

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Департамент математического и компьютерного моделирования

Адаптивный алгоритм Лемпеля-Зива-Велча

Доклад

Направление подготовки 09.03.03 Прикладная информатика

Профиль «Прикладная информатика в компьютерном дизайне»

Обучающийся _____

А.А. Виноходова

Руководитель _____ доцент ИМКТ

А.С. Кленин

Владивосток 2022

Содержание

1	Введение	3
1.1	Суть и назначение	3
1.2	Авторство	3
1.3	История развития	3
1.4	Состояние, реализация	3
1.5	Перспектива использования	3
2	Метод	4
2.1	Формальное описание метода	4
2.1.1	Словарь	4
2.1.2	Кодирование	4
2.1.3	Заполнение словаря	4
2.1.4	Переполнение словаря	4
2.1.5	Декодирование	4
2.2	Псевдокод	4
2.2.1	Кодирование:	4
2.2.2	Декодирование:	5
2.3	Пример	6
2.3.1	Кодирование	6
3	Формальная постановка задачи	7
4	Тестирование	8
4.1	Корректность	8
5	Исследование	9
5.1	Коэффициент сжатия	9
5.2	Зависимость степени сжатия от размера файла	9
6	Список литературы	10

1 Введение

1.1 Суть и назначение

Алгоритм LZW (алгоритм Лемпеля-Зива-Велча) – это универсальный алгоритм сжатия данных без потерь. Он предназначен для кодирования текста и графических изображений. По своей сути алгоритм напоминает канал связи тем, что переводит входную последовательность байт в более удобный для хранения и передачи формат, а затем декодирует ее обратно в точную копию входного сообщения.

1.2 Авторство

Алгоритм LZW был опубликован Терри А. Велчем в 1984 году в качестве улучшения алгоритма LZ78, описанного ранее Абрахамом Лемпелем и Якобом Зивом. Название LZW является аббревиатурой фамилий создателей (англ. Lempel-Ziv-Welch).

1.3 История развития

Алгоритм Лемпеля-Зива-Велча входит в семейство алгоритмов словарного сжатия данных LZ, берущее начало от алгоритмов LZ77 и LZ78. Наиболее распространенными модификациями алгоритма LZW являются: LZC (Lempel-Ziv Compress, 1985 г.), LZT (Lempel-Ziv-Tischer, 1985 г.), LZMW (Lempel-Ziv-Miller-Wegman, 1985 г.) и LZAP (Lempel-Ziv All Prefixes, 1988 г.).

1.4 Состояние, реализация

На момент своего появления алгоритм LZW стал первым широко используемым на компьютерах методом сжатия данных. Он стал популярен благодаря своим невысоким требованиям к программному обеспечению и сравнительно простой реализации.

1.5 Перспектива использования

В настоящее время используется в файлах таких форматов, как TIFF, PDF, GIF, PostScript, а также во многих известных архиваторах, в том числе ZIP, ARJ, LHA.

2 Метод

2.1 Формальное описание метода

2.1.1 Словарь

Инициализируется динамический словарь. Словарь включает все символы используемого алфавита (в данном случае алфавит представляет из себя первые 256 символов из таблицы `ascii`). Коды в словаре изначально имеют длину 8 бит (т.е. словарь включает 256 записей).

2.1.2 Кодирование

Алгоритм считывает текст сообщения посимвольно слева направо и ищет максимальную строку, которой нет в словаре – WK , где W – строка, имеющаяся в словаре, а K – символ, следующий за ней в сообщении. Найденная строка WK вносится в словарь и ей присваивается уникальный код, программа выводит код строки W , а следующая рассматриваемая строка начинается символа K . На выход при этом поступает код строки из словаря, которая короче найденной на 1 символ.

2.1.3 Заполнение словаря

По мере добавления записей в словарь в случае переполнения длина новых кодов увеличивается на 1 бит (т.е. как только потребуется больше 256 записей в словаре, длина новых кодов становится 9 бит, а размер словаря увеличивается до 512 записей). Максимальная длина кодов составляет 16 бит (65536 записей в словаре).

2.1.4 Переполнение словаря

По достижении максимального количества записей словарь перестает пополняться и используется дальше без изменений.

2.1.5 Декодирование

При декодировании сообщения алгоритм создает словарь фраз идентичный тому, что создавался при кодировании. На вход требуется только закодированное сообщение. Процесс декодирования имитирует процесс кодирования и может происходить одновременно с ним.

2.2 Псевдокод

2.2.1 Кодирование:

1. Инициализировать начальный словарь, содержащий все возможные символы;

2. Инициализировать строку W и присвоить ей первый символ входного сообщения;
3. Если КОНЕЦ СООБЩЕНИЯ, то вывести код для W и завершить алгоритм. Считать очередной символ K из входного сообщения;
4. Если фраза WK уже есть в словаре, то присвоить входной фразе W значение WK и перейти к шагу 2;
5. Иначе выдать код W , добавить WK в словарь, присвоить входной фразе W значение K и перейти к шагу 2.

2.2.2 Декодирование:

1. Инициализировать начальный словарь, содержащий все возможные символы;
2. Инициализировать строку W и присвоить ей первый символ декодируемого сообщения;
3. Считать очередной код K из сообщения;
4. Если КОНЕЦ СООБЩЕНИЯ, то вывести символ, соответствующий коду W , иначе:
5. Если фразы под кодом WK нет в словаре, вывести фразу, соответствующую коду W , а фразу с кодом WK занести в словарь;
6. Иначе присвоить входной фразе код WK и перейти к Шагу 2.

2.3 Пример

2.3.1 Кодирование

Входное сообщение, которое необходимо закодировать, имеет вид: ababcbabababaaaaaa. Инициализируется словарь односимвольных строк, содержащий 256 записей. В Таблице 1 приведен фрагмент исходного словаря, который будет использоваться в примере.

Символ	Код
...	...
a	97
b	98
c	99
...	...
	255
...	...

Таблица 1: Фрагмент исходного словаря при кодировании

Каждая новая фраза (в данном случае начиная с ab) заносится в словарь и ей присваивается уникальный код (ab присваивается 256). На выход поступает код записи, которая на символ короче данной (код фразы a - 97)

Новая фраза	Код	Вывод
a	97	
b	98	
c	99	
ab	256	97
ba	257	98
abc	259	256
cb	260	99
bab	261	257
baba	262	261
aa	263	97
aaa	264	263
aaaa	265	264
-	-	97

Таблица 2: Процесс кодирования

Таким образом закодированное сообщение получится следующим: 97 98 256 99 257 261 97 263 264 97. Для наглядности в качестве разделителей выбраны пробелы.

3 Формальная постановка задачи

1. Изучить алгоритм LZW на основе литературных источников, описать его в форме научного доклада.
2. Реализовать адаптивную версию алгоритма LZW, позволяющую по мере заполнения словаря увеличивать длину кодов от 8 до 16 бит.

Формат входного файла: Текстовый файл формата .txt в количестве символов N , включая строчные и прописные латинские символы, символы кириллицы, пробелы, переносы строки, знаки пунктуации.

Формат выходных файлов: Два текстовых файла формата .txt. Первый содержит закодированный текст входного файла, каждый код в файле располагается на отдельной строчке. Второй содержит декодированный текст. Входной файл и файл с декодированным текстом должны совпадать.

Ограничения: $(1 \leq N \leq 2^{20})$.

3. Исследовать алгоритм на предмет зависимости степени сжатия файла от его размера.
4. Результаты работы выложить в репозиторий GitHub

4 Тестирование

4.1 Корректность

Для проверки корректности работы алгоритма представлены тесты следующего формата: в директории Tests находятся 3 директории (in, encoded, decoded). В директории in находится последовательность из n тестов. Нумерация тестов начинается с 0 и не имеет пропусков. Название теста содержит только его порядковый номер. В директории encoded в результате работы функции encode создаются текстовые файлы, названия которых соответствующих порядковому номеру теста. В директории decoded в результате работы функции decode также создаются текстовые файлы, названия которых соответствующих порядковому номеру теста. Проверка на корректность работы программы в конкретном тесте считается пройденной, если если соответственные файлы в директории in и директории decoded равны.

5 Исследование

5.1 Коэффициент сжатия

Для оценки степени сжатия после выполнения теста высчитывается коэффициент сжатия по формуле $k = S_0/S_c$, где S_0 - размер исходного файла, S_c - размер сжатого файла. Чем больше коэффициент сжатия, тем эффективнее сжатие. Если k будет равен единице, то сжатый файл по размеру равен исходному. Если k будет меньше нуля, то сжатый файл по размеру больше исходного.

5.2 Зависимость степени сжатия от размера файла

Исследование зависимости степени сжатия от размера файла производится на тестах, содержание которых сгенерировано случайно. Тесты содержат 2^5 , 2^{10} , 2^{15} и 2^{20} латинских строчных символов таблицы ascii.

6 Список литературы

1. Welch T. A. A technique for high-performance data compression // Computer. — 1984. — Т. 6, № 17. — С. 8–19. — doi:10.1109/MC.1984.1659158.
2. Dinsky [Электронный ресурс] <https://youtu.be/XsllPSupzy4>
3. Arnold R., Bell T.[en]. A corpus for the evaluation of lossless compression algorithms // IEEE Data Compression Conference. — 1997. — С. 201–210. — doi:10.1109/DCC.1997.582019.
4. Bell T.[en], Witten I. H.[en], Cleary J. G.[en]. Modeling for text compression // ACM Computing Surveys[en]. — 1989. — Т. 21, № 4. — С. 557–591. — doi:10.1145/76894.76896.
5. Charikar M., Lehman E., Lehman A., Liu D., Panigrahy R., Prabhakaran M., Sahai A., Shelat A. The smallest grammar problem // IEEE Transactions on Information Theory[en]. — 2005. — Т. 51, № 7. — С. 2554–2576. — doi:10.1109/TIT.2005.2554254.
6. De Agostino S., Silvestri R. A worst-case analysis of the LZ2 compression algorithm // Information and Computation[en]. — 1997. — Т. 139, № 2. — С. 258–268. — doi:10.1006/inco.1997.2668.
7. De Agostino S., Storer J. A. On-line versus off-line computation in dynamic text compression // Information Processing Letters[en]. — 1996. — Т. 59, № 3. — С. 169–174. — doi:10.1016/0020-0190(96)00068-3.
8. Hucke D., Lohrey M., Reh C. P. The smallest grammar problem revisited // String Processing and Information Retrieval (SPIRE). — 2016. — Т. 9954. — С. 35–49. — doi:10.1007/978-3-319-46049-9_4.
9. Lempel A., Ziv J. Compression of individual sequences via variable-rate coding // IEEE Transactions on Information Theory[en]. — 1978. — Т. 24, № 5. — С. 530–536. — doi:10.1109/TIT.1978.1055934.
10. Miller V. S[en], Wegman M. N.[en]. Variations on a theme by Ziv and Lempel // Combinatorial algorithms on words. — 1985. — Т. 12. — С. 131–140. — doi:10.1007/978-3-642-82456-2_9.
11. Sheinwald D. On the Ziv-Lempel proof and related topics // Proceedings of the IEEE[en]. — 1994. — Т. 82, № 6. — С. 866–871. — doi:10.1109/5.286190.
12. Storer J. A. Data Compression: Methods and Theory. — New York, USA: Computer Science Press, 1988. — 413 с. — ISBN 0-7167-8156-5.
13. Ziv J. A constrained-dictionary version of LZ78 asymptotically achieves the finite-state compressibility with a distortion measure // IEEE Information Theory Workshop. — 2015. — С. 1–4. — doi:10.1109/ITW.2015.7133077.

14. Adobe Systems Incorporated. Document management — Portable document format — Part 1: PDF 1.7 (англ.). PDF 1.7 specification. Adobe (1 июля 2008). Дата
15. Wikipedia — Lempel — Ziv — Welch
16. Семенюк В.В. — Экономное кодирование дискретной информации
17. Метод LZW — сжатия данных — алгоритмы и методы
18. Алгоритмы сжатия и компрессии
19. Алгоритм LZW — Понятие алгоритма
20. Вирт Н. Алгоритмы и структуры данных/Н. Вирт. М.: Мир,1989.
21. Сибуя М. Алгоритмы обработки данных/М. Сибуя, Т. Ямамото. М.: Мир,1986.
22. Костин А.Е. Организация и обработка структур данных в вычислительных системах: учеб.пособ. для вузов/А.Е. Костин, В.Ф. Шаньгин . М.: Высш.шк., 1987.
23. Кнут Д. Искусство программирования для ЭВМ.Т.1: Основные алгоритмы:пер. с англ./Д. Кнут. М.:Мир,1978.
24. Кнут Д. Искусство программирования для ЭВМ.Т.3: Сортировка и поиск.: пер. с англ./Д.Кнут. М.:Мир,1978.
25. Кормен Т. Алгоритмы: построение и анализ./Т. Кормен, Ч. Лейзерсон, Р.Ривест. М.: МЦНМО, 2000
26. Кричевский Р.Е. Сжатие и поиск информации/Р.Е. Кричевский. М.: Радио и связь, 1989
27. Интернет ресурс. <https://habr.com/ru/post/132683/>