Министерство образования и науки
Российской Федерации
федеральное государственное бюджетное
образовательное учреждение высшего образования

РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ

Е.А.Чернецова

Теория информации и кодирования

Практикум

САНКТ-ПЕТЕРБУРГ
РГГМУ
2021

УДК 621.391
ББК 22.18

Чернецова Е.А. Теория информации и кодирования. Практикум. – СПб.: РГГМУ, 2021. – 172 с.

В практикум по дисциплине ”Теория информации и кодирования” включены теоретические сведения, практические задачи и упражнения, которые охватывают все разделы дисциплины, читаемой на кафедре "Информационные технологии и системы безопасности" для студентов, обучающихся по специальности «Информационная безопасность телекоммуникационных систем». Содержащиеся в практикуме сведения теории, методические указания и рекомендации по выполнению практических работ позволяют использовать его в качестве дополнительного пособия для закрепления курса лекций.

Практикум предназначен для студентов гидрометеорологического университета и может быть полезным для всех желающих ознакомиться с основами теории информации и кодирования.

Содержание

1 Понятие информации

Информация – любая совокупность сигналов, воздействий или сведений, которые некоторая система воспринимает от окружающей среды (входная информация), выдает в окружающую среду (выходная информация), а также хранит её в себе (внутренняя, внутрисистемная информация). С точки зрения основных задач системы обработки сигналов, информация – это то, что является в системе объектом хранения, передачи и обработки[1].

Отсутствие информации о состоянии рассматриваемой системы также можно трактовать как неопределенность её состояния.

Соответственно, получаемая информация уточняет знание о состоянии системы, или, другими словами, устраняет (уменьшает) эту неопределенность. Отсюда следует, что сведения, не меняющие наших знаний о состоянии системы, информацией не являются.

Хотя понятие информации абстрактно, она проявляется всегда в материально-энергетической форме в виде сигнала, под которым понимается физический процесс — материальный носитель информации. Это фундаментальное свойство информации. Отметим, что информация всегда связана с кодом – формой ее представления (структурой сигнала).

Информация существует только в форме процесса.

Можно выделить информационные процессы восприятие, запоминание, воспроизведение, хранение, передача и обработкаинформации.

Информация является таковой только для получателя, способного ее получить, распознать и понять.

Информацию, предназначенную для обработки, называют данными. Их часто представляют в

формализованном виде, т.е. записанными в виде символов (чисел).

2 Предмет дисциплины «Теория информации и кодирования»

Теория информации – математическая теория, посвященная изучению и решению ряда фундаментальных вопросов, касающихся повышения эффективности функционирования системдля передачи данных.

Она рассматривает следующие задачи:

– измерение количества информации в ее потоке;

– анализ сигналов, как средств передачи сообщений;

–анализ информационных характеристик источников сообщений и каналов связи;

–исследование возможности кодирования и декодирование сообщений, обеспечивающих максимальную скорость передачи и безошибочность передачи сообщений по каналу связи, как при отсутствии, так и при наличии помех и другие.

При этом сообщения рассматриваются просто как последовательности знаков, без учета смыслового и прагматического (полезного) содержания, для которого пока не разработаны однозначные формализованные критерии. Такой подход имеет право на жизнь в силу того, что информация заключена в последовательности знаков и чем больше передано знаков, тем больше передано информации.

В широком смысле к теории информации относят все проблемы и задачи, связанные с информацией, ее восприятием, передачей, переработкой, хранением и использованием.

3 Понятия, относящиеся к системам связи

Под передачей информации будем понимать ее перенос на расстояние или во времени (хранение) посредством сигналов различной физической природы (механических, электромагнитных, оптических и др.).

Сообщение – форма представления информации. Различают непрерывные и дискретные сообщения. Непрерывные сообщения неразделимы на элементы и описываются непрерывными функциями времени, принимающими непрерывное множество значений. Их типичными примерами являются речь, музыка, сообщения датчиков – измерителей тока, температуры и т.п. Дискретные сообщения состоят из отдельных элементов (знаков)[2].

В простейшем случае это последовательность знаков (символов или первичных сигналов). Соответственно называют и источники информации. Например, если источник в каждый момент времени принимает одно состояние из некоторого набора состояний, то его называют дискретным источником сообщений. Каждому сообщению (состоянию) источника ставится в соответствие знак.

Под знаком понимают реальные различимые получателем материальные объекты. Ими могут быть символы (буквы, цифры, иероглифы и т.п.), предметы, электромагнитные импульсы и т.д. Так данный текст состоит из букв, сообщение азбукой Морзе – из точек, тире (коротких и длинных звуков, импульсов) и пауз, сообщение кипу состоит из узелков. Множество всех знаков называют алфавитом, а количество знаков – объемом (мощностью, глубиной) алфавита. Для двоичного кодирования алфавитом является набор знаков {0, 1}, а объем алфавита равен 2.

Конкретное представление информации называется кодированием в широком смысле. Так слова нашего языка тоже представляют собой коды. Слово «человек» – есть код, соответствующий реально существующему объекту. В технике чаще всего используется числовой двоичный код. Под кодированием в узком смысле или собственно кодированием понимают переход от одного алфавита к другому или от одного кода к другому.

Кодом называют, как само правило перехода, так и представление (образ) знака. Кодирование дискретного сообщения заключается в сопоставлении каждому исходному знаку своего набора символов кода – кодового слова (кодовой комбинации), часто называемого просто кодом знака. Техническое устройство, осуществляющее кодирование, называется кодером.

При передаче каждому сообщению сопоставляется сигнал, под которым понимается физический процесс, являющийся материальным носителем информации. Сообщение «записывается» в сигнал путем изменения (модуляции) в соответствии с передаваемой информацией одного или нескольких физических параметров (амплитуды, фазы, направления поля, коэффициента отражения и т.д.), которые называются информационными параметрами (ИП)сигнала.

4 Структура дискретной одноканальной системы связи

Связью называется передача и прием информации с помощью технических средств. Совокупность технических средств, предназначенных для передачи и приема сообщений, включая линию связи, называют каналом связи. Поскольку технические средства для дискретных и непрерывных сигналов различаются, каналы также

подразделяют на: дискретные и непрерывные. Под линией связи понимают любую физическую среду (воздух, металл, магнитная лента и т.п.), обеспечивающую поступление сигналов от передающего устройства к приемномуустройству. Системы, позволяющие одновременно передавать по одной линии связи несколько независимых сообщений, называют многоканальными.

Структурная схема одноканальной системы передачи информации приведена на рис.4.1[3].

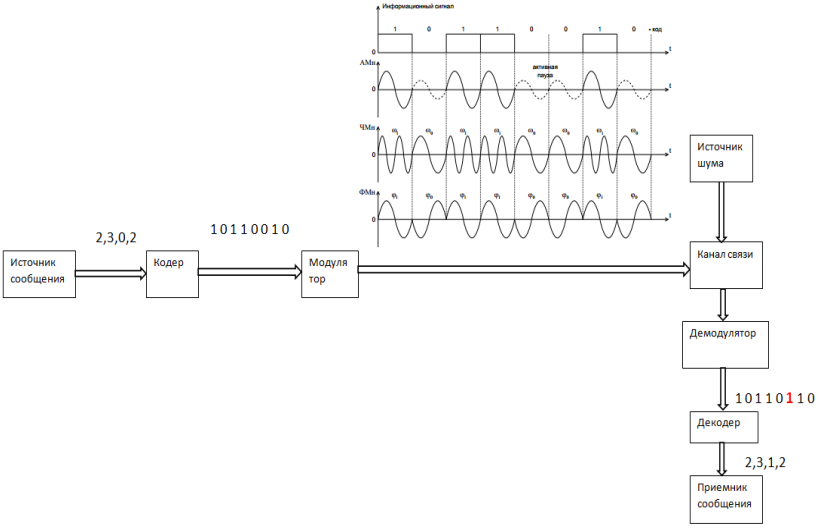


Рис. 4.1. Структурная схема системы передачи информации

Поступающее на вход системы сообщение формируется источником сообщений, состоящим из источника информации и первичного преобразователя (датчик, человек-оператор и т.п.).

5. Преобразование аналогового сигнала в цифровой вид

Использование дискретных сигналов для передачи, хранения и обработки информации имеет много преимуществ по сравнению с аналоговыми, потому что их можно представить в цифровом виде. Обратите внимание, что кодирование возможно только для дискретных сигналов. К основным преимуществам цифровых технологий можно отнести:

1. Возможность передавать информацию многократно без накопления ошибок, что обуславливается тем, что прием сигнала сводится к его обнаружению, а не регистрации точного значения. Это уменьшает требования к каналам связи.

2. Возможность использования помехоустойчивого кодирования при наличии помех, что значительно повышает достоверность передачи информации.

3. Возможность использовать цифровые технологии работы с информацией с помощью микропроцессоров и ЭВМ. Часто они более эффективны, чем аналоговые.

4. Возможность сжатия информации и ее согласования с каналом.

5. Унификация алгоритмов работы и аппаратуры для сигналов различной природы. Это приводит к массовости изготовления типовых узлов, что уменьшает стоимость их производства и эксплуатации, повышает надежность.

Поэтому во многих случаях производится преобразование непрерывных сигналов в дискретные (цифровые). Рассмотрим некоторые связанные с этим понятия. Сигнал, дискретный по одному параметру и непрерывный по другому, называют дискретно-непрерывным. На рис. 5.1, точками показан сигнал

дискретный по времени и непрерывный по уровню. Такой сигнал получается в результате процедуры дискретизации, заключающейся в измерении мгновенных значений ИП в заданные моменты времени, называемые отсчетами (вертикальные пунктирные линии).

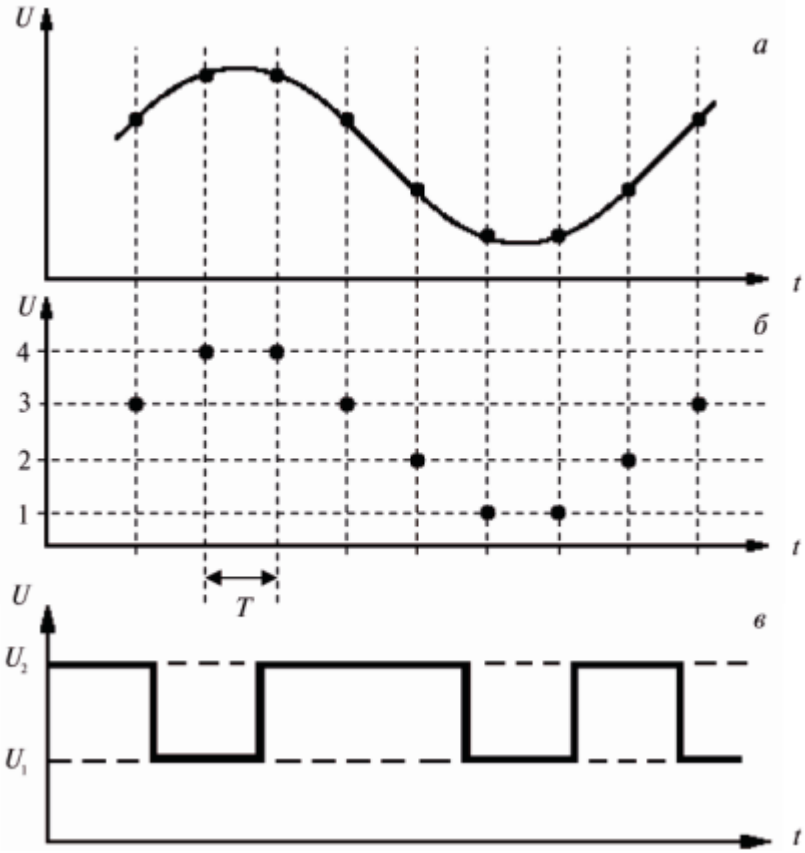


Рис.5.1.Преобразование аналогового сигнала в цифровой

Сигнал дискретный по уровню называется квантованным, а разрешенные значения уровня, которые

он может принимать, называются уровнями квантования. Процедура замены произвольных мгновенных значений на разрешенные называется квантованием. На рис. 5.1,бточками показан сигнал дискретный и по времени, и по уровню (квантованный). Уровни квантования пронумерованы числами от 1 до 4.

Таким образом, в результате дискретизации и квантования непрерывный сигнал сводится к набору точек, координаты которых и являются цифровым сигналом. Если дискретизация равномерная, т.е. отсчеты производятся через одинаковый промежуток времениT, называемый шагом дискретизации, то для записи сигнала в цифровом виде достаточно записать последовательные мгновенные значения или номера уровней для каждого отсчета. Величина $f=1/T$ называется частотой дискретизации.

В большинстве случаев для практического использования (например, прослушивания музыки) дискретный сигнал надо снова превратить в непрерывный. Эта процедура называется восстановлением. Поэтому важно определить условия, при которых это восстановление происходит без ошибок. Разумеется, чем меньше значениеT, тем точнее запоминается сигнал, однако при этом возрастает число запоминаемых точек, поэтому важно определить максимальное значение T_{max} (f_{min}), при котором сигнал восстанавливается без ошибок при минимальном числе точек.

Это значение устанавливается теоремой Котельникова (критерием Найквиста), которая гласит, что для безошибочного восстановления сигнала минимальная частота дискретизации должна быть в два раза больше максимальной частоты спектра сигнала.

Примеры:

5.1. Проводное радио использует частоты до 11 кГц. Какую минимальную частоту дискретизации надо выбрать?

Решение: 11 х 2 = 22 кГц.

5.2. Максимальная на 2008 г. частота дискретизации 2822400 Гц используется для форматаSACD(SuperAudioCompactDisc), разработанного компаниями Sony и Philips в 1999 году. Какую максимальную частоту содержит восстановленная фонограмма?

Решение: 2822400/2=1411200 Гц.

Вопросы для самопроверки:

1. Нарисуйте схему системы передачи информации. Объясните назначение частей.
2. Что такое цифровой сигнал?
3. Что такое дискретизация. Сформулируйте теорему Котельникова.

6. Подходы к измерениюинформации

Важнейшим вопросом теории информации является установление меры количества и качества информации [1]. Здесь существуют три направления, каждое из которых учитывает свой аспект информации и находит свое практическое применение.

– Структурный подход основан на анализе дискретного строения информации, и ее измерения простым подсчетом элементов(символов или неделимых частей информации – квантов). Структурный подход применяется для оценки возможностей аппаратуры и

кодов вне зависимости от конкретных условий их применения.

–Статистический подход учитывает вероятность появления и, следовательно, информативность тех или иных сообщений. Он опирается на понятие энтропии.Этот подход используется для учета конкретных условий применения информационных систем, например, при известных статистических характеристиках источника информации и помех. Оба этих подхода никак не учитывают смысл сообщений, а имеют дело только с их формой записи, благодаря чему могут быть формализованы. Оба подхода иногда объединяют общим названием «синтаксический подход».

–Семантический подход анализирует смысловое содержание сообщений их качество. В нем делаются попытки оценить истинность, целесообразность, ценность, полезность, существенность информации. Оценки полезности и т.п. также называют прагматическим подходом. Эти параметры в общем случае однозначно оценивать пока не научились из-за большой многофакторности подобных оценок, многозначности сообщений и сильной зависимости оценки от личности получателя, его квалификации, опыта, информированности.

Простейшей оценкой полезности является величина: $W = \log_2(p'/p)$, где p и p' – вероятности достижения цели до и после получения информации.

7. Геометрическая мера информации

Определение количества информации объемным методом сводится к подсчету количества дискретных элементов (квантов, символов) в данном сообщении или устройстве.Это соответствует длине, площади или объему

геометрической модели данной информации. Например, размер текстового файла 10 килобайт означает, что текст содержит 10240 символов («геометрически» это можно назвать «длиной» или «объемом» файла или текста).

Объемной мерой удобно определять возможности аппаратуры ИС.Это – максимально возможное количество информации, которое можно разместить в заданных структурных габаритах, и называемоеинформационной емкостью ИС. В частности, так измеряют возможности носителей информации или запоминающих устройств. Если используется двоичное представление информации, то емкость измеряется в битах или байтах (число бит приходящихся на 1 символ, в настоящее время 1 байт = 8 бит).

Говоря, емкость жесткого диска для персонального компьютера равна 120 Гигабайт – мы называем его информационную емкость («объем» диска).

Если величина Хможет принимать значения из диапазона L1, с дискретным отсчетом (шагом) Δ1, то информационная емкость равна количеству возможных значений (квантов)

$$I = m1 = L1/\Delta1.$$

Например, если L1– ширина листа бумаги, а Δ1– ширина символов, то I= m1– максимально возможное количество символов в строке (длина строки в символах).

Пусть отсчеты осуществляются сразу по двум осям Хи У.

Если величина Үможет принимать значения из диапазона L2с дискретным отсчетом Δ2, то количество ее квантов m2= L2/Δ2.Тогда информационная емкость двумерной системы равна I= m1*m2. Возвращаясь к примеру с бумагой, можно принять, что L2– высота листа

бумаги, Δ2– высота строки, m2– максимально возможное количество строк, тогда I – максимально возможное количество символов на листе (площадь листа в символах). Для изображений это будет количество точек (пикселей).

Аналогично можно определить количество информации при большем количестве осей (параметров).

8. Комбинаторная мера информации

Одной из форм кодирования является формирование кодовых слов как комбинаций из некоторых элементов. Например,азбука Морзе – это кодирование комбинациями точек и тире, числовое кодирование – комбинациями цифр (в частности, двоичное кодирование осуществляется комбинациями цифр 1 и 0).

Количество информации в комбинаторной мереопределяется как количество возможных кодовых комбинаций в данном коде. Это позволяет оценить возможность данного кода передавать информацию.

Вычислением числа комбинаций для различных видов соединения элементов занимается раздел математики, называемой комбинаторикой. Здесь приведем формулу только для комбинаций, которые различаются и составом элементов и их порядком при возможности повтора элементов. Пусть у нас имеется hвидов элементов, из которых будем создавать комбинации, состоящие из Lэлементов. Повторения означают, что какие–то из элементов входят в комбинацию несколько раз. Обозначим возможное число комбинаций Q. Такие комбинации называются размещениями из hэ лементов по Lc повторениями. Имеем [4]

$$Q = A_L^{-h} = h^L, \tag{8.1}$$

Если L– число разрядов в числе, а h– основание системы счисления (количество цифр), то (8.1) определяет количество возможных кодовых комбинаций.

Пример 8.1.При двоичном кодировании каждый символ кодируется байтом (восемь цифр 1 и 0). Сколько символов можно закодировать?

Решение. Число комбинаций находим по формуле (8.1) при h= 2, L= 8:

Q= 28= 256 комбинаций (закодированных символов).

9. Аддитивная мера информации Хартли и ее свойства

Пусть некоторый дискретный источник информации (система) имеет Qсостояний (алфавит имеет Qзнаков и т.п.). Возникает вопрос о мере количества информации I, которое солержит эта система. Можно положить I= Q. Однако, это неудобно, потому что такая мера не обладает свойством аддитивности, которому, совершенно очевидно, должна удовлетворять мера информации. Свойство аддитивности (от англ. «add» – складывать) означает, что для нескольких независимых источников количество информации должно складываться:

$$I= I1+ I2+ I3+ \ldots.$$

Пояснить это можно на следующем примере. Имеются два независимых источника информации с числом состояний соответственно Q1 и Q2. Тогда их можно рассматривать как один источник, реализующий пары состояний. Пусть у первого будет Q1 = 2 состояния, обозначенные a и b, а у второго –Q2 = 3 состояния обозначенные x, y и z. Тогда вместе они дадут 2*3 = 6 состояний (ax, ay, az, bx, by, bz), а не 2 + 3 = 5. Т.е. общее число его состояний (пар) Q= Q1*Q2, а не Q1+ Q2.

Нужна же такая мера, которая наряду с монотонной зависимостью от Q, обладала бы еще и свойством аддитивности.В 1928 г. американский ученый Р. Хартли предложил считать величину

$$I= \log_h Q \qquad (9.1)$$

количеством информации, содержащимся в системе, имеющей Qравновероятных состояний (h– в общем случае произвольное число). Эта мера информации в его честь получила название мера Хартли.

Покажем, что мера Хартли обладает свойством аддитивности [2].

$$I= \log_2 Q= \log_2 Q_1 \cdot Q_2= \log_2 Q_1+ \log_2 Q_2= I_1+ I_2.$$

Отметим, что свойством аддитивности, да еще и в сочетании с монотонностью зависимости от Q из всех элементарных функций обладает только логарифмическая функция. Именно это послужило Хартли основанием для введения такой меры. И поэтому до сих пор, несмотря на ряд попыток, этой мере не найдено альтернативы.

Важным достоинством введенной меры является ее прозрачный смысл с точки зрения кодирования. При кодировании состояний комбинационным кодом с помощью некоторого алфавита объема h каждому состоянию можно сопоставить кодовое слово длиной L, например, число из L цифр. По заданному необходимому числу кодовых комбинаций Q значение L легко найти из формулы 9.1:

$$L= \log_h h^L= \log_h Q= I.$$

Таким образом, **введенное количество информации *I* определяет количество символов *L* в кодовом слове, необходимое для кодирования *Q* знаков с помощью алфавита объема *h*.** (Предполагается, что все кодовые слова имеют одинаковое число символов *L*). Другими словами, если мы выберем код с кодовыми словами из *I* символов, то число возможных комбинаций окажется достаточным, чтобы закодировать все *Q* знаков (т.е. каждому из знаков сопоставить кодовую комбинацию). Поскольку *L* должно быть целым, в этом случае дробное значение *I* дополняется до ближайшего большего целого числа, например,

$$I = \log_h Q = 9{,}1 \approx 10.$$

Для числового кодирования введенное количество информации *I* определяет количество разрядов (цифр) в кодовом слове (числе) необходимое для кодирования *Q* знаков в системе счисления с основанием *h*.

Основание логарифма определяет только единицу измерения *I*, поэтому часто в общетеоретических работах вообще не указывается (пишут $I=\log Q$). Современная информационная техника основывается на двоичной логике и кодировании($h=2$). В дальнейшем будет использовано именно это значение.

При этом единицей измерения информации является бит(bit– от англ. binary digit– двоичная единица). В десятичной системе ($h = 10$) единицей измерения информации будет дит (dit – от англ. decimaldigit– десятичная единица), *1* дит = 3,32193 бит.

В теоретических исследованиях иногда используют натуральный логарифм (единица измерения «нит»), который проще дифференцировать и интегрировать.

Пример 9.1. Найти количество информации по Хартли, содержащееся в системе, информационная емкость которой характеризуется десятичным числом Q= 987.

***Решение*:** $I= \log_2 Q= \log_2 987 \approx 9{,}947$ бит ≈10 бит.

В завершение приведем значение количества информации для этого случая в других системах счисления. В натуральной:

$I= \ln Q= \ln 987 \approx 6{,}895 \approx 7$ нит.

В десятичной:

$I= \lg Q= \lg 987 \approx\approx 2{,}994 \approx 3$ дит.

Действительно, для записи числа использованы 3 десятичные цифры. В шестнадцатеричной: $I= \log_{16} Q= \log_{16} 987 \approx 2{,}487 \approx 3$. Последнее означает, что шестнадцатеричный код также имеет 3 разряда. Убедитесь самостоятельно, что $986_{10}= 3DA_{16}$.

Количество информации по Хартли можно интерпретировать как минимальное количество вопросов, которые надо задать (провести опытов) для гарантированного определения точного состояния системы. *h*здесь играет роль «значности» ответа. Например, h= 2 соответствует бинарным вопросам, на которые можно ответить только «да» (1) или «нет» (0), что составляет 1 бит информации. Опыт имеет два исхода. Цепочка ответов на последовательные вопросы «Да. Нет. Да….» может быть записана в виде двоичного числа 101…, длина которого и есть количество информации I. С точки зрения теории вероятности это соответствует рассмотрению только пар: событие — противоположное событие. Мера Хартли ничего не говорит о том, как построить алгоритм решения задачи, какие вопросы задавать и существует ли такой алгоритм вообще. Она

позволяет лишь оценивать придуманные алгоритмы с точки зрения их эффективности.

Пример 9.2.

Имеется 8 монет, одна из которых фальшивая. Известно, что фальшивая монета легче. Требуется определить фальшивую монету с помощью взвешивания на рычажныхвесах.

Решение:

1) h= 2. Количество информации I= log28 = 3 бит. Это означает, что для гарантированного нахождения фальшивки необходимо минимально3 взвешивания. Соответствующая ситуация реализуется делением всех монет на h= 2 группы. Взвешивание имеет 2 исхода – фальшивка в первой или во второй группе, и дает ответ на бинарный вопрос типа: «Левая чашка весовлегче?». Приведем алгоритм решения задачи. Разбив монеты на две группы по 4 штуки, взвешиваем, определяем группу с фальшивкой. Разбиваем подозрительные монеты на две группы по 2 штуки, взвешиваем, находим пару с фальшивкой. И, наконец, третьим взвешиванием находим фальшивую монету. Рассчитаем полученное одним взвешиванием количество информации.

Первое из них уменьшает число подозрительных монет с 8 до 4:

$\Delta I = \log 28 - \log 24 = 1$ бит. Что и следовало ожидать.

Все трехразрядные двоичные числа в этом случае соответствуют всем возможным вариантам результатов взвешиваний.

Например, 101 ≡«да, нет, да» – т.е. при первом взвешивании монета находилась в левой чашке, при втором – в правой и в третьем – снова в левой.

2) $h = 3$. Количество информации $I = \log_3 8 \approx 1{,}893 \approx 2$, т.е.в этом случае потребуется минимально 2 взвешивания. Ситуация реализуется делением всех монет на $h = 3$ группы: две равные кладем на чашки весов. Взвешивание

имеет 3 исхода. Фальшивка либо в более легкой чашке, либо, если чашки уравновешены, в третьей группе. В нашем случае, разбиваем монеты на группы 3 – 3 – 2, положим группы по 3 монеты на весы. Этим мы сразу сузим количество подозрительных монет до 3 или 2, и вторым взвешиванием окончательно проясним ситуацию.

Пример 9.3:

1) Определить количество информации по Хартли, содержащееся в системе, информационная емкость которой характеризуется десятичным числом Q= 2000, для h= 2, 8, 10, 16. Какой разрядности код потребуется в каждом случае, проверьте результат представлением числа Qв каждой системе счисления.

Решение:

$$I = \log_2 2000 = 10{,}96 \approx 11 бит.$$

$$I = \log_8 2000 = 3{,}7 \approx 14 бит.$$

$$I = \log_{10} 2000 = 3{,}3 \approx 4 бит.$$

$$I = \log_{16} 2000 = 2{,}75 \approx 3 бит.$$

2) У вас есть 20 монет с одной фальшивой и безмен (пружинные весы). Сколько взвешиваний вам понадобиться для нахождения фальшивой монеты, если вес настоящей вам известен.

Решение:

Делим монеты на 5 групп по 4 монеты. Проводим 5 взвешиваний. Легкую группу из 4-х монет взвешиваем по одной монете 4 раза.Получаем 9 взвешиваний.

3) Человек задумал число от 1 до 100. Вы пытаетесь угадать его, задавая ему бинарные вопросы типа: «Это число больше 5?». Каково минимальное количество вопросов, позволяющее гарантированно угадать число?

$$\log_2 100 = 6{,}64 \approx 7 вопросов.$$

Алгоритмрешения приведен на рис.9.1.

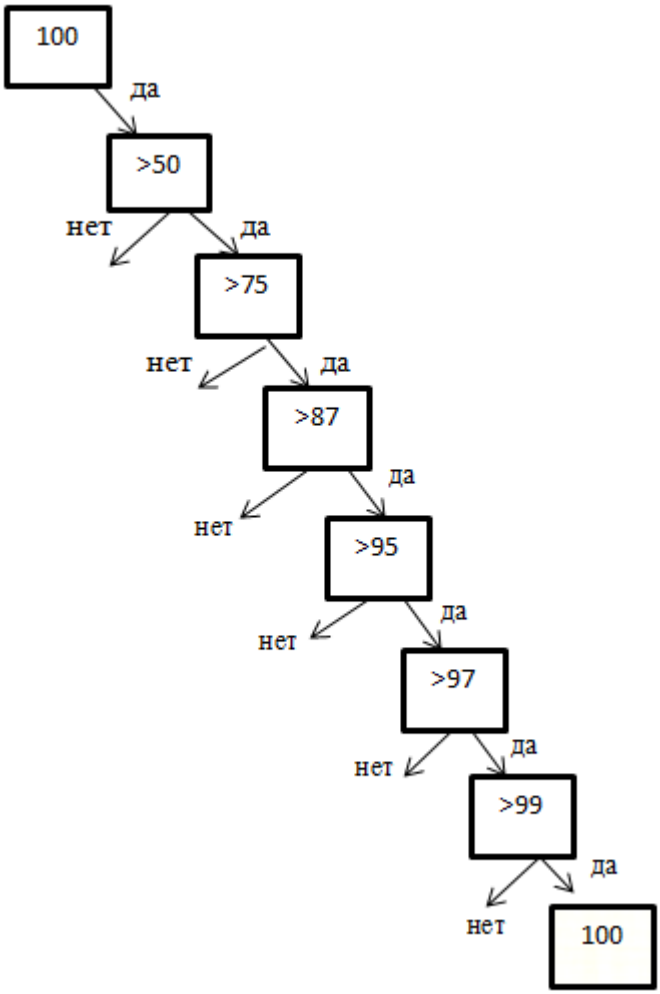


Рис.9.1. Алгоритм решения для примера 9.3

Если дискретный носитель информации представляет собой *N* позиций, в каждой из которых может находиться *Q* символов, то его информационную емкость можно найти по формуле

$$I = N\log_2 Q \qquad (9.2)$$

У точечного носителя (индикатор) имеется всего одна позиция, на которой может отображаться Q символов. В этом случае $I = \log_2 Q$. Например, индикатор в часах может отображать 10 цифр и для него $I = \log_2 10 \approx 3{,}32$ бит. Для двоичного индикатора (лампочка горит или не горит) $Q = 2$ и $I = 1$ бит. Для двумерного носителя информации (лист бумаги, монитор и т.п.), с m_1 отсчетов по одной координате и m_2 по другой получим

$$I = m_1 \cdot m_2 \cdot \log_2 Q.$$

Пример 9.4. На телеграфной ленте могут быть напечатаны 1000 букв. Телеграф использует только 32 заглавных символа. Определите информационную емкость ленты.

Решение:

$I = N\log_2 Q = 1000 \log_2 32 = 5000$ бит.

Сколько цветов можно закодировать 1 и 16 битным кодом?

Решение:

$2^1 = 2$ цвета, $2^{16} = 65535$ цветов.

10. Информация как мера уменьшения неопределенности. Понятие энтропии

Возможен другой подход к определению количества информации, основанный на теории вероятности [2]. Он связан с тем, что сообщения всегда представляют собой случайные события (если заранее знать сообщение, то оно уже не будет информацией). Для пояснения этого с синтаксической точки зрения, можно

рассмотреть побуквенное получение письма. Письмо не является случайной последовательностью букв, но получение той или иной буквы будет случайным процессом.

Рассмотрим в качестве источника информации систему, которая может принимать Q равновероятных состояний (знаки алфавита источника сообщений). В заданный момент система должна принять одно из этих Qсостояний, но мы не знаем какое. Такую ситуацию можно назвать неопределенностью выбора. Пусть нам надо угадать это состояние. Совершенно очевидно, что шансы угадать уменьшаются с увеличением Q(вероятность угадывания $p= 1/Q$). Поэтому можно сказать: чем больше Q, тем больше неопределенность ситуации. **Следовательно, величина Qи связанная с ней мера Хартли являются мерой неопределенности системы. В этом подходе мера Хартли будет называться энтропией (H):**

$$H= \log_h Q= \log_h (1/p)= -\log_h p. \qquad (10.1)$$

Заметим, что помимо аддитивности, мера Хартли удовлетворяет еще одному интуитивному требованию для меры неопределенности: для Q=1 – отсутствие неопределенности – мера должна давать 0.

Полученная информация должна приводить к уменьшению количества рассматриваемых возможных состояний, т.е. к уменьшению значения Q, и, следовательно, энтропии H(иначе полученное нельзя считать информацией, поскольку оно не изменит наших знаний о системе). Другими словами, полученная информация должна приводить к уменьшению неопределенности системы. Таким образом:

Информация – это сведения, которые уменьшают неопределенность ситуации, существующую до их поступления. Степень уменьшения неопределенности в результате передачи сообщения, характеризуемую изменением энтропии, назовем количеством информации (*I*) .

$$I=-\Delta H=H_1-H_2, \qquad (10.2)$$

где H_1– энтропия до получения сообщения, а

H_2–энтропия после получения сообщения.

Точное знание состояния системы соответствует всего одному возможному состоянию системы (Q = 1) и нулевой конечной энтропии $H_2=\log_h 1=0$ (неопределенность отсутствует). Это называют полным снятием неопределенности. При этом количество полученной информации будет равняться исходной энтропии: $I= H_1$. Это значение называют количеством информации, содержащимся в системе.

Единицы измерения количества информации и энтропии зависят от основания логарифма так же, как и у меры Хартли.

Пример10.1. Какое количество информации содержится в сообщении, что на игральной кости выпало: а) четное число,б) число 5.

Решение. Кость имеет $Q_1=6$ равновероятных состояний {1, 2, 3, 4, 5, 6}, т.е. начальная энтропия равна $H_1 = \log_2 6$.

а) После сообщения остаются возможными Q_2 = 3 состояния {2, 4, 6} и энтропия равна $H_2 = \log_2 3$. Следовательно, количество информации $I= -\Delta H= H_1 – H_2 = \log_2 6 – \log_2 3 = \log_2 2 = 1$ бит.

б) Состояние системы известно точно $Q_2 = 1$ и $H_2 = \log_2 1 = 0$.

Следовательно, количество информации $I = -\Delta H = H_1 - H_2 = H_1 = \log_2 6 = 2{,}585$ бит.

Степень неопределенности системы зависит от числа возможных ее состояний Q, однако оно не является исчерпывающей характеристикой степени неопределенности. Продемонстрируем это на примере двух ситуаций с двумя состояниями($Q = 2$).

1) Бросок монеты с равной вероятностью выпадения орла и решки ($p_1 = p_2 = 0{,}5$) и

2) Выстрел с расстояния 100 м по той же монете с вероятностью попасть $p_1 = 0{,}02$, промахнуться $-p_2 = 0{,}98$. В какой ситуации неопределенность больше?

Совершенно очевидно, предсказать результат выстрела гораздо проще. Предположив, что стрелок промахнется, мы почти наверняка угадаем. Следовательно, неопределенность ситуации с монетой больше. Итак, неопределенности этих систем различны, несмотря на одинаковое количество состояний. В результате можно сделать вывод, что в общем случае **энтропия (неопределенность) должна зависеть еще и от распределения априорных (доопытных) вероятностей состояний.**

При вероятностном подходе информация рассматривается как сообщение об исходе случайных событий, к которым также относятся реализации случайных величин и функций. События можно рассматривать как возможные исходы некоторого опыта, причем совокупность всех исходов составляет ансамбль или полную группу событий. Последнее означает несовместность событийи равенство единице суммы всех вероятностей: $p_1 + p_2 + \ldots + p_к = 1$ (обязательная реализация одного из событий группы). Табличное задание распределения вероятностей событий, представленное в табл. 10.1, называют также схемой ансамбля.

Таблица 10.1

Исходы опыта	A_1	A_2	….....	A_i	…..	A_k
Вероятность исхода	p_1	p_2	……..	p_i	.....	p_k

Определим, сколько информации о системе содержится в сообщении о реализации одного из исходов A_i. Воспользуемся классическим определением вероятности как отношения числа благоприятных равновероятных исходов n_iко всему количеству исходов N: $p_i=n_i/N$(например, вероятность 0,98 можно интерпретировать как n_i= 98 благоприятных исходов из N= 100 возможных исходов). Для ансамбля из kслучайных различных исходов (см. табл. 10.1) i= 1,2,…,kи $N= n_1+ n_2+ … + n_k$.

Тогда при сообщении о реализации состоянияA_iнеопределенность уменьшается с Nвозможных событий до n_i(реализовался один из благоприятных вариантов, но неизвестно какой).

Вычисляем полученное количество информации по Хартли:

$$I = \log_2 N - \log_2 n_i = -\log_2 \frac{n_i}{N} = \log_2 p_i .$$

Таким образом величину

$$I = -\log_2 p_i \qquad (10.3)$$

можно назвать количеством собственной информации об исходе A_i.

Отметим вытекающее отсюда следствие, что **наибольшую информацию несут сообщения о самых маловероятных исходах.** Им соответствует наименьшее число благоприятных событий инеопределенность системы. В рассмотренном примере о выстреле по монете

сообщение о промахе нас не удивит (напомним, его вероятность p_1=0,98), поэтому и информации оно несет мало I_1= $-\log_2 0{,}98$ = 0,03 бит. Попадание в монету будет почти сенсационным (p_2= 0,02) и сообщение об этом несет значительно больше информации: I_2= $-\log_2 0{,}02$ = 5,64 бит.

Энтропию всего ансамбля Шеннон предложил рассчитывать как среднее значение (математическое ожидание) собственной информации для всех исходов:

$$H = M[I] = -M[\log_2 p] =$$

$$= \sum_{i=1}^{k} p_i I_i = -\sum_{i=1}^{k} p_i \log_2 p_i. \qquad (10.4)$$

Сформулируем окончательно понятие энтропии для дискретной системы.

Энтропия –характеристика неопределенности системы, равная среднему количеству информации на одно состояние (знаксообщения):

$$H = -\sum_{i=1}^{k} p_i \log_2 p_i =$$

$$= -(p_1 \log_2 p_1 + p_2 \log_2 p_2 + \ldots.. + p_k \log_2 p_k), \textit{бит.} \qquad (10.5)$$

Среднее количество информации на один знак, полученное в сообщении или опыте равно уменьшению энтропии I=–ΔH, а в случае полного снятия неопределенности равно самой энтропии $I = H$.

11.Свойства энтропии

1) Энтропия – величина всегда вещественная, неотрицательная и ограниченная.

2) Энтропия минимальна и равна нулю только в том случае, когда вероятность одного состояния равна 1, а во всех остальных –0.

Это соответствует полному снятию неопределенности (математически это следует из (10.5) в силу того, что $\lim_{p->0} p\log_2 p = 0.$ т.е. можно считать $0\cdot\log_2 0 = 0$). Другими словами, энтропия заранее известного сообщения (состояния) равна нулю.

3)Энтропия максимальна, когда все вероятности равны между собой:

$$p_1=p_2=\ldots=p_к=1/k_n и H=Hmax=\log_2 k. \qquad (11.1)$$

Действительно $H= -k\cdot p\cdot\log_h p= -\log_h(1/k) = \log_h k.$

Как и следовало ожидать, для равновероятных состояний (10.5) совпадает с мерой Хартли (10.1) (*k*здесь означает то же, что и *Q*– количество равновероятных состояний системы).

4) Энтропия аддитивна, т.е. энтропия объединения нескольких статистически независимых источников равна сумме их энтропий (доказательство см. в п.12).

Для иллюстрации первых трех свойств энтропии рассмотрим энтропию системы, способной принимать только два состояния. Если вероятность первого состояния $p_1= p$, то вероятность второго $p_2= 1 – p$. Тогда энтропия как функция p имеет вид

$$H(p) = -p_1\log_2 p_i - p_2\log_2 p_2 =$$
$$= -p\log_2 p - (1-p)\log_2(1-p).$$

График этой функции приведен на рис. 11.1.

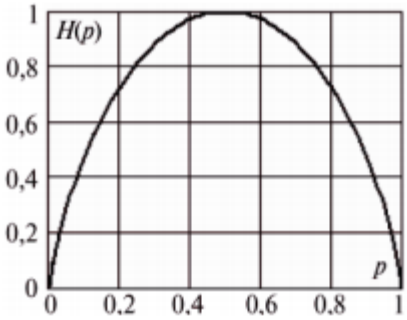


Рис.11.1.Энтропия системы с двумя состояниями

Как и ожидалось, энтропия равна нулю при p_1=p= 0 (p_2= 1) и p_1= p= 1(p_2= 0), и максимальна приp_1= p_2= p= 0,5.

Для броска монеты с равной вероятностью выпадения орла и решки (p_1= p_2= 0,5) энтропия: H= – $p_1\log_2 p_1$– $p_2\log_2 p_2$== –2·0,5·$\log_2$ 0,5 = 1 бит.

Для выстрела с вероятностью попасть p_1= 0,02 и промахнуться –p_2= 0,98:

H= –$p_1\log_2 p_1$– $p_2\log_2 p_2$= –0,02·$\log_2$ 0,02 –0,98$\log_2$0,98 =0,1128+0,0286=0,1414 бит.

Таким образом, неопределенность ситуации с выстрелом значительно ниже, как уже было сказано выше, исходя из интуитивных соображений.

Также, как и мера Хартли, собственное количество информации I_i по(10.3) показывает минимальное количество разрядов, необходимых для двоичного кодирования знака, появляющегося в сообщении с вероятностью p_i, только при этом кодовые слова имеют разную длину, причем наиболее вероятные должны быть самыми короткими. Соответственно, **энтропия – это среднее количество разрядов, необходимое для передачи одного знака алфавита сообщения. Таким**

образом, при двоичном кодировании энтропия указывает минимальную среднюю длину кодовых слов, что нашло применение в теории кодирования.

Не путайте понятие количество информации, характеризующее систему (количество состояний, их статистические характеристики) и связанные с ней алфавит и способы кодирования, с понятием объем информации (информационная емкость), который равен числу символов. Поясним это примером. Пусть имеется алфавит из 32 равновероятных букв, с помощью которого передается слово из трех букв. Тогда количество информации (энтропия) $I = H = \log_2 32 = 5$ бит/символ – есть характеристика алфавита, которая говорит о том, что данный алфавит может быть закодирован двоичным 5–разрядным кодом. При этом объем информациив трехбуквенном слове будет равен $3 \cdot 5 = 15$ бит. Этот алфавит может быть закодирован и 8–разрядным кодом, тогда объем информации в этом слове будет $3 \cdot 8 = 24$ бит. **В общем случае количество информации не превышает ее объема.**

12. Энтропия объединения ансамблей

В предыдущем параграфе рассмотрена простая система. Обратимся теперь к случаю сложной системы, представляющей собой объединение нескольких простых систем (источников информации). С объединением связаны понятия безусловной, условной, совместной и взаимной энтропии. Для простоты рассмотрим объединение двух ансамблей $X = \{x_1, x_2, \ldots, x_n\}$ и $Y = \{y_1, y_2, \ldots, y_m\}$, которым соответствуют безусловные энтропии $H(X)$ и $H(Y)$ (10.5):

$$H(X) = \sum_{i=1}^{n} \log_2 p(x_i)\, H(Y) = \sum_{i=1}^{m} \log_2 p(y_i). \qquad (12.1)$$

Событие, соответствующее объединению двух ансамблей, заключается в одновременном появлении пары событий (x_i, y_j) с совместной вероятностью p_{ij}= $p(x_i, y_j)$. Поэтому объединение ансамблей характеризуется прямоугольной матрицей $P(X, Y)$ вероятностей всех возможных комбинаций состояний систем p_{ij}. Эта матрица имеет размер $n \times m$ и обладает следующими свойствами:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} p(x_i, y_j) = 1,\ p(x_i) = \sum_{j=1}^{m} p(x_i, y_j),$$

$$p(y_i) = \sum_{i=1}^{n} p(x_i, y_j). \qquad (12.2)$$

Совместная энтропия ансамблей Xи Yаналогично выражению (10.5) имеет вид:

$$H(X,Y) = \sum_{i=1}^{n} \sum_{j=1}^{m} p(x_i, y_j) \log_2 p(x_i y_j), \qquad (12.3)$$

Рассмотрим ситуацию в случае независимых систем X и Y, когда они принимают свои состояния независимо одна от другой. Тогда по теореме умножения вероятностей для независимых случайных величин:

$$p(x_i, y_j) = p(x_i, y_j),$$

$$\log_2 p(x_i, y_j) = \log_2 p(x_i) + \log_2 p(y_j), \qquad (12.4)$$

$$H(X,Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j)\log_2 p(x_i y_j) =$$

$$= -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i)p(y_j)(\log_2 p(x_i) + \log_2 p(y_j)) =$$

$$-\sum_{i=1}^{n} p(x_i \log_2 p(x_i)\sum_{j=1}^{m} p(y_j) - \sum_{j=1}^{m} p(y_j)\log_2 p(y_j)\sum_{i=1}^{n} p(x_i).$$

Учитывая $\sum_{j=1}^{m} p(y_j) = 1, \sum_{i=1}^{n} p(x_i) = 1,$

получаем для совместной энтропии

$H(X,Y)=H(X)+H(Y).$ (12.5)

Таким образом, мы доказали свойство аддитивности энтропии: при объединении независимых систем их энтропии складываются. Выражение (12.5) можно было вывести из (12.4) и свойства математического ожидания:

$$H(X,Y) = M\left[-\log_2 p(x,y)\right] = M\left[-\log_2 p(x) - \log_2 p(y)\right] =$$

$$= M\left[-\log_2 p(x)\right] + M\left[-\log_2 p(y)\right] = H(X) + H(Y).$$

Этот вывод легко обобщается на любое количество ансамблей.

При вычислении энтропии объединения систем в общем случае необходимо учитывать статистические связи между их состояниями и использовать общую формулу (12.3). Попробуем понять причину влияния связей на энтропию на примере побуквенного получения письма. Появление очередной буквы не совсем случайно. Знание того, какие сочетания букв возможны, а какие нет, позволят получателю иногда предугадывать будущую букву, иногда по части слова отгадывать само слово. Таким образом, неопределенность и, соответственно, энтропия будут в этом случае меньше, чем при совершенно случайных буквах.

Для статистически зависимых ансамблей вероятность $p(x_i,y_j)$ можно выразить через условные вероятности $p(y_j/x_i)$ или $p(x_i/y_j)$, в соответствии с тем какой ансамбль принимается за причину, а какой за следствие:

$$p(x_i, y_j) = p(x_i)p(y_j/x_i) = p(y_j)\, p(x_i/y_j). \qquad (12.6)$$

Пусть X–причина, а Y– следствие, что соответствует первому равенству в (12.6). Здесь условная вероятность $p(y_j/x_i)$ – это вероятность того, что реализуется состояние y_jиз ансамбля Y при условии, что реализовалось состояние x_iиз ансамбля X. Например, вероятность принять символ y_j,

если послан символ x_i, аналогично $p(x_j/x_i)$ – вероятность, что в тексте после символа x_iследует x_j.

При этом объединение ансамблей характеризуется матрицей условных вероятностей $P(Y/X)$всех возможных комбинацийсостояний системы [5]:

$$P(X,Y)=\begin{Vmatrix} p(y_1/x_1) & p(y_2/x_1) & ..... & p(y_m/x_1) \\ p(y_1/x_2) & p(y_2/x_2) & ..... & p(y_m/x_2) \\ ..... & ..... & ..... & ..... \\ p(y_1/x_n) & p(y_2/x_n) & ..... & p(y_m/x_n) \end{Vmatrix}. \qquad (12.7)$$

Суммы по строкам элементов матрицы (12.7) обладают свойством:

$$\sum_{j=1}^{m} p(y_j/x_i)=1. \qquad (12.8)$$

Частной условной энтропией системы Y относительно отдельного события x_iназовем величину:

$$H(X/x_i)=-\sum_{j=1}^{m} p(y_j/x_i)\log_2 p(y_j/x_i), \qquad (12.9)$$

которая характеризует неопределенность состояний системы Y, если система X примет состояние x_i (неопределенность того, какой символ будет принят в

приемнике, если был послан x_i, или неопределенность того, какой символ будет следующим, если предыдущим был x_i).

Полной условной энтропиейили просто условной энтропией(иногда ее также называют зависимой энтропией) системы Y относительно системы X назовем усредненное значение частной энтропии по всем состояниям системы X:

$$H(X/Y) = \sum_{i=1}^{n} p(x_i) H(Y/x_i) =$$

$$= -\sum_{i=1}^{n} \left[p(x_i) \sum_{j=1}^{m} p(y_j/x_i) \log_2 p(y_j/x_i) \right]. \quad (12.10)$$

Она характеризует среднюю неопределенность, приходящуюся на одно состояние системы Y при известных состояниях системы X (неопределенность того, какой символ принят в приемнике, если известен передаваемый ансамбль символов X).

Аналогично можно определить энтропию $H(X/Y)$, учитывающую вероятности $p(x_i/y_j)$, которая характеризует среднюю неопределенность, приходящуюся на одно состояние системы X при известных состояниях системы Y (среднюю неопределенность того, какой символ послан, если известен получаемый ансамбль символов Y).Условную энтропию (12.10) можно записать аналогично (10.4).

$$H(X/Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i)\,p(y_j/x_i)\log_2 p(y_j/x_i) =$$

$$= -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j)\log_2 p(y_j/x_i) = M\left[-\log_2 p(y/x)\right].$$

Тогда из (12.3), используя (12.6) легко получаем выражение для совместной энтропии объединения двух ансамблей при наличии статистических связей:

$$H(X,Y) = M\left[-\log_2 p(x,y)\right] = M\left[-\log_2 (p(x)\,p(y/x)\right] =$$

$$= M\left[-\log_2 p(x) - \log_2 p(y/x)\right] =$$

$$= M\left[-\log_2 p(x)\right] + M\left[-\log_2 p(y/x)\right] = H(X) + H(Y/X).$$

Аналогично доказывается $H(X, Y)$ = H(Y) + H(X/Y). Окончательно

$$H(X, Y) = H(X) + H(Y/X) = H(Y) + H(X/Y). \qquad (12.11)$$

13. Основные свойства условной энтропии

1) Условная энтропия не превышает безусловную: $H(Y/X) \le H(Y)$, $H(X/Y) \le H(X)$ и энтропия объединенной системы[5]

$$H(X, Y) \le H(X) + H(Y).$$

Очевидно, что наличие сведений о реализации состояний одного ансамбля не может увеличить неопределенность выбора состояний другого ансамбля. Эта неопределенность может только уменьшиться, если существует взаимосвязь ансамблей.

2) Если сообщения X и Y статистически независимы, то $H(Y/X) = H(Y)$ и $H(X/Y) = H(X)$. В этом случае вся информация о реализациях Yявляется новой по отношению к информациио реализациях X. В этом случае энтропия объединенной системы достигает максимума $H(X, Y) = H(X) + H(\mathrm{Y})$.

3) При наличии однозначной связи состояний двух систем условные энтропии равны нулю

$$H(Y/X) = H(X/Y) = 0 \text{и} H(X, Y) = H(X) = H(Y).$$

Это означает, что сообщения Y не содержат никакой новой информации сверх той, которая содержится в сообщениях X. При передаче это соответствует отсутствию помех (нет потерь и искажений), когда количество принимаемой информации равно количеству посылаемой. Если в канале связи присутствуют помехи, то они повышают неопределенность системы $H(X, Y) > H(X), H(Y)$.

Помехи уничтожают часть информации. Не противоречит ли это выводу о возрастании энтропии – количества информации? Не противоречит. Возрастание энтропии при наличии помех означает, что для компенсации потери части информации, ее надо передавать больше.

Типичным примером объединения ансамблей является передача символов по каналу связи при наличии помех. В этом случае ансамбль $x_1,\ldots, x_n$– это символы, которые посылает передатчик, а $y_1,\ldots, y_m$– это символы, которые принимает приемник.

14. Пропускная способность канала связи без шумов

Безусловная энтропия имеет максимальное значение при равновероятности всех символов. Итак, максимальное значение энтропии на символ имеет место в том случае, когда, во-первых, между символами отсутствуют вероятностные связи, а, во-вторых, когда все символы алфавита равновероятны. Определенное таким образом максимальное значение энтропии источника называется информационной емкостью источника. Информационная емкость источника, использующего алфавит k,вычисляется как

$$H_{max}=\log_2 k. \quad (14.1)$$

Для характеристики использования символов в сообщении введен параметр, называемый избыточностью – R.

$$R=\frac{H_{\max}-H(X)}{H_{\max}}=1-\frac{H(X)}{H_{\max}}=1-M. \quad (14.2)$$

Величину

$$M= H(X) /H_{max} \quad (14.3)$$

называют коэффициентом сжатия M, $H(x)$ – энтропия на один символ сообщения.

Избыточность приводит к увеличению времени передачи информации, излишней загрузке канала связи. Имеется и определенная избыточность в русском языке и в европейских языках.

Наибольшая возможная в данном канале скорость передачи информации называется его пропускной способностью. Пропускная способность

канала есть скорость передачи информации при использовании «наилучших» (оптимальных) для данного канала источника, кодера и декодера, поэтому она характеризует только канал.

Номинальная (техническая) скорость – битовая скорость передачи данных без различия служебных и пользовательских данных.Эффективная (информационная) скорость – скорость передачи пользовательских данных (нагрузки). Этот параметр зависит от соотношения накладных расходов и полезных данных.

Пропускная способность дискретного (цифрового) канала без помех.Пропускной способностью канала связи называется верхняя грань скорости передачи информации при заданных фиксированных ограничениях[3]:

$$C=H_{max}*V_T=H_{max}/\overline{\tau_c} \qquad (14.4)$$

где $\overline{\tau_c}$– средняя длина символа,

V_T– скорость передачи символов.

Как уже указывалось, $H_{max} = \log_2 k$.

Таким образом,

$$C=\log_2 k/\overline{\tau_c}. \qquad (14.5)$$

Для бинарного канала (k = 2) при одинаковой длительности обоих сигналов

$$C = \frac{1}{\tau_c} \approx \Delta f_{эфф},$$

где $\Delta f_{эфф}$ – эффективная полоса пропускания канала.

Предельные возможности канала никогда не используются полностью. Степень его загрузки характеризуется коэффициентом использования канала

$$\lambda = I'(Z)/C,$$

где $I'(Z)=H(z)/\tau$ – производительность источника сообщений Z(бит/с), а

τ – средняя длительность выдачи одного знака.

15. Теоремы Шеннона о кодировании для канала связи без помех и для канала связи с помехами

Введенные понятия важны в связи с доказанными Шенноном теоремами.

Теорема 1.О кодировании для канала без помех: При $I'(Z) < C$ существует способ кодирования, позволяющий передавать по каналу все сообщения, вырабатываемые источником без задержки.

При $I'(Z) > C$ передача сообщений без их накопления (задержки) невозможна. На этой теореме основано эффективное кодирование (минимизация количества символов на сообщение).

Теорема 2.О кодировании для канала с помехами: При $I'(Z) < C$ существует способ кодирования, позволяющий передавать по каналу все сообщения, со сколь угодно малой вероятностью ошибки. При $I'(Z) > C$ такая передача сообщений без задержки невозможна. На этой теореме основано помехоустойчивое кодирование. Не указывая путей построения эффективных кодов, эта теорема определяет предельные возможности эффективного кодирования.

Вопросы для самоконтроля:

1. Что такое энтропия? Каковы основные свойства энтропии для дискретных систем?

2. В чем отличия количества информации по Хартли и по Шеннону?

3. Что такое информация и количество информации, и их связь с энтропией?

4. Что такое энтропия объединения двух систем в случае независимых и зависимых систем? Назовите свойства условной энтропии.

5. В чем различие между понятиями объем информации и количество информации?

6. В чем суть понятия взаимной информации? Приведите выражения для взаимной информации для случаев полной независимости и полной зависимости систем.

7. Что такое избыточность источника?

8. Что такое пропускная способность канала связи? Чем отличается пропускная способность от скорости передачи информации?

9. Чем отличается информационная скорость передачи от технической, и в каких единицах эти скорости измеряются?

10. Приведите выражение пропускной способности для дискретного канала без помех и с помехами. Как изменяется пропускная способность канала связи при воздействии помех.

11. Сформулируйте основные теоремы Шеннона о кодировании для канала с помехами и без них.

Примеры*:*

15.1. Оценить количество информации в сообщениях, о том, что из колоды из 36 карт вытащили

короля, пику, пикового короля. Какое сообщение несет больше информации?

Решение: $H_{max} = \log_2 36 = 5{,}17$. Король 4 состояния: $H_k = \log_2 4 = 2$, $I_1 = \Delta H = H_{max} - H_k = 5{,}17 - 2 = 3{,}17$ бит.

Пика –9 состояний: $H_n = \log_2 9 = 3{,}16$, $I_2 = \Delta H = H_{max} - H_n = 5{,}17 - 3{,}16 = 2{,}01$ бит.

Пиковый король 1 состояние: $H_{пк} = \log_2 1 = 0$, $I = 5{,}17$бит.

15.2. Сколько информации несет сообщение о том, что взятое наудачу целое положительное число, не превосходящее 20, окажется простым?

Решение: Простое число делится только на 1 и на самое себя.

$H_{max} = \log_2 20 = 4{,}32$ бит.

Простые числа <20 это: 2,3,5,7,11,13,17,19.

$H_{пр} = \log_2 8 = 3$. $I = H_{max} - H_{пр} = 4{,}32 - 3 = 1{,}32$бит.

15.3. Алфавит содержит 4 символа с вероятностями появления соответственно $p_1 = 0{,}10$; $p_2 = 0{,}45$; $p_3 = 0{,}2$; $p_4 = 0{,}25$. Длительности символов $\tau_1 = 5$с; $\tau_2 = 2$с; $\tau_3 = 4$с, $\tau_4 = 3$с. Определить количество информации на символ сообщения и скорость передачи сообщений.

Решение: Скорость передачи информации по каналу без помех, если символы не равновероятны:

$$R = \frac{H(i)}{\tau},$$

где τ – длительность сигнала,

$H(i) = -p_i \log_2 p_i$.

$H(1) = -0{,}1\log_2 0{,}1 = 0{,}3322 = I_1$(бит).

$R_1 = 0{,}3322/5 = 0{,}0664$ бит/с.

$H(2) = -0{,}45\log_2 0{,}45 = 0{,}5184 = I_2$(бит).

$R_2 = 0{,}5184/2 = 0{,}2592$ бит/с.

$H(3) = -0{,}2\log_2 0{,}2 = 0{,}4644 = I_3$(бит).

R_3 =0,4644/4=0,1164 бит/с.

H (4)=–0,25log$_2$0,25= 0,5 = I_4(бит). R_4 =0,5/3=0,1667 бит/с.

15.4. Определить пропускную способность симметричного канала, передающего 50 символов 0 или 1 в секунду, если вероятность ошибки p = 0,1. Достаточна ли пропускная способность этих каналов для передачи информации, поставляемой источником с производительностью 40 бит/с?

Решение:

C=V_T (1–0,1) = 50*0,9 =45 бит/с.

Пропускная способность 40 бит/сдля передачи этой информации недостаточна.

15.5. Алфавит содержит 3 символа с вероятностями появления соответственно p_1 = 0,6; p_2 = 0,3; p_3 = 0,1. Время передачи одного символа τ = 0,02 с. Определить скорость передачи информации, если помехи в канале связи заданы канальной матрицей $P(X/Y)$

$$P(X/Y) = \begin{Vmatrix} 0,9 & 0,05 & 0,05 \\ 0,2 & 0,6 & 0,2 \\ 0,1 & 0,15 & 0,75 \end{Vmatrix},$$

6.

Решение:

$$H_{\max} = -\sum_{i=1}^{3} p_i \log_2 p_i =$$

$$= -\left[0,6\log_2 0,6 + 0,3\log_2 0,3 + 0,1\log_2 0,1\right] = 1,2955 \textit{бит},$$

$$H(Y/X) = -\sum_{i=1}^{3} | p(x_i) \sum_{i=1}^{3} p(y_i / x_i) \log_2 p(y_i / x_i) | =$$

$$= -[0,6(0,9\log_2 0,9 + 0,05\log_2 0,05 + 0,05\log_2 0,05] -$$

$$-[0,3(0,2\log_2 0,2 + 0,6\log_2 0,6 + 0,2\log_2 0,2)] -$$

$$-[0,1(0,1\log_2 0,1 + 0,15\log_2 0,15 + 0,75\log_2 0,75)] =$$

$$= 0,8581 бит.$$

Rсим= (*Hmax*–*H*(*Y*/*X*)/t=(1,2955–0,8581)/0,02= 21,8708 =22 бит/с.

15.6. Для канала связи, с канальной матрицей $P(X, Y)$

$$P(X,Y) = \begin{Vmatrix} 0,4 & 0 & 0,1 \\ 0 & 0,1 & 0 \\ 0,1 & 0 & 0,3 \end{Vmatrix}$$

найдите все виды энтропий. Определить: чему равна его пропускная способность, если знаки вырабатываются со скоростью 100 знаков в секунду?

Решить самостоятельно.

16. Модель симметричного дискретного канала связи

Симметричный дискретный канал обладает следующими свойствами:

1) Набор передаваемых $\{x_i\}$ и принимаемых $\{y_i\}$ символов одинаков. Пусть этот набор состоит из m символов, тогда канал называется m–м.

2) Для всех x_i вероятность ошибочного приема одинакова (обозначим ее p), соответственно, одинакова и вероятность безошибочного приема $p(y_i/x_i) = 1–p$.

3) Ошибка приема заключается в том, что при передаче символа x_i принимается символ y_j ($j \neq i$). Таким образом, имеем (m – 1) возможную ошибку, вероятности которых полагаются одинаковыми $p(y_j/x_i) = p/(m – 1)$ при $j \neq i$. Канальная матрица имеет вид:

$$P(Y/X) = \begin{Vmatrix} 1-p & \frac{p}{m-1} & .... & \frac{p}{m-1} \\ \frac{p}{m-1} & 1-p & .... & \frac{p}{m-1} \\ .... & .... & .... & .... \\ \frac{p}{m-1} & \frac{p}{m-1} & .... & 1-p \end{Vmatrix}. \qquad (16.1)$$

Вычисляем условную энтропию $H(Y/X)$ по формуле (12.10):

$$H(Y/X) = -\sum_{i=1}^{m}\left[p(x_i)\sum_{j=1}^{m} p(y_j/x_i)\log_2 p(y_j/x_i)\right] =$$

$$= -\sum_{i=1}^{m}\left[p(x_i)\big((1-p)\log_2(1-p) + (m-1)\frac{p}{m-1}\log_2\frac{p}{m-1}\right] =$$

$$= -\left((1-p)\log_2(1-p) + p\log_2\frac{p}{m-1}\right)\sum_{i=1}^{m} p(x_i) =$$

$$= (1-p)\log_2 \frac{1}{1-p} + p\log_2 \frac{m-1}{p}. \qquad (16.2)$$

Здесь учтено, что при суммировании по j (по строкам матрицы (16.1)) каждая сумма содержит одно слагаемое с $(1-p)$и $(m-1)$ слагаемое с $p/(m-1)$ и

$$\sum_{i=1}^{m} p(x_i) = 1.$$

Видно, что для симметричного канала $H(Y/X)$ не зависит от распределения вероятности входных сигналов $p(x_i)$, следовательно, их выбором можно максимизировать только энтропию $H(Y)$. Тогда из (14.4) получаем

$$C = V_T \max\{H(Y) - H(Y/X)\} = V_T [\max H(Y) - H(Y/X)]. \quad (16.3)$$

По свойству энтропии она максимальна в случае равновероятных и независимых символов y_iи равна $\max H(Y) = \log_2 m$.В силу симметрии канала это достигается для равновероятных и независимых символов x_i. Учтя (16.2) получаем:

$$C = V_T\left(\log_2 m - (1-p)\log_2 \frac{1}{1-p} - p\log_2 \frac{m-1}{p}\right). \qquad (16.4)$$

Наиболее простым является двоичный симметричный канал $m = 2$, который схематически изображен на рис. 16.1.

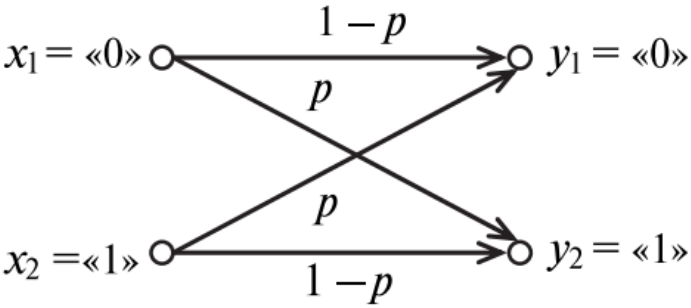


Рис.16.1. Двоичный симметричный канал

Линии со стрелками указывают, в какие принимаемые символы y_jмогут перейти передаваемые символы x_j. Рядом со стрелками приведены вероятности такого перехода. Канальная матрица в этом случае имее вид

$$P(Y/X)=\begin{Vmatrix} 1-p & p \\ p & 1-p \end{Vmatrix}.$$

А пропускная способность (16.4)

$$C = V_T\left(1+(1-p)\log_2(1-p)+p\log_2 p\right). \qquad (16.5)$$

Заметим, что при $p = 0$ (нет помех) получаем$C = V_T$.

При $p = 0{,}5$ $C = 0$, что означает невозможность связи. Это связано с невозможностью определить, правилен или ошибочен принятый символ.

Если набор принимаемых символов отличается от передаваемых, то канал перестает быть симметричным, а канальная матрица перестает быть квадратной (число строк в ней равно числу передаваемых, столбцов – принимаемых символов).

Однако, если все строки канальной матрицы будут содержать одинаковый набор элементов (вероятностей), то $H(Y/X)$ в этом случае также не будет зависеть от распределения вероятности входных сигналов $p(x_i)$, и можно применить данный подход. Иногда такой канал тоже называют симметричным.

17. Модель дискретного канала связи со стиранием

Несколько иная ситуация возникает в каналах со стиранием. Стирание означает, что принимаемый сигнал настолько искажается, что его нельзя сопоставить ни с одним из уже известных символов. В этом случае удобно считать, что принят какой-то неизвестный символ y_{m+1}. Рассмотрим простейший пример подобной ситуации:

симметричный двоичный канал со стиранием, схема которого приведена на рис. 17.1.

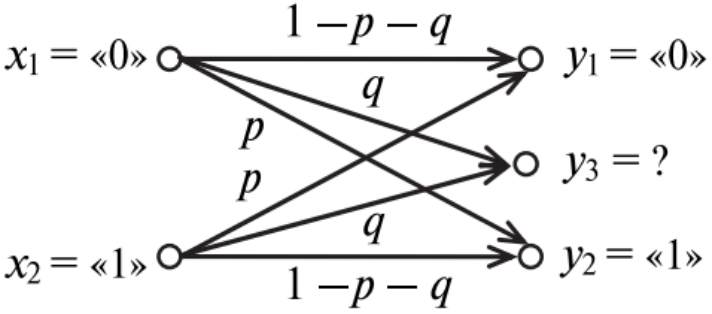


Рис.17.1. Симметричный двоичный канал со стиранием

Симметричность здесь проявляется еще и равенством вероятностей стирания q (переход в символ y_3).

Канальная матрица и условная энтропия в этом случае имеет вид:

$$P(Y/X)=\begin{Vmatrix} 1-p-q & p & q \\ p & 1-p-q & q \end{Vmatrix}. \quad (17.1)$$

$$H(Y/X) = -(p(x_1)+p(x_2))(1-p-q)\log_2(1-p-q) + \\ + p\log_2 p + q\log_2 q) =$$

$$-[(1-p-q)\log_2(1-p-q) + p\log_2 p + q\log_2 q]. \quad (17.2)$$

По-прежнему $H(Y/X)$ не зависит от распределения вероятности входных сигналов $p(x_i)$ и можно применить формулу (16.3). Отличие заключается в том, что здесь нельзя взять $\max H(Y) = \log_2 m = \log_2 3$. Дело в том, что вероятность $p(y_3) = q$ символа y_3 фиксирована и не может быть изменена подбором распределения $p(x_i)$, поэтому нельзя полагать все символы y_iравновероятными. Однако очевидно (это легко доказать), что энтропия максимальна, если равновероятны y_1 и y_2. Их вероятности при этом легко найти: $p(y_1) = p(y_2) = (1 - q)/2$. Это соответствует равновероятными независимым символам x_i. Тогда из (10.5) получим

$$\max H(Y) = -q\log_2 q - \frac{1-q}{2}\log_2\frac{1-q}{2} - \frac{1-q}{2}\log_2\frac{1-q}{2} =$$

$$= -q\log_2 q - (1-q)\log_2\frac{1-q}{2}$$

и из (16.3) при учете (16.6) получаем:

$$C = V_T[-(1-q)\log_2\frac{1-q}{2} + (1-p-q)\log_2(1-p-q) + p\log_2 p]. \quad (17.3)$$

На практике часто имеет место ситуация $p<<q$, т.е. вероятность того, что помеха сотрет символ выше, чем то, что искаженный символ спутают с другим. В этом случае можно положить$p = 0$ и из (17.3) получить$C = V_T(1 - q)$. Этот результат имеетясный смысл, ведь q– доля искаженных символов, значит$(1 - q)$ – доля неискаженных символов, передающих информацию.

Пример 17. Определить пропускную способность симметричного дискретного канала связи со скоростью манипуляции $V_T = 20$ бод в предположении независимости

передаваемых символов x_i. Помехи в канале определяются канальной матрицей условных вероятностей.

$$P(Y/X)=\begin{Vmatrix} 0{,}3 & 0{,}3 & 0{,}2 & 0{,}2 \\ 0{,}2 & 0{,}2 & 0{,}3 & 0{,}3 \end{Vmatrix}.$$

Решение. В системе 2 входных и 4 выходных символа. Пропускную способность найдем по формуле (16.2). Условную энтропию найдем по формуле (12.10):

$$H(Y/X)=-\sum_{i=1}^{2}\left[p(x_i)\sum_{j=1}^{4} p(y_j/x_i)\log_2 p(y_j/x_i)\right]=$$

$$=-p(x_1)\,[0{,}3\log_2(0{,}3)+[0{,}3\log_2 0{,}3+0{,}2\log_2 0{,}2+$$
$$+0{,}2\log_2 0{,}2]=$$

$$-p(x_2)\,[0{,}2\log_2 0{,}2+0{,}2\log_2 0{,}2+0{,}3\log_2 0{,}3+$$

$$+0{,}3\log_2 0{,}3]=$$

$$=[0{,}3\log_2 0{,}3+0{,}3\log_2 0{,}3+0{,}2\log_2 0{,}2+$$

$$+0{,}2\log_2 0{,}2][\,p(x_1)+p(x_2)\,]=1{,}97.$$

Здесь учтено $p(x_1)+p(x_2)=1$. $H(Y/X)$ не зависит от распределения вероятности входных сигналов $p(x_i)$, следовательно, справедливо соотношение (16.3), где $\max H(Y)=\log_2 4=2$. Тогда

$$\max H(Y)-H(Y/X)=2-1{,}97=0{,}03 \text{ бит/символ}$$

и $C=V_T[\max H(Y)-H(Y/X)]=20*0{,}03=0{,}6$ бит/с.

Важно подчеркнуть, что при наличии помех пропускная способность канала определяет наибольшее количество информации в единицу времени, которое может быть передано со сколь угодно малой вероятностью ошибки.

18. Основные понятия кодирования информации

Кодирование возможно только для дискретных сообщений, которые состоят из отдельных элементов – знаков. Множество всех знаков называют алфавитом, а число знаков – объемом алфавита. Код (фр. code произошло от лат. codex– свод законов) – совокупность знаков и системы правил, при помощи которых информация может быть представлена (закодирована) в виде набора этих знаков[6].

Кодирование – отождествление знаков или групп знаков исходного кода (сообщения) со знаками или группами знаков выходного кода. Знаки кода называют также буквами и символами. Чтобы различать, знаки исходного кода будем называть буквами, а знаки выходного кода символами. Декодирование – операция восстановления исходного сообщения.

Перечислим цели, которые может преследовать кодирование информации.

1) Представление информации в цифровом виде для обеспечения простоты, надежности и эффективности информационных устройств. Это означает простоту аппаратуры различения символов, минимальное время передачи или объем памяти для хранения, простоту выполнения арифметических и логических операций.

2) Уменьшение размера сообщений (сжатие). Это дает выигрыш в объеме запоминающего устройства или времени передачи, уменьшает энергетические и другие

затраты на работу с сообщением, повышает эффективность системы. Такое кодирование получило название эффективного (оптимального, экономного).

3) Повышение помехоустойчивости и достоверности при передаче или хранении информации. Другими словами – это возможность обнаруживать и исправлять всообщении ошибки определенного вида, которые создаются помехами. Такое кодирование получило название помехоустойчивого. Код, не обладающий помехоустойчивостью, называется простым.

4) Криптографическая защита информации от несанкционированного доступа. В этом случае код обычно называют шифром, а процессы кодирования и декодирования получили название шифрование и дешифрование.

5) Кодовое разделение сигналов для передачи нескольких сообщений по одному каналу связи.

6) Согласование сигнала и канала связи.

Отметим, что разные цели кодирования могут противоречить друг другу. Так сжатие уменьшает размер сообщения, а помехоустойчивое кодирование увеличивает его. На практике может реализовываться как одна из целей, так и несколько сразу. При этом за реализацию каждой цели отвечает свое устройство

Рассмотрим основные понятия кодирования.

Кодовая комбинация (кодовое слово) – последовательность символов кода, соответствующая букве или группе букв исходного алфавита. Для краткости словом «код» часто называют сами кодовые комбинации. Число символов в кодовой комбинации называется длиной кодовой комбинации. Например, в коде Windows 1251 букве «д» соответствует кодовая комбинация 11100100, имеющая длину 8.

Основу кода составляет кодовая таблица, устанавливающая соответствие между буквами и кодовыми комбинациями. Она может быть простой, если буквы и комбинации записываются построчно как в табл. 18.1, и перекрестной, когда передаваемая буква находится на пересечении строк и столбцов как в табл. 18.2.

Если длина кодовой комбинации одинакова для всех букв, то код называется равномерным. Например, ASCII (American Standart Code for Information Interchange– американский стандартный код для обмена информацией) – это семиэлементный равномерный код, международный телеграфный код МТК-2 (происходит от кода Бодо) – 5 элементный (см. табл. 18.1).

Таблица 18.1

Буква	Код ASCII	Код Морзе	Буква	Код ASCII	Код Морзе	Код Боде
1	0110001	.– – – –	B	1000010	– ...	10011
2	0110010	..– – –	Q	1010001	– – .–	11101
A	1000001	.– –	R	1010010	.–.	01010

Таблица 18.2

Первые 3 бита –> Код ASCII Последние 4 бита	0	1	1
	1	0	0
	1	0	1
0001	1	A	Q
0010	2	B	R

Если длина неодинакова, то код называется неравномерным, например, код Морзе (см. табл. 18.1), коды Шеннона–Фано и Хаффмана. По виду символов коды разделяют на нечисловые (код Морзе) и числовые или цифровые (ASCII). В дальнейшем рассматриваются только цифровые коды. Их разделяют по основанию кода – основанию системы счисления (объему алфавита).

Коды, у которых основание равно двум, называются двоичными (алфавит {0, 1}). Коды с основанием больше двух принято называть многоосновными.

Основные характеристики двоичных кодов. Вес кодовой комбинации w (вес Хэминга) – количество единиц в кодовой комбинации. Например, кодовая комбинация 0110010 имеет вес $w = 3$. Весовая характеристика кода – число кодовых комбинаций определенного веса.

Различие между двумя кодовыми комбинациями характеризуется расстоянием Хэмминга d (кодовым расстоянием). Оно равно числу разрядов, в которых комбинации отличаются одна от другой. Для вычисления кодового расстояния надо найти вес
суммы этих комбинаций по модулю два.

Сложение по модулю два (записывается в виде $a + b \pmod 2$ или $a \oplus b$) наиболее распространенная при кодировании и декодировании операция в двоичной СС, определяемая равенствами: $0 \oplus 0 = 0$; $0 \oplus 1 = 1$; $1 \oplus 0 = 1$; $1 \oplus 1 = 0$. В логике ее называют также неравнозначностью и «исключающее ИЛИ» (XOR).

Например: 1001111101

$\oplus$<u>1100001010</u>

010 1110111$\rightarrow w = 7$.

Значит, кодовое расстояние между комбинациями 1001111101 и 1100001010 $d = 7$.

19. Графическое представление кода

Геометрически кодовое расстояние для n–разрядных двоичных кодовых комбинаций удобно интерпретировать с помощью n–мерного куба с длиной ребра равной 1. Пусть кодовые комбинации являются координатами вершин этого куба. Тогда кодовое расстояние равно минимальному количеству ребер, которые необходимо пройти, чтобы перейти от одной комбинации к другой. Наглядно на рисунке это просто изобразить для трехразрядного кода ($n = 3$) и трехмерного куба (рис. 19.1).

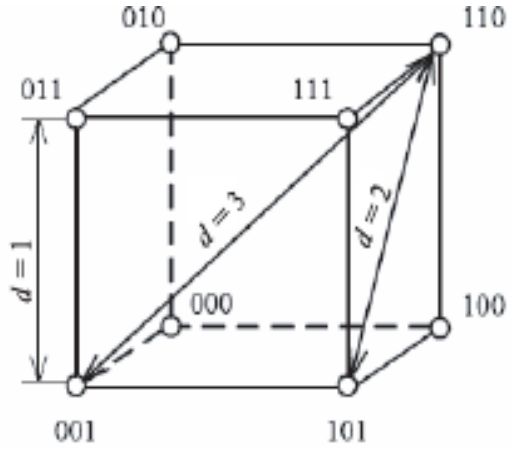


Рис.19.1. Трехмерный куб

Графическое представление кода. Во многих случаях наглядным и полезным для понимания методов и способов кодирования и декодирования является

представление построения кода в виде графа «дерево», его часто называют кодовое дерево. Для трехразрядного равномерного двоичного кода кодовое дерево приведено на рис. 19.2.

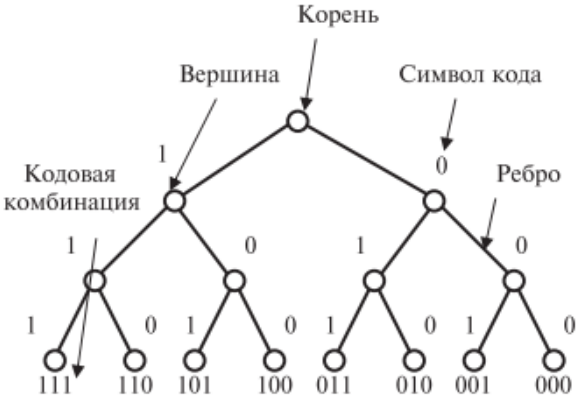


Рис.19.2. Кодовое дерево

Граф представляет собой набор точек (вершин), определенные пары которых соединены линиями (ребрами). Деревом называется связный граф, в котором каждую пару вершин соединяет только одна цепь (последовательность различных ребер). Начальная вершина дерева называется корнем. Число ребер, выходящее из вершины (по направлению от корня) равно основанию кода. Каждому ребру присваивается символ кода (на рис. 19.2 указан у вершины, куда ребро приходит). Запись кодовой комбинации осуществляется движением по ребрам от корня к выбранной вершине (путь).

Число ребер, которые нужно пройти от корня к некоторой вершине, называется порядком этой вершины. Максимальный порядок вершин равен разрядности равномерного кода или максимальной длине кодовых комбинаций.

20. Кодирование для представления информации в цифровом виде

В дискретном сообщении любому знаку, слову, кванту и т.п. можно сопоставить число, например, номер. Передача или хранение сообщений при этом сводится к передаче или хранению чисел. Это простой или первичный код. Код называется полным, если он использует все возможные комбинации.Способ представления (записи) числа с помощью цифр называется системой счисления (СС). Место, занимаемое цифрой, называется разрядом. СС подразделяются на позиционные (взвешенные) и непозиционные (невзвешенные). В соответствии с этим именуются и коды. В непозиционных СС значение цифры не зависит от ее позиции в числе, т.е. число — это просто набор цифр (римская СС). В позиционной СС значение цифры зависит от ее положения в числе. Например, в десятичном числе 22 первая цифра «2» – число десятков, вторая – число единиц. Основной формой представления целых чисел является истинная (натуральная) позиционная СС, в которой веса представляют собой последовательность возрастающих степеней основания СС *m*. Соответствующий код называется естественным или натуральным. Число Q в натуральной, позиционной СС записывается в виде[5]:

$$Q = a_1 a_2 a_3 ..... = \sum_{i=1}^{n} a_i m^{n-1} = a_i m^{n-1} + a_{i-1} m^{n-2} + ... a_2 m^1 + a_1 m^0, \qquad (20.1)$$

где *i*– номер разряда, a_i–цифры (показывают количество единиц *i*–го разряда, принимают значения от 0 до $m - 1$), *n*– количество разрядов (цифр) в числе. Единица каждого следующего разряда больше единицы предыдущего разряда в *m* раз. m_i–1 называют весом разряда. Например, в десятичном (m =10) числе $a_3 a_2 a_1 = 543 = 5 \cdot 10^2 + 4 \cdot 10^1 +$

$3 \cdot 10^0$ цифра $a_3 = 5$ стоит в третьем разряде ($i = 3$) с весом 10^2– «сотни». m единиц одного разряда объединяются в одну единицу следующего разряда.

Приведем пример разных натуральных кодов одного и того же числа Qи его представления в разных натуральных СС.Основание СС m указано нижним индексом после числа,

$F_{16}=15_{10}$, $E_{16}=14_{10}$.

$Q = FE91_{16} = 65169_{10} = 177221 = 1111111010010001_2$,

или через веса (20.1)

$Q = F \cdot 16^3 + E \cdot 16^2 + 9 \cdot 16^1 + 1 \cdot 16^0 = 6 \cdot 10^4 + 5 \cdot 10^3 + 1 \cdot 10^2 + 6 \cdot 10^1 + 9 \cdot 10^0 = 1 \cdot 8^5 + 7 \cdot 8^4 + 7 \cdot 8^3 + 2 \cdot 8^2 + 2 \cdot 8^1 + 1 \cdot 8^0 = 1 \cdot 2^{15} + 1 \cdot 2^{14} + 1 \cdot 2^{13} + 1 \cdot 2^{12} + 1 \cdot 2^{11} + 1 \cdot 2^{10} + 1 \cdot 2^9 + 0 \cdot 2^8 + 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$.

Видно, что чем больше основание СС, тем меньшее число разрядов требуется для представления данного числа, это уменьшает время для передачи числа и память для хранения. Однако при этом увеличивается количество используемых цифр (равно m), и существенно повышаются требования к линии связи и аппаратуре создания и распознавания элементарных сигналов, соответствующих различным цифрам. Логические элементы вычислительных устройств в этом случае должны иметь m устойчивых состояний, что сильно усложняет их конструкцию и уменьшает надежность.

Поэтому с точки зрения удобства физической реализации логических элементов и простоты выполнения в них арифметических и логических действий, предпочтение на практике отдается двоичной СС. Действительно, в этом случае логические элементы должны иметь всего два устойчивых состояния, а различения сигналов сводится к задаче обнаружения (есть или нет импульса), что значительно проще. Первым

применил двоичное кодирование в начале XVII в. английский философ Бэкон, разработавший пятиразрядный код с символами 0 и L. Двоичными кодами представления информации являются коды ASCII, Windows 1251, МТК-2 и др.

Однако двоичный код неудобен при вводе и выводе информации, так как запись двоичных чисел на бумаге или экране оказывается слишком громоздкой. Поэтому, помимо двоичной, получили распространение вспомогательные восьмеричная и шестнадцатеричная СС, которые легко сводятся к двоичной. Для перевода в восьмеричную СС каждые 3 бита двоичного числа (триады) справа налево заменяются на восьмеричную цифру. Для перевода в шестнадцатеричную СС каждые 4 бита двоичного числа (тетрады) заменяются на шестнадцатеричную цифру.

Обратный перевод очевиден.

$110101012 = 011\ 010\ 1012 = 325_8$, $11010101_2 = 1101\ 0101_2 = D5_{16}$.

21. Двоично-десятичный код, унитарный код, код Грея

Кроме того, достаточно трудоемок перевод десятичных чисел в двоичную форму, и в ряде случаев основное время может тратиться на перевод, а не на вычисления. Поэтому. иногда для сокращения времени перевода при обработке данных для представления десятичных чисел используются ненатуральные двоичные взвешенные коды, которые легко сводятся как к двоичной, так и к десятичной СС.

К таким кодам относится, например, двоично-десятичный код, в котором каждую цифру десятичного

числа записывают в виде четырехразрядного двоичного числа (тетрады). Причем при записи тетрад могут использоваться как натуральный двоичный код (т.е. с весами 8-4-2-1), так и ненатуральный (используются веса 2-4-2-1 – код Айкена, 5-1-2-1 и др.).

В ненатуральной позиционной (взвешенной) СС число *Q* записывается в виде:

$$Q = a_1 a_2 a_3 ..... = \sum_{i=1}^{n} a_i m^{n-1} = a_i C_n + a_{i-1} C_{n-1} + ... a_2 C_2 + a_1 C_1, \quad (21.1)$$

где C_n –значения весов.

Приведем примеры записи числа 962_{10} в двоично–десятичныхкодах с разными наборами весов

Исходное число	Набор весов	Двоично-десятичный код	Весовое представление (21.1) первой цифры
962_{10}	8-4-2-1	1001 0110 0010	9 → 1001, 9 = 1·8 + 0·4 + 0·2 + 1·1
	5-1-2-1	1111 1100 0010	9 → 1111, 9 = 1·5 + 1·1 + 1·2 + 1·1
	2-4-2-1	1111 0110 0010	9 → 1111, 9 = 1·2 + 1·4 + 1·2 + 1·1

Недостатками двоично-десятичных кодов является их большая избыточность (из 16 возможных комбинаций в тетраде используются только 10) и то, что выполнение арифметических операций становится более сложным. Достоинством взвешенных кодов является удобство их обработки с помощью ЭВМ. Из недостатков выделим два, особенно важных в системах аналого – цифровых преобразований.

1. Ошибка в разряде с большим весом существеннее, чем в разряде с меньшим весом.

2. При переходе к следующему числу иногда меняются сразу большое число разрядов, например, при переходе 0111–1000 (7–8) меняются все 4 разряда. Это приводит к значительному увеличению времени срабатывания устройства, что может быть недопустимо и приводить к ошибкам. Для устранения этих недостатков применяют невзвешенные коды.

Приведем примеры.

Унитарный код. В нем каждая цифра десятичного числа заменяется соответствующим числом единиц. Для неравномерного варианта: 62 → 111111 11. Для равномерного: 62 → 000111111 000000011.

В технике аналого-цифрового преобразования большое практическое значение имеют коды, у которых при переходе от одного числа к другому изменение происходит только в одном разряде. Это позволяет уменьшить влияние ошибки при неправильном считывании кода. Наибольшее распространение получил код Грея (табл. 21.1).

Таблица 21.1

Десятичное число	Код Грея	Десятичное число	Код Грея	Десятичное число	Код Грея
0	0000	6	0101	11	1110
1	0001	7	0100	12	1010
2	0011	8	1100	13	1011
3	0010	9	1101	14	1001
4	0110	10	1111	15	1000
5	0111				

Из-за особенности построения он также называется рефлексным (отраженным). Столбец старшего разряда

делится пополам и в верхнюю половину записываются нули, а в нижнюю – единицы. Этот уровень деления является «зеркалом» для следующего разряда, у которого делится пополам каждая половинка (всего этот столбец будет разделен на 4 части) и заполняется нулями и единицами так, чтобы нижняя половина столбца (из 2-х частей) была зеркальным отражением верхней. Процесс деления продолжается рекурсивно.

Недостатком кода Грея, как и всех непозиционных кодов, является сложность его обработки (проведения арифметических операций). Поэтому перед обработкой он обычно преобразуется в простой двоичный код.

Перевод кода Грея в двоичные числа производится по следующему алгоритму. Просматриваются цифры слева направо. Все цифры до первой единицы включительно не меняются,а последующие остаются без изменения, если число предшествующих единиц в коде Грея четно, иначе инвертируются ($0 \to 1$, $1 \to 0$). Например, $0111_{\Gamma} \to 0101_2$. Эту процедуру можно записать с помощью суммы по модулю два. Если a_i— цифра простого n–разрядного двоичного кода, аb_j– цифра кода Грея, то

$$a_i = \sum_{j=1}^{l} b_j \pmod 2,$$

Например, $0111_{\Gamma} = b_4b_3b_2b_1 \to a_4a_3a_2a_1 =$
$= b_4\ (b_4 \oplus b_3)(b_4 \oplus b_3 \oplus b_2)(b_4 \oplus b_3 \oplus b_2 \oplus b_1) = 0(0 \oplus 1)(0 \oplus 1 \oplus 1)$xx $(0 \oplus 1 \oplus 1 \oplus 1) = 0101_2$.

Любопытно, что для перевода в десятичную систему позициям кода Грея можно присвоить веса $C_i = (2^i - 1)(-1)^{i+1}$, тогда, например, пятиразрядный код имеет веса {31, –15, 7, –3, 1} и перевод $11011_{\Gamma} = 31–15+0–3+1 = 14_{10}$.

Существует еще цепные, циклические, рефлексные и другие коды.

22. Сущность эффективного кодирования

Эффективное кодирование – вид кодирования обеспечивающий компактное представление информации (минимальный объем), что позволяет уменьшить время передачи или память запоминающего устройства, уменьшить энергетические и другие затраты на работу с сообщением, повысить эффективность работы информационных систем. Отсюда и другое его название экономное кодирование. Получаемый код получил название оптимального кода.

Методы эффективного кодирования подразделяются на:

–**энтропийные** (статистические), в которых кодирование осуществляется на основе собранных о сообщении статистических данных. К ним относятся рассматриваемые ниже коды Шеннона–Фано, Хаффмана и арифметическое кодирование;

–**инкрементальные**или дифференциальные, которые осуществляют сжатие путем кодирования отличий в последовательных записях;

–**словарные**, опирающиеся на кодирование предложений, слов или сочетаний букв, которые содержатся в так называемом словаре.

Суть энтропийного кодирования заключается в том, что, учитывая статистические свойства источника сообщения, можно минимизировать среднее число символов, требующихся для выражения одного знака сообщения. Первыми появились методы, заключающиеся в кодировании букв кодовыми комбинациями переменной длины, которая имеет обратную зависимость от вероятности появления букв в сообщении. При этом наиболее вероятные буквы имеют наиболее короткие коды. Этот подход использовался еще в коде Морзе.

Шеннон показал, что оптимальная длина кодовой комбинации для буквы x_i равна $l_i = -\log_h p_i$, где p_i– вероятность появления буквы в сообщении, h– основание кода. При этом каждый символ кода несет максимальную информацию. Шеннон также доказал теорему, которая утверждает, что:

Сообщения, составленные из букв некоторого алфавита, можно закодировать так, что среднее число символов кода, приходящихся на букву сообщения, будет сколь угодно близко к энтропии источника, но не меньше этой величины. Эта теорема не описывает конкретного метода кодирования, но играет огромную роль в теории кодирования, поскольку определяет минимальную среднюю длину кодовой комбинации $L_{\min} = H$ (H– энтропия) и позволяет оценить эффективность разработанных кодов. Кроме того, ее доказательство указывает один из методов повышения эффективности: кодировать не сами буквы, а их сочетания (блоки). Минимум средней длины кодовой комбинации равный энтропии достигается при кодировании блоков, содержащих бесконечное количество букв.

Далее рассматривается двоичное кодирование, что соответствует использованию при вычислении энтропии двоичного логарифма.

Введем следующие характеристики кодов.

Средняя длина кодовой комбинации $L > L_{\min} = H$:

$$L = \sum_{i=1}^{n} p(x_i)L_i = p_1L_1 + p_2L_2 + \ldots . p_nL_n, \qquad (22.1)$$

где x_i– буква (блок) номер i исходного сообщения;$p_i = p(x_i)$ – ее вероятность;L_i–длина кодовой комбинации; n– количество букв в алфавите источника (блоков).

Величину L также называют стоимостью кодирования. Качество кода характеризуется близостью L

к $L_{\min} = H$. Для его оценки вводится характеристика, называемая эффективность кода:

$$\chi = \frac{L_{\min}}{L} = \frac{H}{L}. \qquad (22.2)$$

Кроме того, вводится характеристика избыточность кода, показывающая насколько данный код длиннее минимально возможного:

$$R = \frac{L - L_{\min}}{L} = 1 - \frac{L_{\min}}{L} = 1 - \frac{H}{L} = 1 - \chi. \qquad (22.3)$$

Обе величины могут записываться в процентах. Если осуществляется переход от первичного кода к оптимальному с целью уменьшения объема сообщения, то говорят о сжатии информации. Степень сжатия можно оценить коэффициентом сжатия *r*, определяемым как отношение объема информации до сжатия к объему после сжатия. Операцию сжатия файлов, предназначенных для хранения во внешней памяти компьютера, часто называют архивацией данных, а соответствующие программы называют архиваторами.

23. Методы сжатия информации

Все методы сжатия информации подразделяются на две группы:

Первая – методы сжатия без потери информации. Это означает, что после декодирования будет полностью восстановлено исходное сообщение. Такое сжатие называют обратимым. К этой группе относятся указанные выше методы эффективного кодирования (энтропийные, дифференциальные, словарные).

Вторая – методы сжатия с регулируемой потерей информации. Они дают значительно лучшее сжатие

(иногда в десятки раз), однако необратимы, т.е. при декодировании не происходит восстановления исходного сообщения. Поэтому они могут применяться только к тем видам данных, для которых потеря части информации не приводит к существенным ошибкам и потере потребительских свойств. Совершенно очевидно, что к программам, текстам, бухгалтерским данным и т.п. они не применимы. В первую очередь они используются для сжатия полноцветной графической информации (рисунки и фотографии), звуковых и видеозаписей, где для полноценного восприятия человеком нет необходимости передавать всю информацию точно.

По работе с информацией выделяют алгоритмы, работающие в реальном времени, в потоке и требующие накопления информации. Если алгоритм просматривает кодируемое сообщение один раз, то его называют однопроходным. Понятно, что алгоритмы, работающие в реальном времени, – однопроходные. Алгоритмы, подстраивающиеся под конкретное сообщение называются адаптивными. Например, кодирование Хаффмана может быть однопроходным адаптивным или двухпроходным.

Сжатие может осуществляться последовательным применением нескольких методов. Это может оказаться более эффективным, поскольку каждый метод сжатия ориентирован на свой тип данных.

24. Алгоритм построения кода Шеннона-Фано

Методы построения энтропийных эффективных кодов впервые были даны американскими учеными Шенноном и Фано для случая отсутствия статистической взаимосвязи между буквами. Их методики существенно не различаются, и поэтому соответствующий код получил

название кода Шеннона – Фано. Здесь алгоритм дан в формулировке Фано.

Алгоритм построения кода Шеннона – Фано: все буквы алфавита сообщения выписывают в порядке убывания вероятностей. Затем их разделяют на две группы так, чтобы суммы вероятностей в каждой из групп как можно меньше отличались друг от друга. Всем буквам верхней группы в качестве первого символа кода приписывают 1, а нижней – 0. Каждую из полученных групп, в свою очередь, разбивают на две подгруппы с приблизительно одинаковыми суммарными вероятностями и добавляют справа по тому же правилу следующий символ кода и т.д. Процесс повторяется до тех пор, пока в каждой подгруппе останется по одной букве.

Примечание. Можно верхней группе приписывать 0, а нижней 1, и добавлять не справа, а слева. Главное, чтобы эти операции были одинаковыми для всех разбиений.

При таком подходе, чем более вероятна буква, тем быстрее она останется одна, и тем короче будет ее код. А именно этого и надо добиться.

Пример 24.1. Построим код Шеннона –Фано для алфавита из восьми букв x_i с заданными вероятностями появления p_i(табл. 24.1). Сравним его с простым равномерным двоичным кодом (РДК).

Решение. Исходные данные и результаты сводятся в табл. 24.1). Каждое разбиение показано подчеркиванием. Кодовое дерево построено на рис 24.1. На нем хорошо видна работа алгоритма. Из корня строят два ребра, соответствующие разбиению алфавита на две почти равновероятные группы. Ребро, соответствующее верхней группе, помечаем 1, а нижней – 0 (для наглядности группы приведены рядом).

Таблица 24.1

Буква	p_i	Код Шеннона — Фано					L_i	РДК
x_1	0,40	1	**1**				2	000
x_2	0,20	1	**0**				2	001
x_3	0,15	0	1	**1**			3	010
x_4	0,10	0	1	**0**			3	011
x_5	0,08	0	0	**1**			3	100
x_6	0,04	0	0	0	**1**		4	101
x_7	0,02	0	0	0	0	**1**	5	110
x_8	0,01	0	0	0	0	0	5	111

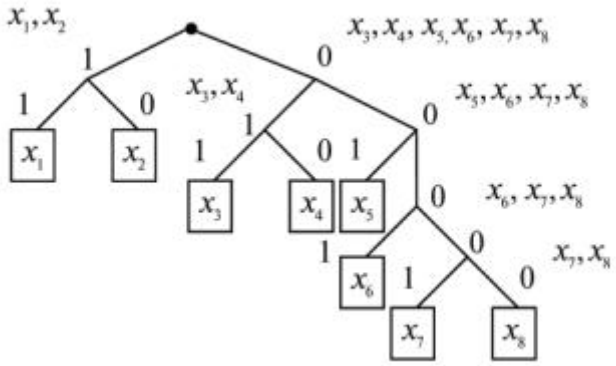


Рис.24.1. Кодовое дерево

Из конца каждого ребра снова строим по два ребра и т.д. Процесс построения заканчивается, когда в группе не останется по одной букве. Каждая концевая вершина графа соответствует своей букве, а путь к ней — кодовому слову.

Рассчитаем характеристики полученного кода. Энтропия алфавита (10.5) равна

$$H = -\sum_{i=1}^{k} p_i \log_2 p_i = -(0{,}4\log_2 0{,}4 + 0{,}2\log_2 0{,}2 + \ldots.$$
$$+ 0{,}01\log_2 0{,}01 = 2{,}392.$$

Таким образом, из теоремы Шеннона следует возможность разработать код со средней длиной кодовой комбинации приближающейся к 2.392 символов. Средняя длина кодовой комбинации (22.1)

$$L = \sum_{i=1}^{n} p(x_i)L_i =$$

=0,4·2+0,2·2+0,15·3+0,1·3+0,08·3+0,04·4+0,02·5+ ,01·5=2,5.

Эффективность кода (22.2) $\chi = H/L$ = 2,392/2,5 = 0,957 (95,7%) и избыточность (22.3) $R = 1 - \chi = 1 - 0{,}957$ = 0,043 (4,3%). Обычный равномерный двоичный код (РДК) имеет все кодовые комбинации одинаковой длины — 3 символа (последняя колонка в табл. 22.1), следовательно, L = 3. Его эффективность χ = 2,392/3 = 0,797 (79,7%) и избыточность R = 1 – 0,797 = 0,203 (20,3%). Таким образом, оптимальный код Шеннона – Фано дает значительный выигрыш (более 15% в длине сообщения) по сравнению с РДК. Этого и следовало ожидать.

Тем не менее, некоторая избыточность у кода Шеннона –Фано все же осталась (4,3%). Ее тоже можно устранить, если перейти к кодированию достаточно большими блоками. Чтобы продемонстрировать это, рассмотрим пример с меньшим размером алфавита (тогда количество блоков не будет слишком большим).

Пример 24.2. Сконструировать код Шеннона – Фано для алфавита из двух букв x_1 и x_2 с вероятностями появления $p(x_1)$ =

= p_1 = 0,9 и p(x_2) = p_2 = 0,1.

Решение. Энтропия такого алфавита: $H(X) = -0{,}9\log_2 0{,}9 - 0{,}1\log_2 0{,}1 = 0{,}47$.

При кодировании по одной букве первая буква будет кодироваться знаком 1, вторая — знаком 0. Тогда $L = 1$, эффективность $\chi = 0{,}47/1 = 0{,}47$ (47%) и избыточность $R = 1 - 0{,}47 = 0{,}53$ (53%). Такой код очень прост, но очень неэффективен.

Произведем кодирование блоков, содержащих по две буквы. По теореме об умножении вероятностей независимых событий вероятность появления блока (сочетания букв) равна произведению вероятностей букв $p(x_i x_j) = p_i p_j$. Тогда, получаем коды, приведенные в табл. 24.2.

Среднее число символов на блок: $L_{бл} = 0{,}81 \cdot 1 + 0{,}09 \cdot 2 + 0{,}09 \cdot 3 + 0{,}01 \cdot 3 = 1{,}29$.

Таблица 24.2

Блоки	Вероятности	Код Шеннона – Фано			L_i
$x_1 x_1$	$p_1 p_1$=0,81	1			1
$x_1 x_2$	$p_1 p_2$=0,09	0	1		2
$x_2 x_1$	$p_2 p_1$=0,09	0	0	1	3
$x_2 x_2$	$p_2 p_2$=0,01	0	0	0	4

Найдем энтропию блоков. Для независимых букв в соответствии с (12.5) получаем $H_{бл} = H(X, X) = H(X) + H(X) = 2H(X) = 2 \cdot 0{,}47 = 0{,}94$. Эффективность кода $\chi = H_{бл}/L_{бл} = 0{,}94/1{,}29 = 0{,}729$ (72,9%), а избыточность: $R = 1 - 0{,}729 = 0{,}271$ (27,1%).

Это значительно лучше, чем при кодировании по одной букве.Заметим, что для независимых букв вместо расчета $H_{бл}$ можно вычислить L, приходящееся на одну букву$L = L_{бл}/2 = 0{,}645$, и использовать $H(X)$. Тогда $\chi = 0{,}47/0{,}645 = 0{,}729$. Видно, что здесь Lближе к энтропии,

чем при кодировании по однойбукве, что и делает эффективность такого кода значительно выше.

Выигрыш достигается за счет того, что наиболее часто встречающийся блок x_1x_1 (его вероятность 0,81, следовательно, он встречается в среднем в 81 случае из 100) кодируется всего одним символом, а остальные блоки в среднем слишком редко встречаются, чтобы оказать существенное влияние на длину.

Пусть, например, мы передаем 1000 букв (500 блоков по две буквы), тогда при кодировании по одному биту $\{x_1 - 1, x_2 - 0\}$, на все сообщение потребуется 1000 бит. Блок x_1x_1 при этом в среднем встречается $0{,}81 \cdot 500 = 405$ раз и при кодировании одним битом занимает 405 бит вместо 810 бит. Выигрыш составляет 405 бит! Блоки x_1x_2 и x_2x_1 встречаются по $0{,}09 \cdot 500 = 45$ рази кодируются соответственно $45 \cdot 2 = 90$ бит (выигрыша нет) и $45 \cdot 3 = 135$ бит (проигрыш 45 бит). И, наконец, блок x_2x_2 встречается $0{,}01 \cdot 500 = 5$ раз и кодируется $5 \cdot 3 = 15$ бит (проигрыш 5 бит). Итак, мы получили выигрыш в 355 бит, и наше сообщение вместо 1000 бит будет иметь длину 645 бит — произошло сжатие на треть. Однако энтропия $H = 0{,}47$, говорит о том, что в идеале можно закодировать это сообщение 470 битами. Таким образом и получаются, вычисленные выше эффективность кода

$\chi = 470/645 = 0{,}729$ (72,9%) и избыточность $R = (645 - 470)/645 = 0{,}271$ (27,1%).

Тем не менее, избыточность кода все еще очень велика. Чтобы уменьшить ее, произведем кодирование блоков, содержащих по три буквы. Результат приведен в табл. 24.3 (учтено, что $p(x_ix_jx_k) = p_ip_jp_k$).

Таблица 24.3

Блоки	p	Код Шеннона – Фано					L_i
$x_1\,x_1\,x_1$	0,729	1					1
$x_1x_1\,x_2$	0,081	0	1	1			3
$x_1x_2\,x_1$	0,081	0	1	0			3
$x_2\,x_1\,x_1$	0,081	0	0	1			3
$x_1x_2\,x_2$	0,009	0	0	0	1	1	5
$x_2x_1\,x_2$	0,009	0	0	0	1	0	5
$x_2x_2\,x_1$	0,009	0	0	0	0	1	5
$x_2x_2\,x_2$	0,001	0	0	0	0	0	5

Среднее число символов на блок: $L_{бл}$ = 0,729·1 + 0,081·3 ++ 0,081·3 + … + 0,001·5 = 1,59, на букву приходится $L = L_{бл}/3$ = 0,53 (энтропия $H_{бл} = 3H(X)$). Это еще ближе к энтропии и эффективность такого кода еще выше: χ = 0,47/0,53 = 0,887 (88,7%), избыточность R = 1 – 0,887 = 0,113 (11,3%). Выигрыш достигается за счет того, что наиболее часто встречающийся блок $x_1x_1x_1$ кодируется всего одним символомУвеличивая число букв в блоках, можно еще больше приблизиться к теоретическому минимуму. Однако значительно увеличивать число букв в блоках невыгодно. С каждой буквой выигрыш все меньше, а количество разных блоков все больше (особенно при большом алфавите). Это сильно увеличивает время кодирования и декодирования и требует накопления букв перед кодированием, что особенно нежелательно при передаче данных в режиме реального времени.

25. Алгоритм построения кода Хаффмана

Очевидно, что метод Шеннона – Фано может давать не самый эффективный код, так как наиболее короткие коды могут соответствовать не только самым вероятным буквам, но и первым во второй подгруппе. От этого

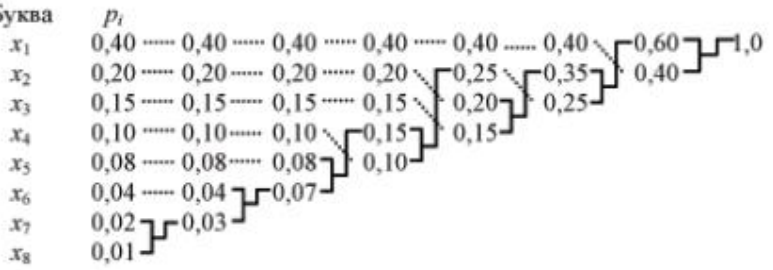
недостатка свободен более эффективный метод Д.А. Хаффмана, предложенный им несколько позднее (1952 г.) также для случая отсутствия статистической взаимосвязи между буквами.

Классический алгоритм Хаффмана: буквы (блоки) располагают в порядке убывания их вероятностей. Складывают вероятности двух последних блоков (с наименьшей вероятностью), и добавляют результат к списку вероятностей, одновременно удаляя те, которые складывались. Вероятности опять располагают в порядке убывания. Процедуру повторяют, пока не останется одна вероятность равная 1. Затем для получения кодовой комбинации строится кодовое дерево (*H*–дерево), отслеживающее объединение вероятностей, начиная от 1 и кончая исходными вероятностями. Двигаясь по кодовому дереву сверху вниз, можно записать для каждой буквы соответствующую ей кодовую комбинацию, присваивая ветви с меньшей вероятностью 0, а с большей – 1.

Пример 25.1. Построить код Хаффмана для алфавита из восьми букв x_iс заданными вероятностями появления p_i из примера 2.1. (При двоичном кодировании каждый символ кодируется байтом (восемь цифр 1 и 0). Сколько символов можно закодировать?).

Решение. Строим табл. 25.1 объединений вероятностей в соответствии с алгоритмом Хаффмана.

Таблица 25.1



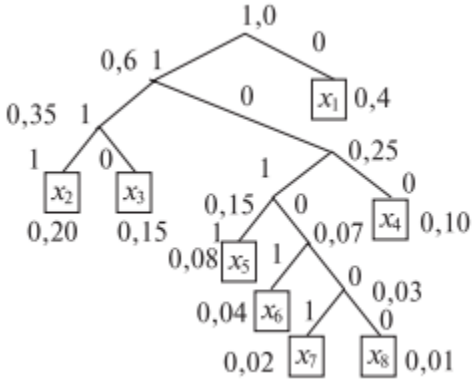
Теперь строим кодовое дерево (рис. 25.1), следуя линиям в табл.25.1, и записываем кодовые комбинации, двигаясь от вероятности 1,0 вниз к каждой из букв.

Рассчитаем характеристики полученного кода. Энтропия алфавита по-прежнему равна H = 2,392, средняя длина кодовой комбинации (22.1)

$$L = \sum_{i=1}^{n} p(x_i)L_i =$$

0,4·1+0,2 ·3+0,15 ·3+0,15 ·3+ 0,1·3+0,08·4+0,04 ·5+ 0,02·6+ 0,06·5=2,45.

Эффективность кода (22.2) χ = 2,392/2,45 = 0,977 (97,7%) и избыточность R = 1 – 0,977 = 0,023 (2,3%). Это лучше чем у кода Шеннона – Фано для того же примера.



Буква	Код	L_i
x_1	0	1
x_2	111	3
x_3	110	3
x_4	100	3
x_5	1011	4
x_6	10101	5
x_7	101001	6
x_8	101000	6

Рис. 25.1. Кодовое дерево Хаффмана и таблица кода

Доказано, что алгоритм кодирования Хаффмана дает самый оптимальный код. Поэтому он стал базой для огромного количества дальнейших исследований в этой области. Даже спустя 50 лет можно найти большое количество публикаций, посвященных как различным его

реализациям, так и поискам лучшего применения. Модифицированные разновидности алгоритма Хаффмана в сочетании с рядом других методов лежат в основе многих современных программ сжатия (стандартов JPEG, MPEG, H.261 и др.) и даже встроены в некоторые телефаксы.

Классический алгоритм Хаффмана имеет один существенный недостаток. Для восстановления содержимого закодированного сообщения декодер должен знать таблицу вероятностей (частот), которой пользовался кодер. Следовательно, длина сжатого сообщения увеличивается на длину таблицы частот, которая должна посылаться впереди данных, что может свести на нет все усилия по сжатию сообщения. Кроме того, необходимость наличия полной частотной статистики перед началом собственно кодирования требует двух проходов по со общению: одного для построения таблицы частот, другого для кодирования.

В настоящее время используется так называемый адаптивный метод Хаффмана, использующий один проход без предварительного получения статистики. Он заключается в том, что после считывания очередной буквы происходит перерасчет частоты ее появления и соответственно изменяется ее код. Алгоритм как бы приспосабливается (адаптируется) к сообщению.

Если сообщение достаточно длинное, то происходит быстрая стабилизация частот появления букв, и эффективность адаптивного метода становится ненамного хуже классического.

26. Префиксные коды. Алгоритм арифметического кодирования

Широко распространенные равномерные двоичные коды (ASCII, Windows 1251 и др.) не являются оптимальными, но имеют огромное достоинство – они быстро и просто декодируется. Для этого последовательность 1 и 0 просто разбивается на блоки по 8 символов, а затем эти блоки по стандартной таблице кодов декодируются. Это достоинство и обуславливает их широкое распространение.

Оптимальные коды имеют различные длины кодовых комбинаций для разных букв и не стандартны, поскольку вероятности появления букв меняются от сообщения к сообщению. Это затрудняет их декодирование, поскольку необходимо определять, где кончается одна кодовая комбинация и начинается другая. Конечно, для этого можно использовать специальный разделительный символ, но это удлиняет сообщение, снижая эффективность кодирования, так как средняя длина кодовой комбинации увеличивается на символ. Более целесообразно обеспечить однозначное декодирование без введения дополнительных символов (такой код иногда называют «код без запятой»). Для этого эффективный код должен быть префиксным.

Префиксный код – это код, у которого ни одна кодовая комбинация не совпадает с началом другой (более длинной). С точки зрения кодового дерева префиксность означает, что ни одна из промежуточных вершин не соответствует коду, только концевые.

Нетрудно убедиться, что коды Шеннона –Фано и Хаффмана – префиксные, т.е. являются однозначно декодируемыми «без запятой».

Для декодирования сообщения можно использовать таблицу кода, но на практике предпочитают использовать кодовое дерево (см. пример 25.1). При декодировании происходит движение по ветвям дерева от 1 к буквам или блокам (на приведенном дереве сверху вниз) в соответствии с кодом: если в коде стоит 1, то выбирается ветвь с 1 и т.д. При достижении буквы она считается декодированной, и движение возобновляется от 1. И так до конца сообщения.

Пример 26.1. Используя кодовое дерево из примера 25.1 декодировать сообщение 10001010011011.

Решение: 1–0–0–x_4; 0–x_1; 1–0–1–0–0–1–x_7; 1–0–1–1–x_5;

Ответ: $x_4x_1x_7x_5$.

Примером непрефиксного кода служит, в частности, такой код: А – 0, Б – 1, В – 11, Г – 111. Попробуйте ответить, что закодировано в сообщении 1111011?

Префиксность накладывает ограничения на набор длин кодов и не позволяет брать кодовые комбинации слишком короткими. Для префиксного кода объема N с длинами комбинаций l_1, l_2,…,l_N должно быть справедливо неравенство Крафта

$$\sum_{i=1}^{n} 2^{-l_i} \le 1 \qquad (26.1)$$

Однозначно декодируемыми «без запятых» могут быть не только префиксные коды. Примером является непрефиксный код {01, 10, 011}, в котором комбинация 01 является началом 011. Но тогда сильно усложняется логика декодирования, поскольку приходится анализировать последующие символы, а то и все сообщение. Например, в сообщении 011010011 первым кодом может быть и 01, и 011. Для декодирования

придется перебирать варианты. Пусть первый код 011, тогда второй однозначно 01 (коды 0 и 010 не существуют), но далее следует два нуля, которые не соответствуют никакому коду. Следовательно, первый код 01. И так далее.

Арифметическое кодирование. Каждая кодовая комбинация, полученная методом Хаффмана, содержит целое число бит. Таким образом, при кодировании по одной букве на каждый символ сообщения затрачивается не менее одного бита (минимум равный 1 бит достигается при кодировании двух букв 0 и 1). Однако энтропия источника часто значительно меньше 1. Как показано в п.24 для повышения эффективности кода можно использовать блочное кодирование. Однако практически это не выгодно, поскольку при большом количестве символов количество блоков очень велико, что сильно увеличивает трудоемкость алгоритма и затрудняет построение дерева.

Одним из лучших методов кодирования, позволяющих кодировать символы менее чем одним битом, является арифметическое кодирование.

Арифметическое кодированиеосновано на преобразование исходной последовательности букв в число из интервала [0, 1), однозначно определяющее эту последовательность, которое затем кодируется двоичным кодом.

Для этого определяется интервал, в котором расположено это число, по следующему алгоритму. Исходный интервал [0,1), длина которого равна 1, разбивается на части, каждая из которых соответствует одной букве S_iалфавита сообщения, и ее длина составляет долю от длины всего интервала, равную вероятности появления p_iэтой буквы (для единичного интервала длины частей равны вероятностям). Выбирается интервал

соответствующий первой букве. Затем он аналогично разбивается на части, и из них выбирается интервал, соответствующий вто рой букве сообщения. Теперь он разбивается на части и берется часть, соответствующий третьей букве, и т.д. Процесс разбиения продолжается до конца сообщения. Обратите внимание, каждая последующая буква уменьшает длину рассматриваемого интервала пропорционально вероятности своего появления. Запишем соответствующие формулы.

Пусть нам известны вероятности $p_n = p(x_n)$ всех букв x_n алфавита сообщения. Обозначим верхнюю и нижнюю границы интервала для текущей буквы $S_i = x_n$ сообщения за H_i и L_i (i– номер буквы в сообщении, n– в алфавите). Тогда границы H_i и L_i можно найти по рекуррентным формулам:

$$L_i=L_{i-1}+Sp_{n-1}\cdot(H_{i-1}-L_{i-1}),H_i=L_{i-1}+Sp_n\cdot(H_{i-1}-L_{i-1}), \quad (26.2)$$

где $L_0 = 0$, $H_0 = 1$, $Sp_0 = 0$.

Кумулятивная вероятность Sp_n определяется формулой:

$$Sp_n = \sum_{i=1}^{n} p_k . \quad (26.3)$$

Интервал, соответствующий последней букве является искомым. Любое число ему принадлежащее однозначно определяет все сообщение. Для длинных последовательностей выбор какого-либо из возможных чисел слабо влияет на качество кодирования, поэтому в качестве определяющего числа часто берется левая граница интервала. Можно поступить по другому. Найти число вида $x/2^m$ (x – целое число, m – наименьшая возможная степень двойки), принадлежащее искомому интервалу, тогда в качестве определяющего числа можно взять x. Этот подход дает самый оптимальный код. Поиск числа x удобно

произвести следующим образом: запишем условие

$$L_i = \frac{x}{2^m} < H_i \text{ или } 2^m L_i \le x < H_i. \quad (26.4)$$

Перебирая m, начиная с единицы, добиваемся, чтобы интервал (26.4) содержал хотя бы одно целое число, которое и будет искомым.

Кодируя выбранное число в двоичной системе, после отбрасывания нулевой целой части и десятичной точки получают окончательный код сообщения.

Пример 26.1. Сообщение передается алфавитом из четырех букв (табл. 26.1). Закодировать арифметическим кодом сообщение «габ».

Таблица 26.1

№ буквы (n)	Буква алфавита (Xn)	Вероятность появления (p_n)	Sp_n	Части единичного (исходного) интервала	
1	А	0,3	0,3	[0,0,3)	0 1
2	Б	0,2	0,5	[0,3,0,5)	1 2
3	В	0,1	0,6	[0,5,0,6)	2 3
4	Г	0,4	1	[0,6,1)	3 4

Первая буква сообщения «г» (i = 1, n = 4). Используя формулы (26.2)–(26.3), получаем

$L_1=L_0+Sp_3\cdot(H_0–L_0)$= 0,6+0·(1–0,6)=0,6,

$H_1=L_0+Sp_4\cdot(H_1–L_1)$ =0+1·(1–0)=1,

Получили интервал [0,6, 1). Вторая буква сообщения «а» (i = 2, n = 1).

$L_2=L_1+Sp_0\cdot(H_0–L_0)$= 0,6+0 ·(1–0,6)=0,6,

$H_2=L_0+Sp_1\cdot(H_1–L_1)$ =0,6+3·(1–0,6)=0,72.

Получили интервал [0,6, 0,72). Третья буква сообщения «б» (i = 3, n = 2).

$L_3=L_2+Sp_1\cdot(H_2–L_2)$= 0,6+0,3 ·(0,72–0,6)=0,636,

$H_3=L_2+Sp_2\cdot(H_2–L_2)$ =0,6+0,5·(0,72–0,6)=0,66.

Получили искомый интервал [0,636, 0,66).

Для перевода дробной части числа в двоичный код последовательно умножаем дробную часть на основание 2, запоминаем полученные целые части и считываем их сверху вниз.

Пример

0,636*2=1,272 –>1
0,272*2=0,544–>0
0,544*2=1,088–>1
0,088*2=0,176–>0
0,176*2=0,352–>0
0,352*2=0,704–>0
0,704*2=1,4 –>1

Считываем полученные числа сверху вниз получили код: 1 0 1 0 0 0 1

В качестве искомого можно взять любое число из интервала [0,636, 0,66), например, 0,636. Переводим его в двоичный код: 0,1010001…. Он бесконечен, поэтому его надо обрезать на таком разряде, чтобы число попадало в заданный интервал (если последним будет 0, его заменяют на 1). Легко убедиться, что надо оставить пять разрядов после запятой. Получим 0,10101 (0,65625). Окончательный арифметический код получим, убрав ноль и точку, 10101.

Найдем код вторым методом на основе неравенства (26.4), которое в данном случае имеет вид $0{,}636 \cdot 2^m < x < 0{,}66 \cdot 2^m$. Перебираем $m = 1,2,3\ldots$, пока неравенству не станет удовлетворять какое–нибудь целое число x. Получим $m = 5$ и $20{,}3 < x < 21{,}1$ и $x = 21$. Таким образом, сообщение может быть закодированос помощью $m = 5$ бит, а его кодом является число $x = 21$, в двоичной системе равное 10101.

Итак, арифметическим кодом сообщения «габ» является 10101. При декодировании разбиение интервалов производится таким же образом. Только теперь

определяется интервал, в который попадает код и выписывается соответствующая буква.

Так для полученного выше кода 101012 = 2110, m = 5 разрядов записываем соответствующее число из единичного интервала

$\frac{x}{2^m} = \frac{21}{2^5} = 0{,}65625$ и проверяем, в интервал какой буквы оно попадает. Для первой буквы: 0,65625 ∈ [0,6, 1), следовательно, это «г». Для второй буквы: 65625 ∈ [0,6, 0,72), следовательно, это «а». И, наконец, для третьей буквы: 0,65625 ∈ [0,636, 0,66), следовательно, это «б». В итоге, получаем –«габ».

Реальную программную реализацию декодирования удобно строить на факте, что в силу способа задания длин отрезков выражение

$x_n = \frac{x - L_i}{H_i - L_0}$ проектирует разбиение текущего интервала на исходный единичный. Это позволяет при декодировании определять значения букв по попаданию x_nв соответствующую часть исходного единичного интервала (см. последнюю колонку таблицы 26.1). H_i и L_iпо-прежнему находятся по формулам (26.2). Например, берем уже рассмотренное сообщение 10101 с x = 0,65625.

$$\frac{x - L_0}{H_0 - L_0} = \frac{0{,}65625 - 0}{1 - 0} \approx 0{,}65625 - \text{«г»},$$

$$\frac{x - L_1}{H_1 - L_1} = \frac{0{,}65625 - 0{,}6}{1 - 0{,}6} \approx 0{,}14 - \text{«а»},$$

$$\frac{x - L_2}{H_2 - L_2} = \frac{0{,}65625 - 0{,}6}{0{,}72 - 0{,}6} \approx 0{,}47 - \text{«б»}.$$

Пример 26.2.

1) Используя свои имя и фамилию, определите вероятности появления букв. Используя их, закодируйте первые 4–5 букв.

Решение:

Закодируем «агв».

$L_1=L_0+Sp_0\cdot(H_0-L_0)= 0+0\cdot(1-0)=0,$

$H_1=L_0+Sp_1\cdot(H_0-L_0) =0+0,3\cdot(1-0)=0,3,$

Получили интервал [0, 0 ,3).

$L_2=L_1+Sp_3\cdot(H_1-L_1)= 0+0,6\cdot(0,3-0)=0,18,$

$H_2=L_1+Sp_4\cdot(H_1-L_1) =0+1\cdot(0,3-0)=0,3.$

Получили интервал [0,18, 0,3).

$L_3=L_2+Sp_2\cdot(H_2-L_2)= 0,18+0,5\cdot(0,3-0,18)=0,24,$

$H_3=L_2+Sp_3\cdot(H_2-L_2) =0,18+0,6\cdot(0,3-0,18)=0,252.$

Получили интервал [0,24, 0,252).

Получиликод: 0,25*2=0,5–>0,

0,5*2=1–>1. (0 1).

2) Используя данные табл. 26.1 декодировать сообщение 01.

***Решение*:**

$01_2=1_{10}$, m =2 разряда. Соответствующее число из единичного интервала:

$$\frac{x}{2^m}=\frac{1}{2^2}=0,25;$$

0,25∈ [0, 0,3)–> первая буква «а»,

0,25∈ [0,18, 0,3)–>вторая буква «г»,

0,25∈ [0,24, 0,252)–>третья буква «в»,

$$\frac{x-L_0}{H_0-L_0}=\frac{0,25}{1-0}=0,25-\text{>"а"}$$

$$\frac{x-L_2}{H_1-L_1}=\frac{0,25-0,0}{0,3-0}=0,83-\text{>"г"}$$

$$\frac{x-L_2}{H_2-L_2}=\frac{0{,}25-0{,}18}{0{,}3-0{,}18}=0{,}58\text{- > "в"}$$

(Ответ: «агв»).

27. Методы сжатия с потерей информации

Сначала о некоторых методах сжатия без потерь.

Метод дифференциального кодирования. Используется для тех данных, у которых разница между соседними значениями значительно меньше самих значений. Например, имеем последовательность из 8 чисел {221, 223, 226, 224, 229, 228, 228, 229}. Ее можно переписать в следующем виде: первое число оставляем без изменений, а вместо остальных чисел записываем разницу с предыдущим, т.е. 223 → 223–221 = 2, 226 → 226–223 = 3 и т.д. Получаем {221, 2, 3, –2, 5, –1, 0, 1}, что значительно короче. При двоичном кодировании каждое число кодировалось бы 1 байтом = 8 бит, т.е. вся последовательность 8·8 = 64 битами. Небольшие разности можно кодировать меньшим числом бит. В нашем случае разница по модулю не превышает 5, следовательно, ее можно кодировать 4 битами (1 бит — знак, 3 бита — значение). Тогда вся последовательность имела бы длину8 + 7·4=36 бит. Получили сжатие почти в 2 раза. Подобная ситуация имеет место при сжатии звука, особенно речи, изображений с плавными изменениями яркости и цвета, видеоизображений.

В настоящее время очень распространенными являются словарные методы кодирования. Словарь — список наборов букв (фраз), которые встречаются в сообщении. При словарном кодировании происходит замена этих фраз ссылками на словарь (индексами). Этот тип сжатия еще известен как макрокодирование или метод

книги кодов. Если словарь составлен заранее, то говорят о статичном кодировании, если свой словарь строится для каждого сообщения, то полуадаптивном кодировании, а если словарем является уже просмотренная часть сообщения, то об адаптивном кодировании. Словарь прикладывается к сообщению.

Классическим статичным словарным методом является кодирование по ключевым словам KWE (Key Word Encoding). Метод эффективен для больших англоязычных текстов

Для русского языка он, обычно, малоэффективен из-за обилия приставок, суффиксов и окончаний. Интересны диадные коды, где кодируются распространенные пары символов (диады). Они используют тот факт, что имеется только 96 текстовых символов (94 печатных, пробел и код для новой строки), а при использовании 8-битного кодирования (ASCII, Windows 1251) возможны 256 кодовых комбинаций. Одиночные символы представлены их обычными кодами, оставшиеся 160 кодов представляют диады. Их может быть и больше, если не все из 96 символов используются. Сжатие происходит благодаря кодированию диад одним байтом вместо двух. Сжатие невелико, но простота кода (одинаковый размер комбинаций, отсутствие операций с битами внутри байтов) обеспечивает большую скорость диадного кодирования.

Одним из самых простых является метод кодирования длин серий RLE (RunLengthEncoding). Вместо последовательности из N одинаковых элементов идущих подряд, записывается первый элемент этой последовательности и число его повторений ($N - 1$). Например: AAAAABCCDDDDDDD = A4B0C1D6.

Вместо исходных 15 байтов получили 8 (коэффициент сжатия8/15 – 53%). Для двоичного кодирования указывать можно только 0. Метод удваивает

длину одиночных символов (B→B0), что при большом их количестве может привести не к сжатию, а наоборот к увеличению кода. Поэтому для текстовых данных этот метод неэффективен. Наилучшими объектами для его применения являются изображения, в которых есть большие одноцветные участки, и файлы баз данных с фиксированной длиной полей, где много пробелов. В частности, на нем основан формат графических данных pcx.

Если сообщение содержит много одиночных символов, то можно использовать модифицированный код RLE. В нем одиночные символы не изменяются, а перед последовательностями ставится префикс. Обозначим префикс символом *, тогда AAAAABCDDEFGGGGGG = *A4BC*D1EF*G5 (коэффициент сжатия 13/17 – 76%). Введение префиксов ухудшает сжатие, но может спасти от увеличения кода.

Почти все современные словарные методы, используемые на практике, принадлежат семейству адаптивных алгоритмов, родоначальником которых стал опубликованный в 1977 г. израильскими математиками А.Лемпелом и Я.Зивом алгоритм LZ77. Его называют LZ-сжатие. К нему относятся, например, LZ77, LZSS, LZ78, LZW, LZH, LZHA и др. Их считают обобщением RLE. Достоинствами алгоритмов LZ являются достаточная простота при высокой степени сжатия. Алгоритмы LZ разделяются на две группы с несколько разными принципами работы. Родоначальником первой группы стал алгоритм LZ77. Эти алгоритмы используют уже просмотренную часть сообщения из последних N букв (скользящее окно) как словарь и заменяют повторные вхождения некоторой подстроки символов ссылкой на место ее первого вхождения. Ссылка состоит из смещения, длины соответствия и первого несовпадающего символа.

Наиболее совершенным алгоритмом этой группы является LZSS (Сторер и Шиманский, 1982). Еще лучше похожий на LZ77 алгоритм LZMA (Lempel Ziv Markov chain Algorithm, 2001 г.

Родоначальником второй группы стал алгоритм LZ78 (1978 г.). В нем словарем служит весь просмотренный текст, который разбирается на фразы, где каждая новая фраза есть самая длинная из уже просмотренных (ее называют префиксом) плюс один новый символ. Она кодируется как номер (индекс) ее префикса плюс этот новый символ. После чего новая фраза добавляется к списку фраз (индексируется), на которые можно ссылаться. Если символ встречается в первый раз, в качестве индекса префикса ставится 0. Следуя этому алгоритму, строка «*aaabbabaabaaabab*», разбирается на 7 фраз (табл. 27.1). Например, последние три символа «*bab*» кодируются как 4*b*, фраза номер 4 = *ba* + «*b*». Результатом кодирования будет строка 0*a*1*a*0*b*3*a*4*a*5*a*4*b*.

Таблица 27.1

Выделенная фраза	*a*	*aa*	*b*	*ba*	*baa*	*baaa*	*bab*
Индекс (номер)	1	2	3	4	5	6	7
Код фразы	0*a*	1*a*	0*b*	3*a*	4*a*	5*a*	3*b*

Исследования в области адаптивного словарного кодирования.

Существуют три направления исследований в данной области: повышение эффективности сжатия, убыстрение работы алгоритма и осуществление сжатия на основании новой системы контекстов.Сейчас лучшие схемы достигают сжатия в 2,3– 2,5 битов/символ для английского текста. Показатель иногда может быть

немного улучшен путем использования больших объемов памяти.

Большинство словарей не содержат имен людей, названий мест, институтов, торговых марок и т.д., хотя они составляют основную часть почти всех документов. Поэтому специальные алгоритмы сжатия, взятые в расчете на лингвистическую информацию более высокого уровня, будут несомненно зависеть от системной сферы.

Сжатие придает хранимым и передаваемым сообщениям некоторую степень секретности. Во-первых, оно защищает их от случайного наблюдателя. Во-вторых, посредством удаления избыточности оно не дает возможности криптоаналитику установить присущий естественному языку статистический порядок. В-третьих, что самое важное, модель действует как очень большой ключ, без которого расшифровка невозможна. Применение адаптивной модели означает, что ключ зависит от всего текста, переданного системе кодирования/ раскодирования во время ее инициализации. Также в качестве ключа может быть использован некоторый префикс сжатых данных, определяющий модель для дальнейшего раскодирования.

Наиболее совершенным алгоритмом этой группы является LZW (Т. Вэлч, 1984 г.)

Алгоритмы семейства LZ используются не только для кодирования текстовой информации, но и для кодирования изображений, в частности, на алгоритме LZW основаны такие популярные форматы графических файлов как gif, tif, tiff, png, а так же pdf (Adobe Acrobat).

Большинство современных архиваторов работает по двухшаговому (конвейерному) типу. Сначала предварительное сжатие производится простым и быстрым методом, например, LZ или RLE, а затем окончательно заканчивается эффективным методом Хаффмана. Так программа PKZIP использует алгоритм DEFLATE,

который использует комбинацию LZ77 и алгоритма Хаффмана.

В табл. 27.2 приведены основные компьютерные программы–архиваторы с расширениями создаваемых файлов и с указанием используемых методов сжатия.

Таблица 27.2

Программа	Расширение	Методы
Zip, unzip, pkzip, pkunzip, winzip	Zip	LZW, LZ77, Хаффмана, DEFLATE
Arj, winarj	Arj	LZ77, Хаффмана
rar, winrar	Rar	
Arc, pkarc	arc	LZ77, Хаффмана
Iha, Iharc, Ihice	ice, Izh	LZ77, Хаффмана
Pak	pak	LZW
Gzip	gz	LZ77, Хаффмана
Compress	z	LZW
7-zip	7z	LZMA, DEFLATE
bzip	bz	Арифметическое
Bzip2	Bz2	Хаффмана

28. Методы сжатия с потерей информации

Актуальность алгоритмов сжатия с потерей информации связана с большим размером графических, аудио и видео файлов. Методы без потери информации не дают такого сильного сжатия, а для видео и аудио информации часто просто неэффективны из-за особенностей этой информации. Большинство методов с потерей информации кодируют не исходные данные, а некоторые линейные преобразования от них, например,

коэффициенты дискретного преобразования Фурье, косинусного преобразования, вейвлет преобразования, преобразований Уолша, Хаара и т.д. Обычно эти методы работают в два этапа: на первом данные преобразуются с потерей их части к виду, который на втором этапе может быть эффективно закодирован уже с помощью метода без потери информации.

Для сжатия с потерями графических изображений принят стандартный метод JPEG по названию группы разработчиков Joint Photographic Experts Group). Он основан на следующих принципах. Пусть имеется полутоновое изображение (в градациях серого). Оно обычно кодируется значениями яркости точек экрана, которым соответствует 1 байт информации (0 – минимальная яркость – черный цвет, 255–максимальная – белый). На первом этапе изображение разбивается на квадраты 8x8 пикселей, для каждого из которых производится дискретное косинусное преобразование. В принципе, все сводится к разложению по базисным функциям. После чего отбрасывается задаваемая предварительно часть самых высокочастотных базисных функций, которые отвечают за самые резкие перепады яркости. Изображение несколько сглаживается, но сохраняет достаточно высокое качество. Параметры оставшихся базисных функций образуют числовую последовательность, которую кодируют методом Хаффмана. Цветное изображение при записи разбивается на два канала — яркости и цветности, каждый из которых кодируется по только что указанному алгоритму. Кроме того, цветовой канал обычно переводят в меньшее разрешение (обычно в 2 раза, что соответствует уменьшению информации в 4 раза), поскольку человеческий глаз воспринимает главным образом яркостную информацию и существенно меньше цветовую.

В настоящее время Фурье–преобразование часто стали заменять на вейвлет–преобразование, основанное на представлении сигнала в виде сумм некоторых специальных базовых функций (волновых пакетов). Например, его использует популярный формат djvu и JPEG 2000. Большую эффективность имеет фрактальный алгоритм сжатия, основанный на обнаружении самоподобных участков в изображении и нахождения преобразования, переводящего одни в другие. Математически он основан на применении систем итерируемых функций (IFS), как правило, являющихся аффинными преобразованиями.

Прямое аффинное преобразование можно записать в виде системы линейных уравнений

$$X = Ax + By + C, \quad Y = Dx + Ey + F.$$

Поскольку в выражениях используется две координаты, а коэффициентов три, то чтобы учесть коэффициенты C и F целесообразно перейти к так называемым однородным координатам. В этом случае в вектор исходных и полученных координат добавляется 1, а в матрицу – строку (0,0,1).Тогда записанные выше аффинные преобразования можно переписать в следующем виде

$$\begin{vmatrix} X \\ Y \\ 1 \end{vmatrix} = \begin{vmatrix} A & B & C \\ D & E & F \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}.$$

Метод *IFS*может быть описан, как последовательный итеративный расчет координат новых точек в пространстве:

$$x_{k+1} = F_x(x_k, y_k),$$

$$y_{k+1} = F_y(x_k, y_k),$$

где F_x и F_y – функции преобразования координат, например, аффинного преобразования.

Основная идея такая: поскольку фракталы могут представлять очень сложные изображения с помощью простых итераций, то описание этих итераций требует значительно меньшего объема информации, чем сгенерированное растровое изображение. Для кодирования изображения необходимо решить обратную задачу, т.е. подобрать для изображения или его фрагмента соответствующие коэффициенты аффинного преобразования. Таким образом, создание изображения с помощью IFS рассматривается как обратная задача сжатию.

Однако из-за медленности (много перебора вариантов) и проблем с патентами он не получил широкого распространения.

Самым распространенным стандартом для сжатия видеоданных являются алгоритмы MPEG (MotionPictureExpertsGroup). Они основаны на том, что при переходе от одного кадра к другому информация на экране меняется мало. Таким образом, эффективнее запоминать полностью только некоторые базовые кадры и изменения в них от кадра к кадру. Для запоминания самого изображения используется алгоритм, аналогичный JPEG.

Для сжатия звука также разработаны алгоритмы MPEG, из которых самым распространенным в настоящее время является MPEG layer 3 (MP3).

Основная идея, на которой основаны все методики сжатия с потерями аудио сигнала, это пренебрежение деталями звучания, лежащими вне пределов которые воспринимает человеческое ухо. Какими конкретно, определяет психоакустическая модель. К сожалению здесь нет точных математических формул. Восприятие звука человеком — сложный, до конца не изученный процесс, поэтому выбор методов сжатия выполняется на основе анализирующего прослушивания и сравнения по-разному сжатых звуков группами экспертов. Здесь можно выделить несколько моментов. Во-первых, использование маскирующего эффекта, заключающегося в том, что человеческое ухо теряет часть звуков, лежащих в той же полосе частот, что и более громкий звук (находясь рядом с сиреной, вы не услышите разговор). Эффект зависит от различия в громкости и частоты звука. Кодер разбивает диапазон звуковых частот на подполосы, в каждой из которых убираются неслышимые звуки. Лучшие программы кодирования учитывают также влияние соседних полос.

Еще одним моментом кодирования является использование модели, опирающейся на особенности человеческого восприятия звука. Сжатие с использованием этой модели основано на удалении заведомо неслышимых частот с более тщательным сохранением звуков, хорошо различаемых человеческим ухом. В области высокой громкости можно также уменьшать количество уровней квантования. Большинство существующих алгоритмов для кодировки человеческого голоса основано на высокой предсказуемости такого сигнала.

Еще одним приемом сжатия является использование так называемого совмещенного стерео. Известно, что слуховой аппарат человека может определить направление

лишь средних частот, значит, высокие и низкие частоты можно кодировать в моносигнал.

На последней стадии сжатия используется алгоритм сжатия Хаффмана. Он позволяет улучшить степень сжатия для относительно однородных сигналов, которые плохо сжимаются с помощью описанных выше приемов. На основе описанных идей строятся алгоритмы сжатия, позволяющие уменьшать размер файла в 10 и более раз практически без потери в качестве звучания.

Уровень сжатия обычно указывают в виде величины потока данных битрейта (bit rate), т.е. количества бит передаваемой информации в секунду. Фактически, это количество бит, затрачиваемых на кодирование 1 секунды музыки.

29. Классификация помехоустойчивых кодов

В настоящее время помехоустойчивые (корректирующие) коды используются как для обнаружения ошибок, так и для их исправления[8]. Классификация корректирующих кодов представлена на рис. 29.1.

Как видно из рис. 29.1, все корректирующие коды делятся на два больших класса – блочные и непрерывные. Блочные коды – коды, в которых каждому знаку сообщения в соответствие ставится кодовая комбинация из *n* символов, т.е. блок из n символов. Блочный код называется равномерным, если значение *n* остается постоянным для всех знаков сообщения. В противном случае код называется неравномерным.

Непрерывные коды представляют собой непрерывную последовательность элементов, не подразделяемую на блоки. Процессы кодирования и декодирования здесь имеют непрерывный характер.

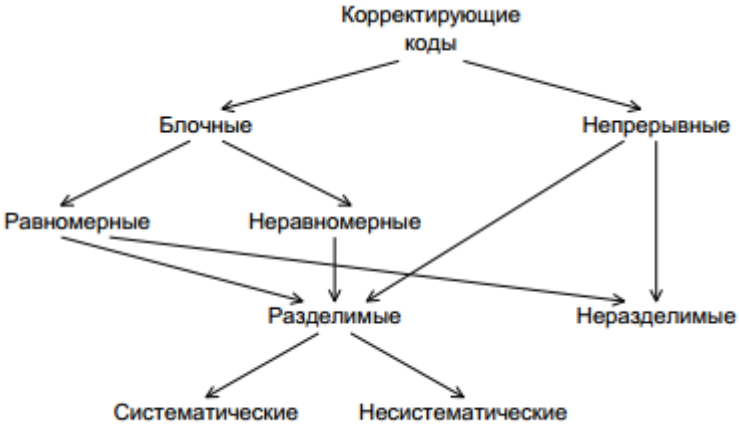


Рис.29.1. Классификация корректирующих кодов

Блочные и непрерывные коды, в свою очередь, могут быть разделимыми и неразделимыми. При кодировании разделимыми кодами в выходных последовательностях символов можно разграничить информационные символы и проверочные. В неразделимых кодах такое разграничение невозможно.

Разделимые коды делятся на систематические и несистематические. Систематическими называются коды, в которых контрольные (проверочные) элементы представляют собой различные линейные комбинации информационных элементов. Несистематические коды таким свойством не обладают. В настоящее время среди

корректирующих кодов наибольшее распространение имеют блочные разделимые систематические коды.

30. Общие принципы использования избыточности при построении корректирующих кодов

*Каналы с ошибкой.*Разберёмся сперва, откуда вообще берутся ошибки, которые мы собираемся исправлять. Перед нами стоит следующая задача. Нужно передать несколько блоков данных, каждый из которых кодируется цепочкой двоичных цифр. Получившаяся последовательность нулей и единиц передаётся через канал связи. Но так сложилось, что реальные каналы связи часто подвержены ошибкам. Вообще говоря, ошибки могут быть разных видов – может появиться лишняя цифра или какая-то пропасть. Но мы будем рассматривать только ситуации, когда в канале возможны лишь замены нуля на единицу и наоборот. Причём опять же для простоты будем считать такие замены равновероятными.
Ошибка – это маловероятное событие (а иначе зачем нам такой канал вообще, где одни ошибки?), а значит, вероятность двух ошибок меньше, а трёх уже совсем мала. Мы можем выбрать для себя некоторую приемлемую величину вероятности, очертив границу «это уж точно невозможно». Это позволит нам сказать, что в канале возможно не более, чем k ошибок. Это будет характеристикой канала связи.

Для простоты введём следующие обозначения. Пусть данные, которые мы хотим передавать, будут двоичные последовательности фиксированной длины. Чтобы не запутаться в нулях и единицах, будем иногда обозначать их заглавными латинскими буквами (*A*, *B*, *C*, …). Что именно передавать, в общем-то неважно, просто с буквами в первое время будет проще работать.

Кодирование и декодирование будем обозначать прямой стрелкой (→), а передачу по каналу связи – волнистой стрелкой (⇝). Ошибки при передаче будем подчёркивать.Например, пусть мы хотим передавать только сообщения *A*=0 и *B*=1. В простейшем случае их можно закодировать нулём и единицей (сюрприз!):*A*→0,*B*→1.Передача по каналу, в котором возникла ошибка будет записана так:*A*→0⇝1_→*B*.

Цепочки нулей и единиц, которыми мы кодируем буквы, будем называть кодовыми словами. В данном простом случае кодовые слова – это 0 и 1.

Код с утроением.Давайте попробуем построить какой-то корректирующий код. Что мы обычно делаем, когда кто-то нас не расслышал? Повторяем дважды:A→00,B→11.

Правда, это нам не очень поможет. В самом деле, рассмотрим канал с одной возможной ошибкой:A→00⇝01_→?.Какие выводы мы можем сделать, когда получили 01? Понятно, что раз у нас не две одинаковые цифры, то была ошибка, но вот в каком разряде? Может, в первом, и была передана буква B. А может, во втором, и была передана *A*.То есть, получившийся код обнаруживает, но не исправляет ошибки. Ну, тоже неплохо, в общем-то. Но мы пойдём дальше и будем теперь утраивать цифры.*A*→000,*B*→111.

Проверим в деле:*A*→000⇝01_0→*A*?.

Получили 010. Тут у нас есть две возможности: либо это *B* и было две ошибки (в крайних цифрах), либо это *A* и была одна ошибка. Вообще, вероятность одной ошибки выше вероятности двух ошибок, так что самым правдоподобным будет предположение о том, что передавалась именно буква *A*. Хотя правдоподобное — не значит истинное, поэтому рядом и стоит вопросительный знак.

Если в канале связи возможна максимум одна ошибка, то первое предположение о двух ошибках становится невозможным и остаётся только один вариант — передавалась буква *А*.Про такой код говорят, что он исправляет одну ошибку. Две он тоже обнаружит, но исправит уже неверно.Это, конечно, самый простой код. Кодировать легко, да и декодировать тоже. Ноликов больше – значит, передавался ноль, единичек — значит единица.Если немного подумать, то можно предложить код исправляющий две ошибки. Это будет код, в котором мы повторяем одиночный бит.

Расстояния между кодами. Рассмотрим поподробнее код с утроением. Итак, мы получили работающий код, который исправляет одиночную ошибку. Но за всё хорошее надо платить: он кодирует один бит тремя. Не очень – то и эффективно.И вообще, почему этот код работает? Почему нужно именно утраивать для устранения одной ошибки? Наверняка это всё неспроста.Давайте подумаем, как этот код работает. Интуитивно всё понятно. Нолики и единички – это две непохожие последовательности. Так как они достаточно длинные, то одиночная ошибка не сильно портит их вид.

Пусть мы передавали 000, а получили 001. Видно, что эта цепочка больше похожа на исходные 000, чем на 111. А так как других кодовых слов у нас нет, то и выбор очевиден.Но что значит «больше похоже»? А всё просто! Чем больше символов у двух цепочек совпадает, тем больше их схожесть. Если почти все символы отличаются, то цепочки «далеки» друг от друга.Можно ввести некоторую величину $d(\alpha,\beta)$, равную количеству различающихся цифр в соответствующих разрядах цепочек α и β. Эту величину называют расстоянием Хэмминга. Чем больше это расстояние, тем меньше похожи две цепочки.

Например, $d(010,010)=0$, так как все цифры в соответствующих позициях равны, а вот $d(010101,011011)=3$.

010101
011011
=001110

Расстояние Хэмминга называют расстоянием между кодаминеспроста. Ведь в самом деле, что такое расстояние? Это какая-то характеристика, указывающая на близость двух точек, и для которой верны утверждения:

1. Расстояние между точками неотрицательно и равно нулю только, если точки совпадают.
2. Расстояние в обе стороны одинаково.
3. Путь через третью точку не короче, чем прямой путь.

Достаточно разумные требования.Математически это можно записать так (нам это не пригодится, просто ради интереса посмотрим):

1. $d(x,y)\geqslant 0, d(x,y)=0\Leftrightarrow x=y$;
2. $d(x,y)=d(y,x)$;
3. $d(x,z)+d(z,y)\geqslant d(x,y)$.

Предлагаю читателю самому убедиться, что для расстояния Хэмминга эти свойства выполняются.

Окрестности. Таким образом, разные цепочки мы считаем точками в каком-то воображаемом пространстве, и теперь мы умеем находить расстояния между ними. Правда, если попытаться сколько – нибудь длинные цепочки расставить на листе бумаги так, чтобы расстояния Хэмминга совпадали с расстояниями на плоскости, мы можем потерпеть неудачу. Но не нужно переживать. Всё

же это особое пространство со своими законами. А слова вроде «расстояния» лишь помогают нам рассуждать.

Пойдём дальше. Раз мы заговорили о расстоянии, то можно ввести такое понятие как окрестность. Как известно, окрестность какой-то точки – это шар определённого радиуса с центром в ней. Шар? Какие ещё шары! Мы же о кодах говорим.

Но всё просто. Ведь что такое шар? Это множество всех точек, которые находятся от данной не дальше, чем некоторое расстояние, называемое радиусом. Точки у нас есть, расстояние у нас есть, теперь есть и шары.

Так, скажем, окрестность кодового слова 000 радиуса 1 – это все коды, находящиеся на расстоянии не больше, чем 1 от него, то есть отличающиеся не больше, чем в одном разряде. То есть это коды:

{000,100,010,001}.

Да, вот так странно выглядят шары в пространстве кодов.А теперь посмотрите. Это же все возможные коды, которые мы получим в канале в одной ошибкой, если отправим 000! Это следует прямо из определения окрестности. Ведь каждая ошибка заставляет цепочку измениться только в одном разряде, а значит удаляет её на расстояние 1 от исходного сообщения.

Аналогично, если в канале возможны две ошибки, то отправив некоторое сообщение *x*, мы получим один из кодов, который принадлежит окрестности *x* радиусом 2.
Тогда всю нашу систему декодирования можно построить так. Мы получаем какую-то цепочку нулей и единиц (точку в нашей новой терминологии) и смотрим, в окрестность какого кодового слова она попадает.

Сколько ошибок может исправить код?Чтобы код мог исправлять больше ошибок, окрестности должны быть

как можно шире. С другой стороны, они не должны пересекаться. Иначе если точка попадёт в область пересечения, непонятно будет, к какой окрестности её отнести.В коде с удвоением между кодовыми словами 00 и 11 расстояние равно 2 (оба разряда различаются). А значит, если мы построим вокруг них шары радиуса 1, то они будут касаться (Рис.30.1). Это значит, точка касания будет принадлежать обоим шарам и непонятно будет, к какому из них её отнести.Именно это мы и получали. Мы видели, что есть ошибка, но не могли её исправить.

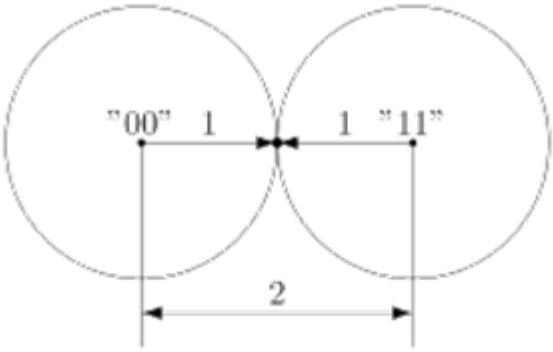


Рис.30.1.Окресности с расстоянием 2

Что интересно, точек касания в нашем странном пространстве у шаров две — это коды 01 и 10. Расстояния от них до центров равны единице. Конечно же, в обычно геометрии такое невозможно, поэтому рисунки — это просто условность для более удобного рассуждения.В случае кода с утроением, между шарами будет зазор (Рис.30.2).

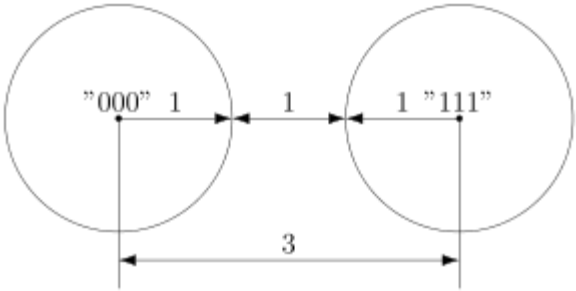


Рис.30.2. Окрестности кода с утроением

Минимальный зазор между шарами равен 1, так как у нас расстояния всегда целые (ну не могут же две цепочки отличаться в полутора разрядах).В общем случае получаем следующее состояние (Рис.30.3).

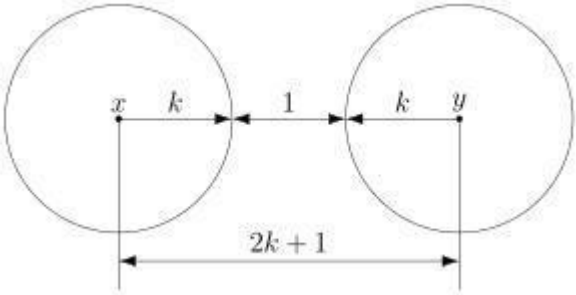


Рис.30.3. Общий случай

Этот очевидный результат на самом деле очень важен. Он означает, что код с минимальным кодовым расстоянием $d_{\min}$ будет успешно работать в канале с k ошибками, если выполняется соотношение:

$$d_{\min} \geqslant 2k+1.$$

Полученное равенство позволяет легко определить, сколько ошибок будет исправлять тот или иной код. А сколько код ошибок может обнаружить? Рассуждения такие же. Код обнаруживает k ошибок, если в результате

не получится другое кодовое слово. То есть, кодовые слова не должны находиться в окрестностях радиуса k других кодовых слов. Математически это записывается так:$d_{\min} \geqslant 2k+1$.

Рассмотрим пример. Пусть мы кодируем 4 буквы следующим образом.

A→10100,
B→01000,
C→00111,
D→11011.

Чтобы найти минимальное расстояние между различными кодовыми словами, построим таблицу попарных расстояний.

	A	B	C	D
A	–	3	3	4
B	3	–	4	3
C	3	4	–	3
D	4	3	3	–

Минимальное расстояние $d_{\min}$ =3, а значит $3 \geqslant 2k+1$, откуда получаем, что такой код может исправить до k=1 ошибок. Обнаруживает же он две ошибки.Рассмотрим пример:
A→10100⇝1011_0.

Чтобы декодировать полученное сообщение, посмотрим, к какому символу оно ближе всего.

A:d(10110,10100)=1,

B:d(10110,01000)=4,

C:d(10110,00111)=2,

$$D{:}d(10110,11011)=3.$$

Минимальное расстояние получилось для символа *A*, значит, вероятнее всего передавался именно он:

$$A\rightarrow 10100 \rightsquigarrow 1011\_0\rightarrow A.$$

Итак, этот код исправляет одну ошибку, как и код с утроением. Но он более эффективен, так как в отличие от кода с утроением здесь кодируется уже 4 символа.

Таким образом, основная проблема при построении такого рода кодов – так расположить кодовые слова, чтобы они были как можно дальше друг от друга, и их было побольше.

Для декодирования можно было бы использовать таблицу, в которой указывались бы все возможные принимаемые сообщения, и кодовые слова, которым они соответствуют. Но такая таблица получилась бы очень большой. Даже для нашего маленького кода, который выдаёт 5 двоичных цифр, получилось бы $2^5=32$ варианта возможных принимаемых сообщений. Для более сложных кодов таблица будет значительно больше.Попробуем придумать способ коррекции сообщения без таблиц. Мы всегда сможем найти полезное применение освободившейся памяти.

31. Понятие порождающей, проверочной матриц и синдрома корректирующего кода

Для изложения дальнейшего материала нам потребуются матрицы. А при умножении матриц, как известно мы складываем и перемножаем числа. И тут есть проблема. Если с умножением всё более-менее хорошо, то как быть со сложением? Из-за того, что мы работаем только с одиночными двоичными цифрами, непонятно, как

сложить 1 и 1, чтобы снова получилась одна двоичная цифра. Значит вместо классического сложения нужно использовать какое-то другое.

Введём операцию сложения как сложение по модулю 2 (хорошо известный программистам XOR):

$$0+0=0,0+1=1,1+0=1,1+1=0.$$

Умножение будем выполнять как обычно. Эти операции на самом деле введены не абы как, а чтобы получилась система, которая в математике называется полем. Поле – это просто множество (в нашем случае из 0 и 1), на котором так определены сложение и умножение, чтобы основные алгебраические законы сохранялись. Например, чтобы основные идеи, касающиеся матриц и систем уравнений по-прежнему были верны. А вычитание и деление мы можем ввести как обратные операции.Множество из двух элементов {0,1} с операциями, введёнными так, как мы это сделали, называется полем Галуа GF(2). GF – это Galoisfield, а 2 – количество элементов.

У сложения есть несколько очень полезных свойств, которыми мы будем пользоваться в дальнейшем.

$$x+x=0.$$

Это свойство прямо следует из определения.

$$x+y=x-y.$$

А в этом можно убедиться, прибавив y к обеим частям равенства. Это свойство, в частности означает, что мы можем переносить в уравнении слагаемые в другую сторону без смены знака.Проверяем корректность.

Вернёмся к коду с утроением: $A\rightarrow 000, B\rightarrow 111$.

Для начала просто решим задачу проверки, были ли вообще ошибки при передаче. Как видно, из самого кода, принятое сообщение будет кодовым словом только тогда, когда все три цифры равны между собой.Пусть мы приняли вектор–строку x из трёх цифр. (Стрелочки над векторами рисовать не будем, так как у нас почти всё — это вектора или матрицы.)

$$\cdots\rightsquigarrow x=(x_1,x_2,x_3).$$

Математически равенство всех трёх цифр можно записать как систему:

$$\begin{cases} x_1 = x_2, \\ x_2 = x_3. \end{cases}$$

Или, если воспользоваться свойствами сложения в GF(2), получаем

$$\begin{cases} x_1 + x_2 = 0, \\ x_2 + x_3 = 0. \end{cases}$$

Или

$$\begin{cases} 1\cdot x_1 + 1\cdot x_2 + 0\cdot x_3 = 0, \\ 0\cdot x_1 + 1\cdot x_2 + 1\cdot x_3 = 0. \end{cases}$$

В матричном виде эта система будет иметь вид: $\boldsymbol{Hx}^{\mathrm{T}}=0$,

где $H = \begin{Vmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{Vmatrix}$.

Транспонирование здесь нужно потому, что $\boldsymbol{x}$ – это вектор. Непрерывные коды представляют собой непрерывную последовательность элементов, не подразделяемую на блоки. Процессы кодирования и декодирования здесь имеют непрерывный характер.

$\boldsymbol{x}$ –строка, а не вектор-столбец. Иначе мы не могли бы умножать его справа на матрицу.Будем называть матрицу $\boldsymbol{H}$ проверочной матрицей. Если полученное сообщение – это корректное кодовое слово (то есть, ошибки при передаче не было), то произведение проверочной матрицы на это сообщение будет равно нулевому вектору.

Умножение на матрицу – это гораздо более эффективно, чем поиск в таблице, но у нас на самом деле есть ещё одна таблица – это таблица кодирования. Попробуем от неё избавиться.

*Кодирование.*Итак, у нас есть система для проверки

$$\begin{cases} x_1 + x_2 = 0, \\ x_2 + x_3 = 0. \end{cases}$$

Её решения – это кодовые слова. Собственно, мы систему и строили на основе кодовых слов. Попробуем теперь решить обратную задачу. По системе (или, что то же самое, по матрице $\boldsymbol{H}$) найдём кодовые слова.

Правда, для нашей системы мы уже знаем ответ, поэтому, чтобы было интересно, возьмём другую матрицу:

$$H = \begin{Vmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{Vmatrix}.$$

Соответствующая система имеет вид:

$$\begin{cases} x_1 + x_3 = 0, \\ x_2 + x_3 + x_5 = 0, \\ x_4 + x_5 = 0. \end{cases}$$

Чтобы найти кодовые слова соответствующего кода нужно её решить.В силу линейности сумма двух решений системы тоже будет решением системы. Это легко доказать. Если a и b – решения системы, то для их суммы верно

$$H(a+b)^T = Ha^T + Hb^T = 0+0=0,$$

что означает, что она тоже – решение.

Поэтому если мы найдём все линейно независимые решения, то с их помощью можно получить вообще все решения системы. Для этого просто нужно найти их всевозможные суммы.

Выразим сперва все зависимые слагаемые. Их столько же, сколько и уравнений. Выражать надо так, чтобы справа были только независимые. Проще всего выразить x_1,x_2,x_4.

Если бы нам не так повезло с системой, то нужно было бы складывая уравнения между собой получить такую систему, чтобы какие-то три переменные

встречались по одному разу. Ну, или воспользоваться методом Гаусса. Для GF(2) он тоже работает.
Итак, получаем:

$$\begin{cases} x_1 = x_3, \\ x_2 = x_3 + x_5, \\ x_4 = x_5. \end{cases}$$

Чтобы получить все линейно независимые решения, приравниваем каждую из зависимых переменных к единице по очереди.

$x_3=1,x_5=0,x_1=1,x_2=1,x_4=0 \Rightarrow x(1)=(1,1,1,0,0),x_3=0,x_5=1:x_1=0,x_2=1,x_4=1 \Rightarrow x(2)=(0,1,0,1,1).$

Всевозможные суммы этих независимых решений (а именно они и будут кодовыми векторами) можно получить так:

$$a_1x(1)+a_2x(2),$$

где a_1,a_2 равны либо нулю или единице.

Так как таких коэффициентов два, то всего возможно $2^2=4$ сочетания.Но посмотрите! Формула, которую мы только что получили – это же снова умножение матрицы на вектор.

$$\begin{Vmatrix} a_1 \\ a_2 \end{Vmatrix} \cdot \begin{Vmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{Vmatrix} = aG.$$

Строчки здесь – линейно независимые решения, которые мы получили. Матрица G называется порождающей. Теперь вместо того, чтобы сами составлять таблицу кодирования, мы можем получать кодовые слова простым умножением на матрицу:

$$a \to aG.$$

Найдём кодовые слова для этого кода. (Не забываем, что длина исходных сообщений должна быть равна 2 – это количество найденных решений.)

$$00 \to 00000, 01 \to 01011, 10 \to 11100, 11 \to 10111.$$

Итак, у нас есть готовый код, обнаруживающий ошибки. Проверим его в деле. Пусть мы хотим отправить 01 и у нас произошла ошибка при передаче. Обнаружит ли её код?
Берем те столбцы матрицы $\boldsymbol{H}$, где $x=1$ и складываем. Получаем синдром. Если он не равен нулю, то в x имеется ошибка. Берем столбцы матрицы $\boldsymbol{H}$, где $x=1$, и складываем. Получаем синдром.Если он не равен 0, то в x имеется ошибка. Например,

$X\underline{01111}$ 010
$\boldsymbol{H}10100$ 110
01101 001
00011 011
110

$$a=01 \to aG=01011 \rightsquigarrow x=011\underline{1}11 \to \boldsymbol{H}x^{T}=(110)^{T}\neq 0.$$

А раз в результате не нулевой вектор, значит, код заподозрил неладное. Провести его не удалось. Ура, код работает!

Для кода с утроением, кстати, порождающая матрица выглядит очень просто:

$$G=|111|.$$

Подобные коды, которые можно порождать и проверять матрицей называются линейными (бывают и нелинейные), и они очень широко применяются на практике.

Реализовать их довольно легко, так как тут требуется только умножение на константную матрицу.

Ошибка по синдрому.

Ну, хорошо, мы построили код обнаруживающий ошибки. Но мы же хотим их исправлять!Для начала введём такое понятие, как вектор ошибки. Это вектор, на который отличается принятое сообщение от кодового слова. Пусть мы получили сообщение x, а было отправлено кодовое слово v. Тогда вектор ошибки по определению

$$e=x-v.$$

Но в странном мире GF(2), где сложение и вычитание одинаковы, будут верны и соотношения:

$$v=x+e, x=v+e.$$

В силу особенностей сложения, как читатель сам может легко убедиться, в векторе ошибки на позициях, где произошла ошибка, будет единица, а на остальных ноль.Как мы уже говорили раньше, если мы получили сообщение x с ошибкой, то $\boldsymbol{Hx}^{\boldsymbol{T}}\neq 0$. Но ведь векторов, не равных нулю много! Быть может то, какой именно ненулевой вектор мы получили, подскажет нам характер ошибки?Назовём результат умножения на проверочную матрицу синдромом:

$$s(x)=\boldsymbol{Hx}^{T}.$$

И заметим следующее

$$s(x)=\boldsymbol{Hx}^{T}=\boldsymbol{H}(v+e)^{T}=\boldsymbol{He}^{T}=s(e).$$

Это означает, что для ошибки синдром будет таким же, как и для полученного сообщения.

Разложим все возможные сообщения, которые мы можем получить из канала связи, по кучкам в зависимости от синдрома. Тогда из последнего соотношения следует, что в

каждой кучке будут вектора с одной и той же ошибкой. Причём вектор этой ошибки тоже будет в кучке. Вот только как его узнать?А очень просто! Помните, мы говорили, что у нескольких ошибок вероятность ниже, чем у одной ошибки? Руководствуясь этим соображением, наиболее правдоподобным будет считать вектором ошибки тот вектор, у которого меньше всего единиц. Будем называть его лидером.

Давайте посмотрим, какие синдромы дают всевозможные 5–элементные векторы. Сведем все комбинации в табл.31.1. Сразу сгруппируем их и подчеркнём лидеров – векторов с наименьшим числом единиц.

Таблица 31.1

$s(x)$	x	X 11100
000	00000, 11100, 01011, 10111	10100 01101 00011
001	00010, 11110, 01001, 10101	
010	01000, 10100, 00011, 11111	
011	01010, 10110, 00001, 11101	
100	10000, 011000, 11011, 00111	s = 100 010 110 000
101	10010, 01110, 11001, 00101	
110	11000, 00100, 10011, 01111	
111	11010, 00110, 10001, 01101	

В принципе, для корректирования ошибки достаточно было бы хранить таблицу соответствия синдрома лидеру.

Обратите внимание, что в некоторых строчках два лидера. Это значит, для данного синдрома два паттерна ошибки равновероятны. Иными словами, код обнаружил две ошибки, но исправить их не может.

Лидеры для всех возможных одиночных ошибок находятся в отдельных строках, а значит, код может исправить любую одиночную ошибку. Ну, что же…. Попробуем в этом убедиться.

$a=01 \rightarrow a\boldsymbol{G}=01011 \rightsquigarrow x=011\_11 \rightarrow s(x)=\boldsymbol{H}\boldsymbol{x}^T=(110)^T \rightarrow e=(00100)$

.

Вектор ошибки равен (00100), а значит ошибка в третьем разряде. Как мы и загадали.Порождающая матрица кода $\boldsymbol{G}$ – это матрица, состоящая из минимального количества линейно независимых кодовых векторов, с помощью которых можно сгенерировать все кодовые векторы. Кодовые слова $\boldsymbol{U}=\boldsymbol{x}\boldsymbol{G}$, где $\boldsymbol{x}$ – информационное сообщение.Для каждой матрицы $k \cdot n$ (где k – количество информационных бит, n – количество бит на выходе кодера) генератора G существует проверочная матрица $\boldsymbol{H}$ размером $(n–k)n$, такая, что строки матрицы $\boldsymbol{G}$ ортогональны к строкам матрицы $\boldsymbol{H}$. Иными словами, $\boldsymbol{G}\boldsymbol{H}^T=0$, где $\boldsymbol{H}^T$– транспонированная матрица $\boldsymbol{H}$, а 0 – нулевая матрица размерностью $k(n\text{-}k)$. $\boldsymbol{H}^T$– это матрица размером $n \cdot (n–k)$, строки которой являются столбцами матрицы $\boldsymbol{H}$, а столбцы – строками матрицы $\boldsymbol{H}$.Синдром – это результат проверки принадлежности полученного приемником сигнала набору кодовых слов.

$$s(x)=\boldsymbol{H}\boldsymbol{x}^T.$$

Класс смежности – это сомножество векторов ошибок, в котором каждый член имеет один и тот же синдром. Синдром каждого класса смежности отличается от синдромов других классов смежности; именно этот синдром используется для определения ошибочных комбинаций.

32. Принципы построения линейных блочных кодов

Код линейный, когда проверочные биты являются линейными комбинациями информационных. Свойство линейного кода: сумма (разность) кодовых векторов дает вектор, принадлежащий этому коду. Свойство группового кода: минимальное кодовое расстояние между кодовыми векторами равно минимальному весу ненулевых векторов (вес равен числу единиц в кодовой комбинации).

Пример группового кода: n=3– длина кода(мощность), k=2 – размерность кода.

$$Базис = \left|\begin{matrix}1 & 1 & 0\\ 0 & 1 & 1\end{matrix}\right| \begin{matrix}X_1\\ X_2\end{matrix}, \left\|\begin{matrix}0 & 0 & 0\\ 1 & 1 & 0\\ 0 & 1 & 1\\ 1 & 0 & 1\end{matrix}\right\| \begin{matrix}X_1\\ X_2\\ X_3\end{matrix}.$$

$$0 = 0 \cdot X_1 + 0 \cdot X_2,$$
$$X_1 = 1 \cdot X_1 + 0 \cdot X_2,$$
$$X_2 = 0 \cdot X_1 + 1 \cdot X_2,$$
$$X3 = 1 \cdot X_1 + 1 \cdot X_2.$$

Когда речь идет о линейных кодах, кодовые комбинации принято называть кодовыми векторами (КВ).Линейный

код обычно обозначают как (*n*,*k*), где *n*– значность КВ, *k*– число информационных символов. Следовательно, число проверочных (контрольных) символов *m*= *n*– *k*. Построение линейного (*n*,*k*) –кода начинается с выбора числа информационных разрядов в кодовых векторах. Это число выбирается, исходя из требуемого объема кода *Q*, т.е. максимального числа сообщений, которое требуется передавать.

В случае передачи двоичным кодом величина *k*должна удовлетворять неравенству:

$$2^k - 1 \geq Q \qquad (32.1)$$

(единица вычитается из 2^k потому, что нулевая комбинация обычно не используется при передаче, т.к. не изменяет состояния канала).После выбора *k* определяется число контрольных разрядов *m*, необходимое для получения требуемой корректирующей способности кода.

Если требуется исправлять все одиночные ошибки (кодовое расстояние $d \geq 3$), величина *m* выбирается из следующих соображений. Под действием помех может быть искажен любой символ в *n*–значном КВ, т.е. для каждого кодового вектора возможно (*n*+1) исходов передачи (+1 учитывает правильную передачу). С помощью контрольных символов нужно различать все возможные исходы передачи. Это возможно, если выполняется условие:

$$2^m \geq C_n^1 + 1 = k + m + 1 \qquad (32.2)$$

где C_n^1 – число сочетаний из n по 1.

Уравнение (32.2) является трансцендентным относительно m, поэтому при небольшихk величину m определяют простым подбором, принимая максимальное значение m, удовлетворяющие (32.2).

При больших k для определения m при $d = 3$ можно использовать эмпирическое соотношение:

$$m=\mathrm{E}\log_2[(k + 1)+ \mathrm{E}\log_2(k +1)], \qquad (32.3)$$

где E – знак округления до ближайшего большего числа.

Если необходимо исправлять не только все единичные, но и все двойные независимые ошибки, величина m должна выбираться в соответствии с условием:

$$2^m \geq C_n^1 + C_n^2 + 1 \qquad (32.4)$$

В общем случае для исправления всех независимых ошибок кратности до S включительно

$$2^m \geq C_n^1 + C_n^2 + ... + C_n^S + 1. \qquad (32.5)$$

После определения m составляется образующая матрица, состоящая из k строк и n столбцов. В общем виде образующая матрица имеет вид:

$$G = \left|\begin{matrix} a_{11} & a_{12} & a_{13} & ..... & a_{1n} \\ a_{21} & a_{21} & a_{21} & ..... & a_{2n} \\ .... & .... & ..... & ..... & .... \\ a_{k1} & a_{k1} & a_{k1} & a_{k1} & a_{kn} \end{matrix}\right| \begin{matrix} v_1 \\ v_2 \\ .. \\ v_k \end{matrix} \qquad (32.6)$$

Кодовые векторы, входящие в образующую матрицу, являются исходными разрешенными. Поскольку двоичный линейный код является групповым по сложению, остальные разрешенные КВ получаются путем суммирования строк по модулю 2 образующей матрицы сначала попарно, затем по три, наконец, всех k –строк.

В качестве строк образующей матрицы могут быть взяты любые КВ, отвечающие следующим условиям. Они должны быть:

1) n–значными;

2) отстоящими друг от друга на заданное кодовое расстояние;

3) ненулевыми;

4) иметь вес не менее заданного кодового расстояния кода;

5) линейно–независимыми.

КВ матрицы $\boldsymbol{G}$ являются разрешенными, что и объясняет первые два требования.

Третье требование связано с тем, что любой групповой код по сложению имеет нулевой элемент, представляющий собой КВ из одних нулей. При сложении такого КВ с любым другим, результат будет совпадать с вторым слагаемым, что недопустимо, т.к. при этом не

выполняется требование замкнутости по операции сложения.

Четвертое требование объясняется тем, что КВ образующей матрицы $\boldsymbol{G}$ являются разрешенными и, следовательно, должны удовлетворять условию $d_{\min} \geq 2S+1.$ Кодовое расстояние между двумя любыми КВ равно весу w вектора, полученного в результате их сложения по модулю 2. Но нулевой КВ, не входящий в $\boldsymbol{G}$, также является разрешенным. Отсюда следует, что вес КВ, входящих в матрицу $\boldsymbol{G}$, должен быть не менее заданного кодового расстояния кода.

Пятое требование гарантирует, что ни один из векторов образующей матрицы (32.6) не будет результатом суммирования по модулю 2 каких-либо других векторов этой же матрицы (в противном случае не будет получено требуемое число разрешенных КВ).

Линейно-независимыми являются КВ, для которых выполняется неравенство:

$$a_1 v_1 \oplus a_2 v_2 \oplus .. a_k v_k \neq 0, \quad (32.7)$$

где v_i, i =1,. ..k – кодовые векторы;

a_i – коэффициенты, принимающие произвольные значения 0 и 1 за исключением

$a_1 = a_2 = \ldots = a_i = \ldots . a_k = 0$.

Последним шагом в построении линейного кода является составление проверочной (контрольной) матрицы, имеющей n столбцов и m строк. В общем виде контрольная матрица имеет вид:

$$G = \begin{vmatrix} \beta_{11} & \beta_{12} & \beta_{13} & ..... & \beta_{1n} \\ \beta_{21} & \beta_{21} & \beta_{21} & ..... & \beta_{2n} \\ .... & .... & ..... & ..... & .... \\ \beta_{k1} & \beta_{k1} & \beta_{k1} & \beta_{k1} & \beta_{kn} \end{vmatrix} \begin{matrix} u_1 \\ u_2 \\ .. \\ u_k \end{matrix} . \qquad (32.8)$$

Элементы β, составляющие контрольную матрицу, представляют собой элементы КВ, ортогональных любым разрешенным кодовым векторам. Если обозначить через *V*–разрешенные КВ данного линейного кода, а через *U*–вектор контрольной матрицы, то условие ортогональности КВ *V* и *U* в математической форме записывается так:

$$V \cdot U = \sum_{i=1}^{n} a_i \beta_i = 0, \qquad (32.9)$$

где a_i и β_i, i =1,…,n –элементы разрешенных КВ и векторов контрольной матрицы соответственно.

Кроме того, что любой вектор контрольной матрицы должен быть ортогонален любому разрешенному КВ, матрица ***H*** в целом должна удовлетворять следующему требованию: в контрольной матрице не должно быть нулевых и одинаковых столбцов.

После построения контрольной матрицы линейный код является полностью определенным. На этапе кодирования КВ формируется так, чтобы он был ортогонален каждому из векторов контрольной матрицы, а на этапе декодирования принятый КВ, возможно содержащий ошибки, проверяется на ортогональность векторам матрицы ***H***.

Пример 32.

Построить линейный (n, k)–код, позволяющий исправлять все одиночные ошибки, если требуемый объем кода Q=12.

Решение.

1. Определяем требуемое число информационных разрядов. Согласно (32.1) имеем$2^k \geq Q+1$, $2^k \geq 13$, откуда k= 4.

2. В соответствии с (31.3) определяем требуемое число контрольных разрядов:

$$m=\text{Elog}_2[(k\ +\ 1)+ \text{Elog}_2(\text{k}+1)]= \text{Elog}_2(5+3)]=3,\ m=3\ ,$$

Следовательно, $n=k+m=7$, а код имеет формат (7, 4).

3. Составляем образующую матрицу $\boldsymbol{G}$.

Так как линейный код должен исправлять однократные ошибки, то кодовое расстояние между комбинациями образующей матрицы должно удовлетворять условию (32.4):

$$d_{\min} \geq 2S+1=3, \qquad (32.10)$$

где S–кратность исправляемой ошибки,S=1.

Теперь, учитывая изложенные выше общие требования к КВ образующей матрицы, переходим к построению самой матрицы. Существует несколько подходов к построению $\boldsymbol{G}$. В самом общем случае при построении $\boldsymbol{G}$ предварительно не определяют места расположения информационных и контрольных символов. Это становится ясным только после построения контрольной

матрицы. Такой подход осуществляется в [10]. В некоторых работах предлагается сразу же назначать места расположения контрольных символов. Так, например, в [7] для кода (7.4) предлагается такое построение кодовых векторов:

$$m_1 m_3\, k_4 m_3 k_3 k_2 k_1.$$

По мнению автора такая структура КВ несколько упрощает алгоритм определения контрольных символов.

В дальнейшем при построении $\boldsymbol{G}$ будем следовать методике, изложенной в [10].

Пусть образующая матрица имеет вид:

$$G = \left|\begin{array}{ccccccc|c} 0 & 1 & 0 & 0 & 1 & 0 & 1 & v_1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & v_2 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & v_3 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & v_k \end{array}\right| .$$

Входящие в нее векторы удовлетворяют всем сформулированным выше условиям.

4. Строим контрольную матрицу $\boldsymbol{H}$.

Ранее отмечалось, что все КВ контрольной матрицы должны быть ортогональны всем разрешенным комбинациям данного линейного кода.

Кодовые векторы образующей матрицы – разрешенные. Контрольная матрица $\boldsymbol{H}$ содержит три КВ и, следовательно, для ее построения в соответствии с (31.9) достаточно иметь три разрешенных КВ. Возьмем в

качестве таковых векторы v_1, v_2 и $v3$ образующей матрицы ***G***.

Определим условия, при которых любой КВ матрицы ***H*** будет ортогонален любому из указанных разрешенных КВ.

В соответствии с (32.9) имеем:

$$\begin{array}{ccccccccc} & \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 & -\,u \\ \times & 0 & 1 & 0 & 0 & 1 & 0 & 1 & -\,v_1 \\ \hline & 0 & \oplus\ \beta_2 & \oplus\ 0 & \oplus\ 0 & \oplus\ \beta_5 & \oplus\ 0 & \oplus\ \beta_7 & = \beta_2 \oplus \beta_5 \oplus \beta_7 = 0 \end{array}$$

$$\begin{array}{ccccccccc} & \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 & -\,u \\ \times & 1 & 0 & 0 & 1 & 1 & 0 & 0 & -\,v_3 \\ \hline & \beta_1 & \oplus\ 0 & \oplus\ 0 & \oplus\ \beta_4 & \oplus\ \beta_5 & \oplus\ 0 & \oplus\ 0 & = \beta_1 \oplus \beta_4 \oplus \beta_5 = 0 \end{array}$$

$$\begin{array}{ccccccccc} & \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 & -\,u \\ \times & 0 & 0 & 1 & 0 & 1 & 1 & 0 & -\,v_2 \\ \hline & 0 & \oplus\ 0 & \oplus\ \beta_3 & \oplus\ 0 & \oplus\ \beta_5 & \oplus\ \beta_6 & \oplus\ 0 & = \beta_3 \oplus \beta_5 \oplus \beta_6 = 0 \end{array}$$

Итак, для того, чтобы любой КВ матрицы ***H*** был ортогонален любому разрешенному КВ, векторы контрольной матрицы должны удовлетворять следующим условиям:

$$\beta_2 \oplus \beta_5 \oplus \beta_7 = 0 \qquad (32.11)$$
$$\beta_3 \oplus \beta_5 \oplus \beta_6 = 0$$
$$\beta_1 \oplus \beta_4 \oplus \beta_5 = 0.$$

Теперь необходимо подобрать β_i, i =1,…,7 , принимающие значения 0 и 1, так, чтобы они удовлетворяли соотношениям (32.11). При этом необходимо следить, чтобы в контрольной матрице не оказалось нулевых и одинаковых столбцов.

Пусть $\beta_2 = \beta_5 = 1$, тогда из (32.1) следует, что $\beta_7 = 0.$

Если β_3 =1, тогда, согласно (32.2), β_6 = 0 .

При β_1 =1, β_4 = 0 , что следует из (32.3).

Итак, первый КВ контрольной матрицы определен: $\boldsymbol{U}_1$ = 1110100.

Определение 2–го КВ контрольной матрицы осуществляется аналогично.

Пусть $\beta_2 = \beta_7 = 1$, тогда $\beta_5 = 0.$

$\beta_3 = \beta_6 = 1$,.

β_1 =1, тогда β_4 = 0.

Принять β_1 =1было нельзя, так как тогда первые два символа в столбцах 1, 2 и 3 разрядов были бы единичными, а это привело бы в дальнейшем к появлению одинаковых

столбцов в контрольной матрице, что недопустимо. Итак, $\boldsymbol{U}_2$= 0110011.

При выборе 3-го КВ контрольной матрицы нужно ввести ряд ограничений, предотвращающих появление одинаковых и нулевых столбцов. В рассматриваемом случае указанные ограничения имеют вид:

$$\beta_2 \neq \beta_3,\ \beta_6 \neq \beta_7,\ \beta_4 = 1.$$

Тогда $\beta_2 = \beta_6 = 1$

$$\beta_3 = \beta_7 = 0,$$

$$U_3(2) - \beta_5 = 1, -\beta_3 = 0$$

$$\boldsymbol{U}_3 = 0101110 \cdot$$

Контрольная матрица в окончательном виде:

$$H = \left|\begin{array}{ccccccc|c} 1 & 1 & 1 & 0 & 1 & 0 & 0 & U_1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & U_2 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & U_3 \end{array}\right|$$

33. Кодирование на основе порождающих полиномов

Рассмотрим вопрос, как используется теория полей многочленов для защиты нформации от помех в канале связи. Считается, что помехи в канале связи действуют независимо и мы не в состоянии повлиять на случайное изменение нуля на единицу, и наоборот; мы можем лишь

на выходе канала связи исправить кодовое слово, восстановив его первоначальную форму. С этой целью и используются специальные *корректирующие коды*, которые основываются на введении информационной *избыточности*. При корректирующем кодировании в каждое кодовое слово, помимо информационных символов, вводят *проверочные*, или *корректирующие*. Например, можно ввести лишь один корректирующий символ в конце информационного слова, для которого определить: 0 — число единиц в кодовом слове четно и 1 — число единиц в кодовом слове нечетно. Если принятое кодовое число имеет четное число единиц, а корректирующий символ равен 1, то в канале связи произошел сбой.

При регистрации сбоя осуществляется повторная передача сообщения. Такой корректирующий код с проверкой на четность единиц в информационном слове используется для контроля передачи информации между отдельными регистрами компьютера. Это самый простой способ проверки. Для контроля считываемой информации из оперативной памяти компьютера используются так называемые *коды Хэмминга*. *Циклические коды* применяются в основном при передаче данных между компьютером и периферийными устройствами, в частности, дисководами. В свою очередь, циклические коды являются подклассом в большом классе линейных кодов, удовлетворяющих дополнительным структурным требованиям. Важность циклических кодов обусловлена еще и тем, что они приводят к очень эффективным процедурам шифровки и дешифровки, легко реализуемым с помощью логических схем. Рассмотрим кратко принцип этого кодирования.

Информационное сообщение $a = a_0 a_1 \ldots a_{k-1}$ будем записывать с помощью многочлена:

$$a(x) = a_0 + a_1x + a_2x^2 + ... + a_{k-1}x^{k-1}.$$

Если многочлен $a(x)$ умножить на x^m, то символы, составляющие сообщение, будут коэффициентами при более высоких степеях. Далее, $a(x)x^m$ разделим на примитивный многочлен $g(x)$, тем самым, найдя частное от деления $p(x)$ и остаток $r(x)$. Зашифрованное информационное слово определится многочленом:

$$f(x) = a(x)x^m - r(x) = g(x)p(x),$$

степень которого равна $n = k + m$. Кодовое слово f будет состоять из k информационных и m проверочных символов. В случае, когда полученное слово f' совпадает с переданным словом f, то при делении многочлена $f'(x)$ на $g(x)$ получается нулевой остаток $r(x)$. Но если остаток отличен от нуля, то в канале связи произошел сбой:

$$f'(x) = g(x)p(x) + r(x), \quad \text{где } r(x) \neq 0.$$

Пусть $a = 1001$, $k = 4$, $m = 3$, $n = 7$, $g(x) = 1 + x + x^3$ – примитивный многочлен. Найдем многочлен $f(x)$, отвечающий коду a:

$$a(x) = 1 + x^3; \quad x^3a(x) = g(x)p(x) + r(x) =$$

$$= (1 + x + x^3)(x + x^3) + (x + x^2);$$

$$f(x) = x + x^2 + x^3 + x^6; \quad f = 011\ 1001.$$

В кодовом слове f первые три символа 011 являются проверочными, последние четыре 1001 – информационными. Пусть при передаче f произошел сбой, и мы приняли слово $f' =$ 011 1011. При делении

многочлена $f'(x)$ на $g(x)$ получаем остаток $r(x) = 1 + x + x^2$, что соответствует коду ошибки $r = 111$. Если сбой произойдет не в шестом символе, а в любом другом месте, то код ошибки изменится.

Существует несколько иной способ кодирования и декодирования информации, а именно, при шифровке многочлен информационного слова $a(x)$ умножается на порождающий многочлен $g(x)$; так получается многочлен $f(x)$. При дешифровке кодовый многочлен $f(x)$ умножается на проверочный $h(x)$, в результате получим:

$$A(x) = f(x)h(x) = a(x)g(x)h(x) = a(x)(x^n + 1) = a(x)s(x)a(x),$$

где $s(x)$ – синдром ошибки, исполняющий роль кода ошибки: $s = r$.

При такой форме кодирования при том же информационном слове и порождающем многочлене, что и в предыдущем случае, сформируется другой кодовый многочлен:

$$f(x) = (1 + x^3)(1 + x + x^3) = 1 + x + x^4 + x^6, \quad f = 1100101.$$

В процессе декодирования получим нулевой синдром:

$$A(x) = f(x)h(x) = (1 + x + x^4 + x^6)(1 + x + x^2 + x^4) = 1 + x^3 + x^7 + x^{10};$$

$$A = asa = 1001\ 000\ 1001; \quad s = 000.$$

Пусть произошел сбой, и было получено слово $f' = 1100001$; тогда в результате декодирования возникнет набор: $A' = 1001\ 111\ 1101$ с синдромом ошибки $s = 111$.

Вместо многочленов можно использовать матрицы. Чтобы получить кодовое слово f, нужно информационное слово a умножить на порождающую матрицу G, т.е. $f = aG$. Так, если $a = 011$ – информационное слово из $k = 3$ символов и задана порождающая матрица G размерности 3x5, то кодовое слово f будет состоять из $n = 5$ символов, из которых два последних ($m = 2$) являются проверочными:

$$f = aG = \begin{vmatrix} 0 & 1 & 1 \end{vmatrix} \cdot \begin{vmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{vmatrix} = 01110.$$

34. Алгебраические системы на базе групп

Откуда же берутся порождающие матрицы ***G***? Порождающая матрица получается путем последовательного сдвига соответствующего порождающего многочлена $g(x)$ по разрядам вправо. Последовательному сдвигу вправо отвечает умножение $g(x)$ на x^i:

$$\boldsymbol{G} = \begin{vmatrix} g(x) \\ xg(x) \\ x^2 g(x) \\ \dots \\ x^{k-1} g(x) \end{vmatrix} = \begin{vmatrix} g_0 & g_1 & g_2 & \dots & g_m & 0 & \dots & 0 \\ 0 & g_0 & g_1 & \dots & g_{m-1} & g_m & \dots & 0 \\ 0 & 0 & g_0 & \dots & g_{m-2} & g_{m-1} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & g_0 & g_1 & \dots & g_m \end{vmatrix}.$$

Порождающая матрица $\boldsymbol{G}$ имеет размерность $(k-1)\cdot m$, поскольку для сдвига берутся степени x^i в пределах $0 < i < k-1$, а степень порождающего многочлена $g(x)$ равна m. Мы знаем, что каждому порождающему многочлену соответствует проверочный многочлен $h(x)$, который удобно записать в порядке убывания степеней:

$$g(x) = g_0 + g_1 x + g_2 x^2 + \dots + g_m x^m,$$

$$h(x) = h_k x^k + h_{k-1} x^{k-1} + \dots + h_0,$$

причем $x^n + 1 = g(x)h(x)$, где, напомним, n –общее число символов, k –число информационных, а m– число проверочных символов в кодовом слове. Если существует порождающая матрица $\boldsymbol{G}$, то должна существовать и соответствующая ей проверочная матрица $\boldsymbol{H}$. Действительно, такую матрицу можно получить путем последовательного сдвига проверочного многочлена $h(x)$ влево:

$$\boldsymbol{H} = \begin{vmatrix} h(x) \\ xh(x) \\ x^2h(x) \\ \dots \\ x^{m-1}h(x) \end{vmatrix} = \begin{vmatrix} 0 & \dots & 0 & h_k & \dots & h_2 & h_1 & h_0 \\ 0 & \dots & h_k & h_{k-1} & \dots & h_1 & h_0 & 0 \\ 0 & \dots & h_{k-1} & h_{k-2} & \dots & h_0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ h_k & \dots & h_0 & 0 & \dots & 0 & 0 & 0 \end{vmatrix}.$$

Предположим, у нас есть конкретный порождающий многочлен $g(x) = 1 + x^2 + x^3$. Проверочный многочлен $h(x)$ находится простым делением многочлена $x^7 + 1$ на заданный многочлен $g(x)$; в результате имеем: $h(x)= x^4 + x^3 + x^2 +1$. В соответствии с вышеприведенными определениями, находим конкретный вид порождающей $\boldsymbol{G}$ и проверочной $\boldsymbol{H}$ матриц:

$$\boldsymbol{G}= \begin{vmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{vmatrix},$$

$$\boldsymbol{H}= \begin{vmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{vmatrix}.$$

$$\boldsymbol{GH}^* = (\boldsymbol{HG}^*)^* = 0.$$

Последнее равенство говорит о том, что матрицы $\boldsymbol{G}$ и $\boldsymbol{H}$ ортогональны относительно друг друга

(звездочка означает операцию транспонирования матрицы).

Рассмотренные $\boldsymbol{G}$ и $\boldsymbol{H}$ матрицы называются *ленточными*, потому что нули и единицы вдоль обеих диагоналей этих матриц образуют своеобразные ленты. Но любая ленточная матрица может быть сведена к *систематическому* виду:

$$\boldsymbol{G}' = (E_{k \cdot k} \mid G_{m \cdot k}), \quad \boldsymbol{H}' = (H_{k \cdot m} \mid E_{m \cdot m}),$$

где $E_{k \cdot k}$ и $E_{m \cdot m}$ – единичные матрицы.

Существует, по крайней мере, два способа сведения ленточных матриц к систематическому виду. Первый наиболее надежный способ состоит в нахождении ряда остаточных многочленов. Если $r_i(x)$ остаточный многочлен от деления x_i на порождающий многочлен $g(x)$, то сумма элементов $r_i(x) + x_i$ дает строки систематической матрицы $\boldsymbol{G}$. Аналогичным способом находятся строки проверочной матрицы $\boldsymbol{H}$. Второй способ заключается в том, чтобы найти соответствующие линейные комбинации строк или столбцов исходных матриц ленточного типа. Найдем $\boldsymbol{G}$ и $\boldsymbol{H}$ для предыдущего случая:

$\boldsymbol{G}$: $x^3=(x^3+x^2+1)\cdot 1+x^2+1$, $1+x^2+x^3$;
$x^4=(x^3+x^2+1)\cdot(x+1)+x^2+x+1$, $1+x+x^2+x^4$;
$x^5=(x^3+x^2+1)\cdot(x^2+x+1)+x+1$, $1+x+x^5$;
$x^6=(x^3+x^2+1)\cdot(x^3+x^2+1)+x^2+x$, $x+x^2+x^6$.
$\boldsymbol{H}$: $x^6=(x^4+x^3+x^2+1)\cdot(x^2+x)+x^3+x^2+x$, $x^6+x^3+x^2+x$;
$x^5=(x^4+x^3+x^2+1)\cdot(x+1)+x^2+x+1$, x^5+x^2+x+1;
$x^4=(x^4+x^3+x^2+1)\cdot 1+x^3+x^2+1$, $x^4+x^3+x^2+1$.

$$G' = \left(\begin{array}{ccc|cccc} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right)$$

$$H' = \left(\begin{array}{ccc|cccc} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{array}\right)$$

Эти две систематические матрицы можно было бы получить путем сложения векторов–столбцов исходных ленточных матриц (напоминаем, счет столбцов для G матрицы начинается с нуля, а для H матрицы — с шести).

G' : 0' = 1 + 2, 1' =1 + 5, 2'= 3, 3' = 0, 4' = 1, 5' = 4 + 1, 6' = 6;

H' : 0' = 5, 1' = 3, 2' = 4, 3' = 2, 4' = 6, 5' = 1, 6' = 0.

Теперь поясним, как составить проверочные соотношения и определить коды ошибок s_i по известной проверочной матрице. С этой целью выпишем три равенства, отвечающих строкам матрицы H' :

$$s_1 = h_6 + h_3 + h_2 + h_1 = 0,$$
$$s_2 = h_5 + h_2 + h_1 + h_0 = 0,$$
$$s_3 = h_4 + h_3 + h_2 + h_0 = 0.$$

При ошибке в символе h_0 суммы s_2 и s_3 изменятся на 1, а при ошибке в h_1 не будут равны нулю суммы s_1 и s_2. Таким образом, можно составить все коды ошибок для всех символов. Результат сведен в таблицу.

h_i	$s_1 s_2 s_3$
h_0	011
h_1	110
h_2	111
h_3	101
h_4	001
h_5	010
h_6	100

При одновременном появлении ошибок в двух символах, например в h_5 и h_6, коды ошибок будут складываться; в данном случае он становится таким же, как и при одиночной ошибке в символе h_1. Поэтому, характеризуя код с точки зрения помехозащищенности, мы должны сказать, что он обнаруживает и исправляет любые одиночные ошибки, а также обнаруживает, но не исправляет двойные ошибки.

35. Разработка структуры кодера на основе полиномиальных коэффициентов

Рассмотримопределение асимптотической эффективности сверточных кодов, построение зависимости выигрыша (в дБ) от отношения сигнал/шум.

Сверточные коды генерируются с помощью регистра сдвига и могут быть описаны с помощью полиномиальных коэффициентов. Сверточные коды называются так потому, что выходные символы кодового устройства можно рассматривать как продукт свертки импульсной характеристики кодового устройства и входной последовательности. На рис.35.1 показана структура кодера со скоростью кодирования 2/3 – на два

входных символа приходится три выходных. В этой структуре выходы регистров сдвига, дающие вклад в выходной сигнал, обозначены «1», а не дающие обозначены «0».

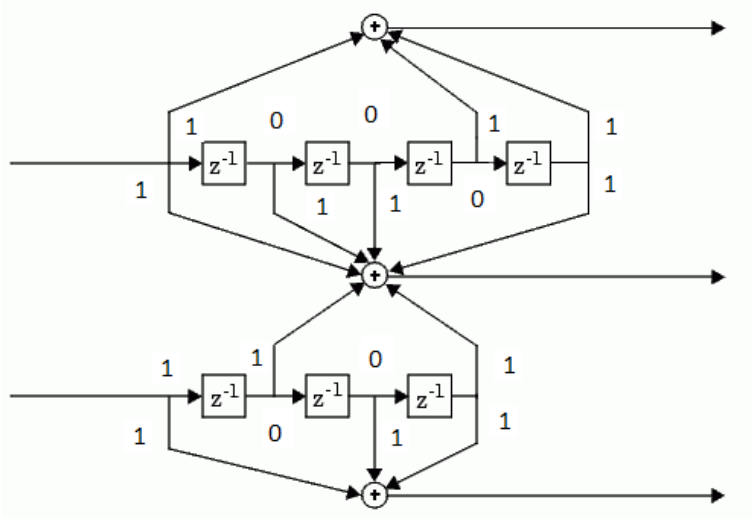


Рисунок 35.1. Структура кодера со скоростью кодирования 2/3

Таким образом, если структура кодера, представленного на рис. 35.1, задается в виде полиномиальных коэффициентов, то ее можно описать, учитывая ненулевые коэффициенты при переменных многочлена:

$$1+X^3+X^4,$$

$$1+X+X^2+X^4,$$

$$1+X+X^3,$$

$$1+X^2+X^3.$$

Запишем получающиеся двоичные последовательности как коэффициенты многочлена в виде табл.35.1.

Таблица 35.1. Задание структуры кодера с помощью порождающих многочленов

Переменные многочлена	1	X	X^2	X^3	X^4
Кодовые Последова-тельности	1	0	0	1	1
	1	1	1	0	1
	1	1	0	1	
	1	0	1	1	

При моделировании структуры сверточного кодера в пакете МАТЛАБ с помощью функции poly2trellis учитываем, что в первой ветви кодера используется пять регистров сдвига, а во второй 4. Параметр ConstraintLength кодовое ограничение, следовательно, будет равен [5 4], параметр CodeGenerator – генератор кода – представляет собой матрицу *k* x *n* (количество входных бит х количество выходных бит) в нашем случае имеет размерность 2 х 3 и содержит в качестве коэффициентов кодовые последовательности, представленные в таблице 35.2, записанные в виде восьмеричных чисел[1]:

[1] Перевод можно сделать с помощью калькулятора Windows, вид – «Программист»

Таблица 35.2

Код	Восьмеричное число
10011	23
11101	35
1101	15
1011	13

Еще два коэффициента будут равны нулю. Таким образом, мы получаем структуру кодера, приведенную на рис.35.2, заданную с помощью функции МАТЛАБ poly2trellis([5 4],[23 35 0;0 15 13]).

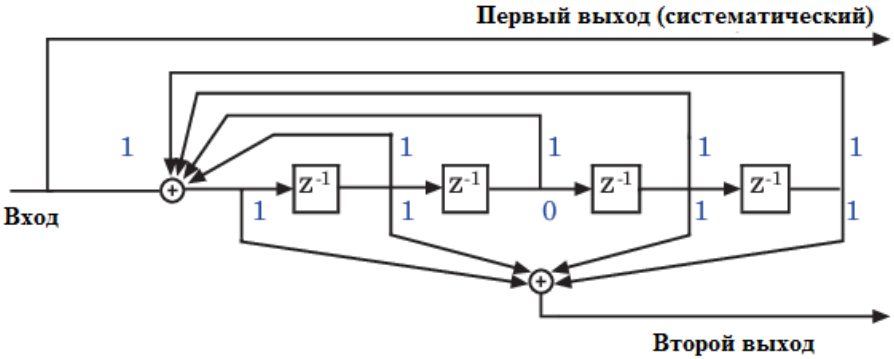


Рис. 35. 2. Систематический кодер с обратной связью

Код является систематическим, если фактические слова сообщения появляются как часть кодовых слов. Рис.35.2. показывает пример систематического кодирующего устройства.

Первый полином генератора совпадает с полиномом связи обратной связи, потому что первый вывод

соответствует систематическим битам. Полином обратной связи представлен бинарным вектором [1 1 1 1 1], соответствуя верхней строке двоичных цифр в схеме. Эти цифры указывают на связи от выходных параметров регистров к сумматору. Начальный 1 соответствует входному биту. Восьмеричное представление двоичного числа 11111 равняется 37.

Второй полином генератора представлен бинарным вектором [1 1 0 1 1], соответствуя более низкой строке двоичных цифр в схеме. Восьмеричное число, соответствующее двоичному числу 11011, равняется 33.

36.Циклические коды

Циклические коды –подкласс класса линейных кодов, которые удовлетворяют следующим циклическим свойствам:

если $C=[c_{n-1},c_{n-2},\ldots,c_1,c_0]$ –
кодовое слово циклического кода, тогда $[c_{n-2},c_{n-3},\ldots,c_2,c_1]$, полученное циклическим сдвигом элементов кода, также являются кодовым словом.

Все циклические сдвиги образуют кодовые слова. Как следствие циклического свойства, эти коды обладают значительным количеством структурных удобств, которые можно использовать при реализации операций кодирования и декодирования. Большое количество алгоритмов эффективных кодеров и декодеров жёстких решений были сделаны посредством циклических кодов, что сделало возможным в практических системах связи строить блоковые коды большой длины с большим количеством кодовых слов. Описание специфических алгоритмов находится вне кругозора этой книги. Наша

основная цель – вкратце описать некоторые характеристики циклических кодов. При работе с циклическими кодами принято с кодовым словом $[c_{n-1},c_{n-2},\ldots,c_1,c_0]$ связывать полином $C\{p\}$ степени $\leq n-1$, определённый так

$$C\{p\} = c_{n-1}p^{n-1} + c_{n-2}p^{n-2} + \ldots c_1 p + c_0 \qquad (36.1)$$

Для двоичного кода каждый из коэффициентов полинома является или нулем, или единицей.

Теперь предположим, мы формируем полином

$$pC\{p\} = c_{n-1}p^{n} + c_{n-2}p^{n-1} + \ldots c_1 p^2 + c_0 p.$$

Этот полином не может представить кодовое слово, так как его степень может быть равна n (если $C_{n-1}=1$). Однако если мы разделим $pC\{p\}$ на p^{n+1}, мы получим

$$\frac{pC(p)}{p^{n+1}} = C_{n-1} + \frac{C_1(p)}{p^{n+1}}, \qquad (36.2)$$

Где $C_1(p) = c_{n-2}p^{n-1} + c_{n-3}p^{n-2} + \ldots c_0 p + c_{n-1}.$

Заметим, что полином $C_1(p)$ представляет кодовое слово, которое как раз образовано из кодового слова с циклическим сдвигом на одну позицию. Поскольку $C_1(p)$ представляет собой остаток, полученный делением $pC(p)$ на p^{n+1}, мы говорим, что

$$C_1(p) = pC(p) \bmod p^{n+1}. \quad (36.3)$$

Аналогичным образом, если $C(p)$ представляет кодовое слово в циклическом коде, тогда $p^i C(p) \bmod p^{n+1}$ также является кодовым словом циклического кода. Так что можно написать

$$p^i C(p) = Q(p)(p^n + 1) + C_i(p), \quad (36.4)$$

Где $C_i(p)$ - полином и представляет слово циклического кода, а $Q(p)$ – частное.

Мы можем генерировать циклический (n,k) код, используя **порождающий полином** $g(p)$ степени (n–k) с двоичными коэффициентами, который является множителем при факторизации полинома p^{n+1}. Порождающий полином в общем виде можно записать так

$$g(p) = p^{n-k} + g_1 p + 1 \quad (36.5)$$

Мы также определяем **полином информационного сообщения** $X(p)$

$$X(p) = x_{n-1}p^{k-1} + x_{n-2}p^{k-2} + \ldots + x_1 p + x_0, \quad (36.6)$$

где $[x_{n-1}, x_{n-2}, \ldots, x_1, x_0]$ определяет k информационных бит. Ясно, что произведение $X(p)g(p)$ – это полином степени меньшей или равной n–1, который можетпредставлять кодовое слово.

Заметим, что имеется 2^k полиномов $\{X_i(p)\}$ и, следовательно, имеется 2^k возможных кодовых слов,

которые можно формировать при заданном $g(p)$.Допустим, что мы обозначим эти кодовые слова так

$$C_m(p) = X_m(p)g(p), \quad m=1,2,\ldots, 2^k \qquad (36.7)$$

Чтобы показать, что кодовые слова в (36.7) удовлетворяют циклическому сдвигу, рассмотрим какое – либо кодовое слово $C(p)$ в (36.7), Циклический сдвиг $C(p)$ дает

$$C_1(p) = pC(p)+c_{n-1}(p^n+1) \qquad (36.8)$$

и, поскольку и p^n+1 и $C(p)$ делятся на $g(p)$ без остатка, то и $C_1(p)$ делится на $g(p)$ без остатка, т.е. $C_1(p)$ можно представить как

$$C_1(p) = X_1(p)g(p) .$$

Следовательно, циклический сдвиг в кодовом слове $C(p)$, генерируемый согласно (36.7), порождает другое кодовое слово.

Из вышесказанного мы видим, что кодовые слова, обладающие циклическими свойствами, можно генерировать умножением 2^k сообщений на уникальный полином степени $n–k$ $g(p)$ – называемый порождающим полиномом циклического (n,k) кода, который является множителем при факторизации p^n+1. Циклический код, генерируемый указанным образом, занимает подпространство S_0 векторного пространства S. Размерность подпространства S_0 равна k.

Пример 36.1. Рассмотрим код с длиной блока n=7. Полином p^7+1можно разложить на следующие множители

$$p^7+1=(p+1)(p^3+p^2+1)(p^3+p+1). \quad (36.9)$$

Чтобы синтезировать циклический код (7,4), мы можем взять один из двух порождающих полиномов:

$$g_1(p) = p^3+p^2+1 \quad (36.10)$$

или

$$g_2(p) = p^3+p+1.$$

Коды, генерируемые порождающими полиномами $g_1(p)$ и $g_2(p)$, эквивалентны. Кодовые слова кода (7,4), генерируемые полиномом $g_1(p) =p^3+p^2+1$ даны в табл. 36.1. В общем, полином p^n+1 можно факторизовать так:

$$p^n+1=g(p)h\,(p).$$

где $g(p)$ означает порождающий полином для циклического (n,k) кода, $h(p)$ означает проверочный полином степени k. Последний можно использовать для генерирования дуального кода.

С этой целью можно использовать полином, обратный $h\,(p)$, определяемый так:

$$p^k h(p^{-1})=p^k(p^{-k}+h_{k-1}p^{-k+1}+\ldots+h_1p^{-1}+1)=1+h_{k-1}p+h_{k-2}p^2+\ldots + h_1p^{k-1}+p^k \quad (36.11)$$

Ясно, что p^n+1 делится без остатка и на обратный полином.

Следовательно, $p^k h(p^{-1})$ является порождающим полиномом циклического $(n,n–k)$ кода.

Этот циклический код дуален коду (n,k), генерируемому порождающим полиномом $g(p)$. Таким образом, $(n,n–k)$ дуальный код образует нуль–пространство циклического кода.

Таблица 36.1. Циклический код (7,4). Порождающий полином $g_1(p) = p^3+p^2+1$

Информационные биты				Кодовые слова						
p^3	p^2	p^1	p^0	p^6	p^5	p^4	p^3	p^2	p^1	p^0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	1	0	1
0	0	1	0	0	0	1	1	0	1	0
0	0	1	1	0	0	1	0	1	1	1
0	1	0	0	0	1	1	0	1	0	0
0	1	0	1	0	1	1	1	0	0	1
0	1	1	0	0	1	0	1	1	1	0
0	1	1	1	0	1	0	0	0	1	1
1	0	0	0	1	1	0	1	0	0	0
1	0	0	1	1	1	0	0	1	0	1
1	0	1	0	1	1	1	0	0	1	0
1	0	1	1	1	1	1	1	1	1	1
1	1	0	0	1	0	1	1	1	0	0
1	1	0	1	1	0	1	0	0	0	1
1	1	1	0	1	0	0	0	1	1	0
1	1	1	1	1	0	0	1	0	1	1

Пример 36.2. Рассмотрим циклический код, дуальный коду (7,4), генерированному в примере 36.1. Этот дуальный циклический код (7,3) связан с проверочным полиномом

$h_1=(p+1)(p^3+p^2+1)=p^4+p^3+p^{2+}1).$ (36.12)

Обратный полином равен

$$p^4h_1(p^{-1}) = 1+p+p^2+p^4.$$

Этот полином генерирует дуальный код (7, 3), данный в таблице 36.2.

Таблица 36.2. Дуальный код (7, 3).
Порождающий полином $p^4h_1(p^{-1}) = 1+p+p^2+p^4$

Информационные биты			Кодовые слова						
p^2	p^1	p^0	p^6	p^5	p^4	p^3	p^2	p^1	p^0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	1	1	1	0
0	1	1	0	0	1	1	0	0	1
1	0	0	0	1	0	1	1	0	0
1	0	1	0	1	0	1	0	1	1
1	1	0	0	1	1	0	0	1	0
1	1	1	0	1	1	0	1	0	1

Можно убедиться в том, что кодовые слова дуального кода (7,3) ортогональны кодовым словам циклического кода (7,4) из примера 35.1. Заметим, что ни код (7,4), ни код (7,3) не являются систематическими.

Желательно показать, как можно получить порождающую матрицу из порождающего полинома циклического кода (n,k).Как указано выше, порождающую матрицу для (n,k) кода можно сконструировать из любого набора линейно независимых кодовых слов. По заданному порождающему полиному $g(p)$ легко генерировать набор линейно

независимых кодовых слов – это кодовые слова, соответствующие набору k линейно независимых полиномов $p^{k-1}g(p),p^{k-2}g(p),\ldots,pg(p),g(p)$.

Поскольку любой из полиномов степени меньшей или равной $(n-1)$, который делится на $g(p)$, можно выразить как линейную комбинацию этих полиномов, набор образует базис размерностью k. Следовательно, кодовые слова, связанные с этими полиномами, формируют базис размерности k для циклического кода (n,k).

Пример 36.3. Четыре строки порождающей матрицы для циклического кода (7,4) с порождающим полиномом $g_1(p) = p^3+p^2+1$ можно получить из полиномов

$$p^i g_1(p) = p^{3+i}+p^{2+i}+p^i \ ,i=3,2,1,0.$$

Легко видеть, что порождающая матрица равна

$$\boldsymbol{G}_1=\begin{vmatrix}1&1&0&1&0&0&0\\0&1&1&0&1&0&0\\0&0&1&1&0&1&0\\0&0&0&1&1&0&1\end{vmatrix}. \qquad (36.13)$$

Аналогично,
порождающая матрица для циклического кода (7,4),
образуемая порождающим полиномом $g_2(p) = p^3+p+1$,
равна

$$\boldsymbol{G}_2=\begin{vmatrix}1&0&1&1&0&0&0\\0&1&0&1&1&0&0\\0&0&1&0&1&1&0\\0&0&0&1&0&1&1\end{vmatrix}. \qquad (36.14)$$

Проверочные матрицы, соответствующие $\boldsymbol{G}_1$ и $\boldsymbol{G}_2$, можно сконструировать аналогичным образом, используя соответствующие обратные полиномы (задача 8.8).Заметим, что порождающие матрицы, полученные таким конструированием, не имеют систематическуюформу.

Далее можем сконструировать порождающую матрицу циклического ко да всистематической форме $\mathbf{G} = [\mathbf{I}_k, \mathbf{P}]$от порождающего полинома следующим образом. Для начала заметим, что i–я строка $\boldsymbol{G}$ соответствует полиному формы $p^{n-1}+R_i(p), i=1,2,\ldots,k,$ где $R_i(p)$–полином степени, меньшей, чем $n–k$.

Эту форму можно получить делением p^{n-1} на $g(p)$. Таким образом, имеем

$$\frac{p^{n-1}}{\mathrm{g(p)}} = Q_1(p) + \frac{R_i(p)}{g(p)}, i= 1, 2,...,k,$$

или, что эквивалентно

$$p^{n-1} = Q_1(p)g(p)+R_i(p), \ \ i=1,2,\ldots,k, \qquad (36.15)$$

где $Q_1(p)$ – частное. Но $p^{n-1}+R_i(p)$ представляет кодовое слово в циклическом коде, поскольку $p^{n-1}+R_i(p)=Q_1(p)g(p)$.

Следовательно, желательный полином, соответствующий i-й строке $\boldsymbol{G}$, равен $p^{n-1}+R_i(p)$.

Пример 36.4. Для циклического кода (7,4) с порождающим полиномом $g_2(p) = p^3+p+1$, ранее рассмотренного в примере 35.3, имеем

$$p^6 = (p^3+p+1)g_2(p) + p^2+1,$$
$$p^5 = (p^2+1)g_2(p) + p^2+p+1,$$
$$p^4 = pg_2(p) + p^2+p,$$
$$p^3 = g_2(p) + p+1.$$

Следовательно, порождающая матрица кода в систематической форме

$$\mathbf{G}_2 = \begin{vmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{vmatrix} \qquad (36.16)$$

и соответствующая проверочная матрица

$$\mathbf{H} = \begin{vmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{vmatrix}. \qquad (36.17)$$

Оставляем в качестве упражнения для читателя показать, что порождающая матрица $\boldsymbol{G}_1$, даваемая (36.13) и матрица $\boldsymbol{G}_2$ в систематической форме (36.14) генерируют одинаковые наборы кодовых слов.

Методконструирования порождающей матрицы ***G*** в систематической форме согласно (36.15) также подразумевает, что систематический код можно генерировать непосредственно из порождающего полинома $g(p)$. Предположим, что мы умножим полином сообщения $X(p)$ на p^{n-k}. Тогда получим

$$p^{n-k}X(p)=x_{k-1}p^{n-1}+x_{k-2}p^{n-2}+\ldots+x_1p^{n-k+1}+x_0p^{n-k}.$$

В систематическом коде этот полином представляет первые k бит слова $C(p)$. К этому полиному мы должны прибавить полином степени меньшей, чем $n-k$, представляющей проверочные символы. Теперь, если $p^{n-k}X(p)$ разделить на $g(p)$, результат равен

$$\frac{p^{n-k}X(p)}{g(p)}=Q(p)+\frac{r(p)}{g(p)}$$

или

$$p^{n-k}X(p)=Q(p)g(p)+r(p), \qquad (36.18)$$

где $r(p)$ имеет степень меньшую, чем $n-k$.

Ясно, что $Q(p)g(p)$ – это кодовое слово циклического кода. Следовательно, суммируя (по mod2) $r(p)$ с обеими частями (36.18), мы получаем желаемый систематический код.Подытоживая, скажем, что систематический код можно генерировать так:

– умножаем полином сообщения $X(p)$ на p^{n-k};

– делим $p^{n-k}X(p)$ на$g(p)$, чтобы получить остаток $r(p)$; и

– добавляем $r(p)$ к $p^{n-k}X(p)$.

Ниже мы продемонстрируем, как эти вычисления можно выполнить, используя сдвиговые регистры с обратной связью.

Поскольку $p^{n+1}=g(p)h(p)$или, что эквивалентно, $g(p)h(p)=0 \mod(p^{n+1})$, мы видим, что полиномы $g(p)$ и $h(p)$ ортогональны. Далее, полиномы $p^i g(p)$ и $p^i h(p)$ также ортогональны для всех i и j. Однако, векторы, соответствующие полиномам $g(p)$ и $h(p)$, ортогональны только, если порядок следования элементов одного из этих векторов реверсировать. То же утверждение применимо к векторам, соответствующим полиномам $p^i g(p)$ и $p^i h(p)$. Действительно, если проверочный полином $h(p)$ используется как порождающий для $(n,n-k)$ дуального кода, ансамбль кодовых слов, полученный так, включают в себя те же кодовые слова, которые генерируются обратным полиномом, за исключением того, что кодовые векторы реверсированы. Это подразумевает, что порождающая матрица для дуального кода, полученная от обратного полинома $p^k h(p^{-1})$, может также получить непосредственно от $h(p)$. Поскольку проверочная матрица $\boldsymbol{H}$ для циклического кода (n,k) является порождающей матрицей для дуального кода, следует, что $\boldsymbol{H}$ также можно получить от $h(p)$. Следующий пример иллюстрируют это соотношение.

Пример 36.5. Дуальный код для циклического кода (7,4), порождаемого полиномом $g(p) =p^3+p^2+1$ – это код (7,3), который порождается обратным полиномом $p^4 h_1(p^{-1}) = p^4+p^2+p+1$. Однако, мы можем также использовать $h_1(p)$ для получения порождающей матрицы для дуального кода. Матрица, соответствующая полиному $p^i h_1(p)$, i =2,1,0, равна

$$\mathbf{G}_{h1} = \begin{vmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{vmatrix}.$$

Порождающая матрица для дуального кода (7, 3), которая является проверочной матрицей для циклического кода (7, 4), состоит из строк $\boldsymbol{G}_M$, взятых в инверсном порядке

$$\mathbf{H}_1 = \begin{vmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{vmatrix}.$$

Таким образом, можно убедиться, что $\mathbf{G_1H_1}^{\mathbf{T}}=0$.

Заметив, что вектор H_1, состоит из всех семи двоичных вектор–столбцов длины 3, исключая вектор из одних нулей. Но это как раз описание проверочной матрицы для кода Хемминга (7,4).

Следовательно, циклический код (7,4) эквивалентен коду Хемминга (7,4), рассмотренному ранее в примерах.

Кодеры для циклических кодов. Операции кодирования при создании циклических кодов можно выполнить при помощи линейных сдвигающих регистров с обратной связью, с использованием порождающего или проверочного полинома. Сначала рассмотрим использование $g(p)$.

Как указано выше, генерирование систематического циклического кода включает три ступени, а именно: умножение полинома сообщения $X(p)$ на p^{n-k}; деление этого произведения на $g(p)$ и в заключение, прибавление остатка $r(p)$ к p^{n-}

$^{k}X(p)$. Из этих трёх ступеней только деление является нетривиальным.

Деление полинома $A(p)= p^{n-k}X(p)$ степени n–1 на полином

$$g(p)=g_{n-k}p^{n-k} + g_{n-k-1}p^{n-k-1}+\ldots+g_1p+ g_0$$

можно выполнить посредствам (n–k) ячеек регистра сдвига с обратной связью, показанного на рис. 36.1.

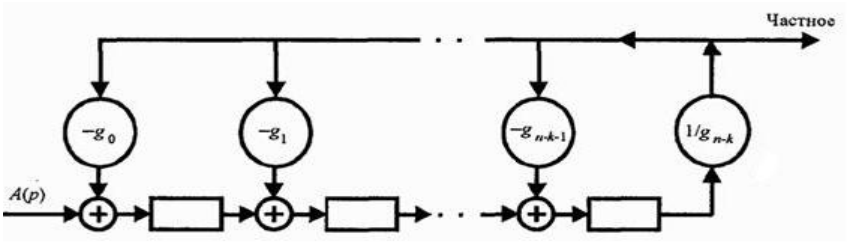


Рис.36.1. Регистр сдвига с обратной связью для деления полинома $A(p)$ на $g(p)$

Первоначально ячейки сдвига регистра содержит одни нули. Коэффициенты $A(p)$ поступают и продвигаются по регистру сдвига по одному коэффициенту за такт, начиная с коэффициентов более высокого порядка, т.е. с a_{n-1}, затем a_{n-2} и так далее. После (n–k–1)–го сдвига, первый ненулевой выход частного равен $q_1=(g_{n-k})^{-1}a_{n-1}$. Последующие выходы генерируются так, как показано на рис. 36.2.

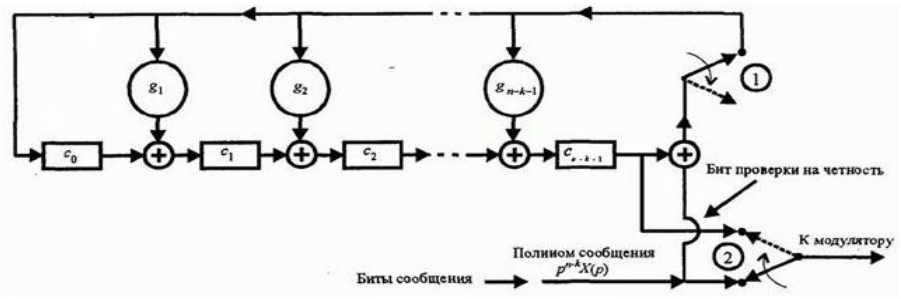


Рис. 36.2. Циклический кодер с использованием порождающего полинома $g(p)$

Для образования каждого выходного коэффициента линии мы должны вычесть полином $g(p)$, умноженный на этот коэффициент, как при обычном «длинном» делении. Это вычитание производится посредством обратной связи. Таким образом, регистр сдвига на рис. 36.1обеспечивает деление двух полиномов.В нашем случае $g_{n-k}=g_0=1$, и для двоичных кодов арифметические операции выполняются по mod2. Следовательно, операция вычитания сводится к сложению по mod2. Далее мы будем только интересоваться генерированием проверочных символов для каждого кодового слова, поскольку код систематический. Как следствие, кодер циклического кода принимает вид, показанный на рис. 36.2.

Первые k бит на выходе кодера просто равны k информационным битам. Эти k бит одновременно поступают на регистр сдвига, поскольку ключ 1 замкнут. Заметим, что умножение полиномов p^{n-k}и $X(p)$ явно не производится. После того как все k информационных бита попали на вход кодера (и к модулятору), положения двух ключей рис. 36.2 меняются на обратные. Начиная с этого времени, содержимое регистра сдвига просто даёт $n-k$ проверочных символов, которые соответствуют

коэффициентам полинома остатка. Эти n-k бита последовательно отправляются на модулятор.

Пример 36.6. Регистр сдвига для кодирования циклическим кодом (7,4) с порождающим полиномом $g_1(p) = p^3+p+1$ показан на рис. 36.3.

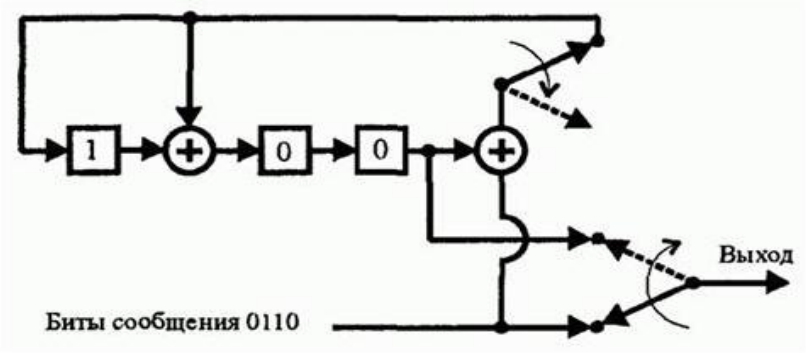


Рис. 36.3. Циклический кодер (7,4) с использованием порождающего полинома $g(p)$

Предположим, что сообщением является цепочка 0110. Содержание сдвигового регистра дано таблицей:

Вход	Шаг сдвига	Содержимое регистра
	0	0 0 0
0	1	0 0 0
1	2	1 1 0
1	3	1 0 1
0	4	1 0 0

Таким, образом, три проверочных символа: 100. Они соответствуют кодовым символам: c_5=0, c_6=0, c_7=1. Вместо использования порождающего полинома $g(p)$, мы можем

выполнить кодер циклического кода, используя проверочный полином

$$h(p)=p^k + h_{k-1}p^{k-1}+\ldots+h_1p+ 1.$$

Соответствующий кодер показан на рис. 36.4.

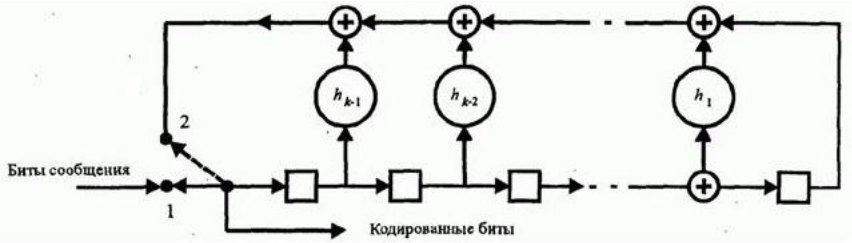


Рис. 36.4. Циклический кодер $n-k$ с использованием проверочного полинома $h(p)$

Первоначально k информационных символа (бита) передвигаются по регистру сдвига и одновременно поступает на модулятор. После того, как все k информационных символов пройдут по регистру, ключ переводится в положение 2, и сдвиговой регистр работает ещё n-k такта, генерируя n-k проверочных символов, как показано на рис. 36.4.

Пример 36.7. Проверочный полином для циклического кода (7,4), генерируемый порождающим полиномом $g(p) = p^3+p+1$, равен $h(p)=p^4+p^2+p+1$. Кодер для этого кода, основанный на проверочном полиноме, иллюстрируется на рис. 36.5. Если на входе кодера действует сообщение 0110, то проверочными символами являются $c_5=0$, $c_6=0$, $c_7=1$, как легко проверить.

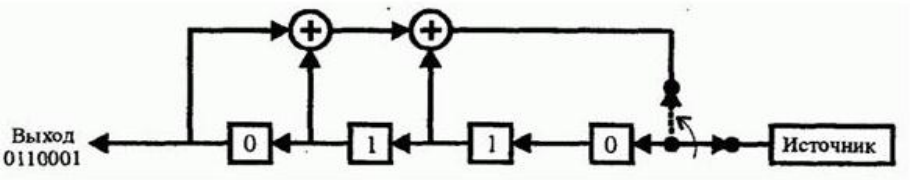


Рис. 36.5. Циклический кодер (7,4) с использованием проверочного полинома $h(p)=p^4 + p^2+p+ 1$

Следует заметить, что кодер, базирующийся на порождающем полиноме, проще, когда $n-k <k$, $(k > \frac{1}{2}n)$, т.е. для больших скоростей кода $R_c > \frac{1}{2}$. В то же время кодер, базирующийся на проверочном полиноме, проще, когда $k<n-k (k > \frac{1}{2}n)$, что соответствует низкоскоростным кодам $R_c > \frac{1}{2}$.

Циклические коды Хемминга. Класс циклических кодов включает коды Хемминга, которые имеют длину блока $n = 2^m - 1$ и $n - k=m$ проверочных символов, где m – любое положительное целое число. Циклические коды Хемминга эквивалентны кодам Хемминга, описанным в разделе 30.

Циклический код Голея (23, 12). Линейный код Голея (23,12) можно генерировать как циклический код посредством порождающего полинома

$$g(p)=p^{11}+p^9+p^7+p^6+p^5+p \quad (36.19)$$

Кодовые слова имеют минимальное расстояние $d_{\min}=7$.

Коды сдвигового регистра максимальной длины. Коды сдвигового регистра максимальной длины – это класс циклических кодов

$$(n\text{-}k)=(2^m-1,m), \quad (36.20)$$

где m – положительное целое. Кодовые слова обычно генерируются посредством m–каскадного цифрового регистра сдвига с обратной связью, основанного на проверочном полиноме. Для каждого передаваемого кодового слова mинформационных бит вводятся в регистр сдвига, а ключ перемещается с позиции 1 на позицию 2. Содержание регистра сдвигается влево в общей сложности 2^m-1 раз. Такая операция генерирует систематический код с требуемой длиной $n=2^m-1$. Для примера, кодовые слова, генерируемые 3–каскадным регистром сдвига по рис. 36.6, сведены в таблицу 36.3

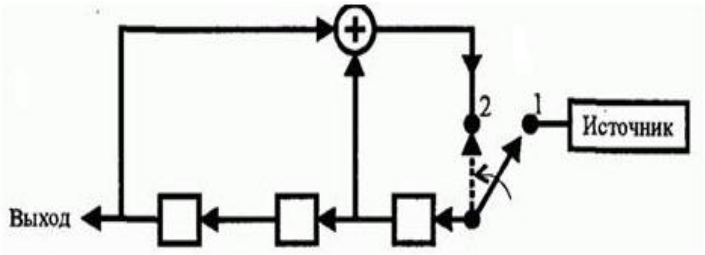


Рис. 36.6. Трёхкаскадный регистр сдвига m=3 с обратной связью

Таблица 36.3. Код сдвигового регистра максимальной длины для m=3

Информационные биты			Кодовые слова						
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	1	0	1
0	1	0	0	1	0	0	1	1	1
0	1	1	0	1	1	1	0	1	0
1	0	0	1	0	0	1	1	1	0
1	0	1	1	0	1	0	0	1	1
1	1	0	1	1	0	1	0	0	1
1	1	1	1	1	1	0	1	0	0

Заметим, что, за исключением кодовых слов из одних нулей, все кодовые слова, генерируемые регистром сдвига, представляют собой циклические сдвиги единственного кодового слова. Это легко увидеть из схемы состояний регистра сдвига, которая иллюстрируется на рис. 36.7 для m=3. Когда регистр сдвига после первоначального заполнения сдвигается $2^m - 1$ раз, он проходит через все возможные $2^m - 1$ состояния.

Таким образом, регистр сдвига возвращается обратно к начальному состоянию за $2^m - 1$ шагов сдвига. Следовательно, выходная последовательность периодическая, и её период равен $n = 2^m - 1$. Поскольку имеется $2^m - 1$ возможных состояний, такая длина соответствует максимально возможному периоду. Это и объясняет, что $2^m - 1$ кодовых слов являются различными циклическими сдвигами единственного кодового слова. Коды регистра сдвига максимальной длины существуют для любой положительной величины m.

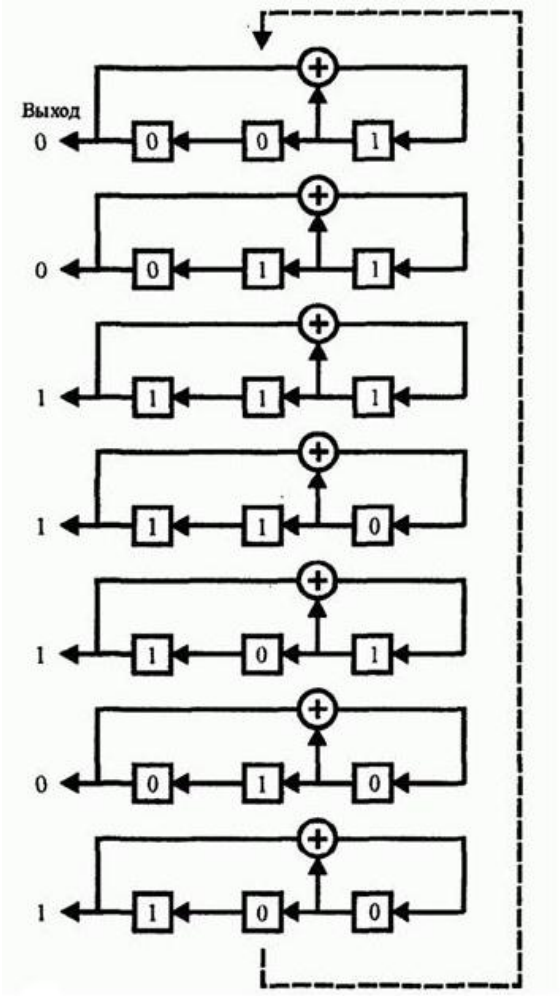


Рис.36.7.Семь шагов последовательности максимальной длины, генерируемой регистром сдвига с*m*=3

Таблица 36.4. показывает номера ячеек, которые объединяются в сумматоре по mod2, в регистре сдвига максимальной длины для $2 \le m \le 34.$

Другая особенность кодовых слов в регистре сдвига максимальной длины заключается в том, что каждое кодовое слово, за исключением слова из одних нулей, содержит $2^m - 1$ единиц и $2^m - 1$ нулей. Поскольку код линейный, его вес является также минимальным расстоянием кода, т.е.

$$d_{\min} = 2^m - 1.$$

В заключение заметим, что код регистра сдвига максимальной длины (7,3), показанный в табл. 36.3 идентичен коду (7,3), данному в табл. 36.2 и является дуальным коду Хемминга (7,4), данному в табл. 36.1. Это не совпадение. Коды регистра сдвига максимальной длины являются дуальными кодами для циклических кодов Хемминга $(2^m - 1, 2^m - 1 - m).$ Регистр сдвига для генерирования кода максимальной длины можно также использовать для генерирования периодической двоичной последовательности с периодом $n = 2^m - 1$. Двоичная периодическая последовательность имеет периодическую автокорреляционную функцию $n = \phi(m)$ со значениями $\phi(m) = n$ для $m=0$ $m = 0, \pm n, \pm 2n, \ldots$ и $\phi(m) = -1$ для других сдвигов. Эта импульсно–подобная автокорреляционная функция подразумевает, что спектр мощности близок к равномерному и, следовательно, *m*-последовательность проявляет свойства белого шума. Поэтому последовательности максимальной длины называют псевдошумовыми (ПШ) последовательностями и они находят применение для

скремблирования данных и для генерации широкополосных сигналов с рассеянным спектром.

Таблица 36.4. Соединения в сдвиговом регистре для генерирования последовательности максимальной длины

m	№№ отводов к сумматору по модулю 2	*m*	№№ отводов к сумматору по модулю 2	*m*	№ № отводов к сумматору по модулю 2	*m*	№№ отводов к сумматору по модулю 2
2	1,2	11	1, 10	20	1, 18	29	1, 28
3	1,3	12	1, 7, 9, 10	21	1, 20	30	1, 8, 29, 30
4	1,4	13	1, 10, 11, 13	22	1, 22	31	1, 29
5	1,4	14	1, 5, 9, 14	23	1, 19	32	1, 11,31,32
б	1,6	15	1, 15	24	1, 18, 23, 24	33	1, 21
7	1,7	16	1,5, 14, 16	25	1, 23	34	1, 8,33,34
8	1, 5, 6, 7	17	1, 15	26	1, 21, 25, 26		
9	1,6	18	1, 12	27	1, 23, 26, 27		
10	1,8	19	1, 15, 18, 19	28	1, 26		

Коды Боуза-Чоудхури-Хоквингема (БЧХ). БЧХ коды представляют большой класс циклических кодов как с двоичным, так и с недвоичным алфавитами.Двоичные БЧХ коды можно построить с параметрами

$N = 2^m - 1,$

$N - k \le mt,$

$d_{\min} = 2t+1.$

где $m \ge 3$ и t– произвольные положительные целые числа. Этот класс двоичных кодов предоставляет разработчику систем связи большой выбор длин блока и скоростей кода. Недвоичные БЧХ коды включают в себя мощные коды Рида-Соломона, которые будут описаны ниже.

Порождающие полиномы для БЧХ кодов можно конструировать из множителей полинома $p^{2^m-1}+1$.В таблице 36.5. Коэффициенты порождающих полиномов для БЧХ кодов длины $7 \le n \le 255,$ соответствующие $3 \le m \le 8$, приведеныв приложении. Коэффициенты даны в восьмеричной форме, причём самая левая цифра соответствует слагаемому полинома с наивысшей степенью.

Так, коэффициентами полинома для кода (15,5) является 2467, что соответствует двоичной форме 10 100 110 111. Следовательно, порождающий полином равен

$$g(p)=p^{10}+p^8+p^5+p^4+p^2+p+1.$$

Более общий список порождающих полиномов для БЧХ кодов дан Питерсоном и Уэлдоном (1972), которые дали таблицы множителей полиномов $p^{2^m-1}+1$ для $m \le 34.$

ПРИЛОЖЕНИЕ

Коэффициенты порождающих полиномов (в восьмеричной форме) для кодов БЧХ длиной

n	k	t	$g(p)$
7	4	1	13
15	11	1	23
	7	2	721
	5	3	2467
31	26	1	45
	21	2	3551
	16	3	107657
	11	5	5423325
	6	7	313365047
63	57	1	103
	51	2	12471
	45	3	1701317
	39	4	166623567

36	5	1033500423
30	6	157464165547
24	7	17323260404441
18	10	1363026512351725
16	11	6331141367235453
10	13	472622305527250155
7	15	5231045543503271737
127 120	1	211
113	2	41567
106	3	11554743
99	4	3447023271
92	5	624730022327
85	6	130704476322273
78	7	26230002166130115
71	9	6255010713253127753
64	10	1206534025570773100045
57	11	335265252505705053517721

50	13	5444651252331401242150142 1
43	14	17721772213651227521220574343
36	15	3146074666522075044764574721735
29	21	403114461367670603667530141176155
22	23	12337607040472252243544562663764 7043
15	27	22057042445604554770523013762217 6043 53
8	31	70472640527510306514762242715677331330217
255 247	1	435
239	2	267543
231	3	156720665
223	4	75626641375
215	5	23157564726421
207	6	16176560567636227
199	7	7633031270420722341
191	8	2663470176115333714567

25 18 9 52755313540001322236351
5 7
1 22634710717340432416300455
17 0
9 154162142123423560770616306 7
1
17 1 75004155100756025515747245146 01
1
1 37575130054076650157225064646776 33
16 2
3 16421301735371655253041653054410117 11
1
15 3 46140173206017556157072273024745356744 5
5
1 21571333147151015126125027744214202416547 1
14 4
7 12061405224206600371721032651614122627250 6
1 267
13 5
9 60526665572100247263636404600276352556313 4
1 72737
13 8
1 22205772322066256312417300235347420176574 7
1 50154441
12 9
3 10656667253473174222741416201574332252411 0
2 764323 03431
11 1
5 67502650303274441727236317247325110755507 6
2 272072434
10 2
7 4561
2
11013676341474323643523163430717204620672 2

99 3 545273311

91 2 721317
5
6670003563765750002027034420736617462101532671176 65
87
2 41342355
6
2402471052064432151555417211233116320544425036255 76
79
2 43221706035
7
1075447505516354432531521735770700366611172645526 76
71
2 13656702543301
63 9
7315425203501100133015275306032054325414326755010 55

55 3 7044426035473617
0
2533542017062646563033041377406233175123334145446 04
47
3 5005066024552543173
1
1520205605523416113110134637642370156367002447076 23
45
4 73033202157025051541

37 2 513633025506700741417744724543753042073570
617432343

2347644354737403044003
29 4
3 302571553667307146552706401236137711534224
232420117

21 4114060254757410403565037
4
5 125621525706033265600177315360761210322734
140565307
13
4542521153121614466513473725
4
7 464173200505256454442657371425006600433067
9 744547656

140317467721357026134460500547
5
5 157260252174724632010310432553551346141623
672120440

74545112766115547705561677516057
5
9

6
3

Список использованных источников

1. Лидовский В.В. Теория информации: Учебное пособие. – М.: Компания Спутник+, 2004. – 111 с.
2. Вернер М. Основы кодирования.Учебник для ВУЗов.–М.: Техносфера,2004.–288 с.
3. Чернецова Е.А. Системы и сети передачи информации. Монография. – СПб.: изд. РГГМУ, 2013, часть.3, Системы цифровой связи.– 133 с.
4. Спиридонов А.И. «Основы теории информации и кодирования». Учебное пособие. – СПб/Псков, Изд. СПбГПУ, 2004 – 140 с.
5. Павлов, Ю. Н. Теория информации для бакалавров: учебное пособие / Ю. Н. Павлов, Е. В. Смирнова, Е. А. Тихомирова. – Москва, Издательство МГТУ им. Н. Э. Баумана, 2016. – 173 с.
6. Понятов А.А. Теория информации и кодирования: Уч. пос. / А.А. Понятов. – М.: МИИТ, 2010. – 188 с.
7. Зверева Е.Н. , Лебедько Е.Г. Сборник примеров и задач по основам теории информации и кодирования сообщений. – СПб: НИУ ИТМО, 2014. – 76 с.
8. Помехоустойчивые коды : учеб.пособие / В. Г. Журавлев, Н. Ю. Куранова, Ю. Ю. Евсеева ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир: Изд-во ВлГУ, 2013. – 96 с.

Учебное издание

Чернецова Елена Анатольевна,
кандидат технических наук, доцент

Теория информации и кодирования
Практикум

Печатается в авторской редакции.

Подписано в печать 02.03.2021. Формат 60×90 1/16.
Гарнитура Times New Roman. Печать цифровая.
Усл. печ. л. 10,75. Тираж 15 экз. Заказ № 1037.
РГГМУ, 192007, Санкт-Петербург, Воронежская, 79.