

# Последняя лекция! Сжатие без учёта контекста. Арифметическое (интервальное) кодирование

Александра Игоревна Кононова

МИЭТ

15 декабря 2022 г. — актуальную версию можно найти на

<https://gitlab.com/illinc/otik>

Л/р по АК (необязательная) перенесена на конец семестра;  
соответствующие семинар и лекция — тоже

# Идея арифметического кодирования

- 1 сообщение  $C = c_1 c_2 \dots c_n$  соответствует вещественному числу (точке)  $z \in [0; 1)$ ;
- 2 точка  $z$  представляется в двоичной системе счисления.

Сколько бит требуется для полной записи  $z \in [0; 1)$ ?

Сжатие:

- 1  $z$  сохраняется **ровно** с той точностью, чтобы восстановить  $n$  символов  $C$  (цикл масштабирования);
- 2 при задании  $z$  учитываются частоты символов  $C$ .

# Арифметическое кодирование без сжатия

Строка  $D$ -ичного алфавита  $C = c_1c_2c_3\ldots$  — точка  $z = \overline{0, c_1c_2c_3\ldots_D}$ .

Двоичные цифры (биты)  $b_1b_2b_3\ldots$  дробной части двоичной записи того же числа  $z = \overline{0, b_1b_2b_3\ldots_2} = \overline{0, c_1c_2c_3\ldots_D}$  — 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  $\overline{0, c_1c_2c_3\ldots c_{n_D}}$  двоичное представление может быть бесконечно длинным → **округление**;
- округление к ближайшему требует для корректного округления вычислить на одну цифру больше, чем надо

# Перевод $10 \rightarrow 2$ и $2 \rightarrow 10$

$$\overline{0, c_1 c_2 c_3 \dots} \cdot D = c_1 + \overline{0, c_2 c_3 \dots}, \quad 0 \leq c_1 < D, \quad \overline{0, c_2 c_3 \dots} \in [0; 1)$$

$$0,135 \cdot 2 = 0,270 \quad 0,560 \cdot 2 = 1,120 \quad 0,360 \cdot 2 = 0,720$$

$$0,270 \cdot 2 = 0,540 \quad 0,120 \cdot 2 = 0,240 \quad 0,720 \cdot 2 = 1,440$$

$$0,540 \cdot 2 = 1,080 \quad 0,240 \cdot 2 = 0,480 \quad 0,440 \cdot 2 = 0,880$$

$$0,080 \cdot 2 = 0,160 \quad 0,480 \cdot 2 = 0,960 \quad 0,880 \cdot 2 = 1,760$$

$$0,160 \cdot 2 = 0,320 \quad 0,960 \cdot 2 = 1,920 \quad 0,760 \cdot 2 = 1,520$$

$$0,320 \cdot 2 = 0,640 \quad 0,920 \cdot 2 = 1,840 \quad 0,520 \cdot 2 = 1,040$$

$$0,640 \cdot 2 = 1,280 \quad 0,840 \cdot 2 = 1,680 \quad 0,040 \cdot 2 = 0,080$$

$$0,280 \cdot 2 = 0,560 \quad 0,680 \cdot 2 = 1,360 \quad 0,080 \cdot 2 = 0,160$$

Далее всё повторится:

$$0,135_{10} = 0,001(00010100011110101110)_2$$

$$0,0010001010_2 = \frac{1}{2^3} + \frac{1}{2^7} + \frac{1}{2^9} = \frac{1}{8} + \frac{1}{128} + \frac{1}{512} = 0,134765625_{10} \neq 0,135_{10}$$

# Перевод $10 \rightarrow 2$ (округление к нулю)

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

$$0,135 \cdot 2 = 0,270 \quad 0,136 \cdot 2 = 0,272$$

$$0,270 \cdot 2 = 0,540 \quad 0,272 \cdot 2 = 0,544$$

$$0,540 \cdot 2 = 1,080 \quad 0,544 \cdot 2 = 1,088$$

$$0,080 \cdot 2 = 0,160 \quad 0,088 \cdot 2 = 0,176$$

$$0,160 \cdot 2 = 0,320 \quad 0,176 \cdot 2 = 0,352$$

$$0,320 \cdot 2 = 0,640 \quad 0,352 \cdot 2 = 0,704$$

$$0,640 \cdot 2 = 1,280 \quad 0,704 \cdot 2 = 1,408$$

$$0,280 \cdot 2 = 0,560 \quad 0,408 \cdot 2 = 0,816$$

$$0,560 \cdot 2 = 1,120 \quad 0,816 \cdot 2 = 1,632$$

$$0,120 \cdot 2 = 0,240 \quad 0,632 \cdot 2 = 1,264$$

$$0,135_{10} \approx 0,0010001011_2 \text{ (до 3 десятичных знаков после запятой)}$$

## Перевод $2 \rightarrow 10$ (округление к нулю)

$$0,00100010110_2 = \frac{1}{2^3} + \frac{1}{2^7} + \frac{1}{2^9} + \frac{1}{2^{10}} = \frac{1}{8} + \frac{1}{128} + \frac{1}{512} + \frac{1}{1024} =$$

$$= 0,1357421875_{10} \approx 0,135_{10} \text{ (округление к нулю до 3 десятичных знаков)}$$

Для перевода чисел описанным способом (как  $10 \rightarrow 2$ , так и  $2 \rightarrow 10$ ), необходимо делать **точные** вычисления.

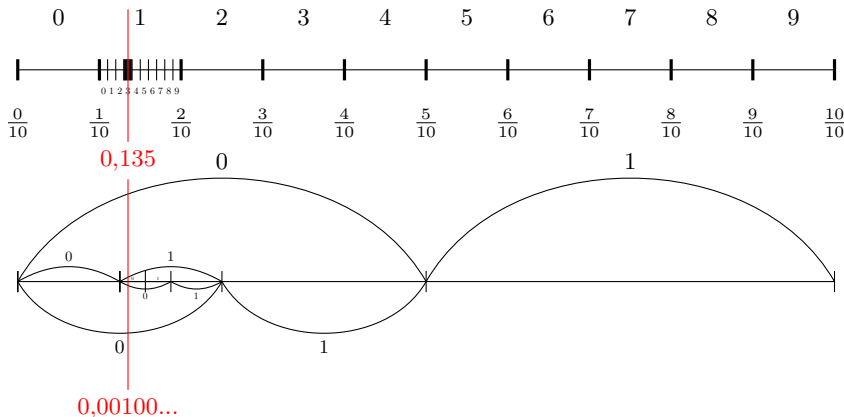
- ❶ Одинарная точность (float) — 23 бита мантиисы (7–8 десятичных цифр)
- ❷ Двойная точность (double) — 52 бита мантиисы (15–17 десятичных цифр)
- ❸ Расширенная двойная точность (long double в GCC, Extended в Паскале) — 64 бита мантиисы (19–20 десятичных цифр)

- ❶ Вещественные типы не подходят для чисел сверхвысокой точности.
- ❷ Действия над длинными числами необходимо выполнять посимвольно:
  - посимвольные вычисления (от младшего разряда) абсолютно точны;
  - посимвольный перевод  $CC1 \rightarrow CC2$  (от старшего разряда) неточен, если нет возможности работать с произвольными  $\frac{n}{m}$ , но погрешности перевода  $CC1 \rightarrow CC2$  и  $CC2 \rightarrow CC1$  в целых числах компенсируют друг друга (восстановится исходное число).



# Геометрическая интерпретация

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



# Посимвольная обработка: $D \leftrightarrow 2$

$$\overline{0, c_1 c_2 \dots c_{n_D}} \left( c_i \in \{0, \dots, D-1\} \right) \approx z = \overline{0, b_1 b_2 b_3 \dots b_{m_2}} \left( b_i \in \{0, 1\} \right)$$

$$\overline{0, c_1 c_2 \dots c_{n_D}} \leq z < \overline{0, c_1 c_2 \dots (c_n + 1)_D} \quad \text{считаем } \overline{0, (9+1)_{10}} = 1,0$$

	$i$	$l_i$	$t_i$
$[0, 1) \supseteq [l_1, t_1) \supseteq \dots \supseteq [l_n, t_n) \ni z$	0	0	1
	1	$\overline{0, c_1}_D$	$\overline{0, (c_1 + 1)}_D$
$l_i \leq z < t_i$	2	$\overline{0, c_1 c_2}_D$	$\overline{0, c_1 (c_2 + 1)}_D$
	...		
	$n$	$\overline{0, c_1 c_2 \dots c_{n_D}}_D$	$\overline{0, c_1 c_2 \dots (c_n + 1)}_D$

Накапливается погрешность

$$[0, 1) \supseteq [\lambda_1, \tau_1) \supseteq \dots \supseteq [\lambda_m, \tau_m) \supseteq \dots \ni z$$

$$\lambda_k \leq z < \tau_k$$

$$\underbrace{\overline{0, c_1 \dots c_i}}_{l_i} \xi_D \leq \lambda_k \text{ и } \tau_k \leq \underbrace{\overline{0, c_1 \dots c_i (\xi + 1)}_D}_{t_{i+1}}$$

$$\underbrace{\hspace{1.5cm}}_{l_{i+1}} \quad \downarrow \quad c_{i+1} = \xi$$

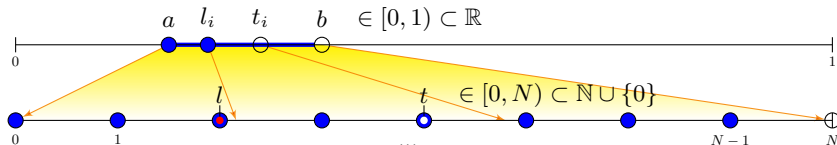
$k$	$\lambda_k$	$\tau_k$
0	0	1
1	$\overline{0, b_1}_2$	$\overline{0, (b_1 + 1)}_2$
2	$\overline{0, b_1 b_2}_2$	$\overline{0, b_1 (b_2 + 1)}_2$
...		
$m$	$\overline{0, b_1 \dots b_{m_2}}_2$	$\overline{0, b_1 \dots (b_m + 1)}_2$
$m+1$	$\overline{0, b_1 \dots b_{m_2} 0}_2$	$\overline{0, b_1 \dots b_m 1}_2$
$m+2$	$\overline{0, b_1 \dots b_{m_2} 00}_2$	$\overline{0, b_1 \dots b_m 01}_2$
...		



# Целочисленная реализация

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

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



Далее  $l$  и  $t$  — не настоящие  $l_i$  и  $t_i$  из  $[0, 1)$ , а их изображения на  $[0, N)$  (целые).

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  $[a, b] \Rightarrow$  новые изображения  $l$  и  $t$  масштабируются из старых изображений ( $l_i, t_i \in \mathbb{R}$  недоступны).

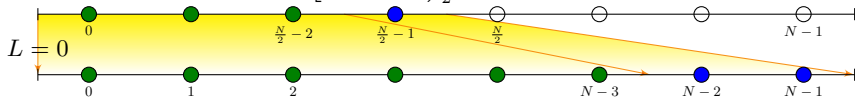
$D \neq 2^\alpha \Rightarrow$  границы цифр  $\frac{\Delta \cdot \xi}{D}$  / символов  $\frac{\Delta \cdot \omega_j}{D}$  пересчитываются по новым  $l, t$

$\Rightarrow$  **погрешность округления:** для уменьшения  $N : D^\gamma$  (также  $N : 2^\alpha$ ,  $\alpha \gg \gamma \gg 1$ )

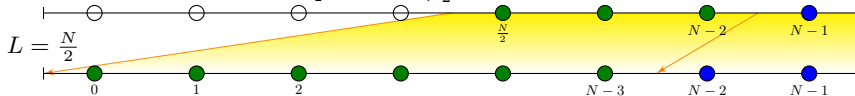
# Масштабирование и биты двоичного представления

Вначале  $[a, b) = [0, 1) \Rightarrow z \in [l_i, t_i) \subseteq [a, b)$  м.б. любым;  
 приближаем часть ПИ  $[L, L + \frac{N}{2})$  — изображаемый ПИ  $[a, b)$  уменьшается вдвое,  
 длины на изображении увеличиваются вдвое (масштабирование  $x \rightarrow 2(x - L)$ ).

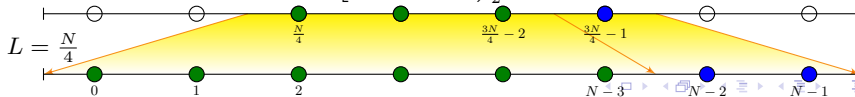
$[a, b) = [0, \frac{1}{2}) = [0, 0; 0, 1)_2 = [0, 0; 0, 0(1))_2$  : первый бит  $z \in [a, b)$  всегда 0



$[a, b) = [\frac{1}{2}, 1) = [0, 1; 1, 0)_2 = [0, 1; 0, 1(1))_2$  : 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



Позиционная запись  $[0; 1)$  по основанию  $D$   
 Посимвольная обработка:  $D \leftrightarrow 2$   
 Арифметическое сжатие  
 Полуинтервалы и отрезки

Целочисленная реализация  
 Масштабирование и биты двоичного представления  
 Перевод  $D \rightarrow 2 \diamond 021\_ar\_samples/*: 135, 112 \leftrightarrow$   
 Перевод  $2 \rightarrow D$

# Перевод $D \rightarrow 2 \diamond 021\_ar\_samples/*: 135, 112 \hookrightarrow$

Входной поток: цифры  $C = c_1 c_2 \dots c_n$ , выходной — биты  $B = b_1 b_2 b_3 \dots b_m$   
 $[l, t)$  соответствует текущей цифре  $c_i$ , текущий бит  $b_k$  — весь  $[0, N)$

❶  $l = 0, t = N$ , бит зарезервировано  $\beta = 0$ , № цифры  $i = 0$

❷ чтение  $(++i)$  цифры  $C \rightarrow c_i$ :  $\Delta = t - l$ ,  $\begin{cases} l \rightarrow l + \frac{\Delta \cdot c_i}{D} \\ t \rightarrow l + \frac{\Delta \cdot (c_i + 1)}{D} \end{cases}$

❸ масштабирование  $l, t$  и запись бита  $b = \frac{2L}{N}$ , пока возможно (м. б. несколько раз):

$$[l, t) \subseteq [0, \frac{N}{2}) \implies [0, \frac{N}{2}) \rightarrow [0, N) \text{ и } 0 \rightarrow B \left( \underbrace{011\dots 1}_{\beta} \rightarrow B, \beta \rightarrow 0 \right)$$

$$[l, t) \subseteq [\frac{N}{2}, N) \implies [\frac{N}{2}, N) \rightarrow [0, N) \text{ и } 1 \rightarrow B \left( \underbrace{100\dots 0}_{\beta} \rightarrow B, \beta \rightarrow 0 \right)$$

$$[l, t) \subseteq [\frac{N}{4}, \frac{3N}{4}) \implies [\frac{N}{4}, \frac{3N}{4}) \rightarrow [0, N) \text{ и } ++\beta$$

❹ если достигнуто  $i = n$  — запись  $1 \rightarrow B$  (фактически  $\underbrace{100\dots 0}_{\beta} \rightarrow B$ ,

но завершающие нули подразумеваются) и завершение;

иначе переход к шагу ❶

Если  $\frac{\Delta \cdot c_i}{D}$  и  $\frac{\Delta \cdot (c_i + 1)}{D}$  неточны — перевод неточен, но восстановление м. б. возможно



Позиционная запись  $[0; 1)$  по основанию  $D$   
 Посимвольная обработка:  $D \leftrightarrow 2$   
 Арифметическое сжатие  
 Полуинтервалы и отрезки

Целочисленная реализация  
 Масштабирование и биты двоичного представления  
 Перевод  $D \rightarrow 2 \diamond 021\_ar\_samples/*: 135, 112 \hookrightarrow$   
 Перевод  $2 \rightarrow D$

Перевод  $2 \rightarrow D$ 

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

$[\lambda, \tau)$  соответствует текущему биту  $b_k$ ,  $[l, t)$  — текущей цифре  $c_i = \xi$ ,  
 $[\lambda, \tau) \subseteq [l, t) \subseteq [0, N)$

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

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 цифры, если возможно:  $\Delta = t - l$ ,  $\xi \in \{0, \dots, D - 1\}$

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

③ масштабирование  $l, \lambda, \tau, t$ , пока возможно (м. б. несколько раз):

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

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

Если  $\frac{\delta \cdot b_k}{2}$  и  $\frac{\delta \cdot (b_k + 1)}{2}$  неточны — восстановление невозможно  $\implies N : 2^\alpha, \alpha \gg 1$

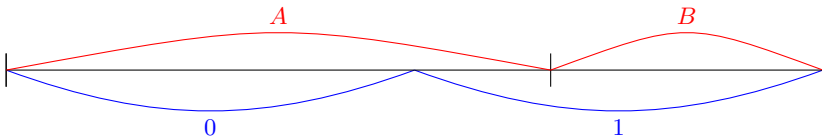
# Арифметическое сжатие

- 1 Цепочка символов  $T$ -ичного алфавита соответствует вещественному числу (точке) сверхвысокой точности в диапазоне  $[0, 1)$ .

Соответствие аналогично позиционной системе счисления по основанию  $T$ , но диапазон разбивается на  $T$  **неравных частей пропорционально частотам символов**.

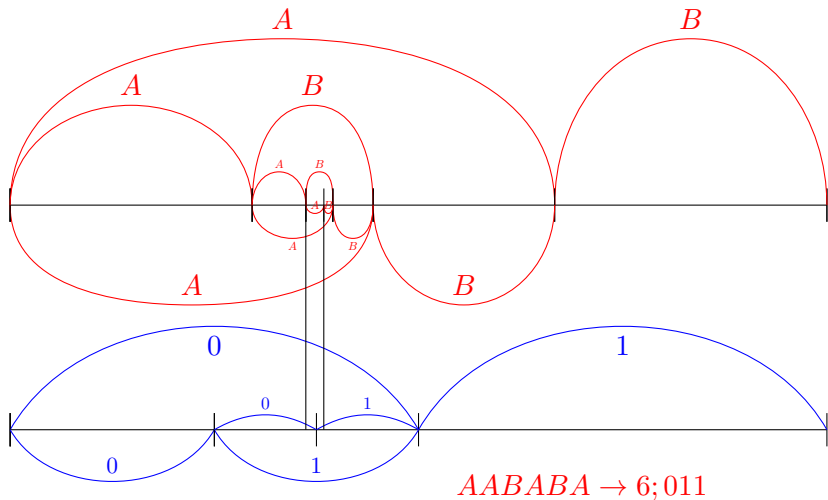
- 2 Полученная точка представляется в двоичной системе счисления.

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



**Интервальное кодирование** — целочисленное,  $[a, b) \subseteq [0, 1) \rightarrow [0, N)$

# Геометрическая интерпретация



# Арифметическое сжатие

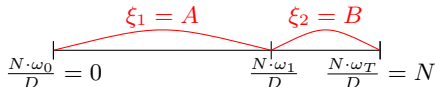
Отличие от перевода между СС — неравные вероятности символов  $\xi_j$ .

Алфавит из  $T$  символов:  $\xi_1, \xi_2, \dots, \xi_T$ , частоты  $\nu_1, \nu_2, \dots, \nu_T \in \mathbb{N} \cup \{0\}$ , сортируются по убыванию  $\nu_1 \geq \nu_2 \geq \dots \geq \nu_T$ .

Деление пропорционально  $\nu_j$ :

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

$D = \omega_T = \sum_j \nu_j$  — делитель.



Изменение отрезка при чтении символа  $c_i = \xi_j$ :

$$\Delta = t - l, \quad \begin{cases} l \rightarrow l + \frac{\Delta \cdot \omega_{j-1}}{D} \\ t \rightarrow l + \frac{\Delta \cdot \omega_j}{D} \end{cases}$$

- Выбор  $N$ : ①  $N \gg 4D^2$  (два деления  $\Delta > \frac{N}{4}$  — достаточно большие ПИ);  
 ②  $N = 2^\alpha$ ,  $\alpha \gg 1$  (перерасчёт  $\lambda$  и  $\tau$  **абсолютно точен**);  
 ③ расчёты производятся без переполнения ( $N \cdot D < \max(\text{type})$ ).

Сжатие  $\diamond$  octocat  $\hookrightarrow$ 

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  $C \rightarrow c_i = \xi_j$ :  $\Delta = t - l$ ,  $\begin{cases} l \rightarrow l + \frac{\Delta \cdot \omega_{j-1}}{D} \\ t \rightarrow l + \frac{\Delta \cdot \omega_j}{D} \end{cases}$
- 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  $b \rightarrow B$  ( $\underbrace{b \bar{b} \bar{b} \dots \bar{b}}_{\beta} \rightarrow B, \beta \rightarrow 0$ )  
 $\left[ \frac{N}{4}, \frac{3N}{4} \right) \rightarrow [0, N)$  и  $++\beta$
- 3 если достигнуто  $i = n$  — запись  $1 \rightarrow B$  ( $1 \underbrace{00 \dots 0}_{\beta} \rightarrow B$ ) 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 —  $\nu_j$  и  $\omega_j$  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  $B \rightarrow b_k$ :  $\delta = \tau - \lambda$ , 
$$\begin{cases} \lambda \rightarrow \lambda + \frac{\delta \cdot b_k}{2} \\ \tau \rightarrow \lambda + \frac{\delta \cdot (b_k + 1)}{2} \end{cases}$$

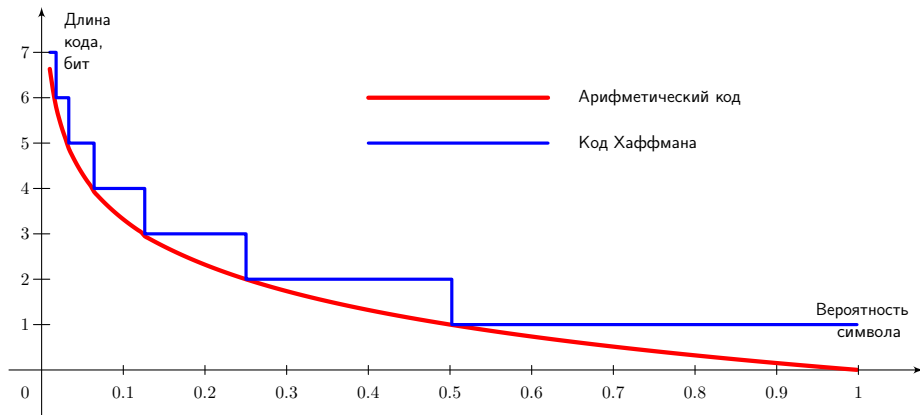
② получение и запись символа, если возможно:  $\Delta = t - l$ ,  $j \in \{1, \dots, T\}$

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

③ масштабирование  $l, \lambda, \tau, t$ , 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

# Сравнение с кодами Хаффмана



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

Не увеличивает размера исходных данных в худшем случае.

# 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  $\xi_j$  или бита  $b$

$$\begin{cases} l \rightarrow l + \frac{\Delta \cdot \omega_{j-1}}{D} \\ t \rightarrow l + \frac{\Delta \cdot \omega_j}{D} \end{cases}$$

$$\begin{cases} l \rightarrow l + \frac{\Delta \cdot \omega_{j-1}}{D} \\ h \rightarrow l + \frac{\Delta \cdot \omega_j}{D} - 1 \end{cases}$$

$$\begin{cases} \lambda \rightarrow \lambda + \frac{\delta \cdot b}{2} \\ \tau \rightarrow \lambda + \frac{\delta \cdot (b+1)}{2} \end{cases}$$

$$\begin{cases} \lambda \rightarrow \lambda + \frac{\delta \cdot b}{2} \\ \chi \rightarrow \lambda + \frac{\delta \cdot (b+1)}{2} - 1 \end{cases}$$

Масштабирование  $[L, L + \frac{N}{2}) \rightarrow [0, N)$ ,  $L \in \{0, \frac{N}{4}, \frac{N}{2}\}$

$$[l, t) \subseteq [L, L + \frac{N}{2}) \iff \begin{cases} L \leq l \\ t \leq L + \frac{N}{2} \end{cases}$$

$$[l, h] \subseteq [L, L + \frac{N}{2}) \iff \begin{cases} L \leq l \\ h < L + \frac{N}{2} \end{cases}$$

$$\begin{cases} l \rightarrow 2(l - L) \\ t \rightarrow 2(t - L) \end{cases}$$

$$\begin{cases} l \rightarrow 2(l - L) \\ h \rightarrow 2(h - L) + 1 \end{cases}$$

$$2t - 1 = 2(h + 1) - 1 = 2h + 1$$

Navigation icons: back, forward, search, etc.

Позиционная запись  $[0; 1)$  по основанию  $D$   
 Посимвольная обработка:  $D \leftrightarrow 2$   
 Арифметическое сжатие  
 Полуинтервалы и отрезки

Основные соотношения для  $t$  и  $h$

Д. Ватолин, целочисленный цикл (отрезки)

Д. Ватолин, запись бита (bits\_plus\_follow)

## 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 }
```



Позиционная запись  $[0; 1)$  по основанию  $D$   
Посимвольная обработка:  $D \leftrightarrow 2$   
Арифметическое сжатие  
Полуинтервалы и отрезки

Основные соотношения для  $t$  и  $h$   
Д. Ватолин, целочисленный цикл (отрезки)  
Д. Ватолин, запись бита (bits\_plus\_follow)

## Д. Ватолин, 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



Позиционная запись  $[0; 1)$  по основанию  $D$   
Посимвольная обработка:  $D \leftrightarrow 2$   
Арифметическое сжатие  
Полуинтервалы и отрезки

Основные соотношения для  $t$  и  $h$   
Д. Ватолин, целочисленный цикл (отрезки)  
Д. Ватолин, запись бита (bits\_plus\_follow)

# Спасибо за внимание!

МИЭТ

[www.miet.ru](http://www.miet.ru)

Александра Игоревна Кононова

[illinc@mail.ru](mailto:illinc@mail.ru)

[gitlab.com/illinc/raspisanie](https://gitlab.com/illinc/raspisanie)