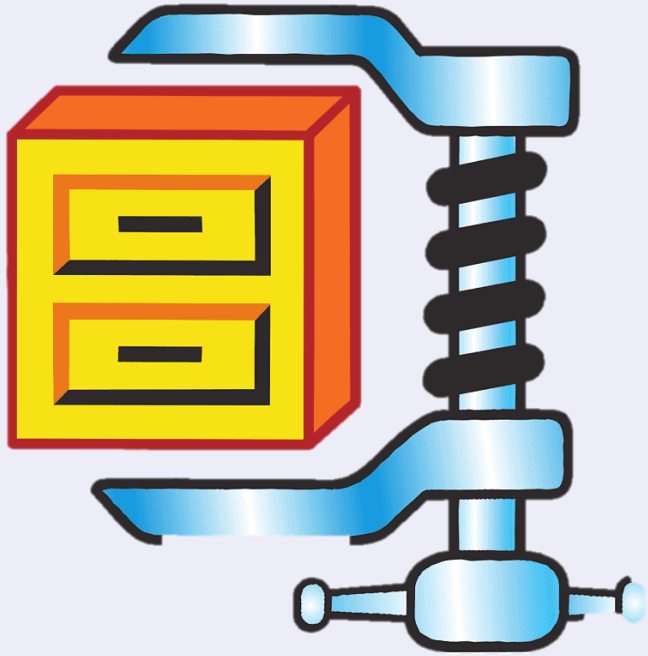
Алгоритм Лемпеля-Зива-Велча (Lzw)

Выполнил студен
Б9121-09.03.03ПИКД
Лифарь Марина

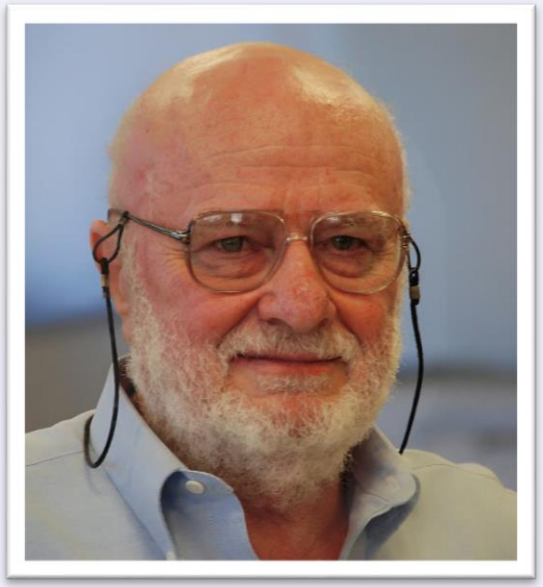
Введение

Сжатие файла — это уменьшение его размера при сохранении исходных данных. обычно делится на два основных типа: с потерями и без потерь
Формула : $Kc=(Vc/Vo)*100\%$

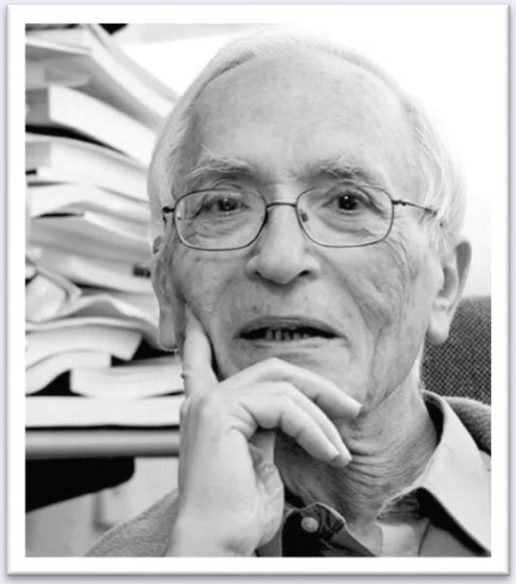


История

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



Авраам Лемпель
10 февраля 1936 г. (86 лет)



Яаков Зив
27 ноября 1931 г. (91 год)

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

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

Применение

Adobe® PostScript®

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

UNISYS

В настоящее время патент принадлежит компании Unisys.

Описание

Процесс сжатия выглядит следующим образом. Последовательно считываются символы входного потока и происходит проверка, существует ли в созданной таблице строк такая строка.

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

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	-	-

Кодирование

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.

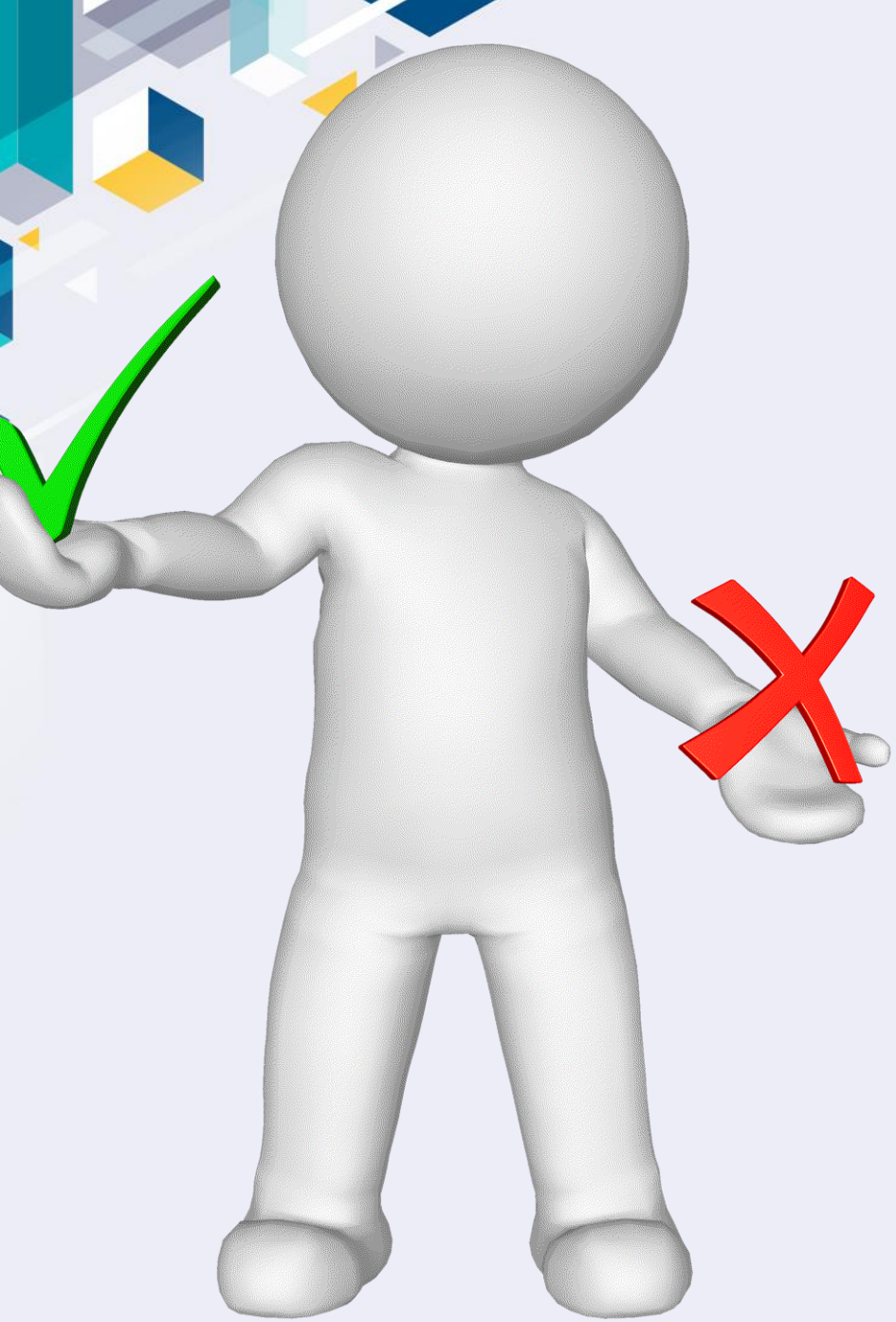
Данные		На выходе	Новая запись	
Биты	Код		Полная	Частичная
000	0	a	- -	5: a?
001	1	b	5: ab	6: b?
000	0	a	6: ba	7: a?
010	2	c	7: ac	8: c?
101	5	ab	8: ca	9: ab?
000	0	a	9: aba	10: a?
011	3	d	10: ad	11: d?
1001	9	aba	11: da	12: aba?
1000	8	ca	12: abac	13: ca?
0110	6	ba	13: cab	14: ba?
0100	4	e	14: bae	-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

Стоимость

Пример 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.

Заключение

LZW — алгоритм сжатия на основе "словаря". Это означает, что вместо сведения в таблицу количества символов и построения деревьев **LZW** кодирует данные, обращаясь к словарю

Результат работ выложен на GitHub
https://github.com/Marlifa/lzw.cpp.git