Сжатие с учётом контекста. Метод кодирования длин повторений (Run Length Encoding, RLE). Словарные методы, где словарём является несжатый текст — семейство LZ77

Александра Игоревна Кононова

МИЭТ

30 сентября 2025 г. — актуальную версию можно найти на `https://gitlab.com/illinc/otik`

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 $= 1111 = 15$

Далее основной единицей измерения является символ=байт, а не бит.

Количество бит в байте обозначим k.

Методы кодирования, коды и алгоритмы с учётом контекста

Далее рассматриваются **методы сжатия с учётом контекста:** RLE, LZ77 и LZ78.

1. Оптимальный код источника Маркова, где $|code(c_1 \dots c_n)| \to I(c_1 \dots c_n)$ (Хф/АС с усл. вер.) не используется как метод сжатия: даже для 1 порядка нужны $256^2 = 65536$ частот.

2. Если метод Хаффмана однозначно лучше Шеннона—Фано и тем более Шеннона, то для сжатия с учётом контекста **нет однозначно лучшего метода.**

3. Для заданного метода — много кодов, **нет однозначно лучшей реализации.**

   Есть файлы, которые «наивный» RLE сжимает лучше и быстрее всего.

4. Среди схожих реализаций одного метода однозначно **худшей** (но обычно более наглядной) является та, где часть значений недопустима. «Наивный» RLE (L, c) хуже, чем $(L-1, c)$.

5. Для сложных кодов RLE и **любого кода LZ77 кодирование принципиально неоднозначно** (разные алгоритмы $\implies$ разные длины и скорости). Декодирование — однозначно.

Семейство RLE (Run Length Encoding) — концепция

Модель источника данных — Маркова первого порядка (аналоговый сигнал), при этом:

$$\forall a \neq b\colon \begin{cases} p(a|a) = p(b|b) = r, \\ p(a|b) = p(b|a) = s, \end{cases} \quad r \gg (T-1)s, \quad \text{где } T \text{ — размер алфавита.} \tag{1}$$

Run Length Encoding (RLE): $\mathrm{AAAAAAAABCCCC} \to 8 \times \mathrm{A},\ 1 \times \mathrm{B},\ 4 \times \mathrm{C}$

Повторение символа c подряд L раз $(L \times c)$ — цепочку длины L, $L_{\min} \leqslant L \leqslant L_{\max}$ — будем записывать как пару $\begin{Bmatrix} L \\ c \end{Bmatrix}$ (сжатая цепочка):

- цепочки длины более $L_{\max}$ символов — делятся на несколько;
- последовательности символов, где ни один не повторяется $L_{\min}$ раз подряд — несжатый текст.

RLE — не код, а **семейство кодов,** основанных на одном принципе сжатия и похожих моделях источника (одна формула (1), разные r и s): 1 ни один не оптимален для своего источника;
2 $L_{\min}$ вычисляется для конкретного варианта из соотношения длин кодов сж/несж;
3 $L_{\max}$ — из $L_{\min}$ и способа кодирования L (код L обозначим $\widetilde{L} = L - \Delta_L$).

Подсемейства RLE — по способу отделения сжатых цепочек от несжатого текста

1. Несжатого текста нет: $L \geqslant 1$ (то есть $L_{\min} = 1$) — **«наивная» реализация RLE,** RLE-н
   - несжатого текста нет $\implies$ пару $\{L, c\}$ можно записывать парой байтов;
   - порядок этих байтов неважен: $(\widetilde{L}, c)$ или $(c, \widetilde{L})$.

   Для прочих RLE: • $L \geqslant 2$, • есть несжатый текст (любые сочетания любых байтов).

2. Несжатый текст группируется в цепочки $\{L, c_1 \ldots c_L\}$, сж/несж различаются флаг-битом θ (θ и $\widetilde{L}$ помещаются в один байт, обозначим его $\widetilde{L}_\theta = \widetilde{L} \cup \theta$) — **RLE с флаг-битом,** RLE-θ
   - байт $\widetilde{L}_\theta$ всегда первый, чтобы прочесть θ и отличить c от $c_1 \ldots c_L$;
   - на $\widetilde{L}$ остаётся только $k-1$ бит;
   - для несжатой цепочки $L^{\text{несж}} \geqslant 1$ — файлы вида `B`, `A...AABCC..CC`: $L^{\text{несж}}_{\min}$ и $L^{\text{сж}}_{\min}$ разные.

3. Несжатый текст записывается как есть (кроме $c_i = p$: они экранируются), сжатые цепочки предваряются односимвольным префиксом p — **RLE с префиксом,** RLE-p
   - код сж. цепочки длины L состоит из трёх символов $(p, \widetilde{L}, c)$ $\implies$ для сжатия $L \geqslant 4$.

«Наивный» RLE ($L_{\min} = 1$), две реализации

Код K1, опции: ❶ порядок $(\widetilde{L}, c)$: два последовательных байта $b_1 = \widetilde{L}, b_2 = c$; ❷ смещение $\Delta_L = 0$ — увеличивает размер кода, не ускоряет кодирование, зато наглядно $(\widetilde{L} = L)$.

Следствие из опций: $L_{\max} = \max(\widetilde{L}) = 2^k - 1$, с учётом $k = 4$: $L_{\max} = 15$

$C =$ 0101 2222 2222 2222 2222 3453 3333 3367 89AB CDEF (40 байтов)

$K1(C) =$ 1011 1011 F212 1314 1573 1617 1819 1A1B 1C1D 1E1F (40 байтов)

Код K2, опции: ❶ порядок $(\widetilde{L}, c)$; ❷ смещение $\Delta_L = 1$ $(\widetilde{L} = L - 1)$.

Следствие из опций: $L_{\max} = \max(\widetilde{L}) + \Delta_L = 2^k - 1 + 1 = 2^k$, с учётом $k = 4$: $L_{\max} = 16$

$K2(C) =$ 0001 0001 F203 0405 6306 0708 090A 0B0C 0D0E 0F (38 байтов)

В лучшем случае код K2 вдвое короче кода K1 (фрагмент из шестнадцати 2: F2 vs F212), в худшем случае K2 не длиннее K1 (00 vs 10) — и то, и то при **любом** k.

Время кодирования/декодирования K2 такое же, как у K1 — аналогично, при **любом** k.

Схема данных кодирования «наивной» реализацией метода RLE

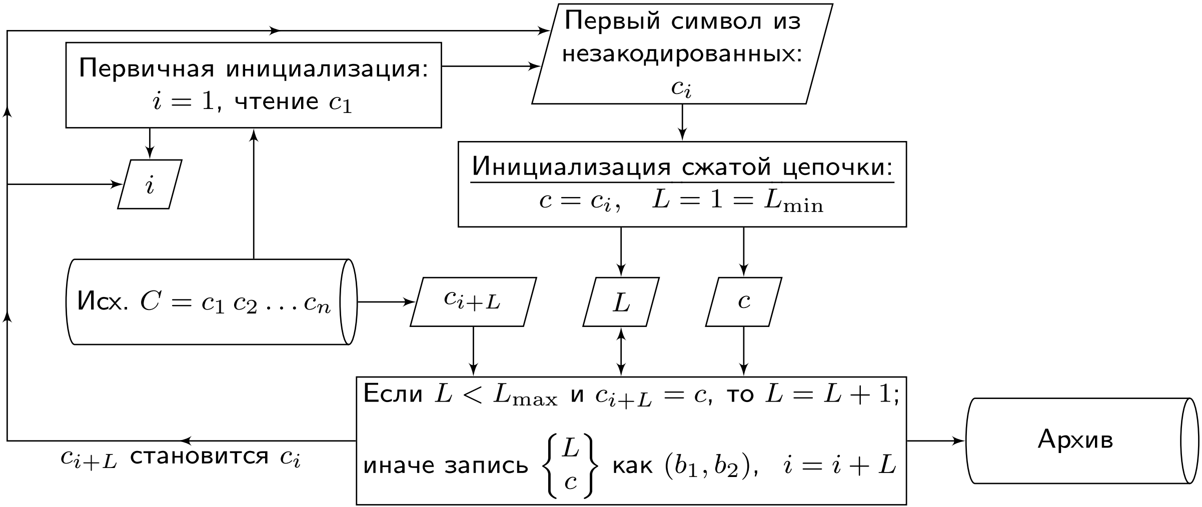
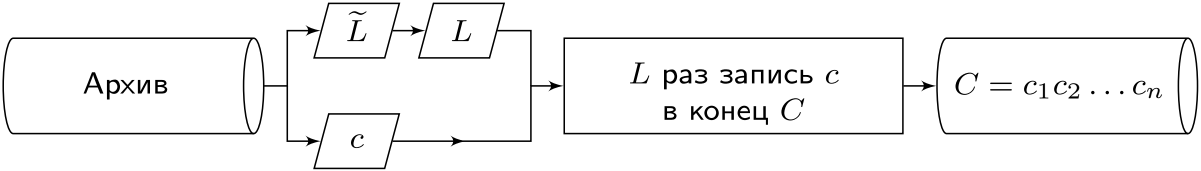


Схема данных декодирования «наивной» реализации метода RLE



RLE с флаг-битом сжатая/несжатая цепочка

Всегда: • $L_{\min}^{\text{несж}} = 1$; • порядок $(\widetilde{L}_\theta, c)/(\widetilde{L}_\theta, c_1 \ldots c_L)$; • $\max(\widetilde{L}) = 2^{k-1} - 1$. Далее опции:

1. Флаг-бит θ сж/несж может быть $0/1$ или $1/0$ — не влияет на длину кода.
2. Положение θ в байте $\widetilde{L}_\theta$ — не влияет на длину кода; обычно θ — старший: удобнее читать дамп.
3. Выбор $L_{\min}^{\text{сж}}$ — между 2 и 3:
   - по длине кода — непредсказуемо: ср. файлы 001122 и 0112;
   - по скорости: кодирование $L_{\min}^{\text{сж}} = 2$ немного быстрее, декодирование одинаково;
4. Смещение Δ_L — от 0 до $L_{\min}$ (так как $L_{\min}^{\text{сж}} \neq L_{\min}^{\text{несж}}$, то и Δ_L могут различаться для сж/несж).

Следствие из опций: расчёт $L_{\max} = \max(\widetilde{L}) + \Delta_L$. При $\Delta_L^{\text{сж}} \neq \Delta_L^{\text{несж}}$ получим и $L_{\max}^{\text{сж}} \neq L_{\max}^{\text{несж}}$.

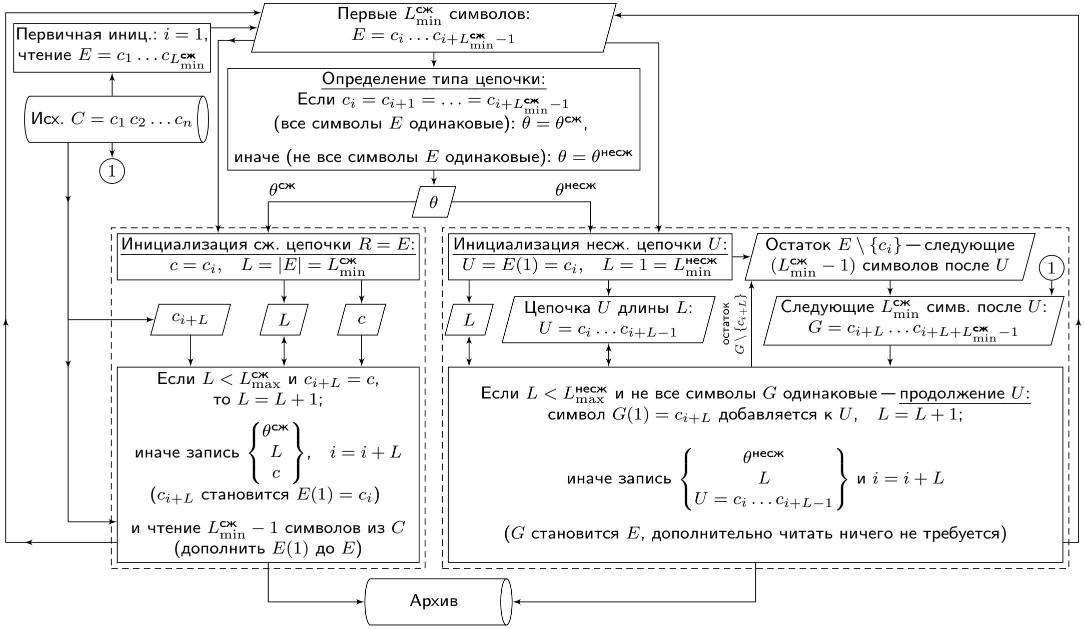
Код K3: 1 θ сж/несж может быть $0/1$; 2 θ старший в байте $\widetilde{L}_\theta$; 3 $L_{\min}^{\text{сж}} = 2$; 4 $\Delta_L^{\text{сж}} = \Delta_L^{\text{несж}} = 0$.

Тогда для кода K3: $L_{\max}^{\text{сж}} = L_{\max}^{\text{несж}} = 2^{k-1} - 1$, с учётом $k = 4$ получим $L_{\max}^{\text{сж}} = L_{\max}^{\text{несж}} = 7$.

Код K4: 1–3 как A; 4 $\begin{cases} \Delta_L^{\text{сж}} = L_{\min}^{\text{сж}} = 2, \\ \Delta_L^{\text{несж}} = L_{\min}^{\text{несж}} = 1 \end{cases}$ (максимальное смещение).

Для K4: $\begin{cases} L_{\max}^{\text{сж}} = 2^{k-1} - 1 + 2 = 2^{k-1} + 1, \\ L_{\max}^{\text{несж}} = 2^{k-1} - 1 + 1 = 2^{k-1}; \end{cases}$ с учётом $k = 4$ получим $L_{\max}^{\text{сж}} = 9$ и $L_{\max}^{\text{несж}} = 8$.

Схема данных кодирования RLE с флаг-битом сжатая/несжатая цепочка



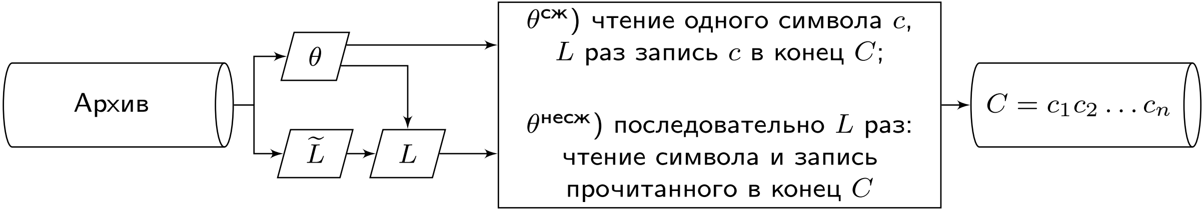
Кодирование К3 сообщения $C = $ 0101 2222 2222 2222 2222 3453 3333 3367 89AB CDEF

1. $i = 1$) первые $L^{\text{сж}}_{\min} = 2$ (два) символа: 01 не все одинаковы $\implies$ инициализация несжатой $(\theta = 1)$ цепочки $U = 0$ длины $L = L^{\text{несж}}_{\min} = 1$ (один) символ, остаток 1;
2. $i = 1, \theta = 1, L = 1, U = 0$) остаток 1 + следующий 0 = следующие за U два символа: $G = 10$ не все одинаковы $\implies$ первый из них $(c_{i+L} = 1)$ в U, второй (0) — остаток;
3. $i = 1, \theta = 1, L = 2, U = 01$) следующие два: $G = 01$ не все одинаковы $\implies$ 0 в U;
4. $i = 1, \theta = 1, L = 3, U = 010$) следующие два: $G = 12$ не все одинаковы $\implies$ 1 в U;
5. $i = 1, \theta = 1, L = 4, U = 0101$) следующие два: $G = 22$ все одинаковы $\implies$
   - запись текущей $\{\theta = 1, L = 4, U = 0101\}$ $\implies$ новое $i = i + L = 5$;
   - инициализация сжатой $(\theta = 0)$ цепочки из $c = 2$ длины $L = L^{\text{сж}}_{\min} = 2$;
6. $i = 5, \theta = 0, L = 2, c = 2$) следующий символ: $c_{i+L} = c_7 = 2$ совпадает с c $\implies$ $L = L + 1 = 3$;
7. $i = 5, \theta = 0, L = 3, c = 2$) следующий символ: $c_{i+L} = c_8 = 2$ совпадает с c $\implies$ $L = L + 1 = 4$;

...

8. $i = 5, \theta = 0, L = 6, c = 2$) следующий символ: $c_{i+L} = c_{11} = 2$ совпадает с c $\implies$ $L = L + 1 = 7$ это $L^{\text{сж}}_{\max}$ $\implies$ запись текущей $\{\theta = 0, L = 7, c = 2\}$...

Схема данных декодирования RLE с флаг-битом сжатая/несжатая цепочка



RLE с односимвольным префиксом

Всегда: • $L_{\min}^{c\neq p} = 4$; • порядок $(p, \widetilde{L}, c)$; • $\widetilde{L} \neq 0$; • экранирование p как $(p, 0)$; • $\max(\widetilde{L}) = 2^k - 1$.

1. Выбор $L_{\min}^{c=p}$ — от 1 до $L_{\min}^{c\neq p} = 4$:
   - по длине кода — непредсказуемо;
   - по скорости: кодирование $L_{\min}^{c=p} = L_{\min}^{c\neq p} = 4$ немного быстрее, декодирование одинаково;
2. Алгоритм выбора префикса p из множества самых редких байтов.
   Конкретное значение p — не опция *кода*, а характеристика *файла* C, сохраняется в архив.
3. Смещение Δ_L — от 0 до $L_{\min} - 1$ ($\widetilde{L} = L - \Delta_L$ должен быть ненулевым);
   так как $L_{\min}^{c=p}$ и $L_{\min}^{c\neq p}$ могут различаться, то и Δ_L могут различаться для $c = p$ и $c \neq p$.

Следствие из опций: расчёт $L_{\max} = \max(\widetilde{L}) + \Delta_L$. При $\Delta_L^{c=p} \neq \Delta_L^{c\neq p}$ получим и $L_{\max}^{c=p} \neq L_{\max}^{c\neq p}$.

Код K5: (1) $L_{\min}^{c=p} = L_{\min}^{c\neq p} = 4$; (2) p — наименьший по значению из подходящих; (3) $\Delta_L^{c=p} = \Delta_L^{c\neq p} = 0$.

Тогда для кода K5: $L_{\max}^{c=p} = L_{\max}^{c\neq p} = 2^k - 1$, с учётом $k = 4$ получим $L_{\max}^{c=p} = L_{\max}^{c\neq p} = 15$.

$C =$ 0101 2222 2222 2222 2222 3453 3333 3367 89AB CDEF (40 байтов)

Множество самых редких байтов: $\{4, 5, 6, 7, 8, 9, \mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}\}$ — по (2) $p = 4$.

Схема данных кодирования RLE с односимвольным префиксом

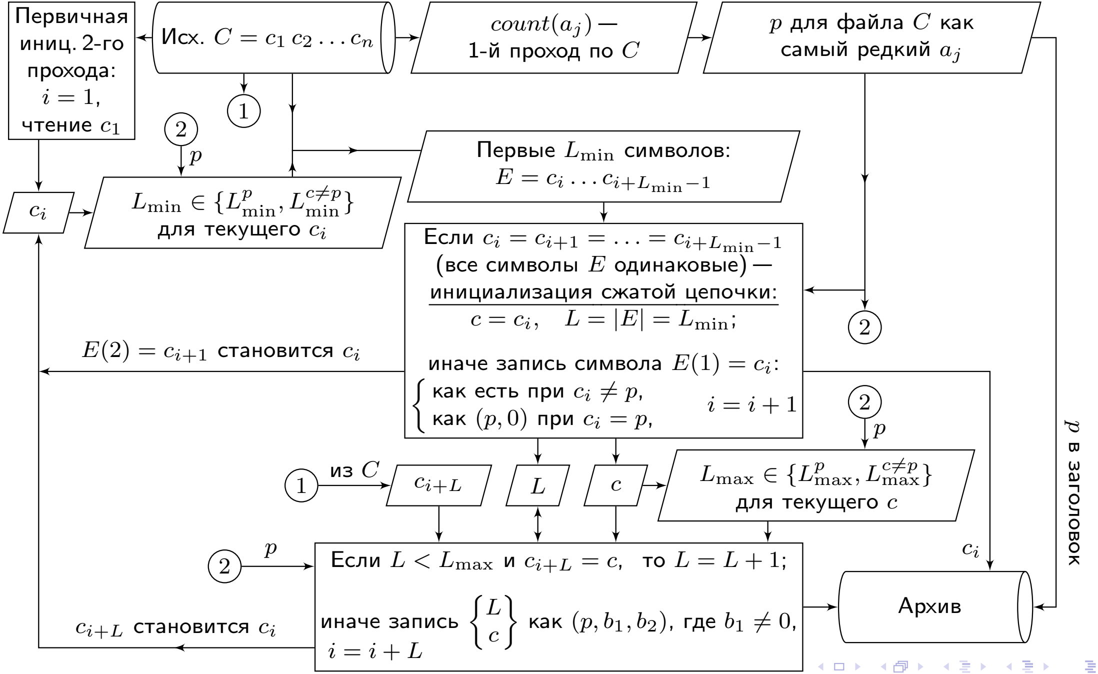


Схема данных декодирования RLE с односимвольным префиксом