Министерство науки и высшего образования Российской Федерации
Московский государственный университет геодезии и картографии

Основы теории кодирования и сжатия сообщений

Москва

2020

УДК 621.391.
ББК 22.19
Ш 14

Рецензенты: профессор И.И. Лонский;

канд. техн. наук Е.А. Бровко (Институт гуманитарных наук, экономики и информационных технологий г. Бургас, Болгария);

канд. техн. наук С.В. Шайтура (Институт гуманитарных наук, экономики и информационных технологий г. Бургас, Болгария)

Составитель: Шавенько Н.К.

Основы теории кодирования и сжатия сообщений: учебно-методическое пособие. – М: МИИГАиК, 2020. – 87 с.

ISBN 978-5-91188-027-9

Содержит теоретические положения разделов курса «Теория информационных процессов и систем»: основы теории кодирования и основы теории сжатия сообщений. Представленные разделы служат теоретическим базисом для специализированных курсов, связанных с автоматической обработкой информации.

Содержание пособия соответствует программе курса «Теория информационных процессов и систем» и «Моделирование информационных процессов и систем» для студентов высших учебных заведений, обучающихся по специальностям направления 09.03.02 и 09.04.02 «Информационные системы и технологии».

Пособие состоит из двух теоретических глав и лабораторного практикума. В первой главе содержатся общие сведения по основам теории кодирования, вторая глава посвящена краткому изложению теории сжатия сообщений. Содержание теоретических глав предполагает, что студенты знакомы с основными понятиями из курса физики и высшей математики.

Кроме этого пособие включает в себя лабораторный практикум, предназначенный для закрепления теоретического материала и представленный в виде подробного описания выполнения предложенной лабораторной работы.

Описание лабораторной работы выполнено из расчета на то, что к моменту её выполнения студентами изучен соответствующий теоретический материал, а также приобретены навыки работы на персональном компьютере в программном продукте «MathCad».

Для студентов высших учебных заведений, обучающихся по направлению 09.03.02 и 09.04.02 «Информационные системы и технологии».

При подготовке пособия использованы теоретические положения, отраженные в научных статьях, в учебной литературе и монографиях.

УДК 621.391.

ББК 22.19

1. ОСНОВЫ ТЕОРИИ КОДИРОВАНИЯ СООБЩЕНИЙ

1.1. Кодирование. Основные понятия

Различные методы кодирования широко используются в практической деятельности человека с незапамятных времён. Например, десятичная позиционная система счисления – это способ кодирования натуральных чисел. Другой способ кодирования натуральных чисел – римские цифры, причем этот метод более наглядный и естественный, действительно, палец – *I*, пятерня – *V*, две пятерни – *X*. Однако при этом способе кодирования труднее выполнять арифметические операции над большими числами, поэтому он был вытеснен способом кодирования, основанном на позиционных системах счисления, в частности, на десятичной системе счисления.

Широко известны способы числового кодирования геометрических объектов и их положения в пространстве: декартовы координаты и полярные координаты, каждый из которых имеет свои особенности.

Из этих примеров можно заключить, что различные способы кодирования обладают присущими только им специфическими особенностями, которые в зависимости от целей кодирования могут быть как достоинством конкретного способа кодирования, так и его недостатком.

В картографии кодирование заключается в замене реальных объектов различными условными знаками. И в этом случае различные способы кодирования отличаются присущими им специфическими особенностями.

До *XX* века методы и средства кодирования играли вспомогательную роль, но с появлением компьютеров ситуация радикально изменилась. Кодирование находит широчайшее применение в информационных технологиях и часто является решающим фактором при решении самых разных задач, таких, как:

– представление данных произвольной природы (чисел, текста, графики, аудио и видео сообщений) в памяти компьютера;

– оптимальная (эффективная) передача данных по каналам связи;

– защита информации (сообщений) от несанкционированного доступа;

– обеспечение помехоустойчивости при передаче данных по каналам связи;

– сжатие информации.

Считается, что термин кодирование произошел от французского слова code – свод законов.

С точки зрения теории информации кодирование – это процесс (или правило) однозначного сопоставления алфавита исходного источника сообщения и некоторой совокупности условных символов, осуществляемый по определенному правилу. А код (кодовый алфавит) – это полная совокупность (множество) различных условных символов (символов кода), которые могут использоваться для кодирования исходного сообщения и которые возможны при данном правиле кодирования. Число же различных кодовых символов, составляющих кодовый алфавит, называют объемом кода или объёмом кодового алфавита. Очевидно, что объём кодового алфавита не может быть меньше объёма алфавита кодируемого исходного источника сообщения. Таким образом, кодирование – это преобразование исходного сообщения в совокупность или последовательность кодовых символов, отображающих сообщение, передаваемое по каналу связи.

Кодирование может быть числовым (цифровым) и нечисловым, в зависимости от вида, в котором представлены кодовые символы: числа в какой–либо системе счисления или иные какие–то объекты или знаки соответственно.

В большинстве случаев кодовые символы представляют собой совокупность или последовательность неких простейших составляющих, например, последовательность цифр в кодовых символах числового кода, которые называются элементами кодового символа. Местоположение или порядковый номер элемента в кодовом символе определяется его позицией.

Число элементов символа кода, используемое для представления одного символа алфавита исходного источника сообщений, называют значностью кода. Если значность кода одинакова для всех символов алфавита исходного сообщения, то код называют равномерным (пример - номера телефонов), в противном случае – неравномерным (пример – азбука Морзе). Число элементов, входящих в кодовый символ, иногда называют длиной кодового символа.

С точки зрения избыточности все коды можно разделить на избыточные коды и неизбыточные. В избыточных кодах число элементов кодовых символов может быть сокращено за счет более эффективного использования оставшихся элементов, а в неизбыточных кодах сокращение числа элементов в кодовых символах невозможно.

Задачи кодирования при наличии помех и при их отсутствии существенно различны. Поэтому различают эффективное (статистическое) кодирование и корректирующее (помехоустойчивое) кодирование. При эффективном кодировании ставится задача добиться представления символов алфавита источника сообщений минимальным числом элементов кодовых символов в среднем на один символ алфавита источника сообщений за

счет уменьшения избыточности кода, что ведет к повышению скорости передачи сообщения.

А при корректирующем (помехоустойчивом) кодировании ставится задача снижения вероятности ошибок в передаче символов исходного алфавита путем обнаружения и исправления ошибок за счет введения дополнительной избыточности кода.

Отдельно стоящей задачей кодирования является защита сообщений от несанкционированного доступа, искажения и уничтожения их. При этом виде кодирования кодирование сообщений осуществляется таким образом, чтобы, даже получив их, злоумышленник не смог бы их раскодировать. Процесс такого вида кодирования сообщений называется шифрованием (или зашифровкой), а процесс декодирования – расшифрованием (или расшифровкой). Само кодированное сообщение называют шифрованным (или просто шифровкой), а применяемый метод кодирования – шифром.

Довольно часто в отдельный класс выделяют методы кодирования, которые позволяют в результате их применения построить закодированные сообщения, имеющие меньшую длину (содержат меньшее количество символов) по сравнению с исходным сообщением без существенных потерь содержащейся в исходном сообщении количества информации. Такие методы кодирования называют методами сжатия или упаковки данных. Качество сжатия определяется коэффициентом сжатия, который обычно измеряется в процентах и который показывает, на сколько процентов кодированное сообщение короче исходного.

При автоматической обработке информации с использованием ЭВМ, как правило, используют числовое (цифровое) кодирование, при этом, естественно, возникает вопрос обоснования используемой системы счисления. Общепризнанным в настоящее время является позиционный принцип построения системы счисления. При этом значение каждого символа (цифры) зависит от его положения - позиции в ряду символов, представляющих число. Единица каждого следующего разряда больше единицы предыдущего разряда в m раз, где m - основание системы счисления. Полное число получают, суммируя значения по разрядам:

$$Q = \sum_{i=1}^{l} a_i \cdot m^{i-1} = a_l m^{l-1} + a_{l-1} m^{l-2} + ... + a_2 m^1 + a_1 m^0,$$

где i - номер разряда данного числа;

l - количество разрядов;

a_i - множитель, принимающий любые целочисленные значения в пределах от

0 до m-1 и показывающий, сколько единиц i - ого разряда содержится в числе.

Основанием системы счисления (m) называется количество различных цифр или каких-либо символов, используемых для изображения чисел в этой системе. Например, для записи чисел в широко применяемой десятичной системе счисления используется десять различных цифр, из чего следует что её основание равно 10. Основанием системы счисления может быть всякое натуральное число — 2, 3, 4, 16 и т.д. То есть, существует бесконечное множество различных позиционных систем.

При числовом кодировании сообщений, выбор основания используемой позиционный системы счисления является весьма важен.

Действительно, при уменьшении основания системы счисления упрощается алфавит элементов символов кода и уменьшается объём их алфавита, но происходит удлинение символов кода. Например, при десятичном кодировании 32 букв русского алфавита используется двухразрядный код, а при двоичном кодировании код содержит ($q = \log_2 32$) 5 разрядов. В частности, код числа, записанного в двоичной системе счисления, в среднем приблизительно в 3,5 раза длиннее десятичного кода.

С другой стороны, чем больше основание системы счисления, тем меньшее число разрядов требуется для представления одного символа кода, а, следовательно, и меньшее время для его передачи, но с ростом основания системы счисления существенно повышаются требования к каналам связи и техническим средствам распознавания элементарных сигналов, соответствующих различным элементам символов кода.

Кроме этого следует учитывать, что во всех системах обработки информации приходится хранить большие информационные массивы, поэтому одним из существенных критериев при выборе основания используемой системы счисления является минимизация количества электронных элементов в устройствах хранения информации, а также их простота и надежность.

При определении необходимого количества элементов технических устройств памяти разумно исходить из практически оправданного предположения, что для фиксации каждого из элементов символов кода требуется количество простейших электронных элементов (например, транзисторов), равное основанию системы счисления m. Тогда для хранения в некотором устройстве символа кода или числа, состоящего из n элементов, потребуется M электронных элементов:

$$M = m \cdot n, \tag{1.1}$$

а количество различных чисел или различных символов кода (т.е. объём кода), которые могут быть записаны в этом устройстве (N), будет равно:

$$N = m^n \quad (1.2)$$

Прологарифмировав выражение (1.2) и выразив из него n, получим:

$$n = \ln N / \ln m.$$

Преобразовав выражение (1.1) к виду

$$M = m \ln N / \ln m, \quad (1.3)$$

можно определить, какое основание системы счисления (m) обеспечит минимальное количество технических элементов M при заданном N. Продифференцировав по m функцию $M = f(m)$ и приравняв её производную к нулю, получим:

$$\frac{dM}{dm} = \ln N \cdot \frac{\ln m - 1}{\ln^2 m} = 0 \quad (1.4)$$

Очевидно, что для любого конечного m

$$\ln N / \ln^2 m \neq 0$$

и, следовательно, $\ln m - 1 = 0$, откуда $m = e \approx 2{,}7$.

Так как основание системы счисления может быть только целым числом, то m выбирают равным 2 или 3. Для примера зададимся максимальной емкостью устройства хранения $N = 10^6$ чисел. Тогда при различных основаниях систем счисления (m) количество элементов (M) в таком устройстве хранения будет, в соответствии с выражением (1.3), следующее (табл. 1.1).

Таблица 1.1

m	2	3	4	5	10	20
M	39,2	38,2	39,2	42,9	60	91,2

Следовательно, если исходить из минимизации количества оборудования, то наиболее выгодными окажутся двоичная, троичная и четверичная системы счисления, которые близки по этому параметру. Но так как техническая реализация устройств, работающих в двоичной системе счисления, значительно проще, то наибольшее распространение при числовом кодировании получили коды на основе системы счисления по основанию 2.

Кроме двоичных кодов, распространение получили восьмеричные коды (с основанием системы счисления равным 8).

Часто используются двоично-десятичные коды, в которых десятичные цифры какого-либо символа представляются двоичными кодами.

Например, если существует необходимость закодировать исходный алфавит, объёмом 64 символов, то это можно сделать используя двоичный ($q = \log_2 64$) 6 разрядный код или восьмеричный ($q = \log_8 64$) 2 разрядный код. При этом символ с номером 13

при двоичном кодировании получает код 001101, при восьмеричном кодировании – 15, а в двоично-десятичном коде - 0001 0011.

1.2. Избыточность кодов.

Одной из основных характеристик различных рода кодов является их избыточность. Понятие избыточности означает, что фактическая энтропия элементов символов кода *(H)* меньше, чем их максимально возможная энтропия (H_{max}), т. е. число символов в сообщении или элементов в символе кода больше, чем это требовалось бы при полном их использовании.

Понятие избыточности легко пояснить следующим примером.

Одна из основных проблем передачи информации это выделение полезного сообщения из зашумлённого помехами сигнала. Один из путей решения этой проблемы заключается в передаче дополнительных символов, то есть во введении в сообщение дополнительной избыточности.

Действительно, в соответствии с теоремой Котельникова непрерывное сообщение (сигнал) можно передать последовательностью его мгновенных отсчетов с интервалами между ними Δt

$$\Delta t = \frac{1}{2 \cdot f_{\max}},$$

где f_{max} – верхняя граничная частота в спектре сигнала.

При наличии помех промежутки между отсчетами (Δt_n) приходится уменьшать, то есть брать $\Delta t_n < \Delta t$

$$\Delta t_n < \frac{1}{2 \cdot f_{\max}} \quad .$$

В этом случае увеличивается число используемых мгновенных отсчетов, то есть увеличивается избыточность сообщения и тем самым повышается его помехозащищенность.

Пусть сообщение из n символов содержит количество информации I. Если сообщение обладает избыточностью, то его (при отсутствии шума) можно передать меньшим числом символов n_0 ($n_0 < n$). При этом количество информации (I_1 и I_{1max}), приходящееся на один символ сообщения, будет связано соотношением:

$$I_1 = I/n < I_{1max} = I/n_0 \qquad (1.5)$$

и, следовательно,

$$n \cdot I_1 = n_0 \cdot I_{1max}. \qquad (1.6)$$

За меру избыточности обычно принимается величина R:

$$R = \frac{n - n_0}{n} = 1 - \frac{n_0}{n} = 1 - \frac{I_1}{I_{1max}} = 1 - \frac{H_1}{H_{max}}, \quad (1.7)$$

где H_1 и H_{max} – энтропия избыточного и неизбыточного сообщения соответственно.

Таким образом, избыточность – это свойство, характеризующее возможность представления тех же сообщений в более «экономной» форме.

При кодировании избыточных сообщений также возникает определенная исходная избыточность кодов, соответствующая избыточности исходных сообщений. Наличие исходной избыточности уменьшает пропускную способность каналов и увеличивает формат сообщений. Вместе с тем в процессе передачи информации избыточность сообщений и кодов является средством, полезным для борьбы с внешними возмущающими воздействиями.

По наличию избыточности коды делятся на избыточные и неизбыточные. Для неизбыточных кодов характерно то, что они позволяют просто определить различные символы исходного сообщения. Переход от неизбыточного кода к избыточному осуществляется путем добавления излишних позиций в кодовых символах, которые можно получить либо путем различных логических операций, выполняемых над основными информационными позициями, либо путем использования алгоритмов, связывающих неизбыточный и избыточный коды. Например, если есть символы сообщения A_1; A_2; A_3; A_4, то их можно закодировать в двоичном неизбыточном равномерном коде:

$$A_1 = 00; \quad A_2 = 01; \quad A_3 = 10; \quad A_4 = 11.$$

Для получения избыточного кода можно ввести еще одну позицию в кодовых символах, значение которой будет определяться как сумма значений предшествующих символов по модулю два

$$A_1 = 000; \quad A_2 = 011; \quad A_3 = 101; \quad A_4 = 110.$$

Особенностью такого кода является то, что он позволяет обнаружить любую единичную ошибку (ошибку в одной из позиций кода), произошедшую в процессе передачи кодовых символов.

1.3. Эффективное кодирование равновероятных символов сообщений

Эффективное кодирование используется в каналах без шума, т. е. в таких каналах, где помехи отсутствуют, либо ими можно пренебречь. Основной задачей кодирования в

таком канале является обеспечение максимальной скорости передачи информации, близкой к пропускной способности канала передачи. При эффективном кодировании устраняют избыточность, что приводит к сокращению средней длины символов кода, а, следовательно, и длины всего сообщения, а значит, позволяет уменьшить время передачи сообщений или объем памяти, необходимой для их хранения.

В случае, когда все символы алфавита исходного кодируемого сообщения независимы и их появление равновероятно, построение оптимального эффективного кода не представляет трудностей. Действительно, если символы сообщения (x_i) равновероятны, взаимно независимы и объем алфавита исходного источника сообщений равен m, то вероятность появления любого i–го символа данного сообщения ($P(x_i)$) будет одинакова, т. е.

$$P(x_i) = \frac{1}{m}\ , \qquad i = 1,.., m, \tag{1.8}$$

а энтропия такого сообщения ($H(x)$) будет равна:

$$H(x) = -\sum_{i=1}^{m} P(x_i) \cdot \log_2 P(x_i) = \log_2 m\,. \tag{1.9}$$

Если для кодирования используется числовой код по основанию k (объем алфавита элементов кодовых символов равен k), то энтропия используемых элементов кодовых символов (H_1) при условии, что элементы символов кода появляются равновероятно и взаимно независимо, определится из соотношения:

$$H_1 = \log_2 k. \tag{1.10}$$

Тогда длина эффективного равномерного кода, т. е. минимальное число элементов кода в кодовом символе ($l_{эфф.}$), может быть найдена из соотношения

$$l_{эфф} \geq \frac{H(x)}{H_1} = \frac{\log_2 m}{\log_2 k} = \frac{\log_2 k^n}{\log_2 k} \cong n\,, \tag{1.11}$$

где $m = k^n$.

Таким образом, эффективное кодирование сообщений, символы которых равновероятны и взаимно независимы, обеспечивается равномерными кодами с длиной кодовых символов, определяемой формулой (1.11). Для построения равномерного кода достаточно пронумеровать символы исходного алфавита в какой-либо системе счисления и записать их символами кода длиной $l_{эфф}$ в избранной системе счисления. Например, при двоичном числовом кодировании 32 букв русского алфавита используется равномерный двоичный код длиной ($l_{эфф} \geq \log_2 32$) 5 двоичных разрядов.

1.4. Эффективное кодирование неравновероятных символов сообщений

Очевидно, что при неравновероятном появлении символов исходного алфавита равномерный код является избыточным, т.к. его энтропия H_0 ($H_0 = \log_k m$), полученная при условии, что все символы его алфавита равновероятны всегда больше энтропии H данного алфавита (полученной с учетом неравномерности появления различных символов алфавита), т.е. информационные возможности данного кода используются не полностью.

Например, для русского алфавита, с учетом неравномерности появления различных букв энтропия $H \approx 4{,}35$, соответствующий ему равномерный двоичный 5-ти разрядный код обладает энтропией H_0 ($H_0 = \log_2 32$) равной 5 бит/символ.

Устранение избыточности достигается применением неравномерных кодов, в которых символы, имеющие наибольшую вероятность появления, кодируются наиболее короткими кодовыми символами, а более длинные кодовые символы присваиваются редко появляющимся символам исходного алфавита.

Если появление символов кодируемого сообщения неравновероятно, в общем виде правило получения оптимального эффективного кода неизвестно. Однако из общих соображений можно представить принципы его построения.

Очевидно, что эффективное кодирование будет оптимальным, если неравномерное распределение вероятностей появления символов алфавита источника сообщений, с помощью определенным образом выбранного кода, переводят в равновероятное распределение вероятностей появления независимых элементов кодовых символов. В этом случае среднее количество информации, приходящееся на один элемент символа кода, будет максимальным и число элементов в символе кода можно сократить. Для определения вида кода, удовлетворяющего этому требованию, можно рассмотреть «функцию стоимости» (цены передачи символов исходного сообщения) Q в виде:

$$Q = \sum_{i=1}^{m} P(x_i) \cdot W_i \ , \qquad (1.12)$$

где $P(x_i)$ – вероятность появления i–го символа из алфавита кодируемого сообщения;

m – объем алфавита;

W_i – стоимость передачи i–го символа алфавита, которая пропорциональна длине кодового символа.

Эффективный код должен минимизировать «функцию стоимости» Q. Если передача всех элементов символов кода имеет одинаковую стоимость, то «стоимость» кодового символа будет пропорциональна длине соответствующего кодового символа. Следовательно, в общем случае (при неравновероятных символах исходного сообщения) код должен быть неравномерным, и построение эффективного кода должно основываться

на следующих принципах:

−длина i – го кодового символа (n_i) должна быть обратно пропорциональна вероятности появления соответствующего символа исходного кодируемого сообщения (x_i);

−начало более длинного кодового символа не должно совпадать с началом более короткого (для возможности разделения кодовых символов без применения разделительных знаков);

−в длинной последовательности кодовых символов элементы символов кода должны быть независимы и равновероятны.

Если i-й (i = 1,2….m) символ исходного алфавита, вероятность появления которого P_i, получает соответствующий кодовый символ длинной q_i, то средняя длина кодовых символов $q_{ср}$ определяется в соответствии с (1.12) выражением:

$$q_{ср} = \sum_{i=1}^{m} P_i \cdot q_i$$

Задачей эффективного кодирования является минимизация средней длины кодовых символов ($q_{ср}$). Считая значность (длину) кодовых символов одинаковой ($q_{ср}$), а появление элементов кодовых символов равновероятным, можно определить максимальную энтропию кодового алфавита как $q_{ср} \cdot \log_2 m$, которая не может быть меньше энтропии исходного алфавита (H), т.е. $q_{ср} \log_2 m \geq H$. Следовательно,

$$q_{ср} \geq \frac{H}{\log_2 m} \ . \qquad (1.13)$$

При использовании двоичного кода ($m = 2$) выражение (1.13) принимает вид:

$$q_{ср} \geq H,$$

или

$$\sum_{i=1}^{m} P_i \cdot q_i = \sum_{i=1}^{m} P_i \cdot \log_2 P_i .$$

Чем ближе численное значение $q_{ср}$ к энтропии исходного алфавита (H), тем более эффективным будет кодирование. В идеальном случае, когда $q_{ср} \approx H$, код называют *эффективным*.

Теоретическое обоснование возможности эффективного кодирования сообщений, передаваемых по каналу, обеспечивает теорема, доказанная К. Шенноном, которая носит название первой теоремы Шеннона (Основная теорема Шеннона о кодировании для каналов без помех). Эта теорема гласит, что если источник сообщений имеет энтропию H

[бит/символ], а канал обладает пропускной способностью C [бит/сек] (пропускная способность характеризует максимально возможное значение скорости передачи информации), то всегда можно найти способ кодирования, который обеспечит передачу символов сообщения по каналу со средней скоростью

$$V_{cp} = \left(\frac{C}{H} - \varepsilon \right) \text{ [символов/сек]},$$

где ε – сколь угодно малая величина.

Обратное утверждение гласит, что передача символов сообщения по каналу со средней скоростью $V_{cp.} > C/H$ невозможна и, следовательно,

$$V_{\max} = \frac{C}{H} \text{ [символов/сек]}. \qquad (1.14)$$

Эта теорема часто приводится в иной формулировке: сообщения источника сообщений с энтропией H всегда можно закодировать последовательностями элементов кодовых символов с объемом алфавита m так, что среднее число элементов символов кода на один символ кодируемого сообщения (q_{cp}) будет сколь угодно близко к величине H / $\log_2 m$, но не менее ее.

Не вдаваясь в доказательство этой теоремы, отметим, что эта теорема показывает возможность наилучшего эффективного кодирования, при котором обеспечивается равновероятное и независимое поступление элементов символов кода, а, следовательно, и максимальное количество переносимой каждым из них информации, равное $log_2 m$ (бит/элемент). К сожалению, теорема не указывает конкретного способа эффективного кодирования, она лишь говорит о том, что при выборе каждого элемента кодового символа необходимо, чтобы он нес максимальное количество информации, а, следовательно, все элементы символов кода должны появляться с равными вероятностями и взаимно независимо.

Исходя из изложенных принципов, разработан ряд алгоритмов эффективного кодирования как для взаимно независимых символов сообщения, так и для взаимозависимых. Суть их состоит в том, что они сокращают среднюю длину кодовых символов путем присвоения кодовых символов минимальной длины символам исходного сообщения, которые встречаются наиболее часто, а более длинные кодовые символы присваиваются редко появляющимся символам исходного алфавита.

Для эффективного кодирования взаимно независимых символов источников сообщений наиболее часто используют коды построенные на основе алгоритма Шеннона – Фено или алгоритма Хаффмена.

1.5. Алгоритмы эффективного кодирования неравновероятных взаимнонезависимых символов источников сообщений

Алгоритм Шеннона – Фено. Суть этого алгоритма, при использовании двоичного кода (объем алфавита элементов символов кода равен 2), заключается в следующем.

Все символы алфавита исходного источника сообщений ранжируют, т. е. располагаются в порядке убывания вероятностей их появления. Затем все символы алфавита делят на две группы, приблизительно равной суммарной вероятности их появления. Все символы первой группы получают «0» в качестве первого элемента кодового символа, а все символы второй группы –«1». Далее группы делятся на подгруппы по тому же правилу примерно равных суммарных вероятностей, и в каждой подгруппе аналогично присваивается вторая позиция кодовых символов. Процесс повторяется до закодирования всех символов алфавита кодируемого исходного источника сообщений, т.е. до тех пор, пока в подмножестве не окажется только по одному символу исходного алфавита.. В кодовый символ, соответствующий последнему символу в ранжированной группе, может быть добавлен, в качестве последнего элемента «0», для того, чтобы конец любого символа кода не являлся началом другого, что позволяет исключить разделительные элементы между символами кода. Табл. 1.2 иллюстрирует процесс построения кода Шеннона – Фено на примере источника сообщений, алфавит которого состоит из восьми символов m_i (i= 1, 2,….8).

Таблица 1.2

Номер символа (i)	Символы алфавита(m_i)	Вероятности появления символов (P_i)	Номера разбиений	Кодовые символы
1	m_1	1/2	*I*	0
2	m_2	1/4	*II*	10
3	m_3	1/8		110
4	m_4	1/16	*III*	1110
5	m_5	1/32	*IV*	11110
6	m_6	1/64	*V*	111110
7	m_7	1/128		1111110
8	m_8	1/256	*VI*	11111110

На рис. 1.1 представлен граф кодирования (кодовое дерево), который показывает, как «расщепляется» ранжированная последовательность символов кодируемого источника

сообщений на группы и отдельные символы и какие кодовые символы присваиваются группам и отдельным символам алфавита исходного источника сообщений на каждом шаге разбиения.

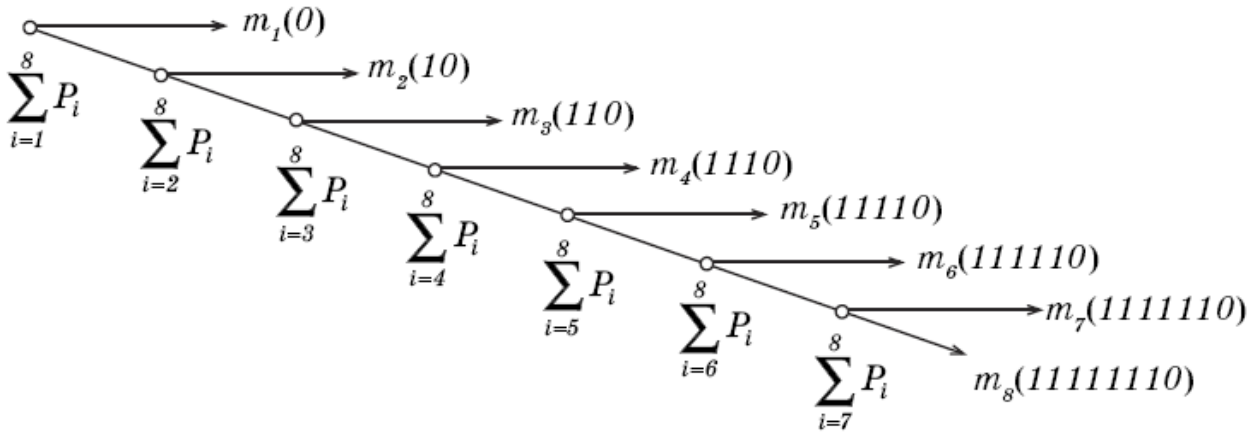


Рис. 1.1. Граф кодирования по алгоритму Шеннона – Фено

Алгоритм Шеннона – Фено применим и при иных числовых основаниях кода ($k > 2$). В этом случае алгоритм получения кода аналогичен рассмотренному примеру, только после ранжирования символов алфавита кодируемого источника сообщений их разбивается на k групп и подгрупп примерно одинаковой суммарной вероятности.

Представляет интерес сравнение эффективного кодирования равномерным кодом и неравномерным кодом по алгоритму Шеннона–Фено.

В качестве примера, можно рассмотреть предложенный выше (табл. 1.2) исходный источник сообщений с объемом алфавита, равным 8, и соответствующими вероятностями появления отдельных символов (P_i). Для кодирования используем двоичный код (k =2).

Энтропия рассматриваемого источника сообщений (H_u) определяется формулой Шеннона:

$$H_u = -\sum_{i=1}^{8} P_i \cdot \log_2 P_i = 1{,}96 \text{ (бит/символ).} \qquad (1.15)$$

Максимально же возможное значение энтропии источника сообщений ($H_{u.max}$), при условии равновероятного и взаимнонезависимого появления символов, находится по формуле Хартли:

$$H_{u.\max} = -\log_2 \frac{1}{m} = \log_2 8 = 3. \qquad (1.16)$$

Избыточность рассматриваемого источника сообщений (R_u) может быть найдена из соотношения:

$$R_u = 1 - \frac{H_u}{H_{u.\max}} = 0.35. \quad (1.17)$$

Используя формулу для эффективного равномерного кода (1.11) при k = 2, получим значность равномерного двоичного кода (n_p)

$$n_p = \log_k m = \log_2 8 = 3, \quad (1.18)$$

и избыточность равномерного кода ($R_{рк}$)

$$R_{рк} = 1 - \frac{H_u}{n_p \cdot \log_2 k} \cong 0{,}35 \quad (1.19)$$

Энтропия элементов символов равномерного кода $(H_{1.p})$, т. е. количество информации, приходящееся на один элемент символа кода, будет равна:

$$H_{1.p} = \frac{H_u}{n_P} \cong 0{,}65 \text{ (бит / элемент символа кода).} \quad (1.20)$$

При использовании эффективного кодирования по алгоритму Шеннона–Фено соответствующие информационные параметры кода будут следующие.

Средняя длина неравномерного кода ($n_н$) определяется выражением

$$n_н = \sum_{i=1}^{m} n_i \cdot p_i = 2, \quad (1.21)$$

где n_i –значность i– го кодового символа, соответствующего символу алфавита m_i.

Избыточность неравномерного кода ($R_{нк}$) определим из соотношения:

$$R_{нк} = 1 - \frac{H_u}{n_H \cdot \log_2 k} \cong 0{,}02. \quad (1.22)$$

Энтропия элементов символов эффективного неравномерного кода $(H_{1н})$ может быть легко найдена:

$$H_{1н} = \frac{H_u}{n_н} = 0.98 \text{ (бит/элемент символа кода).} \quad (1.23)$$

Из сравнения выражений (1.20) и (1.23) видно, что, при использовании эффективного кодирования по алгоритму Шеннона–Фено, энтропия элементов символов неравномерного кода на 50% выше, чем энтропия элементов символов в случае использования равномерного кода. Если предположить, что скорость передачи по каналу элементов символов кода (W) одинакова для равномерного и неравномерного кода, то скорость передачи информации (V), будет определяемая выражением

$$V = H \cdot W, \quad (1.24)$$

где H –энтропия элементов символа кода.

В этом случае скорость передачи информации (V), определяемая выражением (1.24)

также будет на 50% выше при использовании эффективного кодирования по алгоритму Шеннона–Фено по сравнению с кодированием равномерным кодом.

При кодировании по алгоритму Шеннона - Фено некоторая избыточность в кодовых символах, как правило, остается ($q_{ср} > H$).

Эту избыточность часто можно устранить, если перейти к блочному кодированию, при этом существенно повышается эффективность кодирования. Для иллюстрации такого вида кодирования рассмотрим процедуру эффективного кодирования двоичным числовым кодом сообщений, генерируемых источником сообщений с объемом алфавита, равным 2 (m = 2), т. е. с алфавитом, состоящим только из двух символов m_1 и m_2 с вероятностями появления $P(m_1) = 0{,}9$ и $P(m_2) = 0{,}1$ и, следовательно, с энтропией $H = 0{,}47$.

При посимвольном кодировании по алгоритму Шеннона–Фено эффект уменьшения избыточности отсутствует, так как на каждый символ сообщения будет приходиться один символ кода, состоящий из одного элемента.

Произведем теперь кодирование по алгоритму Шеннона–Фено блоков, состоящих из возможных последовательностей двух символов исходного источника сообщений, считая символы взаимнонезависимыми. Результат приведен в табл. 1.3.

Таблица 1.3

Блоки	Вероятности появления блоков.	Номера разбиений	Кодовые комбинации
$m_1 m_1$	0,81		1
$m_1 m_2$	0,09	*I*	01
$m_2 m_1$	0,09	*II*	001
$m_2 m_2$	0,01	*III*	0001

Среднее число элементов символов кода на один символ исходного сообщения, вычисленное по формуле (1.21), равно 0,645, что значительно ниже, чем при посимвольном кодировании.

Кодирование блоков, соответствующих последовательностям из трех символов источника сообщений, дает еще больший эффект. Результат приведен в табл. 1.4.

Таблица 1.4

Блоки	Вероятности появления блоков.	Номера разбиений	Кодовые комбинации
$m_1 m_1 m_1$	0,729	I	1
$m_2 m_1 m_1$	0,081		011
$m_1 m_2 m_1$	0,081	*III*	010

$m_1 m_1 m_2$	0,081	*II*	001
$m_2 m_2 m_1$	0,009	*IV*	00011
$m_2 m_1 m_2$	0,009	*VI*	00010
$m_1 m_2 m_2$	0,009	*V*	00001
$m_2 m_2 m_2$	0,001	*VII*	00000

Среднее число элементов символов кода на один символ источника сообщений равно 0,53.

Теоретический минимум H = 0,47 может быть достигнут при кодировании блоков неограниченной длины.

Следует подчеркнуть, что увеличение эффективности кодирования при укрупнении блоков не связано с учетом все более далеких статистических связей, т.к. рассматриваются исходные алфавиты с независимыми символами. Повышение эффективности определяется лишь тем, что набор вероятностей получившихся при укрупнении блоков можно делить на более близкие по суммарным вероятностям подгруппы.

Алгоритм Шеннона–Фено не всегда приводит к однозначному построению кода, так как при разбиении на подгруппы можно сделать большими по суммарной вероятности как верхние, так и нижние подгруппы. Этого недостатка лишен алгоритм Хаффмена, который гарантирует однозначное построение эффективного кода.

Алгоритм Хаффмена. Дэвид Хаффман (1925—1999) руководил кафедрой Компьютерных наук Массачусетсского технологического института (США). Идея метода, предложенного им в 1952г. заключается в том, что часто встречающимся символам исходного сообщения присваивают короткими кодовые символы, а редко встречающиеся — более длинные.

Суть этого алгоритма, при использовании двоичного кода, состоит в следующем. Все символы исходного алфавита источника сообщений ранжируют, т. е. выписывают в столбец в порядке убывания вероятностей их появления. Два последних символа (с наименьшими вероятностями появления) объединяют в один вспомогательный символ, которому приписывают их суммарную вероятность.

Вероятности символов, не участвовавших в объединении, и вероятность вспомогательного символа вновь ранжируют, т. е. располагают в порядке убывания вероятностей в дополнительном столбце и два последних символа объединяются. Процесс продолжается до тех пор, пока не получим единственный вспомогательный символ с вероятностью, равной единице. Пример кодирования по алгоритму Хаффмена приведен в таб. 1.5.

Таблица 1.5

Символы	Вероятности	Вспомогательные столбцы						
		1	2	3	4	5	6	7
m_1	0,22	0,22	0,22	0,26	0,32	0,42	0,52	1
m_2	0,20	0,20	0,20	0,22	0,26	0,32	0,42	
m_3	0,16	0,16	0,16	0,20	0,22	0,26		
m_4	0,16	0,16	0,16	0,16	0,20			
m_5	0,10	0,10	0,16	0,16				
m_6	0,10	0,10	0,10					
m_7	0,04	0,06						
m_8	0,02							

Для наглядности на рис. 1.2 показан граф кодирования (кодовое дерево), который иллюстрирует ранжирование символов на группы и отдельные символы, причем из точки, соответствующей вероятности 1, направляем две ветви: одной из них (с большей вероятностью) присваиваем символ 1, а второй – символ 0.

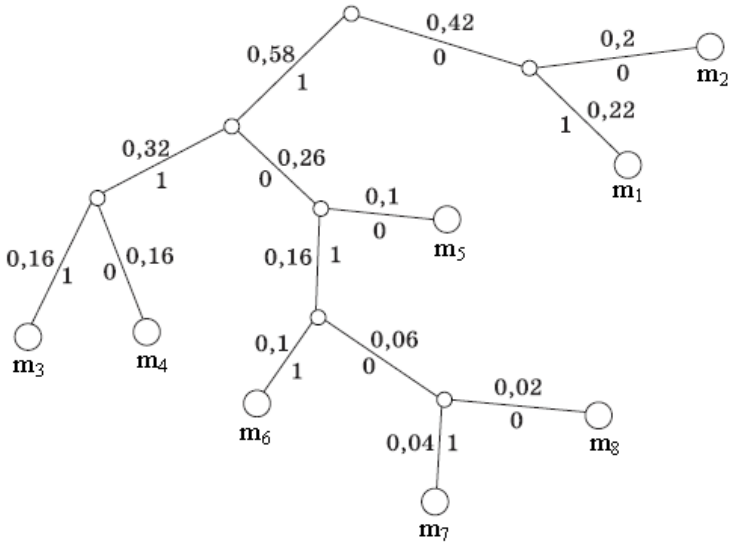


Рис 1.2. Граф кодирования по алгоритму Хаффмена

Такое последовательное ветвление продолжим до тех пор, пока не дойдем до вероятности каждого символа. Двигаясь по кодовому дереву сверху вниз, можно записать для каждого символа источника сообщений соответствующую ему комбинацию элементов (кодовый символ).

Различные символы, генерируемые источником сообщения, и соответствующие им кодовые символы представлены в табл. 1.6.

Таблица 1.6

m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
01	00	111	110	100	1011	10101	10100

Алгоритм Хаффмана по сравнению с алгоритмом Шеннона–Фено оказывается более простым, быстрым и наиболее оптимальным среди алгоритмов, кодирующих каждый символ исходного сообщения по отдельности целым числом элементов в символе кода.

Этот алгоритм можно использовать и при ином числовом основании кода, т.е. при $k>2$, а также использовать блоки, как это рассмотрено в алгоритме Шеннона – Фено.

Эффективность рассмотренных алгоритмов достигается, благодаря присвоению более коротких кодовых комбинаций (кодовых символов) символам источника сообщений, вероятность которых более высока, и более длинных кодовых комбинаций символам источника сообщений с малой вероятностью. Это ведет к различиям в длине кодовых символов и, как следствие, к трудностям при их декодировании. При построении неравномерных кодов необходимо обеспечить возможность их раздельного однозначного декодирования. В равномерных кодах такая проблема не возникает, т.к. при декодировании достаточно последовательность кодовых элементов разделить на группы, каждая из которых состоит из q элементов, в соответствии со значностью равномерного кода. В неравномерных кодах можно использовать разделительный элемент (символ) между кодовыми символами (так поступают, например, при передаче сообщений с помощью известной азбуки Морзе), но при этом существенно снижается эффективность кода, т. к. средняя длина кодового символа фактически увеличивается на один разделительный символ. Целесообразнее обеспечить декодирование без введения дополнительных разделительных элементов символов. Такие коды называются разделимыми.

Проблема кодирования символов исходного сообщения последовательностями кодовых символов переменной длины без использования разделительных символов была решена американскими учеными К.Шенноном и Р.М. Фено. Было предложено условие, достаточное для однозначного декодирования кодовых сообщений с переменной длиной кодовых символов, обычно называемое условием Фено или свойством префиксности. Оно гласит: никакой кодовый символ не должен является началом другого кодового символа. Таким образом, раздельное декодирование кодовых символов переменной длины без использования разделительных символов можно выполнить, если в эффективном коде ни

одна кодовая комбинация не будет совпадать с началом более длинной кодовой комбинации, т.е. необходимо запретить такие кодовые комбинации, начальные части которых уже использованы в качестве самостоятельной кодовых символов. Пример префиксного кода для символов исходного сообщения с объёмом алфавита равном восьми: 00, 10, 010, 110, 0110, 0111, 1110, 1111. Его эффективно использовать, если есть два часто встречающихся символа, два символа, встречающихся со средней частотой, и четыре редко встречающихся символа.

Префиксом или началом называют первые элементы в кодовом символе, а последние элементы – окончанием или постфиксом. Коды, обладающие свойством префиксности, называют префиксными кодами. Легко заметить, что коды, построенные по алгоритмам Шеннона–Фено или Хаффмена, являются префиксными. Это означает, что никакой символ не является началом какого-либо другого символа.

1.6. Алгоритмы эффективного кодирования неравновероятных взаимозависимых символов сообщений

Взаимозависимость символов кодируемых сообщений существенно снижает их энтропию. Принципы построения алгоритмов эффективного кодирования взаимозависимых символов сообщений основаны на устранении взаимной зависимости символов сообщения, которое может быть осуществлено путем укрупнения алфавита исходного источника сообщения. Для этого подлежащие кодированию сообщения последовательно разбиваются на двух –, трех – или n – знаковые сочетания (блоки), вероятности, появления которых известны, а затем эти сочетания кодируются в соответствии с алгоритмами Шеннона–Фено или Хаффмена.

Недостаток этого алгоритма состоит в том, что при его использовании не учитываются связи между символами, входящими в состав соседних сочетаний.

Этот недостаток может быть устранен при кодировании по методу диграмм, триграмм или в общем случае k–грамм (k–граммой называют последовательность из k смежных символов сообщения). При $k = 2$ сочетание смежных знаков называют диграммой, при $k = 3$ –триграммой и т. д. В процессе кодирования по методу k – грамм производят непрерывное последовательное перемещение k – граммы по сообщению с шагом, равным одному символу. Этот процесс (получение 3–х k – грамм) иллюстрирует Рис. 1.3, где x_i – символы сообщения.

Рис. 1.3. Иллюстрация процесса получение k – грамм

Если вероятности появления различных k – грамм известны, то их эффективное кодирование, в частности, может быть выполнено по алгоритмам Шеннона–Фено или Хаффмена. Конкретное значение k выбирается, исходя из степени взаимозависимости между символами сообщения и сложности технической реализации кодирующих и декодирующих устройств.

1.7. Недостатки алгоритмов эффективного кодирования.

При построении различных эффективных кодов необходимо обеспечить возможность их однозначного декодирования. Для этого необходимо получаемым последовательностям элементов кодовых символов присваивать соответствующие им символы исходного алфавита.

В равномерных кодах при выполнении операции декодировании никаких проблем не возникает, т.к. при декодировании последовательности элементов кодовых символов достаточно разделить её на группы, каждая из которых состоит из равного числа элементов, равного значности используемого равномерного кода.

А в неравномерных кодах приходится использовать разделительный символ между группами элементов кодовых символов соответствующих отдельными символами исходного алфавита (так поступают, например, при передаче сообщений с помощью азбуки Морзе). В случае отказа от разделительных символов, следует запретить такие последовательности элементов кодовых символов, начальные части которых уже использованы в качестве самостоятельной их последовательности присвоенной определённому символу исходного алфавита, т.е. использовать префиксные коды.

Основным недостатком при использования алгоритмов эффективного кодирования является специфическое влияние помех на достоверность процесса декодирования, которое проявляется в том, что даже одиночная ошибка, заключающаяся в потере или в появлении какого-либо элемента кодовых символов, может перевести исходную последовательность элементов кодовых символов в другую последовательность элементов кодовых символов, не равную ей по длительности. Это может привести к неверному декодированию ряда последующих последовательностей элементов кодовых символов, к так называемому треку ошибки. Хотя существуют методы, позволяющие свести трек ошибки к минимуму.

Существенным недостатком систем эффективного кодирования на основе неравномерных кодов является сложность их технической реализации, так как эти системы непременно должны включать в себя буферные устройства и устройства накопления. Использование этих устройств вызвано тем, что длина кодовых символов различна, а каналы связи эффективно работают только в том случае, если символы сообщений поступают на них с постоянной скоростью. Кроме этого, при блочном кодировании необходимо «накапливать» символы исходного алфавита, прежде чем присвоить их совокупности какой–либо кодовый символ.

1.8. Помехоустойчивое (корректирующее) кодирование. Общие понятия.

Высокие требования к достоверности и надежности передачи, обработки и хранения информации в системах передачи данных, в вычислительных системах и сетях, в системах управления и различных информационных системах требуют такого кодирования сообщений, которое обеспечивало бы безошибочную ее передачу, а в случае появления ошибок – их обнаружение и исправление.

Ошибка в кодовом символе появляется при его передаче по каналу связи вследствие замены одних элементов символа другими, появления дополнительных элементов или их потери под воздействием помех. Например, если двоичный кодовый символ 0110111 принят как 0100110, то имеет место двукратная ошибка, так как происходит замена (искажение) двух элементов символа кода.

Коды, обладающие способностью обнаружения и исправления ошибок, называют помехоустойчивыми или корректирующими. Следует подчеркнуть, что корректирующее кодирование применяют только в тех случаях, когда возможности других более простых способов повышения помехозащищенности, таких, как многократное повторение или фильтрация, исчерпаны. Это обусловлено усложнением систем передачи информации из-

за необходимости введении в них специфических кодирующих и декодирующих устройств.

Большинство разработанных корректирующих кодов предназначено для исправления взаимно независимых ошибок определенной кратности.

Взаимно независимыми ошибками называют такие искажения в передаваемых символах кода, при которых вероятность появления любой комбинации искаженных элементов символов зависит только от числа искаженных элементов символов кода k и вероятности искажения единичного элемента в кодовом символе p.

При взаимно независимых ошибках вероятность искажения любых k элементов в n-разрядном кодовом символе (p_k)будет определяться выражением:

$$p_k = c_n^k p^k (1-p)^{n-k},$$

где p – вероятность искажения одного элемента в кодовом символе;

k – число искаженных элементов в кодовом символе;

n – число элементов в символе кода;

c_n^k – число сочетаний из n по k.

Если учесть, что $p << 1$, то очевидно, что наиболее вероятны ошибки низшей кратности. Их следует обнаруживать и исправлять в первую очередь.

Построение корректирующих кодов основано на введении в коды избыточности.

Избыточность – это свойство кода, которое, в частности, позволяет исправить ошибки, допущенные при передаче или приёме символов сообщения. Эффективность введённой избыточности может характеризоваться следующими параметрами:

-помехозащищённостью (число или процент) – величина обратная вероятности неисправленных, пропущенных ошибок;

- удлинением кодовых символов или всего сообщения (в процентах).

Очевидно, что для эффективности помехоустойчивого кода его помехозащищённость должна быть достаточно велика, а удлинение кодовых символов - достаточно мало.

Существует два подхода исправления ошибок допущенных при передаче или приёме сообщений.

Первый из них, называемый связью с переспросом, заключается в том, что в случае обнаружения ошибки, принимающая сторона переспрашивает передающую сторону и просит повторить символ сообщения, в котором обнаружена ошибка. В этом случае обнаружение ошибок, допущенных при передаче или приёме символов сообщения, может быть выполнено путём добавления к сообщению или некоторой его части какого-либо

контрольного символа или элемента символа кода, который позволяет проверить правильность приёма сообщения. Примером подобного вида связи может служить телеграф, где при передаче телеграммы всегда указывается число слов в ней (контрольный символ). Пересчитав число слов в телеграмме и сравнив его с контрольной суммой, можно оценить правильность передачи телеграммы. Основной задачей при использовании канала с переспросом является обнаружение ошибки, т. к. «переспрос» сам по себе не представляет трудности.

Другой подход к исправлению ошибок допущенных при передаче или приёме сообщений заключается в создании специальных кодов, которые позволяют не только обнаруживать ошибки, но и самостоятельно их исправлять. Такие коды называются самокорректирующимися или кодами с исправлением ошибок. Примером такого кода может служить обычный человеческий язык, позволяющий исправлять ошибки сообщения без обращения к источнику сообщения, который, возможно, и сделал ошибку, но это возможно лишь в том случае, если приёмник знаком с грамматикой, правилами и исключениями используемого языка.

Простейший метод исправления ошибок, допущенные при передаче или приёме в каналах с переспросом, заключается в дублировании, то есть повторении передаваемого символа. Это позволяет выявить ошибку, если в обоих частях полученных символах (основной и дублирующей) их элементы не совпадают. Ошибка при таком способе кодирования останется незамеченной только в том случае, если будут одинаковым образом искажены соответствующие элементы в основной и дублирующей части передаваемого символа. Очевидно, что такое событие (одинаковым образом искажены соответствующие элементы в основной и дублирующей части передаваемого символа) встречается значительно реже, чем одиночная ошибка. Например, если принять вероятность искажения одного элемента в символе двоичного кода равной P ($P<<1$), то вероятность искажения соответствующих элементов в основной и дублирующей чисти передаваемого символа будет равна P^2 ($P^2<<P$). Поэтому и помехозащищённость такого кода будет существенно выше.

В принципе возможно не только удвоение числа элементов символов сообщений, но их утроение и так далее, в этом случае помехозащищённость кода ещё более возрастает.

Недостатком такого вида повышения помехозащищённости кодов является существенное (в два и более раз) удлинение сообщений. Поэтому простое дублирование символов не является оптимальным способом повышения помехозащищённости кодов.

Подавляющее большинство существующих в настоящее время помехоустойчивых кодов (как самокорректирующихся, так и основанных на связи с переспросом) обладают

требуемыми свойствами, благодаря их алгебраической структуре. Поэтому их называют алгебраическими кодами. Хотя существуют и иные коды, у которых корректирующее действие основано на оценке вероятности искажения каждого символа кода.

Кодовые комбинации (кодовые символы) алгебраических кодов включают в себя две группы элементов кодовых символов: информационные элементы и проверочные элементы. Совокупность информационных элементов кодового символа соответствует символу кодируемого сообщения, а проверочные (избыточные) элементы добавляются к информационным элементам и служат для обнаружения и исправления ошибок.

Все алгебраические коды можно разделить на два больших класса: блочные (блоковые) и непрерывные.

Блочные коды представляют собой совокупность кодовых символов, состоящих из отдельных комбинаций (блоков) элементов символов кода, которые кодируются и декодируются независимо. При этом каждому символу кодируемого исходного сообщения ставится в соответствие блок (комбинация) из n элементов символов кода, куда включаются информационные (k элементов) и ($n - k$) проверочных элементов. Блочный код называют равномерным, если длина (n) для всех блоков одинакова.

В случае блочных кодов процедура кодирования заключается в сопоставлении каждому символу исходного сообщения последовательности из k элементов символов кода, соответствующих этому символу исходного сообщения, которые входят в блоки состоящие из n элементов символов кода ($k \leq n$). В операциях кодирования и декодирования участвуют только указанные k элементов символов кода, и декодированное сообщение не зависит от других элементов символа кода в передаваемом сообщении.

Непрерывные (древовидные) коды представляют собой непрерывную последовательность элементов кодовых символов, причем введение проверочных элементов производится непрерывно, без разделения ее на независимые блоки.

Различают разделимые и неразделимые блочные коды. В разделимых кодах информационные и проверочные элементы символов кода отчетливо разграничены и всегда занимают одни и те же определенные позиции (разряды). Такие коды часто называют (n, k) коды, где n –длина кодового символа, k –число информационных элементов в нем.

При кодировании неразделимыми кодами разделение кодового символа на информационные элементы и проверочные невозможно.

Как блочные коды, так и непрерывные могут быть разделимыми и неразделимыми.

Среди разделимых кодов выделяют систематические (линейные) и несистематические. Систематическими кодами называют коды, в которых проверочные

элементы являются линейными комбинациями информационных. Линейные коды составляют самый большой класс разделимых кодов. Отличительной особенностью этих кодов является то, что значения проверочных элементов в каждом символе определяют в результате проведения линейных операций над определенными информационными элементами. В частности, для двоичных кодов каждый проверочный элемент выбирают таким образом, чтобы его сумма с определенным информационным элементом была равна 0. Элемент на проверочной позиции имеет значение 1, если число единиц информационных разрядов, входящих в данное проверочное равенство, нечетно, и 0, если оно четно. Число проверочных равенств (а, следовательно, и число проверочных элементами) и номера конкретных информационных разрядов, входящих в каждое из равенств, определяется тем, какие и сколько ошибок должен исправлять или обнаруживать данный код. Проверочные элементы могут располагаться на любом месте в кодовой комбинации. При декодировании определяется справедливость проверочных равенств. В случае двоичных кодов такое определение сводится к проверкам на четность числа единиц среди символов, входящих в каждое из равенств (включая проверочные). Совокупность проверок дает информацию о том, имеется ли ошибка, а в случае необходимости и о том, на каких позициях элементы символа кода искажены.

Линейные коды наиболее распространены ещё и потому, что их использование существенно упрощает техническую реализацию кодирующих и декодирующих устройств.

Любой двоичный линейный код является групповым, так как совокупность входящих в него кодовых символов образует группу, поэтому и любой (n, k) код может рассматриваться как групповой код. В этом случае он может быть записан в виде матрицы, включающей k линейно независимых строк по n символов в каждой и, наоборот, любая совокупность k линейно независимых n-разрядных кодовых комбинаций может рассматриваться как образующая матрица некоторого группового кода. Среди всего многообразия таких кодов можно выделить коды, у которых строки образующих матриц связаны дополнительным условием цикличности.

Все строки образующей матрицы такого кода могут быть получены циклическим сдвигом одной комбинации, называемой образующей для данного кода. Коды, удовлетворяющие этому условию, получили название циклических кодов. Сдвиг осуществляется справа налево, причем крайний левый символ каждый раз переносится в конец комбинации. Например, на Рис.1.4 представлена совокупность кодовых комбинаций (символов), получающихся циклическим сдвигом комбинации 001011:

$$G = \begin{bmatrix} 0\,0\,1\,0\,1\,1 \\ 0\,1\,0\,1\,1\,0 \\ 1\,0\,1\,1\,0\,0 \\ 0\,1\,1\,0\,0\,1 \\ 1\,1\,0\,0\,1\,0 \\ 1\,0\,0\,1\,0\,1 \end{bmatrix}.$$

Рис.1.4

Очевидно, что число возможных циклических (n, k)-кодов значительно меньше числа различных групповых (n, k)-кодов.

Частичная классификация помехоустойчивых кодов приведена в табл. 1.7.

Таблица 1.7

1.9. Теоретические основы помехоустойчивого кодирования

Теоретические основы помехоустойчивого кодирования сообщений базируются на основной теореме Шеннона о кодировании для канала с помехами. Эта теорема гласит:

– при любой производительности источника сообщений, меньшей чем пропускная способность канала, существует такой способ кодирования, который позволяет обеспечить передачу всей информации, создаваемой источником сообщений, со сколь угодно малой вероятностью ошибки;

– не существует способа кодирования, позволяющего вести передачу информации со сколь угодно малой вероятностью ошибки, если производительность источника сообщений больше пропускной способности канала.

Анализируя теорему, важно отметить, что она устанавливает теоретический предел процесса передачи информации, создаваемой источником сообщений, со сколь угодно

малой вероятностью ошибки, и показывает, что помехи в канале не накладывают ограничений на точность и достоверность передачи сообщений, а ограничения накладываются лишь на скорость передачи, при которой может быть достигнута сколь угодно высокая достоверность передачи.

Из чего следует, что:

- обеспечение передачи сообщений с весьма малой вероятностью ошибки и достаточно высокой эффективностью возможно при кодировании чрезвычайно длинными последовательностями элементов символов кода;

- при любой конечной скорости передачи информации, вплоть до пропускной способности канала, сколь угодно малая вероятность ошибки достигается лишь при безграничном увеличении длительности символов кода.

Таким образом, безошибочная передача при наличии помех возможна лишь теоретически.

Теорема, к сожалению, не указывает конкретных путей построения кодов, обеспечивающих идеальную надежность и достоверность передачи сообщений, однако она доказывает, что такие пути существуют, и обосновывает принципиальную возможность такого кодирования, позволяя вести практическую разработку конкретных помехоустойчивых кодов.

На практике точность передачи информации и эффективность каналов связи дополнительно ограничивается двумя факторами:

- размером и стоимостью аппаратуры кодирования/декодирования;

- временем задержки передаваемого сообщения.

Для определения конкретных направлений, обеспечивающих повышение надёжности и достоверности передачи сообщений, обратимся к общей информационной модели системы передачи сообщений, которую в общей интегрированной форме можно представить в следующем виде (Рис. 1.5).

Рис. 1.5. Общая информационная модель системы передачи сообщений

Пусть u – исходное сообщение, генерируемое источником сообщения, датчик сообщения преобразует его в сигнал $x = \Psi(u)$, затем этот сигнал проходит по каналу и поступает на приёмник, в котором преобразовывается в сообщение v: $v = \Psi^{-1}(x)$.

Таким образом, процесс передачи сообщений u описывается следующим образом

(рис. 1.6):

– на передающем конце канала символы из пространства исходных сообщений *U* преобразуется в пространство сигналов *X*;

– на приёмном конце канала из пространства сигналов *X* восстанавливается символы пространства полученных сообщений *V*.

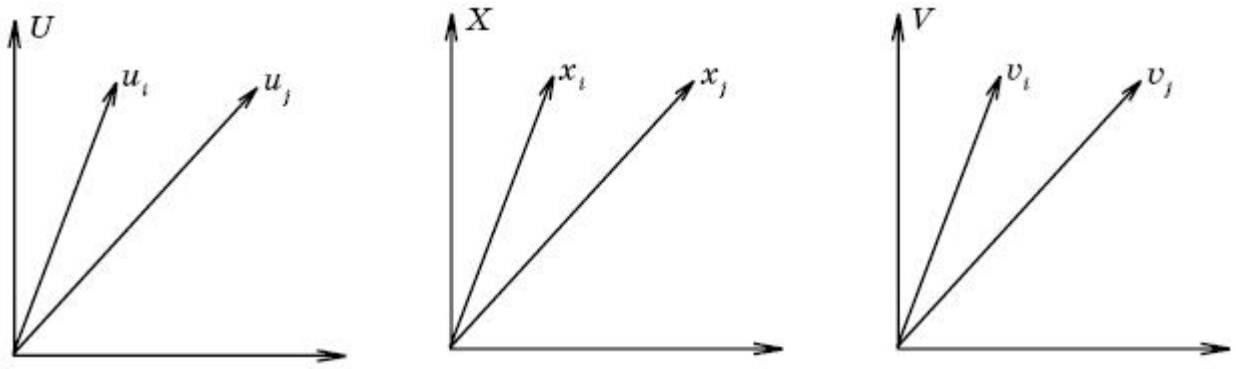


Рис. 1.6. Иллюстрация процесса передачи сообщений

При отсутствии помех выполняется однозначное соответствие между исходным символом сообщения (u_i) и принятым символом сообщения (v_i). Действительно,

$$v_i = \Psi^{-1}(x_i) = \Psi^{-1}[\Psi(u_i)] = u_i, \qquad (1.25)$$

При наличии помех сигнал *X* при прохождении через канал будет искажаться и в общем случае, $v_i \neq u_i$, что приводит к искажению исходного сообщения и вызывает трудности в разделении и опознании исходных символов сообщений. Для безошибочного приёма необходимо идентифицировать принятое сообщение v , то есть определить, какому исходному сообщению u_i оно соответствует.

Выполнить эту идентификацию можно следующим образом. Если собственные шумы в приёмнике отсутствуют, то можно однозначно определить принятый сигнал *x*, соответствующий полученному символу сообщения *v*: $v = \Psi^{-1}(x)$ и определить положение сигнала *x* в пространстве сигналов *X* (рис. 1.7).

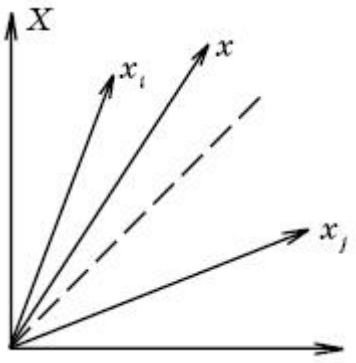


Рис. 1.7. Иллюстрация процесса идентификации символов сообщений.

Вектору x можно присвоить значение ближайшего к нему известного вектора из пространства сигналов X , то есть, $x = x_i$, если

$$|x - x_i| < |x - x_j| \tag{1.26}$$

для любого $i \neq j$..

Приемник, работающий по такому принципу идентификации, называют идеальным приемником.

Ошибка приема идеального приемника (прием сообщения u_i вместо переданного u_j) происходит лишь тогда, когда сигнал x под действием помех переходит границу раздела областей этих сообщений, то есть тогда, когда

$$|x_i - x| > \frac{1}{2}|x_i - x_j|. \tag{1.27}$$

Вероятность появления такого события, а, следовательно, ошибки (P_0), может служить мерой помехоустойчивости или надежности системы связи *(П)*

$$П = \log_2\left(\frac{1}{P_0}\right) = -\log_2 P_0, \tag{1.28}$$

где P_0 – вероятность появления ошибки на выходе приемника.

Очевидно, что с увеличением расстояния между векторами x_i и x_j , помехоустойчивость повышается.

Если величина помехи ξ определяется как $\xi = x_i - x$, то для безошибочного приема сообщения x необходимо выполнение условия

$$|x - x_i| < |x - x_j| \tag{1.29}$$

или

$$|\xi| \le |x_i - \xi - x_j| \le |\Delta x_{ij} - \xi|, \tag{1.30}$$

где $\Delta x_{ij} = \left| x_i - x_j \right|$ -- расстояние между векторами x_i и x_j.

Тогда условие безошибочного приема сообщения запишется в виде

$$\left|\Delta x_{ij}\right| \geq 2\left|\xi\right|.$$

Если помеха *(ξ)* имеет плотность распределения вероятностей *f*(ξ), то вероятность появления ошибки P_0 можно определить из выражения:

$$P_0 = P\left\{\left|\xi\right| \geq \frac{\Delta x_{ij}}{2}\right\} = \int_{-\infty}^{-\frac{\Delta x_{ij}}{2}} f(\xi)dx + \int_{+\frac{\Delta x_{ij}}{2}}^{+\infty} f(\xi)dx. \qquad (1.31)$$

В общем случае, пространство сообщений *U* имеет размерность *n* (например, каждый символ кодового сообщения состоит из *n* элементов). Тогда условие безошибочного приема получим из следующих соображений.

Так как

$$X = \Psi(U), \qquad (1.32)$$

то

$$\Delta X_k = \frac{\partial \Psi}{\partial U_k} \cdot \Delta U_k, \quad \text{а} \quad \Delta X^2 = \sum_{k=1}^{n} \Delta X_k^2,$$

где X_k – *k*–я координата пространства *X* (*k*-й элемент кодового символа).

ΔX -- расстояние между векторами соответствующими расстоянию между символами.

Следовательно,

$$\left|\Delta X\right|^2 = \sum_{k=1}^{n} \Delta X_k^2 = \sum_{k=1}^{n} \left(\frac{\partial \Psi}{\partial u_k} \cdot \Delta U_k\right)^2. \qquad (1.33)$$

Если $\frac{\partial \Psi}{\partial U_k}$ и ΔU_k независимы, то минимальное расстояние между символами d_{min} должно удовлетворять условию

$$(d_{\min})^2 = \left|\Delta X\right|^2 = \sum_{k=1}^{n}\left(\frac{\partial \Psi}{\partial U_k} \cdot \Delta U_k\right)^2 \geq \left(\frac{\partial \Psi}{\partial U}\right)^2_{\max} \cdot \sum_{k=1}^{n}(\Delta U_k) = \dot{\Psi}^2_{\max} \cdot r^2, \qquad (1.34)$$

где $\dot{\psi}_{\max} = \left(\frac{\partial \Psi}{\partial U}\right)_{\max}$;

r – расстояние между двумя ближайшими символами.

Как следует из выше приведённых выводов теории помехоустойчивости, основной практический принцип построения корректирующих (помехозащищенных) кодов

основывается на допущении, что принятый кодовый символ после воздействия на него помехи всегда остается в некоторой близости от кодового символа, однозначно соответствующего передаваемому символу исходного сообщения. Т.е. при взаимно независимых ошибках наиболее вероятен переход в кодовый символ, отличающуюся от исходного в наименьшем числе элементов кодового символа. Это позволяет в дальнейшем отождествить принятый символ с ближайшим к нему символом кодового алфавита путём нахождения минимального расстояния из всех расстояний между принятым символом и всеми символами кодового алфавита.

Исходя из этого, создание специальных помехоустойчивых цифровых кодов, позволяющих не только обнаружить, но и при определенных условиях исправить возникающие ошибки, основано на увеличении минимального расстояния между символами кодового алфавита (d_{min}) путём увеличения избыточности кодов.

Это достигается за счёт введения в процессе кодирования дополнительных (избыточных) элементов символов кода, которые удовлетворяют некоторым дополнительным условиям, и проверка этих условий в процессе дешифрирования дает возможность обнаружить и исправить ошибки.

1.10. Принципы практического построения корректирующих кодов

Принципы практического построения корректирующих кодов удобно продемонстрировать на примере блочных систематических кодов (в частности алгебраических кодов), как наиболее наглядных и широко применяемых.

Для алгебраических (n, k) корректирующих кодов избыточность кода (R_n, R_k) определяется выражениями

$$R_n = \frac{n-k}{n} = 1 - \frac{k}{n} = 1 - \frac{H}{H_{\max}}, \quad R_n = (0-1); \tag{1.35}$$

$$R_k = \frac{n-k}{k} = \frac{n}{k} - 1 = \frac{H_{\max}}{H} - 1, \quad R_k = (0-\infty); \tag{1.36}$$

где n – общее число элементов в кодовом символе;

k – число информационных элементов в кодовом символе или минимально возможное число элементов в кодовом символе.

Величина R_k, с областью изменения от 0 до ∞ предпочтительнее, так как лучше отвечает смыслу понятия избыточности.

Коды, обеспечивающие заданную помехоустойчивость при минимально возможной избыточности, называют оптимальными.

Очевидно, что при оптимальном кодировании, когда лишние элементы в кодах отсутствуют, т. е. когда $n = k$, избыточность равна 0.

Для оценки кодов с точки зрения возможности обнаружения и исправления ошибок используют понятие кодового расстояния (для двоичных кодов -- число кодовых переходов) d_{ij}, которое характеризует степень отличия любых двух кодовых символов, принадлежащих одному коду. Проще говоря, кодовое расстояние выражается числом разрядов, в которых элементы кодовых символов отличаются один от другого.

В общем случае, кодовым расстоянием (d_{ij}) между кодовыми символами k_i и k_j называют суммарный результат сложения по модулю m_k одноименных разрядов кодовых символов k_i и k_j

$$d_{ij} = \sum_{k=1}^{n} k_{ik} \oplus k_{jk}, \tag{1.37}$$

где $k_{ik} \oplus k_{jk} \begin{cases} k_{ik} + k_{jk}, & \text{если} \quad k_{ik} + k_{jk} < m_k \\ k_{ik} + k_{jk} - m_k, & \text{если} \quad k_{ik} + k_{jk} \geq m_k \end{cases}$;

k_{ik} и k_{jk} – k разряд кодовых символов k_i и k_j;

$\oplus$ – символ сложения по модулю m_k;

m_k – основание кода (основание системы счисления числового кода).

При сложении символов двоичных кодов по модулю 2 сумма ($k_i \oplus k_j$) равна 1 тогда и только тогда, когда k_i и k_j; не совпадают). Таким образом, чтобы получить кодовое расстояние между двумя символами двоичного кода, достаточно подсчитать число единиц в сумме этих комбинаций взятой по модулю 2. Например, если даны символы двоичного числового кода k_i = 10111 и k_j = 01010, то кодовое расстояние между этими символами равно 4 (d_{ij}= 4).

Аналогичным образом кодовое расстояние может быть посчитано и для числовых кодов с иными основаниями, отличными от 2.

Кодовое расстояние иногда называют расстоянием Хемминга.

Минимальное расстояние, взятое по всем парам разрешенных кодовых символов, называют минимальным кодовым расстоянием.

Более полное представление о свойствах кода дает матрица кодовых расстояний D, элементы которой d_{ij} ($i,j = 1, 2,\ldots, m$) равны расстояниям между каждой парой из всех m разрешенных кодовых символов.

Для иллюстрации построения корректирующих кодов часто прибегают к геометрической модели. Например, при использовании двоичных кодов любая n-разрядная двоичная кодовая комбинация может быть интерпретирована как вершина n-

мерного единичного куба, т.е. куба с длиной ребра, равной 1, причём общее число их равно 2^n.

Действительно при n = 2 кодовые комбинации (общим числом 4) располагаются в вершинах единичного плоского куба - квадрата (Рис.1.8), а при n = 3 кодовые комбинации располагаются в вершинах единичного трёхмерного куба (Рис 1.9).

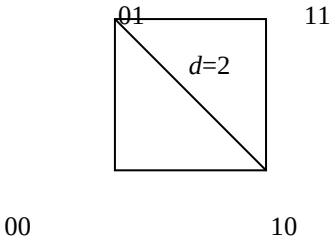


Рис. 1.8.

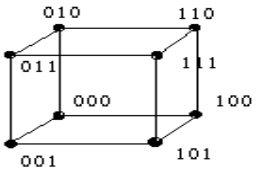


Рис. 1.9.

В общем случае n-мерный единичный куб имеет 2^n вершин, что равно наибольшему возможному числу кодовых комбинаций.

Такая модель дает простую геометрическую интерпретацию кодовому расстоянию между отдельными кодовыми комбинациями. Оно соответствует наименьшему числу ребер единичного куба, которые необходимо пройти, чтобы попасть от одной комбинации к другой.

Таким образом, способность кода обнаруживать и исправлять ошибки обусловлена наличием избыточных элементов в символах кода. Поэтому корректирующие коды всегда являются избыточными.

В качестве примера можно рассмотреть некоторые методы построения блочных корректирующих кодов.

1.11. Некоторые методы построения блочных корректирующих кодов

1. Коды, построенные на основе увеличения кодового расстояния.

Идея обнаружения и исправления ошибок в таких кодах заключается в следующем. Для передачи символов сообщения используют не все N возможных комбинаций элементов символов кода длиной n, а только часть из них N_0, которые называются

разрешенными символами кода. Оставшиеся ΔN комбинаций ($\Delta N = N - N_0$) называют запрещенными. Ошибка обнаруживается тогда, когда приемник получает запрещенную комбинацию элементов символов кода. Всякий код, у которого $\Delta N > 0$, способен обнаруживать ошибки в ΔN случаях из N. Доля обнаруживаемых ошибок (σ) определяется выражением

$$\sigma = \frac{\Delta N}{N} = 1 - \frac{N_0}{N}. \qquad (1.38)$$

Избыточность такого корректирующего кода (R_k) легко определяется из соотношения:

$$R_k = \frac{H(N)}{H(N_0)} - 1 = \frac{n \cdot \log_2 m_k}{\log_2 N_0} - 1, \qquad (1.39)$$

где $H(N_0)$ – энтропия кода с объемом алфавита N_0 ;

$H(N)$ – энтропия кода с объемом алфавита N;

n – число элементов в кодовом символе;

m_k – основание кода.

Удобно рассмотреть построение такого корректирующего кода на примере двоичного кода.

Предположим, что исходное сообщение представляет собой множество символов, составляющих исходный алфавит, каждый из которых состоит из k двоичных элементов. После кодирования этому сообщению соответствует множество кодовых символов состоящих из n двоичных элементов символов кода, причем $n > k$.

Объем алфавита исходного сообщение равен 2^k, а объем кодового алфавита равен 2^n.

Из общего числа 2^n кодовых символов только 2^k кодовых символа соответствуют символам исходного сообщения. Их называют *разрешенными кодовыми символами*.

Остальные (2^n - 2^k) возможных кодовых символов при кодировании не используются. Их называют запрещенными кодовыми символами.

Искажения исходного сообщения в процессе передачи или кодирования сводятся к тому, что некоторые из символов исходного сообщения кодируются кодовыми символами не соответствующим им.

Так как каждый из 2^k разрешенных кодовых символов в результате действия помех может трансформироваться в любой другой из возможных, то всегда имеется $2^k \cdot 2^n = 2^{k+n}$ возможных случаев кодирования. В это число входят:

1) 2^k случаев безошибочного кодирования;

2) $2^k(2^k-1)$ случаев перехода в другие разрешенные кодовые символы, что соответствует числу необнаружеваемых ошибок;

3) $2^k(2^n-2^k)$ случаев перехода в неразрешенные кодовые символы, которые могут быть обнаружены.

Следовательно, часть обнаруживаемых ошибочных кодовых символов от общего числа возможных случаев передачи составляет

$$\frac{2^k(2^n-2^k)}{2^k\cdot 2^n}=1-\frac{2^k}{2^n}.$$

Исправление обнаруженных ошибочных кодовых символов в корректирующих кодах заключается в том, что, обнаружив ошибку, определяют кодовое растояние от полученного запрещенного кодового символа k_i до всех разрешенных символов кода k_j (j=1, 2, ..., N_0) и, в качестве переданного символа кода, считается тот из разрешенных символов кода, до которого расстояние минимально. Таким образом, исправление обнаруженных ошибок корректирующими кодами производится с помощью трех последовательных операций:

– определения кодового расстояния между принятым кодовым символом и всеми разрешенными кодовыми символами;

– отыскания минимального кодового расстояния;

– присвоения принятому кодовому символу значения ближайшего к нему (по кодовому расстоянию) разрешенного кодового символа.

Декодирование после приема какого-либо кодового символа производится таким образом, что принятый кодовый символ отождествляется с тем разрешенным символом, который находится от него на наименьшем кодовом расстоянии. Такое декодирование называется декодированием по методу максимального правдоподобия.

Исправление обнаруженных ошибочных кодовых символов можно рассматривать как некое правило разбиения всего множества запрещенных кодовых символов на 2^k подмножеств M_i (j=1…2^k), каждому из которых ставится в соответствие один из разрешенных кодовых символов. Способ разбиения на подмножества зависит от того, какие ошибки должны исправляться конкретным кодом.

При получении запрещенного кодового символа, принадлежащего подмножеству M_i, принимают решение, что передавался запрещенный символ A_i. Ошибка будет исправлена в тех случаях, когда полученный символ действительно образовалась из A_i, т.е. в (2^n-2^k) случаях.

Всего число случаев перехода в неразрешенные кодовые символы равно $2^k(2^n - 2^k)$. Таким образом, при наличии избыточности любой код способен исправлять ошибки. Отношение числа исправляемых кодом ошибочных кодовых символов к числу обнаруживаемых ошибочных комбинаций равно

$$\frac{(2^n - 2^k)}{2^k(2^n - 2^k)} = \frac{1}{2^k}$$

Например, в трехзначном равномерном двоичном коде число возможных кодовых комбинаций равно 8: (000, 001, 010, 011, 100, 101, 010, 111), минимальное кодовое расстояние между которыми (d_{ij}) равно 1. Ошибка в любом одном элементе символа кода превращает передаваемый разрешенный кодовый символ в другой, но также разрешенный. Это случай безизбыточного кода, не обладающего корректирующей способностью.

Если увеличить минимальное кодовое расстояние до 2 (d_{ij}=2), т. е. сделать разрешенными кодовые символы, отличающимися в двух элементах (001, 111, 010, 100), то ни один из разрешенных кодовых символов при одиночной ошибке не переходит в другой разрешенный кодовый символ. Например, подмножество разрешенных кодовых символов может быть образовано по принципу четности в нем числа единиц. Например, для n = 3:

000, 011, 101, 110 – запрещенные кодовые символы;

001, 010, 100, 111 – разрешенные кодовые символы.

В этом случае помехозащищенность кода увеличивается. Действительно, если в процессе передачи возникает ошибка только в одном элементе любого разрешенного символа, то эта комбинация превращается в запрещенную, что свидетельствует о появлении одиночной ошибки в символе кода, хотя и неизвестно, какой именно. Такой код не позволяет выяснить, какой конкретно из передаваемых элементов символа кода искажен, а значит и не позволяет исправить ошибку.

Код обнаруживает одиночные ошибки, а также другие ошибки нечетной кратности (при n = 3 - тройные).

Для исправления одиночной ошибки в кодовом символе необходимо сопоставить подмножество запрещенных кодовых символов

Чтобы эти подмножества не пересекались, кодовое расстояние между разрешенными кодовыми символами должно быть не менее трех. При n = 3 за разрешенные кодовые символы можно, например, принять 000 и 111. Тогда разрешенному кодовому символу 000 необходимо приписать подмножество запрещенных кодовых символов 001, 010, 100, образующихся в результате единичной

ошибки в символе 000.

Подобным же образом разрешенному кодовому символу 111 необходимо приписать подмножество запрещенных кодовых символов: 110, 011, 101, образующихся в результате возникновения единичной ошибки в разрешенном кодовом символе 111.

В этом случае, т.е. при увеличении кодового расстояния до 3 (d_{ij}=3), помехозащищенность кода еще более возрастает.

Действительно, если в процессе передачи произойдет ошибка в одном или двух элементах любого из двух разрешенных символов кода, то возникнет запрещенная комбинация элементов. Таким образом, применение этого кода дает возможность обнаружить две или, если произойдет только одна ошибка, что более вероятно, то можно восстановить переданный символ кода.

Из приведенного примера видно, что для обнаружения единичной ошибки в символе кода требуется один избыточный элемент в символах кода, чтобы обеспечить минимальное кодовое расстояние (расстояние между разрешенными символами (d_{ij})), равное 2, а для исправления в символах кода одной ошибки необходимо увеличить минимальное кодовое расстояние между символами кода (d_{ij}) до 3, т. е. ввести в символы кода два избыточных элемента.

Очевидно, что при необходимости обнаруживать ошибки кратности до p включительно минимальное кодовое расстояние между разрешенными кодовыми комбинациями должно быть, по крайней мере, на единицу больше p, т.е. $d_{min} \geq p + 1$.

Действительно, в этом случае ошибка, кратность которой не превышает p, не в состоянии перевести одну разрешенную кодовую комбинацию в другую.

В общем случае для того, чтобы код позволял обнаруживать все ошибки кратности p и исправлять все ошибки кратности q ($p>q$), его кодовое расстояние (d_{ij}) должно удовлетворять соотношению:

$$d_{ij} = 1 + p + q, \qquad (1.40)$$

где $p \geq q$,

p – количество обнаруживаемых ошибок в символах;

q – количество исправляемых ошибок в символах кода.

Таким образом, для обеспечения возможности исправления всех ошибок кратности до q включительно (при декодировании по методу максимального правдоподобия), каждая из ошибок должна приводить к запрещенному символу, относящемуся к подмножеству исходного разрешенного кодового символа.

Ошибка будет не только обнаружена, но и исправлена, если расстояние до

искаженного кодового символа (r) остается ближе к соответствующему разрешенному символу, чем к любому другому разрешенному кодовому символу, то есть выполняется условие: $r \le \frac{1}{2} d_{ij}$ или $d_{ij} \ge 2r + 1$. Это условие соответствует тому, что каждая из ошибок должна приводить к запрещенному символу, относящемуся к подмножеству исходной разрешенной кодовой комбинации.

Такой метод исправления одиночных независимых ошибок можно интерпретировать следующим образом. В подмножество каждого разрешенного символа относят все вершины, лежащие в сфере с радиусом $(d_{ij}-1)/2$ и центром в вершине, соответствующей данному разрешенному кодовому символу. Если в результате действия помехи символу переходит в точку, находящуюся внутри сферы с радиусом $(d_{ij}-1)/2$, то такая ошибка может быть исправлена.

Если помеха смещает точку разрешенного символа на границу двух сфер (расстояние $d_{ij}/2$) или больше (но не в точку, соответствующую другому разрешенному символу), то такое искажение может быть обнаружено.

Для кодов с независимым искажением символов лучшие корректирующие коды такие, у которых точки, соответствующие разрешенным кодовым комбинациям, расположены в n мерном пространстве равномерно.

Свойства корректирующих кодов по обнаружению и исправлению ошибок в зависимости от величины минимального кодового расстояния между разрешенными символами приведены в табл. 1.8

Таблица 1.8

Характеристики кодов			Свойства кодов
d_{ij}	p	q	
1	0	0	Отличает одну кодовую комбинацию от другой
2	1	0	Обнаруживает одну ошибку
3	1	1	Обнаруживает и исправляет одну ошибку
	2	0	Обнаруживает две ошибки
4	2	1	Обнаруживает две ошибки и исправляет одну
	3	0	Обнаруживает три ошибки
5	2	2	Обнаруживает и исправляет две ошибки
	3	1	Обнаруживает три ошибки и исправляет одну
	4	0	Обнаруживает четыре ошибки
			и т. д.

2. Коды, построенные на основе проверки на четность.

Проиллюстрируем построение корректирующего кода на следующем примере. Пусть исходный алфавит, состоящий из четырех символов, закодирован двоичным равномерным кодом: x_1 = 00; x_2 = 01; x_3 = 10; x_4 = 11. Этот код использует все возможные комбинации длинной 2 элемента, и поэтому не может обнаруживать ошибки (так как d_{ij} = 1).

Припишем к каждому кодовому символу один элемент - «0» или «1» так, чтобы число единиц в нем было четное, то есть x_1 = 000; x_2 = 011; x_3 = 101; x_4= 110 (Таблица 1.9)

Таблица 1.9

Неизбыточные кодовые символы	Контрольный избыточный элемент	Избыточные кодовые символы, обнаруживающие одиночную ошибку
00	0	000
01	1	011
10	1	101
11	0	110

Кодовое расстояние для полученного кода d_{ij} = 2, и, следовательно, он способен обнаруживать все однократные ошибки. Так как любая запрещенная комбинация содержит нечетное число единиц, то для обнаружения ошибки достаточно проверить кодовый символ на четность (например, суммированием по модулю 2 элементов кодового символа). Если число единиц в символе четное, то сумма по модулю 2 его элементов будет равна 0, если нечетное, то - 1. Признаком четности называют инверсию этой суммы. Вид контроля, когда по линии передается нечетное число единиц, по строгой терминологии называют *контролем по нечетности*.

Этот метод построения помехозащищенных кодов заключается в том, что в каждый кодовый символ двоичного кода добавляется для проверки один избыточный элемент (0 или 1) так, чтобы общее количество единиц в каждом символе кода было четным. При получении символов, таким образом построенных избыточных кодов, перед их раскодированием производится проверка их на четность. При одиночной ошибке количество единичных элементов в кодовом символе станет нечетным, что дает возможность обнаружить ошибку и осуществить повторную передачу.

Одиночная ошибка в символах двоичного кода инвертирует признак четности. Однако две ошибки инвертируют его дважды, то есть оставят без изменения, поэтому двойную ошибку контроль по четности не обнаруживает. Рассуждая аналогично, легко прийти к выводу, что контроль по четности обнаруживает все нечетные ошибки и не

реагирует на любые чётные. Пропуск чётных ошибок – это не какой-либо дефект такого вида кодов, это следствие предельно малой избыточности, равной всего одному разряду. Для большей помехозащищенности требуется соответственно и большая избыточность. Если ошибки друг от друга не зависят, то из необнаруживаемых ошибок чаще всего будут встречаться двойные ошибки, а при вероятности одиночной ошибки, равной P, вероятность двойной будет P^2. Поскольку в устройствах связи $P<<1$, необнаруженные двойные ошибки встречаются значительно реже, чем обнаруженные одиночные. Хотя этот вывод верен лишь для взаимно независимых ошибок.

Контроль по четности – самый дешевый по аппаратурным затратам вид контроля, и применяется он очень широко. Практически любой канал передачи цифровых данных или запоминающее устройство, если они не имеют какого-либо иного более мощного метода помехозащищенности, защищены контролем по четности.

Чтобы код был способен не только обнаруживать, но и исправлять однократные ошибки, необходимо добавить еще не менее двух разрядов. Это можно сделать различными способами, например, повторить первые две цифры: x_1 = 00000; x_2 = 01101; x_3 = 10110; x_4 = 11011.

Матрица расстояний этого кода представлена в Таблице 1.10, из которой видно, что $d_{ij} \geq 3$, что отвечает неравенству ($d_{ij} \geq 1+1+1$) и, следовательно, могут быть исправлены однократные ошибки.

Таблица 1.10.

	x_1	x_2	x_3	x_4
x_1		3	3	4
x_2	3		4	3
x_3	3	4		3
x_4	4	3	3	

Близким к рассмотренному способу построения корректирующих кодов является способ добавления контрольных избыточных элементов, на основе суммирования по основанию «2» элементов символов двоичного кода и на основе простейших логических операций над элементами символов кода.

3. На основе защиты сдвоенными элементами.

В таких кодах количество элементов в символах двоичного кода удваивается, причем к каждому элементу «0» приписывается «1», а к каждому элементу «1»

приписывается элемент «0». Например, исходный кодовый символ 11010 по этому методу кодируется следующим образом (рис. 1.10):

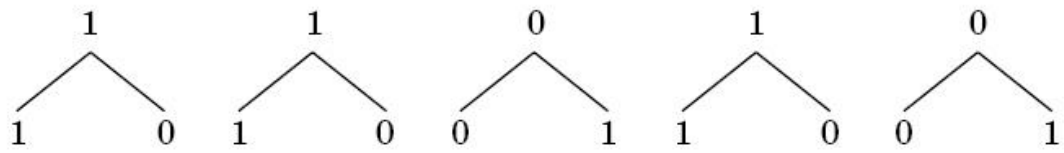


Рис. 1.10. Построение корректирующего кода на основе защиты сдвоенными элементами

Полученный корректирующий код позволяет обнаружить одиночную ошибку в каждом разряде путем сложения по модулю 2 каждой пары элементов символа корректирующего кода. Если вследствие единичного сбоя в какой–либо паре сумма будет равна 0, то это является сигналом ошибки в данном разряде.

1.12. Самокорректирующиеся коды

Самокорректирующиеся коды являются помехозащищёнными кодами, способными к обнаружению и самостоятельному исправления обнаруженных ошибок (самокоррекции), так как в самих таких кодах содержится информация о средствах их исправления. Процесс самокоррекции таких кодов удобно рассмотреть на примере блочного двоичного кода при самокоррекции его отдельного блока, соответствующего какому-либо символу исходного сообщения.

Пусть блок состоит из k двоичных элементов символа кода и представляет собой следующую последовательность: $a1, a2, \ldots ak$

Обнаружение ошибок можно провести путём проверки на чётность. Для этого каждый символ кода дополним ещё одним элементом – чётностью b, который выражается через элементы символа кода следующим образом:

$$b = \left\{ \begin{array}{c} 1, \text{если } a1 + a2 + \ldots + ak - \text{нечётное число} \\ 0, \text{если } a1 + a2 + \ldots + ak - \text{чётное число} \end{array} \right\},$$

и исходный блок запишется в виде: $a1, a2, \ldots ak, b$.

Элемент чётности b просто приписывается к исходному блоку справа. Как было показано ранее, это даёт возможность обнаружить наличие ошибок в символе кода, если число ошибок нечетно. Однако позиция, где в блоке расположена ошибка, неизвестна и для её исправления необходимо потребовать повторную передачу этого блока, т. е.

использовать обратную связь.

Для создания самокорректирующигося кода с возможностью исправления ошибок, без требования повторной передачи обнаруженного блока с ошибками, можно поступить следующим образом.

Пусть дан исходный блок конкретной длины *k*, например *k* = 12: *a1, a2,. . . ,a11, a12.*

Представим этот блок не в виде последовательности элементов, а в виде матрицы следующего вида (Рис. 1.11)

1	2	3	4
5	6	7	8
9	10	11	12

Рис. 1.11

Определим чётности всех строк и всех столбцов этой матрицы, приписав элементы чётности по её сторонам (Рис. 1.12):

a1	a2	a3	a4	b1
a5	a6	a7	a8	b2
a9	a10	a11	a12	b3
b4	b5	b6	b7	

Рис. 1.12

где *b1, b2, b3* – чётность строк, а *b4, b5, b6, b7* – чётность столбцов, причём

$$b1 = \left\{ \frac{1, \text{если } a1 + a2 + a3 + a4 - \text{нечётное число}}{0, \text{если } a1 + a2 + a3 + a4 - \text{чётное число}} \right\},$$

$$b2 = \left\{ \frac{1, \text{если } a5 + a6 + a7 + a8 - \text{нечётное число}}{0, \text{если } a5 + a6 + a7 + a8 - \text{чётное число}} \right\},$$

……,

$$b7 = \left\{ \frac{1, \text{если } a4 + a8 + a12 - \text{нечётное число}}{0, \text{если } a4 + a8 + a12 - \text{чётное число}} \right\},$$

Исходя из полученной матрицы можно получить новый избыточный блок считывая последовательно её строки:

a1, a2, a3, a4, b1 a5, a6, a7, a8, b2 a9, a10, a11, a12, b3, b4, b5, b6, b7, который позволяет восстановить исходный блок без повторной передачи. Действительно, если

ошибка произошла при передаче исходного блока (изменение одного символа на обратный) то это приведет к нарушению двух условий четности в строке и в столбце соответствующих этому символу. По ним легко восстановить правильное значение элемента исходного символа, для чего достаточно элемент, стоящий на пересечении строки и столбца с нарушенными условиями четности, изменить на обратный.

Например, если исходный блок имеет вид 101101000111, то его можно записать в виде матрицы (Рис.1.13):

1	0	1	1
0	1	0	0
0	1	1	1

Рис. 1.13

Эту матрицу можно дополнить элементами чётности(Рис.1.14):

1	0	1	1	1
0	1	0	0	1
0	1	1	1	1
1	0	0	0	

Рис. 1.14

Полученный избыточный блок имеет вид: 1011101001011111000. И он позволяет восстанавливать исходный блок без повторной передачи. Действительно, пусть под действием помех в канале связи принят следующий блок, соответствующий некоему кодовому символу(Рис.1.15):

1	0	1	1	1
0	1	0	0	1
1	1	1	1	1
1	0	0	0	

Рис. 1.15

Проверка на четность показывает, что четность нарушена в первом столбце и последней строке основного блока. Это значит, что элемент $a9$ должен быть изменён на обратный т.е. $a9 = 0$.

Ошибка может появиться и в элементах чётности *b1, b2, . . . b7.* В этом случае нарушается лишь одно условие чётности, что говорит об ошибке в элементе чётности.

Например, полученный блок после представления в матричной форме имеет вид:

1	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	0	

Рис. 1.16

Здесь нарушается четность только во второй строке, следовательно, ошибка в элементе четности $b2$, который должен быть изменён на обратный: $b2 = 1$.

Можно построить самокорректирующийся код, позволяющий исправить две, три и более ошибок в блоке. Для этого нужно делать дополнительную проверку на четность (например по диагоналям или ходом шахматного коня).

Самокорректирующиеся коды являются избыточными. Так в рассмотренном примере исходный блок ($k = 12$) после введения избыточности вырос до $k1$ ($k1 = 19$), т.е. избыточность достигла 60%. Важно отметить, что с увеличением длины блоков избыточность самокорректирующихся кодов существенно уменьшается. Так при $k = 100$ (матрица 10×10) необходимо 20 проверок на чётность и избыточность такого самокорректирующегося кода равна 20%. С ростом k избыточность будет ещё более уменьшатся.

Самокорректирующиеся коды являются примером самовосстанавливающихся систем, избыточность которых обеспечивает их устойчивость даже при действии существенных помех. Их избыточность позволяет им действовать весьма надёжно в очень «ненадёжной» среде.

1.13. Понятие качества корректирующего кода

Одной из основных характеристик корректирующего кода является избыточность кода, указывающая степень удлинения кодовой комбинации для достижения определенной корректирующей способности кода.

Если на каждые m элементов символов на выходе канала приходится k информационных и (m-k) проверочных элементов, то относительная избыточность кода может быть выражена одним из соотношений: $R_m = (m-k)/m$ или $R_k = (m-k)/k$.

Величина R_k, изменяющаяся от 0 до ∞, предпочтительнее, так как лучше отвечает смыслу понятия избыточности. Коды, обеспечивающие заданную корректирующую

способность при минимально возможной избыточности, называют оптимальными.

Однако не всегда целесообразно стремиться к использованию кодов, близких к оптимальным. Необходимо учитывать другой, не менее важный показатель качества корректирующего кода – сложность технической реализации процессов кодирования и декодирования.

Если информация должна передаваться по медленно действующему и дорогостоящему каналу связи, а кодирующее и декодирующее устройства предполагается выполнить на высоконадежных и быстродействующих элементах, то сложность этих устройств не играет существенной роли. Решающим фактором в этом случае является повышение эффективности пользования канала связи, поэтому желательно применение корректирующих кодов с минимальной избыточностью.

Если же корректирующий код должен быть применен в системе, выполненной на элементах, надежность и быстродействие которых равны или близки надежности и быстродействию элементов кодирующей и декодирующей аппаратуры, то допустимо применение корректирующих кодов с достаточно высокой избыточностью.

Это возможно, например, для повышения достоверности воспроизведения информации с запоминающего устройства ЭВМ. Тогда критерием качества корректирующего кода является надежность системы в целом, то есть с учетом возможных искажений и отказов в устройствах кодирования и декодирования. В этом случае часто более целесообразны коды с большей избыточностью, но простые в технической реализации.

1.14. Кодирование как средство защиты информации от несанкционированного доступа. Основные определения

Часто хранимая и передаваемая информация может представлять интерес для лиц, желающих использовать ее в своих интересах, поэтому важную роль играет информационная безопасность, которая должна обеспечить защиту конфиденциальной информации от ознакомления с ней конкурирующих групп. Это вынуждает использовать различные виды защиты информации от несанкционированного доступа, т. е. защиты информации от лиц, которым она не предназначена.

Особую важность эта проблема приобрела с распространением письменности в человеческом обществе, когда появилась потребность в обмене письменными сообщениями, что, в свою очередь, вызвало необходимость сокрытия их содержимого от посторонних. Методы сокрытия содержимого письменных сообщений можно разделить на

четыре группы.

Один из них предполагает защиту ее чисто силовыми методами – охрана документа (носителя информации) физическими лицами, передача его специальным курьером и т. п.

Ко второй группе относятся методы маскировки, которые осуществляют скрытие самого факта наличия сообщения. Реализация этого метода чаще всего связана с использованием, так называемых, симпатических чернил, которые проявляются при соответствующем воздействии на документ, но могут использоваться и более экзотические методы. Например, один из них приведён в трудах древнегреческого историка Геродота. На голове раба, которая брилась наголо, записывалось нужное сообщение и, когда волосы достаточно отрастали, раба отправляли к адресату, который снова брил его голову и считывал сообщение.

Третью группу составляют различные методы тайнописи или криптографии (от греческих слов kryptos - тайный и grapho - пишу), которые применяются для изменения сообщения с целью сделать текст непонятным для непосвященных.

С развитием науки и техники стали применяться методы четвёртой группы, которые ориентированы на создание специальных технических устройств, например, устройства инвертирования речи.

В отдельную группу можно выделить методы сокрытия истинного содержимого письменных сообщений путём замены целых слов, путём использования некоего тайного языка. Например, «Тётя завтра приезжает в гости». Здесь не известно кто такая «Тётя» и где обитают «гости». Секретное содержимое сообщений, скрытое подобными методами сокрытия, невозможно установить не имея ключей соответствия между словами и их зашифрованными значениями.

Представляет интерес рассмотрение третьей группы способов защиты информации, которые основываются на процессе кодирования, то есть на преобразовании исходных сообщений и их символов (данных) в форму (код), при которой исходные сообщения становятся доступными только при наличии у получателя некоторой специфической информации (ключа), позволяющей выполнить обратное преобразование и получить исходное сообщение. Такой вид защиты информации называют криптографической защитой информации и осуществляют её специфическими операциями кодирования и декодирования, которые носят названия шифрования и дешифрования соответственно.

В общем случае методы третьей группы (методы криптографии) можно определить как некоторое множество отображений одного пространства (пространства возможных исходных сообщений) в другое пространство (пространство возможных криптограмм). Каждое конкретное отображение из этого множества соответствует шифрованию при

помощи конкретного ключа.

Эта область теории кодирования обладает собственной терминологией.

Область знаний о шифрах, методах их создания и раскрытия называется криптографией (или тайнописью).

Криптография - это составная часть криптологии - науки, занимающейся шифрами. Еще одна составляющая криптологии - криптоанализ. Криптограф занимается методами обеспечения секретности и подлинности информации, а криптоаналитик пытается выполнить обратную задачу, т.е. вскрыть шифр или подделать кодированные сигналы так, чтобы они были приняты за подлинные.

Сообщение, текст которого необходимо сделать непонятным или недоступным для посторонних, называют исходным сообщением или открытым текстом.

Зашифрованное сообщение называют криптограммой.

Шифрование (зашифрование) данных - процесс преобразования открытых данных в зашифрованные данные (шифротекст, криптограмму) при помощи шифра.

Шифр - это множество обратимых преобразований формы сообщения с целью его защиты от несанкционированного прочтения. В шифре всегда различают два элемента: алгоритм и ключ.

Ключ - конкретное секретное состояние некоторого параметра (параметров), обеспечивающее выбор одного преобразования из совокупности возможных для используемого метода шифрования.

Шифры, в которых для зашифровки и расшифровки используется один и тот же ключ, называются симметричными.

В настоящее время широкое распространение получили шифры с открытым ключом. Эти шифры не являются симметричными – для зашифровки и расшифровки используется различные ключи. При этом ключ, используемый для зашифровки, является открытым (не секретным) и может быть сообщен всем желающим отправить шифрованное сообщение, а ключ, используемый для расшифровки, является закрытым и хранится в секрете получателем шифрованных сообщений. Даже знание всего зашифрованного сообщения и открытого ключа, с помощью которого оно было зашифровано, не позволяет дешифрировать сообщение без знания закрытого ключа.

Расшифрование данных – это преобразование по криптографическому алгоритму зашифрованного текста в открытый с использованием известного криптографического ключа.

Дешифрование данных - процесс преобразования зашифрованного текста в открытый при неизвестных ключе и алгоритме.

Свойство шифра противостоять раскрытию (его стойкость к дешифрированию) называется криптостойкостью (или надёжностью) и обычно измеряется сложностью алгоритма дешифрирования.

Любая криптографическая система шифрования надежна лишь настолько, насколько полно она отвечает следующим требованиям:

- невозможность ее раскрытия даже при известном тексте, а в случае раскрытия сообщения - гарантия безопасности сообщений, которые были переданы ранее, и тех, которые будут переданы в дальнейшем;

- достаточно большое число вариантов шифрования, не позволяющее раскрыть истинное содержание информации даже с использованием современных вычислительных средств;

- высокая сложность шифра, не позволяющая раскрыть его с применением различных математических методов;

- гарантированная надежность хранения ключа и алгоритма шифрования, а также самих шифровальных устройств.

В практической криптографии криптостойкость шифра оценивается из экономических соображений. Если раскрытие шифра стоит (в денежном отношении, включая необходимые компьютерные ресурсы, специальные устройства и т. п.) больше, чем сама зашифрованная информация, то шифр считается достаточно надёжным.

Методы криптографического закрытия могут иметь как программную, так и аппаратную реализацию.

Программная реализация осуществляется на основе вычислительных процессов, как на этапе шифрования, так и на этапе дешифрирования. Аппаратная реализация основана на использовании специальной аппаратуры.

Основной задачей криптографической защиты является обеспечение невозможности доступа к информации при условии, что потенциальный противник обладает любым техническим оборудованием, способным перехватить и записать криптограммы, и ему известны некоторые фрагменты криптограмм и соответствующие им части исходного сообщения.

Основное требование к шифру состоит в том, чтобы расшифровка была возможна только при наличии санкции, то есть некоторой дополнительной информации (или устройства), которая называется ключом шифра. Процесс декодирования шифровки без ключа называется дешифрированием (или просто раскрытием шифра).

1.15. Примеры некоторых простейших шифров.

Необходимость в обеспечении секретности важных посланий возникла еще в глубокой древности. Коды и шифры определяли исход многих войн и политических интриг на протяжении всей истории человечества. Со временем люди находили новые, все более сложные способы делать послания недоступными чужим глазам. Поэтому с давних пор и по сей день известно большое число различных методов криптографического закрытия (шифров) как чисто информационных, так и механических, которые имеют различные степени сложности и надёжности защиты. Представляет интерес рассмотрение некоторых простейших из них на примере шифрования текстов.

Стеганография – это наука о тайной передаче сообщений путем сокрытия самого факта наличия сообщения, это искусство скрытия письма, чтобы незаметно доставить его получателю. Не стоит путать ее с криптографией, которая скрывает содержание сообщения, а не его наличие. Эта техника более древняя, чем различные коды и шифры. Например, сообщение может быть написано на бумаге, а затем покрыто ваксой и проглочено, или нанесено на бритую голову курьера, которое будет скрыто заново выросшими волосами. Чаще всего для стенографии использовали повседневные объекты. Когда-то в Англии использовался такой метод: под некоторыми буквами на первой странице газеты стояли крохотные точки, почти невидимые невооруженным глазом, секретное сообщение получалось при чтении только помеченных букв. Иногда писали сообщение первыми буквами слов, составляющих какой либо текст, или использовали невидимые чернила. Была распространена практика уменьшения целых страниц текста до микроскопического размера, так что их было легко пропустить при чтении чего-то относительно безобидного. Стеганографию часто используют в сочетании с другими методами шифрования, так как всегда есть шанс, что ваше скрытое послание обнаружат и прочитают.

Транспозиция. В транспозирующих шифрах буквы переставляются по заранее определенному правилу. Например, можно каждое слово писать задом наперед, или менять местами каждые две буквы. Подобные шифры использовались в Гражданскую Войну в США и в Первую Мировую Войну, чтобы посылать важные сообщения. Сложные ключи могут сделать такой шифр довольно сложным на первый взгляд, но многие сообщения, закодированные подобным образом, могут быть расшифрованы простым перебором ключей на компьютере.

Шифр простой подстановки. При этом методе шифровки все символы алфавита однозначно заменяют другими символами этого же или иного алфавита. Если объём алфавита исходного сообщения равен n и замена производится из этого же алфавита, то

существует $n!$ способов замены символов исходного сообщения, т. е. существует $n!$ различных ключей. Примером такого шифра может служить шифр (азбука) Морзе. В азбуке Морзе каждая буква алфавита, все цифры и наиболее важные знаки препинания имеют свой кодовый символ, состоящий из череды коротких и длинных импульсов, часто называемых «точками и тире». Так, А — это «•—», Б — «—•••», и т.д.

В художественной литературе классическим примером шифра простой подстановки является шифр «Пляшущие человечки» (К. Дойль).

Историческим примером шифра подстановки является шифр Цезаря (I век до н. э.). По легенде шифр Цезаря называется так, потому что его использовал сам Юлий Цезарь. На самом деле шифр Цезаря — это не один шифр, а целых двадцать шесть, использующих один и тот же принцип.

Применительно к тексту на русском языке он состоит в следующем. Выписывается весь алфавит, а затем под ним выписывается тот же алфавит, но со сдвигом, например, на 3 буквы:

а б в г д ..... ъ э ю я

г д е ё ж ..... я а б в

При шифровании буква *а* заменяется буквой *г, б – д* и так далее. Получатель сообщения проделывал обратную последовательность операций и восстанавливал исходное сообщение. Ключом в шифре Цезаря является величина сдвига алфавита в нижней строке, которая в принципе может быть любой (от 1 до 32). На шифре Цезаря базируется огромное число других, более сложных шифров, но сам по себе он не представляет интереса из-за легкости дешифровки. Защищённость такого метода шифрования довольно низкая, для его взлома достаточно перебрать всего лишь 32 комбинации, правда, сначала нужно убедиться, что используется именно шифр Цезаря.

К шифрам простой подстановки относят и различные алфавитные шифры, основанные на различной очерёдности символов алфавита используемого при шифровании. Например:

А Б В Г Д Е Ё Ж З И К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ъ Ы Э Ю Я

Щ Ь Ъ Ы Э Ю Я А Б В Г Д Е Ё Ж З И К Л М Н О П Р С Т У Ф Х Ц Ч Ш,

в котором вверху представлены символы исходного алфавита, а внизу соответствующие им символы того же алфавита с изменённой очерёдностью. Недостатком этого шифра является необходимость хранения или запоминания довольно сложной таблицы соответствия, что затруднительно. Этого недостатка лишены алфавитные шифры с использованием в качестве ключа кодового слова. Нижняя часть таблицы соответствия таких кодов представляет собой исходный алфавит, перед которым

пишется кодовое слово без повтора букв и символы (буквы) входящие в кодовое слово исключаются из исходного алфавита в нижней части таблицы. В качестве примера приведёна таблица соответствия алфавитного шифра, в котором использовано в качестве ключа слово «школа».

А Б В Г Д Е Ё Ж З И К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ъ Ы Э Ю Я

Ш К О Л А Б В Г Д Е Ё Ж З И М Н П Р С Т У Ф Х Ц Ч Щ Ь Ъ Ы Э Ю Я

Разновидностью рассмотренных шифров простой подстановки является их цифровой вид, в котором буквы алфавита заменяются их порядковыми номерами.

Примером такого типа шифров может являться, так называемый, книжный шифр, который заключается в том, что каким–либо малозаметным способом помечаются буквы секретного сообщения в тексте книги или другого печатного текста. Интересно отметить, что во время Первой Мировой Войны германские шпионы использовали этот шифр, нанося симпатическими чернилами точки на букве газетного текста. Книжный шифр в современном его виде имеет несколько иной вид. Суть его состоит в замене на номер строки и номер этой буквы на заранее оговоренной странице некоторой книги. Ключом такого шифра является книга и используемая страница в ней. Этот шифр применялся даже во времена Второй Мировой Войны.

Ещё одним примером шифра подстановки может служить, так называемый, квадрат Полибия. Шифрование в этом случае заключается в следующем. В квадратную матрицу с числом элементов, равным или большим объёма исходного алфавита, на место каждого элемента в произвольном порядке вписываются все буквы алфавита. Шифруемая буква заменяется её координатами в матрице. При расшифровке каждая пара чисел определяла соответствующую букву сообщения. Ключом является расположение букв в исходной матрице. Отметим, что при произвольном расположении букв в матрице возникает некоторое затруднение: либо надо помнить отправителю и получателю сообщения заданное расположение букв в матрице (ключ шифра), что затруднительно, либо иметь при себе запись этого ключа, что представляет опасность ознакомления с ключом посторонних лиц. Поэтому в ряде случаев ключ представляют следующим образом. Берётся какое – либо «ключевое слово», которое легко запомнить, удаляют из него повторы букв и записывают его в начальных элементах матрицы. На место остальных элементов записывают остальные буквы алфавита в естественном порядке. Разновидностью такого шифра является, так называемый, «тюремный шифр», при котором матрица заполняется буквами в порядке их следования в алфавите.

Рассмотренные шифры простой подстановки относятся к одному и тому же типу шифров — шифров моноалфавитной замены. Это значит, что в этих шифрах каждая буква

заменяется на одну и только одну другую букву или символ. Эти шифры достаточно просты, но не обеспечивают высокой степени защиты. Такие шифры очень легко расшифровать даже без знания ключа. Делается это при помощи частотного анализа, так как буквы любого языка обладают определенной и различной вероятностью появления. В зашифрованном таким образом тексте статистические свойства исходного сообщения сохраняются, поэтому, анализируя криптограммы достаточной длительности, можно их дешифрировать, исходя из их статистических свойств. К сожалению, этот принцип работает только для длинных сообщений. Короткие сообщения просто не содержат в себе достаточное число символов, чтобы с достаточной достоверностью выявить соответствие наиболее часто встречающихся символов буквам из обычного алфавита. По легенде Мария Стюарт использовала невероятно сложный моноалфавитный шифр с несколькими вариациями, но когда его наконец-то взломали, прочитанные сообщения дали ее врагам достаточно поводов, чтобы приговорить ее к смерти.

Шифры перестановки. Суть этого шифра заключается в следующем. Берётся некоторое число n и записывается в строку ряд чисел 1, 2, …, n затем под ними записываются те же цифры в произвольном порядке. Например, для $n = 5$:

1 2 3 4 5

4 3 2 5 1

После этого записывается шифруемое сообщение без пропусков и разбивается на группы по n букв. Если число букв до n не кратно n, то последняя группа дополняется до n любыми буквами. После этого буквы каждой группы переставляются в соответствии с выбранной двухстрочной таблицей: первая буква становится четвёртой, вторая – третьей и так далее. После выполнения перестановки в каждой группе полученный текст записывается без пропусков. Ключом этого шифра является таблица перестановок. При дешифрировании криптограмма разбивается на группы по n букв и буквы перестанавливаются в обратном порядке.

Шифр Вижинера. Этот шифр сложнее, чем моноалфавитные шифры (шифры простой подстановки). Шифр Виженера использует тот же принцип, что и шифр Цезаря, за тем исключением, что каждая буква меняется в соответствии с некоторым кодовым словом.

Шифрование по этому методу заключается в следующем. Каждая буква алфавита нумеруется, скажем, для русского языка ставятся в соответствие цифры от 1 ($А = 1$) до 33 ($Я = 33$)

А	*Б*	*В*	…	*Я*
1	*2*	*3*	…	*33*

В качестве ключа используется некоторое слово или просто определённая последовательность букв. Этот ключ подписывается с повторением под шифруемым сообщением так, чтобы под каждой буквой исходного сообщения находилась одна буква ключа. Криптограмма формируется как последовательность цифр, получаемых в результате суммирования числовых эквивалентов, соответствующих букве исходного сообщения и букве стоящего под ней ключа, и приведённой по модулю 33 (объём алфавита). Степень надёжности закрытия сообщений достаточно высока, так как этот шифр нарушает статистическое распределение вероятностей появления отдельных символов (букв) в сообщении.

Для обеспечения достаточно высокой надёжности закрытия необходимо использовать весьма длинные ключи, что сопряжено с определёнными трудностями. Шифр Вижинера с неограниченным неповторяющимся ключом известен как шифр Вернама.

Очень долгое время шифр Виженера считался невзламываемым. Чтобы его расшифровать, для начала угадывают длину кодового слова и применяют частотный анализ к каждой *n*-ной букве послания, где *n* — предполагаемая длина кодового слова. Если длина была угадана верно, то и сам шифр вскроется с большей или меньшей долей вероятности. Если предполагаемая длина не дает верных результатов, то пробуют другую длину кодового слова.

Шифрование гаммированием. Суть этого метода шифрования заключается в том, что цифровые эквиваленты символов сообщения (букв) складываются с псевдослучайной последовательностью чисел, именуемой гаммой, и приводятся по модулю *k*, где *k* – объём алфавита источника сообщений. Таким образом, ключом в этом способе шифрования является псевдослучайная последовательность чисел.

Псевдослучайную последовательность генерируют на основе регистров сдвига с обратными связями. Соответствующим выбором обратных связей можно добиться генерирования последовательностей с периодом повторения $T = 2^n - 1$ символов, где *n* – число разрядов регистра. Получаемые таким образом последовательности чисел являются псевдослучайными, так как они удовлетворяют ряду основных тестов на случайность, что существенно затрудняет раскрытие такого ключа, и в то же время являются детерминированными, что позволяет обеспечить однозначность дешифрирования сообщений.

Надёжность криптографического закрытия методом гаммирования в основном зависит от длины периода неповторяющейся части гаммы и, если длина периода

превышает длину шифруемого сообщения, то раскрыть криптограмму, опираясь только на статистические результаты обработки, теоретически невозможно.

Однако, если известно некоторое число цифровых эквивалентов символов сообщения и соответствующие им символы криптограммы, дешифрирование довольно легко выполнить, так как преобразование, осуществляемое при гаммировании, является линейным. Для полного раскрытия криптограммы достаточно всего $2n$ соответствующих пар символов исходного сообщения и символов криптограммы.

Шифр Энигмы. Энигма — это шифровальная машина, использовавшаяся нацистами во времена Второй Мировой Войны. В простейшем случае она состоит из сопряжённых между собой шестерён и клавиатуры. При шифровании на экране показывалась буква, которой шифровалась соответствующая буква на клавиатуре. То, какой будет зашифрованная буква, зависело от начальной конфигурации шестерён. Существовало более ста триллионов возможных комбинаций положения шестерён, и в процессе набора текста колеса сдвигались сами, так что шифр менялся на протяжении всего сообщения. Все Энигмы были идентичными, так что при одинаковом начальном положении шестерён на разных машинах текст был одинаковый. Имея список положений шестерён на каждый день, можно легко расшифровывать сообщения друг друга, но без знания положений шестерён расшифровать послания невозможно.

Настоящие коды. Вопреки распространенному мнению, код и шифр — это не одно и то же. В настоящих кодах каждое слово исходного кодируемого сообщения целиком заменяют на какое-то иное кодовое слово, в то время как в шифре заменяются отдельные символы сообщения (в случае текста это буквы). Расшифровывается такое послание с помощью кодовой книги, где записано соответствие всех настоящих слов кодовым, как в словаре. Преимущества такого способа в том, что сообщению необходимо быть чрезвычайно длинным, чтобы можно было его взломать с помощью частотного анализа, так что такие коды полезнее некоторых шифров. Многие страны использовали коды, периодически их меняя, чтобы защититься от частотного анализа. При этом способе кодирования кодовая книга является критическим предметом, и в случае, если она будет похищена, код полностью раскрыт и с ее помощью больше будет невозможно что-либо зашифровать, и придется придумывать новый код, что требует огромных усилий и затрат времени.

Шифрование публичным ключом. Этот алгоритм шифрования в различных модификациях широко применяющийся в различных компьютерных системах. Принцип шифрования заключается в том, что есть два ключа: открытый и секретный. Открытый ключ — это некое очень большое число (порядка несколько десятков или сотен знаков),

имеющее только два делителя, помимо единицы и самого себя. Математически чрезвычайно трудно найти делители очень большого числа. Эти два делителя являются секретным ключом, и при перемножении дают открытый (публичный) ключ. Например, открытый ключ — это 1961, а секретный — 37 и 53. Открытый ключ используется для того, чтобы зашифровать сообщение, а секретный — чтобы расшифровать. Даже знание всего зашифрованного сообщения и открытого ключа, с помощью которого оно было зашифровано, не позволяет дешифрировать сообщение без знания закрытого ключа. Без секретного ключа расшифровать сообщение невозможно

2. СЖАТИЕ СООБЩЕНИЙ

2.1. Общие понятия и определения.

Из теории информации известно, что чем больше энтропия источника сообщения, тем большее количество информации содержит в среднем каждый символ сообщения. Соответственно, при передаче одинакового количества информации сообщение тем длиннее, чем меньше присущая ему энтропия. Увеличивая энтропию сообщения можно сделать его короче.

Пусть энтропии двух сообщений равны H_1 и H_2 соответственно, причём $H_1 < H_2$, а количество информации, содержащееся от них одинаковое (I), тогда справедливо равенство:

$$I = n_1 H_1 = n_2 H_2,$$

где n_1 и n_2 - число символов в соответствующих сообщениях (длина сообщений).

Откуда следует:

$$\frac{H_1}{H_2} = \frac{n_2}{n_1} = \mu .$$

Коэффициент μ характеризует долю излишних символов в сообщении или долю излишних элементов в символах кода. В этом случае его называют *коэффициентом избыточности* и, если он отличен от единицы, можно говорить об исходной избыточности сообщений. При помехоустойчивом кодировании избыточность играет положительную роль, благодаря ей сообщения более защищены от помех. Однако при хранении информации избыточность играет отрицательную роль, так как увеличивает объёмы обрабатываемых массивов данных и используемых ресурсов памяти и, соответственно, замедляет процессы обмена информацией. Поэтому для эффективного хранения и передачи информации необходимо сокращать длину сообщений, производя их сжатие путём устранения их избыточности. Избыточность исходных сообщений

устраняют их эффективным кодированием путём использования оптимальных неравномерных кодов, которые укорачивают сообщения по сравнению с использованием равномерных кодов, и тем самым повышают энтропию сообщений. В дальнейшем длина закодированных сообщений может быть существенно сокращена после применения специальных методов, разработанных с этой целью. Комплекс операций, нацеленных на сокращение длины сообщений (без потери содержащейся в них информации и с возможностью восстановления их первоначального вида) называют сжатием (архивацией, упаковкой, компрессией) сообщений или данных. Под *данными* обычно понимают информацию, представленную в формализованном виде, позволяющем осуществлять ее обработку с помощью технических средств.

В общем виде, сжатие данных производится путём использования специальных алгоритмов преобразования фрагментов данных, позволяющих при прямом преобразовании (сжатии, упаковке) уменьшить длину данных (т.е. сделать количество символов в сжатом сообщении меньше, чем в исходном), а при обратном преобразовании (восстановлении) восстановить исходное сообщение в первоначальном (или близком к нему) виде.

В этом случае коэффициент μ определяет степень сокращения длины исходного сообщения при переходе к сжатому сообщению, и называется *коэффициентом сжатия.* Коэффициент сжатия может быть как постоянным, так и переменным. Во втором случае он может быть определён либо для каждого конкретного сообщения, либо оценён по некоторым критериям как:

- средний (обычно по некоторому тестовому набору данных);
- максимальный (случай наилучшего сжатия);
- минимальный (случай наихудшего сжатия);
- какой-либо другой.

В результате сжатия (архивации) исходных сообщений получают архивные файлы или архивы. Программа, осуществляющая сжатие или архивацию файлов в архив, а также распаковку архивов часто называют *архиватор.*

Современные архиваторы при упаковке данных позволяют сохранять их файловую структуру. Поэтому результатом работы программы-архиватора является *архив (архивный файл)* — файл, содержащий в себе один или несколько других файлов, вместе с метаданными. Метаданные содержат информацию об именах и длине оригинальных файлов для их восстановления, поэтому большинство архиваторов сохраняют метаданные файлов, предоставляемые операционной системой, такие, как время создания и права доступа. Подробное описание структуры данных записанных в компьютерном *архивном*

файле определяется его *форматом*. *Формат файла* иногда указывается в его имени, как часть, отделённая точкой (обычно эту часть называют расширением имени файла). Файлы, содержимое которых соответствует одному формату или одному семейству форматов, иногда называют файлами одного типа.

Важно отметить, что, в общем случае, по одному только содержимому файла принципиально невозможно установить, является ли данный файл исходным файлом или сжатым образом этого файла. Например, если получен текст "4 Abc", то невозможно установить, является ли он исходным оригинальным файлом, либо следует его считать сжатой копией текста "AbcAbcAbcAbc". В этом случае может потребоваться дополнительная информация о том, как следует интерпретировать содержимое полученного файла. Поэтому архивированные файлы обычно имеют специальное расширение (.ZIP, .ARJ, .RAR), которое в данном случае несет информацию о том, каково содержимое файла Расширение файла не является частью его содержимого. Поэтому переименование файла и изменение его расширения не изменит содержимого этого файла, но автоматическое распознавание, того является ли файл оригинальным или сжатым, станет невозможным.

К основным характеристикам архиваторов относят *степень сжатия* и *скорость сжатия*. Эти характеристики, как правило, обратно зависимы. То есть, чем больше скорость сжатия, тем меньше степень сжатия, и наоборот. Причём, сжатие данных обычно происходит значительно медленнее, чем обратная операция.

Современные архиваторы применяют различные методы сжатия в зависимости от типа и особенностей сжимаемой информации.

Основным критерием различия между различными методами сжатия информации является наличие или отсутствие потерь информации при их использовании. С этой точки зрения все методы сжатия данных можно разделить на два основных класса: методы *сжатия без потерь* и методы *сжатия с потерями*. При использовании методов сжатия без потерь возможно полное восстановление исходных данных. Используемые в этих методах алгоритмы являются обратимыми: любой массив входных данных при сжатии и последующей распаковке восстанавливается абсолютно точно. Поэтому методы сжатия без потерь (обратимые алгоритмы) можно применять для сжатия информации любого рода. Сжатие же с потерями позволяет восстановить данные с искажениями, обычно несущественными с точки зрения дальнейшего использования восстановленных данных.

С точки зрения практического применения, алгоритмы сжатия без потерь универсальны, т. к. их применение возможно для данных любого типа, в то время как возможность применения алгоритмов сжатия с потерями должна быть обоснована. Для некоторых типов данных искажения не допустимы в принципе. В их числе:

- символические данные, изменение которых неминуемо приводит к изменению их семантики, программы и их исходные тексты, двоичные массивы и т. п.;

- жизненно важные данные, изменения в которых могут привести к критическим ошибкам, например, данные получаемые с медицинской измерительной аппаратуры или контрольных приборов летательных, космических аппаратов и т. п.;

- многократно подвергаемые сжатию и восстановлению промежуточные данные при многоэтапной обработке графических, звуковых и видеоданных.

Сжатие без потерь обычно используется для передачи и хранения текстовых данных, компьютерных программ и т. п., реже — для сокращения объёма аудио- и видеоданных, цифровых фотографий, то есть в тех случаях, когда искажения недопустимы или крайне нежелательны. Сжатие с потерями, обладающее значительно большей, чем сжатие без потерь, эффективностью, обычно применяется для сокращения объёма аудио и видеоданных и цифровых изображений, т.е. в тех случаях, когда степень сжатия является приоритетом, а полное соответствие исходных и восстановленных данных не требуется. Коэффициент сжатия с потерями при этом сильно зависит от допустимой погрешности сжатия или качества восстановленного документа, которое обычно выступает как параметр используемого алгоритма сжатия. В общем случае постоянный коэффициент сжатия способны обеспечить только методы сжатия данных с потерями.

По области использования программы-архиваторы можно разделить на три категории.

1. Программы, используемые для сжатия исполняемых файлов, причем все файлы, которые прошли сжатие, свободно запускаются, но изменение их содержимого, например, русификация, возможно только после их разархивации.

2. Программы, используемые для сжатия мультимедийных файлов, причем можно после сжатия эти файлы свободно использовать, хотя, как правило, при сжатии изменяется их формат (внутренняя структура), а иногда и ассоциируемая с ними программа, что может привести к проблемам с запуском.

3. Программы, используемые для сжатия любых видов файлов и каталогов, причем в основном использование сжатых файлов возможно, только после разархивации.

2.2. Методы и алгоритмами сжатия.

Действие методов сжатия данных основано на использовании алгоритмов сжатия (архиваторов). Суть работы архиваторов заключается в том, что они находят в сжимаемых файлах избыточные элементы сообщений (лишние символы, повторяющиеся участки,

пробелы и т. п.) и эффективно кодируют их (с целью получения минимального объема), а затем при распаковке восстанавливают исходные файлы по особым отметкам. Простейшие архиваторы просто последовательно объединяют (упаковывают) содержимое файлов в архив.

Самым известным методом архивации файлов является сжатие последовательностей одинаковых символов. Суть его заключается в следующем. Допустим, что внутри исходного сообщения содержатся последовательности символов, которые многократно повторяются. Вместо того чтобы хранить каждый символ, достаточно фиксировать количество повторяемых символов и их позиции. Например, сжимаемое сообщение длиной 15 символов состоит из следующих символов:

B B B B B L L L L L A A A A A

Архиватор может представить это сообщение в следующем виде:

01 05 B 06 05 L 11 05 A

Такая запись означает, что с первой позиции пять раз повторяется символ "B", с позиции 6 пять раз повторяется символ "L" и с позиции 11 пять раз повторяется символ "A". Для хранения сообщения в такой форме потребуется всего 9 символов, что на 6 символов (40%) меньше чем необходимо для хранения исходного файла.

Описанный метод является простым и очень эффективным способом сжатия сообщений. Однако он не обеспечивает большой экономии объема, если обрабатываемое сообщение содержит небольшое количество последовательностей повторяющихся символов.

Программные реализации этого метода отличаются простотой, высокой скоростью работы, но в среднем обеспечивают невысокое сжатие. Наилучшими объектами сжатия для данного метода являются графические данные, в которых большие одноцветные участки изображения кодируются длинными последовательностями одинаковых символов. Различные реализации этого метода используются при сжатии некоторых типов графических файлов, при факсимильной передаче информации. Для текстовых данных методы сжатия последовательностей одинаковых символов, как правило, неэффективны.

Если сжимаемые сообщения содержат небольшое количество последовательностей повторяющихся символов, более эффективны методы сжатия данных основанные на использовании алгоритмов эффективного кодирования, в частности, алгоритмов кодирования символами кода переменной длины (например, по алгоритму Шеннона – Фено или алгоритму Хаффмана). Эти методы, в том или ином виде, используются практически любым архиватором. Коды с символами переменной длины (неравномерные

коды) позволяют записывать наиболее часто встречающиеся символы и группы символов исходного сообщения короткими кодовыми символами, состоящими всего лишь из нескольких элементов, в то время как редко появляющиеся символы и фразы будут записаны более длинными кодовыми символами. Например, в любом тексте на русском языке буквы О или Е встречаются гораздо чаще, чем Щ или Ъ. Таким образом, используя специальную таблицу соответствия, можно закодировать буквы О или Е кодовыми символами с меньшим числом элементов, а для более редких букв использовать более длинные кодовые символы

Некоторые популярные архиваторы работают на основе *алгоритмов Лемпела-Зива***.** Якоб Зив (Jacob Ziv) и Абрахам Лемпель (Abraham Lempel) - израильские ученые, которые в конце 1970-х годов предложили алгоритмы сжатия LZ77 и LZ78 (цифры в названии указывают год публикации алгоритма).

Оригинальность этих алгоритмов заключается в том, что кодируются не отдельные символы исходного массива данных, а повторяющиеся цепочки символов. Например, любой текст состоит из букв, а буквы составляют слова. Количество слов в тексте намного меньше, чем букв в том же тексте, т. к. длина слов сравнительно велика (в русском языке средняя длина слова составляет 7 - 8 букв). Если кодировать слова целиком и даже целые фразы, а не отдельные буквы, то оказывается, что текст можно сжать гораздо сильнее. Поэтому архиваторы на основе алгоритмов Лемпела-Зива (LZ77 и LZ78) классифицируются как адаптивные словарные кодировщики, в которых текстовые строки заменяются указателями на идентичные им строки, встречающиеся ранее в тексте. Например, все слова какой-нибудь книги могут быть представлены в виде номеров страниц и номеров строк некоторого словаря. Важнейшей отличительной чертой этого алгоритма является использование грамматического разбора предшествующего текста с разложением его на фразы, которые записываются в словарь. Указатели позволяют сделать ссылки на любую фразу в окне установленного размера, предшествующего текущей фразе. Если соответствие найдено, текущая фраза заменяется указателем на своего предыдущего двойника.

Принцип работы алгоритма LZ77 в самом общем виде таков:

- входные данные читаются последовательно, текущая позиция считывания условно разбивает массив входных данных на прочитанную и непрочитанную части, причём прочитанная часть используется как словарь;

- для последовательности первых символов непрочитанной части ищется наиболее длинное совпадение в прочитанной части, если совпадение найдено, то составляется

комбинация {смещение, длина}, где "смещение" указывает, на сколько символов надо сместиться назад от текущей позиции по словарю, чтобы найти совпадение, а "длина" — это длина совпадения, т.е. число первых символов в последовательности символов непрочитанной части;

- если комбинация {смещение, длина} короче совпадения, то она записывается в выходной (закодированный) массив, а текущая позиция перемещается вперед по непрочитанной части на длину совпадающей части);

- если совпадение не обнажено или оно короче комбинации {смещение, длина}, то в выходной (закодированный) массив копируется эта последовательность символов, а текущая граница считывания перемещается вперед по непрочитанной части на длину этой последовательности и анализ повторяется.

На пример: фраза "КОЛОКОЛ_ОКОЛО_КОЛОКОЛЬНИ" закодируется этим алгоритмом таким образом: "КОЛО(-4,3)_(-5,4)О_(-14,7)ЬНИ".

Схема алгоритма LZ78, в самом общем виде, может быть представлена следующим образом:

- алгоритм хранит специальный словарь повторяющихся цепочек символов, в словаре каждой цепочке соответствует короткий код.

- для цепочки первых символов непрочитанной части ищется наиболее длинное совпадение в словаре. Код совпадения записывается в выходной массив, туда же заносится первый несовпавший символ, и текущая позиция перемещается вперед на длину совпадения плюс один символ;

- в словарь добавляется новое слово: "совпадение" + "несовпавший символ", и процесс повторяется до тех пор, пока не будет сжат весь входной массив.

Алгоритмы Лемпеля - Зива тем лучше сжимают данные, чем больше размер входного массива данных. Характерной особенностью обратных алгоритмов LZ77 и LZ78 является то, что, кроме самих сжатых данных, никакой дополнительной информации им не требуется. Начав работать, эти алгоритмы по уже распакованной части восстанавливают информацию, необходимую для распаковки следующих частей сжатых данных. Для сравнения: в алгоритме Хаффмана вместе со сжатыми данными требуется сохранять дерево Хаффмана, иначе распаковка будет невозможна.

В настоящее время существует достаточно много различных модификаций алгоритмов LZ77 и LZ78. Эти алгоритмы достаточно быстры и эффективны, и сейчас занимают лидирующее место среди используемых на практике алгоритмов сжатия. Обобщенно их часто называют методами сжатия со словарем.

При архивации степень сжатия файлов сильно зависит от степени исходной избыточности хранящихся в них данных, которая определяется их типом. К примеру, степень избыточности у видеоданных обычно в несколько раз больше, чем у графических, а степень избыточности графических данных в несколько раз больше, чем текстовых. На практике это означает, что изображения, будучи сжатыми и помещенными в архив, как правило, уменьшаются в размере значительно сильнее, чем текстовые документы. А рисунки в формате JPEG и графические файлы, типа TIF и GIF, уже заранее сжаты, поэтому даже самый лучший архиватор для них малоэффективен. Также крайне незначительно сжимаются исполняемые файлы, программы и архивы. Поэтому одни файлы могут быть сжаты в десятки раз, сжатие же других может и вовсе не уменьшить занимаемый ими объём.

Выбирая инструмент для сжатия данных, следует учитывать как минимум два фактора: эффективность, т. е. оптимальное соотношение между экономией дискового пространства и производительностью работы, и совместимость, т. е. возможность обмена данными с другими пользователями.

Совместимость является важным и значимым, фактором, так как по достигаемой степени сжатия, используемые форматы и инструменты сжатия, различаются на проценты, а высокая вычислительная мощность современных компьюте ров делает время обработки архивов не столь существенным показателем. Поэтому при выборе программы-архиватора важнейшим критерием становится ее способность "воспринимать" наиболее распространенные архивные форматы.

2.3. Архиваторы и архивные форматы.

Архиваторы – это программы, реализующие определённые методы сжатия. Использование методов сжатия весьма полезно при хранении и пересылке различных видов данных и крайне необходимо при работе с графическими, аудио и видео данными.

Исторически первым широкое признание получили универсальные архиваторы, которые способны были работать с различными типами данных. Одним из первых был универсальный архиватор Zip. Со временем были разработаны и другие универсальные программы-архиваторы: **RAR, ARJ, ACE, TAR, LHA** и т. д. В операционной системе Windows наиболее широко используются архиваторы **WinZip** и более поздний **WinRAR,** который имеет более удобный и интуитивно понятный интерфейс, мощную и гибкую систему архивации файлов, высокую скорость работы и который более плотно сжимает

файлы. Оба эти архиватора обеспечивают совместимость с большим числом архивных форматов.

В отличие от универсальных архиваторов, которые могут использоваться для записи различного рода сообщений (в основном текстовых), существуют и специализированные виды архиваторов (и соответствующих им форматов), специализирующихся на сжатии данных какого-либо конкретного вида и тем самым, для этого вида данных, достигая большей эффективности по сравнению с универсальными архиваторами. Одним из наиболее распространённых специализированных видов архиваторов являются архиваторы предназначенные для хранения графической информации, таких как различного рода изображения, фотографии или рисунки. Это вызвано тем, что повышение качества цифровых изображений требует увеличения пространственной частоты их дискретизации, которая увеличивает информационную емкость изображений, что затрудняет их хранение и обработку и вынуждает проводить их сжатие. Форматы, используемые для хранения графической информации, называются графическими форматами.

Как известно, любые изображения при компьютерной обработке представляются в цифровом виде и хранятся в том или ином графическом формате. Все графические форматы делятся на три группы: растровые, векторные, и комплексные (такие как PDF). Каждый из них имеет свои особенности, и выбор правильного формата во многом определяет качество работ при обработке и хранении изображений.

Растровые форматы используются при кодировании растровых изображений, которые получают путём разбиения исходного изображения регулярной квадратной сеткой на элементарные участки (пикселы, элементы разложения) такого размера, что характеристики изображения в них считают постоянными (интегральными). Поэтому растровое изображение можно представить в виде прямоугольной цифровой матрицы, элементами которой являются регулярно расположенные пиксели с соответствующими значениями каких-либо характеристик изображения.

Растровые графические файлы содержат значение избранной оптической характеристики каждого пиксела изображения и поэтому так же представляют собой прямоугольную цифровую матрицу, элементами которой являются численные значения характеристик пикселов. В растровом формате хранятся фотографии, рисунки и обои рабочего стола компьютера. Видеоизображения также являются последовательностью растровых изображений. Наиболее распространенными графическими форматами являются GIF, PNG и JPEG, использующиеся в Интернете, а также BMP (стандартный

формат ОС Windows) и TIFF, применяющийся при хранении отсканированных изображений и использующийся в полиграфии.

Алгоритмы сжатия растровых графических данных без потерь могут основываться на методе, заключающемся в том, что вместо последовательности одинаковых по значению избранной оптической характеристики пикселей в строке изображения записывается значение избранной характеристики и количество его повторений. Такой подход используется при хранении изображений в формате BMP.

Для сложных изображений такой метод малоэффективен, поэтому применяют другие методы. Например, во время обработки создают специальный словарь уже встречавшихся совокупности пикселей. При кодировании очередной последовательности пикселей эта последовательность заменяют на номер соответствующей последовательности по словарю, причем номера часто встречающихся последовательностей имеют более короткие номера, чем редко встречающихся. Этот способ активно применяется при сжатии самых разных данных, в том числе и графических. Такой способ сжатия применяется в графическом формате TIFF, в популярном формате GIF. Аналогичные методы применяются и в современном формате PNG, разработанном специально для применения в сетевых приложениях.

Векторные изображения представляют собой описание исходного изображения на основе комбинаций стандартных геометрических фигур: прямых, окружностей и т.д., поэтому файлы векторных графических форматов содержат не описание пикселов, как у растровых форматов, а формулы, описывающие координаты кривых. Например, прямая линия представляется координатами двух точек, а окружность — координатами центра и величиной радиуса. Это позволяет получать небольшой размер сжатых файлов.

Векторные графические форматы используют для передачи схем и рисунков, состоящих из набора линий, описываемых математическими формулами. В векторных форматах сохраняются логотипы, схематические рисунки, текст, предназначенный для вывода на печать, и другие подобные объекты.

Стандартными средствами преобразовать фотоизображение в векторный формат практически невозможно, для этого требуются специальные программы - конверторы. Наиболее распространенные расширения векторных файлов — AI для пакета Adobe Illustrator, CDR для пакета CorelDRAW и WMF (еще один «стандартный» формат Windows).

Комплексные графические форматы специально созданы с целью устранения проблем с отображением информации в файлах при переходе на компьютер с иной

операционной системой или другой версией операционной системы. Примером такого формата может служить формат PDF (Portable Document Format).

Преимущества этого формата заключаются в следующем:

1. документ, сохраненный в формате PDF, одинаково выглядит в любой операционной системе: Windows, Linux и др.;

2. формат PDF использует эффективный алгоритмы сжатия, например, если объем файла Word, содержащего пару картинок, около мегабайта, то объём точно такого же по содержанию файла PDF равен 300-400 Кбайт.

3. формат PDF содержит в себя всевозможные используемые в документе шрифты.

4. в формат PDF может быть преобразован любой электронный документ. А для прочтения и печати файла PDF используется бесплатная программа (Acrobat Reader).

2.4. Обзор графических растровых форматов.

Формат BMP (Windows Device Independent Bitmap) — это один из старейших форматов, который широко используется в ОС Windows. Основная область применения формата BMP — хранение файлов, использующихся внутри операционной системы (например, обоев для рабочего стола или иконок программ). Файл, сохраненный в этом формате, совместим с любым графическим редактором. Файлы, сохраненные в формате BMP, могут содержать как 256 цветов, так и 16 700 000 оттенков. BMP - самый простой графический растровый формат. Записанный в нем файл представляет собой массив данных, содержащий информацию о цвете каждого пиксела и служебную информацию об объеме и имени файла. Основной недостаток формата, ограничивающий его применение, — большой размер BMP-файлов, что вызывает трудности при их хранении и передаче графических файлов в формате BMP по сети.

Формат GIF. Как и формат BMP, формат GIF (Graphic Interchange Format) является одним из старейших форматов.

Он был разработан для передачи растровых изображений по сетям и позволяет эффективно сжимать изображения, что заметно уменьшает их размер, особенно если в них есть большие участки, закрашенные одним цветом.

Область применения формата GIF — это изображения с резкими цветовыми переходами и бизнес-графика (логотипы, кнопки, элементы оформления и т. п.). А для

изображений, где важно постепенное изменение оттенков (например, для фотографий), данный формат подходит меньше.

Полезная особенность формата — чересстрочная развертка, что позволяет увидеть и оценить изображение еще до завершения загрузки, что особенно выгодно при работе с файлами больших размеров.

Главный недостаток формата GIF заключается в том, что у него небольшой по современным меркам объём цветовой палитры, только 256 цветов (8 бит), что явно недостаточно для требований большинства современных задач, в частности, для фотоизображений. Вторая причина ограничения сферы применения формата GIF заключается в особенностях метода сжатия информации, заключающаяся в том, что данные об изображении записываются построчно и каждая строка обрабатывается отдельно. Вследствие этого сжатие изображений, содержащих мало однотонных областей, неэффективно, и кроме того результат сжатия зависит от расположения и углов поворота объектов.

Формат JPEG. Формат JPEG (Joint Photographic Expert Group - объединенная группа экспертов-фотографов) является одним из наиболее популярных форматов.

Использование простейших алгоритмов сжатия в ряде случаев не эффективно т. к. в реальных изображениях практически не встречаются точно повторяющиеся достаточно длинные последовательности пикселов.

Для этих случаев и был разработан специальный метод сжатия и соответствующий ему графический формат, получивший название JPEG. Метод сжатия JPEG применяется для сжатия статических изображений. Сжатие изображений производится с потерей качества и основано на учёте особенностей зрительного аппарата человека, в частности, того, что человеческий глаз инерционен и более чувствителен к яркости, чем к цвету. При этом часть малозаметных для глаза деталей изображения опускается, и, следовательно, восстановленное после сжатия изображение не полностью соответствует оригиналу. Алгоритм позволяет изначально задавать степень сжатия и таким образом устанавливать компромисс между качеством и объемом данных сжатого изображения, что позволяет добиться довольно большой степени сжатия данных, по сравнению со сжатием, производимым без потерь. В результате этого объём файлов в формате JPEG в десятки или сотни раз меньше, чем объём файлов в формате BMP. Естественно, чем выше уровень сжатия, тем ниже качество. При использовании высоких степеней сжатия изображение существенно искажается: становится заметно разделение на квадраты и

изменение частот в них, появляются размытость фрагментов изображения вокруг четко очерченных объектов. Впоследствии формат JPEG лег в основу стандарта сжатия видео MPEG.

Область применения формата JPEG, в основном, хранение фотоизображений и других, похожих на них изображений, т.к. достаточно сложный алгоритм сжатия графической информации позволяет обрабатывать фотографические изображения с большой эффективностью и достаточно высоким качеством. Алгоритм JPEG — один из базовых алгоритмов сжатия изображений. Его широкое распространение позволило резко расширить возможности и сферу применения цифровых методов обработки изображения. Несмотря на то, что существовали и существуют методы, обеспечивающие более высокое качество и степень сжатия, этот алгоритм получил широкое распространение за счет низких аппаратных требований и высокой скорости работы.

Важно отметить, что при обработке цифрового изображения в графическом редакторе, сохранять его в формате JPEG нужно лишь после выполнения всех действий по редактированию. В противном случае с каждым сохранением его качество будет ухудшаться, что, в конце концов, приведет к существенному искажению изображения.

Формат PNG (Portable Network Graphics) обладает рядом достоинств, выгодно отличающих его от других графических форматов. Самое главное из них — предварительная фильтрация обрабатываемых данных (предварительное преобразование данных в вид, наиболее приемлемый для обработки), что позволяет воспринимать различные видоизменения в изображениях и достигать высокого коэффициента сжатия практически для любого рода изображений.

Формат PNG обладает еще несколькими положительными отличиями по сравнению с форматами JPEG и GIF. Во-первых, существенно увеличен объём цветовой палитры (до 24 бит), во-вторых, он обладает модифицированным вариантом чересстрочной развертки, а в-третьих, встроенной в формат гамма-коррекцией.

Гамма – это соотношение между цифровым цветным изображением и реально наблюдаемыми на мониторе цветами. Необходимость гамма-коррекции ощущается при переносе графического файла с одного компьютера на другой. При этом фото и цветовые характеристики изображения могут отличатся от оригинала. Подобный эффект ярко проявляется тогда, когда на компьютерах установлены различные ОС (MS, Windows или Linux) или когда машины построены на базе разных платформ (например, PC или Macintosh).

Встроенная в формат PNG гамма-коррекция записываются в видеофайл данные о

настройках дисплея, видеоплаты и ПО (информация о гамме), и при переносе этого видеофайла на другой компьютер вид PNG-изображения соответствующим образом корректируются, чтобы фото и цветовые характеристики изображения не изменились.

Формат PNG чрезвычайно удобен для создания различных элементов оформления вебстраниц. Пожалуй, единственный тип изображений на вебстраницах, где формат PNG менее эффективен чем формат GIF — это небольшие кнопки и «смайлики» размером до 700—800 байт в формате GIF. Меньшая эффективность формата PNG объясняется тем, что, за счет служебной информации и описания палитры, такие файлы в формате PNG занимают на 10—30% больше места, чем в формате GIF.

Следует также отметить, что формат PNG не поддерживает анимированные картинки.

Формат TIFF (Tagged Image File Format) - наиболее универсальный формат, и он чаще всего используется в тех случаях, когда необходимо сохранить качество изображения, следовательно, данный формат производит сжатие изображений без потери качества. Представление изображения в формате TIFF гарантирует сохранность его первоначального качества. Но за качество приходится расплачиваться достаточно большим объемом файлов. Поэтому TIFF — это наиболее универсальный формат, обеспечивающий среднюю степень сжатия файла.

2.5. Сравнительная характеристика графических форматов

У всех описанных графических форматов сходное предназначение: добиться уменьшения размера графического файла, чтобы его было удобно хранить и передавать через сеть. Но используемые в них алгоритмы сжатия различаются, что и привело к тому, что сферы их использования различны и каждый из них обладает оптимальными областями применения.

Большинство алгоритмов сжатия рассчитаны на один вполне определенный тип изображений, и чем больше структура изображения отличается от эталонной структуры, тем ниже эффективность алгоритмов сжатия.

Результаты сравнительного анализа основных графических форматов показывают следующее. Формат JPEG предназначен для сжатия изображений с плавными цветовыми переходами, поэтому наиболее целесообразно применение этого формата при хранении изображений и, в частности, фотоизображений. Формат JPEG обеспечивает отличное

качество изображений при малых размерах файла. Для других целей этот формат мало пригоден. Однако, если в дальнейшем предполагается обработка изображений средствами Photoshop, необходимо их представлять в формателTIFF. В этом формате удобно хранить скриншоты, различные схемы и чертежи, поскольку при сохранении изображения в этом формате не происходит потерь качества, в отличие от их сохранения в формате JPEG, в котором неизбежно появятся искажения из-за сжатия с потерями. Именно формат TIFF используется в издательствах. К тому же этот формат поддерживают практически все платформы, как PC, так и Macintosh.

Весьма популярен формат GIF, который предназначен для сжатия изображений с большим количеством однотонных областей. Для нефотографических изображений формат GIF используется в подавляющем числе случаев. Однако, хранить в этом формате фотоизображения нецелесообразно из-за некорректной цветопередачи, вследствие использования в формате GIF цветовой палитры малого объёма (8 бит). Интересно так же то, что, в отличии от других графических форматов, формат GIF позволяет создавать анимированные изображения. Однако следует отметить, что его алгоритм сжатия устарел, и он существенно уступает по качеству сжатия практически всех видов изображений более новому формату PNG.

Формат PNG обладает более эффективным алгоритмом сжатия, большим объёмом цветовой палитры и может использоваться, например, для хранения промежуточных версий подлежащих редактированию фотоизображений, в том случае, когда BMP-файлы занимают слишком много места, а каждое последующее сохранение в JPEG приводит к потере качества. Кроме этого, формат PNG обладает эффективной гамма-коррекцией.

2.6. Форматы сжатия звуковых и видеофайлов

Звуковые и видео сообщения – это непрерывные сообщения, представленные в виде динамических сигналов, этим они в корне отличаются от изображений, представляемыми статическими сигналами. Динамические сигналы характеризуются скоростью их изменения, т. е. их частотным спектром, поэтому динамический сигнал может быть представлен суммой гармонических колебаний, взятых с присущими каждому из них амплитудой, частотой и фазой.

После преобразования в цифровую форму путём дискретизации по времени и квантования по уровню, сигнал представляет собой динамическую последовательность его мгновенных значений, выраженных в цифровой (формализованной) форме.

Информацию, представленную в формализованном виде, позволяющем осуществлять ее обработку с помощью технических средств, называют данными, а последовательность мгновенных значений непрерывных сообщений называют потоком данных.

Основной проблемой при обработке различного рода потоковых данных является их объем. Известно, что точность оцифрованного потока данных определяется частотой дискретизации и числом уровней квантования, а чем больше эти величины — тем больше объем получаемых данных.

Параметром, характеризующим поток данных, влияющим на качество передачи непрерывного сообщения, является величина битрейта (bitrate). *Битрейт* – это скорость создания или передачи информации, т.е. количество информации (в битах), используемое для кодирования (хранения) одной секунды звукового или видео сообщения. Эта характеристика важна для расчёта полной пропускной способности канала, Чаще всего битрейт звуковых и видео сообщений измеряют в килобитах в секунду. Численно битрейт (поток данных) чаще всего характеризуется количеством информации (в битах) приходящемся на одну секунду длительности сообщения, которое получается после его сжатия.

Для передачи и хранения звуковых сообщений используют *аудиофайлы* — компьютерные файлы, содержащие информацию об амплитуде и частоте звуковых колебаний, сохранённую для дальнейшего воспроизведения на компьютере или проигрывателе. Формат представления звуковых данных, используемый при цифровой звукозаписи, а также для дальнейшего хранения записанного материала на компьютере и других электронных носителях информации, часто называют *цифровым аудиоформатом* или *форматом аудиофайла*.

Формат аудиофайла определяет структуру и особенности представления звуковых данных при хранении на запоминающем устройстве ПК.

Существенной особенностью звуковых и видео данных является их очень большой объем и небольшое различие между двумя соседними отсчётами или кадрами из-за высокой частоты дискретизации, что порождает существенную избыточность звуковых и видео данных. Поэтому хранение и передача видео и аудио данных, требует применения различных методов сжатия для устранения их избыточности. Аппаратно для кодирования, хранения и передачи таких данных пользуются специальными программами – кодеками, при помощи которых и производится сжатие потока данных.

Кодеки – это отдельные специализированные программы, вызываемые проигрывателями аудио или видеофайлов, которые служат средствами для кодирования/декодирования (сжатия/распаковки) потока данных или средствами

сохранения – для его сжатия, в частности для звуковых и видео файлов. Качество звука или изображения во многом зависит от используемого кодека. Кодек отмечается в начале файла (или сетевого потока), и его наличие — важное условие работы с данными. Многие кодеки не поставляются вместе с операционной системой, а устанавливаются дополнительно. Для удобства их часто собирают в пакеты (codec-pack).

При разработке методов сжатия аудио и видео данных исходят из того, что сообщения такого вида, как известно, представляет собой непрерывное сообщение с заданным частотным спектром. Проведенные в конце XX века исследования психофизиологических характеристик зрения и слуха обнаружили ряд особенностей человеческого восприятия, которые позволили сделать заключение, что для аудио и видео сообщений абсолютно точное восстановление их исходного вида вовсе не обязательно. Таким образом возможно существенное увеличение степени их сжатия.

Действительно, ограниченный диапазон частот, воспринимаемых органами человека, позволяет пренебречь некоторыми составляющими аудиосообщений (в частности, высокочастотными составляющими), проведя их высокочастотную фильтрацию, что, соответственно, позволяет уменьшить частоту дискретизации, а на заключительном этапе сжатия применить алгоритм Хаффмана.

Фильтрация ведёт к потерям в качестве звуковых и видео сообщений. Однако, если потери не превышают некоего уровня, заметного ухудшения качества не происходит.

Для разработки и стандартизации эффективных методов сжатия аудио- и видеоинформации на рубеже 1980— 1990-х годов в составе Международной организации стандартов (ISO) были созданы Группа экспертов по фотографическим изображениям (Joint Photographic Experts Group, сокращённо JPEG) и Группа экспертов по видеоизображениям (Motion Picture Experts Group, сокр. MPEG).

К середине 1990-х годов этими группами были разработаны специальные высокоэффективные методы сжатия аудио и видео информации, учитывающие особенности человеческого слуха и зрения. Характерной чертой этих методов является возможность регулируемого удаления из сообщений маловажной для человеческого восприятия информации. Поэтому такие алгоритмы сжатия обобщенно называют алгоритмами с регулируемой потерей информации. За счет удаления части информации удается добиться очень большой степени сжатия данных при субъективно незначительной потере качества аудио- и видеоданных. Алгоритмы с регулируемой потерей информации не универсальны, они не могут использоваться для сжатия любых данных, поскольку полное восстановление исходной информации невозможно. Эти методы сложны в реализации, в них используется достаточно сложный математический аппарат.

Непосредственно для сжатия специфических потоковых данных представленных в виде аудио и видео сообщений был разработан и принят в 1992 году стандарт MPEG-1. Этот стандарт включал в себя метод сжатия видеосигнала в поток с битрейтом до 1,5 Мбит/с и сжатие аудиосигнала в поток с битрейтом 64, 128 или 192 Кбит/с на канал, а также алгоритмы синхронизации. Принятый стандарт описывал не алгоритмы, а формат сжатого потока данных. Это позволило в дальнейшем на его основе разработать множество конкретных реализаций алгоритмов сжатия.

Наиболее известными методами сжатия с регулируемой потерей информации являются:

• JPEG — метод сжатия графических данных;

• MP3 — метод сжатия звуковых данных;

• MPEG — группа методов сжатия видеоданных.

2.7. Звуковые форматы сжатия.

При кодировании звукового сигнала основным параметром, влияющим на качество результата, является величина битрейта. Эта характеристика важна для расчёта полной пропускной способности канала.

При использовании максимального значения этого параметра получают звук, наиболее близкий к оригиналу. Примерно до 2000 года широко использовалось значение битрейта равное 128 (Кбит/с), а затем, с возрастанием пропускной способности каналов связи, наиболее распространенным стал битрейт равный 192 (Кбит/с).

Существует три режима сжатия потоковых данных: с постоянным битрейтом, с усреднённым битрейтом и с переменным битрейтом.

Из них наиболее интересен режим сжатия потоковых данных с переменным битрейтом, в этом режиме значение битрейта меняется в зависимости от частотного характера сигнала (функция VBR). Суть этого режима сжатия заключается в следующем. В части звукового сообщения, содержащего высокочастотные составляющие (сжать которые труднее всего), разумно использовать для их кодирования высокое значение битрейта, а при отсутствии высокочастотных составляющих - существенно меньшее значение битрейта.

Выделяют три группы форматов аудиофайлов:

- аудиоформаты без сжатия, такие как WAV, AIFF;
- аудиоформаты со сжатием без потерь (APE, FLAC);
- аудиоформаты со сжатием с потерями (MP3, OGG).

Существует два основных стандарта, которые используют для кодирования, сжатия и хранения звука (и в частности музыки): MP3 и WMA. До сих пор ни один формат, отличный от WMA и MP3, не смог получить такого же широкого распространения. Это объясняется двумя причинам: либо кодек, работающий в формате отличном от WMA и MP3, не очень хорошо сжимает аудиофайлы, либо программное обеспечение для его использования полностью платное.

Кодек WMA разработан фирмой Microsoft как стандарт хранения сжатой аудиоинформации в операционной системе Windows. Стандарт WMA является закрытым для использования другими разработчиками. Преимущество этого стандарта в том, что при таком же качестве звука, что и у стандарта MP3, этот кодек позволяет получить меньший размер файла. Это достигается за счет использования более низкого значения параметров битрейта, чем у MP3-файлов.

В последних версиях кодека значительно переработана психоакустическая модель кодирования аудиоинформации, с целью повышения качества сжатия.

Алгоритм MP3 (точное название — MPEG-1 Layer 3) представляет собой наиболее популярную реализацию стандарта MPEG-1 и использует метод сжатия с потерей информации (с учётом особенностей слухового восприятия и с наименьшими потерями качества звука). Основной особенностью этого стандарта является то, что в нём сделан акцент на улучшении алгоритма специальной психоакустической модели. Принцип его действия заключается в том, что из звукового файла удаляются те частоты, которые человеческое ухо воспринимать не в состоянии, т.е. реализуется метод сжатия с потерей информации, но с учётом особенностей слухового восприятия и с наименьшими потерями качества звука. Помимо сжимаемого звукового фрагмента, алгоритму передается также желаемый *битрейт* — количество информации, используемых для кодирования одной секунды звука. Этот параметр неявно регулирует долю информации, которая будет удалена при сжатии.

Сжатие осуществляется в несколько этапов. На первом этапе звуковой фрагмент разбивается на небольшие участки — фреймы (frames), и в каждом фрейме звук разлагается на совокупность гармонических составляющих с разными частотами, фазами и амплитудами, присущими каждой из них, называемых гармониками. Затем начинается психоакустическая обработка — удаление маловажной для человеческого восприятия звуковой информации, при этом учитываются различные особенности слуха. Заданное значение битрейта определяет количество удаляемой информации, исходя из того какие психоакустические эффекты будут учитываться при сжатии. На последнем этапе данные

сжимаются методами эффективного кодирования (алгоритм Хаффмана).

Алгоритм MP3 позволяет сжимать звуковые файлы в несколько раз. Даже при самом большом значении битрейта (320 кбит/сек.) формат MP3 обеспечивает четырехкратное сжатие аудиоинформации по сравнению с компакт-дисками (формат CD), при таком же субъективном качестве звука. Формат MP3 стал основным форматом для распространения музыкальных файлов через Интернет.

2.8. Форматы сжатия видеофайлов.

Видеофайлы — это компьютерные файлы, в которых информация содержится в последовательности цифровых изобра жений (видеоряд) Видеофайлы используются для передачи и хранения видеоинформации. Специфической особенностью видеофайлов является чрезвычайно большой объёмом содержащейся в них информации, поэтому их использование требует их эффективного сжатия. Для этого используют различные методы сжатия, отличающихся конкретной реализацией алгоритмов, но, несмотря на большое разнообразие реализаций, в основе всех этих алгоритмов лежат несколько общих подходов к сжатию видеофайлов.

Одним из базовых подходов является метод опорного кадра, основой которого является алгоритм сжатия изображений JPEG. Этот метод основан на том, что при переходе от одного кадра к последуюшему, как правило, большая часть изображения остаётся неизменной. Поэтому можно сохранять не весь следующий кадр целиком, а только изменения по сравнению с предыдущим кадром. Например, в фильме есть сцена беседы героев в комнате. При этом от кадра к кадру меняются только выражения лиц, а большая часть изображения неподвижна. Сохранив первый кадр сцены и отличия остальных ее кадров от первого, можно получить сжатый видеофайл с большой степенью сжатия.

При реализации этого подхода рассматриваются три вида кадров: ключевой (опорный) кадр, сохраняемый в потоке полностью (intrapictures), кадры, сжатые со ссылкой на предыдущее изображение (predicted), и кадры, ссылающиеся на два кадра (bidirection).

В случае использования ссылок на кадры записывается и сжимается не весь кадр, а только его изменившиеся части. Двунаправленные и ключевые кадры позволяют сократить накапливающиеся ошибки. В упрощённом виде алгоритм сжатия состоит в том, что в процессе сжатия изображение каждого кадра разбивается регулярной сеткой на микроблоки, которые представляют собой отдельные квадраты, состоящие из 16 пикселей. Полученные микроблоки сравнивают с соответствующими микроблоками

опорного кадра. В случае их отличия микроблок текущего кадра запоминается, в противном случае – запоминается только номер соответствующего микроблока опорного кадра. Отсюда вытекает ограничение: размеры кадра должны быть кратны 16.

Еще один способ сжатия исходных видеофайлов основан на том факте, что, из-за особенностей зрения, человеческий глаз не успевает детально рассмотреть быстро сменяющиеся участки изображения, поэтому их можно кодировать и производить их сжатие с более низким качеством, намного ниже качества сжатия статичных участков изображения.

При этом статическое изображение и быстроменяющееся изображение можно разделить и применять к ним различные алгоритмы сжатия и только после декодирования соединить их в одно изображение.

Как уже упоминалось, стандарт MPEG — это целое семейство методов сжатия видеоданных. На основе стандарта MPEG была **р**азработана и принята группа стандартов сжатия видеоданных, получившая название MPEG-4. MPEG-4 - это название группы экспертов (Motion Picture Experts Group), по аббревиатуре которой и сложились современные названия этих стандартов, а также расширения многих других видеофайлов. Эти стандарты предназначенных для работы с потоком данных (видеопотоком), скорость которого может изменяться от 3 до 10 Мбит/с. Кроме того, формат MPEG позволяет сохранять в одном файле несколько так называемых "потоков данных". Так, в основном потоке можно сохранить фильм, в другом — логотип, в третьем — субтитры и т.д. Потоки данных накладываются друг на друга только при воспроизведении. Такой способ позволяет, например, хранить субтитры в виде текста вместо изображений букв, логотип сохранить всего один раз, а не в каждом кадре, и т.п.

Поскольку в стандарте MPEG-4 нет конкретного описания алгоритмов, существует большое количество различных их реализаций. Эти разновидности формата MPEG отличаются друг от друга своими возможностями, максимальной степенью сжатия, методиками расстановки ключевых (опорных) кадров и другим. Поэтому результаты их работы зачастую сильно различаются по качеству изображения, хотя несовершенный человеческий глаз не всегда может уловить визуальную разницу между файлами, сжатыми по тому или иному алгоритму.

В настоящее время большинство видеофайлов сжаты с использованием стандарта MPEG-4, хотя его реализации могут быть представлены в различных видеоформатах, что объясняется многообразием алгоритмов сжатия цифрового видео. Многие из них принадлежат разным производителям и основаны на различных принципах. Среди

наиболее распространённых видеоформатов можно отметить следующие:

• MPEG-1 — использовался в первых Video CD (VCD-I);

• MPEG-2 — используется в DVD и Super Video CD (SVCD, VCD-II);

• MJPEG — формат сжатия видео, в котором каждый кадр сжимается по методу JPEG;

• MPEG-4 — один из самых популярных и эффективных форматов сжатия видео;

• DivX, XviD — улучшенные модификации формата MPEG-4.

Формат компрессии DivX (технология компрессии от фирмы DivXNetworks) – представляет собой модификацию формата MPEG-4, отличается высокой степенью сжатия с приемлемым качеством.

Формат компрессии RealVideo (собственный формат компании RealNetworks) имеет очень высокую степень сжатия видео и используется для прямой трансляции в Интернете, т.к. он позволяет сжать файлы до небольшим размера, хотя и сравнительно низкого качества.

Формат компрессии Windows Media (собственный формат компании Microsoft), формат аналогичен формату MPEG-4.

Файлы с расширением AVI отличаются тем, что не имеют постоянных параметров сжатия, так как могут быть без компрессии и сжатые, например, по формату DivX, которые также имеют расширение AVI.

Операционная система Windows может работать с видеофайлами различных форматов, благодаря наличию современного программного видеоплеера. В ОС Windows, по умолчанию, это Windows Media Player, причем в комплект его поставки уже входят несколько встроенных кодеков, которые обеспечивают воспроизведение почти всех широко распространённых аудио и видеофайлов.

Лабораторная работа

Эффективное кодирование неравновероятных символов источника дискретных сообщений.

Цель работы. Ознакомление с алгоритмами эффективного кодирования неравновероятных взаимно независимых символов источника дискретных сообщений, их особенностями, методами нахождения информационных характеристик кодов и оценки эффективности кодов по значениям их информационных характеристик.

Теоретическое обоснование

Для оценки эффективности кодов применяют следующие информационные характеристики.

Энтропия исходного (кодируемого) источника сообщений ($H_и$) определяется формулой Шеннона:

$$H_и = -\sum_{i=1}^{m} P_i \cdot \log_2 P_i \ , \qquad (1)$$

где P_i – вероятность появления i-го символа исходного источника сообщений;

m – объём алфавита исходного источника сообщений.

Средняя длина символов кода ($n_к$), которая определяется выражением:

$$n_к = \sum_{i=1}^{m} n_i \cdot p_i \ , \qquad (2)$$

где n_i –значность i– го кодового символа, соответствующего символу исходного алфавита m_i;

p_i – вероятность появления кодового символа m_i.

Длина эффективного равномерного кода т.е. наименьшее число элементов в кодовом символе эффективного равномерного кода ($l_{эфф.}$), как известно, может быть найдена из соотношения:

$$l_{эфф} = ceil(\frac{H_и}{H_1}) = ceil(\frac{\log_2 m}{\log_2 k}) = n_{рк}, \qquad (3)$$

где *ceil* – оператор нахождения ближайшего наибольшего целого;

k – основание числового кода;

$n_{рк}$ - число элементов в кодовом символе равномерного кода.

Избыточность кода ($R_к$) определяют из соотношения:

$$R_к = 1 - \frac{H_и}{n_к \cdot \log_2 k}, . \qquad (3)$$

где k – основание числового кода.

Энтропия элементов символов кода $(H_к)$ может быть легко найдена:

$$H_к = \frac{H_и}{n_н} \quad \text{(бит/элемент символа кода).} \qquad (4)$$

Содержание работы.

1. По номеру в списке группы (N) из Таблицы 1 выбрать закон распределения символов источника дискретных сообщений с объёмом алфавита $m=8$.

Таблица 1.

m_i	N=1	N=2	N=3	N=4	N=5	N=6	N=7	N=8	N=9	N=10
m_1	0.10	0.18	0.07	0.65	0.55	0.60	0.01	0.15	0.30	0/01
m_2	0.51	0.10	0.03	0.05	0.05	0.06	0.02	0.10	0.20	0.05
m_3	0.02	0.47	0.11	0.06	0.16	0.02	0.02	0.30	0.10	0.03
m_4	0.10	0.07	0.33	0.03	0.03	0.10	0.15	0.35	0.05	0.02
m_5	0.02	0.03	0.25	0.02	0.02	0.02	0.02	0.02	0.15	0.10
m_6	0.20	0.02	0.01	0.15	0.02	0.15	0.45	0.02	0.10	0.14
m_7	0.01	0.04	0.17	0.02	0.02	0.03	0.30	0.01	0.07	0.25
m_8	0.04	0.09	0.03	0.02	0.15	0.05	0.03	0.05	0.03	0.40

Произвести эффективное кодирование числовым двоичным кодом заданного источника дискретных сообщений по алгоритму Шеннона-Фено и по алгоритму Хафмена.

Рассчитать для построенных на основе этих алгоритмов неравномерных кодов следующие информационные характеристики:

а) среднюю длину неравномерного кода ($n_{нк}$);

б) избыточность неравномерного кода($R_{нк}$);

в) энтропию элементов символов полученного неравномерного кода ($H_{нк}$).

2. Произвести кодирование этого же источника дискретных сообщений равномерным двоичным цифровым кодом и вычислить для этого кода:

а) среднюю длину равномерного кода ($n_{рк}$);

б) избыточность равномерного кода($R_{рк}$);

в) энтропию элементов символов полученного равномерного кода($H_{рк}$).

Сравнить полученные результаты с соответствующими параметрами неравномерных

двоичных цифровых кодов.

3. Произвести эффективное кодирование двоичным числовым кодом соответствующего Вашему номеру (*N*) источника дискретных сообщений по алгоритму Хаффмена в программном средстве MathCad. Вычислить для этого кода:

а) среднюю длину неравномерного кода ($n_{нк}$);

б) избыточность неравномерного кода($R_{нк}$);

в) энтропию элементов символов полученного кода($H_{нк}$);

Сравнить полученные значения информационных характеристик с соответствующими результатами «ручного» кодирования и результатами кодирования источника дискретных сообщений равномерным двоичным цифровым кодом.

Приложение.

Пример программы, для эффективного кодирования двоичным числовым кодом источника дискретных сообщений по алгоритму Хаффмена в программном средстве MathCad.

Источник дискретных сообщений *X* с алфавитом x_i (i= 1,…, m) и с объёмом алфавита равном m = 9, имеет следующие вероятности появления символов, заданные вектор-строкой:

ORIGIN := 1

Px := (0.04 0.06 0.08 0.10 0.10 0.12 0.15 0.15 0.20),

где, например, $Px^{<3>} = 0.08$ и $Px^{<6>} = 0.12$.

Составляется матрица, в которой вероятности выписываются в первый (основной) столбец в порядке их убывания, т.е.

$Po := reverse\left(sort\left(Px^{T}\right)\right)$;

Po^{T} = (0.2 0.15 0.15 0.12 0.1 0.1 0.08 0.06 0.04) .

Две последних вероятности P1 объединяют в одну вспомогательную вероятность Ps1:

$P1 := \begin{pmatrix} Po_{N-1} \\ Po_{N} \end{pmatrix}$; $P1 = \begin{pmatrix} 0.06 \\ 0.04 \end{pmatrix}$; $Ps1 := \sum P1$; $Ps1 = 0.1$.

Вероятности

Px_1 := (0.20 0.15 0.15 0.12 0.10 0.10 0.08 0.1 0)

снова располагают в порядке их убывания в дополнительном столбце:

$Pd1 := reverse\left(sort\left(Px_1^{T}\right)\right)$;

$Pd1^{T}$ = (0.2 0.15 0.15 0.12 0.1 0.1 0.1 0.08 0) .

Две последне вероятности P2 объединяются в одну вспомогательную вероятность Ps2:

$$P2 := \begin{pmatrix} Pd1_{N-2} \\ Pd1_{N-1} \end{pmatrix};\ P2 = \begin{pmatrix} 0.1 \\ 0.08 \end{pmatrix}\ ;\ Ps2 := \sum P2\ ;\ Ps2 = 0.18 \quad .$$

Вероятности $Px_2 := (0.20 \quad 0.15 \quad 0.15 \quad 0.12 \quad 0.10 \quad 0.10 \quad 0.18 \quad 0 \quad 0)$

снова располагают в порядке их убывания в дополнительном столбце

$$Pd2 := \text{reverse}\left(\text{sort}\left(Px_2{}^T\right)\right);$$

$$Pd2^T = (0.2 \quad 0.18 \quad 0.15 \quad 0.15 \quad 0.12 \quad 0.1 \quad 0.1 \quad 0 \quad 0) \quad .$$

Две последние вероятности P3 объединяются в одну вспомогательную вероятность Ps3:

$$P3 := \begin{pmatrix} Pd2_{N-3} \\ Pd2_{N-2} \end{pmatrix};\ P3 = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix}\ ;\ Ps3 := \sum P3\ ;\ Ps3 = 0.2 \quad .$$

Вероятности $Px_3 := (0.20 \quad 0.18 \quad 0.15 \quad 0.15 \quad 0.12 \quad 0.20 \quad 0 \quad 0 \quad 0)$

снова располагают в порядке их убывания в дополнительном столбце:

$$Pd3 := \text{reverse}\left(\text{sort}\left(Px_3{}^T\right)\right);$$

$$Pd3^T = (0.2 \quad 0.2 \quad 0.18 \quad 0.15 \quad 0.15 \quad 0.12 \quad 0 \quad 0 \quad 0) \quad .$$

Две последние вероятности P4 объединяются в одну вспомогательную вероятность Ps4:

$$P4 := \begin{pmatrix} Pd3_{N-4} \\ Pd3_{N-3} \end{pmatrix};\ P4 = \begin{pmatrix} 0.15 \\ 0.12 \end{pmatrix}\ ;\ Ps4 := \sum P4\ ;\ Ps4 = 0.27 \quad .$$

Вероятности $Px_4 := (0.20 \quad 0.20 \quad 0.18 \quad 0.15 \quad 0.27 \quad 0 \quad 0 \quad 0 \quad 0)$

снова располагают в порядке их убывания в дополнительном столбце

$$Pd4 := \text{reverse}\left(\text{sort}\left(Px_4{}^T\right)\right);$$

$$Pd4^T = (0.27 \quad 0.2 \quad 0.2 \quad 0.18 \quad 0.15 \quad 0 \quad 0 \quad 0 \quad 0) \quad .$$

Две последние вероятности P5 объединяются в одну вспомогательную вероятность Ps5:

$$P5 := \begin{pmatrix} Pd4_{N-5} \\ Pd4_{N-4} \end{pmatrix};\ P5 = \begin{pmatrix} 0.18 \\ 0.15 \end{pmatrix}\ ;\ Ps5 := \sum P5\ ;\ Ps5 = 0.33 \quad .$$

Вероятности $Px_5 := (0.27 \quad 0.2 \quad 0.2 \quad 0.33 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$

снова располагают в порядке их убывания в дополнительном столбце

$$Pd5 := \text{reverse}\left(\text{sort}\left(Px_5{}^T\right)\right);$$

$$Pd5^T = (0.33 \quad 0.27 \quad 0.2 \quad 0.2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0) \quad .$$

Две последние вероятности P6 объединяются в одну вспомогательную вероятность Ps6:

$$P6 := \begin{pmatrix} Pd5_{N-6} \\ Pd5_{N-5} \end{pmatrix};\ P6 = \begin{pmatrix} 0.2 \\ 0.2 \end{pmatrix}\ ;\ Ps6 := \sum P6\ ;\ Ps6 = 0.4 \quad .$$

Вероятности $Px_6 := (0.33 \quad 0.27 \quad 0.4 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$

снова располагают в порядке их убывания в дополнительном столбце

$$Pd6 := \text{reverse}\left(\text{sort}\left(Px_6^{\,T}\right)\right);$$

$Pd6^T = (0.4 \quad 0.33 \quad 0.27 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$.

Две последние вероятности P7 объединяются в одну вспомогательную вероятность Ps7:

$$P7 := \begin{pmatrix} Pd6_{N-7} \\ Pd6_{N-6} \end{pmatrix};\ P7 = \begin{pmatrix} 0.33 \\ 0.27 \end{pmatrix}\ ;\ Ps7 := \sum P7\ ;\ Ps7 = 0.6 \quad .$$

Вероятности $Px_7 := (0.4 \quad 0.6 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$

снова располагают в порядке их убывания в дополнительном столбце

$$Pd7 := \text{reverse}\left(\text{sort}\left(Px_7^{\,T}\right)\right);$$

$Pd7^T = (0.6 \quad 0.4 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$.

Две последние вероятности P8 объединяются в одну вспомогательную вероятность Ps8:

$$P8 := \begin{pmatrix} Pd7_{N-8} \\ Pd7_{N-7} \end{pmatrix};\ P8 = \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix}\ ;\ Ps8 := \sum P8\ ;\ Ps8 = 1 \quad .$$

Вероятности $Px_8 := (0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$

снова располагают в порядке их убывания в дополнительном столбце

$$Pd8 := \text{reverse}\left(\text{sort}\left(Px_8^{\,T}\right)\right);$$

$Pd8^T = (1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$.

На этом при получении дополнительного столбца с вероятностью, равной единице, процесс заканчивается. Матрица M, на основе которой проводится кодирование, принимает вид:

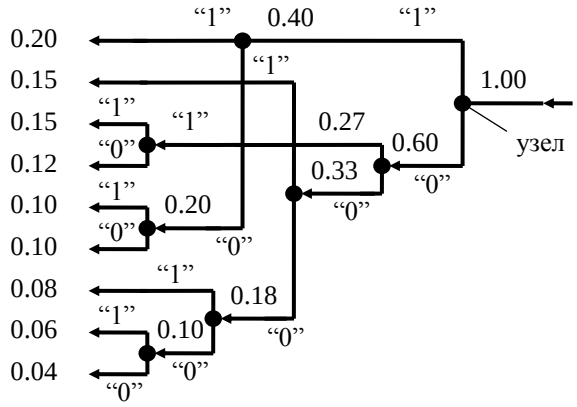
$$M_1 := \text{augment}(\text{augment}(Po, Pd1), \text{augment}(Pd2, Pd3));$$

$$M_2 := \text{augment}(\text{augment}(Pd4, Pd5), \text{augment}(Pd6, \text{augment}(Pd7, Pd8)));$$

$$M := \text{augment}\left(M_1, M_2\right);$$

$$M = \begin{bmatrix} 0.2 & 0.2 & 0.2 & 0.2 & 0.27 & 0.33 & 0.4 & 0.6 & 1 \\ 0.15 & 0.15 & 0.18 & 0.2 & 0.2 & 0.27 & 0.33 & 0.4 & 0 \\ 0.15 & 0.15 & 0.15 & 0.18 & 0.2 & 0.2 & 0.27 & 0 & 0 \\ 0.12 & 0.12 & 0.15 & 0.15 & 0.18 & 0.2 & 0 & 0 & 0 \\ 0.1 & 0.1 & 0.12 & 0.15 & 0.15 & 0 & 0 & 0 & 0 \\ 0.1 & 0.1 & 0.1 & 0.12 & 0 & 0 & 0 & 0 & 0 \\ 0.08 & 0.1 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.06 & 0.08 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.04 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

На основании данной таблицы строим кодовое дерево, ветви которого соответствуют вероятностям, согласно матрице М.



Каждой ветви дерева присваивается символ "1" при выходе из узла вверх и символ "0" при выходе из узла вниз. Движение по кодовому дереву из вершины с P=1.00 к символам, определяемым соответствующими вероятностями, формирует двоичные кодовые символы эффективного кода, приведенные в Таблица 2.

Таблица 2.

Символ исходного сообщения.	Вероятность символа исходного сообщения.	Символ двоичного числового кода.
x_1	0.04	00000
x_2	0.06	00001
x_3	0.08	0001
x_4	0.10	100
x_5	0.10	101
x_6	0.12	010
x_7	0.15	011
x_8	0.15	001

x_9	0.20	11

Согласно таблице кодирования, длину кодовых символов можно описать вектор-строкой:

$$n := (5 \quad 5 \quad 4 \quad 3 \quad 3 \quad 3 \quad 3 \quad 3 \quad 2).$$

Средняя длина кодового символа:

$$n_к = \sum_{i=1}^{m} n_i \cdot p_i \ .$$

Энтропия исходного источника сообщений:

$$H_и = -\sum_{i=1}^{m} (P_x{}^T)_i \cdot \log_2((P_x{}^T)_i, 2) \cdot,$$

Избыточность кода ($R_к$) определяют из соотношения:

$$R_к = 1 - \frac{H_и}{n_к \cdot \log_2 k}.$$

Энтропия элементов символов кода $(H_к)$ может быть легко найдена:

$$H_к = \frac{H_и}{n_к} \quad \text{(бит/элемент символа кода).}$$

Контрольные вопросы.

1. Какой вид кодирования называют эффективным и в чем его специфика?
2. Что такое избыточность кодов?
3. Какие коды называются равномерными?
4. На каких принципах основано построение эффективных кодов при неравновероятном появлении символов сообщения?
5. Принцип построения эффективного кода по алгоритму Шеннона-Фено.
6. Принцип построения эффективного кода по алгоритму Хафмена.

Литература.

1. Журкин И.Г., Шавенько Н.К. «Автоматизированная обработка данных дистанционного зондирования». Учебник для ВУЗов, Москва. ООО «Диона», 2014, 456 стр.
2. Шавенько Н.К Основы теории информации и кодирования. Учебное пособие. –М.: Изд. МИИГАиК, 2019. – 126 с.

Содержание.