

Сжатие с учётом контекста. Словарные методы, где словарём является несжатый текст — семейство LZ77

Александра Игоревна Кононова

МИЭТ

26 января 2021 г. — актуальную версию можно найти на
<https://gitlab.com/illinc/otik>

Источник с памятью

Вероятность порождения элемента определяется предысторией источника → при сжатии необходимо учитывать контекст символа.

источник Маркова (N -го порядка) — состояние на i -м шаге зависит от состояний на N предыдущих шагах:

$i - 1, i - 2, \dots, i - N \rightarrow$ сжимаются не отдельные символы, а устойчивые сочетания — слова:
коды Зива—Лемпеля (LZ77, LZ78);

аналоговый сигнал — источник количественных данных Маркова 1-го порядка → кодирование длин повторений (Run Length Encoding, RLE).

При расчёте количества информации источника Маркова N -го порядка вероятности зависят от предыдущих N символов (непостоянны).

Размер байта – 4 бита

Для компактной иллюстрации ограничений алгоритмов примем, что для устройства «доска» байт (символ кодирования) составляет не 8 бит — октет (как для Intel x86/amd64), и не 6 бит (как для IBM 7030 Stretch), а 4 бита — тетраду, или одну 16-ричную цифру:

0 = 0000 = 0	8 = 1000 = 8
1 = 0001 = 1	9 = 1001 = 9
2 = 0010 = 2	A = 1010 = 10
3 = 0011 = 3	B = 1011 = 11
4 = 0100 = 4	C = 1100 = 12
5 = 0101 = 5	D = 1101 = 13
6 = 0110 = 6	E = 1110 = 14
7 = 0111 = 7	F = 1111 = 15

Символьная таблица (аналог ASCII) даётся отдельно в каждом примере, либо отсутствует.

Концепция RLE

Модель источника данных — Маркова первого порядка (аналоговый сигнал): вероятность символа зависит от предыдущего символа.

Run Length Encoding (RLE): вместо кодирования данных кодируются длины участков, на которых данные сохраняют неизменное значение.

AAAAAAAAABCCCC $\rightarrow 8 \times A, 1 \times B, 4 \times C$

Повторение символа c подряд L раз ($L \times c$) — цепочку длины L , $L \geq 1$ будем записывать как пару (L, c) .

Если c и L одного размера — это имеет смысл (приводит к сжатию цепочки) только при $L > 2$ ($L \geq 3$).

Основные варианты кодирования RLE

- 1 Наивный — любая цепочка $L \times c$ для любого L ($L \geq 1$) заменяется парой (L, c) : $\forall L : L \times c \rightarrow (L, c)$
- 2 Сжатые и несжатые цепочки (флаг-бит) — парой (L, c) заменяются только длинные цепочки $L \geq 3$, короткие цепочки из одного и двух символов объединяются в более длинные несжатые цепочки, каждая из которых также предваряется длиной L ;
для различения сжатых и несжатых цепочек выделяется один бит поля L :
$$L : \begin{cases} L \times c, L \geq 3 & \rightarrow (1 + L, c), \\ c_1 \dots c_L, L \geq 1 & \rightarrow (0 + L, c_1 \dots c_L). \end{cases}$$
- 3 Префикс сжатой цепочки в несжатом тексте — текст записывается как есть, длинные цепочки заменяются **тройками** (p, L, c) , где $p \in A$;
сжатие — только при $L \geq 4$:
$$\begin{cases} L \times c, L \geq 4 & \rightarrow (p, L, c), \\ c_1 \dots c_L, L \geq 1 & \rightarrow c_1 \dots c_L. \\ L \times p, L \geq 1 & \rightarrow (p, L, p). \end{cases}$$

Префикс p выбирается как самый редкий символ текста.

Сравнение (без коррекции длины) I

Исходные сообщения:

- AAAAAAAAAAAAAAAAAABB (18) — хороший случай ($16 \times A$ и $2 \times B$)
- ABCDDEF0012 (12) — нормальный случай
- 0123456789AAAAAAAAABCEDEF (23) — нормальный случай ($8 \times A$)
- 0123456789ABCEDEF (16) — наихудший случай

1 Наивный вариант:

AAAAAAAAAAAAAAAAABB \rightarrow $\begin{bmatrix} \text{FA1A2B} & (6) \\ \text{AFA1B2} & (6) \end{bmatrix}$ для наивного RLE допустимы оба

порядка полей — (L, c) или (c, L) ; далее используется (L, c) для единообразия

AABCDDEF0012 \rightarrow 2A1B1C2D1E1F201112 (18)

0123456789AAAAAAAAABCEDEF \rightarrow 101112131415161718198A1B1C1D1E1F (32)

0123456789ABCEDEF \rightarrow 101112131415161718191A1B1C1D1E1F (32)

Увеличение объёма в наихудшем случае — в 2 раза.

Длинные цепочки встречаются редко, поэтому поле L обычно равно по длине символу c (однобайтовое)

Сравнение (без коррекции длины) II

2 Флаг-бит: $1 \leq L \leq 7$ (3 бита), $1 + 7 \sim 1111 = F$, $1 + 2 \sim 1010 = A$

AAAAAAAAAAAAAAAAABBB \rightarrow $\begin{cases} \text{AAAAAAAA, AAAAAAAAA, ABBB} \rightarrow \text{FAFA4AABB} & (9) \\ \text{AAAAAAAA, AAAAAAAAA, AA, BB} \rightarrow \text{FAFAAAAB} & (8) \end{cases}$

но AABCDDEF0012 \rightarrow $\begin{cases} \text{AABCDDE, F0012} \rightarrow \text{7AABCDDE5F0012} & (14) \\ \text{AA, BC, DD, EF, 00, 12} \rightarrow \text{AA2BCAD2EFA0212} & (15) \end{cases}$

0123456789AAAAAAAAABCDEF \rightarrow 0123456, 789, AAAAAAA, ABCDEF \rightarrow
 \rightarrow 701234563789FA6ABCDEF (21)

0123456789ABCDEF \rightarrow 0123456, 789ABCD, EF \rightarrow 701234567789ABCD2EF (19)

здесь порядок полей (L, c) важен: иначе не декодируются несжатые цепочки

Увеличение объёма в наихудшем случае — для байта-тетрады на $\frac{1}{7}$ объёма файла (к каждому семи символам добавляется восьмой — флаг-бит + L), для байта-октета — на $\frac{1}{127}$ (менее процента).

Сравнение (без коррекции длины) III

3 Префикс в несжатом тексте, $p = E$: AAAAAAAAAAAAAAAAABB \rightarrow EFAABB (6)

Два варианта кодирования одиночного символа p : $1 \times p \rightarrow \begin{cases} (p, 1, p) \\ (p, 0) \end{cases}$

0123456789AAAAAAAABCDEF $\rightarrow \begin{cases} 0123456789\text{E8ABCDE1EF} & (20) \\ 0123456789\text{E8ABCDEOF} & (19) \end{cases} L \neq 0$

но только один — для двойного p : ...DEEF \rightarrow ...DE2EF (а не ...DEOEOF).

Если E1E, то допустимо и $(p, L, c) \sim E1E/E8A$, и $(p, c, L) \sim EE1/EA8$.

Но если мы хотим E0, то только (p, L, c) .

0123456789ABCDEF $\rightarrow \begin{cases} 0123456789ABCDE1EF & (18) \\ 0123456789ABCDEOF & (17) \end{cases}$

Увеличение объёма в наихудшем случае (все символы одиночные) при $(p, 0)$ — на $\nu(p)$ от объёма файла, где $\nu(p)$ — частота p : для байта-тетрады $\nu(p) \leq \frac{1}{16}$, для байта-октета $\nu(p) \leq \frac{1}{256} < 0,4\%$ (p — самый редкий символ).

Коррекция длины

Максимальная длина L определяется разрядностью поля (обычно байта), минимальная — алгоритмом:

Байт	Наивный	Флаг-бит	Префикс
N бит	$1 \leq L < 2^N$	$3 \leq L < 2^{N-1}$ (сж) $1 \leq L < 2^{N-1}$ (н/сж)	$4 \leq L < 2^N$ ($c \neq p$) $2 \leq L < 2^N$ ($c = p$)
4 бита	$1 \leq L \leq 15$	$3 \leq L \leq 7$ (сж) $1 \leq L \leq 7$ (н/сж)	$4 \leq L \leq 15$ ($c \neq p$) $2 \leq L \leq 15$ ($c = p$)
8 бит	$1 \leq L \leq 255$	$3 \leq L \leq 127$ (сж) $1 \leq L \leq 127$ (н/сж)	$4 \leq L \leq 255$ ($c \neq p$) $2 \leq L \leq 255$ ($c = p$)

Коррекция: вместо L записывается $\tilde{L} = L - \min L$

4 бита	$1 \leq L \leq 16$	$3 \leq L \leq 10$ (сж) $1 \leq L \leq 8$ (н/сж)	$4 \leq L \leq 16$ ($c \neq p$) $2 \leq L \leq 16$ ($c = p$)
	$0 \leq \tilde{L} \leq 15$	$0 \leq \tilde{L} \leq 7$	$1 \leq \tilde{L} \leq 15$

Цепочки с коррекцией L длиннее \Rightarrow их количество в файле меньше \Rightarrow файл после кодирования короче.

Сравнение (с коррекцией длины)

1 Наивный: $\forall L : L \times c \rightarrow (L - 1, c)$

AAAAAAAAAAAAAAAABBB $\rightarrow 16 \times A, 2 \times B \rightarrow$ FA1B (4)

2 Флаг-бит:
$$\begin{cases} L \times c, L \geq 3 & \rightarrow (1 + (L - 3), c), \\ c_1 \dots c_L, L \geq 1 & \rightarrow (0 + (L - 1), c_1 \dots c_L). \end{cases}$$

AAAAAAAAAAAAAAAABBB $\rightarrow 10 \times A, 6 \times A, BB \rightarrow$ FABA1BB (7)

0123456789AAAAAAAABCDEF $\rightarrow 01234567, 89, \text{AAAAAAAA}, BCDEF \rightarrow$
 $\rightarrow 701234567189DA4BCDEF$ (20)

0123456789ABCDEF $\rightarrow 01234567, 89ABCDEF \rightarrow 701234567789ABCDEF$ (18)

3 Префикс:
$$\begin{cases} L \times c, c \neq p, L \geq 4 & \rightarrow (p, L - 1, c) \\ L \times p, L \geq 2 & \rightarrow (p, L - 1, p), L - 1 \neq 0 \\ 1 \times p & \rightarrow (p, 0), \end{cases}$$

AAAAAAAAAAAAAAAABBB $\rightarrow 16 \times A, BB \rightarrow$ EFCBB (5)

0123456789AAAAAAAABCDEF $\rightarrow 0123456789E74BCDEOF$ (19)

0123456789ABCDEFEF $\rightarrow 0123456789ABCDEOF$ (17)

Код Зива–Лемпеля, LZ77/LZ1

1977 г., Якоб Зив (Jacob Ziv) и Абрахам Лемпель (Abraham Lempel):
если цепочка символов (не обязательно одинаковых) встречается
более одного раза, то каждое следующее вхождение цепочки
заменяются ссылкой на предыдущее;
ссылка состоит из относительного смещения $S \geq 1$ и длины L
цепочки **в несжатом тексте**: (S, L)

Поиск предыдущего вхождения — основная сложность алгоритмов
семейства LZ77 и неоднозначность кодирования.

Алгоритм LZSS (реализация LZ77) — дерево для ускорения поиска.

Скользящее окно: область перед текущей позицией кодирования,
в которой можем искать и адресовать ($w \leq \max(S)$) ссылки
Если S записывается N битами, то максимальный размер окна
 $w = \max(S) = 2^N - 1$.

Alternative vexillum codicis inf. interpretatio (AVCII): \square , абвгдеиклмнорты
0123456789ABCDEF

012345678901234567890123456789012345678901234567890123

там_корабли_лавировали, _лавировали, _да_не_вылавировали

$$\begin{cases} S_1 = 23 - 11 = 12 = C, \\ L_1 = 13 = D \end{cases}$$
 — пробел №23 принадлежит обоим цепочкам

012345678901234567890123456789012345678901234567890123

там_корабли_лавировали, _лавировали, _да_не_вылавировали

$$\begin{cases} S_2 = 44 - 24 = 20 > 15 = F, \\ L_2 = 10 = A \end{cases}$$

Простейший вариант кодирования — S и L по одному байту, тогда $1 \leq S \leq 15$ и $1 \leq L \leq 15$; максимальное окно $w = \max S = 15$.

При программной реализации (поиск только в окне) будет найдено только одно совпадение: (S_1, L_1) . Возможен случай, когда адресуемых повторов вообще нет, даже односимвольных — как увеличить $\max S$?

там_корабли_лавировали, (S_1, L_1) да_не_вылавировали

$(S_1, L_1) = CD = \text{op}$ — как отличить ссылку от текста?

Основные варианты кодирования LZ77

- 1 Наивный — ссылки чередуются с несжатыми символами: цепочка, за которой следует c — тройка (S, L, c) , несжатый символ c — тройка $(0, 0, c)$. При необходимости текст дополняется (обычно нулями)
- 2 Новопримитивный: ссылка (L, S) , несжатый символ c как $(0, c)$.
- 3 LZSS (Сторер, Сжимански): флаг-бит + символ c /ссылка (S, L) .
- 4 Флаг-байт: символы c и ссылки (S, L) группируются по 8 (по числу бит в байте-октете) штук, каждая группа предваряется байтом, где каждый бит показывает тип соответствующего объекта:

$$\begin{cases} 0, & \text{несжатый символ — байт } c \\ 1, & \text{ссылка — два байта } (S, L), 0 < S \leq w, L \geq 3 \end{cases}$$
 для байта-тетрады группируется, соответственно, по 4 объекта
- 5 Префикс p как маркер ссылки: $\begin{cases} \text{ссылка } (p, S, L) : 0 < S \leq w, L \geq 4, \\ \text{символ } p \rightarrow (p, 0) \end{cases}$

□, абвгдеиклмнорты |S| = |L| = |c|, окно $w = 15$ символов (макс.)
0123456789ABCDEF

0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF012345
012345678901234567890123456789012345678901234567890123
там□корабли□лавиrowали, □лавиrowали, □да□не□вылавиrowали (54)

- 1 Наивный: 00т00а00м00□00к00о00р610600л00и81л61в51рC1вC1лB1,
CDдC1□00н00е61в00ыC1a41и00р00о81a81и (84)
Порядок полей допустим любой — и (S, L) , и (L, S) — здесь (S, L) .
- 2 Новопрimitивный: 0т0а0м0□0к0о0р0а060л0и0□0л0а0в0и0р0о0в0а0
2В0, DC0д0а0□0н0е0□0в0ы0л0а0в0и0р0о0в0а0л0и (82), только (L, S)
- 3 Флаг-байт: 0там□0кора0бли□0лави0рова8ли, CD0да□н0е□вы0лави
0рова0ли00 (55) Допустимы и (S, L) , и (L, S) — в статье SS (S, L) .
- 4 Префикс $p = F = \text{ы}$:
там□корабли□лавиrowали, ыCDда□не□вы0лавиrowали (45)
Префикс p — в начале. Разный порядок полей S и L приводит
к разным способам экранирования символа p . Здесь (p, S, L) .

Увеличение разрядности S

- 1 Записывать S двумя байтами. Для байта-тетрады $S \leq 2^8 - 1 = 255$; для байта-октета $S \leq 2^{16} - 1 = 65\,535$, окно w обычно дополнительно ограничивается ($w \ll \max S$) для ускорения поиска.

Префикс: ссылка занимает 4 символа, тогда $L \geq 5$:

- $\begin{cases} \text{ссылка } (S, L) \rightarrow (p, \text{lo } S, \text{hi } S, L), \\ \text{символ } p \rightarrow (p, 0, 0), \text{ экранируется } S = 0 \end{cases}$
- $\begin{cases} \text{ссылка } (S, L) \rightarrow (p, L, \text{lo } S, \text{hi } S), \\ \text{символ } p \rightarrow (p, 0), \text{ экранируется } L = 0 \end{cases}$

- 2 Добавить к S бит за счёт поля L (пара (S, L) — ровно 2 символа).

Префикс: ссылка занимает 3 символа, $L \geq 4$:

- $\begin{cases} \text{ссылка } (S, L) \rightarrow (p, \text{lo } S, \text{hi } S + L), \\ \text{символ } p \rightarrow (p, 0, 0), \text{ экранируется } S = 0 \end{cases}$
- $\begin{cases} \text{ссылка } (S, L) \rightarrow (p, L + \text{hi } S, \text{lo } S), \\ \text{символ } p \rightarrow (p, 0), \text{ экранируется } L = 0 \end{cases}$

Коррекция S и L

- 1 Наивный: $S \geq 0, L \geq 0 \Rightarrow$ ни S , ни L невозможно скорректировать.
- 2 Флаг-байт: $S \geq 1$ всегда; если ссылка — k символов, то $L \geq k + 1$;
значения 0 не используются как специальные:
$$\begin{cases} S \rightarrow \tilde{S} = S - 1, \\ L \rightarrow \tilde{L} = L - k - 1. \end{cases}$$
- 3 Префикс: $S \geq 1$ всегда; $L \geq k + 1$ (k — длина ссылки, включая префикс):
 - если для экранирования символа-префикса используется $S = 0$,
корректируется L : $L \rightarrow \tilde{L} = L - k - 1 \quad (\tilde{L} \geq 0)$.
 - если для экранирования символа-префикса используется $L = 0$,
корректируются обе:
$$\begin{cases} S \rightarrow \tilde{S} = S - 1 & (\tilde{S} \geq 0), \\ L \rightarrow \tilde{L} = L - k & (\tilde{L} \geq 1). \end{cases}$$

Увеличиваем S за счёт L :
$$\begin{cases} |\tilde{S}| = |c| + 1 : & 0 \leq \tilde{S} \leq 31 \\ |\tilde{L}| = |c| - 1 : & 1 \leq \tilde{L} \leq 7 \end{cases} \quad \begin{array}{l} \text{для} \\ \text{октета:} \end{array} \begin{cases} 0 \leq \tilde{S} \leq 511 \\ 1 \leq \tilde{L} \leq 127 \end{cases}$$

префикс $p = F = \text{ы}$,
$$\begin{cases} \text{ссылка } k = 3 \text{ символа } (p, \tilde{L} + \text{hi } \tilde{S}, \text{ lo } \tilde{S}), \tilde{L} \geq 1, \\ p \rightarrow (p, 0), \end{cases}$$

максимальная коррекция:
$$\begin{cases} \tilde{S} = S - 1 : & 1 \leq S \leq 32, \\ \tilde{L} = L - k = L - 3 : & 4 \leq L \leq 10. \end{cases} \quad \begin{array}{l} \text{для} \\ \text{октета:} \end{array} \begin{cases} 1 \leq S \leq 512 \\ 4 \leq L \leq 130 \end{cases}$$

окно $w = 32$:

там_корабли_лавировали, лавировали, да_не_вылавировали

$$\begin{cases} S_1 = 12 \rightarrow \tilde{S}_1 = 11 \sim 0 \ 1011 \\ L_1 = 10 \rightarrow \tilde{L}_1 = 7 \sim 111 \end{cases} \rightarrow \begin{matrix} 0111 & 1011 \end{matrix}$$

012345678901234567890123456789012345678901234567890123

там_корабли_лавировали, лавировали, да_не_вылавировали

$$\begin{cases} S_2 = 44 - 12 = 32 \rightarrow \tilde{S}_2 = 31 \sim 1 \ 1111 \\ L_2 = 10 \rightarrow \tilde{L}_2 = 7 \sim 111 \end{cases} \rightarrow \begin{matrix} 1111 & 1011 \end{matrix}$$

там_корабли_лавировали, ы7Ви, да_не_вы0ыВВ (41)

Для октета можно увеличить S на два бита:
$$\begin{cases} 0 \leq \tilde{S} \leq 1023 \\ 1 \leq \tilde{L} \leq 63 \end{cases} \Rightarrow \begin{cases} 1 \leq S \leq 1024 \\ 4 \leq L \leq 66 \end{cases}$$

Сравнение RLE и LZ77

Как RLE, так и LZ77 не требуют отдельного словаря.

- 1 Повторение одинаковых символов: Префикс $p = E$, без коррекции

RLE: $AAAAAAAA \sim 8 \times A$

0123456789 **AAAAAAAA**BCDE**F** \rightarrow 0123456789**E8A**BCDE**OF** $23 \rightarrow 19$

LZ77: $\underbrace{AAAAAAAA}_{(p, L, S)} \sim A + (S = 1, L = 7)$ Порядок (p, L, S)

0123456789**A****AAAAAAA**BCDE**F** \rightarrow 0123456789**AE71**BCDE**OF** $23 \rightarrow 20$

- 2 Повторение подстрок общего вида:

RLE: $28 \rightarrow 24$

012345601234789**AAAAAAAA**BCDE**F** \rightarrow 012345601234789**E8A**BCDE**OF**

LZ77: $28 \rightarrow 23$

0123456**01234**789**A****AAAAAAA**BCDE**F** \rightarrow 0123456**E57**789**AE71**BCDE**OF**

Для большинства файлов LZ77 эффективнее RLE.

Спасибо за внимание!

МИЭТ

<http://miet.ru/>

Александра Игоревна Кононова

illinc@mail.ru

Задача №1

- 1 Рассчитать количество информации в сообщении $C = \underbrace{\text{«ля-ля-ля-...-ля»}}_{80 \text{ раз}}$ (кодировка koi8-r)
- 2 Закодировать сообщение C алгоритмом LZ77

Информация и модели источника

Заменяя вероятности символов на их оценки по модели X , получаем **оценку** количества информации:

- 1 Без памяти, $A_1 = \{\text{л}, \text{я}, -\}$:

$$I_1 = 2 \cdot 80 \cdot \left(-\log_2 \left(\frac{80}{239} \right) \right) + 79 \cdot \left(-\log_2 \left(\frac{79}{239} \right) \right) = 254,2 \text{ бит} = 31,8 \text{ байт}$$

- 2 Без памяти, $A_1 = \{\text{ля-}, \text{ля}\}$:

$$I_2 = 79 \cdot \left(-\log_2 \left(\frac{79}{80} \right) \right) + 1 \cdot \left(-\log_2 \left(\frac{1}{80} \right) \right) = 7,6 \text{ бит} = 0,9 \text{ байт}$$

- 3 Маркова, $A_1 = \text{коi8-r}$, $N = 3$ со следующими оценками вероятностей:

$$\begin{aligned} p(c_1 = \text{л}) &= p(c_2 = \text{я}) = p(c_3 = -) = \frac{1}{256}, & p(-|\text{ля}) &= \frac{79}{80}, \\ p(\text{л}|\text{ля-}) &= p(\text{я}|\text{я-л}) = 1, & p(\text{eof}|\text{-ля}) &= \frac{1}{80}: \end{aligned}$$

$$I_3 = 3 \cdot \log_2 256 + 79 \cdot \left(-\log_2 \left(\frac{79}{80} \right) \right) + 1 \cdot \left(-\log_2 \left(\frac{1}{80} \right) \right) = 31,6 \text{ бит} = 3,9 \text{ байт}$$

Задача №2

Используя 4-битный байт, закодируйте:

- 1 Сообщение (64 = 0x40 символов) — LZ77:

	0123456789ABCDEF	0123456789ABCDEF	0123456789ABCDEF	0123456789ABCDEF
00	косил косой косарь	за косарь косой	косой косой космеи	на косе при осе.

код i	0123456789ABCDEF
AVCII: a_i	.aeзийклмнопрсь
$\nu(a_i)$	B1431338111C1392

а) S занимает 1 байт (4 бита); б) S занимает 5 бит; в) S занимает 2 байта.

- 2 16-цветный рисунок 8×8 — RLE, LZ77.