УДК 621.327:681.5

А.В. Яковенко,
кандидат технических наук,
старший научный сотрудник,
С.Н. Пугачев,
В.В. Твердохлеб

АНАЛИЗ ХАРАКТЕРИСТИК АЛГОРИТМА СЖАТИЯ ДАННЫХ LZW

В данной статье рассматривается словарный алгоритм сжатия информации LZW. Подробно описан принцип его работы, на примере рассмотрены основные шаги кодирования и для общего понимания процесса наглядно приведены результаты сжатия данных. Кратко описывается процесс декодирования, принципы составления словаря в процессе кодирования и декодирования, а также на примере рассмотрена его организация и строение узлов дерева словаря. В заключении приведены некоторые проблемные вопросы, связанные с работой метода, которые представляют собой пространство для модернизации и доработки алгоритма.

Ключевые слова: *LZW кодирование, сжатие данных без потерь, организация словаря LZW, недостатки алгоритма сжатия информации LZW.*

У статті розглянуто алгоритм стиснення інформації LZW. Докладно описано принцип його роботи, на прикладі розглянуто основні кроки кодування, і для загального розуміння процесу наглядно наведено результати стиснення даних. Коротко описано процес декодування, принципи складання словника в процесі кодування, а також на прикладі розглянута його організація та будова вузлів дерева словника. У висновку наведено деякі проблемні питання, пов'язані з роботою методу, що становлять простір для модернізації і доопрацювання алгоритму.

Ключові слова: *LZW кодування, стиснення даних без втрат, організація словника LZW, недоліки алгоритму стиснення інформації LZW.*

LZW data compression algorythm is examined. It is described in detail how it works. The basic steps of coding are described for example, and for general understanding of the process the results of data compression are showed. The decoding process is briefly described, the principles of the dictionary coscruction in the process of encoding and decoding are considered, as well as its organization and units construction are shown. In conclusion, several issues, related to the method which represent space for upgrading and improvement of the LZW algorithm, are stated.

Keywords: *LZW coding, lossless data compression, construction of LZW dictionary, weaknesses of the LZW data compression algorithm.*

Введение

Сообщения, передаваемые с использованием каналов связи (речь, музыка, телевизионные изображения и т.д.), в большинстве своем предназначены для не-

посредственного восприятия органами чувств человека и обычно плохо приспособлены для их эффективной передачи по каналам связи. Поэтому они в процессе передачи, как правило, подвергаются кодированию.

Кодирование сообщений может преследовать различные цели. Например, это кодирование с целью засекречивания передаваемой информации. Другим примером кодирования может служить преобразование дискретных сообщений из одних систем счисления в другие (из десятичной в двоичную, восьмеричную и т. п., из непозиционной в позиционную, преобразование буквенного алфавита в цифровой и т. д.). Кодирование в канале или помехоустойчивое кодирование информации может быть использовано для уменьшения количества ошибок, возникающих при передаче по каналу с помехами. Наконец кодирование сообщений может производиться с целью сокращения объема информации и повышения скорости ее передачи или сокращения полосы частот, требуемых для передачи. Такое кодирование называют экономным, безизбыточным или эффективным кодированием, а также сжатием данных.

Передача и хранение информации требуют достаточно больших затрат. И чем с большим количеством информации нам приходится иметь дело, тем дороже это стоит. К сожалению, большая часть данных, которые нужно передавать по каналам связи и сохранять, имеет не самое высокое компактное представление. Скорее эти данные хранятся в форме, обеспечивающей их наиболее простое использование, например: обычные книжные тексты, ASCII коды текстовых редакторов, двоичные коды данных ЭВМ, отдельные отсчеты сигналов в системах сбора данных и т.д. Однако такое наиболее простое в использовании представление данных требует вдвое-втрое, а иногда и в сотни раз больше места для их сохранения и полосу частот для их передачи, чем на самом деле нужно. Поэтому сжатие данных – это одно из наиболее актуальных направлений современной науки.

Существуют два типа систем сжатия данных: системы сжатия без потерь информации (неразрушающее сжатие) и системы сжатия с потерями информации (разрушающее сжатие). В системах сжатия без потерь декодер восстанавливает данные источника абсолютно точно, а в системах сжатия с потерями (или с разрушением) кодирование производится таким образом, что декодер не в состоянии восстановить данные источника в первоначальном виде. Выбор системы неразрушающего или разрушающего сжатия зависит от типа данных, подлежащих сжатию. Поэтому *цель исследований статьи* заключается в анализе базового словарного алгоритма сжатия данных без потери информации

Основная часть

Словарные алгоритмы сжатия информации носят менее математически обоснованный характер нежели к примеру метод Шенона-Фано или Хаффмена, но более практический. Они не рассматривают статистические особенности кодируемых данных и не используют коды переменной длины. Вместо этого они выбирают некоторые последовательности символов, сохраняют их в словаре, а все последовательности кодируются в виде меток, используя словарь. При декодировании осуществляется обратная замена индекса на соответствующую ему фразу из словаря. Словарь – это набор таких фраз, которые, как мы полагаем, будут

встречаться в обрабатываемой последовательности. Индексы фраз должны быть построены таким образом, чтобы в среднем их представление занимало меньше места, чем требуют замещаемые строки. За счет этого и происходит сжатие. Словарь может быть статистическим или динамическим (адаптивным). Первый является постоянным, иногда в него добавляют новые последовательности, но никогда не удаляют. Динамический словарь содержит последовательности, ранее поступившие из входного, при этом разрешается и добавление, и удаление данных из словаря по мере чтения входного файла. В общем случае использование динамического словаря является более предпочтительным. Такой метод сжатия начинает с пустого или маленького исходного словаря, добавляет в него слова по мере поступления из входного файла и удаляет старые слова, поскольку поиск по большому словарю делается медленно.

Наиболее совершенным на данный момент представителем этой группы словарных методов является алгоритм LZW, разработанный в 1984 году Терри Вэлчем. Это весьма популярный вариант алгоритма LZ78. Его главной особенностью является более короткая метка, которая состоит только из указателя на место в словаре. Эффективность LZW зависит от структурных характеристик обрабатываемых данных. Принцип действия алгоритма основан на устранении структурной избыточности, обусловленной наличием одинаковых цепочек элементов. Метод LZW начинается инициализацией словаря всеми символами исходного алфавита. Поскольку словарь уже частично заполнен, первые поступившие символы всегда будут обнаружены в словаре, поэтому метка может состоять всего лишь из указателя, и нет надобности дополнительно хранить код символа, как в алгоритмах LZ77 и LZ78.

Метод LZW накапливает поступающие на вход символы в строке *I*. После каждого нового символа, добавленного в строку *I*, кодер ищет *I* в словаре. Если строка обнаружена, то процесс удлинения *I* продолжается. В некоторый момент добавление нового символа *x* приводит к необнаружению строки *Ix* (символ *x* был добавлен к *I*). Тогда кодер записывает в выходной файл указатель в словаре на строку *I*, сохраняет строку *Ix* (которая теперь будет называться фразой) в словаре на следующей допустимой позиции и инициализирует (присваивает) строке *I* новое значение *x*.

Рассмотрим работу алгоритма на примере (табл. 1). При сжатии изображений инициализация словаря алгоритма происходит кодами всех цветов изображения. Для удобства возьмем только три основных цвета: “красный”, “зеленый”, “синий”; обозначим их соответственно “R”, “G”, “B”, а так же присвоим им коды “1”, “2” и “3”. Чтобы проиллюстрировать процесс кодирования, закодируем следующую строку “RGBGRBRBGRGRBGBRGRGR”. Получаем следующие шаги.

0. Инициализация словаря.

1. Первый входной элемент “R” обнаруживается в словаре под номером 1 (этот номер мы ему условно назначили). Следующий входной элемент “G”, но строки “RG” нет в словаре. Кодер делает следующее: (1) он выдает на выход ссылку 1, (2) сохраняет “RG” в следующей доступной позиции словаря (запись номер 4) и (3) инициализирует *I* строкой “G”.

2. На вход поступает элемент “B”, но строки “GB” нет в словаре. Кодер (1) записывает на выход 2 (код, который мы присвоили “G”), (2) сохраняет в словаре “GB” (запись 5) и (3) инициализирует *I* строкой “B”.

Таблица 1

Кодирование LZW для “RGBGRBRBGRGRBGBRGRGR”

I	В словаре?	Новая запись	Выход	*I*	В словаре?	Новая запись	Выход
R	да			GRG	нет	11-GRG	7(GR)
RG	нет	4-RG	1(R)	G	да		
G	да			GR	да		
GB	нет	5-GB	2(G)	GRB	нет	12-GRB	7(GR)
B	да			B	да		
BG	нет	6-BG	3(B)	BG	да		
G	да			BGB	нет	13-BGB	6(BG)
GR	нет	7-GR	2(G)	B	да		
R	да			BR	да		
RB	нет	8-RB	1(R)	BRG	нет	14-BRG	9(BR)
B	да			G	да		
BR	нет	9-BR	3(B)	GR	да		
R	да			GRG	да		
RB	да			GRGR	нет	15-GRGR	11(GRG)
RBG	нет	10-RBG	8(RB)	R	да		
G	да			R,end	нет		1(R)
GR	да						

Так продолжается до окончания кодируемых данных. Полный процесс кодирования строки “RGBGRBRBGRGRBGBRGRGR” приведен в табл. 1. В итоге для нашего случая выходные данные будут иметь следующий вид (входят только числа, но не символы в скобках). 1(R), 2(G), 3(B), 2(G), 1(R), 3(B), 8(RB), 7(GR), 7(GR), 6(BG), 9(BR), 11(GRG), 1(R).

Декодер начинает работу, так же как и кодер, с инициализации словаря. В нашем случае − кодами всех цветов. Затем он читает входной файл, который состоит из указателей в словаре, использует каждый указатель для того, чтобы восстановить несжатые символы из словаря и записать их в выходной файл. Кроме того, он строит словарь тем же методом, что и кодер.

На первом шаге декодирования декодер вводит первый указатель и использует его для восстановления словарного элемента *I*. Это строка символов, и она записывается декодером в выходной файл. Далее следует записать в словарь строку *Ix*, однако символ *x* еще не известен; это будет первый символ следующей строки, извлеченной из словаря.

На каждом шаге декодирования после первого декодер вводит следующий указатель, извлекает следующую строку *J* из словаря, записывает ее в выходной файл, извлекает ее первый символ *x* и заносит строку *Ix* в словарь на свободную позицию (предварительно проверив, что строки *Ix* нет в словаре). Затем декодер перемещает *J* в *I*. Теперь он готов к следующему шагу декодирования.

В нашем примере “RGBGRBRBGRGRBGBRGRGR” первым символом входного файла декодера является указатель 1. Он соответствует строке “R”, которая извлекается из словаря, сохраняется в *I* и становится первой строкой, записанной в выходной (разжатый) файл. Следующий указатель − 2, поэтому в *J* заносится строка “G”, а содержимое *J* записывается в выходной файл. Первый символ строки *J* добавляется к переменной *I*, образуя строку “RG”, которой нет в словаре, поэтому она добавляется в словарь на позиции 4. Содержимое переменной *J* переносится

в переменную *I*; теперь *I* равно “G”. Следующий указатель − 3, поэтому из словаря извлекается строка “B”, заносится в *J* и пишется в выходной файл. Переменная *I* дополняется до значения “GB”; такой строки нет в словаре, поэтому она туда заносится под номером 5. Переменная *J* переписывается в *I*; теперь *I* равно “B”. На следующем шаге декодер читает указатель 2, записывает “G” в файл и сохраняет в словаре строку “BG”. Так продолжается до конца файла.

Так как кодер и декодер считывают символы и добавляют их в строку *I* до тех пор, пока *I* находится в словаре, в некоторый момент строка *Ix* в словаре не обнаруживается, и тогда строка *Ix* помещается в словарь. Значит, при добавлении новых строк в словарь поступает всего один символ *x*. Это значит, что для каждой словарной строки в словаре найдется “родительская” строка, которая короче ровно на один символ. Поэтому для словаря LZW хорошим решением будет организация словаря в виде дерева, при построении которого добавление новых строк *Ix* делается добавление всего одного символа *x*. Из этого следует, что для каждой словарной строки в словаре найдется “родительская” строка, которая короче ровно на один символ. Пример дерева словаря LZW показан на рис. 1.

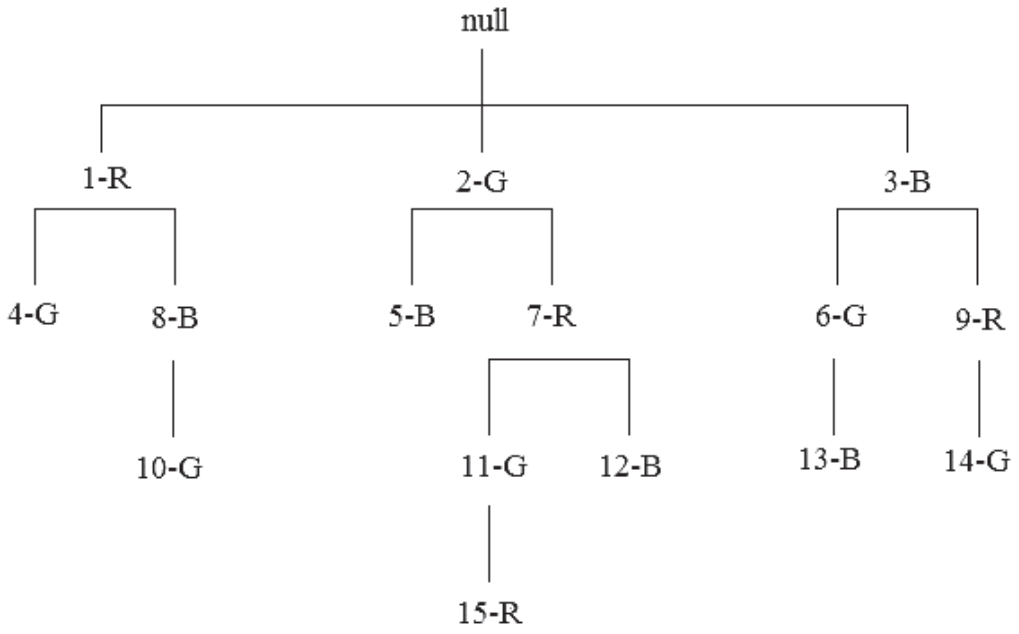

Рис. 1. Дерево словаря LZW

На практике узлы дерева словаря строятся состоящими из трех полей: указателя на родительский узел, указателя (или индекса), созданного в процессе хеширования, и кода символа, лежащего в этом узле (рис. 2).

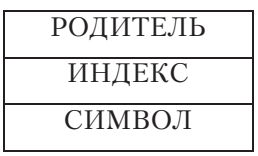

Рис. 2. Структура узла словаря LZW

Выводы

Алгоритм сжатия информации LZW использует те же принципы, что и остальные словарные методы сжатия. Он читает файл символ за символом и добавляет фразы в словарь. Фразы являются отдельными символами и строками символов входного файла. Когда строка входного файла совпадает с некоторой фразой в словаре, в сжатый файл записывается позиция этой фразы или метка. Эффективность LZW зависит от структурных характеристик обрабатываемых данных. Принцип действия алгоритма основан на устранении структурной избыточности, обусловленной наличием одинаковых цепочек элементов. Не смотря на то, что LZW имеет большую эффективность, нежели не только статистические методы, но и методы его группы, он имеет так же и ряд недостатков, которые оставляют пространство для модернизации и доработки.

Среди основных недостатков LZW, которые требуют устранения, – эффект сжатия проявляется медленно. Так как строки в словаре становятся длиннее на один символ на каждом шаге, требуется много времени для появления длинных строк в словаре. Как известно, в самом начале кодирования словарь инициализирован кодами всех элементов файла, и в связи с особенностями его пополнения он быстро становится переполненным. Безусловно, существуют решения, такие как полный сброс словаря или удаление старых записей, но, к сожалению, не существует хорошего алгоритма, который бы определял оптимальность действий: какие из записей целесообразно удалить, а какие оставить.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Сэломон Д.* Сжатие данных, изображений и звука / Д. Сэломон. – М. : Техносфера, 2004. – 368 с.

2. *Ватолин Д.* Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. – М. : ДИАЛОГ-МИФИ, 2003. – 384 с.

3. *Миано Дж.* Форматы и алгоритмы сжатия изображения в действии / Дж. Миано. – М. : Издательство “Триумф”, 2003. – 336 с.

4. *Лидовский В.В.* Теория информации : Учебное пособие / В.В. Лидовский. – М. : Компания “Спутник+”, 2004. – 111 с.

Отримано 03.02.2014