МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

ОТЧЕТ

по лабораторной работе 2

по дисциплине

Прикладная теория информации

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ Ломакин Д. В.

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бубнов Е.А

(подпись) (фамилия, и.,о.)

\_\_\_19-ИВТ-2\_\_\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2021

**Цель работы**

Исследование алгоритмов кодирования и декодирования сообщений с помощью линейных кодов и изучение способов их технической реализации(Код Хемминга)

**Код Хемминга**

**Структурная схема**

Клиент

Сервер

Помеха

**Клиент**

Блок-схема:

Формирование сообщения

Кодирование в кодах Хемминга

Передача

начало

Конец

**Помеха**

Блок-схема:

Искажение

Прием сообщения

Соединение с клиентом и сервером

Начало

Конец

Передача

**Сервер**

Блок-схема:

Декодирование с исправлением ошибок

начало

Конец

Соединение с клиентом

Прием сообщения

Отображение

Р. Хэмминг разработал конкретную конструкцию кода. которая обеспечивает весьма элегантное обнаружение и исправление одиночных ошибок при минимально возможном числе дополнительно вводимых двоичных символов, т.е. при х= log2(l+m+x) или 2x-x-1>=m. Проследим за построением этого кода, когда m = 4. Т.е. при числе основных (информационных) двоичных символов m = 4, число дополнительно введенных, т.е. контрольных символов должно быть не менее трех. Примем, что нам удалось "обойтись" именно тремя дополнительными символами, т.е. удалось сконструировать такой код, при котором каждый из дополнительно введенных трех символов дает нам максимально возможное количество информации, т.е. по одному биту. Тогда в расширенном кодовом наборе окажутся семь двоичных символов:

B1B2B3B4 B5B6B7

(информационные символы) (контрольные символы)

Поскольку символы B1 - B4 заняты кодированием собственно текста, то управлять их значениями нам не дано. Что же касается символов B5 - B7, то они предназначены именно для обнаружения и исправления ошибок и поэтому их значения мы можем увязать со значениями информационных символов произвольными тремя функциями от аргументов B1 - B4:

B5=B5(B1,...,B4) (1.1)  
B6=B6(B1,...,B4) (1.2)

B7=B7(B1,...,B4) (1.3)

такими, чтобы в последующем с помощью трех других функций от аргументов B1 - B7

e0=e0(B1-B7) (1.4)

e1=e1(B1-B7) (1.5)

e2=e2(B1-B7) (1.6)

определить значения e0,e1,e2 содержащие информацию о том, произошла ли ошибка вообще и если да, то на уровне какого именно из семи символов. Очевидно, имеется множество различных вариантов при выборе функций (1.1)- (1.6). Р. Хэмминг поставил перед собой задачу выбора именно такой совокупности функций (1.1) - (1.6), чтобы набор значений e2e1e0 оказался двоичной записью позиции, где произошла ошибка. В случае же, когда ошибка не имела места, набор e2e1e0 должен указать на нулевую позицию, т.е. на несуществующий символ B0. Из двоичной записи этих позиций

000 (0) 1 0 0 (4)

001 (1) 1 0 1 (5)

010 (2) 1 1 0 (6)

011 (3) 1 1 1 (7)

легко уследить, что значение e0 "несет ответственность" за позиции B1, B3, B5 и B7 и поэтому в качестве функции (1.4) берется зависимость

e0 = B1+B3+B5+B7 mod 2. (1.4а)

Аналогично, обращая внимание на то, что значения e1 и e2 отвечают за соответственно B2 B3 B6 B7 , B4 B5 B6 B7, получим

e1= B2+B3+B6+B7 mod 2, (1.5a)

e2= B4+B5+B6+B7 mod 2. (1.6a)

Обратим внимание, что систему (1.4а) - (1.6а) можно рассматривать как развернутую запись матричного уравнения B1

B2

e0 1010101 B3

e1 = 0110011 B4

e2 0001111 B5

B6

B7

или   
Ve=AVa,   
где Ve, - вектор ошибки, указывающий на ее месторасположение; А - основная матрица, столбцы которой суть двоичные записи чисел от одного до семи.

Операция сложения во всех трех уравнениях (1.4а) - (1.6а) осуществляется по модулю два. Подставляя в систему уравнении (1.4а) - (1.6a) e0=e1=e2=0, получим систему из трех уравнений

B1+B1+B5+B7=0 mod2 (1.4б)

B2+B3+B6+B7=0 mod2 (1.5б)

B4+B5+B6+B7=0 mod2, (1.6б)

Приняв в качестве неизвестных величины B5, B6 и B7 получим систему из трех уравнений с тремя неизвестными:

B5+B7=B1+B3 mod2, (1.4в)

B6+B7=B2+B3 mod2, (1.5в)

B5+B6+B7=B4 mod2. (1.6в)

Эта система эквивалентна одному матричному уравнению

B1

101 B5 1010 B2

011 B6 = 0110 B3 (1.7)

111 B7 0001 B4

или

CVe=IVi. (1.7a)

где Ve и Vi, векторы-столбцы, координаты которых представлены соответственно контрольными и информационными разрядами; С и I - так называемые контрольная и информационная матрицы. Столбцы этих матриц суть двоичные записи номеров соответственно контрольных и информационных разрядов.

Решение системы (1.4в) - (1.6в), или. что то же самое, матричного уравнения (1.7) относительно B5, B6, B7 приводит к конкретным выражениям для функций (1.1) -(1.3):

B5=B2+B3+B4 mod2, (1.1а)

B6=B1+B3+B4 mod2, (1.2а)

B7=B1+B2+B4 mod2. (1.3a)

Заметим, что сам Р. Хэмминг в качестве контрольного берет не набор символов B(m+1),B(m+2), ...B(m+x), а нaбор символов, индексы которых представляют целые степени двойки. В случае, когда число контрольных символов равно трем, эти индексы равны 20 =1, 21 = 2 и 22 = 4, т.е. речь идет о наборе символов B1B2B4, относительно которых решение системы (1.4б) - (1.6б) чрезвычайно упрощается:

B1=B3+B5+B7 mod 2,

B2=B3+B6+B7 mod 2,

B4=B5+B6+B7 mod 2.

Это и естественно, поскольку в данном случае вместо (1.7) имеем дело с матричным уравнением

B3

1 0 0 B1 B5

0 1 0 B2 = B6

0 0 1 B4 B7

где контрольная матрица С всегда равна единичной матрице.

Отметив, что при указанной рекомендации Р. Хэмминга контрольная матрица всегда (независимо от m и x) оказывается равной единице, продолжаем рассматривать в качестве контрольных B5B6B7, а качестве информационных - B1B2B3B4.

Рассмотрим, к примеру, набор информационных символов B1B2B3B4 = 1011. С помощью зависимостей (1.1a) - (1.3а) определим набор контрольных (дополнительно введенных, избыточных) символов B5B6B7 = 010. Пусть ошибка произошла на уровне символа B5 т.е. вместо истинного расширенного кодового набора 1011 (0)10 получен код 1011 (1)10. Тогда с помощью зависимостей (1.4а)- (1.6а) найдем

e0=1+1+1+0=1 mod2,

e1=0+1+1+0=0 mod2,

e2=1+1+1+0=1 mod2.

Набор значений e2e1e0=101 является двоичной записью числа "пять", т.е. указывает именно на пятую позицию (на символ B5), где, собственно, и произошла ошибка.

Приведенная схема Р. Хэмминга по конструированию кода, обнаруживающего и исправляющего одиночную ошибку, универсальна, и аналогичный код может быть построен для произвольной пары значений m и x, удовлетворяющих уравнению

2х- х - 1 = m.

**Результат:**

key: ‘1’

[1] = 001100000100

number bad bit: 1

bad code = 001100000101

error = 1

***Код Листинга***

**package** codeHamming;

**import** java.util.Scanner;

**public** **class** Start {

**static** **public** **int** *cnt*;

**static** **public** **int** NextPos(**boolean** reset)

{

*cnt* = reset ? 0 : *cnt* + 1;

**return** (**int**) (Math.*pow*(2, *cnt*) - 1);

}

//----------------------------------------------//

//занесение в сообщение защитных кодов

**static** **public** **void** GetSequence(**boolean**[] code, **int** size, **char** key)

{

**int** pos = *NextPos*(**true**);

**int** i;

**int** crc = 0;

**for** (i = 0; i < size; ++i)

{

**if** (pos == i)

{

code[i] = **false**;

pos = *NextPos*(**false**);

}

**else**

{

code[i] = (**byte**)(key & 0x1) == 1? **true** : **false**;

**if** (code[i])

{

crc ^= i + 1;

}

key >>= 1;

}

}

pos = *NextPos*(**true**);

**while** (pos < size)

{

code[pos] = (**byte**)(crc & 0x1) == 1? **true** : **false**;

crc >>= 1;

pos = *NextPos*(**false**);

}

}

//----------------------------------------------//

//проверка на ошибки

**static** **public** **int** CheckCRC(**boolean**[] code, **int** size)

{

**int** i;

**int** crc = 0;

**for** (i = 0; i < size; ++i)

{

**if** (code[i])

{

crc ^= i + 1;

}

}

**return** crc == 0? -1 : crc;

}

**static** **public** **void** PrintSequence(**boolean**[] code, **int** size)

{

**int** index = size-1;

**while** (index >= 0)

{

System.***out***.print(code[index]? 1 : 0);

index--;

}

System.***out***.println();

}

**static** **public** **void** main(String[] args) {

**int** size = 12;

**boolean**[] code = **new** **boolean**[size];

**char** key;

Scanner in = **new** Scanner(System.***in***);

System.***out***.print("key: ");

key = (**char**) in.nextByte();

*GetSequence*(code, size, key);

System.***out***.printf("[%c] = ", key);

*PrintSequence*(code, size);

**int** number = 0;

System.***out***.printf("number bad bit: ");

number = in.nextInt();

**if** (--number < size)

{

//помеха

code[number] = !code[number];

}

System.***out***.print("bad = ");

*PrintSequence*(code, size);

System.***out***.printf("crc = %d\n", *CheckCRC*(code, size));

}

}

**Вывод:**

Я исследовал алгоритмы кодирования и декодирования сообщений с помощью линейных кодов и изучение способов их технической реализации(***Код хэмминга***)