**Основные принципы блочного кодирования**

**Избыточность и её роль**

**Избыточность** — это добавление дополнительных символов (бит) к исходным данным, что на первый взгляд может показаться "бесполезным". Ведь, с точки зрения сжатия информации, мы хотим минимизировать количество передаваемых данных. Однако в условиях передачи данных по шумному каналу с возможными ошибками эта избыточность становится важнейшим элементом для обеспечения надёжности.

**Зачем добавляются избыточные символы?**

* **Обнаружение ошибок**: Избыточные биты позволяют приемнику проверять, была ли ошибка при передаче данных. С помощью специальных правил декодирования можно определить, было ли полученное кодовое слово искажено.
* **Исправление ошибок**: В некоторых схемах, например, в коде Хэмминга, избыточность не только позволяет обнаружить ошибку, но и её исправить. Благодаря этой возможности системы становятся более надёжными, так как ошибки, вызванные шумом в канале связи, могут быть автоматически исправлены.

Таким образом, добавление избыточных символов увеличивает количество передаваемых данных, но при этом позволяет увеличить надёжность передачи, что критично для цифровых коммуникаций.

Когда мы передаем данные через канал связи, такие как интернет или беспроводную связь, неизбежно возникает проблема ошибок. Эти ошибки могут возникать в виде искажённых битов, что приводит к изменению смысла передаваемых данных. Кодирование с исправлением ошибок (Error-Correcting Codes, ECC) использует избыточность, чтобы противодействовать этим ошибкам.

**Механизм исправления ошибок:**

* Избыточные биты добавляются таким образом, чтобы при получении искажённых данных было возможно восстановить исходное сообщение.
* На стороне приёмника, используя правила декодирования, можно вычислить, какие биты были искажены, и, следовательно, восстановить исходное сообщение без необходимости повторной передачи.

**Пример: Код Хэмминга (7,4) (см. прошлые лекции)**

Рассмотрим процесс кодирования и исправления ошибок на примере кода Хэмминга с параметрами (7,4), который кодирует 4 информационных бита в 7-битовое кодовое слово, добавляя 3 контрольных (избыточных) бита.

**1. Пример кодирования:** Пусть у нас есть исходное сообщение X=1011 (4 информационных бита).

* Шаг 1: Назначим биты d1,d2,d3,d4 ​ как информационные биты сообщения.
* Шаг 2: Рассчитаем 3 контрольных бита p1,p2,p3​, основываясь на чётности комбинаций информационных бит:
  + p1​ проверяет биты d1,d2,d4​
  + p2​ проверяет биты d1,d3,d4​
  + p3​ проверяет биты d2,d3,d4​
* Пусть информационные биты X=1011. Вычисляем контрольные биты:
  + p1=d1⊕d2⊕d4=1⊕0⊕1=0
  + p2=d1⊕d3⊕d4=1⊕1⊕1=1
  + p3=d2⊕d3⊕d4=0

Теперь кодовое слово: p1,p2,d1,p3,d2,d3,d4 ​, т.е. 0110101.

**2. Пример обнаружения и исправления ошибок:** Предположим, что при передаче данных произошла ошибка, и вместо кодового слова 0110101 было получено 0010101, где ошибочно передан второй бит (вместо 1 передана 0).

* Шаг 1: Приёмник проверяет контрольные биты с теми же комбинациями:
  + Проверяем p1=d1⊕d2⊕d4=1⊕0⊕1=0 (ошибки нет).
  + Проверяем p2=d1⊕d3⊕d4=1 (ошибки нет).
  + Проверяем p3=d2⊕d3⊕d4=0 (ошибки нет).
* Шаг 2: На основе этих проверок можно обнаружить, что ошибка произошла во втором бите, и восстановить его до правильного значения.

Таким образом, код Хэмминга позволяет не только обнаружить одиночную ошибку, но и исправить её. Для студентов этот пример будет наглядным подтверждением того, как добавление всего лишь нескольких контрольных битов может значительно повысить надёжность передачи данных.

**Основы помехоустойчивого кодирования**

Задача кодера источника – *представить подлежащие передаче данные в максимально компактной и, по возможности, неискаженной форме*.

При передаче информации по каналу связи с помехами в принятых данных могут возникать ошибки. Если такие ошибки имеют небольшую величину или возникают достаточно редко, информация может быть использована потребителем. При большом числе ошибок полученной информацией пользоваться нельзя.

Для уменьшения количества ошибок, возникающих при передаче информации по каналу с помехами, может быть использовано *кодирование в канале*, или *помехоустойчивое кодирование*.

Возможность использования кодирования для уменьшения числа ошибок в канале была теоретически показана К. Шенноном в 1948 году в его работе "Математическая теория связи". В ней было сделано утверждение, что *если скорость создания источником сообщений (производительность источника) не превосходит некоторой величины, называемой пропускной способностью канала, то при соответствующем кодировании и декодировании можно свести вероятность ошибок в канале к нулю*.

Вскоре, однако, стало ясно, что фактические ограничения на скорость передачи устанавливаются не пропускной способностью канала, а сложностью схем кодирования и декодирования. Поэтому усилия разработчиков и исследователей в последние десятилетия были направлены на поиски эффективных кодов, создание практически реализуемых схем кодирования и декодирования, которые по своим характеристикам приближались бы к предсказанным теоретически.

Кодирование с исправлением ошибок представляет собой *метод обработки сообщений, предназначенный для повышения надежности передачи по цифровым каналам.* Хотя различные схемы кодирования очень непохожи друг на друга и основаны на различных математических теориях, всем им присущи два общих свойства.

**Первое** − *использование избыточности*. Закодированные последо-вательности всегда содержат дополнительные, или избыточные, символы. *Количество символов в кодовой последовательности* ***Y*** *всегда больше, чем необходимо для однозначного представления любого сообщения****λ*** ***i*** *из алфавита.*

**Второе** — *свойство усреднения*, означающее, что *избыточные символы зависят от нескольких информационных символов*, то есть информация, содержащаяся в кодовой последовательности ***X,*** перераспределяется также и на избыточные символы.

Существует два больших класса корректирующих кодов − *блочные и сверточные.* Определяющее различие между этими кодами состоит в отсутствии или наличии памяти кодера.

**Введение в блочные коды**

- **Определение блочного кода**: Каждый блок исходных данных длиной *k* символов кодируется в кодовое слово длиной *n*, где добавляются избыточные символы для исправления ошибок. Это называется (*n*,*k*) -кодом.

- **Пример**: Если *k*=4, это означает, что мы кодируем 4 бита информации в 7-битовое кодовое слово.

**Кодер для блочных кодов** делит непрерывную информационную последовательность ***X*** на блоки-сообщения длиной ***k*** символов.

*Кодер канала* преобразует блоки-сообщения ***X*** в более длинные двоичные последовательности ***Y***, состоящие из ***n*** символов и называемые *кодовыми словами.* Символы (***n-k***), добавляемые к каждому блоку-сообщению кодером, называются *избыточными*. Они не несут никакой дополнительной информации, и *их функция состоит в обеспечении возможности обнаруживать (или исправлять) ошибки*, возникающие в процессе передачи.

***k*** -разрядным двоичным словом можно представить ***2k*** возможных значений из алфавита источника, им соответствует ***2k*** кодовых слов на выходе кодера.

*Такое множество****2k****кодовых слов называется****блочным кодом****.*

Термин " *без памяти* " означает, что *каждый блок из****n****символов зависит только от соответствующего информационного блока из****k****символов и не зависит от других блоков.*

**Кодер для свёрточных кодов** работает с информационной последовательностью без разбиения ее на независимые блоки. В каждый момент времени кодер из небольшого текущего блока информационных символов размером в ***b*** символов (блока-сообщения) образует блок, состоящий из ***v*** кодовых символов (кодовый блок), причем ***v > b.*** При этом кодовый ***v-*** символьный блок зависит не только от ***b-*** символьного блока- сообщения, присутствующего на входе кодера в настоящий момент, но и от предшествующих ***m*** блоков-сообщений. В этом, собственно, и состоит наличие памяти в кодере.

Блочное кодирование удобно использовать в тех случаях, когда исходные данные по своей природе уже сгруппированы в какие-либо блоки или массивы.

При передаче по радиоканалам чаще используется сверточное кодирование, которое лучше приспособлено к побитовой передаче данных. Кроме этого, при одинаковой избыточности сверточные коды, как правило, обладают лучшей исправляющей способностью.

**Классификация блочных кодов**

1. **Линейные блочные коды**

Линейные блочные коды — это класс блочных кодов, которые обладают линейными свойствами, упрощающими их анализ и использование в системах передачи данных. Они базируются на понятиях линейной алгебры, что позволяет применять мощные методы для кодирования и декодирования.

**Определение линейного кода**

**Линейным кодом** называется блочный код, если любое линейное сочетание его кодовых слов также является кодовым словом. Более формально, если *C* — это линейный код длины *n* над полем *GF*(2) (поле из двух элементов 0 и 1), то для любых двух кодовых слов ***x***,***y***∈*C* и любых коэффициентов *a*,*b*∈*GF*(2), линейная комбинация *ax*+*by* также принадлежит *C*. Это означает, что множество кодовых слов образует векторное пространство над полем GF(2).

**Линейность кодов: Пример**

Рассмотрим (7,4)-код Хэмминга. В коде Хэмминга 4 информационных бита преобразуются в 7-битовое кодовое слово с добавлением трёх контрольных битов. Это линейный код, потому что, если сложить (по модулю 2) любые два кодовых слова, результатом будет новое кодовое слово.

**Пример кодовых слов**:

* Кодовые слова x1=0110101 и x2=1101100 являются кодовыми словами кода Хэмминга.
* Линейное сложение этих двух кодовых слов по модулю 2: x1⊕x2=(0110101)⊕(1101100)= 1011001

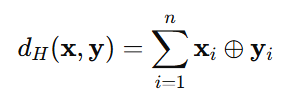
Результат также является кодовым словом, что подтверждает линейность кода.

**2. Коды с минимальным расстоянием Хэмминга**

Минимальное расстояние Хэмминга — это один из ключевых параметров любого корректирующего кода. Оно определяет, насколько надёжным будет код при обнаружении и исправлении ошибок.

**Определение расстояния Хэмминга**

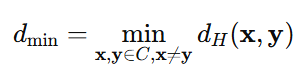
**Расстояние Хэмминга** между двумя кодовыми словами — это количество позиций, в которых они различаются. Если два кодовых слова *x* и *y* имеют одинаковую длину, то расстояние Хэмминга между ними, обозначаемое как *dH*(*x*,*y*), вычисляется следующим образом:



где ⊕ — это сложение по модулю 2 (логическое XOR), а *n* — длина кодовых слов.

**Минимальное расстояние кода**

Минимальное расстояние Хэмминга кода *C* — это минимальное расстояние Хэмминга между любыми двумя различными кодовыми словами в коде *C*. Обозначается как *dmin*​:



Минимальное расстояние кода имеет важное значение, потому что:

* **Обнаружение ошибок**: Код может обнаружить до *dmin*−1 ошибок в кодовом слове.
* **Исправление ошибок**: Код может исправить до  ошибок.

В коде Хэмминга (7,4) минимальное расстояние Хэмминга равно 3. Это означает, что код может:

* **Обнаружить** до 2 ошибок в любом кодовом слове.
* **Исправить** 1 ошибку.

**Геометрическая интерпретация расстояния Хэмминга**

Расстояние Хэмминга можно интерпретировать геометрически: каждый код может быть представлен как множество кодовых слов в пространстве векторов, где кодовые слова — это вершины гиперкуба. Минимальное расстояние Хэмминга определяет, насколько далеко друг от друга находятся эти вершины. Чем больше минимальное расстояние, тем лучше код защищён от ошибок.

**Линейные блочные коды**

Для блочного кода с ***2k*** кодовыми словами длиной в ***n*** символов, если он только не обладает специальной структурой, аппарат кодирования и декодирования является очень сложным. Поэтому ограничим свое рассмотрение лишь кодами, которые могут быть реализованы на практике.

Одним из условий реализуемости блочных кодов при больших ***k*** является условие их линейности.

*Что такое линейный код?*

Блочный код длиной ***n*** символов, состоящий из ***2k*** кодовых слов, называется линейным (***n, k***)-кодом при условии, что все его ***2k*** кодовых слов образуют ***k*** -мерное подпространство векторного пространства ***n-*** последовательностей двоичного поля ***GF(2).***

*Если сказать проще, то двоичный код является линейным, если сумма по модулю 2 (****mod2****) двух кодовых слов также является кодовым словом этого кода.*

*Работая с двоичными кодами, мы постоянно будем сталкиваться с элементами двоичной арифметики, поэтому определим основные понятия.*

*Полем называется множество математических объектов, которые можно складывать, вычитать, умножать и делить.*

*Возьмем простейшее поле, состоящее из двух элементов − нуля - 0 и единицы - 1. Определим для него операции сложения и умножения:*

*0+0=0, 0× 0=0;  
0+1=1, 0× 1=0;  
1+0=1, 1× 0=0;  
1+1=0, 1× 1=1.*

*Определенные таким образом операции сложения и умножения называются сложением по модулю 2 (****mod2****) и умножением по модулю 2.*

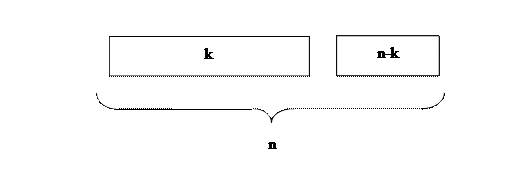
*Отметим, что из равенства****1+1 = 0****следует, что****-1 = 1****и, соответственно,****1+1=1-1****, а из равенства****1×1=1****− что****1:1=1****.*

*Алфавит из двух символов 0 и 1 вместе со сложением и умножением по****mod2****называется полем из двух элементов и обозначается как****GF(2).****К полю****GF(2)****применимы все методы линейной алгебры, в том числе матричные операции.*

*Еще раз обратим внимание на то, что все действия над символами в двоичных кодах выполняются по модулю 2.*

*Желательным качеством линейных блочных кодов является систематичность.*

*Систематический код имеет формат, изображенный на рис. то есть содержит неизменную информационную часть длиной****k****символов и избыточную (проверочную) длиной****n – k****символов.*



*Блочный код, обладающий свойствами линейности и систематичности, называется линейным блочным систематическим (****n, k****)-кодом.*

**Матрицы для кодирования и декодирования**

**Матрица генератора**

Для линейного блочного кода процесс кодирования можно эффективно описать с помощью **матрицы генератора** *G*. Эта матрица преобразует информационные биты в кодовые слова с добавлением избыточности, которая необходима для обнаружения и исправления ошибок. Основная задача матрицы генератора — линейное преобразование входных данных (информационных бит) в кодовые слова.

**Формальное определение**

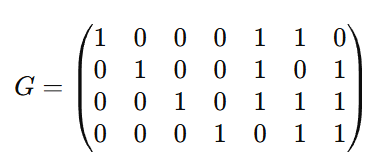
Пусть исходное сообщение — это вектор *x* длины *k* (где *k* — количество информационных бит). Чтобы закодировать это сообщение, умножаем его на матрицу генератора *G*, которая имеет размер *k*×*n* (где *n* — длина кодового слова):

*y*=*xG*

где *y* — это кодовое слово длины *n*. Каждая строка матрицы генератора представляет собой одно кодовое слово, соответствующее конкретной комбинации информационных бит.

**Для кода Хэмминга (7,4)**, у нас 4 информационных бита и 7-битовое кодовое слово, из которых 3 бита — избыточные.

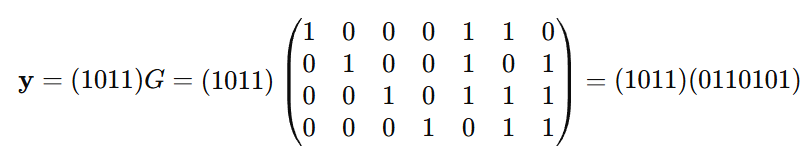
Матрица генератора для кода Хэмминга (7,4) выглядит следующим образом:



- Первые 4 столбца образуют единичную матрицу, которая отвечает за исходные информационные биты.

- Последние 3 столбца представляют контрольные биты, которые вычисляются для проверки и исправления ошибок.

Теперь, чтобы закодировать сообщение, например, x=(1011), умножаем этот вектор на матрицу генератора:



Итак, кодовое слово для сообщения 1011 — это 0110101.

**Проверочная матрица**

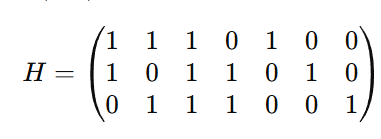
Для того чтобы обнаруживать ошибки на стороне приёмника, используется **проверочная матрица** *H*, которая помогает определить, где именно произошла ошибка, если она возникла. Проверочная матрица позволяет вычислить **синдром ошибки**.

**Определение проверочной матрицы**

Проверочная матрица *H* для линейного кода строится таким образом, чтобы выполнялось условие:

HyT=0

Матрица *H* для кода Хэмминга (7,4) выглядит следующим образом:



Эта матрица имеет размер (*n*−*k*)×*n*, то есть 3×7, и используется для обнаружения ошибок.

**Обнаружение ошибок с помощью проверочной матрицы**

Когда кодовое слово *y* получено приёмником, он умножает его на транспонированную проверочную матрицу *HT* и получает **синдром ошибки** s:

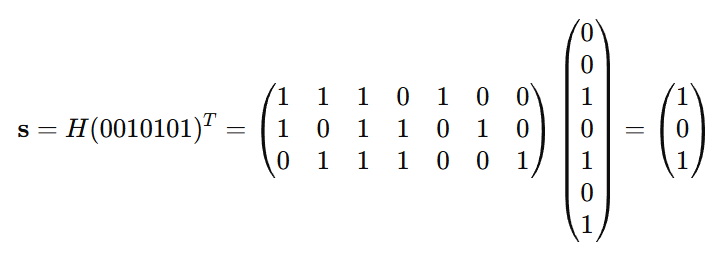
*s*=*HyT*

* Если синдром s=0, это означает, что ошибок нет, и кодовое слово было получено правильно.
* Если синдром не равен нулю (s≠0), это указывает на наличие ошибки, а синдром помогает определить её позицию.

**Пример обнаружения ошибки**

Предположим, что при передаче кодового слова 0110101 произошла ошибка, и приёмник получил 0010101 (второй бит был изменён).

Для обнаружения ошибки умножим полученное кодовое слово на проверочную матрицу:



Синдром (101) указывает на позицию второй ошибки в кодовом слове. Мы можем исправить эту ошибку, изменив второй бит обратно на 1, получив правильное кодовое слово 0110101.

**Применение блочных кодов в реальной жизни**

Блочные коды активно используются в различных областях цифровой связи и хранения данных для обеспечения надёжности информации при её передаче или хранении. Они помогают исправлять ошибки, возникающие из-за помех, неисправностей оборудования или воздействия внешних факторов. Рассмотрим несколько реальных применений блочных кодов, где они играют ключевую роль.

**1. Дисковые массивы RAID**

**RAID (Redundant Array of Independent Disks)** — это технология, которая используется для повышения надёжности хранения данных и увеличения производительности дисковых систем. В основе технологии лежит принцип использования избыточности для восстановления данных в случае повреждения одного из дисков.

**Как блочные коды используются в RAID?**

* **RAID 5** и **RAID 6** используют избыточные данные (похожие на избыточные биты в блочных кодах), которые распределяются по всем дискам. Это позволяет восстановить данные, если один или два диска выходят из строя.
* Примером блочного кода может служить простая схема с добавлением контрольных битов, которые используются для восстановления утерянной информации с повреждённого диска.

**Пример:**

Представьте массив из 4 дисков, где данные разделяются на блоки. Для каждого блока вычисляется контрольная сумма (или контрольный бит), которая хранится на отдельном диске. Если один диск выходит из строя, эти контрольные суммы позволяют восстановить его содержимое.

**2. Wi-Fi (беспроводные сети)**

В беспроводных сетях передачи данных, таких как **Wi-Fi**, блочные коды играют важную роль в защите информации от ошибок, вызванных помехами в канале передачи (например, от стен, электромагнитных шумов и т. д.).

**Как блочные коды работают в Wi-Fi?**

Wi-Fi использует различные схемы кодирования для повышения надёжности передачи данных. Например, в стандарте **802.11** применяются коды исправления ошибок, такие как **код Рида-Соломона** или **сверточные коды**, чтобы минимизировать влияние ошибок, возникающих при передаче пакетов по беспроводным каналам.

* **Кодирование с исправлением ошибок** добавляет избыточные биты к каждому пакету данных, что позволяет приёмнику обнаружить и исправить ошибки.

**Пример:**

Когда вы передаёте файл по Wi-Fi, он делится на пакеты, и к каждому пакету добавляются контрольные биты. Если часть пакета теряется из-за помех, приёмник может исправить ошибки, используя блочные коды.

**3. Мобильные сети**

Мобильные сети, такие как **3G**, **4G**, и **5G**, используют блочные коды для обеспечения надёжной передачи данных в условиях изменяющихся каналов связи. Из-за особенностей мобильных устройств и перемещения пользователей (например, при движении в машине), передача данных может быть нестабильной.

**Как блочные коды используются в мобильных сетях?**

В мобильных сетях активно используются **коды (LDPC)** и **турбокоды**, которые являются мощными блочными кодами для коррекции ошибок. Эти коды обеспечивают исправление значительного количества ошибок при передаче данных, что особенно важно для потокового видео, звонков и интернета.

* Эти коды позволяют снизить количество повторных передач данных и увеличить надёжность связи.

**Пример:**

Когда вы смотрите видео на телефоне через мобильную сеть, блочные коды исправляют ошибки, вызванные колебаниями сигнала или потерей данных, чтобы воспроизведение было плавным.

**Применение блочных кодов в RAID**

1. Симулируйте работу системы RAID 5, используя блочный код с избыточностью.
2. Разделите данные на блоки, добавьте контрольные биты для защиты от выхода из строя одного из дисков.
3. Смоделируйте ситуацию, когда один из блоков повреждён, и восстановите его данные.