Семинар: арифметическое (интервальное) кодирование. Побайтовая запись кода. Случай сообщения без кода

Александра Игоревна Кононова

МИЭТ

8 июля 2024 г. — актуальную версию можно найти на
`https://gitlab.com/illinc/otik`

Кодирование

$C = 5111$, $n = 4$, $T = 2$,

j	1	2
ξ_j	1	5
ν_j	3	1

, $D = 3 + 1 = 4$ (степень двойки $\implies$ без приведения)

ПИ $[l_i, t_i)$ делится $3 : 1$, границы $\Omega_0 = l_i$, $\Omega_1 = l_i + \frac{3\Delta_i}{4}$, $\Omega_2 = l_i + \Delta_i = t_i$, где $\Delta_i = t_i - l_i$:

i	c_i	l_i	t_i	Δ_i
0	—	0	1	1
1	5	$\frac{3}{4}$	1	$\frac{1}{4}$
2	1	$\frac{3}{4}$	$\frac{15}{16}$	$\frac{3}{16}$
3	1	$\frac{3}{4}$	$\frac{57}{64}$	$\frac{9}{64}$
4	1	$\frac{3}{4}$	$\frac{219}{256}$	$\frac{27}{256}$

Далее выбираем $z \in \left[\frac{3}{4}, \frac{219}{256}\right)$ с самым коротким двоичным представлением $\implies$ $z = \frac{3}{4} = 0{,}11_2$.
Числу $z = 0{,}1100\ldots_2$ соответствует битовый поток $B = 110$ и, если первый бит потока записывается старшим разрядом, байт $110_2 = 6$ (если первый младшим, то $011_2 = 3$; здесь не рассматриваем).

В файл записываются $\underbrace{0004}_{n}$, $\underbrace{0300\,0100}_{\vec{\nu}}$ и код 6.

Д-1 [декодирование, вариант 1]: чтение z целиком

$\underbrace{0004}_{n}$, $\underbrace{0300\,0100}_{\vec{\nu}}$, $\underbrace{6}_{\text{код}}$, откуда $n = 4$,

j	1	2
ξ_j	1	5
ν_j	3	1

, $D = 4$.

Пусть после прочтения всех байтов кода битовый поток выдаёт отказ:

k	b_k	λ_k	τ_k	δ_k
0	—	0	1	1
1	1	$\frac{1}{2}$	1	$\frac{1}{2}$
2	1	$\frac{3}{4}$	1	$\frac{1}{4}$
3	0	$\frac{3}{4}$	$\frac{7}{8}$	$\frac{1}{8}$
4	—	$\frac{3}{4}$	—	0

битовый поток прочитан до конца, $z = \frac{3}{4} = 0{,}11_2$.

i	c_i	l_i	t_i	Δ_i	Ω_i^0	Ω_i^1	Ω_i^2	комментарий
0	—	0	1	1	0	$\frac{3}{4}$	1	$z \in [\Omega_i^1, \Omega_i^2)$
1	5	$\frac{3}{4}$	1	$\frac{1}{4}$	$\frac{3}{4}$	$\frac{15}{16}$	1	$z \in [\Omega_i^0, \Omega_i^1)$
2	1	$\frac{3}{4}$	$\frac{15}{16}$	$\frac{3}{16}$	$\frac{3}{4}$	$\frac{57}{64}$	$\frac{15}{16}$	$z \in [\Omega_i^0, \Omega_i^1)$
3	1	$\frac{3}{4}$	$\frac{57}{64}$	$\frac{9}{64}$	$\frac{3}{4}$	$\frac{219}{256}$	$\frac{57}{64}$	$z \in [\Omega_i^0, \Omega_i^1)$
4	1	$\frac{3}{4}$	$\frac{219}{256}$	$\frac{27}{256}$	$\frac{3}{4}$	$\frac{849}{1024}$	$\frac{219}{256}$	$i = n$

$C = 5111$

Д-1 [декодирование, вариант 1]: чтение z целиком

Пусть после прочтения всех байтов кода битовый поток выдаёт нули:

k	b_k	λ_k	τ_k	δ_k	i	c_i	l_i	t_i	Δ_i	Ω_i^0	Ω_i^1	Ω_i^2	комментарий
0	–	0	1	1									
1	1	$\frac{1}{2}$	1	$\frac{1}{2}$									
2	1	$\frac{3}{4}$	1	$\frac{1}{4}$									
3	0	$\frac{3}{4}$	$\frac{7}{8}$	$\frac{1}{8}$									
4	0	$\frac{3}{4}$	$\frac{13}{16}$	$\frac{1}{16}$									$\frac{13}{16} = \frac{52}{64} = \frac{208}{256}$
5	0	$\frac{3}{4}$	$\frac{25}{32}$	$\frac{1}{32}$									$\frac{25}{32} = \frac{50}{64} = \frac{200}{256}$
					0	–	0	1	1	0	$\frac{3}{4}$	1	
					1	5	$\frac{3}{4}$	1	$\frac{1}{4}$	$\frac{3}{4}$	$\frac{15}{16}$	1	
					2	1	$\frac{3}{4}$	$\frac{15}{16}$	$\frac{3}{16}$	$\frac{3}{4}$	$\frac{57}{64}$	$\frac{15}{16}$	
					3	1	$\frac{3}{4}$	$\frac{57}{64}$	$\frac{9}{64}$	$\frac{3}{4}$	$\frac{219}{256}$	$\frac{57}{64}$	
					4	1	$\frac{3}{4}$	$\frac{219}{256}$	$\frac{27}{256}$	$\frac{3}{4}$	$\frac{849}{1024}$	$\frac{219}{256}$	$i = n$

$C = 5111$

На каком k остановиться?

Д-2 [декодирование, вариант 2]: чтение z побитово

k	b_k	λ_k	τ_k	δ_k	i	c_i	l_i	t_i	Δ_i	Ω_i^0	Ω_i^1	Ω_i^2	комментарий
0	—	0	1	1	0	—	0	1	1	0	$\frac{3}{4}$	1	
1	1	$\frac{1}{2}$	1	$\frac{1}{2}$									$\lambda_k \in [\Omega_i^0, \Omega_i^1), \tau_k \in [\Omega_i^1, \Omega_i^2)$
2	1	$\frac{3}{4}$	1	$\frac{1}{4}$									$[\lambda_k, \tau_k) \subseteq [\Omega_i^1, \Omega_i^2)$
					1	5	$\frac{3}{4}$	1	$\frac{1}{4}$	$\frac{3}{4}$	$\frac{15}{16}$	1	$\lambda_k \in [\Omega_i^0, \Omega_i^1), \tau_k \in [\Omega_i^1, \Omega_i^2)$
3	0	$\frac{3}{4}$	$\frac{7}{8}$	$\frac{1}{8}$									$\frac{7}{8} = \frac{14}{16} \implies [\lambda_k, \tau_k) \subseteq [\Omega_i^0, \Omega_i^1)$
					2	1	$\frac{3}{4}$	$\frac{15}{16}$	$\frac{3}{16}$	$\frac{3}{4}$	$\frac{57}{64}$	$\frac{15}{16}$	$\frac{7}{8} = \frac{56}{64}$
					3	1	$\frac{3}{4}$	$\frac{57}{64}$	$\frac{9}{64}$	$\frac{3}{4}$	$\frac{219}{256}$	$\frac{57}{64}$	$\frac{7}{8} = \frac{224}{256}$
4	0	$\frac{3}{4}$	$\frac{13}{16}$	$\frac{1}{16}$									$\frac{13}{16} = \frac{208}{256}$
					4	1	$\frac{3}{4}$	$\frac{219}{256}$	$\frac{27}{256}$	$\frac{3}{4}$	$\frac{849}{1024}$	$\frac{219}{256}$	$i = n$

$C = 5111$

Варианты 1 и 2 — крайние случаи; при точных вычислениях можно комбинировать шаги по k и i произвольно (i — не раньше, чем в варианте 2).

Выбор N; параметры

В общем случае $N \gg 4D^2$ (при $D = 4$: $4D^2 = 64$) и $N \cdot D \leqslant \max(type)$;
в частности, при $D > 10$ возможно $N = 4D^3$. Для $D = 4$ нельзя сказать $4D^3 \gg 4D^2$, но при $D = 2^\kappa$ вычисления точнее (но никогда не абсолютно точны!) $\implies$ возьмём $N = 4D^3 = 256$.

$N = 2^\alpha \geqslant 4D^3$ — для побитовой записи со своевременным масштабированием, включая масштабирование «из средней половины» $\left[\frac{N}{4}, \frac{3N}{4}\right) \to [0, N)$ (переменная β).

1. Любое снижение точности — побайтовая запись, отложенное масштабирование $\implies$ увеличение N.
2. Отказ от β $\implies$ $\Delta \geqslant 1$ вместо $\Delta \geqslant \frac{N}{4}$ $\implies$ невозможность оценить необходимое N.

Худший случай без β для $T > 2$: рабочий ПИ $[l, t) = \left[\frac{N}{2} - 1, \frac{N}{2}\right] = \left[\frac{N}{2} - 1, \frac{N}{2} + 1\right)$ из двух пикселей и на границе $0/1$ — невозможно ни масштабирование, ни деление на T частей.
Возможен без β для любых $\frac{N}{D}$ при особенно неудачном исходном тексте.

---

j	0	1	2
ξ_j	–	1	5
ν_j	–	3	1
ω_j	0	3	4

$D = \omega_0 = 4$,
деление ПИ $[l, t)$ длины $\Delta = t - l$
на ПИ $[\Omega_0, \Omega_1)$ для $\xi_1 = 1$ и ПИ $[\Omega_1, \Omega_2)$ для $\xi_2 = 5$:

$\Omega_0 = l + \frac{\omega_0 \cdot \Delta}{D} = l$; $\quad \Omega_1 = l + \frac{\omega_1 \cdot \Delta}{D} = l + \frac{3\Delta}{4}$; $\quad \Omega_2 = l + \frac{\omega_2 \cdot \Delta}{D} = l + \Delta = t$; $\quad$ округление вниз.

Рассмотрим, кроме $C = 5111$, ещё $\widetilde{C} = 5111\,1115$ с той же таблицей частот.

Кодирование 5111, $N = 256$

$C = 5111$: $\vec{\omega} = (0, 3, 4)$, $D = 4$; $N = 256$ $\left(\frac{N}{4} = 64, \frac{N}{2} = 128, \frac{3N}{4} = 192\right)$:

i	c_i	k	b_k	β	l	t	Δ	комментарий	z (двоичное)
0	–	0	–	0	0	256	256		$0{,}b_1b_2b_3b_4\ldots$
1	5				192	256	64	$[l,t) \subseteq \left[\frac{N}{2}, N\right) \implies 1$; $\left[\frac{N}{2}, N\right) \to [0, N)$	$0{,}1b_2b_3b_4\ldots$
		1	1	0	128	256	128	$[l,t) \subseteq \left[\frac{N}{2}, N\right) \implies 1$; $\left[\frac{N}{2}, N\right) \to [0, N)$	$0{,}11b_3b_4\ldots$
		2	1	0	0	256	256		
2	1				0	192	192		
3	1				0	144	144		
4	1				0	108	108	$[l,t) \subseteq \left[0, \frac{N}{2}\right) \implies 0$; $\left[0, \frac{N}{2}\right) \to [0, N)$	$0{,}110b_4\ldots$
		3	0	0	0	216	216	$i = n$, нельзя масштабировать $\to$ завершение	

В выходном потоке 110 — это дробная часть пикселя 0 (код пикселя $N - 110(1)$);
код сообщения — дробная часть *любой* точки, изображение которой лежит в $[l, t)$.

$0 \in [l, t)$, поэтому код остаётся как есть: $110_2 = 6$; в файл записываются $\underbrace{0004}_{n}$, $\underbrace{0300\,0100}_{\vec{\nu}}$ и код 6.

Если $0 \notin [l, t)$: раз нельзя масштабировать, то $\frac{N}{2} \in [l, t) \implies$ к коду добавляется одна 1.

Кодирование 5111 1115, $N = 256$

$\widetilde{C} = 5111\,1115$: $\vec{\omega} = (0,3,4)$, $D = 4$; $N = 256$ $\left(\frac{N}{4} = 64, \frac{N}{2} = 128, \frac{3N}{4} = 192\right)$; начало как $C = 5111$:

i	c_i	k	b_k	β	l	t	Δ	комментарий
4	1				0	108	108	
		3	0	0	0	216	216	завершение для 5111, но здесь $i < n$
5	1				0	81	81	
		4	0	0	0	162	162	
6	1				0	121	121	$\frac{3\cdot 162}{4} = 121{,}5$; округление всегда вниз
		5	0	0	0	242	242	
7	1				0	181	181	
8	5				135	181	46	
		6	1	0	14	106	92	
		7	0	0	28	212	184	завершение: $i = n$, нельзя масштабировать

В выходном потоке B уже находится код 110010, но $z = 0{,}1100100000\ldots$ соответствует пикселю 0, который не лежит в $[l,t)$. Несуществующему (и, соответственно, не лежащему в $[l,t)$) пикселю $N = 256$ соответствует бесконечная дробь $z = 0{,}110010(1)$.
Пикселю $\frac{N}{2} = 128$ соответствует $z = 0{,}1100101$, и $128 \in [l,t) \implies$ добавляем в выходной поток ещё 1.
Код $B = 110\ 010\ 100 = 624$; в файл: $\underbrace{0008}_{n}$, $\underbrace{0300\,0100}_{\vec{\nu}}$, 624.

Всегда ли при большом $N = 2^\alpha$ и $D = 4$ вычисления точные?

Для любого N при частотах $0300\,0100$ и достаточно большой последовательности $1111111\ldots$ вычисления станут неточными:

- на два-три шага чтения 1 придётся в среднем одно масштабирование (за два Δ изменится в $\frac{9}{16} > \frac{1}{2}$ раз, за три в $\frac{27}{64} < \frac{1}{2}$);
- шаг чтения 1 увеличивает знаменатель в 4 раза;
- масштабирование уменьшает знаменатель только в 2 раза.

В общем случае целочисленного АС код а) не совпадает с концептом и б) может различаться:

- при разных N;
- при разной частоте масштабирования;
- выполняется или нет масштабирование $\left[\frac{N}{4}, \frac{3N}{4}\right) \to [0, N)$ («из средней половины»).

Д-1 0004 0300 0100 6=110, $N = 256$ $\left(\frac{N}{4} = 64, \frac{N}{2} = 128, \frac{3N}{4} = 192\right)$

k	b_k	δ	λ	τ	i	c_i	l	t	Δ	Ω_0	Ω_1	Ω_2	комментарий
0	—	256	0	256	0	—	0	256	256	0	192	256	
1	1	128	128	256									
2	1	64	192	256									
3	0	32	192	224									
4	0	16	192	208									
5	0	8	192	200									
6	0	4	192	196									
7	0	2	192	194									
8	0	1	192	193									$\delta = 1$: если $\lambda \in$ ПИ, то и $[\lambda, \lambda + 1)$ там же
					1	5	192	256	64				
		2	128	130			128	256	128				
		4	0	4			0	256	256				
9	0	2	0	2									
10	0	1	0	1									
										0	192	256	сравниваем λ с Ω_j
					2	1	0	192	192	0	144	192	
					3	1	0	144	144	0	108	144	
					4	1	0	108	108				$i = n$
		2	0	2			0	216	216				

Д-2 0004 0300 0100 6, $N = 256$ $\left(\frac{N}{4} = 64, \frac{N}{2} = 128, \frac{3N}{4} = 192\right)$ (только начало)

k	b_k	δ	λ	τ	i	c_i	l	t	Δ	Ω_0	Ω_1	Ω_2	комментарий
0	–	256	0	256	0	–	0	256	256	0	192	256	
1	1	128	128	256									
2	1	64	192	256									$[\lambda, \tau) \subseteq [\Omega_1, \Omega_2)$
					1	5	192	256	64				$[l, t) \subseteq \left[\frac{N}{2}, N\right)$
		128	128	256			128	256	128				

...

Д-1 0008 0300 0100 624=110 010 100, $N = 256$ $\left(\frac{N}{4} = 64, \frac{N}{2} = 128, \frac{3N}{4} = 192\right)$

k	b_k	δ	λ	τ	i	c_i	l	t	Δ	Ω_0	Ω_1	Ω_2	комментарий
0	—	256	0	256	0	—	0	256	256	0	192	256	
1	1	128	128	256									
2	1	64	192	256									
3	0	32	192	224									
4	0	16	192	208									
5	1	8	200	208									
6	0	4	200	204									
7	1	2	202	204									
8	0	1	202	203									$\lambda = 202_{10} = 11001010_2 = B[1...8]$
					1	5	192	256	64				
		2	148	150			128	256	128				
		4	40	44			0	256	256				
9	0	2	40	42									
10	0	1	40	41						0	192	256	

Д-1 0008 0300 0100 624=110 010 100, $N = 256$ $\left(\frac{N}{4} = 64, \frac{N}{2} = 128, \frac{3N}{4} = 192\right)$ (продолжение)

k	b_k	δ	λ	τ	i	c_i	l	t	Δ	Ω_0	Ω_1	Ω_2	комментарий
10	0	1	40	41			0	256	256	0	192	256	
					2	1	0	192	192	0	144	192	
					3	1	0	144	144	0	108	144	
					4	1	0	108	108				
		2	80	82			0	216	216				
11	0	1	80	81									
					5	1	0	162	162	0	121	162	
					6	1	0	121	121				
		2	160	162			0	242	242				
12	0	1	160	161						0	181	242	
					7	1	0	181	181	0	135	181	
					8	5	0	135	181				$i = n$

Целочисленный АС: побайтовая запись

- В начале Д-1 для $N = 2^\alpha$ читается α бит до $\delta = 1$. Целое число k-битных байтов $\implies$ α делится на k.
- Запись кода байтами вместо битов — отложенное масштабирование $\implies$ N должно быть больше, чем для побитовой записи.

---

1. Для чтения/записи кода 3-битными байта рассмотрим $N = 2^9 = 512$, тогда все пиксели (в том числе l и t) имеют координаты из $[0, 512)$ — $\frac{\alpha}{k} = \frac{9}{3} = 3$-байтовые. Все числа запишем **в восьмеричной системе** (цифра=трёхбитный байт): все пиксели $\frac{\alpha}{k} = 3$-значные.
2. При масштабировании записывается целый **байт**, а длины Δ и δ увеличиваются не в два, а в 2^k раз (для трёхбитного байта — в восемь раз):
   - ПИ $[0, N)$ делится не на две половины (бит 0 и бит 1), а на $S = 2^k$ частей (для $k = 3$ на восемь): для $0 \leqslant b < S$ границы байта b: $\left[\frac{b \cdot N}{S}, \frac{(b+1) \cdot N}{S}\right)$ $\left[b00, (b+1)00\right) = \left[b00, b77\right]$
   - все пиксели, принадлежащие ПИ $\left[\frac{b \cdot N}{S}, \frac{(b+1) \cdot N}{S}\right)$, имеют одинаковый старший байт b (старший из $\frac{\alpha}{k}$ значащих, а не в целом типе);
   - $[l, t) \subseteq \left[\frac{b \cdot N}{2^k}, \frac{(b+1) \cdot N}{2^k}\right) \implies$ запись байта b и $\begin{cases} l_{\text{новое}} = 2^k \cdot (l - \frac{b \cdot N}{2^k}) = 10_8 \cdot (l - b00), \\ t_{\text{новое}} = 2^k \cdot (t - \frac{b \cdot N}{2^k}) = 10_8 \cdot (t - b00). \end{cases}$
3. Из статей: аналога $\left[\frac{N}{4}, \frac{3N}{4}\right) \to [0, N)$ нет, поэтому β не используется (аналог можно придумать, но побитовое со всем возможным масштабированием всё равно точнее; реализация без β — быстрее).

Масштабирование битовыми операциями

- Масштабирование арифметическими операциями выполняется единообразно:
  $\exists b$: $\frac{b\cdot N}{2^k} \leqslant l$ и $t \leqslant \frac{(b+1)\cdot N}{2^k}$ $\implies$ запись байта b и $\begin{cases} l_{\text{новое}} = 2^k\cdot\left(l - \frac{b\cdot N}{2^k}\right) = 10_8\cdot(l - b00), \\ t_{\text{новое}} = 2^k\cdot\left(t - \frac{b\cdot N}{2^k}\right) = 10_8\cdot(t - b00); \end{cases}$
- для l это побитовый сдвиг влево на байт $(l = bx_1x_2 \ \to\ l_{\text{новое}} = x_1x_2 0)$ в рамках α бит;
- если $t < \frac{(b+1)\cdot N}{2^k}$, то для t это тоже сдвиг $(t = by_1y_2 \ \to\ t_{\text{новое}} = y_1y_2 0)$ в рамках α бит;
  но если $t = \frac{(b+1)\cdot N}{2^k} = (b+1)00$, то $t_{\text{новое}} = N = 1000$ вместо 000;
  таким образом, $t_{\text{новое}} = \begin{cases} (t \operatorname{shl} k)\ \&\ (N-1), & t < \frac{(b+1)\cdot N}{2^k}, \\ N, & t = \frac{(b+1)\cdot N}{2^k} \Leftrightarrow (t \operatorname{shl} k)\ \&\ (N-1) = 0; \end{cases}$
  пусть $t_{\text{нн}} = (t \operatorname{shl} k)\ \&\ (N-1)$, тогда $t_{\text{новое}} = \begin{cases} t_{\text{нн}}, & t_{\text{нн}} \neq 0 \\ N, & t_{\text{нн}} = 0 \end{cases} = t_{\text{нн}} + N\cdot(t_{\text{нн}} = 0)$.
- в итоге для реализации битовыми операциями на ПИ:
  для $b = \operatorname{hi}(l)$ верно $t \leqslant \frac{(b+1)\cdot N}{2^k} \implies$ запись b и $\begin{cases} l_{\text{новое}} = (l \operatorname{shl} k)\ \&\ (N-1), \\ t_{\text{новое}} = t_{\text{нн}} + N\cdot(t_{\text{нн}} = 0), \text{ где } t_{\text{нн}} = (t \operatorname{shl} k)\ \&\ (N-1); \end{cases}$
- для реализации на отрезках можно обнаружить попадание в одну из 2^k частей как $\operatorname{hi}(l) = \operatorname{hi}(h) = b$;
  $h_{\text{новое}} = t_{\text{новое}} - 1 = t_{\text{нн}} + N\cdot(t_{\text{нн}} = 0) - 1$, где $t_{\text{нн}} = \big((h+1) \operatorname{shl} k\big)\ \&\ (N-1)$.

Кодирование 5111 1115, $N = 512_{10} = 1000_8$

$\widetilde{C} = 5111\,1115$: $\vec{\omega} = (0,3,4)$, $D = 4$; $N = 1000_8$:

i	c_i	k	b_k	$l(8)$	$t(8)$	$h = t - 1(8)$	$\Delta(8)$	$l(10)$	$t(10)$	$\Delta(10)$
0	–	0	–	000	1000	777	1000	0	512	512
1	5			$l + \frac{3\Delta}{4} = 600$	1000	777	200	384	512	128
2	1			600	$l + \frac{3\Delta}{4} = 740$	737	140	384	480	96
3	1			600	$l + \frac{3\Delta}{4} = 710$	707	110	384	456	72
4	1			600	$l + \frac{3\Delta}{4} = 666$	665	66	384	438	54
		1	6	$(l - 600) \cdot 10_8 = 000$	$(t - 600) \cdot 10_8 = 660$	657	660	0	432	432
5	1			000	$l + \frac{3\Delta}{4} = 504$	503	504	0	324	324
6	1			000	$l + \frac{3\Delta}{4} = 363$	362	363	0	243	243
7	1			000	$l + \frac{3\Delta}{4} = 266$	265	266	0	182	182
8	5			$l + \frac{3\Delta}{4} = 210$	266	265	56	136	182	46
		2	2	100	660	657	560	64	432	368

$i = n$, масштабировать не можем, но кода 62 недостаточно (так как $l \neq 000$): к нему необходимо добавить один из байтов $1, 2, 3, 4, 5, 6$ (значения b, для которых $b00$ лежит в $[l, t)$).
Как правило, это $\mathrm{hi}(l) + 1$, здесь 2 $\implies$ код 622.
В общем случае не совпадает ни с концептом, ни с побитовыми: вычисления менее точны $\implies$ длиннее.

Д-1 0008 0300 0100 622, $N = 512_{10} = 1000_8$

k	b_k	$\delta(8)$	$\lambda(8)$	$\tau(8)$	i	c_i	$l(8)$	$\Omega_1 = l + \frac{3\Delta}{4}(8)$	$t(8)$	$\Delta(8)$
0	–	1000	0	1000	0	–	0	600	1000	1000
1	6	100	600	700						
2	2	10	620	630						
3	2	1	622	623						
					1	5	600	740	1000	200
					2	1	600	710	740	140
					3	1	600	666	710	110
					4	1	600	650	666	66
		10	220	230			000	504	660	660
4	0	1	220	221						
					5	1	000	363	504	504
					6	1	000	266	363	363
					7	1	000	210	266	266
					8	5	210		266	56

Ω_j, как и ранее, не масштабируются, а пересчитываются из отмасштабированных l и t ($650 \to 504$, а не 500).

Д-2 0008 0300 0100 622, $N = 512_{10} = 1000_8$

k	b_k	$\delta(8)$	$\lambda(8)$	$\tau(8)$	i	c_i	$l(8)$	$\Omega_1 = l + \frac{3\Delta}{4}(8)$	$t(8)$	$\Delta(8)$
0	–	1000	0	1000	0	–	0	600	1000	1000
1	6	100	600	700						
					1	5	600	740	1000	200
					2	1	600	710	740	140
					3	1	600	666	710	110
2	2	10	620	630						
					4	1	600	650	666	66
		100	200	300			000	504	660	660
					5	1	000	363	504	504
					6	1	000	266	363	363
3	2	10	220	230						
					7	1	000	210	266	266
4	0	1	220	221						
					8	5	210		266	56

Выбор частот и $N = 2^{12} = 4096_{10} = 10000_8$

Плохой случай — сообщение m с $I(m) > 0$ (то есть $p(m) > 0 \Leftrightarrow$ без c: $\nu(c) = 0$), которое не имеет кода. При кодировании на каком-то шаге началу такого m (возможно, всему m) соответствует ПИ нулевой длины; значит, на предыдущем шаге был ПИ длины $\Delta < D$, а масштабирование было невозможно.

1. Такого не будет при $T = 2$ (в частности, $3 : 1$):
   - ПИ $[l, l+2)$ длины 2 можно разделить на два ПИ ненулевой длины: $[l, l+1)$ и $[l+1, l+2)$, причём при $\nu_1 \geqslant \nu_2$ деление именно такое: $\Omega_1 - l = \left\lfloor \frac{\omega_1 \cdot 2}{D} \right\rfloor = \left\lfloor \frac{\nu_1 \cdot 2}{\nu_1+\nu_2} \right\rfloor \geqslant \left\lfloor \frac{\nu_1 \cdot 2}{\nu_1+\nu_1} \right\rfloor = 1$, но $\Omega_1 - l < \left\lfloor \frac{\nu_1 \cdot 2}{\nu_1+0} \right\rfloor = 2$;
   - ПИ $[l, l+1)$ длины 1 гарантированно попадает внутрь ПИ байта и масштабируется до приемлемой длины.
2. Сообщений без кода не будет при частотах $2 : 1 : 1$ — в числителе степени двойки; l и t либо точно совпадают с границами байтов (масштабирование до приемлемой длины), либо достаточно далеко.

При $D = 4$ возможны только $3 : 1$, $2 : 2$ и $2 : 1 : 1$ — сообщений без кода быть не может. При $D = 8$, вероятно, может, но короткий пример подобрать не удалось. Рассмотрим $D = 16$ и максимально нечётные частоты:

j	0	1	2	3	4	5	6	7	8
ξ_j	–	0	1	2	3	4	5	6	7
ν_j	–	7	3	1	1	1	1	1	1
ω_j	0	7	$10_{10} = 12_8$	$11_{10} = 13_8$	$12_{10} = 14_8$	$13_{10} = 15_8$	$14_{10} = 16_8$	$15_{10} = 17_8$	$16_{10} = 20_8 = D$

возьмём $N = 2^{12} = 4096_{10} = 10000_8$.

Короткие сообщения без кода (восьмеричное представление чисел!)

ξ_j	$-$	0	1	2	3	4	5	6	7
ω_j	0	$7_{10} = 7_8$	$10_{10} = 12_8$	$11_{10} = 13_8$	$12_{10} = 14_8$	$13_{10} = 15_8$	$14_{10} = 16_8$	$15_{10} = 17_8$	$16_{10} = 20_8 = D$

i	c_i	k	b_k	$l(8)$	$t(8)$	$\Delta(8)$	
0	$-$	0	$-$	0000	10000	10000	
1	1			$l + \frac{7\Delta}{20_8} = 3400$	$l + \frac{12_8\Delta}{20_8} = 5000$	1400	
2	0			3400	$l + \frac{7\Delta}{20_8} = 4120$	520	далее выбираем c_i так, чтобы $l < 4000 < t$
3	5			$l + \frac{14_8\Delta}{20_8} = 3774$	$l + \frac{15_8\Delta}{20_8} = 4021$	25	
4	0			3774	$l + \frac{7\Delta}{20_8} = 4005$	11	$\Delta < D$: $\vec{\Omega} - l = (0, 3, 5, 6, 6, 7, 7, 10, 11)$

Так как $l < 4000 < t$, а аналога β нет — масштабирование невозможно (невозможно записать ни одного байта).
На следующем шаге уже не все символы имеют код — в $\vec{\Omega} - l$ есть повторы, $\Omega_2 = \Omega_3 = l + 6$ и $\Omega_4 = \Omega_5 = l + 7$:

i	c_i	k	b_k	$l(8)$	$t(8)$	$\Delta(8)$	
4	0			3774	4005	11	$\vec{\Omega} - l = (0, 3, 5, 6, 6, 7, 7, 10, 11)$
5_3	3			$l + \frac{13_8\Delta}{20_8} = 3774 + 6 = 4002$	$l + \frac{14_8\Delta}{20_8} = 3774 + 6 = 4002$	0	нет кода, авост
5_5	5			$l + \frac{13_8\Delta}{20_8} = 3774 + 7 = 4003$	$l + \frac{14_8\Delta}{20_8} = 3774 + 7 = 4003$	0	нет кода, авост

Таким образом, все сообщения, которые имеют частоты $7311\,1111$ и начинаются с 10503 или с 10505 (см. эту страницу) или с 105010, 105012, 105013, 105014, 105015, 105016 (см. следующую страницу) не получат кода.

Выход из наихудшего случая (или его отсутствие)

Символы $0, 1, 2, 4, 6$ и 7 пока закодировать можно:

4	0			3774	4005	11	$\Delta < D$: $\vec{\Omega} - l = (0, 3, 5, 6, 6, 7, 7, 10, 11)$
5_0	0			3774	$3774 + 3 = 3777$	3	$\Delta \ll D$, но можем записать байт
		1	3	7740	7770	30	
		2	7	7400	7700	300	
		3	7	4000	7000	3000	точность потеряна, но из «ямы» выбрались
5_1	1			$3774 + 3 = 3777$	$3774 + 5 = 4001$	2	наихудший случай: не можем масштабировать и $\vec{\Omega} = (0, 0, 1, 1, 1, 1, 1, 1, 2)$: ещё не авост, но на следующем шаге сможем записать только 1 или 7
5_2	2			$3774 + 6 = 4002$	$3774 + 7 = 4003$	1	
		1	4	0020	0030	10	
		2	0	0200	0300	100	
		3	0	2000	3000	1000	
		4	2	0000	10000	10000	точность потеряна, но из «ямы» выбрались

Все сообщения, которые имеют частоты $7311\,1111$ и начинаются с 10503 или с 10505 (см. пред. страницу) или с 105010, 105012, 105013, 105014, 105015, 105016 (см. эту страницу) не получат кода.
Некоторые (возможно, не все) сообщения, которые имеют частоты $7311\,1111$ и начинаются с 10500, с 105011, 105017, 10502 получат код (но существенно длиннее $I(m)$ из-за потери точности).

Что делать?

1. Плохое (не имеющее кода) сообщение m найдётся для любого N, любого $D > 4$ и почти любого набора частот. Необходимо проверять Δ и корректно обрабатывать авост.

2. Можно всё-таки ввести аналог β, чтобы не допускать $\Delta < D$.

3. Сообщение m, которое не имеет кода для одной связки $\vec{\nu} + N$, может иметь код для немного другой $\implies$ если есть несколько реализаций целочисленного АС с побайтовой записью (которым присвоены разные коды алгоритма), отличающихся:
   - подбором N;
   - значением $D_{\text{жел}}$;
   - способом перенормировки частот из хранящихся в файле $\vec{u}$ $(\max(u_j) = 255)$ к требуемым $\vec{\nu}$ $(\sum \nu_j \approx D_{\text{жел}}) \implies$ могут получиться немного другие $\vec{\nu}$ и/или другое итоговое D;
   - порядком сортировки по умолчанию (выше при $\vec{\Omega} - l = (0, 3, 5, 6, 6, 7, 7, 10, 11)$ часть символов с частотой 1 получила код, часть нет);

   то можно подобрать для сообщения m подходящий (без авоста) цел. АС с побайтовой записью.