- Алфавит источника неизвестен
- Статистика источника неизвестна

- Задача универсального кодирования – получить наименьший объем закодированного сообщения.

- Нумерации
- Двухпроходное побуквенное кодирование
- Словарные методы

8 Простые методы универсального кодирования

Кодирование целых чисел

- Требуется построить процедуру перевода положительных целых чисел в двоичные слова.

- При этом считается, что заранее неизвестен размер алфавита источника информации, а код должен быть однозначно декодируем.

- поэтому подобные методы кодирования относятся к классу универсальных кодов.

- Вероятности чисел убывают с ростом значений элементов и их распределение близко к такому: $P(x+1) \leq P(x)$
- Диапазон значений входных элементов не ограничен или неизвестен.
- При использовании в составе других схем кодирования.

- Простейшие методы
  - унарный код
  - двоичное представление
  - равномерное кодирование (только известного диапазона)

$$x = M \cdot a^{E}$$

- Основная идея кодирования состоит в том, чтобы отдельно кодировать порядок значения числа

  («экспоненту» E)
- и отдельно – значащие цифры числа («мантиссу» M)

- Значащие цифры мантиссы начинаются со старшей ненулевой цифры,
- а порядок числа определяется позицией старшей ненулевой цифры в двоичной записи числа.

методы кодирования целых чисел можно разделить на группы

- Fixed + Variable

  (фиксированная длина экспоненты + переменная длина мантиссы)
- Variable + Variable

  (переменная длина экспоненты + переменная длина мантиссы)

- В кодах класса Fixed + Variable под запись значения порядка числа отводится фиксированное количество бит, а значение порядка числа определяет необходимое количество бит для мантиссы.

- Для кодирования целого числа необходимо произвести с числом две операции:
- определение порядка числа
- выделение бит мантиссы

Пример

- Пусть R=15 – количество бит исходного числа x.
- Отведем E=4 бита под экспоненту
- При записи мантиссы можно сэкономить 1 бит, опуская первую единицу.

	двоичное представление	кодовое слово *E+M*	длина кодового слова
0	000000000000000	0000	4
1	000000000000001	0001	4
2	000000000000010	0010 0	5
3	000000000000011	0010 1	5
4	000000000000100	0011 00	6
5	000000000000101	0011 01	6
6	000000000000110	0011 10	6
7	000000000000111	0011 11	6
8	000000000001000	0100 000	7
9	000000000001001	0100 001	7
10	000000000001010	0100 010	7
…	…	…	..
15	000000000001111	0100 111	7
16	000000000010000	0101 0000	8
17	000000000010001	0101 0001	8
…	…	…	..

Двоичный код класса Variable+Variable для числа x строится следующим образом:

- последовательность нулей (количество нулей равно значению порядка числа),
- единица как признак окончания экспоненты переменной длины,
- мантисса переменной длины (как в кодах Fixed+Variable).

Пример R=11 бит

	двоичное представление	кодовое слово	длина кодового слова
0	00000000000	1	1
1	00000000001	0 1	2
2	00000000010	00 1 0	4
3	00000000011	00 1 1	4
4	00000000100	000 1 00	6
5	00000000101	000 1 01	6
6	00000000110	000 1 10	6
7	00000000111	000 1 11	6
8	00000001000	0000 1 000	8
9	00000001001	0000 1 001	8
10	00000001010	0000 1 010	8
…	…	…	

- Если в рассмотренном выше коде исключить кодовое слово для нуля, то можно уменьшить длины кодовых слов на 1 бит, убрав первый нуль и целые числа будут закодированы *гамма-кодом Элиаса (γ-код Элиаса)*.

Гамма-код Элиаса

число	кодовое слово	длина кодового слова
1	1	1
2 3	0 1 0 0 1 1	3 3
4 5 6 7	00 1 00 00 1 01 00 1 10 00 1 11	5 5 5 5
8 9 10 …	000 1 000 000 1 001 000 1 010 …	7 7 7

- Другим примером кода класса Variable + Variable является *омега-код Элиаса (ω-код Элиаса)*, известный также как рекурсивный код Элиаса.

Чтобы закодировать число:

- Переписать ноль в конец кода.
- Если число, которое требуется закодировать — единица, стоп; если нет, добавить двоичное представление числа в качестве группы в начало кода.
- Повторить предыдущий шаг, с количеством только что записанных цифр(бит) минус один, как с новым числом, которое следует закодировать.

- В этом коде первое значение (кодовое слово для единицы) задается отдельно.
- Другие кодовые слова состоят из последовательности групп с длинами

$$L_1, L_2, ..., L_m$$

- начинающихся с единицы. Конец всей последовательности задается нулевым битом.

- Длина первой группы составляет 2 бита,
- длина каждой следующей группы равна двоичному значению битов предыдущей группы плюс 1. Значение битов последней группы является итоговым значением всей последовательности групп, т.е. первые групп служат лишь для указания длины последней группы

Омега-код Элиаса

	кодовое слово	длина кодового слова
1	0	1
2	10 0	3
3	11 0	3
4	10 100 0	6
5	10 101 0	6
6	10 110 0	6
7	10 111 0	6
8	11 1000 0	7
9	11 1001 0	7
..	…	..
15	11 1111 0	7
16	10 100 10000 0	11
17	10 100 10001 0	11
..	…	..
31	10 100 11111 0	11
32	10 101 100000 0	12

Кодирование длин серий

- Метод кодирования длин серий предложен П. Элиасом (RLE).

- при построении использует коды целых чисел

- Входной поток для кодирования рассматривается как последовательность из нулей и единиц.
- Идея кодирования заключается в том, чтобы кодировать последовательности одинаковых элементов (например, нулей) как целые числа, указывающие количество элементов в этой последовательности.
- Последовательность одинаковых элементов называется *серией*, количество элементов в ней – *длиной серии*.

Пример

- Входную последовательность (общая длина 31бит) можно разбить на серии, а затем закодировать их длины.

000000 1 00000 1 0000000 1 1 00000000 1

6 5 7 0 9

- Используем, например, γ-код Элиаса. Поскольку в коде нет кодового слова для нуля, то будем кодировать длину серии +1, т.е. последовательность

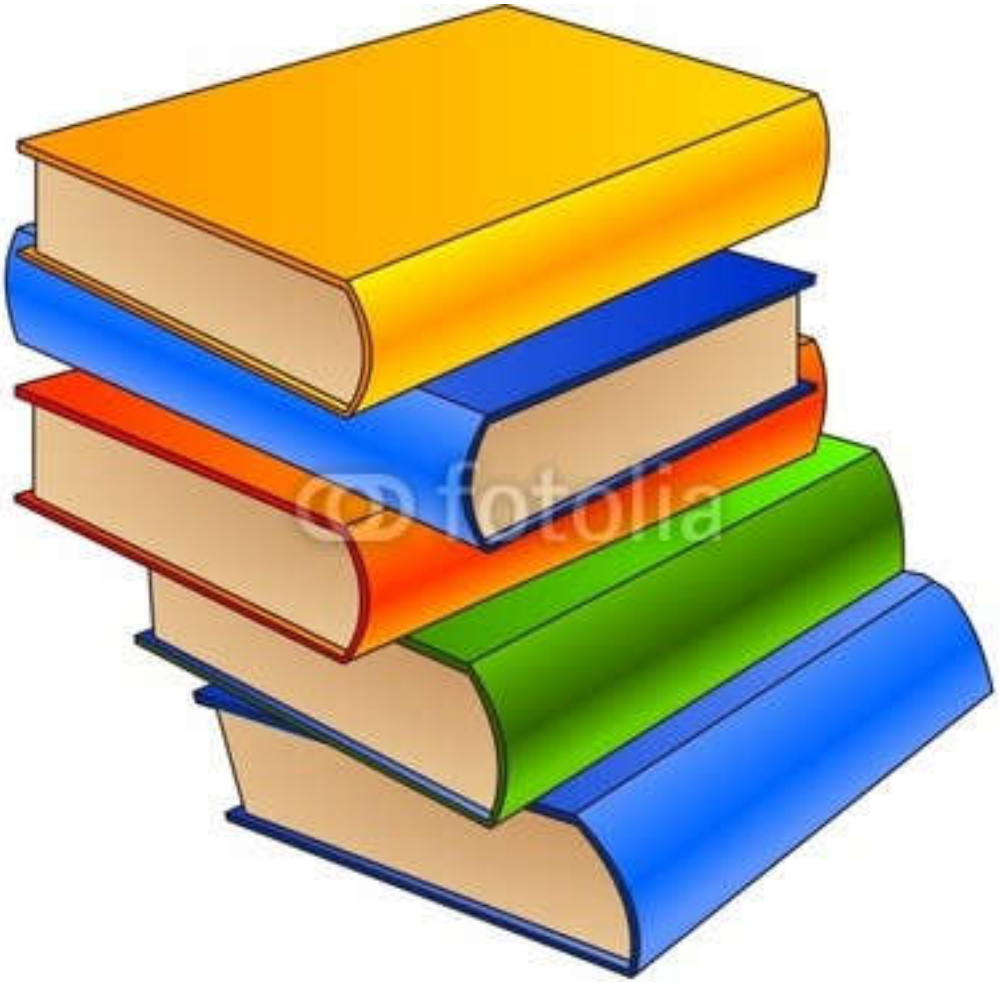
  7 6 8 1 9 $\Rightarrow$

  00111 00110 0001000 1 0001001

- Длина полученной кодовой последовательности равна 25 бит.

- Метод длин серий актуален для кодирования данных, в которых есть длинные последовательности одинаковых бит.

Код «стопка книг»


fotolia

- **До начала кодирования буквы исходного алфавита упорядочены произвольным образом и каждой позиции в стопке присвоено свое кодовое слово,**
- **причем первой позиции стопки соответствует самое короткое кодовое слово, а последней позиции – самое длинное кодовое слово.**

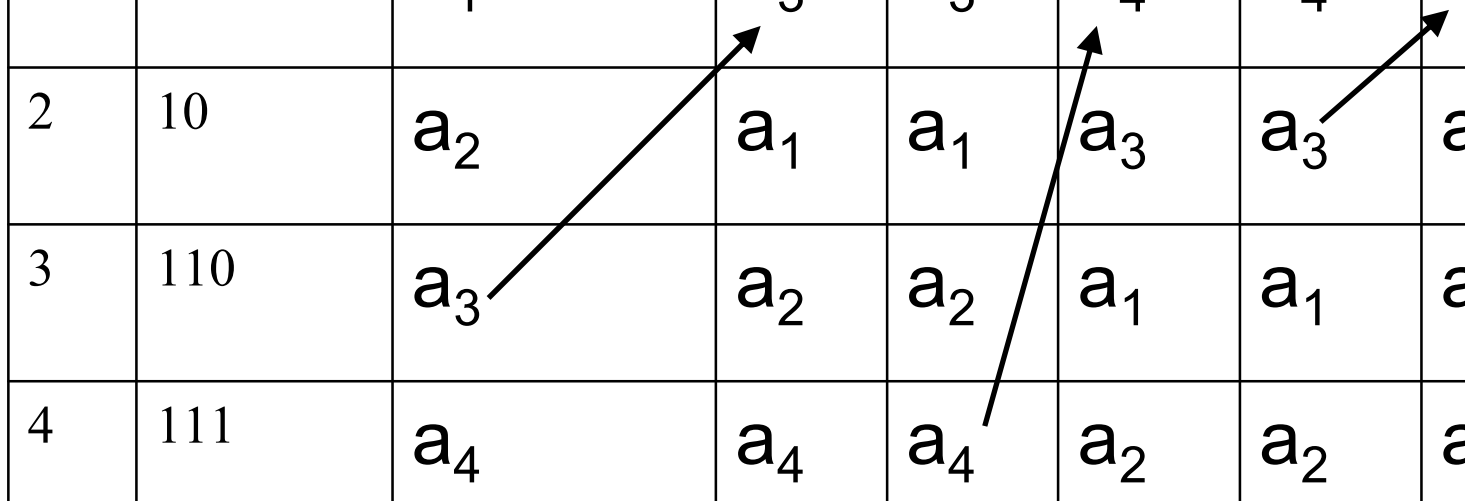
- **Очередной символ кодируется кодовым словом, соответствующим номеру его позиции в стопке, и переставляется на первую позицию в стопке.**

пример кодирования сообщения

- $a_3 a_3 a_4 a_4 a_3 \ldots$
- которое порождает источник информации с алфавитом $A = \{a_1, a_2, a_3, a_4\}$
- Для кодирования номеров позиций символов в стопке можно использовать любой префиксный код, в том числе и ранее рассмотренные коды целых чисел.

$$a_3 a_3 a_4 a_4 a_3 \ldots$$

№	Кодовое слово	Начальная «стопка»	Преобразования «стопки»				
1	0	a_1	a_3	a_3	a_4	a_4	a_3
2	10	a_2	a_1	a_1	a_3	a_3	a_4
3	110	a_3	a_2	a_2	a_1	a_1	a_1
4	111	a_4	a_4	a_4	a_2	a_2	a_2



	a_3	a_3	a_4	a_4	a_3
•	110	0	111	0	10

- **При декодировании используется такая же «стопка книг», находящаяся первоначально в том же состоянии.**
- **Над «стопкой» проводятся такие же преобразования, что и при кодировании. Это гарантирует однозначное восстановление исходной последовательности.**

Интервальный код

- Интервальный код был предложен П. Элиасом в 1987 г. для кодирования источников информации с неизвестной статистикой.

- Для кодирования сообщения используется окно длины W, т.е. при кодировании очередного символа сообщения учитываются W предыдущих символов

$$\ldots \underbrace{x_{i-W} \ldots x_{i-2} x_{i-1}}_{W} x_i x_{i+1} x_{i+2} \cdots$$

- При кодировании символа x_i определяется расстояние (интервал)
- до его предыдущей встречи в окне длины .W

- Обозначим это расстояние $\lambda(x_i)$

- **Если символ x_i содержится в окне, то значение $\lambda(x_i)$ равно номеру позиции символа в окне (позиции в окне нумеруются справа налево).**
- **Если символ x_i отсутствует в окне, то $\lambda(x_i)$ присваивается значение *W*+1**
- **Символ x_i кодируется словом, состоящим из $\lambda(x_i)$ и самого символа x_i**

- **После кодирования очередного символа окно сдвигается вправо на один символ и процесс кодирования продолжается по описанной схеме.**

пример кодирования сообщения

a a b c b b …

которое порождает источник с алфавитом $A = \{a, b, c, d\}$

длина окна W=3

- **Для кодирования чисел $\lambda(x_i)$ можно использовать любой префиксный код, например, такой**

$\lambda(x_i)$	1	2	3	4
код	0	10	110	111

Кодирование интервальным кодом

	Окно W=3			x_i	$\lambda(x_i)$	Код
1				**a**	4	111’a’
2			**a**	**a**	1	0
3		**a**	**a**	**b**	4	111’b’
4	**a**	**a**	**b**	**c**	4	111’c’
5	**a**	**b**	**c**	**b**	2	10
6	**b**	**c**	**b**	**b**	1	0

- **При декодировании используется такое же окно, как и при кодировании.**
- **По принятому кодовому слову можно определить, в какой позиции окна находится данный символ.**
- **Если поступает код 111, то декодер считывает следующий за ним символ.**
- **После этого декодированная буква включается в окно**.

- **Сжатие данных интервальным кодом достигается за счет малых расстояний между встречами более вероятных букв, что позволяет получить более короткие кодовые слова.**