Отцом теории информации принято считать Клода Шеннона, который, начав с углубленного изучения математической статистики, предоставил инженерам полное определение ёмкости коммуникационного канала. Его теория не рассматривает содержание передаваемого сообщения и не обращает особого внимания на процессы передачи информации, но старается оптимизировать процессы сжатия сообщения с небольшими потерями, что требует применения критерия точности. Задача перекодировки информации - как можно больше уменьшить первоначальный объём предоставленных данных. Цель таких усилий – оптимизация режимов эксплуатации устройств, предназначенных для хранения и передачи информации. Процесс сжатия бывает двух видов – с потерей данных и без. Первый вид задействован в компьютерах, которые обязаны обрабатывать разные данные одновременно. Второй вид сжатия применим к сохранению таких видов информации как фото, аудио и видео, поскольку эти виды данных способны сохранять свою информативную функцию, даже при значительных потерях. Теория информации создала задел, на базе которого была организована эффективная практика связи.

В данной лекции сначала рассмотрим общую теорию кодирования информации, затем перейдём к практическому применению кодирования в технике передачи сообщений.

Информация, в том числе графическая и звуковая, может быть представлена в аналоговой или дискретной форме. При аналоговом представлении физическая величина принимает бесконечное множество значений, и её значения изменяются непрерывно. При дискретном представлении физическая величина принимает конечное множество значений, и её величина изменяется скачкообразно. Графическая и звуковая информация преобразуется из аналоговой формы в дискретную путём дискретизации, т. е. разбиения непрерывного (аналогового) сигнала на отдельные элементы. В процессе дискретизации производится кодирование - присвоение каждому элементу конкретного значения в форме кода. Дискретизация - преобразование непрерывного потока информации (например, изображений и звука) в набор дискретных значений, каждому из которых присваивается значение его кода.

В канале связи сообщение, составленное из символов одного алфавита, может преобразовываться в сообщение из символов другого алфавита. Правило, описывающее однозначное соответствие букв алфавитов при таком преобразовании, называют кодом. Перекодировка - процедура преобразования сообщения. Подобное преобразование сообщения может осуществляться в момент поступления сообщения от источника в канал связи (кодирование) и в момент приёма сообщения получателем (декодирование). Устройства, обеспечивающие кодирование и декодирование, называют кодировщиком и декодировщиком.

1. ИСТОРИЯ КОДИРОВАНИЯ

Необходимость кодирования информации возникла задолго до появления компьютеров. Речь, азбука и цифры – есть не что иное, как система моделирования мыслей, речевых звуков и числовой информации. В технике потребность кодирования возникла сразу после создания телеграфа, но особенно важной она стала с изобретением компьютеров.

Область действия теории кодирования распространяется на передачу данных по реальным (или зашумленным) каналам, а предметом является обеспечение корректности переданной информации. Иными словами, она изучает, как лучше упаковать данные, чтобы после передачи сигнала из данных можно было надежно и просто выделить полезную информацию. Иногда теорию кодирования путают с шифрованием, но это неверно: криптография решает обратную задачу, ее цель - затруднить получение информации из данных.

С необходимостью кодирования данных впервые столкнулись более полутораста лет назад, вскоре после изобретения телеграфа. Каналы были дороги и ненадежны, что сделало актуальной задачу минимизации стоимости и повышения надёжности передачи телеграмм. Проблема ещё более обострилась в связи с прокладкой трансатлантических кабелей. С 1845 вошли в употребление специальные кодовые книги; с их помощью телеграфисты вручную выполняли «компрессию» сообщений, заменяя распространенные последовательности слов более короткими кодами. Тогда же для проверки правильности передачи стали использовать контроль чётности, метод, который применялся для проверки правильности ввода перфокарт ещё и в компьютерах первых поколений. Для этого во вводимую колоду последней вкладывали специально подготовленную карту с контрольной суммой. Если устройство ввода было не слишком надежным (или колода - слишком большой), то могла возникнуть ошибка. Чтобы исправить её, процедуру ввода повторяли до тех пор, пока подсчитанная контрольная сумма не совпадала с суммой, сохраненной на карте. Эта схема неудобна, и к тому же пропускает двойные ошибки. С развитием каналов связи потребовался более эффективный механизм контроля.

Первым теоретическое решение проблемы передачи данных по зашумленным каналам предложил Клод Шеннон, основоположник статистической теории информации. Работая в *Bell Labs*, Шеннон написал

работу «Математическая теория передачи сообщений» (1948), где показал, что если пропускная способность канала выше энтропии источника сообщений, то сообщение можно закодировать так, что оно будет передано без излишних задержек. В одной из теорем Шеннон доказал, что при наличии канала с достаточной пропускной способностью сообщение может быть передано с некоторыми временными задержками. Кроме того, он показал возможность достоверной передачи при наличии шума в канале. Формула $C = W \log ((P+N)/N)$, высечена на скромном памятнике Шеннону, установленном в его родном городе в штате Мичиган.

Труды Шеннона дали пищу для множества дальнейших исследований в области теории информации, но практического инженерного приложения они не имели. Переход от теории к практике стал возможен благодаря усилиям Ричарда Хэмминга, коллеги Шеннона по *Bell Labs*, получившего известность за открытие класса кодов «коды Хэмминга». Существует легенда, что к изобретению своих кодов Хэмминга подтолкнуло неудобство в работе с перфокартами на релейной счетной машине *Bell Model V* в середине 40-х годов. Ему давали время для работы на машине в выходные дни, когда не было операторов, и ему самому приходилось возиться с вводом. Хэмминг предложил коды, способные корректировать ошибки в каналах связи, в том числе и в магистралях передачи данных в компьютерах, прежде всего между процессором и памятью. Коды Хэмминга показали, как можно практически реализовать возможности теоремы Шеннона. Хэмминг опубликовал свою статью в 1950, хотя во внутренних отчетах его теория кодирования датируется 1947. Поэтому некоторые считают, что отцом теории кодирования следует считать Хэмминга, а не Шеннона. **Ричард Хэмминг** (1915 - 1998) получил степень бакалавра в Чикагском университете в 1937. В 1939 он получил степень магистра в Университете Небраски, а степень доктора по математике – в Университете Иллинойса. В 1945 Хэмминг начал работать в рамках Манхэттенского проекта. В1946 поступил на работу в Bell Telephone Laboratories, где работал с Шенноном. В 1976 получил кафедру в военно-морской аспирантуре в Монтерей в Калифорнии. Труд, сделавший его знаменитым, фундаментальное исследование кодов обнаружения и исправления ошибок, Хэмминг опубликовал в 1950. В 1956 он принимал участие в работе над IBM 650. Его работы заложили основу языка программирования, который позднее эволюционировал в языки программирования высокого уровня. В знак признания заслуг Хэмминга в области информатики институт IEEE учредил медаль за выдающиеся заслуги в развитии информатики и теории систем, которую назвал его именем.

Хэмминг первым предложил «коды с исправлением ошибок» (*Error-Correcting Code, ECC)*. Современные модификации этих кодов используются во всех системах хранения данных и для обмена между процессором и оперативной памятью. Один из их вариантов, коды Рида-Соломона применяются в компакт-дисках, позволяя воспроизводить записи без скрипов и шумов, вызванных царапинами и пылинками. Существует множество версий кодов, построенных «по мотивам» Хэмминга, они различаются алгоритмами кодирования и количеством проверочных битов. Особое значение подобные коды приобрели в связи с развитием дальней космической связи с межпланетными станциями.

Среди новейших кодов *ECC* следует назвать коды *LDPC (Low-Density Parity-check Code)*. Вообще-то они известны лет тридцать, но особый интерес к ним обнаружился именно в последние годы, когда стало развиваться телевидение высокой чёткости. Коды *LDPC* не обладают 100-процентной достоверностью, но вероятность ошибки может быть доведена до желаемой, и при этом с максимальной полнотой используется пропускная способность канала. К ним близки «турбокоды» (*Turbo Code*), они эффективны при работе с объектами, находящимися в условиях далекого космоса и ограниченной пропускной способности канала.

В историю теории кодирования прочно вписано имя В. А. Котельникова. В 1933 в «Материалах по радиосвязи к I Всесоюзному съезду по вопросам технической реконструкции связи» он опубликовал работу «О пропускной способности «эфира» и «проволоки». Имя Котельникова входит в название одной из важнейших теорем теории кодирования, определяющей условия, при которых переданный сигнал может быть восстановлен без потери информации. Эту теорему называют по-разному, в том числе «теоремой *WKS*» (аббревиатура *WKS* взята от *Whittaker, Kotelnikov, Shannon*). В некоторых источниках используют и *Nyquist-Shannon sampling theorem*, и *Whittaker-Shannon sampling theorem*, а в отечественных вузовских учебниках чаще всего встречается просто «теорема Котельникова». На самом же деле теорема имеет более долгую историю. Ее первую часть в 1897 доказал французский математик Э. Борель. Свой вклад в 1915 внес Э. Уиттекер. В1920 японец К. Огура опубликовал поправки к исследованиям Уиттекера, а в 1928 американец Гарри Найквист уточнил принципы оцифровки и восстановления аналогового сигнала.

2. ТЕОРИЯ КОДИРОВАНИЯ

Теория кодирования информации является одним из разделов теоретической информатики. К основным задачам, решаемым в данном разделе, необходимо отнести следующие:

разработка принципов наиболее экономичного кодирования информации;

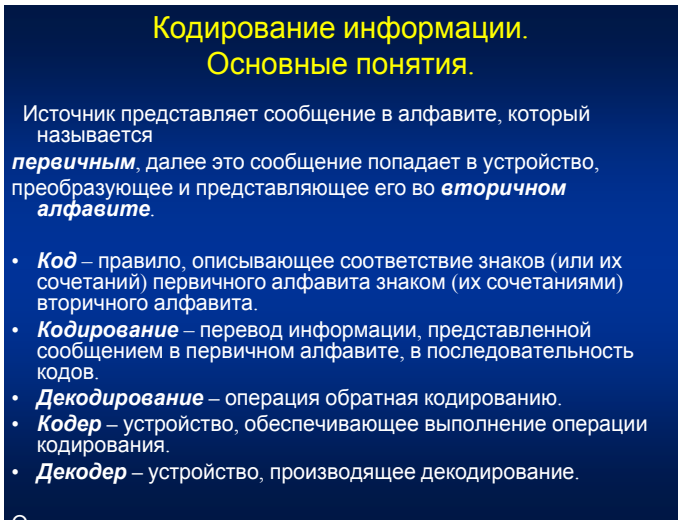
согласование параметров передаваемой информации с особенностями канала связи;

разработка приемов, обеспечивающих надежность передачи информации по каналам связи, т.е. отсутствие потерь информации.

Две последние задачи связаны с процессами передачи информации. Первая же задача – кодирование информации – касается не только передачи, но и обработки, и хранения информации, т.е. охватывает широкий круг проблем; частным их решением будет представление информации в компьютере. С обсуждения этих вопросов и начнем освоение теории кодирования.

2.1 Определение понятий

Кодирование информации - процесс преобразования сигнала из формы, удобной для непосредственного использования информации, в форму, удобную для передачи, хранения или автоматической переработки (Цифровое кодирование, аналоговое кодирование, таблично-символьное кодирование, числовое кодирование). Процесс преобразования сообщения в комбинацию символов в соответствии с кодом называется *кодированием*, процесс восстановления сообщения из комбинации символов называется *декодированием*

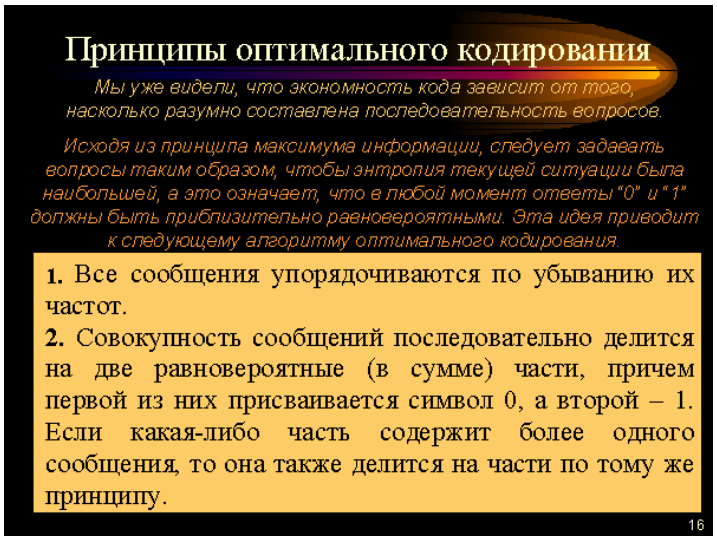


Информацию необходимо представлять в какой-либо форме, т.е. кодировать.

Для представления дискретной информации используется некоторый алфавит. Однако однозначное соответствие между информацией и алфавитом отсутствует. Другими словами, одна и та же информация может быть представлена посредством различных алфавитов. В связи с такой возможностью возникает проблема перехода от одного алфавита к другому, причём, такое преобразование не должно приводить к потере информации.

Алфавит, с помощью которого представляется информация до преобразования называется первичным; алфавит конечного представления – вторичным.

Код – (1) правило, описывающее соответствие знаков или их сочетаний одного алфавита знакам или их сочетаниям другого алфавита; - (2) знаки вторичного алфавита, используемые для представления знаков или их сочетаний первичного алфавита.



Код – совокупность знаков (символов) и система определённых правил, при помощи которой информация может быть представлена (закодирована) в виде набора из таких символов для передачи, обработки и хранения. Конечная последовательность кодовых знаков называется словом. Наиболее часто для кодирования информации используют буквы, цифры, числа, знаки и их комбинации.

Код – набор символов, которому приписан некоторый смысл. Код является знаковой системой, которая содержит конечное число символов: буквы алфавита, цифры, знаки препинания, знаки препинания, знаки математических операций и т.д.

Кодирование – операция отожествления символов или групп символов одного кода с символами или группами символов другого кода.

Кодирование – перевод информации, представленной посредством первичного алфавита, в последовательность кодов.

Кодирование информации – процесс формирования определенного представления информации. В более узком смысле под термином «кодирование» понимают переход от одной формы представления информации к другой, более удобной для хранения, передачи или обработки.

Кодирование информации – процесс преобразования сигналов или знаков одной знаковой системы в знаки другой знаковой системы, для использования, хранения, передачи или обработки.

Декодирование - операция, обратная кодированию, т.е. восстановление информации из закодированного вида (восстановление в первичном алфавите по полученной последовательности кодов).

Шифрование – разновидность кодирования.

Шифр – код, значение и правила использования которого известно ограниченному кругу лиц.

Операции кодирования и декодирования называются обратимыми, если их последовательное применение обеспечивает возврат к исходной информации без каких-либо её потерь.

Примером обратимого кодирования является представление знаков в телеграфном коде и их восстановление после передачи. Примером кодирования необратимого может служить перевод с одного естественного языка на другой – обратный перевод, вообще говоря, не восстанавливает исходного текста. Безусловно, для практических задач, связанных со знаковым представлением информации, возможность восстановления информации по ее коду является необходимым условием применения кода, поэтому в дальнейшем изложении ограничим себя рассмотрением только обратимого кодирования.

Таким образом, кодирование предшествует передаче и хранению информации. При этом хранение связано с фиксацией некоторого состояния носителя информации, а передача – с изменением состояния с течением времени (т.е. процессом). Эти состояния или сигналы будем называть ***элементарными сигналами*** – именно их совокупность и составляет вторичный алфавит.

Любой код должен обеспечивать однозначное чтение сообщения (надежность), так и, желательно, быть экономным (использовать в среднем поменьше символов на сообщение).

Возможность восстановить текст означает, что в языке имеется определенная избыточность, за счет которой мы восстанавливаем отсутствующие элементы по оставшимся. Ясно, что избыточность находится в вероятностях букв и их комбинациях, их знание позволяет подобрать наиболее вероятный ответ.

Компьютер может обрабатывать только информацию, представленную в числовой форме. Вся другая информация (звуки, изображения, показания приборов и т. д.) для обработки на компьютере должна быть преобразована в числовую форму. С помощью компьютерных программ можно преобразовывать полученную информацию, в том числе - текстовую. При вводе в компьютер каждая буква кодируется определенным числом, а при выводе на внешние устройства (экран или печать) для восприятия человеком по этим числам строятся изображения букв. Соответствие между набором букв и числами называется кодировкой символов. Как правило, все числа в компьютере представляются с помощью нулей и единиц, т.е. словами, компьютеры работают в двоичной системе счисления, поскольку при этом устройства для их обработки получаются значительно более простыми.

Математическая постановка задачи кодирования

- A - первичный алфавит. Состоит из N знаков со средней информацией на знак I^A .
- B- вторичный из M знаков со средней информацией на знак I^B .
- Сообщение в первичном алфавите содержит n знаков, а закодированное – m знаков.
- $I_s(A)$-информация в исходном сообщении, $I_f(B)$-информация в закодированном сообщении.

Кодер - программист, специализирующийся на кодировании - написании исходного кода по заданным спецификациям.

Кодер - одна из двух компонент кодека (пары кодер – декодер).

Декодер - некоторое звено, которое преобразует информацию из внешнего вида в вид, применяемый внутри узла. В программном обеспечении: модуль программы или самостоятельное приложение, которое преобразует файл или информационный поток из внешнего вида в вид, который поддерживает другое программное обеспечение.

В процессе преобразования информации из одной формы представления (знаковой системы) в другую осуществляется кодирование. Средством кодирования служит таблица соответствия, которая устанавливает взаимно однозначное соответствие между знаками или группами знаков двух различных знаковых систем. В процессе обмена информацией часто приходится производить операции кодирования и декодирования информации. При вводе знака алфавита в компьютер путем нажатия соответствующей клавиши на клавиатуре выполняется его кодирование, т. е. преобразование в компьютерный код. При выводе знака на экран монитора или принтер происходит обратный процесс - декодирование, когда из компьютерного кода знак преобразуется в графическое изображение.

- $I_S(A) \le I_f(B)$ – условие обратимости кодирования, т.е не исчезновения информации.

  $n^* I(A) \le m^* I(B)$ (заменили произведением числа знаков на среднее информационное содержание знака).
- m/n –характеризует среднее число знаков вторичного алфавита, который используется для кодирования одного знака первичного. Обозначим его K (A, B)
- $K(A, B) \ge I(A) / I(B)$ Обычно $K(A, B) > 1$
- $K^{min}(A, B) = I^{(A)} / I^{(B)}$ – минимальная длинна кода

Кодирование информации распадается на этапы:

Определение объёма информации, подлежащей кодированию.

Классификация и систематизация информации.

Выбор системы кодирования и разработка кодовых обозначений.

Непосредственное кодирование.

2.2 Информация и алфавит

Теорема Шеннона

При отсутствии помех. Всегда возможен такой вариант кодирования сообщения, при котором среднее число знаков кода, приходящихся на один знак первичного алфавита, будет сколь угодно близко к отношению средних информаций на знак первичного и вторичного алфавитов.

Шенноном была рассмотрена ситуация, когда при кодировании сообщения в первичном алфавите учитывается различная вероятность появления знаков, а также равная вероятность Появления знаков вторичного алфавита.

Тогда:

$K^{min}(A, B) = I^{(A)} / \log 2\, M = I^{(A)}$, здесь $I^{(A)}$ - средняя

Хотя естественной для органов чувств человека является аналоговая форма, универсальной все же следует считать дискретную форму представления информации с помощью некоторого набора знаков. В частности, именно таким образом представленная информация обрабатывается компьютером, передаётся по компьютерным и некоторым иным линиям связи. Сообщение - последовательность знаков алфавита. При их передаче возникает проблема распознавания знака: каким образом прочитать сообщение, т.е. по полученным сигналам установить исходную последовательность знаков первичного алфавита. В устной речи это достигается использованием различных фонем (основных звуков разного звучания), по которым мы и отличает знаки речи. В письменности это достигается различным начертанием букв и дальнейшим анализом написанного. Можно реализовать некоторую процедуру, посредством которой выделить из сообщения тот или иной знак. Но появление конкретного знака (буквы) в конкретном месте сообщения – событие случайное. Следовательно, узнавание (отождествление) знака требует получения некоторой порции информации. Можно связать эту информацию с самим знаком и считать, что знак несет в себе некоторое количество информации.

Алфавит [*alphabetos*, от названий первых двух букв греческого А. - альфа и бета; аналогично: азбука — от аз и буки], совокупность графических знаков — букв (например, латинский, русский А.) или слоговых знаков (например, индийский А. деванагари), расположенных в традиционно установленном порядке.

Знак - соглашение (явное или неявное) о приписывании чему-либо (означающему) какого-либо определённого смысла (означаемого). Знак — это материально выраженная замена предметов, явлений, понятий в процессе обмена информацией в коллективе. Знаком также называют конкретный случай использования такого соглашения для передачи информации. Знак может быть составным, то есть состоять из нескольких других знаков. Буквы и слова человеческого языка являются знаками. Цифры и числа являются знаками. Наука о знаковых системах называется семиотикой.

Символы (в компьютере) - цифры, буквы, иероглифы и т.п.

Начнём с самого грубого приближения (будем называть его нулевым) – предположим, что появление всех знаков (букв) алфавита в сообщении равновероятно. Тогда для английского алфавита n_e=27 (с учетом пробела как самостоятельного знака); для русского алфавита n_r=34. Из формулы Хартли

$$I = \log_2 n \qquad (1.15)$$

находим:

$$I_0^{(e)} = \log_2 27 = 4{,}755 \text{ бит.}$$
$$I_0^{(r)} = \log_2 34 = 5{,}087 \text{ бит.}$$

Получается, что в нулевом приближении со знаком русского алфавита в среднем связано больше информации, чем со знаком английского. Например, в русской букве «*а*» информации больше, чем в «*a*» английской! Это, безусловно, не означает, что английский язык – язык Шекспира и Диккенса – беднее, чем язык Пушкина и Достоевского. Лингвистическое богатство языка определяется количеством слов и их сочетаний, а это никак не связано с числом букв в алфавите. С точки зрения техники это означает, что сообщения из равного количества символов будет иметь разную длину (и соответственно, время передачи) и большими они окажутся у сообщений на русском языке.

В качестве следующего (первого) приближения, уточняющего исходное, попробуем учесть то обстоятельство, что относительная частота, т.е. вероятность появления различных букв в тексте (или сообщении) различна. Рассмотрим таблицу средних частот букв для русского алфавита, в который включен также знак «пробел» для разделения слов; с учетом неразличимости букв «е» и «ё», а также «ь» и «ъ» (так принято в телеграфном кодировании), получим алфавит из 32 знаков со следующими вероятностями их появления в русских текстах:

Табл. 1. Частота появления букв

Буква	Частота	Буква	Частота	Буква	Частота	Буква	Частота
пробел	0,175	о	0,090	е, ё	0,072	а	0,062
и	0,062	т	0,053	н	0,053	с	0,045
р	0,040	в	0,038	л	0,035	к	0,028
м	0,026	д	0,025	п	0,023	у	0,021

я	0,018	ы	0,016	з	0,016	ъ, ь	0,014
б	0,014	г	0,013	ч	0,012	й	0,010
х	0,009	ж	0,007	ю	0,006	ш	0,006
ц	0,004	щ	0,003	э	0,003	ф	0,002

Для оценки информации, связанной с выбором одного знака алфавита с учётом неравной вероятности их появления в сообщении (текстах) можно воспользоваться формулой

$$I = -\sum_{i=1}^{n} p(A_i) \cdot \log_2 p(A_i) \qquad (1.14).$$

(Информация опыта равна среднему значению количества информации, содержащейся в каком-либо одном его исходе).

Из неё, в частности, следует, что если p_i – вероятность (относительная частота) знака номер i данного алфавита из N знаков, то среднее количество информации, приходящейся на один знак, равно:

Теорема Шеннона (переформулировка)

При отсутствии помех средняя длина двоичного кода может быть сколь угодно близкой к средней информации, приходящейся на знак первичного алфавита.

$$I = \sum_{i=1}^{n} p_i \cdot \log_2 p_i \qquad (1.17)$$

- формула К. Шеннона.

Применение формулы (1.17) к алфавиту русского языка дает значение средней информации на знак $I_1^{(r)} = 4{,}36$ бит, а для английского языка $I_1^{(e)} = 4{,}04$ бит, для французского $I_1^{(l)} = 3{,}96$ бит, для немецкого $I_1^{(d)} = 4{,}10$ бит, для испанского $I_1^{(s)} = 3{,}98$ бит. Как мы видим, и для русского, и для английского языков учёт вероятностей появления букв в сообщениях приводит к уменьшению среднего информационного содержания буквы, что, кстати, подтверждает справедливость формулы (1.7). Несовпадение значений средней информации для английского, французского и немецкого языков, основанных на одном алфавите, связано с тем, что частоты появления одинаковых букв в них различаются.

В рассматриваемом приближении по умолчанию предполагается, что вероятность появления любого знака в любом месте сообщения остается одинаковой и не зависит от того, какие знаки или их сочетания предшествуют данному. Такие сообщения называются ***шенноновскими*** (или сообщениями без памяти).

Сообщения, в которых вероятность появления каждого отдельного знака не меняется со временем, называются ***шенноновскими***, а порождающий их отправитель – ***шенноновским источником***.

Если сообщение является шенноновским, то набор знаков (алфавит) и вероятности их появления в сообщении могут считаться известными заранее. В этом случае, с одной стороны, можно предложить оптимальные способы кодирования, уменьшающие суммарную длину сообщения при передаче по каналу связи. С другой стороны, интерпретация сообщения, представляющего собой последовательность сигналов, сводится к задаче распознавания знака, т.е. выявлению, какой именно знак находится в данном месте сообщения. А такая задача, как мы уже убедились в предыдущем шаге, может быть решена серией парных выборов. При этом количество информации, содержащееся в знаке, служит мерой затрат по его выявлению.

Особенности вторичного алфавита при кодировании

1. Элементарные коды 0 и 1 могут иметь одинаковые длительности (t0 = t1) или разные (≠).
2. Длина кода может быть одинаковой для всех знаков первичного алфавита (***код равномерный***) или различной (***неравномерный код***)
3. Коды могут строиться для отдельного знака первичного алфавита (***алфавитное кодирование***) или

Последующие (второе и далее) приближения при оценке значения информации, приходящейся на знак алфавита, строятся путем учета корреляций, т.е. связей между буквами в словах. Дело в том, что в словах буквы появляются не в любых сочетаниях; это понижает неопределенность угадывания следующей буквы после нескольких, например, в русском языке нет слов, в которых встречается сочетание щщ или фъ. И напротив, после некоторых сочетаний можно с большей определенностью, чем чистый случай, судить о появлении следующей буквы, например, после распространенного сочетания пр- всегда следует гласная буква, а их в русском языке 10 и, следовательно, вероятность угадывания следующей буквы 1/10, а не 1/33. В связи с этим примем следующее определение:

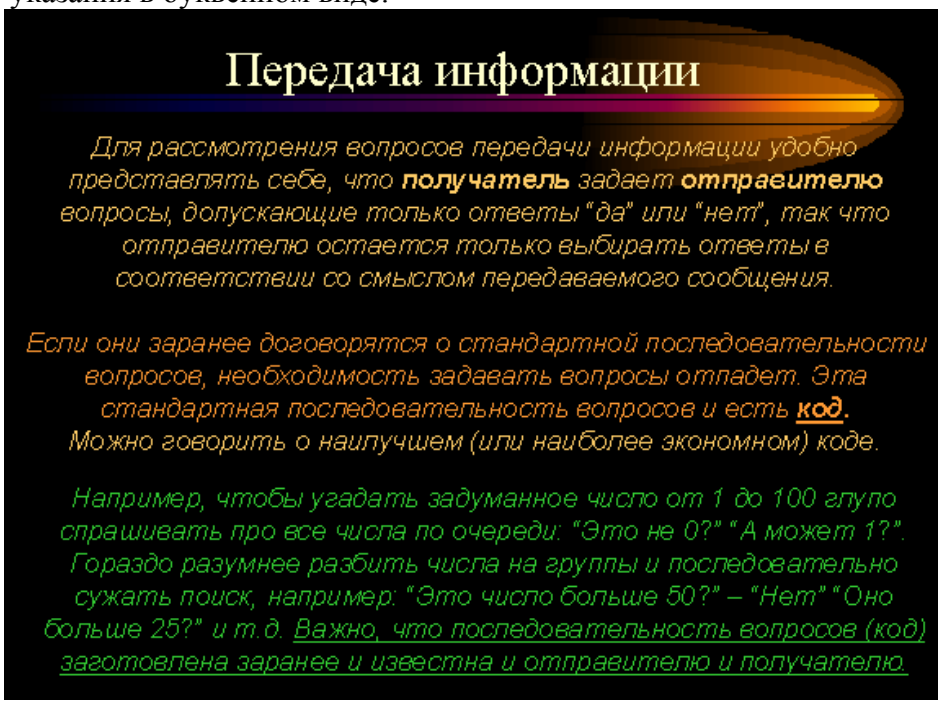
Сообщения (а также источники, их порождающие), в которых существуют статистические связи (корреляции) между знаками или их сочетаниями, называются сообщениями (источниками) с памятью или марковскими сообщениями (источниками).

Учёт в английских словах двухбуквенных сочетаний понижает среднюю информацию на знак до значения $I_2^{(e)}$=3,32 бит, учёт трехбуквенных – до $I_3^{(e)}$=3,10 бит. Шеннон сумел приблизительно оценить $I_5^{(e)}$ ≈ 2,1 бит, $I_8^{(e)}$ ≈1,9 бит. Аналогичные исследования для русского языка дают: $I_2^{(r)}$ = 3,52 бит; $I_3^{(r)}$= 3,01 бит.

Последовательность I_0, I_1, I_2... является убывающей в любом языке. Экстраполируя её на учёт бесконечного числа корреляций, можно оценить предельную информацию на знак в данном языке I_∞, которая будет отражать минимальную неопределенность, связанную с выбором знака алфавита без учета семантических особенностей языка, в то время как I_0 является другим предельным случаем, поскольку характеризует наибольшую информацию, которая может содержаться в знаке данного алфавита. Шеннон ввел величину, которую назвал *относительной избыточностью языка*:

$$R = 1 - \frac{I_\infty}{I_0} \qquad (1.18)$$

Избыточность является мерой бесполезно совершаемых альтернативных выборов при чтении текста. Эта величина показывает, какую долю лишней информации содержат тексты данного языка; лишней в том отношении, что она определяется структурой самого языка и, следовательно, может быть восстановлена без явного указания в буквенном виде.



Исследования Шеннона для английского языка дали значение I_∞≈1,4÷1,5 бит, что по отношению к I_0=4,755 бит создает избыточность 0,68. Подобные оценки показывают, что и для других европейских языков, в том числе русского, избыточность составляет 60 – 70%. Это означает, что возможно почти трехкратное (!) сокращение текстов без ущерба для их содержательной стороны и выразительности. Например, телеграфные тексты делаются короче за счёт отбрасывания союзов и предлогов без ущерба для смысла; в них же используются однозначно интерпретируемые сокращения «ЗПТ» и «ТЧК» вместо полных слов (эти сокращения приходится использовать, поскольку знаки «.» и «,» не входят в телеграфный алфавит). Однако такое «экономичное» представление слов снижает разборчивость языка, уменьшает возможность понимания речи при наличии шума (а это одна из проблем передачи информации по реальным линиям связи), а также исключает возможность локализации и исправления ошибки (написания или передачи) при её возникновении. Именно избыточность языка позволяет легко восстановить текст, даже если он содержит большое число ошибок или неполон. В этом смысле избыточность есть определенная страховка и гарантия разборчивости.

На практике учёт корреляций в сочетаниях знаков сообщения весьма затруднителен, поскольку требует объемных статистических исследований текстов. Кроме того, корреляционные вероятности зависят от характера текстов и целого ряда иных их особенностей. По этим причинам в дальнейшем мы ограничим себя рассмотрением только шенноновских сообщений, т.е. будем учитывать различную (априорную) вероятность появления знаков в тексте, но не их корреляции.

2.3 Двоичная система счисления

Систе́ма счисле́ния - символический метод записи чисел, представление чисел с помощью письменных знаков. Система счисления: даёт представления множества чисел (целых или вещественных); даёт каждому числу уникальное представление (или, по крайней мере, стандартное представление); отражает алгебраическую и арифметическую структуру чисел.

Мы не будем сейчас рассматривать все существующие системы счисления, а сразу перейдём к двоичной системе, имея ввиду, что она широко применяется в практике кодирования сигналов.

Двоичная система счисления - это позиционная система счисления с основанием 2.

Позиционная система счисления – система счисления, в которой один и тот же числовой знак (цифра) в записи числа имеет различные значения в зависимости от того места (разряда), где он расположен.

В этой системе счисления натуральные числа записываются с помощью всего лишь двух символов (в роли которых обычно выступают цифры 0 и 1).

Двоичная система используется в цифровых устройствах, т.к. является наиболее простой и соответствует требованиям: 1) Чем меньше значений существует в системе, тем проще изготовить отдельные элементы, оперирующие этими значениями. В частности, две цифры двоичной системы счисления могут быть легко представлены многими физическими явлениями: есть ток — нет тока, индукция магнитного поля больше пороговой величины или нет и т. д. 2) Чем меньше количество состояний у элемента, тем выше помехоустойчивость и тем быстрее он может работать. Например, чтобы закодировать три состояния через величину индукции магнитного поля, потребуется ввести два пороговых значения, что не будет способствовать помехоустойчивости и надёжности хранения информации. 3) Двоичная арифметика является довольно простой. Простыми являются таблицы сложения и умножения - основных действий над числами. 4) Возможно применение аппарата алгебры логики для выполнения побитовых операций над числами.

В цифровой электронике одному двоичному разряду в двоичной системе счисления соответствует один двоичный логический элемент (инвертор с логикой на входе) с двумя состояниями (открыт, закрыт).

1 + 0 = 1
1 + 1 = 10
10 + 10 = 100

Таблица умножения двоичных чисел

0 • 0 = 0
0 • 1 = 0
1 • 0 = 0
1 • 1 = 1

Использование двоичной системы при измерении дюймами

При указании линейных размеров в дюймах по традиции используют двоичные дроби, а не десятичные, например: $5\frac{3}{4}''$, $7\frac{15}{16}''$, $3\frac{11}{32}''$ и т.д.

Преобразование чисел

Для преобразования из двоичной системы в десятичную используют следующую таблицу степеней основания 2:

512	256	128	64	32	16	8	4	2	1

Начиная с цифры 1 все цифры умножаются на два. Точка, которая стоит после 1 называется двоичной точкой.

Преобразование двоичных чисел в десятичные

Допустим, вам дано двоичное число 110001. Для перевода в десятичное просто запишите его справа налево как сумму по разрядам следующим образом:

$$1\times2^0+0\times2^1+0\times2^2+0\times2^3+1\times2^4+1\times2^5 = 1\times1+0\times2+0\times4+0\times8+1\times16+1\times32 = 49.$$

Можно записать это в виде таблицы следующим образом:

512	256	128	64	32	16	8	4	2	1
				1	1	0	0	0	1
				+32	+16				+1

Точно так же, начиная с двоичной точки, двигайтесь справа налево. Под каждой двоичной единицей напишите её эквивалент в строчке ниже. Сложите получившиеся десятичные числа. Таким образом, двоичное число 110001 равнозначно десятичному 49.

Для того, что бы преобразовывать числа из двоичной в десятичную систему данным методом, надо суммировать цифры слева-направо, умножая ранее полученный результат на основу системы (в данном случае 2). Например, двоичное число 1011011 переводится в десятичную систему так:

0*2+1=1>>1*2+0=2>>2*2+1=5>>5*2+1=11>>11*2+0=22>>22*2+1=45>> 45*2+1=91. В десятичной системе это число будет записано как 91. Или число 101111 переводится в десятичную систему так: 0*2+1=1>>1*2+0=2>>2*2+1=5>>5*2+1=11>> 11*2+1=23 >> 23*2+1=47 То есть в десятичной системе это число будет записано как 47.

Преобразование десятичных чисел к ближайшей степени двойки, не меньшей этого числа. Ниже приведена функция, возвращающая число, не меньшее аргумента, и являющееся степенью двух.

unsigned int to_deg_2(unsigned int num){int i; if (num == 1) return 2;for(num-=1,i=1; i < sizeof (unsigned int)*8; i*=2) num = num|(num>>i); return num+1;}

Допустим, нужно перевести число 19 в двоичное. Следует воспользоваться следующей процедурой:

19 /2 = 9 с остатком 1
9 /2 = 4 с остатком 1
4 /2 = 2 с остатком 0
2 /2 = 1 с остатком 0
1 /2 = 0 с остатком 1

Итак, мы делим каждое частное на 2 и записываем в остаток 1 или 0. Продолжать деление надо пока в делимом не будет 1. Ставим числа из остатка друг за другом, начиная с конца. В результате получаем число 19 в двоичной записи (начиная с конца): 10011.

3. ПРАКТИКА КОДИРОВАНИЯ

3.1 Кодирование сигнала

Кодирование сигнала – это его представление в определенной форме, удобной для последующего использования сигнала, т.е. это правило, описывающее отображение одного набора знаков в другой набор знаков. Тогда отображаемый набор знаков называется исходным алфавитом, а набор знаков, который используется для отображения, - кодовым алфавитом, или алфавитом для кодирования. При этом кодированию подлежат как отдельные символы исходного алфавита, так и их комбинации. Аналогично для построения кода используются как отдельные символы кодового алфавита, так и их комбинации. Например, дана таблица соответствия между натуральными числами трёх систем счисления. Эту таблицу можно рассматривать как некоторое правило, описывающее отображение набора знаков десятичной системы счисления в двоичную и шестнадцатеричную. Тогда исходный алфавит - десятичные цифры от 0 до 9, а кодовые алфавиты - это 0 и 1 для двоичной системы; цифры от 0 до 9 и символы {*A, B, C, D, E, F*} - для шестнадцатеричной.

Кодовой комбинацией (кодом) называется совокупность символов кодового алфавита, применяемых для кодирования одного символа (или одной комбинации символов) исходного алфавита. При этом кодовая комбинация может содержать один символ кодового алфавита. Исходным символом называется символ (или комбинация символов) исходного алфавита, которому соответствует кодовая комбинация. Например, поскольку $8 = 1000_2$ и 8 является исходным символом, 1000 - это кодовая комбинация, или код, для числа 8. В то же время 8 - это исходный символ. Совокупность кодовых комбинаций называется кодом. Взаимосвязь символов (или комбинаций символов, если кодируются не отдельные символы) исходного алфавита с их кодовыми комбинациями составляет таблицу соответствия (таблицу кодов). Обратная процедура получения исходных символов по кодам символов называется декодированием. Очевидно, для выполнения правильного декодирования код должен быть однозначным, т.е. одному исходному символу должен соответствовать точно один код и наоборот.

В зависимости от целей кодирования, различают следующие его виды:

- кодирование по образцу - используется всякий раз при вводе информации в компьютер для её внутреннего представления;
- криптографическое кодирование, или шифрование, – используется, когда нужно защитить информацию от несанкционированного доступа;
- эффективное, или оптимальное, кодирование – используется для устранения избыточности информации, т.е. снижения ее объема, например, в архиваторах;
- помехозащитное, или помехоустойчивое, кодирование – используется для обеспечения заданной достоверности в случае, когда на сигнал накладывается помеха, например, при передаче информации по каналам связи.

3.2 Первая теорема Шеннона

В предыдущей лекции мы рассмотрели все теоремы Шеннона, сейчас обсудим одну из них – первую теорему, теперь уже конкретно к применению в кодировании.

Первая теорема Шеннона о передаче информации, которая называется также основной теоремой о кодировании при отсутствии помех, формулируется следующим образом:

При отсутствии помех передачи всегда возможен такой вариант кодирования сообщения, при котором среднее число знаков кода, приходящихся на один знак кодируемого алфавита, будет сколь угодно близко к отношению средних информаций на знак первичного и вторичного алфавитов.

Используя понятие избыточности кода, можно дать более короткую формулировку теоремы:

При отсутствии помех передачи всегда возможен такой вариант кодирования сообщения, при котором избыточность кода будет сколь угодно близкой к нулю.

Эта теорема открывает принципиальную возможность оптимального кодирования. Но из неё никоим образом не следует, как такое кодирование осуществить практически – для этого должны привлекаться какие-то дополнительные соображения.

Далее в основном ограничим себя ситуацией, когда $M = 2$, т.е. для представления кодов в линии связи используется лишь два типа сигналов – с практической точки зрения это наиболее просто реализуемый вариант (например, существование напряжения в проводе (будем называть это импульсом) или его отсутствие (пауза); наличие или отсутствие отверстия на перфокарте или намагниченной области на дискете); подобное кодирование называется двоичным. Знаки двоичного алфавита принято обозначать «0» и «1», но нужно воспринимать их как буквы, а не цифры. Удобство двоичных кодов и в том, что при равных длительностях и вероятностях каждый элементарный сигнал (0 или 1) несет в себе 1 бит информации ($\log_2 M=1$); тогда из теоремы Шеннона:

$$I_1^{(A)} \le K^{(2)}$$

и первая теорема Шеннона получает следующую интерпретацию:

При отсутствии помех передачи средняя длина двоичного кода может быть сколь угодно близкой к средней информации, приходящейся на знак первичного алфавита.

В двоичной системе кодирования:

$$Q = 1 - \frac{I_1^{(A)}}{K^{(2)}}$$

Определение количества переданной информации при двоичном кодировании сводится к простому подсчету числа импульсов (единиц) и пауз (нулей). При этом возникает проблема выделения из потока сигналов (последовательности импульсов и пауз) отдельных кодов. Приемное устройство фиксирует интенсивность и длительность сигналов. Элементарные сигналы (0 и 1) могут иметь одинаковые или разные длительности. Их количество в коде (длина кодовой цепочки), который ставится в соответствие знаку первичного алфавита, также может быть одинаковым (в этом случае код называется равномерным) или разным (неравномерный код). Наконец, коды могут строиться для каждого знака исходного алфавита (алфавитное кодирование) или для их комбинаций (кодирование блоков, слов). В результате при кодировании (алфавитном и словесном) возможны следующие варианты сочетаний:

Табл. 1. Варианты сочетаний

Длительности элементарных сигналов	Кодировка первичных символов (слов)	Ситуация
одинаковые	равномерная	(1)
одинаковые	неравномерная	(2)
разные	равномерная	(3)
разные	неравномерная	(4)

В случае использования неравномерного кодирования или сигналов разной длительности (ситуации (2), (3) и (4)) для отделения кода одного знака от другого между ними необходимо передавать специальный сигнал – временной разделитель (признак конца знака) или применять такие коды, которые оказываются уникальными, т.е. несовпадающими с частями других кодов. При равномерном кодировании одинаковыми по длительности сигналами (ситуация (1)) передачи специального разделителя не требуется, поскольку отделение одного кода от другого производится по общей длительности, которая для всех кодов оказывается одинаковой (или одинаковому числу бит при хранении).

Длительность двоичного элементарного импульса (τ) показывает, сколько времени требуется для передачи 1 бит информации. Очевидно, для передачи информации, в среднем приходящейся на знак первичного алфавита, необходимо время $K^{(r)}\tau$. Таким образом, можно построить такую систему кодирования, чтобы суммарная длительность кодов при передаче (или суммарное число кодов при хранении) данного сообщения была бы наименьшей.

3.3 Способы кодирования/декодирования информации

Коды по образцу. Данный вид кодирования применяется для представления дискретного сигнала на том или ином машинном носителе. Большинство кодов, используемых в информатике для кодирования по образцу, имеют одинаковую длину и используют двоичную систему для представления кода (и, возможно, шестнадцатеричную как средство промежуточного представления).

Криптографические коды. Криптографические коды используются для защиты сообщений от несанкционированного доступа, потому называются также шифрованными. В качестве символов кодирования могут использоваться как символы произвольного алфавита, так и двоичные коды. Существуют различные методы, рассмотрим два из них: метод простой подстановки и метод Вижинера.

Эффективное кодирование. Этот вид кодирования используется для уменьшения объемов информации на носителе - сигнале. Для кодирования символов исходного алфавита используют двоичные коды переменной длины: чем больше частота символа, тем короче его код.

Эффективность кода определяется средним числом двоичных разрядов для кодирования одного символа – $I_{ср}$ по формуле

$$I_{ñð} = \sum_{i=1}^{k} f_i n_1,$$

где k – число символов исходного алфавита; n_s – число двоичных разрядов для кодирования символа s; f_s – частота символа s; причем $\sum_{i=1}^{ë} f_i = 1$.

При эффективном кодировании существует предел сжатия, ниже которого не «спускается» ни один метод эффективного кодирования - иначе будет потеряна информация. Этот параметр определяется **предельным значением двоичных разрядов** возможного эффективного кода – $I_{пр}$:

$$I_{ïð} = -\sum_{i=1}^{n} f_i \log_2 f_i,$$

где n – мощность кодируемого алфавита, f_i – частота i-го символа кодируемого алфавита.

Существуют два классических метода эффективного кодирования: метод Шеннона-Фано и метод Хаффмена. Входными данными для обоих методов является заданное множество исходных символов для кодирования с их частотами; результат - эффективные коды.

Метод Шеннона-Фано. Этот метод требует упорядочения исходного множества символов по не возрастанию их частот. Затем выполняются следующие шаги:

а) список символов делится на две части (назовем их первой и второй частями) так, чтобы суммы частот обеих частей (назовем их Σ_1 и Σ_2) были точно или примерно равны. В случае, когда точного равенства достичь не удается, разница между суммами должна быть минимальна;

б) кодовым комбинациям первой части дописывается 1, кодовым комбинациям второй части дописывается 0;

в) анализируют первую часть: если она содержит только один символ, работа с ней заканчивается, – считается, что код для ее символов построен, и выполняется переход к шагу г) для построения кода второй части. Если символов больше одного, переходят к шагу а) и процедура повторяется с первой частью как с самостоятельным упорядоченным списком;

г) анализируют вторую часть: если она содержит только один символ, работа с ней заканчивается и выполняется обращение к оставшемуся списку (шаг д). Если символов больше одного, переходят к шагу а) и процедура повторяется со второй частью как с самостоятельным списком;

д) анализируется оставшийся список: если он пуст – код построен, работа заканчивается. Если нет, – выполняется шаг а).

Пример 1. Даны символы *a, b, c, d* с частотами f_a = 0,5; f_b = 0,25; f_c = 0,125; f_d = 0,125. Построить эффективный код методом Шеннона-Фано. Сведем исходные данные в таблицу, упорядочив их по невозрастанию частот:

Исходные символы	Частоты символов
a	0,5
b	0,25
c	0,125
d	0,125

Первая линия деления проходит под символом *a*: соответствующие суммы Σ_1 и Σ_2 равны между собой и равны 0,5. Тогда формируемым кодовым комбинациям дописывается 1 для верхней (первой) части и 0 для нижней (второй)

части. Поскольку это первый шаг формирования кода, двоичные цифры не дописываются, а только начинают формировать код:

Исходные символы	Частоты символов	Формируемый код
a	0,5	1
b	0,25	0
c	0,125	0
d	0,125	0

В силу того, что верхняя часть списка содержит только один элемент (символ *a*), работа с ней заканчивается, а эффективный код для этого символа считается сформированным (в таблице, приведенной выше, эта часть списка частот символов выделена заливкой). Второе деление выполняется под символом *b*: суммы частот Σ_1 и Σ_2 вновь равны между собой и равны 0,25. Тогда кодовой комбинации символов верхней части дописывается 1, а нижней части – 0. Таким образом, к полученным на первом шаге фрагментам кода, равным 0, добавляются новые символы:

Исходные символы	Частоты символов	Формируемый код
a	0,5	1
b	0,25	01
c	0,125	00
d	0,125	00

Поскольку верхняя часть нового списка содержит только один символ (*b*), формирование кода для него закончено (соответствующая строка таблицы вновь выделена заливкой). Третье деление проходит между символами *c* и *d*: к кодовой комбинации символа *c* приписывается 1, коду символа *d* приписывается 0:

Исходные символы	Частоты символов	Формируемый код
a	0,5	1
b	0,25	01
c	0,125	001
d	0,125	000

Поскольку обе оставшиеся половины исходного списка содержат по одному элементу, работа со списком в целом заканчивается.

Таким образом, получили коды:

a - 1, *b* - 01, *c* - 001, *d* - 000.

Определим эффективность построенного кода по формуле:

$$I_{cp} = 0,5*1 + 0,25*01 + 0,125*3 + 0,125*3 = 1,75.$$

Поскольку при кодировании четырех символов кодом постоянной длины требуется два двоичных разряда, сэкономлено 0,25 двоичного разряда в среднем на один символ.

Метод Хаффмена. Этот метод имеет два преимущества по сравнению с методом Шеннона-Фано: он устраняет неоднозначность кодирования, возникающую из-за примерного равенства сумм частот при разделении списка на две части (линия деления проводится неоднозначно), и имеет, в общем случае, большую эффективность кода. Исходное множество символов упорядочивается по не возрастанию частоты и выполняются следующие шаги:

1) объединение частот:

две последние частоты списка складываются, а соответствующие символы исключаются из списка;

оставшийся после исключения символов список пополняется суммой частот и вновь упорядочивается;

предыдущие шаги повторяются до тех пор, пока ни получится единица в результате суммирования и список ни уменьшится до одного символа;

2) построение кодового дерева:

строится двоичное кодовое дерево: корнем его является вершина, полученная в результате объединения частот, равная 1; листьями – исходные вершины; остальные вершины соответствуют либо суммарным, либо исходным частотам, причем для каждой вершины левая подчиненная вершина соответствует большему слагаемому, а правая – меньшему; ребра дерева связывают вершины-суммы с вершинами-слагаемыми. Структура дерева показывает, как происходило объединение частот;

ребра дерева кодируются: каждое левое кодируется единицей, каждое правое – нулём;

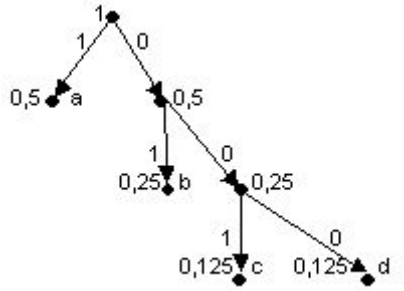
3) формирование кода: для получения кодов листьев (исходных кодируемых символов) продвигаются от корня к нужной вершине и «собирают» веса проходимых рёбер.

Пример 1. Даны символы *a, b, c, d* с частотами $f_a = 0{,}5$; $f_b = 0{,}25$; $f_c = 0{,}125$; $f_d = 0{,}125$. Построить эффективный код методом Хаффмена.

1) объединение частот (результат объединения двух последних частот в списке выделен в правом соседнем столбце заливкой):

Исходные символы	Частоты f_s	Этапы объединения		
		первый	второй	третий
a	0,5	0,5	0,5	1
b	0,25	0,25	0,5	
c	0,125	0,25		
d	0,125			

2) построение кодового дерева:



3) формирование кода:

a - 1; *b* - 01; *c* - 001;*d* -000.

Как видно, полученные коды совпадают с теми, что были сформированы методом Шеннона-Фано, следовательно, они имеют одинаковую эффективность.

Повышение эффективности кодирования

Повысить эффективность кодирования можно, строя код не для символа, а для блоков из n символов, причем частота блока рассчитывается как произведение частот символов, входящих в блок. Рассмотрим этот тезис на примере.

Пример 1. Даны символы a и b с частотами, соответственно, 0,9 и 0,1. Построить эффективный код методом Шеннона-Фано для блоков из двух символов ($n = 2$).

Сформируем список возможных блоков и их частот. При этом частоту блока будем рассчитывать как произведение частот символов, входящих в блок. Тогда имеем:

Блоки исходных символов	***Частоты блоков***
aa	0,81
ab	0,09
ba	0,09
bb	0,01

Построение кода сведём в таблицу:

Блоки исходных символов	Частоты блоков	Этапы построения кода		
		первый	второй	третий
aa	0,81	1	код построен	
ab	0,09	0	1	код построен
ba	0,09	0	0	1
bb	0,01	0	0	0

Таким образом, получены коды:

aa – 1; *ab* - 01; *ba* - 001; *bb* - 000.

Определим эффективность построенного кода. Для этого рассчитаем сначала показатель эффективности для блока символов: $I_{ср}^{блока} = 0{,}81 \cdot 1 + 0{,}09 \cdot 2 + 0{,}09 \cdot 3 + 0{,}01 \cdot 3 = 1{,}28$. Поскольку в блоке 2 символа (n=2), для одного символа $I_{ср} = I_{ср}^{блока}/2 = 1{,}28/2 = 0{,}64$. При посимвольном кодировании для эффективного кода потребуется по одному двоичному разряду. В самом деле, применение метода Шеннона-Фано даёт результат, представленный в таблице:

Исходные символы	Частоты символов	Построение кода
a	0,9	1
b	0,1	0

Таким образом, при блочном кодировании выигрыш составил 1 - 0,64 = 0,36 двоичных разрядов на один кодируемый символ в среднем.

Эффективность блочного кодирования тем выше, чем больше символов включается в блок

Декодирование эффективных кодов

Особенностью эффективных кодов является переменное число двоичных разрядов в получаемых кодовых комбинациях. Это затрудняет процесс декодирования.

Рассмотрим вначале, как происходит декодирование сообщения, если использовались коды постоянной длины.

Пусть кодовая таблица имеет вид:

Исходные символы	Двоичные коды
a	00
b	01
c	10
d	11

а закодированное сообщение - 001000011101.

Поскольку длина кода равна двум символам, в этом сообщении слева направо выделяются по два двоичных символа и сопоставляются с кодовой таблицей.

Тогда имеем:

00 10 00 10 11 01

a c a b d b

Таким образом, в исходном сообщении содержится текст *acabdb*. Декодирование выполнено. Для декодирования кодов переменной длины рассмотренный подход не годится. Но закодированные сообщения могут декодироваться благодаря свойству **префиксности** эффективных кодов: ни одна более короткая кодовая комбинация не является началом более длинной кодовой комбинации. Для раскрытия данного тезиса воспользуемся построенными ранее эффективными кодами:

a -1; *b* - 01; *c* - 001; *d* - 000.

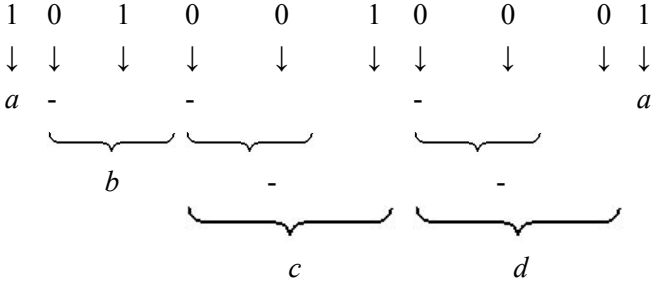
Здесь самым коротким кодом является код для символа *a* со значением 1. Как видно, ни один другой код (более длинный) не имеет в начале символ 1. Второй по длине код для символа *b* имеет значение 01 и, как показывает анализ, не является началом ни для кода 001, ни для кода 000. Таким образом, данный код является префиксным. Свойство префиксности позволяет декодировать сообщения, закодированные эффективными кодами. Пусть получено сообщение 1010010001, составленное из кодов

a - 1; *b* - 01; *c* - 001; *d* - 000.

Выполним его декодирование.

В сообщении слева направо выделяется по одному двоичному символу и делается попытка декодирования в соответствии с заданной таблицей кодов. Если попытка успешна, двоичный символ (или символы) исключается из исходной цепочки и заменяется соответствующим исходным символом. Если попытка не удается, во входной цепочке выделяется следующий двоичный символ и уже с двумя двоичными символами делается попытка их декодирования по таблице кодов. Если попытка и тогда неудачна, выделяют следующий третий и т.д.

Итак, имеем (направление просмотра цепочки слева направо):



Здесь знак «-» означает, что попытка декодирования не удалась. Таким образом, при декодировании получили строку *abcda.*

Отметим, что методы Шеннона-Фано и Хаффмена строят префиксные коды.

Помимо рассмотренных универсальных методов эффективного кодирования на практике часто применяются методы, ориентированные на конкретные виды сообщений. В зависимости от типа исходного сообщения они делятся на методы эффективного кодирования (сжатия) числовых последовательностей, словарей, естественно-языковых текстов.

Помехозащитное кодирование. Этот вид кодирования применяется для обнаружения и/или исправления ошибок, которые могут возникнуть в дискретном сигнале при его передаче по каналам связи. В качестве базового кода, который подвергается помехозащитному кодированию, используется двоичный код постоянной длины. Такой исходный (базовый) код называется **первичным**, поскольку подвергается модификации. Для понимания сущности вопроса рассмотрим, как происходит искажение кодовых комбинаций при наличии помех в каналах связи.

Задание 1. Рассмотри таблицу:

Буква	Код
А	0
М	1

Напиши в «Блокноте» слово, используя код: МАМА

Ответ к заданию 1. 1010

Задание 2. Мартышка написала записку. Что здесь написано? Запиши.

4	2	3	0	9	4	9	1	3	5	9	6	3	1	8	6	4	10	7	0

С	4+2-1	
И	10-1-1	
Р	2+2-2	
Н	5+6-10	
Ы	3+3+1	

У	9+1-6	
Л	4+7-1	
К	4+3-1	
	1+5+3	
А	8-3-2	
!	10-7-3	

Ответы к заданию 2. Ура! У нас каникулы!

Рассмотри таблицу:

А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Э	Ю	Я
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

Задание 3. Составь слова по номерам букв в алфавите и запиши в "Блокнот"

14	29	26	12	1

14	16	15	10	20	16	18

Ответы к заданию 3. Мышка Монитор

Задание 4. Продолжи:

К	Л	А	В	И	А	Т	У	Р	А
12									

П	А	М	Я	Т	Ь
17					

10	15								33

		Д
12		

Ответы к заданию 4. Клавиатура - 12 13 1 3 10 1 20 21 18 1. Память - 17 1 14 33 20 28. Информация - 10 15 22 16 18 14 1 24 10 33. Код - 12 16 5

4. КОНКРЕТНЫЕ МЕТОДЫ КОДИРОВАНИЯ

4.1 Алфавитное неравномерное двоичное кодирование

Данный случай относится к варианту (2) **Табл**. 1. При этом как следует из названия, символы некоторого первичного алфавита (например, русского) кодируются комбинациями символов двоичного алфавита (т.е. 0 и 1), причем, длина кодов и, соответственно, длительность передачи отдельного кода, могут различаться. Длительности элементарных сигналов при этом одинаковы ($\tau_0 = \tau_1 = \tau$). За счет чего можно оптимизировать кодирование в этом случае? Очевидно, суммарная длительность сообщения будет меньше, если применить следующий подход: тем буквам первичного алфавита, которые встречаются чаще, присвоить более короткие по длительности коды, а тем, относительная частота которых меньше – коды более длинные. Но длительность кода – величина дискретная, она кратна длительности сигнала τ передающего один символ двоичного алфавита. Следовательно, коды букв, вероятность появления которых в сообщении выше, следует строить из возможно меньшего числа элементарных сигналов. Построим кодовую таблицу для букв русского алфавита, основываясь на приведенных ранее вероятностях появления отдельных букв.

Очевидно, возможны различные варианты двоичного кодирования, однако, не все они будут пригодны для практического использования – важно, чтобы закодированное сообщение могло быть однозначно декодировано, т.е. чтобы в последовательности 0 и 1, которая представляет собой многобуквенное кодированное сообщение, всегда можно было бы различить обозначения отдельных букв. Проще всего этого достичь, если коды будут разграничены разделителем – некоторой постоянной комбинацией двоичных знаков. Условимся, что разделителем отдельных кодов букв будет последовательность 00 (признак конца знака), а разделителем слов – 000 (признак конца слова – пробел). Довольно очевидными оказываются следующие правила построения кодов:

- код признака конца знака может быть включен в код буквы, поскольку не существует отдельно (т.е. кода всех букв будут заканчиваться 00);
- коды букв не должны содержать двух и более нулей подряд в середине (иначе они будут восприниматься как конец знака);
- код буквы (кроме пробела) всегда должен начинаться с 1;
- разделителю слов (000) всегда предшествует признак конца знака; при этом реализуется последовательность 00000 (т.е. если в конце кода встречается комбинация …000 или …0000, они не воспринимаются как разделитель слов); следовательно, коды букв могут оканчиваться на 0 или 00 (до признака конца знака).

Длительность передачи каждого отдельного кода t_i, очевидно, может быть найдена следующим образом: $t_i = k_i \bullet \tau$, где k_i – количество элементарных сигналов (бит) в коде символа i. В соответствии с приведенными выше правилами получаем следующую таблицу кодов:

Табл. 1. Таблица кодов

Буква	Код	$p_i \cdot 103$	k_i	Буква	Код	$p_i \cdot 10^3$	k_i
пробел	000	174	3	я	1011000	18	7
о	100	90	3	ы	1011100	16	7
е	1000	72	4	з	1101000	16	7
а	1100	62	4	ь,ъ	1101100	14	7
и	10000	62	5	б	1110000	14	7
т	10100	53	5	г	1110100	13	7
н	11000	53	5	ч	1111000	12	7
с	11100	45	5	й	1111100	10	7
р	101000	40	6	х	10101000	9	8
в	101100	38	6	ж	10101100	7	8
л	110000	35	6	ю	10110000	6	8
к	110100	28	6	ш	10110100	6	8
м	111000	26	6	ц	10111000	4	8
д	111100	25	6	щ	10111100	3	8
п	1010000	23	7	э	11010000	3	8
у	1010100	21	7	ф	11010100	2	8

Теперь по формуле можно найти среднюю длину кода $K^{(2)}$ для данного способа кодирования:

$$K^{(2)} = \sum_{i=1}^{32} p_i \cdot k_i = 4{,}964$$

Поскольку для русского языка, $I_1(r)$=4,356 бит, избыточность данного кода, согласно (2), составляет:

$$Q(r) = 1 - 4{,}356/4{,}964 \approx 0{,}122;$$

это означает, что при данном способе кодирования будет передаваться приблизительно на 12% больше информации, чем содержит исходное сообщение. Аналогичные вычисления для английского языка дают значение $K(2) = 4{,}716$, что при $I_1(e) = 4{,}036$ бит приводят к избыточности кода $Q(e) = 0{,}144$.

Рассмотрев один из вариантов двоичного неравномерного кодирования, попробуем найти ответы на следующие вопросы: возможно ли такое кодирование без использования разделителя знаков? Существует ли наиболее оптимальный способ неравномерного двоичного кодирования?

Суть первой проблемы состоит в нахождении такого варианта кодирования сообщения, при котором последующее выделение из него каждого отдельного знака (т.е. декодирование) оказывается однозначным без специальных указателей разделения знаков. Наиболее простыми и употребимыми кодами такого типа являются так называемые префиксные коды, которые удовлетворяют следующему условию (условию Фано): Неравномерный код может быть однозначно декодирован, если никакой из кодов не совпадает с началом (префиксом) какого-либо иного более длинного кода. Например, если имеется код 110, то уже не могут использоваться коды 1, 11, 1101, 110101 и пр. Если условие Фано выполняется, то при прочтении (расшифровке) закодированного сообщения путем сопоставления со списком кодов всегда можно точно указать, где заканчивается один код и начинается другой.

Пример 1. Пусть имеется следующая таблица префиксных кодов:

Табл. 2. Таблица кодов

а	л	м	р	у	ы
10	010	00	11	0110	0111

Требуется декодировать сообщение: 00100010000111010101110000110. Декодирование производится циклическим повторением следующих действий. Отрезать от текущего сообщения крайний левый символ, присоединить к рабочему кодовому слову. Сравнить рабочее кодовое слово с кодовой таблицей; если совпадения нет, перейти к (1). Декодировать рабочее кодовое слово, очистить его. Проверить, имеются ли еще знаки в сообщении; если "да", перейти к (1).

Применение данного алгоритма даёт:

Шаг	Рабочее слово	Текущее сообщение	Распознанный знак	Декодированное сообщение
0	пусто	0010001000011101010111000010	–	–
1	0	0100010000111010101110000110	нет	–
2	00	1000100001110101011100001 10	м	м
3	1	0001000011101010111000 0110	нет	м
4	10	0010000111010101110000110	а	ма
5	0	0100001110101011100001 10	нет	ма
6	00	1000011101010111000011 0	м	мам
...				

Доведя процедуру до конца, получим сообщение: «мама мыла раму».

Таким образом, использование префиксного кодирования позволяет делать сообщение более коротким, поскольку нет необходимости передавать разделители знаков. Однако условие Фано не устанавливает способа формирования префиксного кода и, в частности, наилучшего из возможных.

Способ оптимального префиксного двоичного кодирования был предложен Д.Хаффманом. Построение кодов Хаффмана мы рассмотрим на следующем примере: пусть имеется первичный алфавит А, состоящий из шести знаков $a_1, ..., a_6$ с вероятностями появления в сообщении, соответственно, 0,3; 0,2; 0,2; 0,15; 0,1; 0,05. Создадим новый вспомогательный алфавит A_1, объединив два знака с наименьшими вероятностями (a_5 и a_6) и заменив их одним знаком (например, a_1); вероятность нового знака будет равна сумме вероятностей тех, что в него вошли, т.е. 0,15; остальные знаки исходного алфавита включим в новый без изменений; общее число знаков в новом алфавите, очевидно, будет на 1 меньше, чем в исходном. Аналогичным образом продолжим создавать новые алфавиты, пока в последнем не останется два знака; ясно, что число таких шагов будет равно $N - 2$, где N – число знаков исходного алфавита (в нашем случае $N = 6$, следовательно, необходимо построить 4 вспомогательных алфавита). В промежуточных алфавитах каждый раз будем переупорядочивать знаки по убыванию вероятностей. Всю процедуру построения представим в виде таблицы:

№ знака	Вероятности				
	Исходный алфавит	Промежуточные алфавиты			
		$A^{(1)}$	$A^{(2)}$	$A^{(3)}$	$A^{(4)}$
1	0,3	0,3	0,3	0,4	0,6
2	0,2	0,2	0,3	0,3	0,4
3	0,2	0,2	0,2	0,3	
4	0,15	0,15	0,2		
5	0,1	0,15			
6	0,05				

Теперь в обратном направлении поведем процедуру кодирования. Двум знакам последнего алфавита присвоим коды 0 и 1 (которому какой – роли не играет; условимся, что верхний знак будет иметь код 0, а нижний – 1). В нашем примере знак $a_1(4)$ алфавита $A(4)$, имеющий вероятность 0,6 , получит код 0, а $a_2(4)$ с вероятностью 0,4 – код 1. В алфавите $A(3)$ знак $a_1(3)$ с вероятностью 0,4 сохранит свой код (1); коды знаков $a_2(3)$ и $a_3(3)$, объединенных знаком $a_1(4)$ с вероятностью 0,6 , будут уже двузначным: их первой цифрой станет код связанного с ними знака (т.е. 0), а вторая цифра – как условились – у верхнего 0, у нижнего – 1; таким образом, $a_2(3)$ будет иметь код 00, а $a_3(3)$ – код 01. Полностью процедура кодирования представлена в следующей таблице:

№ знака	Вероятности									
	Исходный алфавит		Промежуточные алфавиты							
			$A^{(1)}$		$A^{(2)}$		$A^{(3)}$		$A^{(4)}$	
1	0,3	00	0,3	00	0,3	00	0,4	1	0,6	0
2	0,2	10	0,2	10	0,3	01	0,3	00	0,4	1
3	0,2	11	0,2	11	0,2	10	0,3	01		
4	0,15	010	0,15	010	0,2	11				
5	0,1	0110	0,15	011						
6	0,05	0111								

Из самой процедуры построения кодов легко видеть, что они удовлетворяют условию Фано и, следовательно, не требуют разделителя. Средняя длина кода при этом оказывается:

$$K^{(2)} = 0,3 \cdot 2 + 0,2 \cdot 2 + 0,2 \cdot 2 + 0,15 \cdot 3 + 0,1 \cdot 4 + 0,05 \cdot 4 = 2,45.$$

Для сравнения можно найти $I_1^{(A)}$ – она оказывается равной 2,409, что соответствует избыточности кода Q = 0,0169, т.е. менее 2%.

Код Хаффмана важен в теоретическом отношении, поскольку можно доказать, что он является самым экономичным из всех возможных, т.е. ни для какого метода алфавитного кодирования длина кода не может оказаться меньше, чем код Хаффмана.

Применение описанного метода для букв русского алфавита дает следующие коды:

Табл. 3. Таблица кодов русских букв

Буква	Код	$p_i \cdot 10^3$	k_i	Буква	Код	$p_i \cdot 10^3$	k_i
пробел	000	174	3	я	0011001	18	6
о	111	90	3	ы	0101100	16	6
е	0100	72	4	з	010111	16	6
а	0110	62	4	ь,ъ	100001	14	6
и	0111	62	4	б	101100	14	6
т	1001	53	4	г	101101	13	6
н	1010	53	5	ч	110011	12	6
с	1101	45	4	й	0011001	10	7
р	00101	40	5	х	1000000	9	7
в	00111	38	5	ж	1000001	7	7
л	01010	35	5	ю	1100101	6	7
к	10001	28	5	ш	00110000	6	8
м	10111	26	5	ц	11001000	4	8
д	11000	25	5	щ	11001001	3	8
п	001000	23	6	э	001100010	3	9
у	001001	21	6	ф	001100011	2	9

Средняя длина кода оказывается равной $K(2)$ = 4,395; избыточность кода $Q(r)$ = 0,00887, т.е. менее 1%.

Таким образом, можно заключить, что существует метод построения оптимального неравномерного алфавитного кода. Не следует думать, что он представляет число теоретический интерес. Метод Хаффмана и его модификация – метод адаптивного кодирования (динамическое кодирование Хаффмана) – нашли широчайшее применение в программах-архиваторах, программах резервного копирования файлов и дисков, в системах сжатия информации в модемах и факсах.

Кодирование энтропии - кодирования словами (кодами) переменной длины, при которой длина кода символа имеет обратную зависимость от вероятности появления символа в передаваемом сообщении. Обычно энтропийные кодировщики используют для сжатия данных коды, длины которых пропорциональны отрицательному логарифму вероятности символа. Таким образом, наиболее вероятные символы используют наиболее короткие коды.

4.2 Равномерное алфавитное двоичное кодирование

В этом случае двоичный код первичного алфавита строится цепочками равной длины, т.е. со всеми знаками связано одинаковое количество информации равное I_0. Передавать признак конца знака не требуется, поэтому для определения длины кодовой цепочки можно воспользоваться формулой: $K^{(2)} \geq \log_2 N$. Приемное устройство просто отсчитывает оговоренное заранее количество элементарных сигналов и интерпретирует цепочку (устанавливает, какому знаку она соответствует). Правда, при этом недопустимы сбои, например, пропуск (непрочтение) одного элементарного сигнала приведет к сдвигу всей кодовой последовательности и неправильной ее интерпретации; решается проблема путем синхронизации передачи или иными способами. С другой стороны, применение равномерного кода оказывается одним из средств контроля правильности передачи, поскольку факт поступления лишнего элементарного сигнала или, наоборот, поступление неполного кода сразу интерпретируется как ошибка.

Примером равномерного алфавитного кодирования является телеграфный код Бодо, пришедший на смену азбуке Морзе. Исходный алфавит должен содержать не более 32-х символов; тогда $K^{(2)} = \log_2 32 = 5$, т.е. каждый знак содержит 5 бит информации. Условие $N \leq 32$, очевидно, выполняется для языков,

основанных на латинском алфавите (N = 27 = 26+”пробел”), однако в русском алфавите 34 буквы (с пробелом) – именно по этой причине пришлось "сжать" алфавит (как в коде Хаффмана) и объединить в один знак "е" и "ё", а также "ь" и "ъ". После такого сжатия N = 32, однако, не остается свободных кодов для знаков препинания, поэтому в телеграммах они отсутствуют или заменяются буквенными аббревиатурами; это не является заметным ограничением, поскольку, как указывалось выше, избыточность языка позволяет легко восстановить информационное содержание сообщения. Избыточность кода Бодо для русского языка $Q^{(r)}$ = 0,129, для английского $Q^{(e)}$ = 0,193.

Другим важным для нас примером использования равномерного алфавитного кодирования является представление символьной информации в компьютере. Чтобы определить длину кода, необходимо начать с установления количество знаков в первичном алфавите. Компьютерный алфавит должен включать:

26×2=52 букв латинского алфавита (с учетом прописных и строчных);

33×2=66 букв русского алфавита;

цифры 0...9 – всего 10;

знаки математических операций, знаки препинания, спецсимволы ≈20.

Получаем, что общее число символов N=148. Теперь можно оценить длину кодовой цепочки: $K^{(2)} \geq \log_2 148 \geq 7{,}21$. Поскольку $K^{(2)}$ должно быть целым, очевидно, $K^{(2)}$= 8. Именно такой способ кодирования принят в компьютерных системах: любому символу ставится в соответствие цепочка из 8 двоичных разрядов (8 бит). Такая цепочка получила название *байт*, а представление таким образом символов – *байтовым кодированием*.

Байт наряду с битом может использоваться как единица измерения количества информации в сообщении. Один байт соответствует количеству информации в одном символе алфавита при их равновероятном распределении. Этот способ измерения количества информации называется также объёмным. Пусть имеется некоторое сообщение (последовательность знаков); оценка количества содержащейся в нём информации согласно рассмотренному ранее вероятностному подходу (с помощью формулы Шеннона) даёт $I_{вер}$, а объемная мера пусть равна $I_{об}$; соотношение между этими величинами: $I_{вер} \leq I_{об}$

Именно байт принят в качестве единицы измерения количества информации в международной системе единиц СИ. 1 байт = 8 бит. Использование 8-битных цепочек позволяет закодировать 2^8=256 символов, что превышает оцененное выше N и, следовательно, дает возможность употребить оставшуюся часть кодовой таблицы для представления дополнительных символов.

Однако недостаточно только условиться об определенной длине кода. Ясно, что способов кодирования, т.е. вариантов сопоставления знакам первичного алфавита восьмибитных цепочек, очень много. По этой причине для совместимости технических устройств и обеспечения возможности обмена информацией между многими потребителями требуется согласование кодов. Подобное согласование осуществляется в форме стандартизации кодовых таблиц. Первым таким международным стандартом, который применялся на и телекоммуникационных системах применяется международный байтовый код больших вычислительных машинах, был *EBCDIC> (Extended Binary Coded Decimal Interchange Code*) – «***расширенная двоичная** кодировка десятичного кода обмена»*. В персональных компьютерах *ASCII (American Standard Code for Information Interchange* – «***американский стандартный код обмена информацией***»). Он регламентирует коды первой половины кодовой таблицы (номера кодов от 0 до 127, т.е. первый бит всех кодов 0). В эту часть попадают коды прописных и строчных английских букв, цифры, знаки препинания и математических операций, а также некоторые управляющие коды (номера от 0 до 31). Ниже приведены некоторые *ASCII*-коды:

Табл. 4. Некоторые *ASCII*-коды

Знак, клавиша	Код двоичный	Код десятичный
пробел	00100000	32
A (лат)	01000001	65
B (лат)	01000010	66
Z	01011010	90
0	00110000	48
1	00110001	49
9	00111001	57

Клавиша *ESC*	00011011	27
Клавиша *Enter*	00001101	13

Вторая часть кодовой таблицы – она считается расширением основной – охватывает коды в интервале от 128 до 255 (первый бит всех кодов 1). Она используется для представления символов национальных алфавитов (например, русского или греческого), а также символов псевдографики. Для этой части также имеются стандарты, например, для символов русского языка это *КОИ–8, КОИ–7* и др.

Как в основной таблице, так и в её расширении коды букв и цифр соответствуют их лексикографическому порядку (т.е. порядку следования в алфавите) – это обеспечивает возможность автоматизации обработки текстов и ускоряет ее.

В настоящее время появился и находит все более широкое применение еще один международный стандарт кодировки – *Unicode*. Его особенность в том, что в нем использовано 16-битное кодирование, т.е. для представления каждого символа отводится 2 байта. Такая длина кода обеспечивает включения в первичный алфавит 65536 знаков. Это, в свою очередь, позволяет создать и использовать единую для всех распространенных алфавитов кодовую таблицу.

4.3 Азбука Морзе

В качестве примера использования кодирования с неравной длительностью элементарных сигналов рассмотрим телеграфный код Морзе («азбука Морзе»). В нём каждой букве или цифре сопоставляется некоторая последовательность кратковременных импульсов – точек и тире, разделяемых паузами. Длительности импульсов и пауз различны: если продолжительность импульса, соответствующего точке, обозначить ▪, то длительность импульса тире составляет 3▪, длительность паузы между точкой и тире ▪, пауза между буквами слова 3▪, пауза между словами (пробел) – 6▪. Таким образом, под знаками кода Морзе следует понимать: «▪» – «короткий импульс + короткая пауза», «-» – «длинный импульс + короткая пауза», «0» – «длинная пауза», т.е. код оказывается троичным.

Свой код Морзе разработал в 1838 г., т.е. задолго до исследований относительной частоты появления различных букв в текстах. Однако им был правильно выбран принцип кодирования – буквы, которые встречаются чаще, должны иметь более короткие коды, чтобы сократить общее время передачи. Относительные частоты букв английского алфавита он оценил простым подсчетом литер в ячейках типографской наборной машины. Поэтому самая распространенная английская буква «E» получила код «точка». При составлении кодов Морзе для букв русского алфавита учёт относительной частоты букв не производился, что, естественно, повысило его избыточность. Как и в рассмотренных ранее вариантах кодирования, произведем оценку избыточности. По-прежнему для удобства сопоставления данные представим в приведенном ниже формате. Признак конца буквы («0») в их кодах не отображается, но учтён в величине k_i – длине кода буквы i.

Среднее значение длины кода $K^{(3)} = 3{,}361$. Полагая появление знаков вторичного алфавита равновероятным, получаем среднюю информацию на знак равной $I^{(2)} = \log_2 3 = 1{,}585$ бит. Так как для русского алфавита $I_1^{(1)} = 4{,}356$ бит, то:

$$Q^{(r)} = 1 - 4{,}356/(3{,}361 - 1{,}585) \approx 0{,}182$$

т.е. избыточность составляет около 18% (для английского языка ≈15%). Тем не менее, код Морзе имел в недалеком прошлом весьма широкое распространение в ситуациях, когда источником и приемником сигналов являлся человек (не техническое устройство) и на первый план выдвигалась не экономичность кода, а удобство его восприятия человеком.

Буква	Код	$p_i \cdot 10^3$	k_i	Буква	Код	$p_i \cdot 10^3$	k_i
пробел	00	174	2	я	.–.–	18	5
о	–––	90	4	ы	–.––	16	5
е	.	72	2	з	––..	16	4
а	.–	62	3	ь,ъ	–..–	14	5
и	..	62	3	б	–...	14	5
т	–	53	2	г	––.	13	4
н	–.	53	3	ч	–––.	12	5
с	...	45	4	й	.–––	10	5
р	.–.	40	4	х	....	9	5
в	.––	38	4	ж	...–	7	5
л	.–..	35	5	ю	..––	6	5
к	–.–	28	4	ш	––––	6	5
м	––	26	2	ц	–.–.	4	5
д	–..	25	4	щ	––.–	3	5
п	.––	23	4	э	..–..	3	6
у	..–	21	4	ф	..–.	2	5

4.4 Блочное двоичное кодирование

Вернёмся к проблеме оптимального кодирования. Пока что наилучший результат (наименьшая избыточность) был получен при кодировании по методу Хаффмана – для русского алфавита избыточность оказалась менее 1%. При этом указывалось, что код Хаффмана улучшить невозможно. На первый взгляд это противоречит первой теореме Шеннона, утверждающей, что всегда можно предложить способ кодирования, при котором избыточность будет сколь угодно малой величиной. На самом деле это противоречие возникло из-за того, что до сих пор мы ограничивали себя алфавитным кодированием. При алфавитном кодировании передаваемое сообщение представляет собой последовательность кодов отдельных знаков первичного алфавита. Однако возможны варианты кодирования, при которых кодовый знак относится сразу к нескольким буквам первичного алфавита (будем называть такую комбинацию блоком) или даже к целому слову первичного языка. Кодирование блоков понижает избыточность. В этом легко убедиться на простом примере.

Пусть имеется словарь некоторого языка, содержащий n = 16000 слов (это, безусловно, более чем солидный словарный запас!). Поставим в соответствие каждому слову равномерный двоичный код. Очевидно, длина кода может быть найдена из соотношения $K^{(2)} \geq \log_2 n \geq 13{,}97 = 14$. Следовательно, каждому слову будет поставлена в соответствие комбинация из 14 нулей и единиц – получатся своего рода двоичные иероглифы. Например, пусть слову "ИНФОРМАТИКА" соответствует код 10101011100110, слову "НАУКА" – 00000000000001, а слову "ИНТЕРЕСНАЯ" – 00100000000010; тогда последовательность:
00000000000011010101110011000000000000001,
очевидно, будет означать *"ИНФОРМАТИКА ИНТЕРЕСНАЯ НАУКА"*.

Легко оценить, что при средней длине русского слова $K^{(r)}$ = 6,3 буквы (5,3 буквы + пробел между словами) средняя информация на знак первичного алфавита оказывается равной $I^{(2)} = K^{(2)}/K^{(r)}$ = 14/6,3 = 2,222 бит, что почти в 2 раза меньше, чем 4,395 бит при алфавитном кодировании. Для английского языка такой метод кодирования дает 2,545 бит на знак. Таким образом, кодирование слов оказывается более выгодным, чем алфавитное.

Ещё более эффективным окажется кодирование в том случае, если сначала установить относительную частоту появления различных слов в текстах и затем использовать код Хаффмана. По относительным частотам 8727 наиболее употребительных в английском языке слов Шеннон, что средняя информация на знак первичного алфавита оказывается равной 2,15 бит. Вместо слов можно кодировать сочетания букв – блоки. В принципе блоки можно считать словами равной длины, не имеющими, однако, смыслового содержания. Удлиняя блоки и применяя код Хаффмана можно добиться того, что средняя информация на знак кода будет сколь угодно приближаться к I_∞. Однако, применение блочного и словесного метода кодирования имеет свои недостатки. 1) Необходимо хранить огромную кодовую таблицу и постоянно к ней обращаться при кодировании и декодировании, что замедлит работу и потребует значительных ресурсов памяти. 2) Помимо основных слов разговорный язык содержит много производных

от них, например, падежи существительных в русском языке или глагольные формы в английском; в данном способе кодирования им всем нужно присвоить свои коды, что приведет к увеличению кодовой таблицы еще в несколько раз. 3) Возникает проблема согласования (стандартизации) этих громадных таблиц, что непросто. 4) Алфавитное кодирование имеет то преимущество, что буквами можно закодировать любое слово, а при кодировании слов – использовать только имеющийся словарный запас. По указанным причинам блочное и словесное кодирование представляет лишь теоретический интерес, на практике же применяется кодирование алфавитное.

5. ОБНАРУЖЕНИЕ И ИСПРАВЛЕНИЕ ОШИБОК В ТЕХНИКЕ СВЯЗИ

Обнаружение ошибок в технике связи - действие, направленное на контроль целостности данных при записи/воспроизведении информации или при её передаче по линиям связи. Исправление ошибок (коррекция ошибок) - процедура восстановления информации после чтения её из устройства хранения или канала связи. Для обнаружения ошибок используют коды обнаружения ошибок, для исправления - корректирующие коды (коды, исправляющие ошибки, коды с коррекцией ошибок, помехоустойчивые коды). В процессе хранения данных и передачи информации по сетям связи неизбежно возникают ошибки. Контроль целостности данных и исправление ошибок - важные задачи на многих уровнях работы с информацией.

В системах связи возможны несколько стратегий борьбы с ошибками:

обнаружение ошибок в блоках данных и *автоматический запрос повторной передачи* повреждённых блоков - этот подход применяется в основном на канальном и транспортном уровнях;

обнаружение ошибок в блоках данных и отбрасывание повреждённых блоков - такой подход иногда применяется в системах потокового мультимедиа, где важна задержка передачи и нет времени на повторную передачу;

исправление ошибок применяется на физическом уровне.

Корректирующие коды - коды, служащие для обнаружения или исправления ошибок, возникающих при передаче информации под влиянием помех, а также при её хранении.

Для этого при записи (передаче) в полезные данные добавляют специальным образом структурированную *избыточную* информацию (контрольное число), а при чтении (приёме) её используют для обнаружения или исправления ошибки. Число ошибок, которое можно исправить, ограничено и зависит от конкретного применяемого кода. С *кодами, исправляющими ошибки*, тесно связаны *коды обнаружения ошибок*. В отличие от первых, последние могут только установить факт наличия ошибки в переданных данных, но не исправить её. Любой код, исправляющий ошибки, может быть также использован для обнаружения ошибок (при этом он будет способен обнаружить большее число ошибок, чем был способен исправить).

По способу работы с данными коды, исправляющие ошибки делятся на *блоковые*, делящие информацию на фрагменты постоянной длины и обрабатывающие каждый из них в отдельности, и *свёрточные*, работающие с данными как с непрерывным потоком.

Пусть кодируемая информация делится на фрагменты длиной k бит, которые преобразуются в *кодовые слова* длиной n бит. Тогда соответствующий блоковый код обычно обозначают (n, k). При этом число $R=k/n$ называется *скоростью кода*.

Если исходные k бит код оставляет неизменными, и добавляет $n - k$ *проверочных*, такой код называется *систематическим*, иначе *несистематическим*. Задать блоковый код можно по-разному, в том числе таблицей, где каждой совокупности из k информационных бит сопоставляется n бит кодового слова. Однако, хороший код должен удовлетворять, как минимум, следующим критериям: 1) способность исправлять как можно большее число ошибок, 2) как можно меньшая избыточность, 3) простота кодирования и декодирования.

Нетрудно видеть, что приведённые требования противоречат друг другу. Именно поэтому существует большое количество кодов, каждый из которых пригоден для своего круга задач.

Практически все используемые коды являются линейными. Это связано с тем, что нелинейные коды значительно сложнее исследовать, и для них трудно обеспечить приемлемую лёгкость кодирования и декодирования.

Линейный блоковый код - такой код, что множество его *кодовых слов* образует k-мерное линейное подпространство в n-мерном линейном пространстве, изоморфное пространству k-битных векторов. Это значит, что операция кодирования соответствует умножению исходного k-битного вектора на невырожденную матрицу, называемую *порождающей матрицей*.

Коды Хемминга - простейшие линейные коды с минимальным расстоянием 3, то есть способные исправить одну ошибку.

Несмотря на то, что декодирование линейных кодов уже значительно проще декодирования большинства нелинейных, для большинства кодов этот процесс всё ещё достаточно сложен. Циклические коды, кроме более простого декодирования, обладают и другими важными свойствами. Циклический код обычно является двоичным. Коды *CRC (cyclic redundancy check* - циклическая избыточная проверка) являются систематическими кодами, предназначенными не для исправления ошибок, а для их обнаружения. Коды Рида-Соломона - недвоичные циклические коды, позволяющие исправлять ошибки в блоках данных. Элементами кодового вектора являются не биты, а группы битов (блоки). Очень распространены коды Рида-Соломона, работающие с байтами (октетами). Математически коды Рида - Соломона являются кодами БЧХ. Хотя блоковые коды, как правило, хорошо справляются с редкими, но большими *пачками ошибок*, их эффективность при частых, но небольших ошибках менее высока.

Свёрточные коды, в отличие от блоковых, не делят информацию на фрагменты и работают с ней как со сплошным потоком данных. Свёрточные коды порождаются дискретной линейной инвариантной во времени системой. Поэтому, в отличие от большинства блоковых кодов, свёрточное кодирование - очень простая операция, чего нельзя сказать о декодировании. Свёрточные коды эффективно работают в канале с белым шумом, но плохо справляются с пакетами ошибок.

Преимущества разных способов кодирования можно объединить, применив *каскадное кодирование*. При этом информация сначала кодируется одним кодом, а затем другим, в результате получается код-произведение. Некоторые коды-произведения специально сконструированы для итеративного декодирования, при котором декодирование осуществляется в несколько проходов, каждый из которых использует информацию от предыдущего. Это позволяет добиться большой эффективности, однако, декодирование требует больших ресурсов. Эффективность кодов определяется количеством ошибок, которые тот может исправить, количеством избыточной информации, добавление которой требуется, а также сложностью реализации кодирования и декодирования.

При передаче информации по каналу связи вероятность ошибки зависит от отношения сигнал/шум на входе демодулятора, поэтому при постоянном уровне шума решающее значение имеет мощность передатчика. В системах спутниковой или мобильной связи остро стоит вопрос экономии энергии, а в телефонной связи неограниченно повышать мощность сигнала не дают технические ограничения. Поскольку помехоустойчивое кодирование позволяет исправлять ошибки, при его применении мощность передатчика можно снизить, оставляя скорость передачи информации неизменной. Энергетический выигрыш определяется как разница отношений сигнал/шум при наличии и отсутствии кодирования.

Коды, исправляющие ошибки, применяются: 1) в системах цифровой связи, в том числе: спутниковой, радиорелейной, сотовой, передаче данных по телефонным каналам. 2) в системах хранения информации, в том числе магнитных и оптических.

Коды, обнаруживающие ошибки, применяются в сетевых протоколах различных уровней.