Кодирование. Сжатие данных. Форматы файлов

Александра Игоревна Кононова

МИЭТ

9 сентября 2021 г. — актуальную версию можно найти на
`https://gitlab.com/illinc/otik`

Кодирование

Кодирование — преобразование дискретной информации

$$x \in X = A_1^+ \quad \rightarrow \quad code(x) \in A_2^+$$

смена алфавита, **сжатие, защита от шума,** шифрование.

x — сообщение, исходный текст, исходная строка, блок;

X — источник сообщений;

A_1 — первичный алфавит (до преобразования);

A_2 — вторичный (алфавит конечного представления).

Обычно A_1 — байты, исходные тексты x — бинарные файлы.

Сжатие

Сжатие (компрессия, упаковка) — кодирование $|code(X)| < |X|$, причём X однозначно и полностью восстанавливается по $code(X)$.

Согласно первой теореме Шеннона $|code(X)| \geqslant I(X)$ (средние!).

Кодирование с $|code(X)| \to I(X)$ и $|code(x)| \to I(x)$ — **оптимальное.**

1. Сжимается не отдельное сообщение x, а источник X.
2. Сжатие возможно только при наличии избыточности в изначальном кодировании X ($|X| > I(X)$).

Если источник X порождает блоки длины N бит с равной вероятностью $\left(p = \frac{1}{2^N}\right)$, он неизбыточен $\to$ не существует такого алгоритма сжатия, который сжимает **любой** блок длины N.

Любой алгоритм сжатия сжимает часто встречающиеся блоки данных за счёт того, что более редкие увеличиваются в размерах.

Последовательность и её сжатие

Источник X генерирует **входную последовательность** $C = c_1 c_2 \ldots c_n \ldots$, $c_i \in A$ — символы пронумерованы (есть «предыдущий» и «последующий»). X неизвестен $\Rightarrow$ строится **модель источника** по входной последовательности.

1. **блок** — конечная входная последовательность (произвольный доступ);
2. **поток** — с неизвестными границами (последовательный доступ).

Алгоритмы сжатия по типу входной последовательности:

1. блочные — статистика всего блока добавляется к сжатому блоку;
2. поточные (адаптивные) — статистика вычисляется только для уже обработанной части потока, «на лету».

Свойства алгоритмов сжатия:

1. степень сжатия $\frac{|X|}{|code(X)|}$ (в среднем по источнику; $\frac{|X|}{|code(X)|} \leqslant \frac{|X|}{|I(X)|}$) и **степень увеличения размера в наихудшем случае;**
2. скорость сжатия и разжатия.

Оптимальное кодирование источника X

Пусть X порождает последовательность из 2^N возможных символов.

1. Равновероятный источник $(I(X) = N)$ — кодирование отдельных символов кодами фиксированной ширины N бит.
2. Стационарный источник без памяти, порождающий символы с разными постоянными вероятностями $(I(X) < N)$ — кодирование отдельных символов кодами переменной ширины: коды Хаффмана, методы семейства арифметического кодирования.
3. Стационарный источник с памятью, порождающий символы с вероятностями, зависящими от контекста $(I(X) < N)$ — кодирование сочетаний символов: словарные методы семейства LZ77 (словарь=текст) и семейства LZ78 (отдельный словарь в виде дерева/таблицы).

Если изначально каждый символ записан кодом фиксированной ширины из N бит $\Rightarrow$ сжатие для 2 и 3.

Модель источника: X неизвестен

Оценка алфавита A_1 и вероятностей источника по сообщению:

$x =$ **«молоко»**

1. A_1 — koi-8, равновероятные символы: $p = \frac{1}{256}$, $I(x) = 6 \cdot \log_2(256) = 48$ (бит
2. A_1 — русский алфавит, равновероятные: $p = \frac{1}{33}$, $I(x) = 6 \cdot \log_2(33) \approx 30{,}3$
3. A_1 — Unicode 12.1, равновероятные: $p = \frac{1}{137\,994}$, $I(x) \approx 6 \cdot 17{,}1 \approx 102{,}4$
4. $A_1 = \{\text{к}, \text{л}, \text{м}, \text{о}\}$, равновероятные: $p = \frac{1}{4}$, $I(x) = 6 \cdot \log_2(4) = 12$
5. $A_1 = \{\text{к}, \text{л}, \text{м}, \text{о}\}$ или koi-8, неравновероятные, стац-й источник без памяти:

   о (3) + к (1) + л (1) + м (1): $p(\text{о}) = \frac{3}{6}$, $p(\text{к}) = p(\text{л}) = p(\text{м}) = \frac{1}{6}$

   $I(x) = -3 \cdot \log_2(\frac{3}{6}) - \log_2(\frac{1}{6}) - \log_2(\frac{1}{6}) - \log_2(\frac{1}{6}) = 3 \cdot \log_2(2) + 3 \cdot \log_2(6) \approx 10{,}8$
6. $A_1 = \{\text{к}, \text{л}, \text{м}, \text{о}\}$ или koi-8, марковский источник первого порядка:

предыдущий	$p(\text{к})$	$p(\text{л})$	$p(\text{м})$	$p(\text{о})$
–	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$
к, л, м	0	0	0	1
о	$\frac{1}{2}$	$\frac{1}{2}$	0	0

$$I(x) = -\log_2(\tfrac{1}{4}) - \log_2(1) - \\ -\log_2(\tfrac{1}{2}) - \log_2(1) - \\ -\log_2(\tfrac{1}{2}) - \log_2(1) = \\ = 2 + 1 + 1 = 4$$

7. $A_1 = \{\text{молоко}, \text{чай}\}$, равновероятные символы: $p = \frac{1}{2}$, $I(x) = 1$

Задачи

Оценить алфавит и построить модели источника: а) равновероятную, б) стационарную без памяти, в) марковскую первого порядка для сообщения x, по модели оценить $I(x)$ и $I(y)$.

1. $x =$ хрюхрюхрюмяухрюмяумяухрюмяумяу (30 символов, 5 «хрю» и 5 «мяу» 0001011011); $y =$ рюх.
2. $x =$ кукукукукарекукукукукарекукукукарекукукукукарекукукукареку (50 символов, 5 «ку» и 5 «кукареку» аналогично); $y =$ кар.

Характеристики кодов

1. Первичный алфавит A_1
2. Оптимальность (неизбыточность)
3. Избыточность (в том числе помехоустойчивость)

} модель источника!

4. Вторичный алфавит A_2 ($A_2 = \{0, 1\}$ — двоичный код)
5. Однозначная декодируемость
6. Разделяемость — код $code(x)$ любой последовательности $x = \overline{a_1 \dots a_n}$ единственным образом разделим на кодовые слова $c_i = code(a_i), a_i \in A_1$:
   1. коды фиксированной ширины — $a, b, c \to$ 00, 01, 10;
   2. коды с разделителем — 1, 11, 111 (0 как разделитель символов);
   3. префиксные коды (дерево) — 0, 10, 11;
   4. прочие — например, $11, 1110111, 11100111$.

Задачи

Построить разделимые коды для $A_1 = \{a, b, c, d, e\}$, $A_2 = \{0, 1\}$:

- фиксированной ширины;
- префиксный;
- «постфиксный»;
- с разделителем;
- не относящийся ни к одной категории.

Формат файла

1. Сигнатура (обычно первые 2-4 байта для общепринятых форматов)
   - быстрое распознавание типа файла (свой/чужой).
2. Метаданные (заголовок)
   - версия формата;
   - исходная длина файла;
   - смещение начала данных, их размер и формат;
   - тип сжатия, параметры для распаковки (обычно чем нестандартнее модель источника, тем объёмнее);
   - тип защиты от помех, параметры для восстановления;
   - зарезервированные поля для выравнивания;
   - контрольная сумма заголовка;
   - контрольная сумма файла и т. д.
3. Данные
   - могут включать вложенные заголовки (контейнеров) с сигнатурами.

Метод кодирования и его реализация

Идея кодирования: $x = A_1^+, x \in X \quad \leftrightarrow \quad code(x) \in A_2^+$

На практике: первичный алфавит — байты, исходный текст — произвольной длины n байт; причём там может встречаться любой символ или их комбинация.

Алгоритм кодирования:

1. собственно алгоритм;
2. **представление данных.**

Программная реализация:

1. дополнение исходного текста при необходимости (обычно нулями) и обрезка декодированного текста до длины n;
2. при сжатии: анализ сжатия/увеличения (запись кода или копии);
3. формирование и чтение заголовка.

$$(n \text{ байт}) \leftrightarrow \begin{cases} \text{модель } X \\ x \in A_1^+ \end{cases} \leftrightarrow \begin{cases} \text{модель } X, \\ \text{алгоритм кодирования}, \\ \text{параметры кодирования}, \\ code(x) \in A_2^+ \end{cases} \leftrightarrow \begin{cases} \text{заголовок}, \\ \text{данные } y \ (m \text{ байт}) \end{cases}$$

Некоторые типовые реализации

Алгоритм работает с блоком длины N байт (после кодирования M байт) — файл дополняется до kN и нарезается на блоки:

Блоки по $N \to M$ байт: (n байт) $\leftrightarrow \begin{cases} n, \\ k \text{ блоков по } N, \\ (k-1)N < n \leqslant kN \end{cases} \leftrightarrow \begin{cases} n, \\ \text{алгоритм}, \\ kM \text{ байт} \end{cases}$

Алгоритм заменяет подстроку $c_i \ldots c_j$ на некоторый кортеж α_i — предваряем кортеж α_i **префиксом** $p = c_k$ (выбираем самый редкий символ): $c_i \ldots c_j \to p\,\alpha_i$, вхождения $p = c_k$ как символа экранируем (заменяем на $p\,\alpha_0$ такое, что никакое $\alpha_i \neq \alpha_0$ и не начинается с α_0):

$$\begin{cases} \ldots(c_i \ldots c_j)\ldots c_k \ldots \\ c_i \text{ — байты} \end{cases} \leftrightarrow \ldots(\alpha_i)\ldots c_k \ldots \leftrightarrow \begin{cases} \text{алгоритм}, \\ \text{значение префикса } p = c_k, \\ \ldots p\,\alpha_i \ldots p\,\alpha_0 \ldots \end{cases}$$

где c_i, p — символы (байты), α_i, α_0 — цепочки символов (байтов).

Простые коды

$x \in A_1 \leftrightarrow y \in A_2$ без сжатия, защиты от помех и шифрования

Простейший базовый код (подразумевается):

0. байт памяти $\leftrightarrow$ беззнаковое целое число $0...B$ (обычно: октет $\leftrightarrow 0...255$)
   натуральный двоичный код $\Longrightarrow$ биты байта имеют номер.

Порядок байтов (если файл читается и записывается на одной платформе — не важен и также подразумевается):

1. N байтов $(\chi_0, \ldots \chi_{N-1})$, $N = 2^s \leftrightarrow$ беззнаковое целое число $0...B^N$;
2. N битов, N произвольное $\leftrightarrow$ беззнаковое целое число $0...2^N$.

Простые коды (фиксированной ширины): беззнаковое целое (код) $\leftrightarrow$?

3. $0...127 \leftrightarrow$ символ из таблицы ASCII;
4. знаковые числа;
5. числа с плавающей или фиксированной запятой;
6. нестандартные цифровые коды (ДДК, Грея, Джонсона) и т. д.

Натуральный двоичный код

Целые неотрицательные числа: от 0 до $2^N - 1$.

Для $N = 4$ — целые 0 до $2^4 - 1 = 16 - 1 = 15$:

0	1	2	3	4	5	6	7
0000	0001	0010	0011	0100	0101	0110	0111
8	9	A (10)	B (11)	C (12)	D (13)	E (14)	F (15)
1000	1001	1010	1011	1100	1101	1110	1111

Циклическая арифметика по модулю 2^N: $2^N \equiv 0$

то есть $(2^N - 1) + 1 = 0$.

Взвешенный: $x = 1 \cdot \texttt{bit[0]} + \ldots + 2^{N-1} \cdot \texttt{bit[N-1]} =$

$= \alpha_0 \cdot \texttt{bit[0]} + \ldots + \alpha_{N-1} \cdot \texttt{bit[N-1]}$.

Код со смещением

Целые числа (возможно — знаковые) в произвольном диапазоне $[a, b]$

— для $x \in [a, b]$ записываем беззнаковое число $y = x - a$ натуральным двоичным кодом.

Дополнительный код

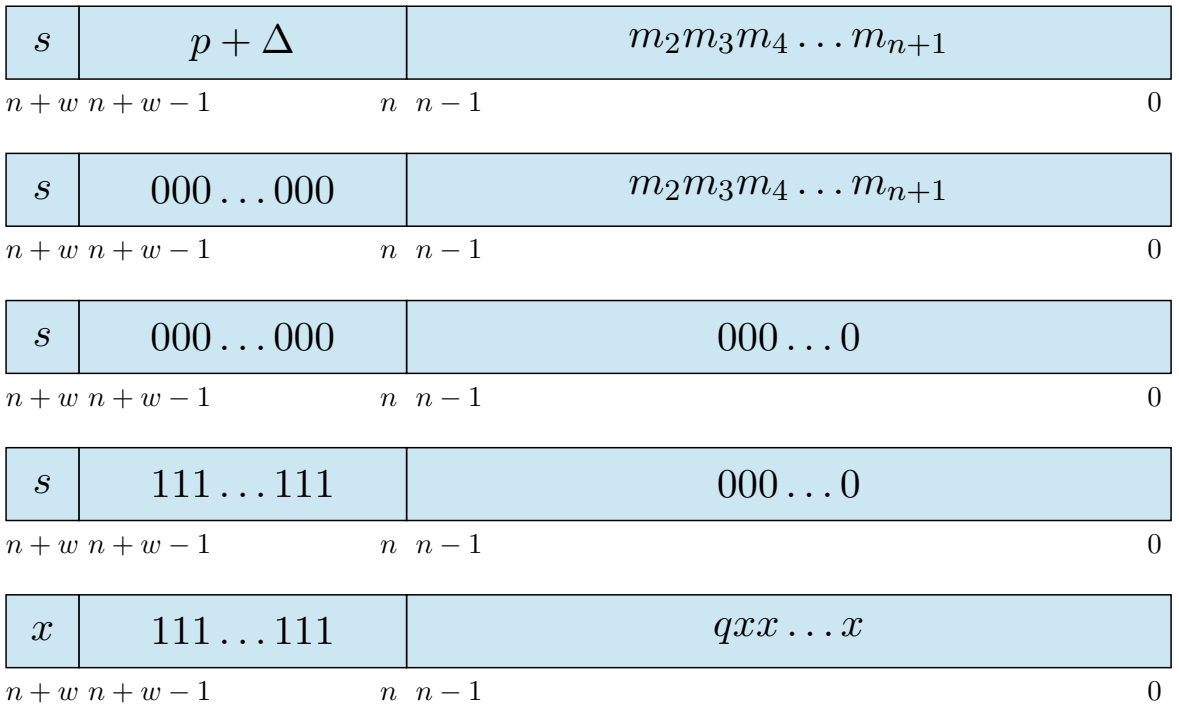
Целые знаковые числа, 0 и ближайшие к 0 положительные представляются как беззнаковые, циклическая арифметика по модулю 2^N: $(-1) = 0 - 1 \equiv 2^N - 1$, $(-2) \equiv 2^N - 2, \ldots$ $(-2^{N-1}) \equiv 2^N - 2^{N-1} = 2^{N-1}$ (считается отрицательным).

Целые числа от -2^{N-1} до $+2^{N-1} - 1$:

0	$+1$	$+2$	$+3$	$+4$	$+5$	$+6$	$+7$	
0000	0001	0010	0011	0100	0101	0110	0111	
	-1	-2	-3	-4	-5	-6	-7	-8
	1111	1110	1101	1100	1011	1010	1001	1000

$(-x) = 0 - x = (-1 - x) + 1 = (\sim x) + 1;$ $\qquad max + 1 = min.$

Числа с плавающей запятой (IEEE 754)



$$(-1)^s \cdot 2^p \cdot \overline{0{,}1 m_2 m_3 m_4 \ldots m_{n+1}}$$
$$p_{min} \leqslant p \leqslant p_{max}$$

$$(-1)^s \cdot 2^{p_{min}} \cdot \overline{0{,}0 m_2 m_3 m_4 \ldots m_{n+1}}$$

$$(-1)^s \cdot 0$$

$$(-1)^s \cdot \infty$$

нечисло (nan)

Нециклическая неассоциативная арифметика: $x + (y + z) \neq (x + y) + z$

$32 = 1 + 8 + 23$ бита — одинарная точность, $float$ $\quad 2^{-126} \leqslant |x| \leqslant 2^{127} \cdot (2 - 2^{-23})$

$64 = 1 + 11 + 52$ бита — двойная, $double$ $\quad 2^{-1022} \leqslant |x| \leqslant 2^{1023} \cdot (2 - 2^{-52})$

Округление: **к ближайшему|чётному,** к ближайшему|∞, к 0, к $+\infty$ (вверх), к $-\infty$ (вниз)

C/C++: `float` $\subseteq$ `double` $\subseteq$ `long double` $\subset \mathbb{R}$, но даже `long double` $\neq \mathbb{R}$

Единичный код

Избыточный невзвешенный
рефлексный (при переходе между кодовыми комбинациями изменяется только один бит)
нециклический ($max + 1 \neq min$) двоичный код

Для N битов — целые 0 до N:

0	1	2	3	4
0000	0001	0011	0111	1111
	0010	0101	1011	
	0100	1001	1101	
	1000	0110	1110	
		1010		
		1100		

Коды Грея и Джонсона

Код Грея — неизбыточный невзвешенный рефлексный циклический двоичный код

0	1	2	3	4	5	6	7
0000	0001	0011	0010	0110	0111	0101	0100
8	9	A	B	C	D	E	F
1100	1101	1111	1110	1010	1011	1001	1000

Код Джонсона — избыточный невзвешенный рефлексный циклический двоичный код

0	1	2	3	4	5	6	7
0000	0001	0011	0111	1111	1110	1100	1000

ASCII и Unicode

ASCII — 128 символов и семибитная (~однобайтовая) кодировка

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Unicode — 137 994 символ (в версии 12.1) и набор кодировок: UTF-8, UTF-16 (UTF-16BE, UTF-16LE) и UTF-32 (UTF-32BE, UTF-32LE)

UTF-8 (до 6 байт) 1 байт 0aaa aaaa 2 байта 110x xxxx 10xx xxxx

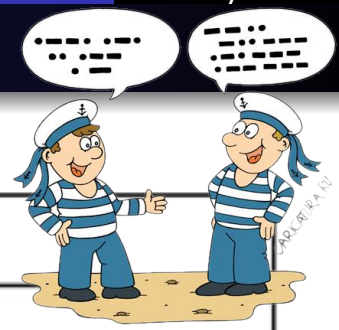
3 байта 1110 xxxx 10xx xxxx 10xx xxxx

4 байта 1111 0xxx 10xx xxxx 10xx xxxx 10xx xxxx

5 байт 1111 10xx 10xx xxxx 10xx xxxx 10xx xxxx 10xx xxxx

6 байт 1111 110x 10xx xxxx 10xx xxxx 10xx xxxx 10xx xxxx 10xx xxxx

Код Морзе (Фридрих Герке)



Азбука Морзе

А •—	К —•—	Ф ••—•	1 •————	. ••••••
Б —•••	Л •—••	Х ••••	2 ••———	, •—•—•—
В •——	М ——	Ц —•—•	3 •••——	; —•—•—•
Г ——•	Н —•	Ч ———•	4 ••••—	: ———•••
Д —••	О ———	Ш ————	5 •••••	? ••——••
Е •	П •——•	Щ ——•—	6 —••••	! ——••——
Ж •••—	Р •—•	Ь,Ъ —••—	7 ——•••	- —••••—
З ——••	С •••	Ы —•——	8 ———••	« •—••—•
И ••	Т —	Э ••—••	9 ————•	(—•——•—
Й •———	У ••—	Ю ••——	0 —————	/ —••—•
		Я •—•—		

Каждой букве или знаку соответствует определённая комбинация кратковременных (точка) и втрое более длинных (тире) импульсов тока, разделённых бестоковым интервалом, равным длительности точки.

Для разделения букв в словах и цифр в многозначных числах применяется тройной бестоковый интервал, заканчивающий каждую комбинацию.

Для разделения слов в тексте служит пятикратный бестоковый интервал.

$$A_2 = \{\cdot, -, \text{межсимвольный интервал}, \text{межсловный интервал}\}$$

Код Бодо (Дональд Мюррей)

Международный телеграфный код №2 (ITA2) +
+ 00000 = МТК-2

		Е	Перевод строки ≡	Пробел	Возврат каретки ≺	Т	А	И	Н	О	С	Р	Х	Д	Л	З	У	Ц	М	Ф	Й	Г	П	Ы	Б	В	Цифры	К	Ж	Ь	Я	Буквы лат.	Буквы рус.
Русский шрифт		**Е**				**Т**	**А**	**И**	**Н**	**О**	**С**	**Р**	**Х**	**Д**	**Л**	**З**	**У**	**Ц**	**М**	**Ф**	**Й**	**Г**	**П**	**Ы**	**Б**	**В**		**К**	**Ж**	**Ь**	**Я**		
Цифры		**3**				**5**	**-**	**8**	**,**	**9**	**'**	**Ч**	**Щ**	кто там?	**)**	**+**	**7**	**:**	**.**	**Э**	**Ю** (зв)	**Ш**	**0**	**5**	**?**	**2**		**(**	**=**	**/**	**1**		
Латинский шрифт		**E**				**T**	**A**	**I**	**N**	**O**	**S**	**R**	**H**	**D**	**L**	**Z**	**U**	**C**	**M**	**F**	**J**	**G**	**P**	**Y**	**B**	**W**		**K**	**V**	**X**	**Q**		
Ведущие отверстия	1	●					●				●			●		●	●			●	●			●	●	●	●	●		●	●	●		
	2		●				●	●				●			●		●	●			●	●	●			●	●	●	●		●	●		
		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	3			●				●	●		●		●				●	●	●	●			●	●				●	●	●	●	●		
	4				●				●	●		●		●				●	●	●	●	●			●		●	●	●	●		●		
	5					●				●			●		●	●			●			●	●	●	●	●	●		●	●	●	●		

фиксированной ширины 5, режимы

Спасибо за внимание!

МИЭТ
www.miet.ru

Александра Игоревна Кононова
illinc@mail.ru
gitlab.com/illinc/raspisanie