Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра программного обеспечения информационных технологий

А. А. Иванюк, С. Б. Мусин

СПЕЦИАЛЬНЫЕ ГЛАВЫ ВЫСШЕЙ МАТЕМАТИКИ:
теория помехоустойчивого кодирования

Практикум
для студентов специальности I-40 01 01
«Программное обеспечение информационных технологий»
дневной и дистанционной форм обучения

Минск 2007

УДК 512.62(075.8)
ББК 22.132 я 73
И 23

Р е ц е н з е н т
доц. кафедры ЭВМ БГУИР,
канд. техн. наук М. М. Татур

Иванюк, А. А.

И 23 Специальные главы высшей математики: теория помехоустойчивого кодирования : практикум для студ. спец. I-40 01 01 «Программное обеспечение информационных технологий» днев. и дист. форм обуч. / А. А. Иванюк, С. Б. Мусин. – Минск : БГУИР, 2007. – 32 с. : ил.

ISBN 978-985-488-240-6

Практикум содержит базовые сведения из теории помехоустойчивого кодирования, изучаемой студентами специальности I-40 01 01 «Программное обеспечение информационных технологий» в рамках дисциплины «Специальные главы высшей математики». По каждой теме приводятся краткие теоретические сведения и варианты заданий для программной реализации рассмотренных алгоритмов.

УДК 512.62(075.8)
ББК 22.132 я 73

ISBN 978-985-488-240-6

СОДЕРЖАНИЕ

1. ДЕКОДИРОВАНИЕ ЛИНЕЙНЫХ ДВОИЧНЫХ КОДОВ ПО ЛИДЕРУ СМЕЖНОГО КЛАССА

1.1. Линейные двоичные коды

Коды позволяют обнаруживать и исправлять ошибки при передаче сообщений в каналах с шумом. Для этой цели выполняется *кодирование* сообщения: к блоку из k символов сообщения (информационных символов) по определенному правилу добавляется r избыточных (проверочных) символов. Совокупность информационных и проверочных символов образует *кодовое слово* длиной n. Кодовые слова, соответствующие всем возможным сообщениям, образуют *(n,k)-код*. При этом если сумма любых двух кодовых слов также является кодовым словом, то код является *линейным*. Код, сообщения которого состоят из символов 0 и 1, называется *двоичным кодом*. В общем случае код задается в векторном пространстве над полем $GF(q)$.

Получатель кодового слова выполняет операцию *декодирования* – преобразование кодового слова в сообщение, в результате переданная информация даже при наличии ошибок может быть восстановлена. Количество символов сообщения, измененных в результате возникновения ошибки, будем называть *кратностью ошибки*.

Предполагаем, что вероятность возникновения ошибки меньшей кратности выше вероятности возникновения ошибки большей кратности. Тогда возможность обнаружения и исправления ошибок в кодовом слове определяется величиной d^*, которая называется минимальным кодовым расстоянием (минимальное расстояние по Хэммингу). Эта величина определяется как минимальное число позиций, в которых два кодовых слова отличаются друг от друга. Для обнаружения t ошибок необходимо и достаточно, чтобы $d^* \geq t+1$, а для их исправления, чтобы $d^* \geq 2t+1$.

Мерой эффективности кода является величина $R = k/n$, которая называется *скоростью кода.* Скорость кода определяет эффективную информационную емкость кода. Доля избыточно передаваемых проверочных символов определяется как $1-R$.

Одним из простейших примеров линейных двоичных кодов с исправлением ошибок являются коды с повторением, где информационный символ передается n раз, $C = \{\underbrace{00\ldots0}_{n}, \underbrace{11\ldots1}_{n}\}$.

Для декодирования кодового слова подсчитывается число нулей и число единиц в полученной последовательности. Минимальное кодовое расстояние такого кода $d^* = n$, что позволяет обнаруживать вплоть до $n-1$ ошибок и корректировать $\frac{n-1}{2}$. Этот код имеет очень низкую скорость передачи информации, так как все позиции, кроме одной, являются проверочными.

Примером высокоскоростных кодов являются коды с одной проверкой на четность, содержащие только один проверочный символ. Проверочный символ

вычисляется таким образом, чтобы четность всех кодовых слов была одинаковой: $c_n = \sum_{n-1}^{i} c_i$. Например, $C = \{000, 011, 101, 110\}$. Если полученное слово содержит нечетное число единиц, то считается, что произошла ошибка. Коды с одной проверкой на четность позволяют обнаруживать любое нечетное число ошибок.

На основе кодов с проверкой на четность строятся итеративные коды. Так, в прямоугольных итеративных кодах сообщение располагается в виде прямоугольной матрицы и к каждой строке и столбцу матрицы добавляется проверка на четность. Любая одиночная ошибка может быть исправлена, так как при ее возникновении соответствующие строке и столбцу проверки на четность не выполняются. Минимальное расстояние такого кода $d^* = 4$, но его избыточность довольно высока. Можно построить итеративные коды с меньшей избыточностью – треугольные коды. В таких кодах проверки на четность, в которые входят как строка, так и соответствующий столбец, располагаются на диагонали.

Пример

Пусть передается сообщение 110101. Представим его в виде треугольника:

110

10

1

Теперь расположим проверки на четность на диагонали:

1100

101

10

1

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

Переписывая полученный треугольник, получим кодовое слово 1100101101.

Если при построении остаются пустые ячейки, то они игнорируются.

1.2. Метод декодирования по лидеру смежного класса

Универсальным методом декодирования линейных кодов является *декодирование по лидеру смежного класса*. Этот метод основан на определении наиболее вероятного вектора ошибок. Обычно более вероятно, что вектором ошибок будет вектор с малым числом единиц, чем вектор с большим числом единиц. Если принятое слово является кодовым, тогда вектор ошибок равен нулю. В противном случае разность принятого кодового слова и вектора ошибок определяет кодовое слово.

Пусть C – (n,k)-код и $\mathbf{v}$ – произвольный двоичный вектор разрядностью n. Тогда множество $\mathbf{v}+C$, определяемое как $\mathbf{v}+C = \{\mathbf{v}+\mathbf{c} \mid \mathbf{c} \in C\}$, называется *смежным классом* C.

Определим вес некоторого n-мерного двоичного вектора как число единиц в данном векторе. Вектор, имеющий минимальный вес в смежном классе, называется *лидером смежного класса*.

Возможными векторами ошибок являются слова из того смежного класса, которому принадлежит принятое слово. Наиболее вероятный вектор ошибок – это лидер смежного класса, содержащего принятое слово.

Для декодирования линейных кодов по лидеру смежного класса строится *таблица смежных классов* (стандартное расположение):

1) первая строка формируется как список всех кодовых слов с нулевым словом на месте крайнего левого элемента;

2) выбирается произвольный вектор $\mathbf{v}_1$, не находящийся в первой строке и имеющий минимальный вес. Вторая строка формируется как смежный класс $\mathbf{v}_1 + C$ с элементом $\mathbf{v}_1$ на месте крайнего левого элемента;

3) выбирается произвольный вектор $\mathbf{v}_2$, не находящийся в первых двух строках и имеющий минимальный вес. Над ним выполняются операции, как и над вектором $\mathbf{v}_1$ на шаге 2.

Данная процедура продолжается до тех пор, пока в таблице не будут содержаться все векторы разрядности n (в таблице все векторы должны быть различны).

Декодирование кодов по лидеру смежного класса осуществляется следующим образом: принятый вектор ищется в построенной матрице и суммируется поразрядно по модулю два с лидером соответствующего смежного класса (операция суммирования двоичных векторов совпадает с операцией получения разности). Результат интерпретируется как сообщение.

Пример

Пусть C – (4,2)-код с повторениями, тогда таблица смежных классов определяется следующим образом:

0000 + C = {0000, 0101, 1010, 1111},
1000 + C = {1000, 1101, 0010, 0111},
0100 + C = {0100, 0001, 1110, 1011},
1100 + C = {1100, 1001, 0110, 0011}.

Примечание.

Смежный класс 0010 + C = {0010, 0111, 1000, 1101} совпадает с классом 1000 + C. Действительно, для смежных классов справедливо свойство: если $\mathbf{v} + C$ – смежный класс C и $\mathbf{u} \in \mathbf{v} + C$, то $\mathbf{u} + C = \mathbf{v} + C$.

Предположим, принято сообщение, содержащее однократную ошибку 0**0**01. Данный вектор находится в третьей строке таблицы смежных классов. Лидером в этой строке является вектор 0100. Суммируя поразрядно по модулю два, получим кодовое слово 0101.

1.3. Задание

1. Напишите процедуру, которая по заданному сообщению длиной k получает кодовое слово, закодированное следующим (n,k)-кодом:

вариант 1 – код с повторениями;
вариант 2 – код с проверкой на четность;
вариант 3 – прямоугольный код;
вариант 4 – треугольный код.

2. Напишите процедуру, которая декодирует заданный вектор длиной n с использованием присущего заданному в п. 1 коду алгоритму декодирования.

3. Напишите процедуру декодирования, которая декодирует заданный вектор длиной n с использованием определенной таблицы смежных классов.

4. Напишите процедуру, которая генерирует все возможные ошибки для заданного кодового слова, запускает любую из реализованных процедур декодирования и определяет, какое число ошибок каждой кратности было обнаружено и исправлено. Проведите анализ полученных результатов.

5. Сравните результаты, выдаваемые реализованными процедурами декодирования.

6. Определите время выполнения и объем используемой памяти для каждой из реализованных процедур декодирования.

7. Сделайте выводы о достоинствах и недостатках исследованных кодов.

8. Сделайте выводы о достоинствах и недостатках метода декодирования линейных двоичных кодов по лидеру смежного класса.

2. СИНДРОМНОЕ ДЕКОДИРОВАНИЕ

2.1. Код Хэмминга и метод синдромного декодирования

На основе кода с одной проверкой на четность можно построить высокоскоростной код, позволяющий исправлять однократные ошибки, где проверки на четность располагаются оптимальным образом – *код Хэмминга*. Оптимальность достигается за счет того, что проверки на четность рассчитываются для каждой из n позиций кодового слова и являются независимыми, т.е. никакая сумма одних проверок не совпадает с другой.

Двоичное число, состоящее из 0 для каждой невыполненной проверки на четность и 1 для каждой выполненной проверки, называется *синдромом*. Основная идея кодов Хэмминга состоит в том, что синдром дает фактическое положение ошибки в одной из n позиций кодового слова, причем нулевой синдром означает, что ошибка отсутствует. Поэтому количество проверочных символов r выбирается как наименьшее целое положительное число, такое, что двоичное представление $n = k + r$ содержит r бит.

Пример

$k = 1,\ r = 2,\ k + r = 3 = 11_2;$

$k = 2,\ r = 3,\ k + r = 5 = 101_2;$

$k = 4,\ r = 3,\ k + r = 7 = 111_2.$

Из соотношения $k + r \le 2^r - 1$ можно определить зависимость количества информационных и проверочных символов: $k \le 2^r - r - 1$.

Каждый из r проверочных символов помещается в позицию кодового слова с номером степени двойки, а k информационных символов – в оставшиеся позиции. Первая проверка на четность рассчитывается на основании тех позиций кодового слова, последняя цифра в двоичном представлении номеров которых равна 1:

$1_{10} = 00\mathbf{1}_2$
$3_{10} = 01\mathbf{1}_2$
$5_{10} = 10\mathbf{1}_2$
$7_{10} = 11\mathbf{1}_2$
…

Вторая проверка содержит те позиции, для которых равна 1 вторая позиция кодового слова:

$2_{10} = 0\mathbf{1}0_2$
$3_{10} = 0\mathbf{1}1_2$
$6_{10} = 1\mathbf{1}0_2$
$7_{10} = 1\mathbf{1}1_2$
…

Аналогичным образом формируются проверки для каждого из r проверочных символов.

Для исправления сообщения необходимо прибавить 1 к символу сообщения, номер которого определяется синдромом.

Пример

Пусть передается следующее информационное сообщение ($k = 4$):

i_4	i_3	i_2	i_1
1	1	0	0

Определим количество проверочных символов: $2^r - r - 1 = 4$, $r = 3$.

Получили (7,4)-код Хэмминга.

Разместим информационные и проверочные символы в кодовом слове:

7	6	5	4	3	2	1
i_4	i_3	i_2	p_3	i_1	p_2	p_1
1	1	0	–	0	–	–

Тогда

$p_1 = i_1 + i_2 + i_4 = 1$, $p_2 = i_1 + i_3 + i_4 = 0$, $p_3 = i_2 + i_3 + i_4 = 0$.

Предположим, что принято сообщение, содержащее однократную ошибку:

i_4	i_3	i_2	p_3	i_1	p_2	p_1
1	1	0	0	**1**	0	1

Определим синдром:

$s_3 = p_3 + i_2 + i_3 + i_4 = 0, s_2 = p_2 + i_1 + i_3 + i_4 = 1, s_1 = p_1 + i_1 + i_2 + i_4 = 1$.

Номер позиции ошибки: $011_2 = 3_{10}$. Исправленное сообщение: 1100001.

Использование синдромов позволяет значительно упростить процедуру декодирования по лидеру смежного класса: достаточно выписать лидеры смежных классов и соответствующие им синдромы. Для полученного сообщения вычисляется синдром, который затем ищется в таблице. Вычитание лидера соответствующего смежного класса из принятого сообщения дает предположительно исправленное сообщение.

2.2. Матричное описание линейных двоичных кодов

Для описания линейных двоичных кодов удобно использовать матричные обозначения. Так, уравнения для проверок на четность могут быть компактно представлены в виде *проверочной матрицы кода* $\mathbf{H}$. Например, проверочная матрица для рассмотренного ранее $(7,4)$-кода Хэмминга формируется следующим образом:

$$\mathbf{H} = \left[\begin{array}{ccccccc|l} p_1 & p_2 & i_1 & p_3 & i_2 & i_3 & i_4 & \\ \hline 0 & 0 & 0 & 1 & 1 & 1 & 1 & p_3 = i_2 + i_3 + i_4 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & p_2 = i_1 + i_3 + i_4 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & p_1 = i_1 + i_2 + i_4 \end{array}\right].$$

Проверочная матрица может быть записана в *систематической форме*. В таком случае первые k столбцов задают информационные символы, а последние $n - k$ – проверочные:

$$\mathbf{H}=\left[\begin{array}{cccc|ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right].$$
$$\underbrace{\qquad\qquad}_{k} \qquad\qquad \underbrace{\qquad\qquad}_{n-k}$$

Если (n,k)-код является линейным, то кодовые слова являются векторами линейного векторного пространства с базисом из k векторов. Эти векторы можно объединить в матрицу, которая называется *порождающей матрицей* кода $\mathbf{G}$. В систематической форме эта матрица записывается следующим образом: $\mathbf{G}=\left[E_k \mid P\right]$, где $\mathbf{P}-\ k\times(n-k)$ – матрица проверочных символов. В этой нотации проверочная матрица определяется следующим образом: $\mathbf{H}=\left[-P^T \mid E_{n-k}\right]$.

Таким образом, по систематической форме можно сразу получить из порождающей матрицы проверочную и наоборот. Для $(7,4)$-кода Хэмминга систематическая форма порождающей матрицы имеет следующий вид:

$$\mathbf{G}=\left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right].$$
$$\underbrace{\qquad\qquad}_{k} \qquad\qquad \underbrace{\qquad\qquad}_{n-k}$$

Код, в котором информационные и проверочные символы разделены, называется *систематическим кодом*.

Процедура вычисления синдрома для принятого сообщения эквивалентна умножению вектора, соответствующего сообщению, на проверочную матрицу кода: $\mathbf{s}=\mathbf{H}\mathbf{v}^T$. При этом суммируются столбцы проверочной матрицы, соответствующие единичным координатам вектора сообщения. Если принятое сообщение является кодовым словом, то синдром равен нулю: $\mathbf{H}\mathbf{c}^T=\mathbf{0}$. Если произошла ошибка, определяемая вектором ошибки, то $\mathbf{H}(\mathbf{e}+\mathbf{c})^T=\mathbf{H}\mathbf{e}^T+\mathbf{H}\mathbf{c}^T=\mathbf{s}$.

При возникновении однократных ошибок (единичный вес вектора ошибки) синдром равен соответствующему столбцу проверочной матрицы. Для кратных ошибок синдром равен сумме столбцов, соответствующих позициям ошибок. Поэтому если код имеет кодовое расстояние d^*, сумма хотя бы одного подмножества из d^* столбцов матрицы $\mathbf{H}$ равна 0 и не может существовать ни одного подмножества из d^*-1 или меньшего количества столбцов, сумма которых равна 0. Иными словами, все подмножества вектор-столбцов матрицы $\mathbf{H}$ линейно независимы.

2.3. Расширенный код Хэмминга

На основании предыдущих рассуждений можно заключить, что проверочная матрица размером $(r \times 2^r - 1)$, столбцы которой представляют двоичную запись чисел $1, 2, \mathbf{K}, 2^r - 1$, является проверочной матрицей кода Хэмминга. Минимальное расстояние кода равно 3, так как сумма любой пары различных чисел не равна нулю. Для того чтобы получить код Хэмминга с расстоянием 4, необходимо добавить общую проверку на четность. Тогда однократная ошибка будет по-прежнему давать правильный синдром, а двукратная – некоторый ненулевой синдром. Полученный $(2^r, 2^r - r - 1, 4)$-код называется *расширенным (или удлиненным) кодом Хэмминга*. Проверочная матрица этого кода имеет следующий вид:

$$\mathbf{H} = \left[\begin{array}{ccccccc|c} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right].$$

Для преобразования к систематической форме необходимо прибавить каждую из верхних строк матрицы к последней строке. Заметим, что для того чтобы определить количество линейно независимых строк проверочной матрицы кода, достаточно преобразовать ее в систематическую форму. В единичной подматрице невозможно существование линейной комбинации с нулевой суммой.

2.4. q-значный код Хэмминга

В общем случае для задания проверочной матрицы кода Хэмминга с фиксированной величиной избыточности r над полем $GF(q)$ в качестве столбцов матрицы следует выбрать из $V(r,q)$ максимально возможное количество ненулевых векторов, любая пара из которых является линейно независимой. Всего в V существует $q^r - 1$ различных ненулевых векторов и $q - 1$ ненулевых скаляров. В качестве столбца проверочной матрицы выбираем произвольный ненулевой вектор. Затем исключаем из рассмотрения все $q - 1$ векторов, произведение которых на ненулевой скаляр дает выбранный вектор.

Таким образом, всего можно выбрать $\frac{q^r - 1}{q - 1}$ столбцов проверочной матрицы. Например, для $V(2,3)$ имеем $\frac{3^2 - 1}{3 - 1} = 4$ столбца, которые выбираются из наборов векторов: $2 \cdot (0, \quad 1) = (0, \quad 2)$, $2 \cdot (1, \quad 0) = (2, \quad 0)$, $2 \cdot (1, \quad 1) = (2, \quad 2)$, $1 \cdot (1, \quad 2) = (2, \quad 1)$.

Будем выбирать те векторы, у которых первой ненулевой компонентой является единица. Отсюда $\mathbf{H}_{GF(3)} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 \end{bmatrix}$. В общем виде для кода с избыточностью $r = 2$ матрица имеет следующий вид:

$$\mathbf{H}_{GF(q)} = \begin{bmatrix} 0 & 1 & 1 & 1 & \mathbf{K} & 1 \\ 1 & 0 & 1 & 2 & \mathbf{K} & q-1 \end{bmatrix}.$$

Пример

Пусть последовательно передаются блоки сообщений, и каждый блок состоит из 4 бит. Тогда, взяв $q = 16$, получим (17, 15)-код Хэмминга, заданный над полем $GF(16)$. Транспонировав часть проверочной матрицы и добавив единичную 15×15 матрицу, получим порождающую матрицу, из которой следует, что к каждому блоку из 15 символов сообщения необходимо добавить 2 проверочных символа. Первый проверочный символ рассчитывается как сумма всех информационных, а второй как сумма произведений информационных символов на их номера. Все операции выполняются в поле $GF(16)$. Полученный код позволяет исправлять однократные ошибки в блоке из 17 символов.

2.5. Коды Боуза–Чоудхури–Хоквингема

На основании идей, использованных при построении проверочной матрицы кода Хэмминга, можно построить код, обладающий большей избыточностью и позволяющий исправлять более одной ошибки.

Рассмотрим матрицу Вандермонда, которая состоит из различных ненулевых элементов поля $a_1, a_2 \mathbf{K} a_r$ и определяется следующим образом:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & \mathbf{K} & 1 \\ a_1 & a_2 & \mathbf{K} & a_r \\ a_1^2 & a_2^2 & \mathbf{K} & a_r^2 \\ \mathbf{M} & \mathbf{M} & & \mathbf{M} \\ a_1^{r-1} & a_2^{r-1} & \mathbf{K} & a_r^{r-1} \end{bmatrix}.$$

Из курса линейной алгебры известно, что для вычисления определителя матрицы Вандермонда используется следующая формула:

$$W = \prod_{j>i=1}^{r} (x_j - x_i).$$

Так как все элементы матрицы не равны нулю и различны, из представленной формулы следует, что любая $r \times r$ подматрица имеет ненулевой определитель и любые r и меньшее количество столбцов являются линейно независимыми. Отсюда матрица Вандермонда служит проверочной матрицей кода

с минимальным расстоянием $d^* = r + 1$. Семейство таких кодов широко используется в теории и практике помехоустойчивого кодирования и называется кодами БЧХ (Боуза–Чоудхури–Хоквингема).

Процедура исправления многократных ошибок кроме вычисления синдрома принятого вектора требует решения системы уравнений в поле.

Пример

Пусть код задан в поле $GF(4)$:

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

×	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

.

Проверочная матрица БЧХ-кода, заданного в поле $GF(4)$ и позволяющего исправлять двукратные ошибки ($d = 4$), будет иметь следующий вид:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 2^2 & 3^2 \\ 1 & 2^3 & 3^3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 2 \\ 1 & 1 & 1 \end{bmatrix}.$$

Пусть $\mathbf{v}$ – принятый вектор, возможно содержащий ошибки ($\mathbf{v} = \mathbf{c} + \mathbf{e}$). Тогда выражения для вычисления синдрома имеют следующий вид:

$\mathbf{s} = (s_1, s_2, s_3, s_4) = \mathbf{H}\mathbf{v}^T$, или

$s_1 = v_1 + v_2 + v_3$, $s_2 = 1v_1 + 2v_2 + 3v_3$, $s_3 = 1^2 v_1 + 2^2 v_2 + 3^2 v_3$, $s_4 = 1^3 v_1 + 2^3 v_2 + 3^3 v_3$, или

$s_1 = v_1 + v_2 + v_3$, $s_2 = v_1 + 2v_2 + 3v_3$, $s_3 = v_1 + 3v_2 + 2v_3$, $s_4 = v_1 + v_2 + v_3$.

В общем случае для исправления двукратной ошибки, где a, b – ошибочные символы, а i, j – их позиции, следует решить систему из четырех уравнений с четырьмя неизвестными:

$$\begin{cases} s_1 = a + b, \\ s_2 = ia + jb, \\ s_3 = i^2 a + j^2 b, \\ s_4 = i^3 a + j^3 b. \end{cases} \rightarrow \begin{array}{l} b(i-j) = is_1 - s_2, \\ bj(i-j) = is_2 - s_3, \\ bj^2(i-j) = is_3 - s_4. \end{array} \rightarrow (is_2 - s_3)^2 = (is_1 - s_2)(is_3 - s_4).$$

Отсюда $(s_2^2 - s_1 s_3)i^2 + (s_1 s_4 - s_2 s_3)i + s_3^2 - s_2 s_4 = 0$ или $Ai^2 + Bi + C = 0$.

Если $\mathbf{s} = \mathbf{0}$, считается, что ошибок не произошло.

Если $\mathbf{s} \neq \mathbf{0}$ и $A = B = C = 0$, то произошла однократная ошибка s_1 в позиции $\frac{s_2}{s_1}$.

Если $\mathbf{s} \neq \mathbf{0}$ и $A \neq 0, C \neq 0, B^2 - 4AC -$ не равный нулю квадрат из $GF(4)$, то произошли две ошибки a и b в позициях i, j,

где $i, j = \dfrac{-B \pm \sqrt{B^2 - 4AC}}{2A}$ и $b = \dfrac{is_1 - s_2}{i - j}$, $a = s_1 - b$.

В противном случае количество ошибок превышает корректирующую способность кода.

2.6. Задание

1. Напишите процедуру кодирования сообщения длиной k с использованием порождающей матрицы произвольного (n,k)-кода.

2. Напишите процедуру декодирования заданного вектора длиной n с использованием проверочной матрицы произвольного (n,k)-кода.

3. Проверьте работоспособность разработанных процедур для порождающей и проверочной матрицы следующих (n,k)-кодов:

вариант 1 – расширенный код Хэмминга и код с повторениями;
вариант 2 – расширенный код Хэмминга и код с проверкой на четность;
вариант 3 – код Хэмминга и треугольный код;
вариант 4 – код Хэмминга и прямоугольный код.

4. Сгенерируйте все возможные кодовые слова расширенного кода Хэмминга для n в диапазоне от 1 до 2^{20} и определите скорость кода.

5. Сделайте выводы о достоинствах и недостатках кодов Хэмминга.

6. Сделайте выводы о достоинствах и недостатках метода синдромного декодирования.

3. МАЖОРИТАРНОЕ ДЕКОДИРОВАНИЕ

3.1. Дуальные коды

Векторы $\mathbf{u}$, $\mathbf{v}$ являются ортогональными, если скалярное произведение $\mathbf{u}\cdot\mathbf{v}=0$. Для заданного (n,k)-кода C *дуальный код* $C^{\perp}$ определяется как множество всех векторов, ортогональных к каждому кодовому слову C:

$$C^{\perp}=\{\mathbf{v}\cdot\mathbf{c}=0 \,|\, \mathbf{c}\in C\}.$$

Векторное пространство $C^{\perp}$ является *нулевым пространством* C.

Из определения синдрома кодового слова следует, что скалярное произведение каждой строки матрицы $\mathbf{G}$ на каждую строку матрицы $\mathbf{H}$ равно нулю:

$$\mathbf{H}\mathbf{G}^{T}=\mathbf{0}.$$

Обе матрицы содержат множество линейно независимых векторов и поэтому могут рассматриваться как базис некоторого линейного пространства. Таким образом, проверочная матрица кода C является порождающей матрицей кода $C^{\perp}$ и наоборот.

Пример

Код с повторением и код с проверкой на четность являются дуальными кодами.

Порождающая матрица расширенного $(8,4)$-кода Хэмминга

$$\mathbf{G}^{\perp}=\begin{bmatrix}1&0&1&0&1&0&1&0\\0&1&1&0&0&1&1&0\\0&0&0&1&1&1&1&0\\1&1&1&1&1&1&1&1\end{bmatrix}$$

совпадает с проверочной матрицей этого кода. Таким образом, этот код является дуальным самому себе. Если переместить последний столбец в начало матрицы, то получим порождающую матрицу кода, который называется кодом Рида–Маллера первого порядка.

3.2. Матрица Адамара

Матрица $\mathbf{H}_m$ размером $m\times m$, состоящая из элементов множества $\{1,\ -1\}$, называется матрицей Адамара, если выполняется следующее условие: $\mathbf{H}_m{}^{T}\mathbf{H}_m=m\mathbf{E}$.

Например, квадратная матрица $\mathbf{H}_2=\left[\begin{array}{c|c}1&1\\\hline 1&-1\end{array}\right]$ является матрицей Адамара. В общем случае матрица Адамара определяется рекурсивно: $\mathbf{H}_{2m}=\left[\begin{array}{c|c}\mathbf{H}_m&\mathbf{H}_m\\\hline \mathbf{H}_m&-\mathbf{H}_m\end{array}\right]$. Из определения следует, что все столбцы и строки

матрицы Адамара являются ортогональными. Данное свойство позволяет использовать матрицу Адамара для построения эффективных кодов.

3.3. Коды Рида–Маллера

Для любых целых чисел m и r, удовлетворяющих условию $0 \le r \le m$, существует *код Рида–Маллера порядка* r, имеющий следующие параметры: $n = 2^m$, $k = \sum_{i=0}^{r} C_m^i$, $d = 2^{m-r}$. Порядок и степень кода Рида–Маллера однозначно характеризуют этот код, поэтому наряду с общим для линейных кодов обозначением (n,k,d)-код имеется специальное $R(r,m)$.

Рассмотрим способ построения порождающей матрицы двоичных кодов Рида–Маллера с помощью матриц Адамара. Заменим в матрице Адамара элемент -1 на 0, затем последовательно будем находить кронекерово произведение $\mathbf{H}_2 \otimes \mathbf{H}_m$:

$$\mathbf{H}_2 \otimes \mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{H}_2 \otimes \mathbf{H}_4 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Получили порождающую матрицу кода $R(3,3)$.

Построение порождающей матрицы кода Рида–Маллера может быть проведено рекурсивно следующим образом:

$$\mathbf{G}_{R(r,m)} = \begin{bmatrix} \mathbf{G}_0 \\ \mathbf{G}_1 \\ \vdots \\ \mathbf{G}_r \end{bmatrix}, \text{ где } \mathbf{G}_0 = [\mathbf{g}_0] = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & 2^m \end{bmatrix},$$

$$\mathbf{G}_1 = \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \mathbf{M} \\ \mathbf{g}_m \end{bmatrix} = \begin{bmatrix} 11\underbrace{\mathbf{K}10}_{2^{m-1}}0\underbrace{\mathbf{K}}_{2^{m-1}} \\ 11\underbrace{\mathbf{K}10}_{2^{m-2}}0\underbrace{\mathbf{K}}_{2^{m-2}}11\underbrace{\mathbf{K}10}_{2^{m-2}}0\underbrace{\mathbf{K}}_{2^{m-2}} \\ \mathbf{M} \\ 1010\underbrace{\mathbf{K}10}_{2^m} \end{bmatrix}, \quad \mathbf{G}_r = \begin{bmatrix} \mathbf{g}_1\mathbf{g}_2 \\ \mathbf{g}_1\mathbf{g}_3 \\ \mathbf{M} \\ \mathbf{g}_{m-1}\mathbf{g}_m \\ \mathbf{g}_1\mathbf{g}_2\mathbf{g}_3 \\ \mathbf{M} \\ \mathbf{g}_{m-r}\mathbf{g}_{m-r+1}\mathbf{K}\mathbf{g}_m \end{bmatrix}.$$

Пример

$$\mathbf{G}_{R(1,3)} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{G}_{R(2,3)} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

Код Рида–Маллера нулевого порядка является $(n,1,2^m)$-кодом с повторением.

Код первого порядка – это код, дуальный расширенному коду Хэмминга.

$R(m,m)$ – всевозможные двоичные вектора длины 2^m.

В общем виде код Рида–Маллера является самодуальным кодом: код r-го порядка дуален коду $(m-r-1)$-го порядка.

3.4. Метод мажоритарного декодирования

Существует много способов выбора строк проверочной и порождающей матрицы кода. Любая линейная комбинация строк может служить проверочным уравнением и любое множество $n-k$ линейно независимых уравнений может быть использовано для построения проверочной матрицы кода. Аналогично, для порождающей матрицы можно найти k различных линейно независимых уравнений. На этом свойстве базируется *метод мажоритарного декодирования* (метод голосования). Данный метод отчасти уже применялся при декодировании кодов с повторением. Основная идея данного метода заключается в определении такого подмножества проверочных символов, чтобы выбор по большинству дал соответствующую позицию кодового слова.

Рассмотрим наиболее простой для реализации алгоритм. Проходя по каждой строке порождающей матрицы кода Рида–Маллера, будем проверять, участвовала ли она в формировании кодового слова. Это позволит определить

наиболее близкое к принятому сообщению кодовое слово и декодировать сообщение.

Пример

Пусть исходное сообщение $\mathbf{m}=(1110)$. Умножив $\mathbf{m}$ на порождающую матрицу $R(1,3)$, получим кодовое слово $\mathbf{c}=(11000011)$. Принято сообщение $\mathbf{c}_e=(11010011)$, содержащее однократную ошибку. Найдем набор характеристических векторов для $\boldsymbol{g}_3$ и вычислим соответствующие произведения:

$\mathbf{g}_1\mathbf{g}_2=(11000000)$, $\mathbf{g}_1\overline{\mathbf{g}}_2=(00110000)$, $\overline{\mathbf{g}}_1\mathbf{g}_2=(00001100)$, $\overline{\mathbf{g}}_1\overline{\mathbf{g}}_2=(00000011)$.

$(\mathbf{g}_1\mathbf{g}_2)\cdot\mathbf{c}_e=(11000000)\cdot(11001011)=0$, $(\mathbf{g}_1\overline{\mathbf{g}}_2)\cdot\mathbf{c}_e=(00110000)\cdot(11001011)=0$,

$(\overline{\mathbf{g}}_1\mathbf{g}_2)\cdot\mathbf{c}_e=(00001100)\cdot(11001011)=1$, $(\overline{\mathbf{g}}_1\overline{\mathbf{g}}_2)\cdot\mathbf{c}_e=(00000011)\cdot(11001011)=0$.

Большинством голосов выбираем $k_3=0$. Аналогично для $\boldsymbol{g}_2$ имеем

$\mathbf{g}_1\mathbf{g}_3=(10100000)$, $\mathbf{g}_1\overline{\mathbf{g}}_3=(01010000)$, $\overline{\mathbf{g}}_1\mathbf{g}_3=(00001010)$, $\overline{\mathbf{g}}_1\overline{\mathbf{g}}_3=(00000101)$.

$(\mathbf{g}_1\mathbf{g}_3)\cdot\mathbf{c}_e=(10100000)\cdot(11001011)=1$, $(\mathbf{g}_1\overline{\mathbf{g}}_3)\cdot\mathbf{c}_e=(01010000)\cdot(11001011)=1$,

$(\overline{\mathbf{g}}_1\mathbf{g}_2)\cdot\mathbf{c}_e=(00001010)\cdot(11001011)=0$, $(\overline{\mathbf{g}}_1\overline{\mathbf{g}}_3)\cdot\mathbf{c}_e=(00000101)\cdot(11001011)=1$.

Большинством голосов выбираем $k_2=1$.

Для $\boldsymbol{g}_1$: $\mathbf{g}_2\mathbf{g}_3=(10001000)$, $\mathbf{g}_2\overline{\mathbf{g}}_3=(01000100)$, $\overline{\mathbf{g}}_1\mathbf{g}_3=(00100010)$, $\overline{\mathbf{g}}_1\overline{\mathbf{g}}_3=(00010001)$.

$(\mathbf{g}_2\mathbf{g}_3)\cdot\mathbf{c}_e=(10001000)\cdot(11001011)=1$, $(\mathbf{g}_2\overline{\mathbf{g}}_3)\cdot\mathbf{c}_e=(010001000)\cdot(11001011)=1$,

$(\overline{\mathbf{g}}_2\mathbf{g}_3)\cdot\mathbf{c}_e=(00100010)\cdot(11001011)=1$, $(\overline{\mathbf{g}}_2\overline{\mathbf{g}}_3)\cdot\mathbf{c}_e=(00010001)\cdot(11001011)=1$.

Большинством голосов выбираем $k_1=1$.

Определяем коэффициент первой строки:

$k_3*\mathbf{g}_3=0*(10101010)=(00000000)$, $k_2*\mathbf{g}_2=1*(11001100)=(11001100)$,

$k_1*\mathbf{g}_1=1*(11110000)=(11110000)$, $k_1*\mathbf{g}_1+k_2*\mathbf{g}_2+k_3*\mathbf{g}_3=(00111100)$.

$\mathbf{c}_e+k_1*\mathbf{g}_1+k_2*\mathbf{g}_2+k_3*\mathbf{g}_3=(11010011)+(00111100)=(11101111)$.

Большинством голосов выбираем $k_0=1$.

$k_0*\mathbf{g}_0=1*(11111111)=(11111111)$,

$k_0*\mathbf{g}_0+k_1*\mathbf{g}_1+k_2*\mathbf{g}_2+k_3*\mathbf{g}_3=(11111111)+(00111100)=(11000011)$.

Получили исправленное кодовое слово и исходное информационное сообщение: $(k_3\ k_2\ k_1\ k_0)=(1110)$.

Алгоритм

Начиная с последней строки порождающей матрицы кода Рида–Маллера, для каждой i-й строки, кроме первой, повторить шаг 1 и 2.

1. Найти 2^{m-r} характеристических векторов и вычислить скалярное произведение каждого из них на полученное сообщение.

2. Определить k_i – коэффициент $\boldsymbol{g}_i$ строки как наиболее часто встречаемое значение из скалярных произведений, определенных на предыдущем шаге.

3. Для определения коэффициента первой строки k_0 необходимо вычислить произведение коэффициентов всех остальных строк на каждый скаляр соответствующей строки. Полученные векторы сложить. Наиболее встречаемое значение в результирующем векторе принимается как значение k_0.

4. Вычислить произведение $\boldsymbol{g}_0 k_0$ и прибавить к сумме произведений остальных коэффициентов и строк. Результат интерпретируется как исходное кодовое слово, а набор коэффициентов – как декодированное сообщение.

Для определения набора характеристических векторов необходимо найти векторы, которые ортогональны всем строкам проверочной матрицы, кроме искомой строки.

3.5. Задание

1. Напишите процедуру кодирования сообщения длиной k кодом Рида–Маллера порядка r.

2. Напишите процедуру мажоритарного декодирования заданного вектора длиной n с использованием следующего (n,k)-кода:

вариант 1 – код Хэмминга;
вариант 2 – расширенный код Хэмминга;
вариант 3 – код дуальный расширенному Хэмминга;
вариант 4 – код Рида–Маллера порядка r.

3. Определите время выполнения и объем используемой памяти для реализованной процедуры мажоритарного декодирования.

4. Определите скорость кода и минимальное расстояние для кода Рида–Маллера и кода, дуального к нему, для различных параметров m и r.

5. Сделайте выводы о достоинствах и недостатках кода Рида–Маллера.

6. Сделайте выводы о достоинствах и недостатках метода мажоритарного декодирования.

4. ДЕКОДИРОВАНИЕ ЦИКЛИЧЕСКИХ КОДОВ

4.1. Циклические коды

Линейный (n,k)-код называется *циклическим*, если любой циклический сдвиг символов кодового слова также является кодовым словом: $(c_{n-1}\mathbf{K}c_1c_0)\in C$, $(c_{n-2}\mathbf{K}c_1c_0c_{n-1})\in C$. Если перестановка координат кодовых слов приводит к циклическому коду, то код называется *эквивалентным* циклическому коду.

Пример

{000, 101, 010, 110} – является циклическим кодом.

{0000, 1001, 0110, 1111} – не циклический код, но является эквивалентным циклическому коду, т. к. после перестановки третьей и четвертой координат получим {0000, 1010, 0101, 1111}.

Кодовые слова циклического кода удобно интерпретировать как набор коэффициентов полинома $c(x)$, имеющего степень, не превышающую $n-1$: $(c_{n-1}\mathbf{K}c_1c_0)\rightarrow \quad c(x)=c_{n-1}x^{n-1}+\mathbf{K}+c_1x+c_0$. Такой способ задания позволяет складывать, умножать и делить кодовые слова. Действительно, сложение двух кодовых слов, заданных соответствующими полиномами, соответствует сложению их векторных представлений. Однако умножение в общем случае приводит к полиномам большей степени. Поэтому будем рассматривать полиномы в классе вычетов по модулю x^n-1. Тогда умножение кодового слова на x^m будет соответствовать его циклическому сдвигу на m позиций. Если полином $c(x)$ является кодовым словом циклического кода, то произведение $x^m c(x)$ также является кодовым словом: $c^{(m)}(x)=[x^m c(x)] \mod x^n-1$. Так как код линейный, произвольная линейная комбинация кодовых полиномов также является кодовым полиномом:

$$\sum_{i=1}^{r} w_i[x^i c(x)] \mod x^n-1=[w(x)c(x)] \mod x^n-1,$$

где r – произвольное положительное целое число, w_i – элемент поля, $\sum_{i=1}^{r} w_i x^i = w(x)$ – произвольный полином степени r.

Рассмотрим полином $g(x)$, имеющий минимальную степень r. С его помощью можно представить $c(x)$:

$$c(x)=q(x)g(x)+r(x),$$

где $q(x)$ – полином частного от деления $c(x)$ на $g(x)$, $r(x)$ – полином остатка, степень которого меньше степени $g(x)$.

Степень $q(x)g(x)$ равна степени $c(x)$ и не превосходит $n-1$. Так как $g(x)$ – кодовый полином, $q(x)g(x)$ и остаток $r(x)=c(x)-q(x)g(x)$ – также

кодовый полином. Это противоречит утверждению, что степень $r(x)$ меньше степени $g(x)$. Таким образом, $r(x)=0$.

Исходя из изложенного, любой кодовый полином циклического кода содержит в качестве множителя нормированный полином

$$g(x)=x^{n-k}+g_{n-k-1}x^{n-k-1}+\mathbf{K}+g_1x+1,$$

являющийся делителем x^n-1, который называется *порождающим полиномом*.

Пример

Для простого кода с проверкой на четность $(k=3)$ соответствие между векторным и полиномиальным представлением определяется следующим образом:

$$\begin{Bmatrix}0&0&0&0\\1&1&0&0\\0&1&1&0\\0&0&1&1\\1&0&0&1\\0&1&0&1\\1&0&1&0\\1&1&1&1\end{Bmatrix}\rightarrow\begin{Bmatrix}0\\x^3+x^2\\x^2+x\\x+1\\x^3+1\\x^2+1\\x^3+x\\x^3+x^2+x+1\end{Bmatrix}\rightarrow\begin{Bmatrix}0\cdot(x+1)\\x^2\cdot(x+1)\\x\cdot(x+1)\\1\cdot(x+1)\\(x^2+x+1)\cdot(x+1)\\(x+1)\cdot(x+1)\\(x^2+x)\cdot(x+1)\\(x^2+1)\cdot(x+1)\end{Bmatrix}.$$

Как видно из примера, порождающим полиномом для кода с проверкой на четность является полином $g(x)=x+1$.

4.2. Порождающая и проверочная матрицы циклических кодов

Из определения циклического кода следует, что порождающую матрицу кода можно построить с помощью $g(x)$:

$$\mathbf{G}=\begin{bmatrix}g(x)\\xg(x)\\\mathbf{M}\\x^{k-1}g(x)\end{bmatrix}=\begin{bmatrix}g_0&g_1&\mathbf{K}&g_{n-k}&0&\mathbf{K}&0\\0&g_0&g_1&\mathbf{K}&g_{n-k}&\mathbf{K}&0\\\mathbf{M}&\mathbf{O}&\mathbf{O}&\mathbf{O}&\mathbf{O}&\mathbf{O}&\mathbf{M}\\0&\mathbf{K}&0&g_0&g_1&\mathbf{K}&g_{n-k}\end{bmatrix}.$$

Для того чтобы закодировать сообщение циклическим кодом, достаточно умножить соответствующий сообщению полином $m(x)$ на $g(x)$.

Определим *проверочный полином* $h(x)$ степени k, такой, что

$$[c(x)h(x)]\mod x^n-1=0.$$

Кодовые слова являются множителями порождающего полинома, поэтому для проверочного полинома должно выполняться следующее соотношение:

$$c(x)h(x) = m(x)g(x)h(x) = c(x)(x^n - 1) \equiv 0.$$

Следовательно, $h(x)g(x) = x^n - 1$ или $h(x) = \dfrac{x^n - 1}{g(x)}$.

Проверочная матрица циклического кода строится с помощью $h(x)$:

$$\mathbf{H} = \begin{bmatrix} h(x) \\ xh(x) \\ \mathbb{M} \\ x^{n-k-1}h(x) \end{bmatrix} = \begin{bmatrix} h_k & h_{k-1} & \mathbf{K} & h_0 & 0 & \mathbf{K} & 0 \\ 0 & h_k & h_{k-1} & \mathbf{K} & h_0 & \mathbf{K} & 0 \\ \mathbb{M} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbb{M} \\ 0 & \mathbf{K} & 0 & h_k & h_{k-1} & \mathbf{K} & h_0 \end{bmatrix}.$$

Пример

Для рассмотренного ранее простого кода с проверкой на четность $(k = 3)$

$$g(x) = x + 1,\ \mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

$$h(x) = \frac{x^4 + 1}{x + 1} = x^3 + x^2 + x + 1,\ \mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}.$$

Следует отметить, что хотя $h(x)$ позволяет построить проверочную матрицу циклического кода C, в общем случае, порождающим полиномом для $C^{\perp}$ является не $h(x)$, а обратный ему полином $\overline{h}(x) = h_k + h_{k-1}x + \mathbf{K} + h_0 x^k$.

Главная особенность циклических кодов в том, что для кодирования и декодирования сообщений нет необходимости в использовании матриц. Умножение сообщения на порождающую или проверочную матрицу соответствует умножению на соответствующий полином.

Изложенный способ построения циклического кода дает несистематический код. Для кодирования сообщения с помощью систематического циклического кода рассмотрим сдвиг информационного сообщения вправо на $n - k$ позиций:

$$x^{n-k}m(x) = \left(\underbrace{0 \quad 0 \quad \mathbf{K} \quad 0}_{n-k} \quad m_0 \quad m_1 \quad \mathbf{K} \quad m_{k-1} \right).$$

Затем, разделив $x^{n-k}m(x)$ на $g(x)$, получим частное и остаток:

$$x^{n-k}m(x) = q(x)g(x) + r(x) \text{ или } x^{n-k}m(x) - r(x) = q(x)g(x),$$

где $r(x) = \begin{pmatrix} r_0 & r_1 & \mathbf{K}, & r_{n-k-1} & 0 & 0 & \mathbf{K} & 0 \end{pmatrix}$.

Так как разность кратна порождающему полиному, она является кодовым словом $c(x) = \begin{pmatrix} -r_0 & -r_1 & \mathbf{K}, & -r_{n-k-1} & m_0 & m_1 & \mathbf{K} & m_{k-1} \end{pmatrix}$, причем код является систематическим.

Будем последовательно находить остаток от деления x^{n-k+i} на $g(x)$ для $i = 0, 1, \mathbf{K}, k-1$:

$$x^{n-k+i} = q_i(x)g(x) + r_i(x),$$

где $r_i(x) = r_{i,n-k-1}x^{n-k-1} + \mathbf{K} + r_{i,1}x + r_{i,0}$.

Полученные кодовые слова – они кратны $g(x)$ – формируют порождающую матрицу в систематической форме:

$$\mathbf{G} = \begin{bmatrix} 0 & \mathbf{K} & 0 & 1 & r_{0,n-k} & \mathbf{K} & r_{0,1} & r_{0,0} \\ 0 & \mathbf{K} & 1 & 0 & r_{1,n-k} & \mathbf{K} & r_{1,1} & r_{1,0} \\ \mathbf{M} & \mathbf{N} & \mathbf{M} & \mathbf{M} & \mathbf{M} & \mathbf{M} & \mathbf{M} & \mathbf{M} \\ 1 & \mathbf{K} & 0 & 0 & r_{k-1,n-k} & \mathbf{K} & r_{k-1,1} & r_{k-1,0} \end{bmatrix}.$$

$\underbrace{\qquad\qquad}_{k}$ $\underbrace{\qquad\qquad\qquad\qquad}_{n-k}$

Аналогичным образом формируется проверочная матрица

$$\mathbf{H} = \begin{bmatrix} r_{k-1,0} & \mathbf{K} & r_{1,0} & r_{0,0} & 0 & \mathbf{K} & 0 & 1 \\ r_{k-1,1} & \mathbf{K} & r_{1,1} & r_{0,1} & 0 & \mathbf{K} & 1 & 0 \\ \mathbf{M} & \mathbf{M} & \mathbf{M} & \mathbf{M} & \mathbf{M} & \mathbf{N} & \mathbf{M} & \mathbf{M} \\ r_{k-1,n-k-1} & \mathbf{K} & r_{1,n-k-1} & r_{0,n-k-1} & 1 & \mathbf{K} & 0 & 0 \end{bmatrix}.$$

$\underbrace{\qquad\qquad}_{k}$ $\underbrace{\qquad\qquad\qquad\qquad}_{n-k}$

Пример

Пусть сообщение $\mathbf{m} = (101) \to m(x) = 1 + x^2$ необходимо закодировать с использованием $(7,3)$-кода, $g(x) = x^4 + x^3 + x^2 + 1$.

$x^{n-k}m(x) = x^4 m(x) = x^4 + x^6$.

$x^4 + x^6 = (1 + x + x^2)(1 + x^2 + x^3 + x^4) + (1 + x)$.

$c(x) = x^{n-k}m(x) - r(x) = (1 + x) + (x^4 + x^6) \to \mathbf{c} = (1100101)$.

$\underbrace{\qquad}_{\mathbf{m}}$

Пусть принято сообщение с ошибкой $\mathbf{c}_e = (1101\mathbf{1}0101)$. Остаток от деления соответствующего сообщению полинома $c(x) = (1 + x + x^2) + (x^4 + x^6)$ на $g(x)$ позволяет исправить ошибку:

$$c(x) \mod g(x) = [x^6 + x^4 + x^2 + x + 1] \mod [x^4 + x^3 + x^2 + 1] = x^2.$$

4.3. Реализация циклических кодов

Одной из основных особенностей циклических кодов является эффективность их аппаратной реализации.

Для аппаратной реализации циклического (n,k)-кода может использоваться линейный сдвиговый регистр с обратной связью (LFSR) с $n-k$-ячейками, соединения в котором определяются порождающим полиномом $g(x)$, либо регистр с k-ячейками, связанными согласно $h(x)$.

LFSR производит операцию деления входного полинома (делимого) на полином-делитель, который задается конфигурацией устройства деления (рис. 1).

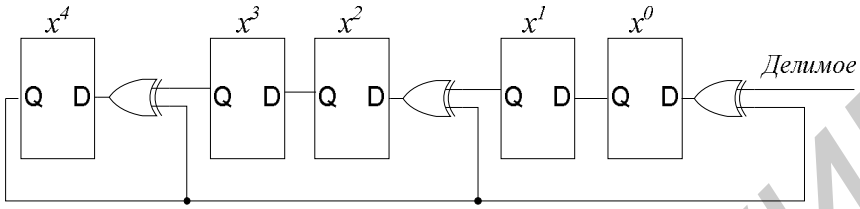

Рис. 1. Устройство деления на полином $x^5 + x^4 + x^2 + 1$

Сначала сдвиговый регистр очищается, а затем в него последовательно поступают биты делимого, при этом каждый сдвиг соответствует операции умножения полиномов. Если из регистра выдвигается бит со значением «1», из содержимого регистра вычитается делитель. На практике условие коррекции содержимого регистра может быть протестировано по содержанию старшего бита регистра до выполнения операции сдвига.

В результате выполнения деления сдвиговый регистр содержит остаток от деления, который добавляется к сообщению, и вместе с ним передается по каналам связи. Приемник выделяет сообщение по маске (при этом контрольные биты заполняются нулями), для которого затем выполняет операцию деления и сравнивает ее с переданной. Равноценной операцией является вычисление контрольной суммы для всего переданного сообщения и проверка на нулевой остаток.

4.4. Некоторые классы циклических кодов

Циклические коды, обнаруживающие и исправляющие пакет ошибок длиной L, описываются полиномом $e(x) = x^i B(x)$,

где $B(x) = x^{L-1} + b_{L-2}x^{L-2} + \mathbf{K} + b_1 x + 1,\ \{b_{L-2}, \mathbf{K}, b_1\} \in GF(2)$.

Обнаруживающая способность для циклического кода, заданного полиномом $g(x)$ степени m, определяется следующим образом:

- если $L-1 < m$, любой пакет ошибок из L-ошибок обнаруживается;
- если $L-1 = m$, обнаруживаются $1-2^{-(m-1)}$ из всех L-пакетов;
- если $L-1 > m$, обнаруживаются $1-2^{m-1}$ из всех L-пакетов.

Пример

Код, заданный порождающим полиномом $g(x) = x^{16} + x^{12} + x^5 + 1$, позволяет обнаруживать все 16-битные пакеты ошибок и 0,999985 из всех пакетов по 18 бит.

Для исправления пакета из L-ошибок код должен содержать по меньшей мере $2L$ проверочных символов. К таким кодам относится код Файра. Этот код задается порождающим полиномом вида $g(x) = (x^{2L-1} - 1)p(x)$, где $p(x)$ – примитивный полином над $GF(q)$, степень которого не меньше длины L исправляемого пакета и который не делит $x^{2L-1} - 1$.

Рассмотренные ранее коды БЧХ, которые позволяют исправлять многократные ошибки, также являются циклическими кодами.

Важный подкласс БЧХ-кодов составляют коды Рида–Соломона. Задаваемые над полем $GF(q)$, они имеют длину блока $n = q - 1$ и порождающий полином $g(x) = (x - a)(x - a^2)\mathbf{K}(x - a^{2t})$. К БЧХ-кодам относится также код Голея. Этот код задается полиномами $g(x) = x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1$ либо $g(x) = x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1$ и имеет кодовое расстояние 7.

4.5. Задание

1. Напишите процедуру кодирования произвольного сообщения длиной k с использованием циклического кода, заданного следующим порождающим полиномом:

вариант 1 – $g(x) = x^3 + x + 1$;

вариант 2 – $g(x) = x^2 + x + 1$;

вариант 3 – $g(x) = x^8 + x^7 + x^6 + x^3 + 1$;

вариант 4 – $g(x) = x^8 + x^4 + x^2 + x + 1$.

2. Напишите процедуру декодирования циклического кода и рассмотрите способы ее оптимизации.

3. Сделайте выводы о достоинствах и недостатках циклических кодов.

5. АДАПТИВНЫЙ СИГНАТУРНЫЙ АНАЛИЗ

5.1. Методы контроля и тестирования ОЗУ

В настоящее время большое внимание уделяется безопасности компьютерных систем. При этом часто под безопасностью понимается только обеспечение конфиденциальности данных, хотя не менее важной задачей является обеспечение надежности функционирования компонентов системы. Наглядно иллюстрирует данную проблему следующий пример.

В операционной системе Microsoft Windows Vista для защиты от злоумышленников применяется рандомизация размещения адресного пространства. Она обеспечивает размещение системы после каждой новой загрузки в новых случайных сегментах памяти. Модернизировав операционную систему, некоторые пользователи столкнулись с проблемой, когда при наличии дефектов в модуле ОЗУ более надежная ОС Vista часто выдает «синий» экран. Данная проблема привела к включению в поставку Windows Vista утилиты для проведения тестирования ОЗУ.

Так как ОЗУ являются неотъемлемой составляющей любой компьютерной системы, важность обеспечения их надежного функционирования не вызывает сомнений. Однако в виду регулярной и однородной структуры эти устройства имеют наибольшую степень интеграции запоминающих элементов на кристалле, что делает их чувствительными к сбоям, которые возникают в результате воздействия радиационных помех. Такие сбои проявляются как ошибки в хранимых и выходных данных. Кроме сбоев причиной ошибок могут быть неисправности, возникающие в результате несовершенства конструкции и технологии на стадии изготовления или на этапе эксплуатации ОЗУ. Сбои в отличие от неисправностей не перманентны, поэтому для выявления ошибок ОЗУ используются два различных подхода. Для обнаружения сбоев в процессе эксплуатации устройства применяются методы контроля, а для выявления неисправностей – методы тестирования.

В методах контроля ОЗУ совокупность запоминающих элементов рассматривается как канал передачи информации, в котором сообщение передается не в пространстве, а во времени (рис. 2).

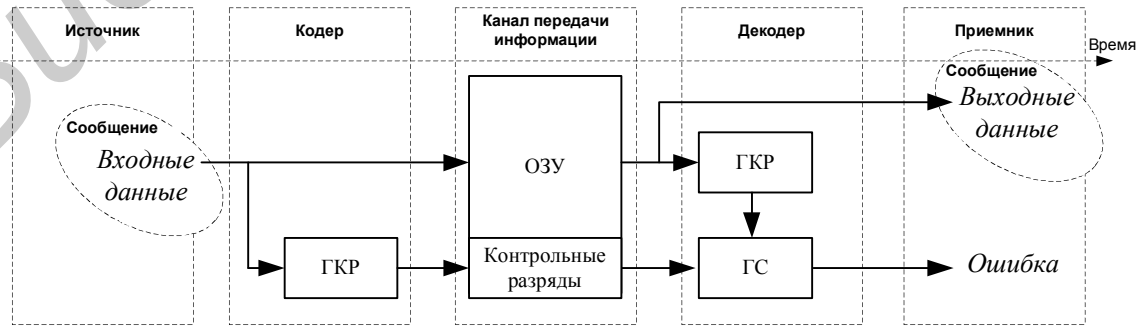

Рис. 2. Контроль ОЗУ

В связи с тем, что и приемник, и источник находятся в одном месте, процедуры кодирования и декодирования обычно совмещены и реализованы одной схемой – генератором контрольных разрядов (ГКР). Схема генератора синдрома (ГС) сравнивает сохраненные ранее контрольные разряды с вновь вычисленными и генерирует информацию об ошибке.

Для обнаружения ошибок, возникающих в результате проявления неисправностей ОЗУ, проводится тестирование. На вход тестируемой схемы подаются тестовые воздействия, которые активизируют определенные неисправности. Так как тестовые воздействия заранее известны, выполнив их сравнение с реакцией схемы, можно судить о ее исправности.

При тестировании ОЗУ возникает необходимость сравнения с эталоном большого количества значений, при этом временные и аппаратурные затраты даже для ОЗУ небольшой емкости неприемлемы. Для сокращения времени тестирования и уменьшения длины эталонной последовательности требуется сжатие выходных реакций с сохранением определенной степени достоверности.

Существует множество способов сжатия реакций схемы на тестовые последовательности, но наиболее эффективным является *сигнатурный анализ* (рис. 3).

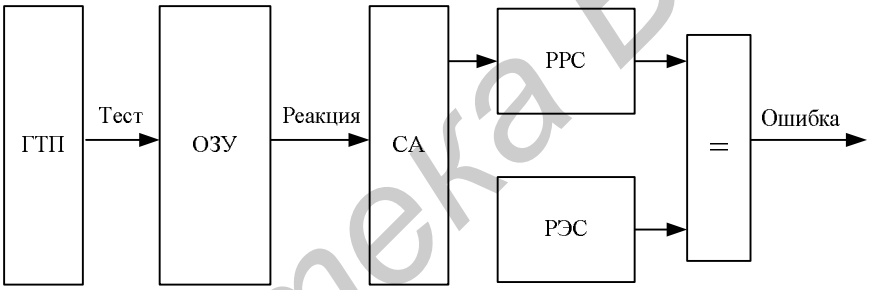

Рис. 3. Тестирование ОЗУ

Генератор тестовых последовательностей (ГТП) задает наборы тестовых данных, которые подаются на вход ОЗУ. Схема сигнатурного анализатора (СА) формирует компактную оценку выходных реакций, которая называется *сигнатурой*. Процесс сжатия осуществляется с потерей информации, поэтому основным требованием, предъявляемым к сигнатуре, является реагирование на искажение конечного числа символов в выходной последовательности, достаточного для определения факта наличия неисправности. Сигнатура, получаемая путем сжатия реакций тестируемой схемы на тестовые воздействия, сохраняется в регистре рабочей сигнатуры (РРС). Она сравнивается с предварительно рассчитанным и сохраненным в регистре эталонной сигнатуры (РЭС) значением.

Классическая схема СА предполагает использование циклических кодов и строится на базе LFSR. Реакция схемы, представленная в виде двоичной последовательности, последовательно поступает на вход LFSR, где делится на порождающий полином циклического кода. Таким образом, сигнатура представляет собой конечное значение остатка от деления. Достоверность определяется вероятностью того, что различные двоичные последовательности

имеют различные сигнатуры. Поэтому для повышения достоверности следует использовать порождающий полином циклического кода, обладающего большим кодовым расстоянием.

5.2. Адаптивный сигнатурный анализ

Основной недостаток применения классического сигнатурного анализа для оперативного контроля ОЗУ заключается в том, что при любом изменении содержимого ОЗУ требуется повторное вычисление эталонной сигнатуры, что существенно замедляет работу схемы.

Для устранения этого недостатка на сигнатурном анализаторе сжимается последовательность адресов, соответствующая хранимым в ОЗУ данным. При изменении данных эталонная сигнатура адаптируется путем суммирования по модулю два предыдущего значения сигнатуры и адреса ячейки, в которой хранимое значение изменилось. Данный метод получил название адаптивного сигнатурного анализа (АСА). Возможность адаптации эталонной сигнатуры позволила применить АСА как метод контроля ОЗУ и таким образом выявлять ошибки, возникшие в результате сбоев.

АСА выполняет вычисление синдрома с использованием проверочной матрицы линейного кода для двоичного вектора, соответствующего всему содержимому ОЗУ. Адреса ячеек ОЗУ рассматриваются как столбцы проверочной матрицы кода Хэмминга (за исключением нулевого адреса), поэтому для выполнения сжатия не требуется хранение проверочной матрицы. Достоверность метода АСА определяется достоверностью кода Хэмминга: обнаруживаются однократные и двукратные ошибки.

Необходимость повышения достоверности методов защиты от ошибок становится определяющим требованием для современных ОЗУ. Совершенствование технологии предполагает увеличение объема и плотности упаковки элементов на кристалле ОЗУ, при этом наблюдается тенденция увеличения количества возникающих многократных ошибок. Поэтому актуальной является задача разработки методов обнаружения многократных ошибок ОЗУ.

Для повышения достоверности АСА можно добавить один контрольный бит четности содержимого всей памяти. В этом случае разрядность сигнатуры АСА будет равна $m+1$ бит ($m-$ разрядность адреса). Значение дополнительного разряда эталонной сигнатуры корректируется при каждой модификации содержимого памяти путем сложения по модулю 2 константного значения 1. Если рассматривать все адресное пространство ОЗУ как транспонированную проверочную матрицу кода Хэмминга, то описанная процедура соответствует добавлению дополнительной проверки на четность (дополнительная строка и столбец матрицы $\mathbf{H}$). Полученная матрица является проверочной матрицей расширенного кода Хэмминга, который позволяет обнаруживать двукратные ошибки и все ошибки нечетной кратности.

Для обнаружения многократных ошибок с помощью АСА можно использовать код, дуальный коду Рида–Маллера второго порядка. При этом каждый m-разрядный адрес дополняется C_m^2 произведениями всех пар, составляющих данный адрес бит, и одним произведением всех составляющих адрес бит. Использование данного метода для повышения достоверности АСА позволяет обнаруживать все ошибки нечетной кратности и ошибки четной кратности, не превышающей шести при общей длине сигнатуры $m + C_m^2 + 1 = 1/2(m^2 + m) + 1$.

Для защиты от ошибок ОЗУ со словарной организацией можно использовать q-значный код Хэмминга.

5.3. Оперативный контроль динамически выделяемой памяти

Современные языки программирования предоставляют возможность выделения блоков различных размеров в большой области памяти, причем информация внутри блока располагается в памяти последовательно. Эта технология получила название *динамическое выделение памяти* и в настоящее время в той или иной степени используется почти во всех программах, которые имеют практическое применение. Она весьма удобна при работе со связанными списками или деревьями во всех случаях, когда необходимо создание множества небольших блоков данных во время выполнения программы, а также там, где необходимо освобождать и повторно использовать память.

Способ управления динамической памятью существенно влияет на производительность, поэтому он очень важен и отличается в различных ОС.

Во многих языках программирования динамическая память выделяется и освобождается вручную программистом. Это позволяет обеспечить высокую производительность и гибкость программы. Однако в случае ошибки программиста последствиями будет не только снижение производительности системы, но и зачастую снижение ее безопасности. Поэтому современные языки высокого уровня предоставляют средства контроля и обслуживания динамической памяти. Среди них можно выделить два основных метода: *счетчик ссылок* и *сборка мусора*. Счетчик ссылок подсчитывает число указателей на данный блок памяти, при его обнулении блок памяти может быть освобожден. Метод сборки мусора требует выделения для каждого блока памяти маркировочного бита. Программа выполняется до тех пор, пока свободная память не израсходована, после чего запускается процедура, которая для освобождения блоков использует маркировочный бит, после чего выполнение программы продолжается.

Ни один из рассмотренных методов не является удовлетворительным, поэтому, как правило, на практике используются их модификации.

Рассмотрим применение АСА для оперативного контроля динамически выделяемой памяти. При выделении участка динамической памяти рассчитывается эталонная сигнатура. При освобождении участка сигнатура

адаптируется. При необходимости провести контроль рассчитывается рабочая сигнатура. Несовпадение сигнатур говорит о наличии утечки памяти.

Пример реализации оперативного контроля динамически выделяемой памяти на языке C++:

```
int sign_stackPtr=0;   // указатель на вершину стека сигнатур
// оператор new с реализацией оперативного контроля
void* operator new(size_t size)
{
    void* ptr = (void*) malloc(size);
    for(int t=0;t<=sign_stackPtr;t++)
        sign[sign_stackPtr] ^= (UINT)ptr;
    return ptr;
}
// оператор delete с реализацией оперативного контроля
void operator delete(void* ptr)
{
    for(int t=0;t<=sign_stackPtr;t++)
        sign[sign_stackPtr] ^= (UINT)ptr;
    free(ptr);
}
// Класс для оперативного контроля динамически выделяемой памяти
// пример использования:
// { SignatureFrame sf;
//  ...
//   signatureAssert();
// }
class SignatureFrame
{
public:
    SignatureFrame() { sign_stackPtr++;sign[sign_stackPtr]=0; }

    ~SignatureFrame() { sign_stackPtr--; }
};
// проверка сигнатуры
void signatureAssert()
{
    if (sign[sign_stackPtr]!=0) error();
}
```

5.4. Задание

1. Напишите программу, которая моделирует работу ОЗУ с защитой от ошибок на базе метода АСА. Программа должна позволять вносить заданное число ошибок определенной кратности в массив запоминающих элементов. Ошибки могут вноситься как по заданным, так и по случайным адресам.

2. Модифицируйте программу с целью повышения достоверности. Для повышения достоверности используйте проверочные матрицы следующих линейных кодов:

вариант 1 – расширенный код Хэмминга;
вариант 2 – код, дуальный коду Рида-Маллера второго порядка;
вариант 3 – q-значный код Хэммнга;
вариант 4 – код БЧХ.

3. Сделайте выводы о достоинствах и недостатках метода адаптивного сигнатурного анализа.

ЛИТЕРАТУРА

1. Хэмминг, Р. В. Теория кодирования и теория информации / Р. В. Хэмминг. – М. : Радио и связь, 1983. – 176 с.

2. Берлекэмп, Э. Алгебраическая теория кодирования / Э. Берлекэмп. – М. : Мир, 1971. – 477 с.

3. Мак-Вильямс, Ф. Дж. Теория кодов, исправляющих ошибки / Ф. Дж. Мак-Вильямс, Н. Дж. А. Слоэн. – М. : Связь, 1979. – 744 с.

4. Питерсон, У. Коды, исправляющие ошибки / У. Питерсон, Е. Дж. мл. Уэлдон. – М. : Мир, 1976. – 594 с.

5. Блэйхут, Р. Теория и практика кодов, контролирующих ошибки / Р. Блэйхут. – М. : Мир, 1986. – 576 с.

6. Кларк, Дж. мл. Кодирование с исправлением ошибок в системах цифровой связи / Дж. мл. Кларк, Дж. Кейн. – М. : Радио и связь, 1987. – 392 с.

7. Теория прикладного кодирования : учеб. пособие. В 2 т. Т. 2 / В. К. Конопелько [и др.]. – Минск : БГУИР, 2004. – 398 с.

8. Ярмолик, В. Н. Контроль и диагностика цифровых узлов ЭВМ / В. Н. Ярмолик. – Минск : Наука и техника, 1988. – 240 с.

9. Краткий курс технической диагностики вычислительных устройств : метод. пособие / сост. М. М. Лукашевич, М. М. Татур. – Минск : БГУИР, 2005. – 36 с.

10. Иванюк, А. А. Проектирование контролепригодных цифровых устройств / А. А. Иванюк, В. Н. Ярмолик. – Минск : Бестпринт, 2006. – 296 с.

11. Кнут, Д. Э. Искусство программирования. В 3 т. Т. 1: Основные алгоритмы / Д. Э. Кнут. – М. : Издательский дом «Вильямс», 2007. – 720 с.

Св. план 2008, поз. 50

Учебное издание

Иванюк Александр Александрович
Мусин Сергей Борисович

СПЕЦИАЛЬНЫЕ ГЛАВЫ ВЫСШЕЙ МАТЕМАТИКИ: теория помехоустойчивого кодирования

Практикум
для студентов специальности I-40 01 01
«Программное обеспечение информационных технологий»
дневной и дистанционной форм обучения

Редактор Н. В. Гриневич
Корректор М. В. Тезина

Подписано в печать 12.12.2007. Формат 60×84 1/16. Бумага офсетная.
Гарнитура «Таймс». Печать ризографическая. Усл. печ. л. 2,09.
Уч.-изд. л. 2,0. Тираж 100 экз. Заказ 564.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0056964 от 01.04.2004. ЛП №02330/0131666 от 30.04.2004.
220013, Минск, П. Бровки, 6