Министерство образования и науки РФ

Белгородский государственный технологический университет им. В. Г. Шухова

Кафедра программного обеспечения вычислительной техники  
и автоматизированных систем

Индивидуальное домашнее задание

по дисциплине «Теория информации»  
на тему «Процедуры сжатия и распаковки алгоритмом LZW»

Автор работы:  
студент группы ПВ-21  
Адаменко Игорь

Руководитель проекта:  
Сергиенко Е. Н.

Белгород  
2014

# Введение

Алгоритм Лемпеля — Зива — Велча (Lempel-Ziv-Welch, LZW) — это универсальный алгоритм сжатия данных без потерь, созданный Абрахамом Лемпелем, Якобом Зивом и Терри Велчем. Он был опубликован Велчем в июне 1984 года в статье «IEEE Computer» для журнала «Информационные теории», в качестве улучшенной реализации алгоритма LZ78, опубликованного Лемпелем и Зивом в мае 1977 года в том же журнале, в статье «IEEE Trans». Алгоритм разработан так, чтобы его можно было быстро реализовать, но он не обязательно оптимален, поскольку не проводит никакого анализа входных данных.

Акроним «LZW» указывает на фамилии изобретателей алгоритма: Лемпель, Зив и Велч, но многие утверждают, что, поскольку патент принадлежал Зиву, то метод должен называться алгоритмом Зива — Лемпеля — Велча.

# Описание алгоритма

Алгоритм LZW представляет собой алгоритм кодирования последовательностей неодинаковых символов. Возьмём для примера строку «Функция WriteOnScreenLZW использует модуль WriteOnScreenFunc.». Анализируя эту строку, мы можем увидеть, что слово «WriteOnScreen» повторяется дважды. В этом слове 13 символов — 104 бит. И если мы сможем заменить это слово в выходном файле, во втором его включении, на ссылку на первое включение, то получим сжатие информации. Алгоритм LZW как-бы «обучается» в процессе сжатия файла. Если существуют повторяющиеся строки в файле, то они будут закодированы в таблицу.

Процесс сжатия выглядит следующим образом. Последовательно считываются символы входного потока, и происходит проверка, существует ли в созданной таблице строк такая строка. Если такая строка существует, считывается следующий символ, а если строка не существует, в поток заносится код для предыдущей найденной строки, строка заносится в таблицу, а поиск начинается снова.

Для декодирования на вход подаётся только закодированный текст, ибо алгоритм может воссоздать таблицу соответствия кодов строкам непосредственно по закодированному тексту. LZW генерирует однозначно декодируемый код за счёт того, что каждый раз, когда генерируется новый код, новая строка добавляется в таблицу строк. Этот алгоритм постоянно проверяет, является ли строка уже известной, и, если так, выводит существующий код без генерации нового. Таким образом, каждая строка будет храниться в единственном экземпляре, и иметь свой уникальный номер. Следовательно, при дешифровании при получении нового кода генерируется новая строка, а при получении уже известного, строка извлекается из словаря.

## Алгоритм кодирования

1. Все возможные символы занести в словарь. Во входную фразу X занести первый символ сообщения.
2. Считать очередной символ Y из сообщения.
3. Если Y — это символ конца сообщения, то выдать код для X, иначе:
   1. Если фраза XY уже имеется в словаре, то присвоить входной фразе значение XY и перейти к п. 2.
   2. Иначе выдать код для входной фразы X, добавить XY в словарь и присвоить входной фразе значение Y. Перейти к п. 2

## Алгоритм декодирования

1. Все возможные символы заносятся в словарь. Во входную фразу X занести первый код декодируемого сообщения.
2. Считать очередной код Y из сообщения.
3. Если Y — это конец сообщения, то выдать символ, соответствующий коду X, иначе:
   1. Если фразы под кодом XY нет в словаре, вывести фразу, соответствующую коду X, а фразу с кодом XY занести в словарь. Перейти к п. 2.
   2. Иначе присвоить входной фразе код XY и перейти к п. 2.

## Достоинства

* Однопроходность.
* Для декодирования не надо сохранять таблицу строк в файл для распаковки. Алгоритм построен таким образом, что мы в состоянии восстановить таблицу строк, пользуясь только потоком кодов.
* Высокая скорость работы, как при упаковке, так и при распаковке, достаточно скромные требования к памяти и простая аппаратная реализация.

## Недостатки

* Не производится анализ входных данных.
* Низкая степень сжатия по сравнению с более популярными алгоритмами.

## Применение

Опубликование алгоритма LZW произвело большое впечатление на всех специалистов по сжатию информации. За этим последовало большое количество программ и приложений с различными вариантами этого метода.

LZW позволяет достичь одну из наилучших степеней сжатия среди других существующих методов сжатия графических данных, при полном отсутствии потерь или искажений в исходных файлах. В настоящее время используется в файлах формата TIFF (опционально), PDF, GIF, PostScript и других, а также отчасти во многих популярных программах сжатия данных (ZIP, ARJ, LHA). Также был реализован в программе compress, которая стала более или менее стандартной утилитой Unix-систем приблизительно в 1986 году.

# Программная реализация

Для реализации алгоритма был написан класс на языке программирования C#, который обрабатывает исходный файл (сжимает) в соответствии с вышеописанном алгоритмом, используя в качестве начального набора символов — таблицу ASCII. После этого создаётся новый файл, в котором далее будет храниться сжатая информация. После сжатия, рассчитывается коэффициент сжатия.

При декодировании, обрабатывается файл с ранее сжатой информацией, после чего создаётся новый файл, в который записывается декодированная информация.

## Пример работы программы

Содержание исходного файла:

— Нигде так не нужна дедукция, как в религии, — сказал он, прислонившись к ставням. — Логик может поднять ее до уровня точной науки. Мне кажется, что своей верой в Божественное провидение мы обязаны цветам. Все остальное — наши способности, наши желания, наша пища — необходимо нам в первую очередь для существования. Но роза дана нам сверх всего. Запах и цвет розы украшают жизнь, а не являются условием ее существования. Только Божественное провидение может быть источником прекрасного. Вот почему я и говорю: пока есть цветы, человек может надеяться.

Содержание сжатого файла:

3]7#”@тp:„Ш3µжжvёW#НФцt?Е‚T$@в<®ёЎСЂ<c®w¬@оvЖЭпCЕЧ0t8џWдp@с„8ќЇч`ё@@nзўмw9њЇ! ЅЬдЎј§®X3‘Ь Ф'ьЬчv»ќ0·YСFfC#Ћ“Йг=ЮUч‹‰Ьеµёњ¶,ЃѓQrР0NЫLgctA]®€3±ч/pїеЋЬ»Ыў€O3єЃ-~cDЊоVx»њ6љћЩ®©:Ь;Ж¶$пt>bZЗ.ЙлWtT-ИэWвyїҐп|•Й=r:мЇ›зтї8·qљ5!›\_±K"{ё”G™}БЅcъGєЊЗp;Ь­'@ѓ=SHNгќ—EЏЂш8еPж:sґьFЮФя8ќЁ)wXf@еsДэЭwЮЊяQЌ#ёл?Ґ}‡TЈ•Њ\_ХU€вdbцP TPЂб>Х Ѓ5^Ц…8оs3”кЃЏҐхQЊ#№TUќC°уWхх\_Џш:UNдt wУЧьт>С·Q7`”ц!TDPSэZ]ФcРы8§DJv'dёыЃЏ Рл9ЋйЭв

Распакованный файл идентичен запакованному.

Размер исходного файла: 589 байт.

Размер сжатого файла: 455 байт.

Процент сжатия: 22,75%

# Заключение

Сложно дать оценку результативности какой-либо техники сжатия данных, поскольку каждая из них работает оптимально на задачах определённого вида. Для LZW-сжатия это задачи, которые содержат повторяющиеся элементы. Это могут быть данные в текстовом файле (повторяющиеся n-граммы), изображения (цветовые точки) и тому подобное. Чем чаще элементы повторяются, тем эффективнее работает алгоритм.

В качестве заключительного примера можно привести процент сжатия данной работы алгоритмом — 40,4%.

# Список литературы

1. Общие сведения об архивации данных (Лекция 14) — [bit.ly/1yaEA4b](http://bit.ly/1yaEA4b)
2. Алгоритм LZW — Викиконспекты — [bit.ly/1oxV2d0](http://bit.ly/1oxV2d0)
3. Алгоритмы LZW, LZ77 и LZ78 / Хабрахабр — [bit.ly/1vvG3Qt](http://bit.ly/1vvG3Qt)
4. Алгоритм Лемпеля — Зива — Велча — Википедия — [bit.ly/1lKa1xd](http://bit.ly/1lKa1xd)
5. Метод LZW-сжатия данных — [bit.ly/1lKbfIJ](http://bit.ly/1lKbfIJ)