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

ДОКЛАД
о практическом задание по дисциплине АИСД
«Алгоритм Лемпеля - Зива - Велча»

направление подготовки 09.03.03 «Прикладная информатика»
профиль «Прикладная информатика в компьютерном дизайне»

Выполнил студент
гр. Б9121-09.03.03пикд
Лифарь Марина Евгеньевна

_____
(подпись)

Руководитель практики
Доцент ИМКТ А.С Кленин
(должность, уч. звание)

_____
(подпись)

«_____» _____ 2022г.

Доклад защищен:
С оценкой _____

Рег. № _____

«____» _____ 2022 г.

г. Владивосток

2022

Введение

Сжатие файла — это уменьшение его размера при сохранении исходных данных. В этом случае файл занимает меньше места на устройстве, что также облегчает его хранение и передачу через интернет или другим способом. Важно отметить, что сжатие не безгранично и обычно делится на два основных типа: с потерями и без потерь. Рассмотрим каждый из них по отдельности.

Степень сжатия файлов характеризуется коэффициентом *Кс,* определяемым как отношение объема сжатого файла *Vc* к объему исходного файла *Vo,* выраженное в процентах:

Кс=(Vc/Vo)*100%

Степень сжатия зависит от используемой программы, метода сжатия и типа исходного файла. Наиболее хорошо сжимаются файлы графических образов, текстовые файлы и файлы данных, для которых степень сжатия может достигать 5 - 40%, меньше сжимаются файлы исполняемых программ и загрузочных модулей - 60 - 90%. Почти не сжимаются архивные файлы. Программы для архивации отличаются используемыми методами сжатия, что соответственно влияет на степень сжатия.

История

Метод сжатия LZW (Lempel-Ziv-Welch) был предложен в 1978-ом году израильскими специалистами Лемпелом и Зивом и позже дорабатывался в Соединенных Штатах. Сжатие в нем, в отличие от алгоритма RLE, выполняется уже за счет одинаковых цепочек байтов. LZW является методом сжатия данных, который способен извлекать преимущества при наличии повторяющихся цепочек данных. Известность алгоритма LZW в существенной мере сопряжена с успехом программы compress. Исходный код последней версии этой программы, которая способна осуществлять как сжатие, так и декомпрессию, имеет размер всего в 1200 строк. Ядро кода сжатия имеет не больше ста строк, а код декомпрессии имеет немного больший размер. Программисты полагают, что это может облегчить чтение и понимание алгоритма, а также предоставляет возможность его адаптации для различных целей. Усовершенствованная версия алгоритма (Terry Welch) была

представлена в 1984-ом году. Алгоритм является, по сути, необычайно простым. LZW-сжатие выполняет замену строки символов определенными кодами. Это осуществляется без какого-нибудь предварительного анализа входной информации. Вместо этого при прибавлении любой новой строки символов реализуется просмотр таблицы строк. Сжатие выполняется, когда код подменяет символьную строку. Коды, которые генерирует алгоритм LZW, могут иметь любую длину, но они обязаны иметь больше битов, чем единичный символ. Алгоритм разработан так, чтобы его было достаточно просто реализовать как программно, так и аппаратно.

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.

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. Метод сжатия графических данных позволяет достичь одну из наилучших степеней сжатия среди других существующих методов, при полном отсутствии потерь или искажений в исходных файлах. В настоящее время используется в файлах формата TIFF, PDF, GIF, PostScript и других, а также отчасти во многих популярных программах сжатия данных (ZIP, ARJ, LHA).

Обычно декодирование LZW намного быстрее процесса кодирования. Автор LZW Терри Уэлч в свое время сумел запатентовать свой алгоритм в США. В настоящее время патент принадлежит компании Unisys. Алгоритм LZW определяется как часть стандарта ITU-T V.42bis, но Unisis установила жесткие условия лицензирования алгоритма для производителей модемов.

Описание

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.

Например, если сжимают байтовые данные (текст), то строк в таблице окажется 256 (от «0» до «255»). Если используется 10-битный код, то под коды для строк остаются значения в диапазоне от 256 до 1023. Новые строки формируют таблицу последовательно, т. е. можно считать индекс строки ее кодом.

Алгоритму декодирования на входе требуется только закодированный текст, поскольку он может воссоздать соответствующую таблицу преобразования непосредственно по закодированному тексту. Алгоритм генерирует однозначно декодируемый код за счет того, что каждый раз, когда генерируется новый код, новая строка добавляется в таблицу строк. LZW 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.

LZW compression for string: "ABABBABCABABBA"

s	c	output	code	string
			1	A
			2	B
			3	C
A	B	1	4	AB
B	A	2	5	BA
A	B			
AB	B	4	6	ABB
B	A			
BA	B	5	7	BAB
B	C	2	8	BC
C	A	3	9	CA
A	B			
AB	A	4	10	ABA
A	B			
AB	B			
ABB	A	6	11	ABBA
A	EOF	1		

- The output codes are: 1 2 4 5 2 3 4 6 1. Instead of sending 14 characters, only 9 codes need to be sent (compression ratio = 14/9 = 1.56).

Пример

Кодирование

Пусть мы сжимаем последовательность «abacabadabacabae».

- Шаг 1: Тогда, согласно изложенному выше алгоритму, мы добавим к изначально пустой строке “a” и проверим, есть ли строка “a” в таблице. Поскольку мы при инициализации занесли в таблицу все строки из одного символа, то строка “a” есть в таблице.
- Шаг 2: Далее мы читаем следующий символ «b» из входного потока и проверяем, есть ли строка “ab” в таблице. Такой строки в таблице пока нет.
- Добавляем в таблицу <5> “ab”. В поток: <0>;
- Шаг 3: “ba” — нет. В таблицу: <6> “ba”. В поток: <1>;
- Шаг 4: “ac” — нет. В таблицу: <7> “ac”. В поток: <0>;
- Шаг 5: “ca” — нет. В таблицу: <8> “ca”. В поток: <2>;
- Шаг 6: “ab” — есть в таблице; “aba” — нет. В таблицу: <9> “aba”. В поток: <5>;
- Шаг 7: “ad” — нет. В таблицу: <10> “ad”. В поток: <0>;
- Шаг 8: “da” — нет. В таблицу: <11> “da”. В поток: <3>;
- Шаг 9: “aba” — есть в таблице; “abac” — нет. В таблицу: <12> “abac”. В поток: <9>;
- Шаг 10: “ca” — есть в таблице; “cab” — нет. В таблицу: <13> “cab”. В поток: <8>;
- Шаг 11: “ba” — есть в таблице; “bae” — нет. В таблицу: <14> “bae”. В поток: <6>;
- Шаг 12: И, наконец последняя строка “e”, за ней идет конец сообщения, поэтому мы просто выводим в поток <4>.

Текущая строка	Текущий символ	Следующий символ	Вывод		Словарь
			Код	Биты	
ab	a	b	0	000	5: ab
ba	b	a	1	001	6: ba
ac	a	c	0	000	7: ac
ca	c	a	2	010	8: ca
ab	a	b	-	-	- -
aba	b	a	5	101	9: aba
ad	a	d	0	000	10: ad
da	d	a	3	011	11: da
ab	a	b	-	-	- -
aba	b	a	-	-	- -
abac	a	c	9	1001	12: abac
ca	c	a	-	-	- -
cab	a	b	8	1000	13: cab
ba	b	a	-	-	- -
bae	a	e	6	0110	14: bae
e	e	-	4	0100	- -

Итак, мы получаем закодированное сообщение «0 1 0 2 5 0 3 9 8 6 4», что на 11 бит короче.

Декодирование

Особенность LZW заключается в том, что для декомпрессии нам не надо сохранять таблицу строк в файл для распаковки. Алгоритм построен таким образом, что мы в состоянии восстановить таблицу строк, пользуясь только потоком кодов.
Теперь представим, что мы получили закодированное сообщение, приведённое выше, и нам нужно его декодировать. Прежде всего, нам нужно знать начальный словарь, а последующие записи словаря мы можем реконструировать уже на ходу, поскольку они являются просто конкатенацией предыдущих записей.

Данные		На выходе	Новая запись	
Биты	Код		Полная	Частичная
000	0	a	- -	5: a?
001	1	b	5: ab	6: b?
000	0	a	6: ba	7: a?
010	2	c	7: ac	8: c?
101	5	ab	8: ca	9: ab?
000	0	a	9: aba	10: a?
011	3	d	10: ad	11: d?
1001	9	aba	11: da	12: aba?
1000	8	ca	12: abac	13: ca?
0110	6	ba	13: cab	14: ba?
0100	4	e	14: bae	- -

Стоимость

Насколько экономичен алгоритм **LZW**? Тестирование производилось с помощью реализации, осуществленной **Иваном Боровым**.

Пример.

oitomii o imi oooitmi o o o ooiimtomiimotoim oi too i i m oio i omtoo toimo t iimiotmii ttmiotoitoomt imo o t mii i i m o ooi o tom totototoi oto i moi t i o mimto toootoi otomtioio t m iim toi m i iooim ittoomooo i i tom t oto iimitimoi o tmi tmi otomi i too t tioot tim mii oiooi tooimioomiootoooioo i oomiotmiotomi i o m ooo i tootmtiti i o ototi o omi i m iti i otooi i toooioomo i tmooi i oti tot iimii mio i moo omt totoo o m mooii imt i totmi i oot i o otito tom tot otoiiii i ootottm o ottoioooimo too imtoimoom oom titoo i oiomotoii i i oto iiioi i tioio too o m too tiotomtii oii o iii tom mot imt tiooi

содержащий 613613 символов может быть закодирован 3×613=18393×613=1839 битами.

Алгоритм **LZW** дал 241241 слово в словарь, стоимость кодирования

3×3+8×4+16×5+32×6+64×7+(241−3−120)×8=1705 бит
1705/1839≈0.927 ,1705/1839≈0.927 ,т.е. получили 88 % экономии.

Более длинный кусок, содержащий 10521052 символа, дал 370370 слов в словарь, стоимость кодирования

3×3+8×4+16×5+32×6+64×7+128×8+(370−3−248)×9=2856 бит
2856/(1052×3)≈0.905 ,2856/(1052×3)≈0.905 ,т.е. эффективность сжатия порядка 10%

Достоинства и недостатки

+ Не требует вычисления вероятностей встречаемости символов или кодов.
+ Для декомпрессии не надо сохранять таблицу строк в файл для распаковки. Алгоритм построен таким образом, что мы в состоянии восстановить таблицу строк, пользуясь только потоком кодов.
+ Данный тип компрессии не вносит искажений в исходный графический файл, и подходит для сжатия растровых данных любого типа.
- Алгоритм не проводит анализ входных данных поэтому не оптимален.

Список литературы

Ссылки

1. Алгоритм Лемпеля — Зива — Велча — Википедия (wikipedia.org)
2. Алгоритмы LZW, LZ77 и LZ78 / Хабр (habr.com)
3. Алгоритм LZW — Викиконспекты (ifmo.ru)
4. Метод LZW-сжатия данных (compression.ru)
5. 2 курс, лекция 22, Сжатие данных, LZW. - YouTube
6. сжатие форматы презентация, доклад (thepresentation.ru)
7. Реализация кодирования LZW (C) - Русские Блоги (russianblogs.com)
8. LZW coding 1 - YouTube
9. Теория кодирования | Демонстрация алгоритма сжатия LZW - YouTube
10. Техника сжатия LZW (Лемпель-Зив-Уэлч) - GeeksforGeeks
11. Алгоритм LZW [VMath]
12. Метод LZW-сжатия данных (algolist.ru)
13. Реферат Алгоритм Лемпеля Зива Велча (baza-referat.ru)
14. LZW compression - Rosetta Code
15. Lempel–Ziv–Welch - Wikipedia
16. LZW Data Compression | Mark Nelson
17. Lempel-Ziv-Welch (LZW) Encoding Discussion and Implementation (dipperstein.com)
18. НОУ ИНТУИТ | Лекция | Подстановочные или словарно-ориентированные алгоритмы сжатия информации. Методы Лемпела-Зива (intuit.ru)
19. Алгоритмы компрессии пиксельных изображений (compuart.ru)
20. LZW compression | What it is & its use in TIFF & PDF (prepressure.com)
21. LZW file compressor - C++ Articles (cplusplus.com)
22. Алгоритм LZW — Студопедия (studopedia.ru)
23. LZW (codecodex.com)
24. 2 курс, лекция 22, Сжатие данных, LZW. - YouTube
25. Algorithm Programming Some Topics in Compression - ppt video online download (slideplayer.com)
26. Простым языком о том, как работает сжатие файлов (tproger.ru)

27.11 (bspu.by)