airborne LiDAR Data Compression. In: 2018 14th International Computer Engineering Conference (ICENCO). Cairo, Egypt: IEEE Computer Society; 2018. p. 17-21. https://doi.org/10.1109/ICENCO.2018.8636136

*Поступила 01.09.2023; одобрена после рецензирования 04.10.2023; принята к публикации 08.10.2023.*
*Submitted 01.09.2023; approved after reviewing 04.10.2023; accepted for publication 08.10.2023.*

Об авторе:

**Шабля Юрий Васильевич**, старший научный сотрудник лаборатории алгоритмов и технологий исследования дискретных структур, ФГБОУ ВО «Томский государственный университет систем управления и радиоэлектроники» (634050, Российская Федерация, г. Томск, пр. Ленина, д. 40), кандидат технических наук, **ORCID: https://orcid.org/0000-0002-9695-7493**, shablya-yv@mail.ru

*Автор прочитал и одобрил окончательный вариант рукописи.*

About the author:

**Yuriy V. Shablya**, Senior Researcher of the Laboratory of Algorithms and Technologies for Discrete Structures Research, Tomsk State University of Control Systems and Radioelectronics (40 Lenin Ave., Tomsk 634050, Russian Federation), Cand. Sci. (Eng.), **ORCID: https://orcid.org/0000-0002-9695-7493**, shablya-yv@mail.ru

*The author has read and approved the final manuscript.*