ЧАСТЬ 3

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Способы контроля правильности передачи данных. ***Код с проверкой на четность.*** Контроль целостности информации при передаче от источника к приемнику может осуществляться с использованием корректирующих кодов.

Простейший корректирующий код — *код с проверкой на четность*, который образуется добавлением к группе информационных разрядов одного избыточного, значение которого выбирается таким образом, чтобы сумма единиц в кодовой комбинации, т.е. вес кодовой комбинации, была всегда четна.

Пример 3.1. Рассмотрим код с проверкой на четность, образованный добавлением контрольного разряда к простому двоичному коду:

	Информационные разряды	Контрольный разряд
0	000	0
1	001	1
2	010	1
3	011	0
4	100	1
5	101	0
6	110	0
7	111	1

Такой код обнаруживает все одиночные ошибки и групповые ошибки нечетной кратности, так как четность количества единиц в этом случае будет также нарушаться.

Следует отметить, что при кодировании целесообразно число единиц в кодовой комбинации делать нечетным и осуществлять контроль на нечетность. В этом случае любая комбинация, в том числе и изобра-

жающая ноль, будет иметь хотя бы одну единицу, что дает возможность отличить полное отсутствие информации от передачи нуля.

Коды Хэмминга. *N* — значный код Хэмминга имеет *m* —информационных разрядов и *k* — контрольных. Число контрольных разрядов должно удовлетворять соотношению

$$k \geq \log_2 (n+1),$$

откуда $$m \leq n - \log_2 (n+1).$$

Код Хэмминга строится следующим образом; к имеющимся информационным разрядам кодовой комбинации добавляется вычисленное по вышеприведенной формуле количество контрольных разрядов, которые формируются путем подсчета четности суммы единиц для определенных групп информационных разрядов. При приеме такой кодовой комбинации из полученных информационных и контрольных разрядов путем аналогичных подсчетов четности составляют корректирующее число, которое равно нулю, при отсутствии ошибки, либо указывает номер ошибочного разряда.

Рассмотрим подробнее процесс кодирования. Пусть первый контрольный разряд имеет нечетный порядковый номер, установим его при кодировании таким образом, чтобы сумма единиц всех разрядов с нечетными порядковыми номерами была равна нулю. Такая операция может быть записана в виде следующего соотношения:

$$E_1 = a_1 \oplus a_3 \oplus a_5 \oplus a_7 \ldots = 0,$$

где a_1, a_3, a_5, a_7 — двоичные символы, размещенные в разрядах с номерами 1, 3, 5, 7, ...

Появление единицы во втором разряде (справа) корректирующего числа означает ошибку в тех разрядах кодовой комбинации, порядковые номера которых (2, 3, 6, 7, ...) в двоичном изображении имеют единицу во втором справа разряде. Поэтому вторая операция кодирования, позволяющая найти второй контрольный разряд, имеет вид

$$E_2 = a_2 \oplus a_3 \oplus a_6 \oplus a_7 \ldots = 0.$$

Рассуждая аналогичным образом:

$$E_3 = a_4 \oplus a_5 \oplus a_6 \oplus a_7 \ldots = 0,$$

$$E_4 = a_8 \oplus a_9 \oplus a_{10} \oplus a_{11} \ldots = 0.$$

После приема кодовой комбинации совместно со сформированными контрольными разрядами выполняются те же операции подсчета, что были описаны выше, а полученное число

$$E_k\, E_{k-1} \ldots E_2\, E_1$$

считается корректирующим, причем при $E_k\, E_{k-1} \ldots E_2\, E_1 = 0$ ошибки отсутствуют, а при наличии ошибок неравными нулю оказываются те суммы E_i, в образовании которых участвовал ошибочный разряд. Корректирующее число при этом будет равно порядковому номеру этого разряда.

Выбор места для контрольных разрядов в каждой из кодовых комбинаций определяется таким образом, чтобы контрольные разряды участвовали только в одной операции подсчета четности. Такими позициями являются целые степени двойки: 1, 2, 4, 8, 16, ...

Пример 3.2. Составим шестизначный код Хэмминга (табл. 3.1) для $n = 6$, $k >= \log_2 7$, $k = 3$, $m = n - k = 3$

Таблица 3.1

Цифра	Простой двоичный код	Код Хэмминга 6 5 4 3 2 1
0	000	0 0 0 0 0 0
1	001	0 0 0 1 1 1
2	010	0 1 1 0 0 1
3	011	0 1 1 1 1 0
4	100	1 0 1 0 1 0
5	101	1 0 1 1 0 1
6	110	1 1 0 0 1 1
7	111	1 1 0 1 0 0

Варианты исправления ошибок:

принят код: 111100 *исправлено* 110100 — *ошибка по корректирующему числу в разряде 4;*
принят код: 111010 *исправлено* 101010 — *ошибка по корректирующему числу в разряде 5;*
принят код: 100000 *исправлено* 000000 — *ошибка по корректирующему числу в разряде 6.*

Циклические коды. Циклические коды — разновидность систематических кодов и поэтому обладают всеми их свойствами. Характерной особенностью циклического кода, определяющей его название, является то, что если

n-значная кодовая комбинация $a_0\, a_1\, a_2 \ldots a_{n-1}\, a_n$ принадлежит данному коду, то и комбинация $a_n\, a_0\, a_1\, a_2 \ldots a_{n-1}$, полученная циклической перестановкой знаков, также принадлежит этому коду.

Идея построения циклических кодов базируется на использовании неприводимых многочленов. Неприводимым называется многочлен, который не может быть представлен в виде произведения многочленов низших степеней, т.е. такой многочлен, который делится только на самого себя или на единицу.

Неприводимые многочлены при построении циклических кодов играют роль так называемых образующих полиномов, от вида которых, собственно, и зависят основные характеристики полученного кода: избыточность и корректирующая способность. В таблице 3.2 указаны неприводимые многочлены со степенями $k = 1, 2, 3, 4$.

Таблица 3.2

K	$P(x)$	$P(1,0)$
$k=1$	$x+1$	11
$k=2$	x^2+x+1	111
$k=3$	x^3+x+1	1011
	x^3+x^2+1	1101
$k=4$	x^4+x+1	10011
	x^4+x^3+1	11001
	$x^4+x^3+x^2+x+1$	11111

Основные принципы кодирования в циклическом коде заключаются в следующем. Двоично-кодированное n-разрядное число представляется полиномом $(n-1)$-й степени некоторой переменной x, причем коэффициентами полинома являются двоичные знаки соответствующих разрядов. Запись, чтение и передача кодовых комбинаций в циклическом коде производятся, начиная со старшего разряда. В соответствии с этим правилом в дальнейшем сами числа и соответствующие им полиномы будем записывать так, чтобы старший разряд оказывался справа:

012345

Пример 3.3. Число 110101 (нумерация разрядов согласно выше приведенному правилу, ведется слева направо от 0 до 5) будет представлено полиномом пятой степени: $1+x+x^3+x^5$.

Следует отметить, что циклическая перестановка разрядов в двоичном представлении числа соответствует умножению полинома на x, при котором x^n заменяется единицей и переходит в начало полинома.

Пример 3.4. Выполним умножение полинома, полученного в предыдущем примере, на x. Новый полином $x+x^2+x^4+x^6$ преобразуем, заменив x^6 на 1.

Окончательно получим $1+x+x^2+x^4$, что соответствует числу 111010.

Циклический код n-значного числа, как и всякий систематический код, состоит из m информационных и k контрольных знаков, причем последние занимают k младших разрядов. Поскольку последовательная передача кодовых комбинаций производится, как уже указывалось, начиная со старших разрядов, контрольные знаки передаются в конце кода.

Образование кода выполняется при помощи так называемого порождающего полинома, $P(x)$ степени k, видом которого определяются основные свойства кода — избыточность и корректирующая способность.

Кодовым полиномом $F(x)$ является полином степени, меньшей $(m+k)$, если он делится без остатка на порождающий полином $P(x)$. После передачи сообщения декодирование состоит в выполнении деления полинома $H(x)$, соответствующего принятому коду, на $P(x)$. При отсутствии ошибок $H(x) = F(x)$, и деление выполняется без остатка. Наличие ненулевого остатка указывает на то, что при передаче или хранении произошли искажения информации.

Для получения систематического циклического кода используется следующее соотношение:

$$F(x) = x^k G(x) \oplus R(x),$$

где $G(x)$ — полином, представляющий информационные символы (*информационный полином*); $R(x)$ — остаток от деления $x^k G(x)$ на $P(x)$.

Пример 3.5. Рассмотрим кодирование восьмизначного числа 10110111. Пусть для кодирования задан порождающий полином третий степени $P(x) = 1+x+x^3$.

Делим $x^3 G(x)$ на $P(x)$: $G(x) = 1+x^2+x^3+x^5+x^6+x^7$;

$$x^3 G(x) = x^3+x^5+x^6+x^8+x^9+x^{10};$$

$$\begin{array}{l|l} x^{10}+x^9+x^8+x^6+x^5+x^3 & 1+x+x^3 \\ \underline{x^{10}+x^8+x^7} & \hline x^7+x^6+x^2 \\ \quad x^9+x^7+x^6+x^5+x^3 & \\ \quad \underline{x^9+x^7+x^6} & \\ \qquad\qquad x^5+x^3 & \\ \qquad\qquad \underline{x^5+x^3+x^2} & \\ \qquad\qquad\qquad R(x) = x^2 & \end{array}$$

Используя соотношение для получения систематического циклического кода, находим $F(x)$:

$$F(x) = (x^3 + x^5 + x^6 + x^8 + x^9 + x^{10}) \oplus x^2.$$

Таким образом, окончательно кодовая комбинация, соответствующая $F(x)$, имеет вид

```
                    00010110111
                    001
контрольные разряды ──→ 001 10110111
```

Практически применяемая процедура кодирования еще более проста. Так как нас интересует только остаток, а частное в конечном результате не используется, то можно производить последовательное вычитание по mod 2 делителя из делимого и полученных разностей до тех пор, пока разность не будет иметь более низкую степень, чем делитель. Эта разность и есть остаток. Такой алгоритм может быть реализован аппаратно при помощи k-разрядного сдвигающего регистра, имеющего обратные связи. Очевидно, что полученный этим способом циклический код будет являться систематическим.

Существует и второй способ получения циклического кода, когда очередная кодовая комбинация образуется путем умножения кодовой комбинации $C(x)$ простого n-значного кода на образующий полином $P(x)$. При этом способе образования циклических кодов информационные и контрольные символы в комбинациях циклического кода не отделены друг от друга, что затрудняет процесс декодирования. Поэтому такой способ кодирования применяется реже, чем первый.

Пример 3.6. Рассмотрим кодирование вторым способом, причем при выполнении операций будем использовать непосредственно записи исходных кодовых комбинаций в двоичном виде. Дан порождающий полином вида $P(1,0) = 1101$. Требуется построить циклический код из простого четырехзначного кода вторым способом. Для построения в качестве примера используем исходную комбинацию $C(1,0) = 0011$. Операция умножения этой комбинации на образующий полином запишется следующим образом:

```
  0011
  1101
  ----
  0011
⊕ 0011
  ----
  0011
  ----
0010111 — циклический код для 0011.
```


Аналогичным образом получены остальные кодовые комбинации несистематического циклического кода, приведенные в табл. 3.3.

Таблица 3.3

Простой четырехсимвольный код $C(x)$	Циклический (7,4) — код $C(x)$ $P(x)$, где $P(1,0) = 1101$
0000	0000000
0001	0001101
0010	0011010
0011 *	0010111 *
0100	0110100
0101	0111001
0110	0101110
0111	0100011
1000	1101000
1001	1100101
1010	1110010
1011	1111111
1100	1011100
1101	1010001
1110	1000110
1111	1001011

Проблема обнаружения различного типа ошибок с помощью циклического кода, как уже указывалось, сводится к нахождению нужного порождающего полинома. В нашу задачу не входит рассмотрение этого вопроса, достаточно полно основные принципы подбора порождающих полиномов изложены в учебнике[1].

Эффективное кодирование информации. ***Общие положения.*** Напомним, что *кодирование* — это представление сообщений в форме, удобной для передачи по данному каналу, а *декодирование* — восстановление информации по принятому сигналу. Одним из важнейших вопросов кодирования является повышение его эффективности, т.е. путем устранения избыточности снижение среднего числа символов, требующихся на букву сообщения. Такое кодирование получило название *эффективное кодирование*. Из сказанного выше следует, что эффективное кодирование решает задачу максимального *сжатия информации*.

[1] *Вернер М.* Основы кодирования. М. : Техносфера, 2006.

Учитывая статистические свойства источника информации, можно минимизировать среднее число двоичных символов, требующихся для выражения одной буквы сообщения, что при отсутствии помех позволяет уменьшить время передачи сообщения и его объем.

Эффективное кодирование базируется на *основной теореме Шеннона для канала без помех*, суть которой сводится к следующему: *сообщения, составленные из букв некоторого алфавита, можно закодировать так, что среднее число двоичных символов на букву сколь угодно близко к энтропии источника этих сообщений, но не меньше этой величины.*

Наличие в сообщениях избыточности позволяет рассматривать вопрос о сжатии данных, т.е. о передаче того же количества информации с помощью последовательностей символов меньшей длины — *методы сжатия без потерь* (*обратимые*). Для этого используют специальные алгоритмы сжатия, уменьшающие избыточность. Эффект сжатия оценивают *коэффициентом сжатия*

$$K = n/q,$$

где n — число минимально необходимых символов для передачи сообщения;

q — число символов в сообщении, сжатом данным методом.

Так, при эффективном двоичном кодировании n равно энтропии источника информации.

Наряду с методами сжатия, не уменьшающими количество информации в сообщении, применяют методы сжатия, основанные на потере малосущественной информации, — *методы сжатия с потерями* (*необратимые*). Следует отметить, что применение методов сжатия с потерями неприемлемо для некоторых видов информации, например текстовой или числовой.

Обратимое сжатие всегда приводит к снижению объема выходного потока информативности. Из выходного потока при помощи восстанавливающего алгоритма можно получить исходный поток.

Основные методы обратимого сжатия:

- Шеннона — Фано (Shannon — Fano);
- Хаффмена (Huffman);
- LZW (Lanper — Ziv — Welch);
- арифметическое сжатие.

Преобразование входного потока данных, при котором выходной поток представляет похожий по внешним характеристикам на входной поток объект, однако отличается от него объемом. Используется понятие качества — степени соответствия исходного и результирующего потока.

Основные алгоритмы необратимого сжатия:

- MPEG (Moving Pictues Experts Group);
- JPEG (Joint Photographic Expert Group);
- фрактальное сжатие.

Простейший способ сжатия числовой информации, представленной в коде ASCII, заключается в использовании сокращенного кода с четырьмя битами на символ вместо восьми, так как в ASCII-кодировке цифры, а также символы «.», «,» и « » в качестве первого кодового символа имеют нуль, который может быть отброшен.

Среди широко используемых благодаря своей простоте алгоритмов сжатия наиболее известен *алгоритм RLE* (Run Length Encoding), в котором вместо передачи цепочки из одинаковых символов передаются символ и значение длины цепочки. Этот метод эффективен при передаче растровых изображений, особенно монохромных, где имеется большое число цепочек из единиц и нулей, но малополезен при передаче текста. Алгоритм RLE реализован в формате PCX.

Пример 3.7. Рассмотрим сжатие по алгоритму RLE.
Исходный текст (9 байт): 77 77 77 77 11 11 11 01 FF
В исходной последовательности:
1) выделяется повторяющаяся последовательность байт;
2) заменяется счетчиком повторов (04) и кодирующим байтом (77);
3) 00 — нет повторений;
4) 02 — сколько байт не повторяется;
5) какие байты не повторяются (01 FF).
Результат сжатия (8 байт): 04 77 03 11 00 02 01 FF.

Недостатками алгоритма RLE являются:

- низкая степень сжатия;
- сильная зависимость от количества повторов байт в исходной информационной последовательности.

Достоинство этого алгоритма состоит в простоте его реализации.

Основная сфера использования методов сжатия с потерями — графические, аудио- и видеофайлы, где есть возможность, частично пожертвовав качественными характеристиками информации, добиться значительного уменьшения ее объемов.

Наибольшее распространение среди методов сжатия информации без потерь нашли статистические алгоритмы сжатия, учитывающие вероятность появления отдельных символов в информационном потоке. В первую очередь это алгоритмы Шеннона — Фано и Хаффмена.

Алгоритм Шеннона — Фано. Эффективное кодирование методом Шеннона — Фано базируется на основной теореме Шеннона для канала без помех.

Алгоритм Шеннона — Фано:

- буквы алфавита сообщений выписываются в таблицу в порядке убывания вероятностей;
- буквы алфавита разделяются на две группы так, чтобы суммы вероятностей в каждой группе были по возможности одинаковы;
- всем буквам верхней половины в качестве первого символа приписывается единица, а всем нижним — нули;
- каждая из полученных групп в свою очередь разбивается на две подгруппы с одинаковыми суммарными вероятностями и т.д., процесс повторяется до тех пор, пока в каждой подгруппе останется по одной букве.

Пример 3.8. Рассмотрим алфавит из семи букв, приписав каждой букве вероятность ее появления в сообщении p_i (существуют специальные таблицы вероятности появления букв русского или латинского алфавитов в сообщениях, например, табл. 3.4).

Таблица 3.4

Буква	Вероятность p_i	Процесс получения кода	Код Шеннона
A	1/4	1 1 II	11
E	1/4	1 I 0	10
F	1/8	0 1 1 III	011
C	1/8	0 1 II 0	010
B	1/8	0 0 1 III	001
D	1/16	0 0 0 1 IV	0001
G	1/16	0 0 0 0	0000

$$H(\xi) = \sum_{k=1}^{N} p_k \log_2 1/p_k = 2/4 + 2/4 + 3/8 + 3/8 + 3/8 + 4/16 + 4/16 = 2{,}625.$$

При обычном двоичном кодировании, не учитывая статистических характеристик, для представления каждой буквы потребуется три символа. Наибольший эффект сжатия получается в случае, когда вероятности букв представляют собой целочисленные отрицательные степени двойки. Среднее число символов на букву в этом случае точно равно энтропии.

Среднее число символов на букву

$$L = \sum_{i=1}^{N} p_i n_i,$$

где p_i — вероятность появления i-го символа алфавита; n_i — количество символов в кодовой комбинации i-го символа алфавита.

Для рассмотренного примера 3.8

$$L = 1/4 \times 2 + 1/4 \times 2 + 1/8 \times 3 + 1/8 \times 3 + 1/8 \times 3 + 1/16 \times 4 + 1/16 \times 4 = 2{,}625.$$

Разность величин $(L - H)$ — *избыточность кода*, а величина $(L - H)/L$ — *относительная избыточность*.

Избыточность может трактоваться как мера бесполезно совершаемых альтернативных выборов. Поскольку в практических случаях отдельные знаки никогда не встречаются одинаково часто, то кодирование с постоянной длиной кодовых слов в большинстве случаев избыточно. Несмотря на это, руководствуясь техническими соображениями, такое кодирование применяется достаточно часто.

Алгоритм Хаффмена. Суть алгоритма Хаффмена сводится к следующему:

- буквы алфавита сообщений выписываются в основной столбец таблицы в порядке убывания вероятностей;
- две последние буквы объединяются в одну вспомогательную букву, которой приписывается суммарная вероятность;
- вероятности букв, не участвовавших в объединении, и полученная суммарная вероятность снова располагаются в порядке убывания вероятностей, а две последние объединяются до тех пор, пока не получают единственную вспомогательную букву с вероятностью единица;
- далее для построения кода используется бинарное дерево, в корне которого располагается буква с вероятностью единица, при ветвлении ветви с большей вероятностью присваивается код единица, а с меньшей — код ноль (возможно левой — единица, а правой — ноль).

Пример 3.9. Рассмотрим условный алфавит из восьми букв, каждой из которых приписана соответствующая вероятность ее появления в сообщении (табл. 3.5).

Таблица 3.5

Буква	Вероятность	Вспомогательные столбцы вероятностей	Код Хаффмена
Z1	0,22	0,22 0,22 0,26 0,32 0,42 0,58 1	01
Z2	0,20	0,20 0,20 0,22 0,26 0,32 0,42	00
Z3	0,16	0,16 0,16 0,20 0,22 0,26	110
Z4	0,16	0,16 0,16 0,16 0,20	111
Z5	0,10	0,10 0,16 0,16	100
Z6	0,10	0,10 0,10	1011
Z7	0,04	0,06	10101
Z8	0,02		10100

$$L = 0{,}22 \times 2 + 0{,}20 \times 2 + 0{,}16 \times 3 + 0{,}16 \times 3 + 0{,}10 \times 3 + 0{,}10 \times 4 + 0{,}04 \times$$
$$\times 5 + 0{,}02 \times 5 = 2{,}8;$$
$$H = 2{,}76;$$
$$L - H = 0{,}04.$$

Для сравнения: в коде Шеннона с таким же распределением вероятностей $L - H = 0{,}08$.

Бинарноое дерево изображено на рис. 3.1.

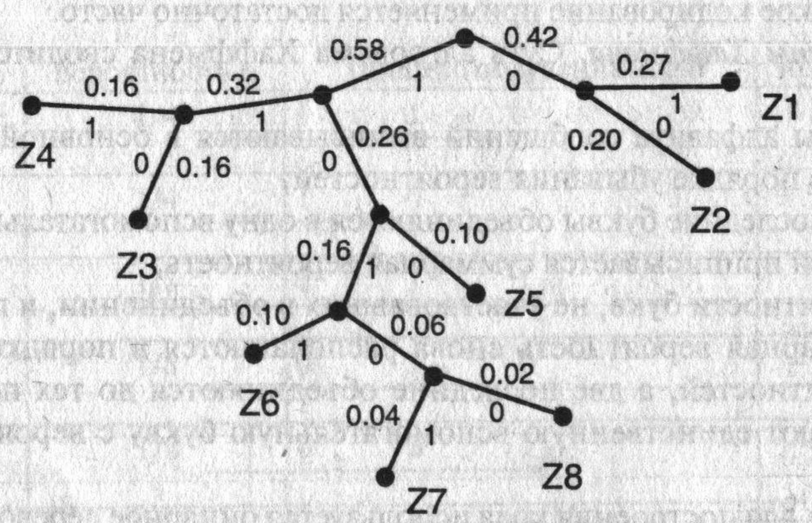

Рис. 3.1. Бинарное дерево

Для построения кодов Шеннона — Фано и Хаффмена можно использовать таблицу вероятности отдельных букв в русском языке (табл. 3.6).

Таблица 3.6

Буква	Вероятность	Буква	Вероятность
_	0,175	Я	0,018
О	0,090	Ы	0,016

Окончание

Буква	Вероятность	Буква	Вероятность
Е	0,072	З	0,016
А	0,062	Ъ,Ь	0,014
И	0,062	Б	0,014
Т	0,053	Г	0,013
Н	0,053	Ч	0,012
С	0,045	Й	0,010
Р	0,040	Х	0,009
В	0,038	Ж	0,007
Л	0,035	Ю	0,006
К	0,028	Ш	0,006
М	0,026	Ц	0,004
Д	0,025	Щ	0,003
П	0,023	Э	0,003
У	0,021	Ф	0,002

Задания для самостоятельной работы[1]

1. Написать процедуру, обеспечивающую эффективное кодирование и подсчет среднего количества информации на знак H; избыточности $(L - H)$ и относительной избыточности получаемого кода $(L - H)/L$.

Для тестирования программы использовать два контрольных варианта, приведенных в табл. 3.7

Таблица 3.7

Буква	Вариант 1		Вариант 2	
	Вероятность	Код Шеннона	Вероятность	Код Хаффмена
Z1	0,22	11	1/2	1
Z2	0,20	101	1/4	01
Z3	0,16	100	1/8	001
Z4	0,16	01	1/16	0001
Z5	0,10	001	1/32	00001
Z6	0,10	0001	1/64	000001
Z7	0,04	00001	1/128	0000001
Z8	0,02	00000 $H = 2{,}76$; $L = 2{,}84$; $L - H = 0{,}08$	1/128	0000000 $H = 1\ 63/64$; $L = 1\ 63/64$; $L - H = 0$

[1] Вариант 1 — код Шеннона; вариант 2 — код Хаффмена.

2. Источник сообщений порождает знаки A, B, C с вероятностями 0,7; 0,2; 0,1.

Написать процедуру, позволяющую:

- определить энтропию источника;
- указать для этих трех знаков оптимальное бинарное кодирование и определить среднюю длину кодовых комбинаций;
- закодировать все пары AA, AB, ... (табл. 3.8);
- построить для этих девяти пар оптимальный бинарный код;
- увеличить блочность кода до трехсимвольных комбинаций и построить оптимальный бинарный код.

Сделать вывод об изменении избыточности кода с увеличением блочности. Как это влияет на эффективность кода?

Таблица 3.8

Буква	Вероятность	Код	Знак	Вероятность	Код пары
A	0,7	0	AA	0,49	1
B	0,2	11	AB	0,14	011
C	0,1	10	BA	0,14	010
			AC	0,07	0011
			CA	0,07	0010
			BB	0,04	0001
			BC	0,02	00001
			CB	0,02	000001
			CC	0,01	000000
$H = 1{,}1571$; $L = 1{,}3$; $L_{знак} = 2{,}33$; $L_{символ} = L_{знак}/2 = 1{,}165$					

LZW-сжатие. Одним из наиболее широко используемых в настоящее время методов сжатия без потерь является алгоритм Лемпела — Зива — Уэлча (Lempel — Ziv — Welch).

Метод сжатия был предложен в 1977 г., усовершенствованный (Terry Welch) вариант был опубликован в 1984 г. В дальнейшем этот метод неоднократно модернизировался. Этот алгоритм используется, в частности, стандартной процедурой UNIX Compress. Алгоритмы группы LZ (LZ77, LZ78, LZSS) лежат в основе почти всех современных программных и аппаратных средств сжатия информации. Архиваторы PKZIP, LHA, Stacker, SuperStor, Dblspace и многие другие используют ту или иную модификацию алгоритма LZ.

Идея сжатия, предложенная Лемпелом и Зивом, заключается в следующем: если в тексте сообщения появляется последовательность из двух ранее уже встречавшихся символов, то эта последовательность

объявляется новым символом, для нее назначается код, который при определенных условиях может быть значительно короче исходной последовательности. В дальнейшем в сжатом сообщении вместо исходной последовательности записывается назначенный код. При декодировании повторяются аналогичные действия и потому становятся известными последовательности символов для каждого кода.

Процесс кодирования.

1. Каждому символу исходного алфавита присваивается определенный код (здесь код — порядковый номер, начиная с нуля).

2. Выбирается первый символ сообщения и заменяется на его код.

3. Выбираются следующие два символа и заменяются своими кодами. Одновременно этой двухсимвольной комбинации присваивается свой код (обычно это порядковый номер, равный числу уже использованных кодов). Так, если алфавит включает восемь символов, имеющих коды от 000 до 111, то первая двухсимвольная комбинация получит код 1000, следующая — код 1001 и т.д.

4. Выбираются из исходного текста очередные 2, 3, ... N-символьные комбинации до тех пор, пока не образуется еще не встречавшаяся комбинация. Тогда этой комбинации присваивается очередной код, и поскольку совокупность из первых $N - 1$ символов уже встречалась, то она имеет свой код, который и записывается вместо этих $N - 1$ символов. Каждый акт введения нового кода назовем шагом кодирования.

5. Процесс продолжается до исчерпания исходного текста.

Процесс декодирования. При декодировании код первого символа, а затем второго и третьего заменяются на символы алфавита. При этом становится известным код комбинации второго и третьего символов. В следующей позиции могут быть только коды уже известных символов и их комбинаций. Процесс декодирования продолжается до исчерпания сжатого текста.

Пример 3.10. Пусть исходный текст представляет собой двоичный код (первая строка табл. 3.9), т.е. исходный алфавит $A = \{1,0\}$. Коды этих символов соответственно также 0 и 1.

Образующийся по методу Лемпела — Зива LZ-код показан во второй строке табл. 3.9. В третьей строке отмечены шаги кодирования, после которых происходит переход на представление кодов A с увеличенным числом разрядов r. Так, на первом шаге вводится код 10 для комбинации 00 и поэтому на следующих двух шагах $r = 2$, после третьего шага $r = 3$, после седьмого шага $r = 4$. В общем случае $r = k$ после шага $2^{k-1} - 1$.

В приведенном примере LZ-код оказался даже длиннее исходного кода, так как обычно короткие тексты с небольшим количеством повторяющихся символов не дают эффекта сжатия. Эффект сжатия проявляется в достаточно длинных текстах и особенно заметен, например, в графических файлах.

Таблица 3.9

Исходный текст	0.00.000. 01. 11. 111.1111. 110. 0000.00000. 1101. 1110.
LZ-код	0.00.100.001.0011.1011.1101. 1010.00110.10010.10001.10110.
r	2 3 4
Вводимые коды	- 10 11 100 101 110 111 1000 1001 1010 1011 1100

Усовершенствованный алгоритм сжатия и распаковки по методу Лемпела — Зива — Уэлча приведен ниже:

Алгоритм компрессии:

```
СТРОКА = очередной символ из входного потока
WHILE входной поток не пуст DO
     СИМВОЛ = очередной символ из входного потока
     IF СТРОКА + СИМВОЛ в таблице строк THEN
        СТРОКА = СТРОКА + СИМВОЛ
     ELSE
        вывести в выходной поток код для СТРОКА
        добавить в таблицу строк СТРОКА + СИМВОЛ
        СТРОКА = СИМВОЛ
     END of IF
END of WHILE
вывести в выходной поток код для СТРОКА
```

Алгоритм декомпрессии:

```
Читать СТАРЫЙ_КОД
вывести СТАРЫЙ_КОД
WHILE входной поток не пуст DO
     читать НОВЫЙ_КОД
     СТРОКА = перевести НОВЫЙ_КОД
     вывести СТРОКУ
     СИМВОЛ = первый символ СТРОКИ
     добавить в таблицу перевода СТАРЫЙ_КОД + СИМВОЛ
     СТАРЫЙ_КОД = НОВЫЙ_КОД
END of WHILE
```

На рисунке 3.2 представлен пример программной реализации алгоритма LZW.

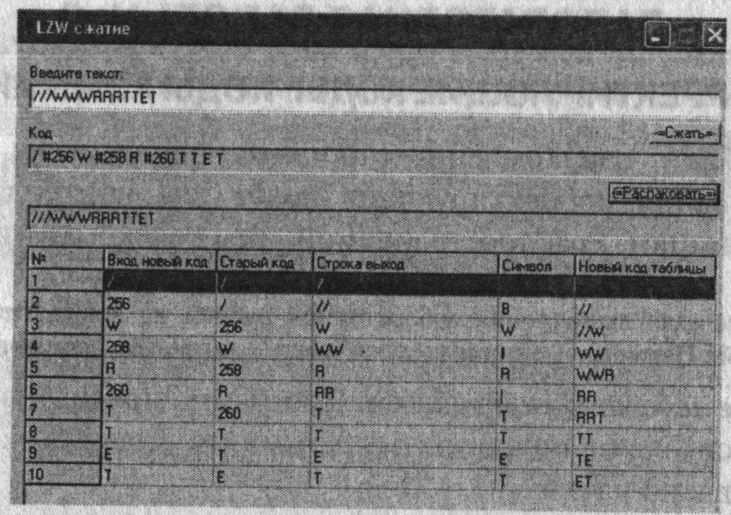

Рис. 3.2. Пример программной реализации алгоритма LZW