

Неразделимые энтропийные коды. Арифметическое (интервальное) кодирование

Александра Игоревна Кононова

МИЭТ

17 октября 2024 г. — актуальную версию можно найти на
<https://gitlab.com/illinc/otik>

О терминах

Символ — элемент качественной информации $a \in A$ (множество A — алфавит).

Текст — последовательность $m \in A^+$ таких элементов.

На практике для всех алгоритмов, где алфавит может быть произвольным, **символ кодирования = байт** (так как в большинстве ЭВМ байт 8-битен — это 00...FF), **исходный текст = любой бинарный файл**, сжатый текст — тоже бинарный файл:

- использование в программе для ЭВМ символов меньших, чем байт — неудобно;
- использование символов фиксированной разрядности больших, чем байт \Rightarrow слишком большой алфавит \Rightarrow объёмные структуры данных для восстановления.
- использование в качестве символа кодирования печатного символа ASCII или koi8r/cp1251/dos/iso/maccyrillic не позволяет рассматривать в качестве исходного текста произвольный файл и приводит к труднодиагностируемым ошибкам;
- использование печатного символа UTF-8 (144 697 символов Unicode в 2023 г.) — то же самое + проигрыш в объёме.

В книгах для наглядности используются обозначения A, B, \dots и т. п. (маленький алфавит + визуальное отличие символа от индекса или частоты), но в программе это всё равно байты!



Арифметическое кодирование без сжатия — все ПИ одной длины

Строка D -ичного алфавита $C = c_1 c_2 c_3 \dots$ — точка $z = \overline{0, c_1 c_2 c_3 \dots_D}$.

Двоичные цифры (биты) $b_1 b_2 b_3 \dots$ дробной части двоичной записи того же числа $z = \overline{0, b_1 b_2 b_3 \dots_2} = \overline{0, c_1 c_2 c_3 \dots_D}$ — код строки C .

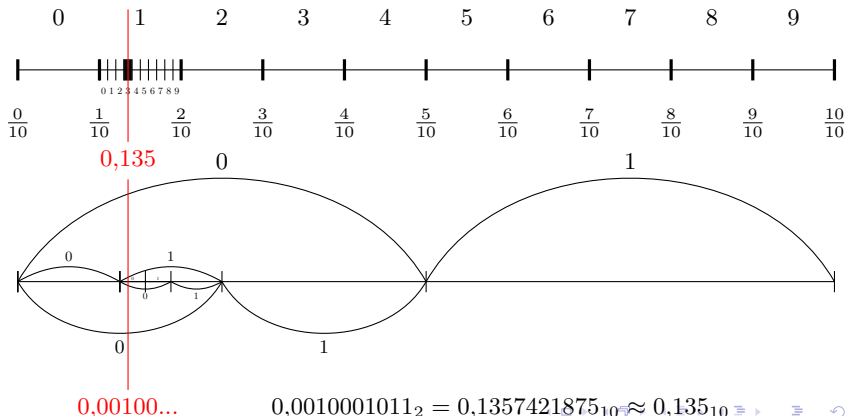
- Любое значение $z \in [0, 1)$ имеет **бесконечное** число цифр (часть их может быть 0) в любой СС;
даже для конечного числа D -ичных цифр $\overline{0, c_1 c_2 c_3 \dots c_{nD}}$ двоичное представление может быть бесконечно длинным → **округление**;
- округление к ближайшему требует для корректного округления вычислить на одну цифру больше, чем надо → **округление к 0**.

Геометрическая интерпретация кодирования без сжатия

$0,135_{10} = 0,001(00010100011110101110)_2$ — конечное число цифр окр-ся к 0 до $0,13\textcolor{red}{4}_{10}$

$$0,135_{10} \sim \begin{cases} \text{исходная длина (3 цифры)} \\ \text{любое значение } z \text{ из полуинтервала } [0,135; 0,136) = [0,135(0); 0,135(9)) \end{cases}$$

Цифра D -ичного алфавита $c_1 c_2 c_3 \dots$ — полуинтервал $\subseteq [0, 1)$



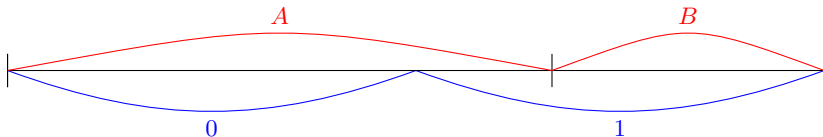
АС (концепт)
Интервальное кодирование (базовые положения)
Приведение частот
Интервальное кодирование (реализация)

О терминах
Идея арифметического кодирования (АС)
Арифметическое кодирование без сжатия — все ПИ одной длины
Геометрическая интерпретация кодирования без сжатия

Арифметическое сжатие (концепт) — длина ПИ равна вероятности строки

- 1 Бесконечная строка $c_1c_2...c_nc_{n+1}...$ символов T -ичного алфавита A соответствует вещественному числу (точке) бесконечной точности в диапазоне $[0, 1)$.
Соответствие аналогично позиционной системе счисления по основанию T , но диапазон разбивается на T **неравных частей пропорционально частотам символов**.
- 2 Конечная строка из n символов $c_1c_2...c_n$ — полуинтервал (ПИ) $[l_n, t_n) \subset ... \subset [l_1, t_1) \subset [0, 1)$; каждая его точка соответствует бесконечной строке, начинающейся с $c_1c_2...c_n \implies$ выбирается $z \in [l_n, t_n)$ с самым коротким двоичным представлением.
- 3 Полученная точка z представляется в двоичной системе счисления: $z = 0.b_1b_2...b_m$.
Исходное количество символов $n +$ битовая строка $b_1b_2...b_m$ — код $c_1c_2...c_n$.

Сжимаем текст $AABABA$. Вероятность символа $A = \frac{2}{3}$, $B = \frac{1}{3} \implies$ диапазон $[0; 1)$ делится $2 : 1$.



- 1 Все строки — ПИ $[0, 1)$: начинающиеся с A — ПИ $[0, \frac{2}{3})$, начинающиеся с B — ПИ $[\frac{2}{3}, 1)$.
- 2 Строки $A...$ — ПИ $[l_1, t_1) = [0, \frac{2}{3}) \subseteq [0, 1)$: вторая A $[0, \frac{4}{9})$, вторая B $[\frac{4}{9}, \frac{2}{3})$ и т. д.

⏪ ⏩ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏺ ⏻ ⏼ ⏽ ⏾ ⏿ 🔍 ↺

АС (концепт)

Интервальное кодирование (базовые положения)

Приведение частот

Интервальное кодирование (реализация)

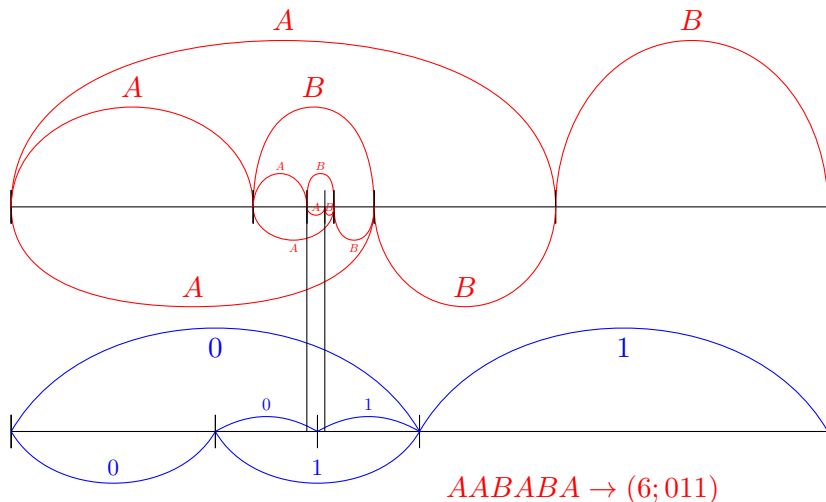
Геометрическая интерпретация арифметического сжатия

Длина арифметического кода (концепт; целочисленный длиннее)

Сравнение с кодами Хаффмана (один символ)

Точность, необходимая для реализации концепта арифметического сжатия

Геометрическая интерпретация арифметического сжатия



БАНАН $\rightarrow (5; 11010111) \leftrightarrow 011$ [_ac_samples/banan_realset_integerset.pdf](#)

АС (концепт)
 Интервальное кодирование (базовые положения)
 Приведение частот
 Интервальное кодирование (реализация)

Геометрическая интерпретация арифметического сжатия
 Длина арифметического кода (концепт; целочисленный длиннее)
 Сравнение с кодами Хаффмана (один символ)
 Точность, необходимая для реализации концепта арифметического сжатия

Длина арифметического кода (концепт; целочисленный длиннее)

Вероятности: в данной главе везде моделью является источник без памяти, $p(\xi_j) = \text{const}$

Для строки $C = c_1 c_2 \dots c_n$ длина итогового ПИ $t_n - l_n = \Delta_n = p(c_1) \cdot p(c_2) \cdot \dots \cdot p(c_n) = p(C)$

Если $\Delta_n \geq \frac{1}{2^k}$, то $\exists z \in [l_n, t_n)$ не более k двоичных разрядов после запятой (м.б. короче):

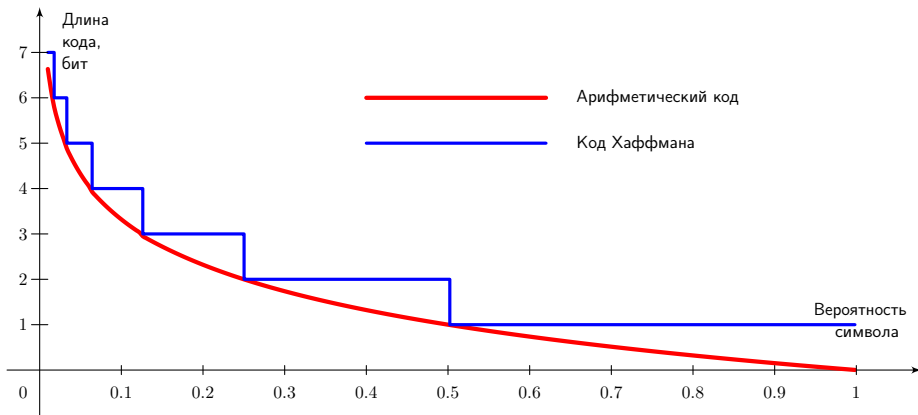
$$|code(C)| \leq k = \left\lceil \log_2 \left(\frac{1}{\Delta_n} \right) \right\rceil = \left\lceil \log_2 \left(\frac{1}{p(C)} \right) \right\rceil = \lceil I(C) \rceil \text{ бит}$$

Для $AABABA$: $\Delta_6 = \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{1}{3} \cdot \frac{2}{3} = \frac{2^4}{3^6} \approx 0,02 \gtrsim 2^{-6} \implies \text{длина кода} \leq 6 \text{ бит}$

20 A и 10 B (30 символов): $\Delta_{30} = \frac{2^{30}}{3^{30}} \approx 2^{-27,5} \implies \leq 28 \text{ бит}$

Начиная с какого m сообщение из $2m$ A и m B кодируется гарантированно короче $3m$ бит?

Сравнение с кодами Хаффмана (один символ)



Степень сжатия на типичных данных на 1-10% лучше кода Хаффмана.

Не увеличивает размера исходных данных в худшем случае.

АС (концепт)

Интервальное кодирование (базовые положения)

Приведение частот

Интервальное кодирование (реализация)

Геометрическая интерпретация арифметического сжатия

Длина арифметического кода (концепт; целочисленный длиннее)

Сравнение с кодами Хаффмана (один символ)

Точность, необходимая для реализации концепта арифметического сжатия

Точность, необходимая для реализации концепта арифметического сжатия

Для концепта арифметического сжатия необходимо представлять все границы ПИ $[0, 1) \supseteq [l_1, t_1) \supseteq \dots \supseteq [l_n, t_n) \ni z$ (дроби со знаменателем D^n) **абсолютно точно**.

- 1 При использовании *float/double* выполняется непредсказуемое округление + на ПИ $[0, 1)$ используется 30/62 бит в целом и 23/52 бита мантиссы.
- 2 В двоичной СС конечно представимы только дроби со знаменателем 2^n , в десятичной — с $2^n 5^k \implies$ «неограниченные» десятичные числа не годятся.
- 3 При округлении при кодировании и декодировании погрешность накапливается.

Целочисленная реализация арифметического сжатия — **интервальное кодирование**:

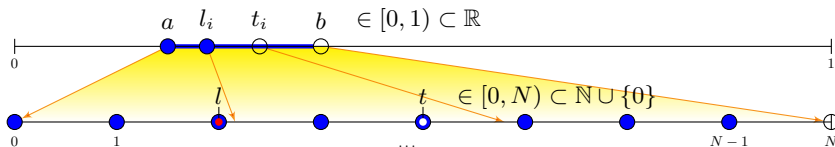
- 1 погрешность при декодировании должна накапливаться точно таким же образом, как при кодировании \implies исходную строку можно восстановить;
- 2 чем она больше, тем длиннее код \iff **код не совпадает с концептом**, хотя начало будет похожим;

в интервальном кодировании некоторый **фрагмент** $[a, b)$ вещественного ПИ $[0, 1)$ изображается в виде целочисленного ПИ $[0, N)$; операции — целочисленные.

Цифровой микроскоп Ньютона

Некоторый **фрагмент** $[a, b]$ вещественного полуинтервала (ПИ) $[0, 1)$ изображается в виде целочисленного ПИ $[0, N)$

$$[a, b] \rightarrow [0, N) \left([l_i, t_i] \subseteq [a, b] \subseteq [0, 1) \right) \quad N \gg 4D^2, \frac{N}{4} \in \mathbb{N}$$



Рассматриваем вместо настоящих l_i и t_i из $[0, 1)$ их изображения на $[0, N)$ (целые) — одной паре l_i и t_i могут соответствовать **разные** изображения l и t при рассмотрении разных $[a, b]$.

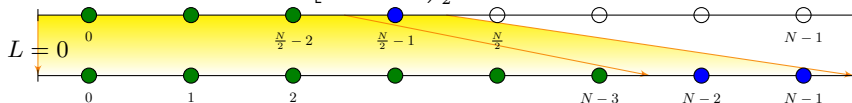
Вначале $[a, b] = [0, 1)$, потом каждый раз при возможности уменьшается вдвое, чтобы изображающий $[l_i, t_i]$ целочисленный ПИ $[l, t]$ не стал короче $\frac{N}{4}$.

Приближаем («микроскоп Ньютона») одну из половин $[a, b] \implies$ новые изображения l и t масштабируются из старых изображений ($l_i, t_i \in \mathbb{R}$ недоступны).

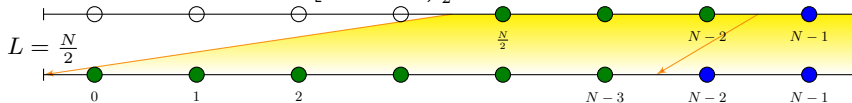
Масштабирование (нормализация) и биты двоичного представления

приближаем часть ПИ $[L, L + \frac{N}{2})$ — изображаемый ПИ $[a, b)$ уменьшается вдвое, длины на изображении увеличиваются вдвое (масштабирование $x \rightarrow 2(x - L)$).

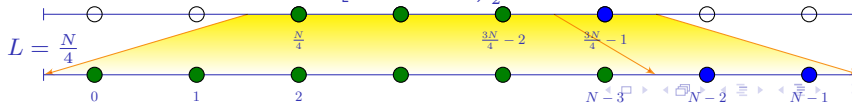
$[a, b) = [0, \frac{1}{2}) = [0, 0; 0, 1)_2 = [0, 0; 0, 0(1))_2$: первый бит $z \in [l_i, t_i) \subseteq [a, b)$ всегда 0



$[a, b) = [\frac{1}{2}, 1) = [0, 1; 1, 0)_2 = [0, 1; 0, 1(1))_2$: первый бит $z \in [a, b)$ всегда 1



$[a, b) = [\frac{1}{4}, \frac{3}{4}) = [0, 01; 0, 11)_2 = [0, 01; 0, 10(1))_2$: второй бит $z \in [a, b)$ инверсен первому



АС (концепт)
Интервальное кодирование (базовые положения)
Приведение частот
Интервальное кодирование (реализация)

Цифровой микроскоп Ньютона
Масштабирование (нормализация) и биты двоичного представления
Частота масштабирования
Величины и требования

Частота масштабирования

- Шаг масштабирования = бит выходного потока:
 - 1 простейший случай (размер выходного файла наименьший; медленно; рассматривается ниже) — масштабирование выполняется при любой возможности, **выходной поток битовый**;
 - 2 байтовый выходной поток — масштабирование откладывается (увеличивается выходной файл; быстрее; необходимо $N \gg 256D^2$).
- Реализовать кодирование более-менее длинного сообщения без масштабирования невозможно.
- Схема масштабирований при декодировании должна быть **точно такой же**, как и при кодировании — иначе декодированное не совпадёт с исходным. Также должны совпадать и ν_j ($\implies D$ и ω_j), и N , и разрядность вычислений.

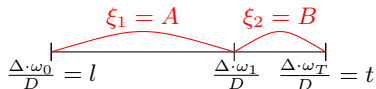
Величины и требования

Алфавит из T символов: $\xi_1, \xi_2, \dots, \xi_T$, частоты $\nu_1, \nu_2, \dots, \nu_T \in \mathbb{N}$, сортируются по убыванию $\nu_1 \geq \nu_2 \geq \dots \geq \nu_T$.

Деление пропорционально ν_j :

$$\begin{cases} \omega_0 &= 0, \\ \dots & \\ \omega_j &= \omega_{j-1} + \nu_j, \\ \dots & \\ \omega_T &= \omega_{T-1} + \nu_T. \end{cases}$$

$D = \omega_T = \sum_j \nu_j$ — делитель, не обязательно 2^k .



Изменение отрезка при чтении символа $c_i = \xi_j : \Delta = t - l$, $\begin{cases} l \rightarrow l + \frac{\Delta \cdot \omega_{j-1}}{D} \\ t \rightarrow l + \frac{\Delta \cdot \omega_j}{D} \end{cases}$

новый $[l, t)$ снова делится: в вещественном концепте — до исчерпания $c_1 c_2 \dots c_n$; в целочисленной — пока не станет слишком маленьким (затем масштабирование).

Выбор N для кодирования: $N \dot{\vdash} 4$ (для масштабирования);

расчёт без переполнения \Rightarrow ① $N \cdot D \leq \max(type)$; $\Delta > \frac{N}{4}$ дважды делится на символы \Rightarrow ② $N \gg 4D^2$. $\left. \begin{array}{l} \Rightarrow \text{огр. } D \end{array} \right\} \Rightarrow$

Выбор N для декодирования: деление на 2 всегда точно \Rightarrow ③ $N = 2^\alpha$, $\alpha \gg 1$

Приведение частот

Для вычислений требуется T ненулевых частот в порядке убывания ($\nu_1 \geq \nu_2 \geq \dots \geq \nu_T$, $\sum_{j=1}^T \nu_j = D$);

но хранить, как и для Хаффмана, нужно частоты всех 2^k возможных символов (k -битных байтов):

$\vec{\nu} = (\nu(0), \nu(1), \dots, \nu(2^k - 1))$, из них T ненулевых и $2^k - T$ нулевых; $\sum_{j=0}^{2^k-1} \nu(j) = D$.

Из файла длины n символов $\overrightarrow{count} = (count(0), count(1), \dots, count(2^k - 1))$; $\sum_{j=0}^{2^k-1} count(j) = n$.

- 1 Для вычислений требуется $D^2 \ll \frac{N}{4} \implies$ частоты **обязательно** нормируются:

$$\begin{cases} \nu(0) + \nu(1) + \dots + \nu(2^k - 1) \approx D_{\text{желаемое}}, \\ \nu(0) : \nu(1) : \dots : \nu(2^k - 1) \approx count(0) : count(1) : \dots : count(2^k - 1), \end{cases} \quad \nu(j) \in \mathbb{N} \cup \{0\}.$$
- 2 Арифметическое/интервальное кодирование принципиально не может записать символ с $\nu_j = 0 \implies$ ненулевые частоты не должны переходить в нулевые:

$$\nu(j) = 0 \Leftrightarrow count(j) = 0.$$

Выбор $D_{\text{жел}}$

Пусть $\nu_1, \nu_2, \dots, \nu_T \in \mathbb{N}$ (то есть ненулевые!), такие, что $\sum_{j=1}^T \nu_j = D$:

- чем меньше D , тем точнее вычисления \implies меньше размер сжатых данных;
 - но при слишком малых D менее точны $\nu(i) : \nu(j) \approx \text{count}(i) : \text{count}(j) \implies$ модель не соответствует исходному файлу \implies больше размер сжатых данных.
- 1 Если $D < T$, такие $\nu_1, \nu_2, \dots, \nu_T \in \mathbb{N}$ вообще невозможны.
 - 2 Если $D = T$, то $(\nu_1, \nu_2, \dots, \nu_T) = (1, 1, \dots, 1) \implies$ код фиксированной ширины.
 - 3 Если $D = 2T$, при равных частотах $(\nu_1, \nu_2, \dots, \nu_T) = (2, 2, \dots, 2)$; большие отклонения от $\text{count}(i) : \text{count}(j) = 1 : 1$ возможно отразить, малые нет.
И т. д.: при равных частотах и $D = 4T$ $\vec{\nu} = (4, 4, \dots, 4)$, при $D = 8T$ $(8, 8, \dots, 8)$...

Для k -битного байта $T_{\text{max}} = 2^k$, в общем случае $D_{\text{жел}} \geq 8T_{\text{max}} = 8 \cdot 2^k = 2^{k+3}$.

Если частоты приведены к $\sum \nu_j = D_{\text{жел}} = 2^\kappa$, для записи ν_j достаточно κ бит ($\kappa > k$, более байта):

- либо $\exists i: \nu_i = 2^\kappa$ и $\forall j \neq i: \nu_j = 0 \implies$ записывается $\nu_i = 1$ и $\nu_j = 0$ при $j \neq i$,
 $|\text{code}(c_1 c_2 \dots c_n)| = 0$ бит (вырожденный случай);
- либо $\forall j: \nu_j < 2^\kappa \implies$ все ν_j записываются κ битами.



Приведение $\sum count(j) = n$ к $\sum \nu(j) = D_{\text{жел}}$

А Соотношения всех частот немного искажаются, $D_{\text{жел}} \geq T$:

$$x(j) = \begin{cases} 0, & count(j) = 0, (2^k - T) \text{ штук;} \\ \frac{(count(j)-1) \cdot (D_{\text{жел}} - T)}{n - T} + 1, & count(j) > 0, T \text{ штук;} \end{cases} \quad \nu(j) = \text{round}_{\Sigma} (x(j)).$$

Б Соотношения меньших искажаются сильнее, $D_{\text{жел}} \geq 4T$:

$$x(j) = \begin{cases} 0, & count(j) = 0; \\ 1, & 1 \leq count(j) < \frac{3n}{2D_{\text{жел}}}, T_1 \text{ штук, } \sum count = n_1; \\ \frac{count(j) \cdot (D_{\text{жел}} - T_1)}{n - n_1}, & count(j) \geq \frac{3n}{2D_{\text{жел}}}; \end{cases} \quad \nu(j) = \text{round}_{\Sigma} (x(j)).$$

Если округлять к ближайшему, $\nu(j) = \text{round} (x(j))$, то $\sum \nu(j) \approx D_{\text{жел}}$ — не обязательно точно.

round_{Σ} **с сохранением суммы:** $\nu(j) = \lfloor x(j) \rfloor$, $\Delta_{\nu} = D_{\text{жел}} - \sum \nu(j)$ — целое и $\Delta_{\nu} = \sum \{x(j)\} \leq T$; Δ_{ν} раз ищем j с наибольшим $\{x(j)\}$ и заменяем $\nu(j) \rightarrow \nu(j) + 1$, $\{x(j)\} \rightarrow 0$; в итоге $\sum \nu(j) = D_{\text{жел}}$.

Целочисленное, $x(j)$ вида $\frac{u(j) \cdot D}{U}$: $\nu(j) = \left\lfloor \frac{u(j) \cdot D}{U} \right\rfloor$; вместо $0 \leq \{x(j)\} < 1$ целое $0 \leq (u(j) \cdot D) \% U < U$.

Приведение (А) и (В)

Пусть $D_{\text{жел}} = 2^6 = 64$ и $\overrightarrow{\text{count}} = (0, 3, 1, 46, 47, 1, 0, 2)$, $n = \sum \text{count}(j) = 100$; ненулевых частот $T = 6$, $y(j) = \lfloor x(j) \rfloor$, $r(j) \sim \{x(j)\}$, $\nu(j)$ округлены с сохранением суммы и $\tilde{\nu}(j)$ к ближайшему.

Приведение (А): $D_{\text{жел}} - T = 58$, $n - T = 94$:

$$\begin{cases} \text{count}(j) = 0: & y(j) = 0, & r(j) = 0; \\ \text{count}(j) > 0: & y(j) = \left\lfloor \frac{(\text{count}(j)-1) \cdot (D_{\text{жел}}-T)}{n-T} \right\rfloor + 1, & r(j) = ((\text{count}(j) - 1) \cdot (D_{\text{жел}} - T)) \% (n - T). \end{cases}$$

Приведение (В): $\frac{3n}{2D_{\text{жел}}} \approx 2,3 \Rightarrow$ значений $1 \leq \text{count}(j) < \frac{3n}{2D_{\text{жел}}}$ в массиве $T_1 = 3$, их сумма $n_1 = 4$:

$$\begin{cases} \text{count}(j) = 0: & y(j) = 0, & r(j) = 0; \\ 1 \leq \text{count}(j) < \frac{3n}{2D_{\text{жел}}}: & y(j) = 1, & r(j) = 0; \\ \frac{3n}{2D_{\text{жел}}} \leq \text{count}(j): & y(j) = \left\lfloor \frac{\text{count}(j) \cdot (D_{\text{жел}} - T_1)}{n - n_1} \right\rfloor, & r(j) = (\text{count}(j) \cdot (D_{\text{жел}} - T_1)) \% (n - n_1). \end{cases}$$

(А)

j	0	1	2	3	4	5	6	7	\sum / Δ_ν
$r_A(j)$	0	22	0	72	36	0	0	58	
$y_A(j)$	0	2	1	28	29	1	0	1	62 (2)
$\nu_A(j)$	0	2	1	29	29	1	0	2	64
$\tilde{\nu}_A(j)$	0	2	1	29	29	1	0	2	64

(В)

j	0	1	2	3	4	5	6	7	\sum / Δ_ν
$r_B(j)$	0	87	0	22	83	0	0	0	
$y_B(j)$	0	1	1	29	29	1	0	1	62 (2)
$\nu_B(j)$	0	2	1	29	30	1	0	1	64
$\tilde{\nu}_B(j)$	0	2	1	29	30	1	0	1	64

Приведение (A): $Max \rightarrow \sum$

$D_{\text{жел}} = 64$ и $\overrightarrow{count} = (0, 45, 30, 46, 47, 30, 0, 40)$, $n = \sum count(j) = 238_{10} = 356_8$, ненулевых $T = 6$.

① Непосредственное (A) $\overrightarrow{count} \rightarrow \vec{\nu}$:

j	0	1	2	3	4	5	6	7	\sum
$r(j)$	0	0	58	58	116	58	0	174	
$y(j)$	0	12	8	12	12	8	0	10	$62 (\Delta_y = 2)$
$\nu(j)$	0	12	8	12	13	8	0	11	64
$\tilde{\nu}(j)$	0	12	8	12	13	8	0	11	64

Разрядность $\nu(j)$ больше байта \Rightarrow массив частот объёмнее, чем для Хаффмана.

② Вначале для записи в файл (A) \overrightarrow{count} к однобайтовым \vec{u} : $\vec{u} = (0, 7, 5, 7, 7, 5, 0, 6)$, $\sum u(j) = 37$;

j	0	1	2	3	4	5	6	7	\sum
$r(j)$	0	7	15	7	7	15	0	11	
затем (A) $\vec{u} \rightarrow \vec{\nu}$: $y(j)$	0	12	8	12	12	8	0	10	$62 (\Delta_y = 2)$
$\nu(j)$	0	12	9	12	12	9	0	10	64
$\tilde{\nu}(j)$	0	12	8	12	12	8	0	10	62

В файл: $\underbrace{0356}_n$, $\underbrace{07577506}_{\vec{u}}$ и код по $\vec{\nu} = (0, 12, 9, 12, 12, 9, 0, 10)$ (или по $\tilde{\vec{\nu}}$ — как реализовано).

Две перенормировки \Rightarrow модель хуже подходит файлу \Rightarrow больше размер; для 8-битного байта менее выражено, но на некоторых файлах код Хаффмана эффективнее целочисленной реализации АС.

Задачи для семинаров и вопросы

- 1 Найдите возможные пары $N = 2^\alpha$ и $D_{\text{жел}} = 2^\kappa$ для алфавита из октетов (8-битных байтов IBM PC), $T_{\text{max}} = 2^8 = 256$ и а) 32-битной арифметики (все пары), б) 64-битной арифметики (граничные).
- 2 Оцените максимально возможное $\frac{\nu_{\text{max}}}{\nu_{\text{min}}}$ для октетов и $D_{\text{жел}} \in \{2T, 4T, 8T, 16T\}$.
- 3 Рассчитайте $\vec{\nu}$ при равных частотах, а также максимально возможное $\frac{\nu_{\text{max}}}{\nu_{\text{min}}}$ для октетов и $D_{\text{жел}} = 2^{16}$.

- 4 Пусть байт=октет, а $D_{\text{жел}} = 2^{12} = 16T_{\text{макс}}$. В общем случае для $\nu(j)$ требуется полтора октета, но во многих случаях достаточно одного: примерно равные частоты всех байтов — $\nu \approx \frac{2^{12}}{256} = 16$; печатных ASCII-символов $\frac{2^{12}}{95} \approx 40$ для ненулевых; строчной латиницы $\frac{2^{12}}{25} \approx 160$ для ненулевых.

Как без дополнительных служебных полей понять при чтении массива из 256 октетов с младшими частями $\nu(j)$ — достаточно их или позже должны следовать $\frac{4}{8} \cdot 256 = 128$ октетов с 256 4-битными старшими частями $\nu(j)$?

Реализуемо ли предложенное для $D_{\text{жел}} = 2^{11}$ или 2^{10} ?

- 5 Как выгоднее сохранять частоты октетов — 8-разрядными ($\max \nu_j = 2^8 - 1$) с перенормировкой перед расчётом или готовыми κ -разрядными ($\sum \nu_j = 2^\kappa$, $\kappa > 8$)? Обоснуйте ответ.

Сжатие \diamond БАНАН \hookrightarrow 011_ac_samples/banan_realset_integerset.pdf

Входной поток: символы $C = c_1 c_2 \dots c_n$, выходной — биты $B = b_1 b_2 b_3 \dots b_m$

- 0 $l = 0, t = N$, бит зарезервировано $\beta = 0$, позиция символа $i = 0$
- 1 чтение $(++i)$ символа $C \rightarrow c_i = \xi_j$: $\Delta = t - l$, $\begin{cases} l \rightarrow l + \frac{\Delta \cdot \omega_{j-1}}{D} \\ t \rightarrow l + \frac{\Delta \cdot \omega_j}{D} \end{cases}$, авост при $t = l$.
- 2 масштабирование l, t : $\begin{cases} l \rightarrow 2(l - L) \\ t \rightarrow 2(t - L) \end{cases}$ и запись бита b , пока возможно (м.б. неск.):
 $\left[\frac{b \cdot N}{2}, \frac{(b+1) \cdot N}{2} \right), b \in \{0, 1\} \rightarrow [0, N)$ и запись $b \rightarrow B$ ($\underbrace{b \bar{b} b \dots \bar{b}}_{\beta} \rightarrow B, \beta \rightarrow 0$)
 $\left[\frac{N}{4}, \frac{3N}{4} \right) \rightarrow [0, N)$ и $++\beta$
- 3 если достигнуто $i = n$ — запись $1 \rightarrow B$ ($\underbrace{1 0 0 \dots 0}_{\beta} \rightarrow B$) и завершение;
 иначе переход к шагу 1

-
- 1 Полученное $z \in [0, 1)$ соответствует **бесконечно длинной строке**
 $c_1 c_2 \dots c_n c_{n+1} c_{n+2} \dots \rightarrow$ необходимо сохранить исходную длину n .
 - 2 Поточный вариант — ν_j и ω_j пересчитываются.

Распаковка

Символов — n , входной поток — биты $B = b_1 b_2 b_3 \dots b_m 000 \dots$,
выходной поток — символы $C = c_1 c_2 \dots c_n$.

① $l = \lambda = 0, t = \tau = N$, № бита $k = 0$, № символа $i = 0$

① чтение $(++k)$ бита $B \rightarrow b_k$: $\delta = \tau - \lambda$, $\begin{cases} \lambda \rightarrow \lambda + \frac{\delta \cdot b_k}{2}, \\ \tau \rightarrow \lambda + \frac{\delta \cdot (b_k + 1)}{2}. \end{cases}$

② получение и запись символа, если возможно: $\Delta = t - l$, $j \in \{1, \dots, T\}$

$$\exists j: [\lambda, \tau) \subseteq \left[l + \frac{\Delta \cdot \omega_{j-1}}{D}, l + \frac{\Delta \cdot \omega_j}{D} \right) \implies ++i, \xi_j \rightarrow C, \begin{cases} l \rightarrow l + \frac{\Delta \cdot \omega_{j-1}}{D} \\ t \rightarrow l + \frac{\Delta \cdot \omega_j}{D} \end{cases}$$

③ масштабирование l, λ, τ, t , пока возможно (м. б. несколько раз):

$[l, t) \subseteq [L, L + \frac{N}{2}) \implies [L, L + \frac{N}{2}) \rightarrow [0, N)$ (не влияет на выходной поток);
когда уже невозможно — переход к шагу ②

④ если достигнуто $i = n$ — завершение, иначе переход к шагу ①

$N = 2^\alpha \implies$ чтение бита точно \implies на шаге ① читаем сразу несколько бит (при $k = 0$ читаем α бит), чтобы получить ПИ вида $[\lambda, \lambda + 1)$.

Полуинтервалы и отрезки

Выше в целочисленной реализации рабочий диапазон рассматривался как **полуинтервал** $[l, t)$, $l \leq z < t$, где t — **невключаемая** верхняя граница.

Тот же самый диапазон можно представить как **отрезок** $[l, h]$, $l \leq z \leq h$, где $h = t - 1$ — **включаемая** верхняя граница.

Реализовать кодирование и декодирование можно как для полуинтервалов $[l, t)/[\lambda, \tau)$, так и для отрезков $[l, h]/[\lambda, \chi]$, но **все** соотношения будут различаться (см. следующий лист)!

Основные соотношения для t и h

$[l, t) = [l, h]$	\iff	$h = t - 1$	\iff	$t = h + 1$
Полуинтервал $[l, t)$, $l \leq z < t$		Отрезок $[l, h]$, $l \leq z \leq h$		
Длина				
$\Delta = t - l$	$\delta = \tau - \lambda$	$\Delta = h - l + 1$	$\delta = \chi - \lambda + 1$	
Чтение символа ξ_j или бита b				
$\begin{cases} l \rightarrow l + \frac{\Delta \cdot \omega_{j-1}}{D}, \\ t \rightarrow l + \frac{\Delta \cdot \omega_j}{D}, \end{cases}$ авост при $t = l$		$\begin{cases} l \rightarrow l + \frac{\Delta \cdot \omega_{j-1}}{D}, \\ h \rightarrow l + \frac{\Delta \cdot \omega_j}{D} - 1, \end{cases}$ авост при $h < l$		
$\begin{cases} \lambda \rightarrow \lambda + \frac{\delta \cdot b}{2} \\ \tau \rightarrow \lambda + \frac{\delta \cdot (b+1)}{2} \end{cases}$		$\begin{cases} \lambda \rightarrow \lambda + \frac{\delta \cdot b}{2} \\ \chi \rightarrow \lambda + \frac{\delta \cdot (b+1)}{2} - 1 \end{cases}$		
Масштабирование $[L, L + \frac{N}{2}) \rightarrow [0, N)$, $L \in \{0, \frac{N}{4}, \frac{N}{2}\}$				
$[l, t) \subseteq [L, L + \frac{N}{2}) \iff \begin{cases} L \leq l \\ t \leq L + \frac{N}{2} \end{cases}$		$[l, h] \subseteq [L, L + \frac{N}{2}) \iff \begin{cases} L \leq l \\ h < L + \frac{N}{2} \end{cases}$		
$\begin{cases} l \rightarrow 2(l - L) \\ t \rightarrow 2(t - L) \end{cases}$		$\begin{cases} l \rightarrow 2(l - L) \\ h \rightarrow 2(h - L) + 1 \end{cases}$		
		$2t - 1 = 2(h + 1) - 1 = 2h + 1$		

Д. Ватолин, целочисленный цикл (отрезки)

```
1 l[0] = 0; h[0] = 65535; i = 0; delitel = b[c_last];
2 First_qtr = (h[0] + 1)/4; Half = First_qtr*2; Third_qtr = First_qtr*3;
3 bits_to_follow = 0; // масштабирований [First_qtr; Third_qtr)
4 while (not DataFile.EOF()) {
5     c = DataFile.ReadSymbol(); i++; // Кодированный символ
6     j = IndexForSymbol(c);          // и его номер в алфавите
7     l[i] = l[i-1] + b[j-1]*(h[i-1] - l[i-1] + 1)/delitel;
8     h[i] = l[i-1] + b[j]*(h[i-1] - l[i-1] + 1)/delitel - 1;
9     for(;;) {                        // Варианты масштабирования
10         if (h[i] < Half)              // [l; h] лежит в [0; Half)
11             bits_plus_follow(0);
12         else if (l[i] >= Half) {      // [l; h] лежит в [Half, max)
13             bits_plus_follow(1);
14             l[i] -= Half; h[i] -= Half;
15         }
16         else if ((l[i] >= First_qtr) && (h[i] < Third_qtr)) {
17             bits_to_follow++;
18             l[i] -= First_qtr; h[i] -= First_qtr;
19         } else break;
20         l[i] += l[i]; h[i] += h[i] + 1; // масштабирование *2
21     }
22 }
```

АС (концепт)
Интервальное кодирование (базовые положения)
Приведение частот
Интервальное кодирование (реализация)

Сжатие \diamond БАНАН \leftrightarrow 011_ac_samples/banan_realset_integerset.pdf
Распаковка
Полуинтервалы и отрезки
Основные соотношения для t и h
Д. Ватолин, целочисленный цикл (отрезки)

Д. Ватолин, целочисленный цикл (отрезки), запись бита (bits_plus_follow)

```
1 void bits_plus_follow (int bit)
2 {
3     CompressedFile.WriteBit(bit);
4     for(; bits_to_follow > 0; bits_to_follow--)
5         CompressedFile.WriteBit(!bit);
6 }
```

$\text{bits_to_follow} = \beta$ — количество масштабирований из средней половины $\left[\frac{1}{4}; \frac{3}{4}\right) \rightarrow [0; 1)$ подряд

Дмитрий Ватолин, МГУ,
Media data compression.
Сжатие без потерь



АС (концепт)
Интервальное кодирование (базовые положения)
Приведение частот
Интервальное кодирование (реализация)

Сжатие \diamond БАНАН \leftrightarrow 011_ac_samples/banan_realset_integerset.pdf
Распаковка
Полуинтервалы и отрезки
Основные соотношения для t и h
Д. Ватолин, целочисленный цикл (отрезки)

Спасибо за внимание!

МИЭТ

www.miet.ru

Александра Игоревна Кононова

illinc@mail.ru

gitlab.com/illinc/raspisanie

Нормировка на точную сумму

При $D = 2^k$ вычисления точнее \implies либо код немного короче, либо $D^2 < \frac{N}{4}$ вместо \ll :
 нужно найти $\vec{\nu}$ с $\sum \nu(j) = D_{\text{жел}}$, где $D_{\text{жел}} = 2^k \geq 2^{k+2}$ определено на этапе разработки алгоритма;
 $\overrightarrow{\text{count}} = (\text{count}(0), \text{count}(1), \dots, \text{count}(2^k - 1))$, $\sum \text{count}(j) = n$ — из исходного файла.

❶ Если $n = D_{\text{жел}}$, приведение не требуется: $\nu(j) = \text{count}(j)$ для всех.

❷ Если $n < D_{\text{жел}}$: для $\text{count}(j) = 0$ записываем $\nu(j) = 0$;
 осталось T штук ненулевых частот $\text{count}(j) \geq 1$:

- рассчитываем нецелые $x(j) = \frac{\text{count}(j)}{n} \cdot D_{\text{жел}} > \text{count}(j) \geq 1$; $\sum x(j) = D_{\text{жел}}$;
- округляем $x(j)$ до целых с сохранением суммы: $\nu(j) = \text{round}_{\sum} (x(j))$.

Нулевые $\text{count}(j) = 0$ можно и по общему правилу: $x(j) = \frac{\text{count}(j)}{n} \cdot D_{\text{жел}} = 0$ — целые $\implies \nu(j) = 0$.

❸ Если $n > D_{\text{жел}}$, то может быть $x(j) = \frac{\text{count}(j)}{n} \cdot D_{\text{жел}} < 1 \implies$ дополнительное искажение.
 Как и для нормировки на максимум (для Хаффмана), есть два способа:

- искажение всех соотношений $\nu(i) : \nu(j) \approx \text{count}(i) : \text{count}(j)$ понемногу (A);
- более точное $\nu(i) : \nu(j) \approx \text{count}(i) : \text{count}(j)$ для больших частот, а для малых $\nu(i) : \nu(j) = 1 : 1$ (B).

Здесь и далее $z(j) = \text{round}_{\Sigma}(x(j))$ – округление **набора** значений с сохранением суммы:

- точно сохранив сумму: $\sum z(j) = D$;
- максимально сохранив значения: $z(j) \approx x(j)$.

- 1 $y(j) = \lfloor x(j) \rfloor$ (округляем все значения вниз);
 - 2 $\Delta_y = D_{\text{жел}} - \sum y(j)$ (рассчитываем, сколько не хватает до D); Δ_y целое; $\Delta_y = \sum \{x(j)\} \leq T$;
 - 3 упорядочиваем значения по убыванию дробной части исходных $\{x(j)\}$:
 - для первых Δ_y штук $z(j) = y(j) + 1$ ($x(j)$ с самыми большими дробными частями округляем вверх);
 - для остальных $z(j) = y(j)$ (остальные — с меньшими дробными частями — округляем вниз).
- (Альтернативный 3 без пересортировок: Δ_y раз ищем j с наибольшим $\{x(j)\}$ и заменяем $y(j) \rightarrow y(j) + 1$, $\{x(j)\} \rightarrow 0$; после завершения Δ_y замен полагаем $z(j) = y(j)$).

Полученные таким образом $z(j)$: ● целые; ● $z(j) \approx x(j)$; ● $\sum z(j) = \sum y(j) + \Delta_y = D$;

- если $\forall j: x(j) \geq a$ для какого-либо $a \in \mathbb{Z}$, то и $\forall j: z(j) \geq a$;
- если среди $x(j)$ есть R штук целых — все они сохраняют значение при округлении: $z(j) = x(j)$.

Целочисл., $x(j)$ вида $\frac{u(j) \cdot D}{U}$: сразу $y(j) = \left\lfloor \frac{u(j) \cdot D}{U} \right\rfloor$; вместо $0 \leq \{x(j)\} < 1$ целое $0 \leq (u(j) \cdot D) \% U < U$

(A) $\sum count(j) = n$ к $\sum \nu(j) = D_{\text{жел}} \geq T$, искажение всех частот

Приведение (A) — аналогично $\text{round} \left(\frac{count(i)-1}{\max(count)-1} \cdot (Max - 1) \right) + 1$ для нормировки на максимум:

❶ Для $count(j) = 0$ записываем $\nu(j) = 0 \implies$ осталось $T \leq 2^k$ ненулевых значений:

- $count(j) \geq 1, \sum count(j) = n$;
- необходимо привести к $\nu(j) \geq 1$ с суммой $\sum \nu(j) = D_{\text{жел}}$.

❷ Рассмотрим значения $u(j) = count(j) - 1$, тогда $u(j) \geq 0$, $\sum u(j) = \sum (count(j) - 1) = n - T$;
будем приводить к $\theta(j) = \nu(j) - 1$, $\theta(j) \geq 0$, $\sum \theta(j) = \sum (\nu(j) - 1) = D_{\text{жел}} - T$:

$$x(j) = \frac{u(j)}{n-T} \cdot (D_{\text{жел}} - T) \implies \theta(j) = \text{round}_{\Sigma} (x(j)) \implies \nu(j) = \theta(j) + 1.$$

Так как round_{Σ} проще реализовать на массиве «без дыр», а целочисленные значения сохраняются:

$$\text{❶ } x(j) = \begin{cases} 0, & count(j) = 0, (2^k - T) \text{ штук,} \\ \frac{(count(j)-1) \cdot (D_{\text{жел}} - T)}{n-T} + 1, & count(j) > 0, T \text{ штук;} \end{cases} \quad \text{❷ } \nu(j) = \text{round}_{\Sigma} (x(j)).$$

«Целочисленная дробная часть» $r(j) = \{x(j)\} \cdot (n - T)$, $0 \leq r(j) < (n - T)$:

$$\begin{cases} count(j) = 0: y(j) = 0, & r(j) = 0; \\ count(j) > 0: y(j) = \left\lfloor \frac{(count(j)-1) \cdot (D_{\text{жел}} - T)}{n-T} \right\rfloor + 1, & r(j) = ((count(j) - 1) \cdot (D_{\text{жел}} - T)) \% (n - T). \end{cases}$$

Применимо и для $n \leq D_{\text{жел}}$, при этом $1 \rightarrow 1$; результат отличается от $\text{round}_{\Sigma} \left(\frac{count(j)}{n} \cdot D_{\text{жел}} \right)$.



(В) $\sum count(j) = n$ к $\sum \nu(j) = D_{\text{жел}} \geq 4T$, все малые частоты в 1

Приведение (В) — аналогично $\text{round}\left(\frac{count(i)}{\max(count)} \cdot Max\right)$: для больших $\nu(j) \approx \frac{count(j)}{n} \cdot D_{\text{жел}}$,
при $\frac{1}{2} < \frac{count(j)}{n} \cdot D_{\text{жел}} < \frac{3}{2} \left(\frac{1}{2} \cdot \frac{n}{D_{\text{жел}}} < count(j) < \frac{3}{2} \cdot \frac{n}{D_{\text{жел}}} \right) \nu(j) \approx 1$; при меньшем принудительно 1.

1 Для $count(j) = 0$ записываем $\nu(j) = 0 \Rightarrow$ осталось $T \leq 2^k$ ненулевых $count(j)$, $\sum count(j) = n$.

2 Для $1 \leq count(j) < \frac{3n}{2D_{\text{жел}}}$ записываем $\nu(j) = 1$ (пусть их $T_1 \leq T$ штук и $\sum count(j) = n_1 \leq n$).

При этом $T_1 < T$ и $n_1 < n$ (если $T_1 = T$, то $n = \sum count(j) < \sum \frac{3n}{2D_{\text{жел}}} = T \cdot \frac{3n}{2D_{\text{жел}}}$, но $D_{\text{жел}} \geq 4T$).

3 Оставшиеся $T - T_1$ частот с $\sum count(j) = n - n_1$ приводим к $\sum \nu(j) = D_{\text{жел}} - T_1$.

$$x(j) = \begin{cases} 0, & count(j) = 0, \\ 1, & 1 \leq count(j) < \frac{3n}{2D_{\text{жел}}}, \\ \frac{count(j) \cdot (D_{\text{жел}} - T_1)}{n - n_1}, & count(j) \geq \frac{3n}{2D_{\text{жел}}}; \end{cases} \Rightarrow \nu(j) = \text{round}_{\Sigma} (x(j)).$$

При $D_{\text{жел}} \geq 4T$ в третьем случае $\frac{count(j) \cdot (D_{\text{жел}} - T_1)}{n - n_1} > \frac{\frac{3n}{2D_{\text{жел}}} \cdot (D_{\text{жел}} - T)}{n} = \frac{3}{2} \cdot \left(1 - \frac{T}{D_{\text{жел}}}\right) \geq \frac{9}{8} > 1$.

Применимо и для $n \leq D_{\text{жел}}$, при этом $1 \rightarrow 1$; отличается от $\text{round}_{\Sigma} \left(\frac{count(j)}{n} \cdot D_{\text{жел}} \right)$ и от (А).

Приведение (А) и (В): $n < D_{\text{жел}}$

Пусть $D_{\text{жел}} = 2^6 = 64$ и $\overrightarrow{\text{count}} = (0, 3, 1, 16, 17, 1, 0, 2)$, $n = \sum \text{count}(j) = 40$; ненулевых частот $T = 6$:

Приведение (А): $D_{\text{жел}} - T = 58$, $n - T = 34$:

$$\begin{cases} \text{count}(j) = 0: & y(j) = 0, & r(j) = 0; \\ \text{count}(j) > 0: & y(j) = \left\lfloor \frac{(\text{count}(j)-1) \cdot (D_{\text{жел}}-T)}{n-T} \right\rfloor + 1, & r(j) = ((\text{count}(j) - 1) \cdot (D_{\text{жел}} - T)) \% (n - T). \end{cases}$$

j	0	1	2	3	4	5	6	7	Σ
$r(j)$	0	14	0	20	10	0	0	24	
$y(j)$	0	4	1	26	28	1	0	2	62
$\nu(j)$	0	4	1	27	28	1	0	3	64

$\Delta_y = D_{\text{жел}} - \sum y = 2$

Приведение (В): $\frac{3n}{2D_{\text{жел}}} \approx 0,9 \Rightarrow$ значений $1 \leq \text{count}(j) < \frac{3n}{2D_{\text{жел}}}$ нет, $T_1 = n_1 = 0$

$$\begin{cases} \text{count}(j) = 0: & y(j) = 0, & r(j) = 0; \\ 1 \leq \text{count}(j) < \frac{3n}{2D_{\text{жел}}}: & y(j) = 1, & r(j) = 0; \\ \frac{3n}{2D_{\text{жел}}} \leq \text{count}(j): & y(j) = \left\lfloor \frac{\text{count}(j) \cdot (D_{\text{жел}} - T_1)}{n - n_1} \right\rfloor, & r(j) = (\text{count}(j) \cdot (D_{\text{жел}} - T_1)) \% (n - n_1). \end{cases}$$

j	0	1	2	3	4	5	6	7	Σ
$r(j)$	0	32	24	24	8	24	0	8	
$y(j)$	0	4	1	25	27	1	0	3	61
$\nu(j)$	0	5	2	26	27	1	0	3	64

$\Delta_y = D_{\text{жел}} - \sum y = 3$