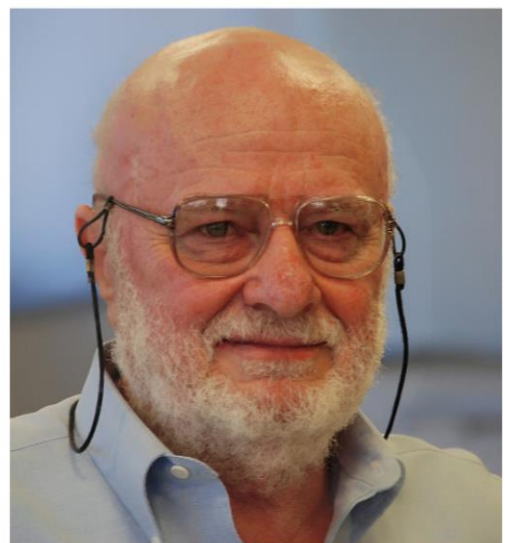


Алгоритм Лемпеля-Зива-Велча (Lzw)

Выполнил студент
Б9121-09.03.03ПИД
Лифарь Марина

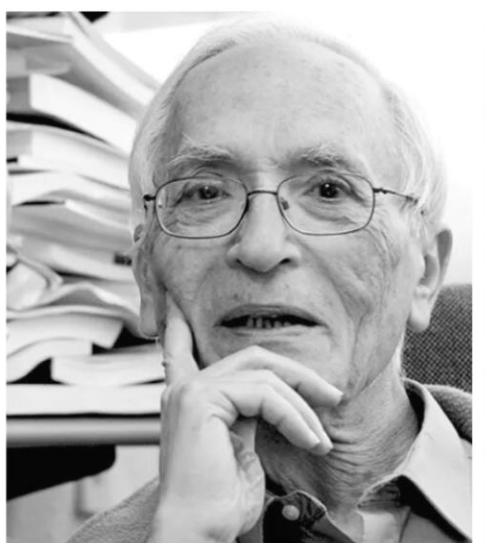
История

Метод сжатия LZW был предложен в 1978-ом году израильскими специалистами Лемпелом и Зивом



Авраам Лемпель

10 февраля 1936 г. (86 лет)



Яков Зив

27 ноября 1931 г. (91 год)

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.

Усовершенствованная версия алгоритма (Terry Welch) была представлена в 1984-ом году.



Терри Велч

20 января 1939 г.-22 ноября 1988 г. (49 лет)

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).



UNISYS

В настоящее время патент принадлежит компании Unisys. Алгоритм LZW определяется как часть стандарта ITU-T V.42bis, но Unisys установила жесткие условия лицензирования алгоритма для производителей модемов.

Unisys Corporation-американская глобальная информационно-технологическая компания, базирующаяся в Блу-Белл, штат Пенсильвания, и предоставляющая ИТ-услуги, программное обеспечение и технологии.

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.

LZW compression for string: “ABABBABCSABABBA”

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 = $14/9 = 1.56$).

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.

Пример

Кодирование

Пусть мы сжимаем последовательность «abacabadabacabae».

- Шаг 1: Тогда, согласно изложенному выше алгоритму, мы добавим к изначально пустой строке «а» и проверим, есть ли строка «а» в таблице. Поскольку мы при инициализации занесли в таблицу все строки из одного символа, то строка «а» 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 «е», 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 на прошлом слайде, 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

Пример.

oitomii o imi oooitmi o o o ooiimtomiiimotoim oi too i i m oio i omtoo toimo t iimiotmii
ttmimotoitoomt imo o t mii i i m o ooi o tom tototoi oto i moi t i o mimto toootoi otomtiao t
m iim toi m i ioaim ittoomooo i i tom t oto iimitimoi o tmi tmi otomi i too t tioot tim mii
oiooi tooimioomiootooooioo 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 ottoiooooimo too imtoimoom oom titoo i oiomotoii i i oto iiioi
i tioio too o m too tiotomtii oii o iii tom mot imt tiooi

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

содержащий 613613 символов может быть закодирован $3 \times 613 = 18393 \times 613 = 1839$ битами.

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

$3 \times 3 + 8 \times 4 + 16 \times 5 + 32 \times 6 + 64 \times 7 + (241 - 3 - 120) \times 8 = 1705$ бит

$1705/1839 \approx 0.927$, $1705/1839 \approx 0.927$, т.е. получили 88 % экономии.

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

$3 \times 3 + 8 \times 4 + 16 \times 5 + 32 \times 6 + 64 \times 7 + 128 \times 8 + (370 - 3 - 248) \times 9 = 2856$ бит $2856/(1052 \times 3) \approx 0.905$, $2856/(1052 \times 3) \approx 0.905$,

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.

Спасибо за внимание

