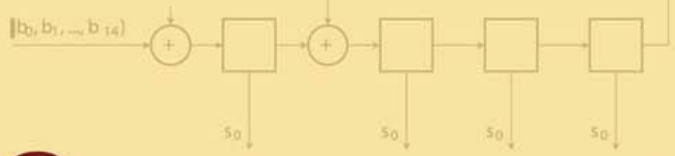
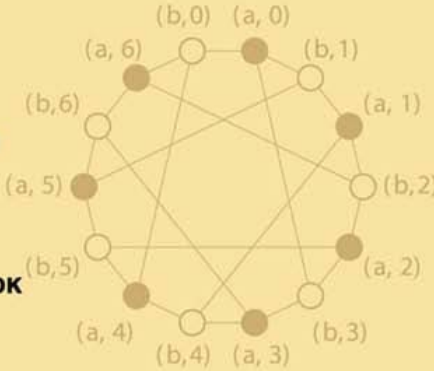
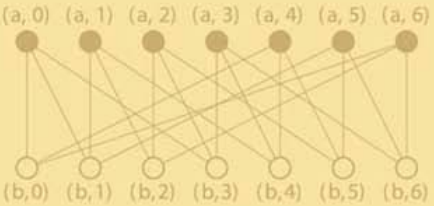
Б. Д. Кудряшов



Основы теории кодирования

- Потенциальная эффективность кодирования
- Алгебраические коды
- Коды над решетками. Сверточные коды
- Каскадные коды, кодированная модуляция, турбокоды
- Коды с малой плотностью проверок на четность





bhv

УДК 519.72
ББК 32.811.4
К88

Кудряшов Б. Д.

К88 Основы теории кодирования: учеб. пособие. — СПб.: БХВ-Петербург, 2016. — 400 с.: ил. — (Учебная литература для вузов)

ISBN 978-5-9775-3527-4

В учебное пособие, ориентированное на семестровый курс лекций, включены классические разделы теории кодирования: линейные коды, основы построения и декодирования алгебраических кодов. Рассказывается о представлении кодов решетками, о декодировании по максимуму правдоподобия. Приведены основы теории сверточных кодов, введение в каскадные коды, модуляционные коды и турбо-коды. Отдельная глава посвящена низкоплотностным кодам, находящим все более широкое применение в телекоммуникационных стандартах. Все необходимые математические сведения приведены в виде приложений к главам учебного пособия. В книге много численных примеров, детальных алгоритмов, примеров программ MATLAB.

Для студентов технических вузов, обучающихся по информационным специальностям, аспирантов и инженеров

УДК 519.72
ББК 32.811.4

Рецензенты:

Е .Т. Мирончиков — д-р техн. наук, проф. Петербургского государственного унивеситета путей сообщения;

Ф. И. Соловьева — д-р физ.-мат. наук, проф. Новосибирского государственного университета.

Подписано в печать 31.08.15.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 32,25.
Тираж 700 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28.

ISBN 978-5-9775-3527-4

Оглавление

2. Линейные коды

Мы знаем из предыдущей главы, что коды — это наборы кодовых слов. Чтобы быть эффективными, коды должны быть длинными, обладать большим минимальным расстоянием при заданной скорости, иметь разумную сложность кодирования и декодирования.

Даже для не очень длинных кодов списки кодовых слов слишком велики, чтобы можно было сохранять их в памяти кодера и декодера. Наш первый шаг к сокращению сложности описания кодов — построение линейных кодов, т.е. кодов, слова которых образуют линейное пространство. Для описания линейного кода достаточно задать базис пространства, а кодирование сводится к умножению на матрицу. К сожалению, в этом разделе мы не получим ответ на вопрос, как декодировать линейные коды с приемлемой для практики сложностью. Задача декодирования линейных кодов общего вида почти так же сложна, как и задача декодирования кода, заданного списком кодовых слов. Чтобы упростить декодирование, нам придется дополнительно сузить класс кодов. Собственно, этому вопросу будут посвящены все последующие разделы курса.

2.1. Арифметика пространства двоичных последовательностей

Над двоичными символами можно выполнять операции сложения и умножения. Мы рассматриваем числа 0 и 1 как элементы кольца вычетов по модулю 2 и как группу по умножению. Иными словами, множество чисел $\{0,1\}$ образует простейшее *поле* из двух элементов F_2. Ниже приведены таблицы сложения и умножения (табл.

2.1). Аналогично определяется арифметика поля вычетов по модулю произвольного простого q. Это поле мы обозначаем через F_q.

Таблица 2.1. Таблицы сложения и умножения в поле F_2

$+$	0	1
0	0	1
1	1	0

$\times$	0	1
0	0	0
1	0	1

Упражнение 2.1. Постройте таблицы сложения и умножения для $q = 3$.

От вычислений над скалярными величинами перейдем к изучению действий над последовательностями (векторами). Для произвольного множества A мы через A^n обозначаем множество последовательностей длины n, составленных из элементов множества A.

Определим сумму векторов $\boldsymbol{x} = (x_1, \ldots, x_n)$ и $\boldsymbol{y} = (y_1, \ldots, y_n)$ как покомпонентную сумму $\boldsymbol{x} + \boldsymbol{y} = (x_1 + y_1, \ldots, x_n + y_n)$. Умножение вектора $\boldsymbol{x}$ на скалярную величину $a \in F_2$ определим как $a\boldsymbol{x} = (ax_1, \ldots, ax_n)$.

Множество векторов, замкнутое относительно операций сложения и умножения на скаляр, называется *линейным векторным пространством.*

Пример 2.1. Примеры двоичных линейных пространств:

А) $\{00, 01, 10, 11\}$,

Б) $\{000, 001, 010, 011, 100, 101, 110, 111\}$,

В) $\{000, 011, 101, 110\}$,

Г) $\{000, 111\}$.

Множество F_2^n всех двоичных последовательностей длины n, очевидно, является линейным пространством. В частности, пространства примеров А и Б совпадают с F_2^2 и F_2^3. Такие пространства

содержат в общем случае 2^n векторов. Пространство примера В содержится в F_2^3, но не совпадает с ним, т.е. является его *линейным подпространством.*

Заметим, что каждое из трех пространств примера 2.1 содержит нулевой вектор. Этим свойством обладает любое линейное пространство (почему?).

Ключевым понятием в описании линейных пространств является понятие базиса.

Базисом линейного пространства называется наибольшее возможное множество линейно независимых векторов пространства. Число базисных векторов называется *размерностью пространства.*

Пример 2.2. Для линейных пространств примера 2.1 имеем:

А) Размерность равна 2, пример базиса: $\{01, 10\}$.

Б) Размерность равна 3, примеры базисов: $\{001, 010, 100\}, \{110, 011, 111\}$.

В) Размерность равна 2, пример базиса: $\{011, 101\}$.

Г) Размерность равна 1, базис: $\{111\}$.

Основные свойства базиса и порожденного им пространства:

- Любой вектор пространства единственным образом может быть представлен в виде линейной комбинации базисных векторов.
- Число элементов векторного пространства размерности k равно q^k.

Доказательство свойств оставляем читателю в качестве упражнения.

Определение 2.1. *Линейным q-ичным кодом (n, k)-кодом C называется любое k-мерное подпространство пространства F_q^n всевозможных векторов длины n.*

Скоростью линейного (n,k)*-кода* называется отношение $R = k/n$. В двоичном случае ($q = 2$) скорость измеряется в битах на символ канала.

Мы уже замечали, что хорошим будет код, слова которого «далеки» друг от друга в смысле расстояния Хэмминга, которое для последовательностей $\boldsymbol{x}$, $\boldsymbol{y}$ длины n определяется как

$$d(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{n} d(x_i, y_i),$$

где

$$d(x,y) = \begin{cases} 0, & x = y; \\ 1, & x \neq y. \end{cases}$$

Расстояние Хэмминга удовлетворяет аксиомам расстояния:

1. $d(\boldsymbol{x},\boldsymbol{y}) = 0$ если и только если $\boldsymbol{x} = \boldsymbol{y}$
2. $d(\boldsymbol{x},\boldsymbol{y}) = d(\boldsymbol{y},\boldsymbol{x})$ (симметричность)
3. $d(\boldsymbol{x},\boldsymbol{y}) \leq d(\boldsymbol{x},\boldsymbol{z}) + d(\boldsymbol{z},\boldsymbol{y})$ (неравенство треугольника)

Таким образом, имеем метрическое пространство.

Минимальным расстоянием кода C называется наименьшее из попарных расстояний между кодовыми словами:

$$d = \min_{\boldsymbol{x},\boldsymbol{y}\in C, \boldsymbol{x}\neq\boldsymbol{y}} d(\boldsymbol{x},\boldsymbol{y}). \tag{2.1}$$

Важность этой характеристики кода обусловлена следующим фактом.

Теорема 2.1. *Код с минимальным расстоянием* d *исправляет любые комбинации ошибок кратности* $t \leq \lfloor (d-1)/2 \rfloor$

Доказательство. Рассмотрим декодирование по минимуму расстояния Хэмминга. Если произошло t ошибок, то расстояние от переданного кодового слова до принятой последовательности равно t. Нужно доказать, что не найдется другого слова на расстоянии t или меньше. Для этого достаточно воспользоваться неравенством треугольника и определением минимального расстояния. □

Для линейного кода формула (2.1) может быть упрощена:

Теорема 2.2.

$$d = \min_{\boldsymbol{x}\in C, \boldsymbol{x}\neq 0} d(\boldsymbol{x}, \mathbf{0}) = \min_{\boldsymbol{x}\in C, \boldsymbol{x}\neq 0} w(\boldsymbol{x}), \tag{2.2}$$

где $w(\boldsymbol{x}) = d(\boldsymbol{x}, \mathbf{0})$ — число ненулевых элементов в последовательности $\boldsymbol{x}$ или вес вектора $\boldsymbol{x}$ в метрике Хэмминга.

Упражнение 2.2. Докажите теорему 2.2.

Упражнение 2.3. Найдите скорости и минимальные расстояния кодов примера 2.1. Убедитесь в том, что использование теоремы 2.2 упрощает поиск минимального расстояния. Во сколько раз?

После введения понятия базиса стало понятнее, как можно сэкономить память, предназначенную для хранения информации о коде: достаточно помнить не все пространство, а только его базис, а конкретные кодовые слова порождать в процессе кодирования. Это означает, что, например, для (1000,500)-кода вместо 2^{500} кодовых слов достаточно хранить 500 базисных слов длины 1000. Для этого достаточно 500 Кбит памяти. Таким объемом памяти никого сегодня не удивишь. На самом деле, и эта цифра завышена. Например, можно использовать циклические коды, в которых слова являются циклическим сдвигом друг друга. Тогда можно будет хранить всего одно слово, т.е. всего 1000 бит. И даже эта величина может быть уменьшена.

Декодирование линейного кода, как и кода общего вида, может быть выполнено путем сравнения каждого кодового слова с принятой из канала последовательностью и выбора ближайшего к ней в смысле некоторой метрики (например, метрики Хэмминга). На первый взгляд, сложность декодирования линейного кода намного меньше, чем кода общего вида. Ведь для перебора по множеству кодовых слов теперь нет необходимости хранить полный список кодовых слов. Однако если речь идет о переборе по 2^{500} кодовым последовательностям, то сложность обработки отдельной последовательности уже не имеет значения. Нужно думать о методах сокращения перебора. В следующих параграфах мы сделаем самые первые шаги в направлении упрощения декодирования двоичных кодов.

Еще один важный вопрос: не потеряем ли мы в потенциальной эффективности кодирования, ограничивая себя рассмотрением только линейных кодов? Можно доказать, что по крайней мере для симметричных каналов и при достаточно больших длинах кодов характеристики линейных кодов (в среднем по множеству кодов) такие же, как и характеристики кодов общего вида. Подробнее об этом можно почитать в учебниках по теории информации [6, 13].

2.2. Порождающая и проверочная матрицы

Определение 2.2. *Порождающей матрицей линейного (n, k)-кода называется матрица размера $k \times n$, строки которой — его базисные векторы.*

Например, матрицы

$$G = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \quad G = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$$

являются порождающими для кодов В) и Г) примера 2.1.

Мы знаем, что кодовые слова — линейные комбинации базисных векторов, т.е. строк матрицы G. Это означает, что слова могут быть получены умножением вектора на матрицу. Сообщение записывается в виде вектора $\boldsymbol{m} = (m_1, \ldots, m_k)$, и соответствующее сообщению кодовое слово вычисляется по формуле

$$\boldsymbol{c} = \boldsymbol{m}G.$$

Тем самым, вектор из k бит превращается в последовательность из n двоичных кодовых символов, передаваемых по каналу или записываемых в память запоминающего устройства.

Сделаем первые шаги к пониманию того, как можно упростить декодирование линейного кода.

Предположим, что для некоторого двоичного вектора $\boldsymbol{h} = (h_1, \ldots, h_n)$ все кодовые слова (n, k)-кода $C = \{\boldsymbol{c}_i\}, i = 0, \ldots, 2^k - 1$ удовлетворяют тождеству

$$(\boldsymbol{c}_i, \boldsymbol{h}) = 0, \quad i = 0, \ldots, 2^k - 1, \tag{2.3}$$

в котором запись $(\boldsymbol{x}, \boldsymbol{y})$ обозначает скалярное произведение векторов $\boldsymbol{x}$ и $\boldsymbol{y}$.

Про такой вектор $\boldsymbol{h}$ мы скажем, что он ортогонален коду, и назовем его *проверкой* по отношению к коду. Найдя такой вектор, мы могли бы проверять с помощью тождества (2.3), является ли принятая из канала последовательность кодовым словом.

Заметим теперь, что (2.3) справедливо для всех кодовых слов, если оно справедливо для базисных векторов, т.е. если

$$G\boldsymbol{h}^{\mathrm{T}} = 0, \tag{2.4}$$

где верхний индекс T обозначает транспонирование.

Чем больше таких «проверок» мы найдем, тем, по-видимому, больше ошибок сумеем обнаружить и исправить.

Упражнение 2.4. Докажите, что проверки образуют линейное пространство.

Это упражнение очень простое. Пространство проверок назовем пространством, ортогональным линейному коду, или *проверочным пространством*.

Упражнение 2.5. Докажите, что в порождающей матрице (n, k)-кода найдутся k линейно независимых столбцов.

Здесь нужно вспомнить, что такое ранг матрицы, и воспользоваться тем, что ранги матрицы, вычисленные по строкам и по столбцам, равны.

Упражнение 2.6. Найдите размерность линейного пространства проверок.

Решение. Чтобы справиться с этим непростым упражнением, воспользуемся утверждением упражнения 2.5. Найдем и зафиксируем список номеров k линейно независимых столбцов матрицы G и назовем этот набор индексов *информационной совокупностью*.

Совокупность индексов остальных позиций назовем *проверочной совокупностью*. Сами позиции и записанные на них символы будем называть соответственно информационными и проверочными. Чуть позже смысл этих названий станет яснее.

Выберем произвольно и зафиксируем значения вектора $\boldsymbol{h}$ на $r = n - k$ позициях проверочной совокупности. Какими должны быть значения на позициях информационной совокупности, чтобы выполнилось (2.4)?

Просуммируем в каждом уравнении слагаемые, соответствующие проверочной совокупности, обозначим сумму $\boldsymbol{h}^*$ и перенесем сумму в правую часть. Получится система линейных уравнений. Столбцы матрицы G, соответствующие информационной совокупности, образуют базис в пространстве столбцов, и, значит, вектор $-\boldsymbol{h}^*$ может быть единственным образом записан в виде их линейной комбинации.

Мы получили взаимно-однозначное соответствие между значениями компонент вектора $\boldsymbol{h}$ на информационных и проверочных позициях. Поскольку значения вектора $\boldsymbol{h}$ на проверочных позициях можно выбрать 2^{n-k} способами (q^{n-k} способами для q-ичного кода), в проверочном пространстве будет 2^{n-k} (соответственно, q^{n-k}) элементов, и размерность пространства будет равна $r = n - k$. □

Упражнение 2.7. На примере порождающей матрицы

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

убедитесь в правильности приведенных выше рассуждений.

Результат упражнения 2.6 сформулируем в виде теоремы.

Теорема 2.3. *Размерность проверочного пространства линейного (n, k)-кода равна $r = n - k$.*

Число r называется *избыточностью* кода.

Базис проверочного пространства запишем в виде матрицы

$$H = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{r1} & h_{r2} & \cdots & h_{rn} \end{pmatrix},$$

называемой *проверочной матрицей* кода.

Проверочная и порождающая матрицы связаны соотношением

$$GH^{\mathrm{T}} = 0. \tag{2.5}$$

Из этого соотношения мы видим, что для любого кодового слова $\boldsymbol{c} \in C$ имеет место

$$\boldsymbol{c}H^{\mathrm{T}} = 0. \tag{2.6}$$

Это тождество можно использовать как критерий принадлежности произвольной последовательности коду, т.е. для обнаружения ошибок.

Зная G, можно найти H. Для того чтобы понять, как это сделать, заметим, что один тот же код можно задать разными порождающими матрицами, выбирая по-разному базис пространства. Больше того, заменив в G любую строку на любую линейную комбинацию этой строки с другими строками, мы получаем новую порождающую матрицу того же кода.

Перестановка столбцов матрицы G, вообще говоря, приводит к другому коду, но этот другой код по своим характеристикам не отличается от исходного. Коды, различающиеся только нумерацией позиций, называются *эквивалентными*.

Понятно, что для каждого кода найдется такой эквивалентный ему код, что первые k позиций кодовых слов образуют информационную совокупность, иными словами, первые k столбцов порождающей матрицы образуют невырожденную подматрицу размера $k \times k$. Заменяя строки линейными комбинациями строк (метод Гаусса), полученную матрицу можно привести к виду

$$G = \begin{pmatrix} I_k & P \end{pmatrix}, \tag{2.7}$$

где I_k — единичная матрица порядка k, а P — некоторая матрица размера $k \times r$.

Матрица вида (2.7) называется порождающей матрицей, приведенной к *систематическому виду*, а соответствующий код называется *систематическим*. Кодирование для систематического кода проще, чем для кода общего вида:

$$\boldsymbol{c} = \boldsymbol{m}G = \begin{pmatrix} \boldsymbol{m} & \boldsymbol{m}P \end{pmatrix}, \tag{2.8}$$

т.е. в кодовом слове первые k позиций — просто копия информационной последовательности $\boldsymbol{m}$, а остальные $r = n - k$ (проверочных) позиций получаются умножением информационного вектора

на матрицу размера $k \times r$, что иногда существенно меньше, чем $n \times r$. Соответственно, и информация о систематическом коде занимает существенно меньше памяти, чем информация о линейном коде общего вида.

Замечание. Формально выражения «систематический код» и «несистематический код» неправильные, поскольку один и тот же код (совокупность кодовых слов) может быть описан как систематической, так и несистематической порождающей матрицей. В этом смысле любой код одновременно и систематический, и несистематический. Тем не менее, мы не будем отступать от привычной устоявшейся терминологии.

Для двоичного систематического кода с порождающей матрицей в форме (2.7) проверочная матрица может быть вычислена по формуле

$$H = \begin{pmatrix} P^{\mathrm{T}} & I_r \end{pmatrix}. \tag{2.9}$$

Упражнение 2.8. Проверьте (2.9). Подсказка: для этого нужно подставить (2.9) и (2.7) в (2.5).

Упражнение 2.9. Обобщите (2.9) на случай кода над произвольным полем F_q.

Как найти проверочную матрицу для несистематического кода?

Нужно привести матрицу к систематическому виду и воспользоваться (2.8). Если первые k столбцов порождающей матрицы образуют невырожденную подматрицу (первые k позиций образуют информационную совокупность), то для приведения к систематической форме достаточно таких операций как перестановка строк и замена строк линейными комбинациями строк. Если нет — нужно будет сначала найти информационную совокупность и перенумеровать позиции так, чтобы первые позиции стали информационными. После того, как найдена проверочная матрица эквивалентного систематического кода, нужно восстановить исходный порядок следования символов кодового слова.

Упражнение 2.10. Сформулируйте алгоритм нахождения порождающей матрицы по проверочной.

Упражнение 2.11. Объясните, почему любой набор из номеров k линейно-независимых столбцов называется информационной совокупностью.

Упражнение 2.12. Постройте порождающие и проверочные матрицы для кодов из примера 2.1.

Подведем итоги этого основополагающего параграфа.

- Линейный (n, k)-код может быть задан любой из двух матриц: порождающей G размера $k \times n$ либо проверочной H размера $r \times n$. По G легко найти H приведением кода к систематической форме. Аналогично по H находится G.
- Матрица G используется при кодировании (формула (2.8)), матрицей H можно воспользоваться при декодировании, по меньшей мере, для обнаружения ошибок. Выполнение тождества (2.6) свидетельствует о том, что данная последовательность принадлежит коду.

2.3. Вычисление минимального расстояния по проверочной матрице

Формула (2.2) нахождения минимального расстояния в терминах порождающей матрицы записывается как

$$d = \min_{\boldsymbol{m} \neq 0} w\left(\boldsymbol{m}G\right). \tag{2.10}$$

Вычисление по этой формуле требует перебора по $2^k - 1$ ненулевым словам кода. Возвращаясь к примеру (1000,500)-кода, легко понять, почему на практике мы часто пользуемся кодами, минимальное расстояние которых никому не известно.

Казалось бы, для кода (1000,900) проблема еще сложнее. Однако заметим, что при $r < k$, т.е. когда скорость кода больше $1/2$, проверочная матрица имеет меньший размер, чем порождающая. Хотелось бы воспользоваться этим обстоятельством для упрощения поиска минимального расстояния кода.

Согласно формуле (2.6), кодовому слову $\boldsymbol{c}$ веса $w(\boldsymbol{c})$ соответствует некоторый набор из $w(\boldsymbol{c})$ столбцов проверочной матрицы H, сумма которых — нулевой вектор. Это означает, что соответствующие столбцы линейно зависимы. Следовательно, для нахождения минимального расстояния кода нужно найти минимальный набор линейно зависимых столбцов проверочной матрицы. Приходим к следующей теореме.

Теорема 2.4. *Минимальное расстояние линейного (n, k)-кода равно d в том и только в том случае, когда любые $d-1$ столбцов проверочной матрицы линейно независимы и существует набор из d линейно зависимых столбцов.*

Подумав над тем, сколько всего может быть линейно независимых столбцов в матрице с r строками, получаем следующую интересную границу на минимальное расстояние.

Теорема 2.5. *Граница Синглтона: минимальное расстояние линейного (n, k)-кода удовлетворяет неравенству $d \leq r+1 = n-k+1$.*

Доказательство. Ранг проверочной матрицы равен r, следовательно $r+1$ столбцов всегда линейно зависимы. Граница Синглтона следует из теоремы 2.4.

Еще одно доказательство прямо следует из представления порождающей матрицы в систематической форме (2.7) и теоремы 2.2. Минимальный вес строк порождающей матрицей, очевидно, является оценкой сверху на минимальное расстояние кода. □

Упражнение 2.13. Докажите, что:

А) если в H нет нулевых столбцов, то минимальное расстояние кода не меньше 2;

Б) если в H все столбцы различны и ненулевые, то минимальное расстояние кода не меньше 3;

В) если одна из строк проверочной матрицы H двоичного кода не содержит нулей, то минимальное расстояние кода четно.

Упражнение 2.14. Найдите минимальные расстояния кодов примера 2.1.

Дуальным к данному коду называется код, порождающая матрица которого является проверочной матрицей данного кода.

Упражнение 2.15. Найдите минимальные расстояния кодов, дуальных к кодам примера 2.1.

2.4. Примеры кодов

Сформулированные в упражнениях свойства линейных кодов позволяют построить несколько хороших кодов.

Пример 2.3. Код с порождающей матрицей

$$G = (I_n)$$

представляет собой (n, n)-код со скоростью 1 и минимальным расстоянием $d = 1$. Поскольку число проверочных символов $r = n - k = 0$, проверочная матрица неопределена.

Пример 2.4. $(n, n-1)$-код с проверочной матрицей

$$H = \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}$$

(вектор из n единиц) имеет минимальное расстояние $d = 2$. Его порождающая матрица имеет вид

$$G = \begin{pmatrix} 1 & 0 & \cdots & 0 & 1 \\ 0 & 1 & \cdots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 \end{pmatrix}.$$

Это весьма экономный код. Он позволяет ценой одного избыточного символа обнаруживать любые однократные ошибки и любые другие комбинации ошибок нечетного веса. Его называют *кодом с проверкой на четность*.

Пример 2.5. Пусть

$$G = \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}.$$

Этот код — двойственный к коду предыдущего примера. Очевидно, в таком коде всего два слова, т.е. имеем $(n, 1)$-код. Его проверочная матрица

$$H = \begin{pmatrix} 1 & 0 & \cdots & 0 & 1 \\ 0 & 1 & \cdots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 \end{pmatrix}.$$

Расстояние кода $d = n$, он исправляет комбинации ошибок кратности до половины длины кодового слова.

Пример 2.6. (Коды Хэмминга). Построим код, исправляющий любые однократные ошибки. Для этого нужно составить проверочную матрицу из различных столбцов. При числе проверочных символов r длина кода равна $n = 2^r - 1$, а число информационных символов равно $k = 2^r - r - 1$. Такие $(2^r - 1, 2^r - r - 1)$-коды называются *кодами Хэмминга.* Таким образом, имеем бесконечную последовательность кодов с расстоянием 3. В качестве примера рассмотрим случай $r = 3$, т.е.

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

Для кода Хэмминга легко указать процедуру исправления ошибок. Предположим, что передаваемое сообщение имеет вид $\boldsymbol{m} = (1011)$. Ему соответствует кодовое слово $\boldsymbol{c} = \boldsymbol{m}G = (1011010)$.

Предположим, что при передаче этой последовательности произошла одна ошибка и принятая последовательность имеет вид

$$\boldsymbol{x} = \boldsymbol{c} + \boldsymbol{e} = (1011010) + (0100000) = (1111010),$$

где через $\boldsymbol{e}$ обозначен вектор ошибки.