Сжатие без учёта контекста. Арифметическое (интервальное) кодирование

Александра Игоревна Кононова

МИЭТ

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

Идея арифметического кодирования

1. сообщение $C = c_1c_2 \ldots c_n$ соответствует вещественному числу (точке) $z \in [0; 1)$;
2. точка z представляется в двоичной системе счисления.

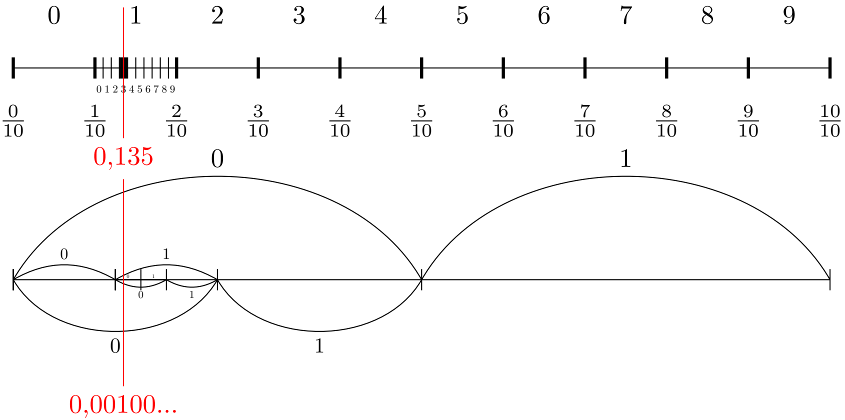
Сколько бит требуется для полной записи $z \in [0; 1)$?

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 $c_1c_2c_3\ldots$ — точка $z = \overline{0, c_1c_2c_3\ldots_D}$



Перевод $10 \to 2$ и $2 \to 10$

$$\overline{0,c_1c_2c_3\ldots} \cdot D = c_1 + \overline{0,c_2c_3\ldots}, \qquad 0 \leqslant c_1 < D, \quad \overline{0,c_2c_3\ldots} \in [0;1)$$

$0{,}135 \cdot 2$	$=$	$\mathbf{0},270$	$0{,}560 \cdot 2$	$=$	$\mathbf{1},120$	$0{,}360 \cdot 2$	$=$	$\mathbf{0},720$
$0{,}270 \cdot 2$	$=$	$\mathbf{0},540$	$0{,}120 \cdot 2$	$=$	$\mathbf{0},240$	$0{,}720 \cdot 2$	$=$	$\mathbf{1},440$
$0{,}540 \cdot 2$	$=$	$\mathbf{1},080$	$0{,}240 \cdot 2$	$=$	$\mathbf{0},480$	$0{,}440 \cdot 2$	$=$	$\mathbf{0},880$
$0{,}080 \cdot 2$	$=$	$\mathbf{0},160$	$0{,}480 \cdot 2$	$=$	$\mathbf{0},960$	$0{,}880 \cdot 2$	$=$	$\mathbf{1},760$
$0{,}160 \cdot 2$	$=$	$\mathbf{0},320$	$0{,}960 \cdot 2$	$=$	$\mathbf{1},920$	$0{,}760 \cdot 2$	$=$	$\mathbf{1},520$
$0{,}320 \cdot 2$	$=$	$\mathbf{0},640$	$0{,}920 \cdot 2$	$=$	$\mathbf{1},840$	$0{,}520 \cdot 2$	$=$	$\mathbf{1},040$
$0{,}640 \cdot 2$	$=$	$\mathbf{1},280$	$0{,}840 \cdot 2$	$=$	$\mathbf{1},680$	$0{,}040 \cdot 2$	$=$	$\mathbf{0},080$
$0{,}280 \cdot 2$	$=$	$\mathbf{0},560$	$0{,}680 \cdot 2$	$=$	$\mathbf{1},360$	$0{,}080 \cdot 2$	$=$	$\mathbf{0},160$

Далее всё повторится: $\quad 0{,}135_{10} = 0{,}001(00010100011110101110)_2$

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

Любое значение $z \in [0,1)$ имеет **бесконечное** число цифр (часть их может быть 0) в любой СС $\to$ округление

Перевод $10 \to 2$ (округление к нулю)

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

$0{,}135 \cdot 2$	$=$	$\mathbf{0}{,}270$	$0{,}136 \cdot 2$	$=$	$\mathbf{0}{,}272$
$0{,}270 \cdot 2$	$=$	$\mathbf{0}{,}540$	$0{,}272 \cdot 2$	$=$	$\mathbf{0}{,}544$
$0{,}540 \cdot 2$	$=$	$\mathbf{1}{,}080$	$0{,}544 \cdot 2$	$=$	$\mathbf{1}{,}088$
$0{,}080 \cdot 2$	$=$	$\mathbf{0}{,}160$	$0{,}088 \cdot 2$	$=$	$\mathbf{0}{,}176$
$0{,}160 \cdot 2$	$=$	$\mathbf{0}{,}320$	$0{,}176 \cdot 2$	$=$	$\mathbf{0}{,}352$
$0{,}320 \cdot 2$	$=$	$\mathbf{0}{,}640$	$0{,}352 \cdot 2$	$=$	$\mathbf{0}{,}704$
$0{,}640 \cdot 2$	$=$	$\mathbf{1}{,}280$	$0{,}704 \cdot 2$	$=$	$\mathbf{1}{,}408$
$0{,}280 \cdot 2$	$=$	$\mathbf{0}{,}560$	$0{,}408 \cdot 2$	$=$	$\mathbf{0}{,}816$
$0{,}560 \cdot 2$	$=$	$\mathbf{1}{,}120$	$0{,}816 \cdot 2$	$=$	$\mathbf{1}{,}632$
$0{,}120 \cdot 2$	$=$	$\mathbf{0}{,}240$	$0{,}632 \cdot 2$	$=$	$\mathbf{1}{,}264$

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

Перевод $2 \to 10$ (округление к нулю)

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

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

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

---

1. Одинарная точность (float) — 23 бита мантиссы (7–8 десятичных цифр)
2. Двойная точность (double) — 52 бита мантиссы (15–17 десятичных цифр)
3. Расширенная двойная точность (long double в GCC, Extended в Паскале) — 64 бита мантиссы (19–20 десятичных цифр)

---

1. Вещественные типы не подходят для чисел сверхвысокой точности.
2. Действия над длинными числами необходимо выполнять посимвольно.

Посимвольная обработка: $D \leftrightarrow 2$

$$\overline{0{,}c_1c_2\ldots c_n}_D \;\big(c_i \in \{0,\ldots,D-1\}\big) \approx z = \overline{0{,}b_1b_2b_3\ldots b_m}_2 \;\big(b_i \in \{0,1\}\big)$$

$$\overline{0{,}c_1c_2\ldots c_n}_D \leqslant z < \overline{0{,}c_1c_2\ldots(c_n+1)}_D$$

считаем $\overline{0{,}(9+1)}_{10} = 1{,}0$

$$[0,1) \supseteq [l_1,t_1) \supseteq \ldots \supseteq [l_n,t_n) \ni z$$

$$l_i \leqslant z < t_i$$

i	l_i	t_i
0	0	1
1	$\overline{0{,}c_1}_D$	$\overline{0{,}(c_1+1)}_D$
2	$\overline{0{,}c_1c_2}_D$	$\overline{0{,}c_1(c_2+1)}_D$
$\ldots$		
n	$\overline{0{,}c_1c_2\ldots c_n}_D$	$\overline{0{,}c_1c_2\ldots(c_n+1)}_D$

Накапливается погрешность

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

$$\lambda_k \leqslant z < \tau_k$$

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

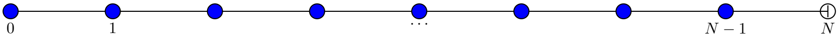
$$\Downarrow$$

$$c_{i+1} = \xi$$

k	λ_k	τ_k
0	0	1
1	$\overline{0{,}b_1}_2$	$\overline{0{,}(b_1+1)}_2$
2	$\overline{0{,}b_1b_2}_2$	$\overline{0{,}b_1(b_2+1)}_2$
$\ldots$		
m	$\overline{0{,}b_1\ldots b_m}_2$	$\overline{0{,}b_1\ldots(b_m+1)}$
$m+1$	$\overline{0{,}b_1\ldots b_m0}_2$	$\overline{0{,}b_1\ldots b_m1}_2$
$m+2$	$\overline{0{,}b_1\ldots b_m00}_2$	$\overline{0{,}b_1\ldots b_m01}_2$
$\ldots$		

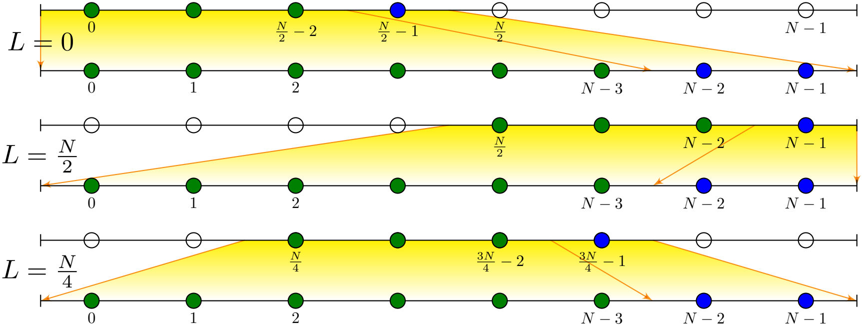
Целочисленная реализация

$[0,1) \to [0,N)$ $\qquad N >> D^2,\ \frac{N}{4} \in \mathbb{N}$



Возможно $l = t \implies$ масштабирование $[l,t) \subseteq [L, L+\frac{N}{2})$: $\begin{cases} l \to 2(l-L) \\ t \to 2(t-L) \end{cases}$

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



Перевод $D \to 2$ ♢ arc-135-*, $\hookrightarrow$

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 $[0, N)$

0. $l = 0, t = N$, бит зарезервировано $\beta = 0$, № цифры $i = 0$
1. чтение цифры $C \to c_i$: $\Delta = t - l$, $\begin{cases} l \to l + \frac{\Delta \cdot c_i}{D} \\ t \to l + \frac{\Delta \cdot (c_i + 1)}{D} \end{cases}$
2. масштабирование l, t и запись бита $b = \frac{2L}{N}$, пока возможно:
$$[l,t) \subseteq \left[0, \tfrac{N}{2}\right) \implies \left[0, \tfrac{N}{2}\right) \to [0, N) \text{ и } 0 \to B \left(0\,\underbrace{11\dots1}_{\beta} \to B,\ \beta \to 0\right)$$
$$[l,t) \subseteq \left[\tfrac{N}{4}, \tfrac{3N}{4}\right) \implies \left[\tfrac{N}{4}, \tfrac{3N}{4}\right) \to [0, N) \text{ и } {++\beta}$$
$$[l,t) \subseteq \left[\tfrac{N}{2}, N\right) \implies \left[\tfrac{N}{2}, N\right) \to [0, N) \text{ и } 1 \to B \left(1\,\underbrace{00\dots0}_{\beta} \to B,\ \beta \to 0\right)$$
3. $++i$ и переход к чтению цифры c_i (к шагу 1);
если невозможно $(i > n)$ — завершение и запись $1 \to B$ (так как $\frac{N}{2} \in [l, t)$)
(фактически $1\,\underbrace{00\dots0}_{\beta} \to B$, но завершающие нули подразумеваются)

Перевод $2 \to D$

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 $c_i = \xi$, $[\lambda, \tau) \subseteq [l, t) \subseteq [0, N)$

0. $l = \lambda = 0, t = \tau = N$, № бита $k = 0$, № цифры $i = 0$
1. чтение бита $B \to b_k$: $\Delta = \tau - \lambda$, $\begin{cases} \lambda \to \lambda + \frac{\Delta \cdot b_k}{2} \\ \tau \to \lambda + \frac{\Delta \cdot (b_k + 1)}{2} \end{cases}$
2. получение и запись цифры, если возможно: $\Delta = t - l$, $\xi \in \{0, \dots, D-1\}$

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

3. масштабирование l, λ, τ, t, пока возможно: $[l, t) \subseteq [L, L + \frac{N}{2}) \implies$
$\implies [L, L + \frac{N}{2}) \to [0, N)$ (не влияет на выходной поток)
4. $++k$ и переход к чтению бита b_k (к шагу 1);
если достигнуто $i = n$ — завершение

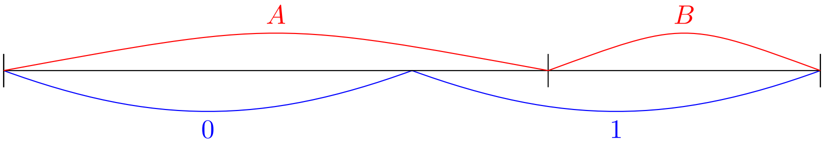
Арифметическое сжатие

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.**

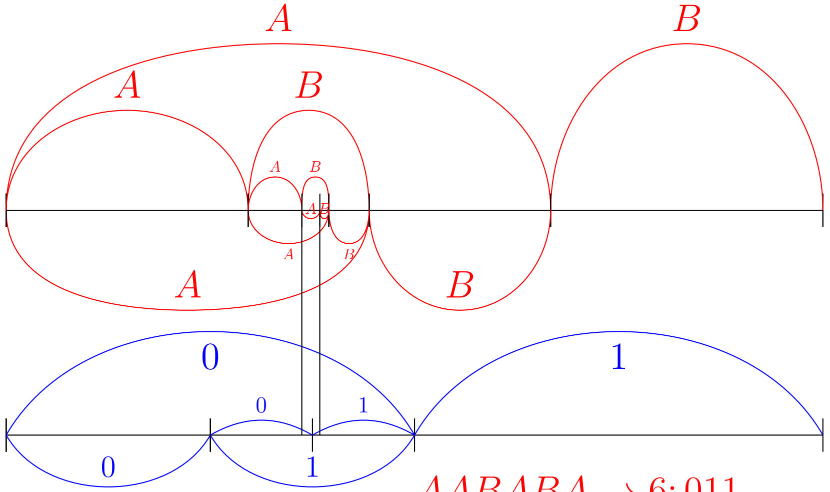
2. Полученная точка представляется в двоичной системе счисления.

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



Интервальное кодирование — целочисленное, $[0, 1) \to [0, N)$

Геометрическая интерпретация



$AABABA \to 6; 011$

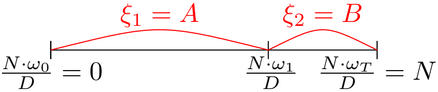
Арифметическое сжатие

Отличие от перевода между СС — неравные вероятности символов ξ_j.

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

Деление пропорционально ν_j: $\begin{cases} \omega_0 & = & 0, \\ \ldots & & \\ \omega_j & = & \omega_{j-1} + \nu_j, \\ \ldots & & \\ \omega_T & = & \omega_{T-1} + \nu_T. \end{cases}$

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



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

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

Сжатие ♢ arc-octocat ↪

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 символа $C \to c_i = \xi_j$: $\Delta = t - l$, $\begin{cases} l \to l + \frac{\Delta \cdot \omega_{j-1}}{D} \\ t \to l + \frac{\Delta \cdot \omega_j}{D} \end{cases}$
2. масштабирование l, t: $\begin{cases} l \to 2(l - L) \\ t \to 2(t - L) \end{cases}$ и запись бита b, пока возможно:
   $\left[\frac{b \cdot N}{2}, \frac{(b+1) \cdot N}{2}\right), b \in \{0, 1\} \to [0, N)$ и запись $b \to B$ $\left(b \underbrace{\bar{b}\bar{b} \dots \bar{b}}_{\beta} \to B,\ \beta \to 0\right)$
   $\left[\frac{N}{4}, \frac{3N}{4}\right) \to [0, N)$ и $++\beta$
3. $++i$ и переход к чтению символа c_i (к шагу 1);
   если невозможно $(i > n)$ — завершение и запись $1 \to B$

---

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

Распаковка

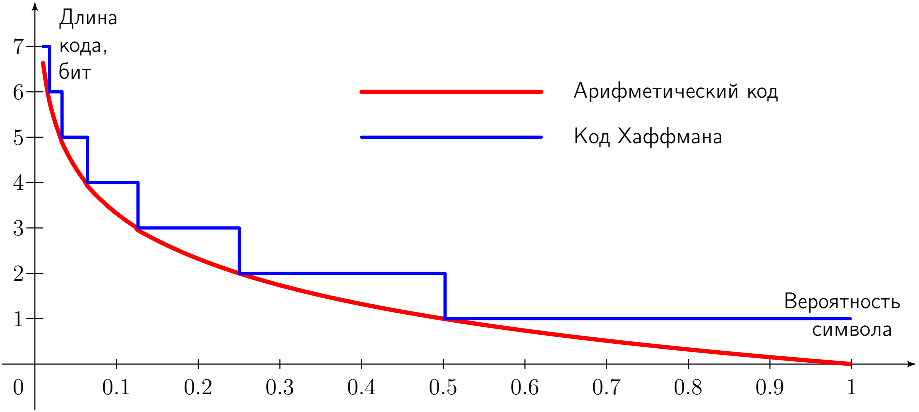
Символов — n, входной поток — биты $B = b_1 b_2 b_3 \ldots b_m 000\ldots$,
выходной поток — символы $C = c_1 c_2 \ldots c_n$.

0. $l = \lambda = 0, t = \tau = N$, № бита $k = 0$, № символа $i = 0$
1. чтение бита $B \to b_k$: $\Delta = \tau - \lambda$, $\begin{cases} \lambda \to \lambda + \frac{\Delta \cdot b_k}{2} \\ \tau \to \lambda + \frac{\Delta \cdot (b_k+1)}{2} \end{cases}$
2. получение и запись символа, если возможно: $\Delta = t - l$, $j \in \{1, \ldots, T\}$

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

3. масштабирование l, λ, τ, t, пока возможно (не влияет на выходной поток): $[l, t) \subseteq [L, L + \frac{N}{2}) \Longrightarrow [L, L + \frac{N}{2}) \to [0, N)$
4. $++k$ и переход к чтению бита b_k (к шагу 1);
   если достигнуто $i = n$ — завершение

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** $[l, t)$, $l \leqslant z < t$,
где t — **невключаемая** верхняя граница.

Тот же самый диапазон можно представить
как **отрезок** $[l, h]$, $l \leqslant z \leqslant h$,
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

$$[l,t) = [l,h] \quad\Longleftrightarrow\quad h = t-1 \quad\Longleftrightarrow\quad t = h+1$$

Полуинтервал $[l,t)$, $l \leqslant z < t$	Отрезок $[l,h]$, $l \leqslant z \leqslant h$
Длина	
$\Delta = t - l \qquad (\tau - \lambda)$	$\Delta = h - l + 1 \qquad (\chi - \lambda + 1)$
Чтение символа ξ_j или бита b	
$\begin{cases} l \to l + \frac{\Delta\cdot\omega_{j-1}}{D} \\ t \to l + \frac{\Delta\cdot\omega_j}{D} \end{cases}$	$\begin{cases} l \to l + \frac{\Delta\cdot\omega_{j-1}}{D} \\ h \to l + \frac{\Delta\cdot\omega_j}{D} - 1 \end{cases}$
$\begin{cases} \lambda \to \lambda + \frac{\Delta\cdot b}{2} \\ \tau \to \lambda + \frac{\Delta\cdot(b+1)}{2} \end{cases}$	$\begin{cases} \lambda \to \lambda + \frac{\Delta\cdot b}{2} \\ \chi \to \lambda + \frac{\Delta\cdot(b+1)}{2} - 1 \end{cases}$

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

$[l,t) \subseteq \left[L, L+\frac{N}{2}\right) \Longleftrightarrow \begin{cases} L \leqslant l \\ t \leqslant L + \frac{N}{2} \end{cases}$	$[l,h] \subseteq \left[L, L+\frac{N}{2}\right) \Longleftrightarrow \begin{cases} L \leqslant l \\ h < L + \frac{N}{2} \end{cases}$
$\begin{cases} l \to 2(l-L) \\ t \to 2(t-L) \end{cases}$	$\begin{cases} l \to 2(l-L) \\ h \to 2(h-L)+1 \end{cases}$
	$2t - 1 = 2(h+1) - 1 = 2h + 1$

Д. Ватолин, целочисленный цикл (отрезки)

```
l[0] = 0; h[0] = 65535; i = 0; delitel = b[c_last];
First_qtr = (h[0] + 1)/4; Half = First_qtr*2; Third_qtr = First_qtr*3;
bits_to_follow = 0; // масштабирований [First_qtr; Third_qtr)
while (not DataFile.EOF()) {
  c = DataFile.ReadSymbol(); i++; // Кодируемый символ
  j = IndexForSymbol(c);          // и его номер в алфавите
  l[i] = l[i-1] + b[j-1]*(h[i-1] - l[i-1] + 1)/delitel;
  h[i] = l[i-1] +   b[j]*(h[i-1] - l[i-1] + 1)/delitel - 1;
  for(;;) {                   // Варианты масштабирования
    if (h[i] < Half)          // [l; h] лежит в [0; Half)
      bits_plus_follow(0);
    else if (l[i] >= Half) {  // [l; h] лежит в [Half, max)
      bits_plus_follow(1);
      l[i] -= Half; h[i] -= Half;
    }
    else if ((l[i] >= First_qtr) && (h[i] < Third_qtr)) {
      bits_to_follow++;
      l[i] -= First_qtr; h[i] -= First_qtr;
    } else break;
    l[i] += l[i]; h[i] += h[i] + 1; // масштабирование *2
  }
}
```

Д. Ватолин, запись бита (bits_plus_follow)

```
void bits_plus_follow (int bit)
{
  CompressedFile.WriteBit(bit);
  for(; bits_to_follow > 0; bits_to_follow--)
    CompressedFile.WriteBit(!bit);
}
```

`bits_to_follow` — количество $\left[\frac{1}{4}; \frac{3}{4}\right) \to \left[0; 1\right)$

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

Спасибо за внимание!

МИЭТ
`http://miet.ru/`

Александра Игоревна Кононова
`illinc@mail.ru`