УДК 303.717

СРАВНЕНИЕ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ СЖАТИЯ LZ77 И LZ78

В. В. Кудинов, О. К. Сапрыкин

Рассматриваются программная реализация и исследование основных принципов работы словарных методов сжатия семейства LZ Лемпеля – Зива, сравнение эффективности этих методов. Разработанные математические модели словарных методов LZ позволяют продемонстрировать технологию формирования словарей, а также оценить эффективность методов сжатия. Описываются базовые алгоритмы сжатия семейства LZ – LZ77, LZ78.
***Ключевые слова**: информация, сообщение, сжатие информации, эффективное кодирование, метод Лемпеля – Зива, LZ77, LZ78.*

Введение

За последние двадцать лет наблюдается стремительный рост использования компьютеров, сопровождаемый постоянным увеличением их возможностей. Это привело к значительному увеличению пользователей персональных компьютеров и области их применения. Развитие технологий привело к увеличению объема и качества данных. Для сокращения размера файлов были разработаны алгоритмы сжатия данных, которые могут уменьшить размер файлов как незначительно, так и значительно.

Существует множество различных алгоритмов сжатия данных, которые продолжают активное развитие. Самые популярные алгоритмы сжатия – код Шеннона – Фано, код Хаффмана, код Лемпеля – Зива и алгоритм RLE.

Важно отметить, алгоритмы Лемпеля – Зива, которые были разработаны в конце 70-х годов XX века, и до сих пор широко используются в различных системах. Следует подчеркнуть, что эти алгоритмы обеспечивают сжатие без потерь, что очень важно для работы с цифровыми данными. Самыми базовыми алгоритмами являются LZ77, LZ78.

Алгоритм LZW является дальнейшим развитием LZ78 и использует более эффективный алгоритм вставки новых кодов в словарь. LZW широко используется для сжатия данных в форматах GIF и TIFF. Алгоритмы Лемпеля – Зива являются одними из самых распространенных методов сжатия данных и используются во многих современных компьютерных программных приложениях [1].

Виды алгоритмов сжатия информации

Сжатие информации – проблема актуальная в наше время, она тесно связана с развитием проблемы кодирования и шифровки информации. Целью процесса сжатия является получение компактной выходной последовательности из изначального сообщения.

Существующие методы разделены на две основные группы:

Методы сжатия без потерь – алгоритмы, при использовании которых зашифрованная информация может быть полностью восстановлена.

Методы сжатия с потерями – алгоритмы, при использовании которых распакованные данные отличаются от исходных.

Существует несколько подходов к проблеме сжатия информации без потерь. Некоторые из них имеют сложную математическую базу, в то время как другие основаны на свойствах информационного потока и достаточно просты в алгоритмическом плане.

Главным недостатком методов Шеннона – Фано и Хаффмана является необходимость в предварительной информации о распределении символов в последовательности, алгоритм RLE неэффективен в условиях, где нет часто повторяющихся символов.

В кодах семейства Лемпеля – Зива такие недостатки отсутствуют. В 1977 году Авраам Лемпель и Яков Зив представили метод сжатия информации, позже названный LZ77. Этот метод применяется в программных продуктах для упаковки текстов, таких как Compress, LHA, PKZIP и ARJ. Измененная версия метода LZ78 используется для упаковки двоичных данных. Алгоритм предполагает кодирование последовательности бит путем разбивки ее на фразы с последующим кодированием этих фраз [2].

Словарные методы сжатия данных без потерь

Методы, основанные на словарном подходе, не рассматривают статистические модели, они также не используют коды переменной длины. Идея словарных методов заключается в том, что входную последовательность символов рассматривают как последовательность строк, содержащих произвольное количество символов, строки символов заменяются на такие коды, что их можно трактовать как индексы строк некоторого словаря.

Словарь может быть статическим или динамическим (адаптивным). Первый является постоянным; иногда в него добавляют новые последовательности, но никогда не удаляют. Динамический словарь содержит последовательности, ранее поступившие из входного файла, при этом разрешается и добавление, и удаление данных из словаря по мере чтения входного файла.

Образующие словарь строки будем далее называть

фразами. При декодировании осуществляется обратная замена индекса на соответствующую ему фразу словаря. Можно сказать, что исходная последовательность преобразуется таким путем, что она представляется в таком алфавите, что его «буквы» являются фразами словаря, состоящими в общем случае из произвольного количества символов входной последовательности.

Уменьшение размера возможно в первую очередь за счет того, что обычно в сжимаемых данных встречается лишь малая толика всех возможных строк длины n, поэтому для представления индекса фразы требуется, как правило, меньшее число битов, чем для представления исходной строки [3].

В статье рассматриваются алгоритмы Лемпеля – Зива LZ77 и LZ78.

Алгоритм LZ77 основывается на идее замены повторяющихся последовательностей данных ссылками на предыдущие вхождения, что позволяет эффективно уменьшать объем хранимой информации.

Основные компоненты алгоритма:

1. Словарь: в LZ77 используется скользящее окно (или буфер), которое хранит часть уже обработанных данных. Это окно делится на две части: часть, которая содержит уже обработанные данные (источник), и часть, которая содержит еще не обработанные данные.

2. Кодирование: алгоритм ищет в словаре совпадения с текущей последовательностью символов из входных данных. Если совпадение найдено, оно заменяется на пару:

– (сдвиг, длина): где «сдвиг» указывает, насколько далеко назад нужно вернуться для нахождения совпадения, а «длина» – длина совпадающей последовательности;

– если совпадений нет, символ просто добавляется в выходной поток.

3. Выходные данные: результатом работы алгоритма является последовательность пар (сдвиг, длина) и оставшихся символов, которые не были сжаты.

Одной из основных проблем данного алгоритма является неограниченный размер словаря. Поэтому было предложено множество вариантов, включая установку границ словаря, например, запрет на добавление новых слов, если достигнут лимит словаря, или удаление наименее используемых слов [4].

Для исключения недостатков данного алгоритма во многих архиваторах он используется в комбинации с другими алгоритмами сжатия. Например, LZ77 используют в таких системах сжатия, как Deflate, gzip, zip и других [5].

Алгоритм LZ78 не использует скользящее окно и не помещает в словарь все встречаемые при кодировании строки, а лишь «перспективные» с точки зрения вероятности последующего использования. На каждом шаге в словарь добавляется новая фраза, которая представляет собой соединение одной из фраз словаря, имеющей самое длинное совпадение со строкой буфера, и текущего символа, для удобства обозначим фразу и текущий символ как S и s соответственно.

Алгоритм LZ78 применяется не только в области сжатия текстовых данных, но и в сфере сжатия изображений. Этот инновационный подход позволяет заменить повторяющиеся участки в изображении ссылками на уже сжатые фрагменты, что в свою очередь приводит к уменьшению размера файла и снижению требований к ресурсам при его отображении.

Благодаря своей простоте и одновременно высокой эффективности алгоритм LZ78 активно внедряется в современные архиваторы и сжатые файловые системы. Он используется для сжатия архивов, упаковки файлов и хранения их на диске. Этот мощный инструмент с успехом применяется в различных областях, где требуется эффективное сжатие данных без потери их качества.

Кроме того, алгоритм LZ78 находит применение в сжатии видеоконтента. Сжатие видеоданных является одной из ключевых технологий в современных системах мультимедиа. Алгоритм LZ78 позволяет убрать из видеокадров повторяющуюся информацию, что существенно сокращает размер видеофайла и повышает качество воспроизведения [6].

Алгоритм LZ78 создает и использует словарь как при кодировании, так и при декодировании. Изначально предполагается, что словарь пуст, и он заполняется в процессе кодирования.

Первая запись в словаре – это первый символ кодируемой последовательности. На каждой итерации кодируется префикс, который увеличивается, пока он находится в словаре. Кодовые слова состоят из номера самого длинного найденного префикса и следующего символа.

После кодирования эта пара добавляется в словарь, и алгоритм продолжает работу с последующим символом. В более поздних версиях алгоритма предлагается инициализировать словарь, заполняя его заранее известной информацией. Например, все символы алфавита входного потока могут быть добавлены в словарь [7].

В отличие от LZ77 в словаре не может быть одинаковых фраз. Кодер порождает только последовательность кодов фраз. Каждый код состоит из номера «родительской» фразы S и символа s. В начале обработки словарь пуст. Далее, теоретически, словарь может расти бесконечно, то есть на рост самого словаря алгоритм не налагает дополнительных ограничений.

Проблемы использования этого алгоритма для сжатия измерительных данных похожи на алгоритм LZ77, но немного смягчаются благодаря более оптимальному методу кодирования [8].

Программная реализация алгоритмов LZ77 и LZ78

Чтобы исследовать алгоритмы, создадим их математические модели, проведем верификацию моделей и проведем численные эксперименты. Для данной реализации используем язык программирования C++.

Для начала рассмотрим алгоритм LZ77:

Первый шаг – создадим файл, в который будем записывать входную последовательность.

Второй шаг – программная реализация процесса компрессии. В цикле ищем совпадения в окне заданного размера, выделяем подстроки для сравнения и окно для поиска, запоминаем длину совпадения и смещения. Если совпадение найдено, запишем сжатую последовательность в выходной вектор и перейдем к следующему символу, если совпадение не найдено, то запишем символ как отдельный блок в виде «(0, 0, символ)».

Третий шаг – запишем последовательность в выходной файл.

На рис. 1 представлена блок-схема алгоритма LZ77.

Теперь рассмотрим алгоритм LZ78:

Первый шаг – создадим файл, в который будем записывать входную последовательность.

Второй шаг – программная реализация процесса компрессии. Создадим переменную для поиска в тексте символа или группы символов, которые уже имеются в библиотеке. Если совпадений с библиотекой не найдено, выведем пару в виде «(0,текущий символ)» и добавим новую запись в словарь, в обратном же случае выведем последовательность в виде «(индекс подстроки, следующий символ)» и добавим новую запись.

Третий шаг – запишем последовательность в выходной файл.

На рис. 2 представлена блок-схема алгоритма LZ78.

Результат работы алгоритмов

Передавать будем несколько последовательностей разной длины и сравним объем передаваемого сообщения до и после работы алгоритмов.

Текст передаваемого сообщения № 1: «ABACA BACABADACA». Длина кода: 15 · 8 = 120 бит. Результаты работы алгоритмов представлены в табл. 1.

Текст передаваемого сообщения № 2: «МАМА МЫЛА РАМУ МЫЛОМ». Длина кода: 20 · 8 = 160 бит. Результаты работы алгоритмов представлены в табл. 2.

Текст передаваемого сообщения № 3: «Однажды в сказке, жил-был принц, который хотел найти свою принцессу. Он отправился в долгое путешествие, чтобы найти свою любовь. Но на пути его ожидали много препятствий и опасностей. Он встретил зверей и птиц, которые помогали ему или мешали. Но он не сдавался и продолжал свой поиск». Длина кода: 286 · 8 = 2288 бит. Результаты работы алгоритмов представлены в табл. 3.

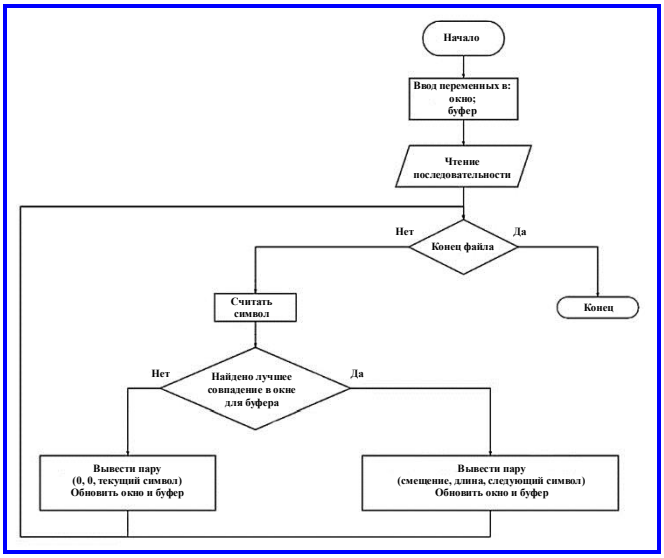

Рис. 1. Блок-схема алгоритма LZ77

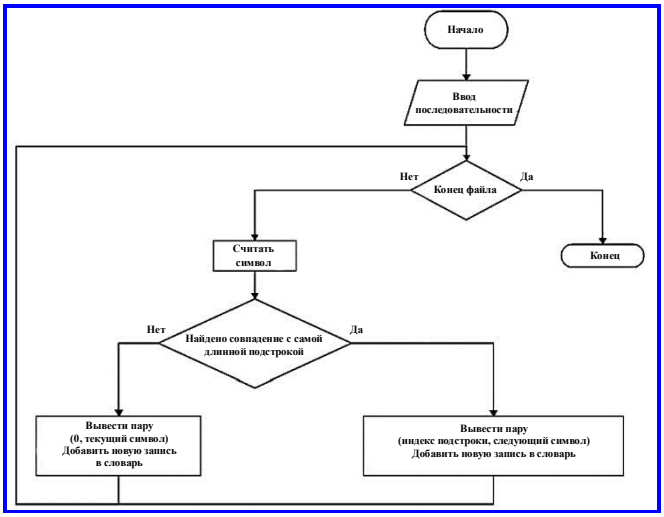

Рис. 2. Блок-схема алгоритма LZ78

Таблица 1

Алгоритм	LZ77 при оптимальных значениях: окно = 12 буфер = 4	LZ78
Результат работы	(0, 0, A) (0, 0, B) (2, 1, C) (4, 4, A) (4, 2, D) (6, 3, ∅)	(0, A) (0, B) (1, C) (1, B) (3, A) (2, A) (0, D) (5, ∅)
Длина закодированного сообщения	18 · 8 = 144 бит	16 · 8 = 128 бит

Таблица 2

Алгоритм	**LZ77 при оптимальных значениях: окно = 11 буфер = 2**	**LZ78**
Результат работы	(0, 0, М) (0, 0, А) (2, 2, _) (3, 1, Ы) (0, 0, Л) (5, 2, Р) (10, 2, У)	(0, М) (0, А) (1, А) (0, _) (1, Ы) (0, Л) (2, _) (0, Р) (2, М) (0, У) (4, М) (0, Ы) (6, О) (1, ∅)
Длина закодированного сообщения	22 · 8 = 176 бит	28 · 8 = 224 бит

Таблица 3

Алгоритм	**LZ77 при оптимальных значениях: окно = размер входной строки () буфер = размер входной строки ()**	**LZ78**
Результат работы	(0, 0, О) (0, 0, д) (0, 0, н) (0, 0, а) (0, 0, ж) (4, 1, ы) (0, 0,) (0, 0, в) (2, 1, с) (0, 0, к) (9, 1, з) (3, 1, е) (0, 0, ,) (8, 1, ж) (0, 0, и) (0, 0, л) (0, 0, -) (0, 0, б) (17, 1, л) (8, 1, п) (0, 0, р) (9, 1, н) (0, 0, ц) (15, 2, к) (0, 0, о) (0, 0, т) (2, 1, р) (15, 1, й) (8, 1, х) (8, 2, е) (21, 2, н) (36, 1, й) (7, 1, и) (43, 2, в) (13, 1, ю) (32, 6, е) (11, 1, с) (0, 0, у) (0, 0, .) (11, 1, О) (9, 1,) (30, 2, п) (16, 1, а) (23, 1, и) (34, 1, с) (0, 0, я) (75, 3, д) (14, 1, л) (0, 0, г) (3, 1, е) (34, 2, у) (51, 2, ш) (34, 2, т) (23, 2, е) (72, 2, ч) (71, 2, б) (103, 2, н) (64, 10, л) (3, 1, б) (6, 1, в) (0, 0, ь) (61, 2, Н) (6, 1,) (22, 2,) (44, 3, и) (5, 1, е) (54, 2,) (2, 1, ж) (8, 1, д) (15, 1, л) (12, 2, м) (21, 1, о) (14, 3, п) (85, 1, е) (3, 1, я) (26, 1, с) (67, 3, й) (12, 1, и) (28, 2, п) (26, 1, с) (23, 2, с) (86, 2, й) (116, 5, в) (10, 2, р) (11, 1, т) (116, 2,) (184, 1, в) (7, 1, р) (20, 2,) (33, 2, п) (14, 2, ц) (179, 8, е) (14, 2, о) (69, 1, о) (68, 1, а) (76, 3, е) (8, 1, у) (29, 2, л) (84, 3, е) (143, 1, а) (7, 2, .) (115, 4, о) (63, 2, н) (34, 2, с) (107, 2, в) (18, 2, с) (180, 2, и) (106, 3, о) (183, 3, ж) (14, 2,) (158, 3, й) (60, 3, и) (273, 2, .)	(0, О) (0, д) (0, н) (0, а) (0, ж) (2, ы) (0,) (0, в) (7, с) (0, к) (4, з) (10, е) (0, ,) (7, ж) (0, и) (0, л) (0, -) (0, б) (0, ы) (16,) (0, п) (0, р) (15, н) (0, ц) (13,) (10, о) (0, т) (0, о) (22, ы) (0, й) (7, х) (28, т) (0, е) (20, н) (4, й) (27, и) (9, в) (28, ю) (7, п) (22, и) (3, ц) (33, с) (0, с) (0, у) (0, .) (7, О) (3,) (32, п) (22, а) (8, и) (16, с) (0, я) (7, в) (7, д) (28, л) (0, г) (28, е) (39, у) (27, е) (0, ш) (42, т) (50, е) (25, ч) (27, о) (18, ы) (7, н) (35, т) (15,) (43, в) (38,) (16, ю) (18, о) (8, ь) (45,) (0, Н) (28,) (3, а) (58, т) (68,е) (56, о) (7, о) (5, и) (2, а) (16, и) (7, м) (3, о) (80,) (21, р) (33, п) (52, т) (43, т) (50, й) (7, и) (81, п) (4, с) (86, с) (59, й) (74, О) (47, в) (91, р) (33, т) (15, л) (7, з) (8, е) (22, е) (30,) (68, п) (36, ц) (25, к) (32, о) (29, е) (39, о) (0, м) (28, г) (4, л) (79, м) (44,) (102, и) (85, е) (60, а) (84, .) (7, Н) (76, о) (47, н) (33,) (43, д) (4, в) (115, с) (52,) (107, р) (28, д) (55, ж) (115,) (69, о) (106, п) (28, и) (43, к) (45,)
Длина закодированного сообщения	341 · 8 = 2728 бит	278 · 8 = 2224 бит

Заключение

В данной статье была рассмотрена теоретическая база и практическая реализация алгоритмов сжатия данных LZ77 и LZ78. Были созданы математические модели этих алгоритмов, проведена их верификация и численные эксперименты. Важно отметить, что несмотря на наличие теоретического материала по алгоритмам семейства Лемпеля – Зива в общедоступных источниках, программной реализации этих алгоритмов, выполненной на языке программирования C++, в свободном доступе практически нет. Также можем сделать вывод, что LZ77 более эффективен для последовательностей, где часто встречаются повторяющиеся фразы, а LZ78 может быть более эффективен для разнообразных типов данных, так как он строит словарь на основе фактических фраз.

Литература

1. Salomon, D. Data compression: the complete reference / D. Salomon. – New York : Springer, 2007. – 1111 p.

2. Сергеенко, В. С. Сжатие данных, речи, звука и изображений в телекоммуникационных системах / В. С. Сергеенко, В. В. Баринов. – Москва : РадиоСофт, 2009. – 360 с.
3. Спиваковский, А. М. Управляемое сжатие данных / А. М. Спиваковский. – Санкт-Петербург : Ниц Арт-Печатный Цех, 2018. – 258 с.
4. Совершенствование алгоритмов сжатия-восстановления сигналов для систем телеизмерений / Е. А. Ломтев, М. Г. Мясникова, Н. В. Мясникова [и др.] // Измерительная техника. – 2015. – № 3. – С. 11–15.
5. Рябко, Б. Я. Сжатие данных с помощью «мнимого скользящего окна» / Б. Я. Рябко // Проблемы передачи информации. – 1996. – Т. 32. – №. 2. – С. 22–30.
6. Беспалова, Н. М. Обзор основных алгоритмов сжатия данных / Н. М. Беспалова, Л. А. Сологубова, Ф. Н. Байбекова // Информационная безопасность – актуальная проблема современности. Совершенствование образовательных технологий подготовки специалистов в области информационной безопасности. – 2019. – № 1 (10). – С. 145–148.
7. Информационные технологии XXI века: сборник научных трудов / Тихоокеанский государственный университет ; ответственный редактор В. В. Воронин. – Хабаровск : Издательство ТОГУ, 2023. – 422, [2] с. – URL: https://www.elibrary.ru/download/elibrary_53737268_90436074.pdf (дата обращения: 25.04.2024).
8. Майоров, А. А. Общая и теоретическая информатика : учебное пособие / А. А. Майоров, В. П. Седякин. – Москва : МИИ-ГАиК, 2017. – 128 с.

Поступила в редакцию 07.07.2024

***Владислав Владимирович Кудинов**, инженер-конструктор 1 категории,*
e-mail: vlad_kud77@mail.ru, т. +7 (977) 937-56-40.
(Акционерное общество «Особое конструкторское бюро Московского энергетического института»).
***Олег Константинович Сапрыкин**, инженер, e-mail: sfsdc.01@mail.ru, т. +7 (903) 293-30-14.*
(Акционерное общество «Научно-исследовательский институт точных приборов»).

COMPARISON OF THE EFFICIENCY OF LZ77 AND LZ78 COMPRESSION ALGORITHMS

V. V. Kudinov, O. K. Saprykin

***T**he purpose of the work is the software implementation and study of the basic principles of operation of dictionary compression methods of the Lempel – Ziv family, and comparison of the effectiveness of these methods. Developed mathematical models of LZ dictionary methods, allowing to demonstrate the technology for forming dictionaries, as well as evaluate the effectiveness of compression methods. This article describes compression algorithms of the Lempel – Ziv family (LZ77, LZ78).*
***Key words**: information, message, information compression, efficient coding, Lempel – Ziv method, LZ77, LZ78.*

References

1. Salomon, D. Data compression : the complete reference / D. Salomon. – New York : Springer, 2007. – 1111 p.
2. Sergeenko, V. S. Compression of data, speech, sound and images in telecommunication systems / V. S. Sergeenko, V. V. Barinov. – Moscow : RadioSoft, 2009. – 360 p.
3. Spivakovsky, A. M. Controlled data compression / A. M. Spivakovsky. – St. Petersburg : Nits Art Printing Shop, 2018. – 258 p.
4. Improvement of signal compression-reconstruction algorithms for telemetry systems / E. A. Lomtev, M. G. Myasnikova, N. V. Myasnikova [et. al] // Measuring technology. – 2015. – No. 3. – P. 11–15.
5. Ryabko, B. Ya. Data compression using an «imaginary sliding window» / B. Ya. Ryabko // Problems of information transmission. – 1996. – Vol. 32. – No. 2. – P. 22–30.
6. Bespalova, N. M. Review of the main data compression algorithms / N. M. Bespalova, L. A. Sologubova, F. N. Baybekova // Information security – an urgent problem of our time. Improving educational technologies for training specialists in the field of information security. – 2019. – No. 1 (10). – P. 145–148.
7. Information technologies of the XXI century: collection of scientific papers / Tikhookean State University ; executive editor V. V. Voronin. – Khabarovsk : Publishing House of TOGU, 2023. – 422, [2] p. – URL : https://www.elibrary.ru/download/elibrary_53737268_90436074.pdf (date of access: 25.04.2024).
8. Mayorov, A. A. General and theoretical computer science: textbook / A. A. Mayorov, V. P. Sedyakin. – Moscow : MII-GAiK, 2017. – 128 p.

***Vladislav Vladimirovich Kudinov**, design engineer of the 1st category,*
e-mail: vlad_kud77@mail.ru, t. +7 (977) 937-56-40.
(Joint Stock Company «Special Design Bureau of the Moscow Power Engineering Institute»).
***Oleg Konstantinovich Saprykin**, engineer, e-mail: sfsdc.01@mail.ru, t. +7 (903) 293-30-14.*
(Joint Stock Company «Scientific Research Institute of Precision Instruments»).