Logo

Description automatically generated

Факультет: ПИиКТ

Информатика

\_Лабораторная работа\_№2

Вариант: 37

1. (33,60,87,2) 2- (38)

**Студент: Абрабоу Ахмед Елсаид А.И**

**№ группы: P3110**

**Номер студента: 333879**

**Преподаватель: Балакшин Павел Валерьевич**

**Санкт-Петербург**

**2021 г.**

**Задание:**

* Построить схему декодирования классического кода Хэмминга (7;4), которую представить в отчёте в виде изображения.
* Показать, исходя из выбранных вариантов сообщений (по 4 у каждого – часть №1 в варианте), имеются ли в принятом сообщении ошибки, и если имеются, то какие. Подробно прокомментировать и записать правильное сообщение.
* На основании номера варианта задания выбрать 1 полученное сообщение в виде последовательности 11-символьного кода. 6. Построить схему декодирования классического кода Хэмминга (15;11), которую представить в отчёте в виде изображения.
* Показать, исходя из выбранного варианта сообщений (по 1 у каждого – часть №2 в варианте), имеются ли в принятом сообщении ошибки, и если имеются, то какие. Подробно прокомментировать и записать правильное сообщение.
* Сложить номера всех 5 вариантов заданий. Умножить полученное число на 4. Принять данное число как число информационных разрядов в передаваемом сообщении. Вычислить для данного числа минимальное число проверочных разрядов и коэффициент избыточности.
* Необязательное задания для получения оценки «5» (позволяет набрать от 86 до 100 процентов от максимального числа баллов БаРС за данную лабораторную). Написать программу на любом языке программирования, которая на вход из командной строки получает набор из 7 цифр «0» и «1», записанных подряд, анализирует это сообщение на основе классического кода Хэмминга (7,4), а затем выдает 1000111правильное сообщение (только информационные биты) и указывает бит с ошибкой при его наличии.

**Отчет:**

* **(33) 🡪 (0100010)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **R1** | **R2** | **N3** | **R4** | **N5** | **N6** | **N7** |
| **0** | **1** | **0** | **0** | **0** | **1** | **0** |

**N3 N5 N7 =000=0 И R1=0 ОШИБКИ НЕТ**

**N3 N6 N7 =010=1 И R2=1 ОШИБКИ НЕТ**

**N5 N6 N7 =010=1 И** **R4=0 ошибка** НОМЕР НЕПРАВИЛЬНОГО ТОВАРА= 4=4 ДЛЯ R4

ПРАВИЛЬНОЕ КОДИРОВАНИЕ=(0101010)

* **(60) 🡪 (0011100)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **R1** | **R2** | **N3** | **R4** | **N5** | **N6** | **N7** |
| **0** | **0** | **1** | **1** | **1** | **0** | **0** |

**N3 N5 N7 =110=0 И R1=0 ОШИБКИ НЕТ**

**N3 N6 N7 =100=1 И R2=0 ошибка**

**N5 N6 N7 =100=1 И** **R4=1 ОШИБКИ НЕТ** НОМЕР НЕПРАВИЛЬНОГО ТОВАРА= 2=2 ДЛЯ R2

ПРАВИЛЬНОЕ КОДИРОВАНИЕ=(0111100)

* **(87)** 🡪 **(0011110)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **R1** | **R2** | **N3** | **R4** | **N5** | **N6** | **N7** |
| **0** | **0** | **1** | **1** | **1** | **1** | **0** |

**N3 N5 N7 =110=0 И R1=0 ОШИБКИ НЕТ**

**N3 N6 N7 =110=0 И R2=0 ОШИБКИ НЕТ**

**N5 N6 N7 =110=0 И** **R4=1 ошибка** НОМЕР НЕПРАВИЛЬНОГО ТОВАРА= 4=4 ДЛЯ R4

ПРАВИЛЬНОЕ КОДИРОВАНИЕ=(0010110)

* **(2)** 🡪 **(0010000)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **R1** | **R2** | **N3** | **R4** | **N5** | **N6** | **N7** |
| **0** | **0** | **1** | **0** | **0** | **0** | **0** |

**N3 N5 N7 =100=1 И R1=0 ошибка**

**N3 N6 N7 =100=1 И R2=0 ошибка**

**N5 N6 N7 =000=0 И** **R4=0 ОШИБКИ НЕТ** НОМЕР НЕПРАВИЛЬНОГО ТОВАРА= 1+2=3 ДЛЯ N3

ПРАВИЛЬНОЕ КОДИРОВАНИЕ=(0000000)

(38) 🡪 (010101001100010)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **R1** | **R2** | **N3** | **R4** | **N5** | **N6** | **N7** | **R8** | **N9** | **N10** | **N11** | **N12** | **N13** | **N14** | **N15** |
| **0** | **1** | **0** | **1** | **0** | **1** | **0** | **0** | **1** | **1** | **0** | **0** | **0** | **1** | **0** |

**N3 N5 N7 N9 N11 N13 N15 =0001000=1 И R1=0 ошибка**

**N3 N6 N7 N10 N11 N14 N15=0101010=1 И R2=1 ОШИБКИ НЕТ**

**N5 N6 N7 N12 N13 N14 N15=0100010=0 И** **R4=1 ошибка**

**N9 N10 N11 N12 N13 N14 N15 = 1100010=1 И** **R8=0 ошибка**

НОМЕР НЕПРАВИЛЬНОГО ТОВАРА= 1+4+8=13 ДЛЯ N13

ПРАВИЛЬНОЕ КОДИРОВАНИЕ=**(010101001100110)**

Program to implement Hamming Error Correction Code in Java  
/\*

Program to implement Hamming Error Correction Code in Java  
By: Abrabou Ahmed Elsaid

\*/

import java.util.\*;

class Hamming {

public static void main(String args[]) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the number of bits for the Hamming data:");

int n = scan.nextInt();

int a[] = new int[n];

for(int i=0 ; i < n ; i++) {

System.out.println("Enter bit no. " + (n-i) + ":");

a[n-i-1] = scan.nextInt();

}

System.out.println("You entered:");

for(int i=0 ; i < n ; i++) {

System.out.print(a[n-i-1]);

}

System.out.println();

int b[] = generateCode(a);

System.out.println("Generated code is:");

for(int i=0 ; i < b.length ; i++) {

System.out.print(b[b.length-i-1]);

}

System.out.println();

// Difference in the sizes of original and new array will give us the number of parity bits added.

System.out.println("Enter position of a bit to alter to check for error detection at the receiver end (0 for no error):");

int error = scan.nextInt();

if(error != 0) {

b[error-1] = (b[error-1]+1)%2;

}

System.out.println("Sent code is:");

for(int i=0 ; i < b.length ; i++) {

System.out.print(b[b.length-i-1]);

}

System.out.println();

receive(b, b.length - a.length);

}

static int[] generateCode(int a[]) {

// We will return the array 'b'.

int b[];

// We find the number of parity bits required:

int i=0, parity\_count=0 ,j=0, k=0;

while(i < a.length) {

// 2^(parity bits) must equal the current position

// Current position is (number of bits traversed + number of parity bits + 1).

// +1 is needed since array indices start from 0 whereas we need to start from 1.

if(Math.pow(2,parity\_count) == i+parity\_count + 1) {

parity\_count++;

}

else {

i++;

}

}

// Length of 'b' is length of original data (a) + number of parity bits.

b = new int[a.length + parity\_count];

// Initialize this array with '2' to indicate an 'unset' value in parity bit locations:

for(i=1 ; i <= b.length ; i++) {

if(Math.pow(2, j) == i) {

// Found a parity bit location.

// Adjusting with (-1) to account for array indices starting from 0 instead of 1.

b[i-1] = 2;

j++;

}

else {

b[k+j] = a[k++];

}

}

for(i=0 ; i < parity\_count ; i++) {

// Setting even parity bits at parity bit locations:

b[((int) Math.pow(2, i))-1] = getParity(b, i);

}

return b;

}

static int getParity(int b[], int power) {

int parity = 0;

for(int i=0 ; i < b.length ; i++) {

if(b[i] != 2) {

// If 'i' doesn't contain an unset value,

// We will save that index value in k, increase it by 1,

// Then we convert it into binary:

int k = i+1;

String s = Integer.toBinaryString(k);

//Nw if the bit at the 2^(power) location of the binary value of index is 1

//Then we need to check the value stored at that location.

//Checking if that value is 1 or 0, we will calculate the parity value.

int x = ((Integer.parseInt(s))/((int) Math.pow(10, power)))%10;

if(x == 1) {

if(b[i] == 1) {

parity = (parity+1)%2;

}

}

}

}

return parity;

}

static void receive(int a[], int parity\_count) {

// This is the receiver code. It receives a Hamming code in array 'a'.

// We also require the number of parity bits added to the original data.

// Now it must detect the error and correct it, if any.

int power;

// We shall use the value stored in 'power' to find the correct bits to check for parity.

int parity[] = new int[parity\_count];

// 'parity' array will store the values of the parity checks.

String syndrome = new String();

// 'syndrome' string will be used to store the integer value of error location.

for(power=0 ; power < parity\_count ; power++) {

// We need to check the parities, the same no of times as the no of parity bits added.

for(int i=0 ; i < a.length ; i++) {

// Extracting the bit from 2^(power):

int k = i+1;

String s = Integer.toBinaryString(k);

int bit = ((Integer.parseInt(s))/((int) Math.pow(10, power)))%10;

if(bit == 1) {

if(a[i] == 1) {

parity[power] = (parity[power]+1)%2;

}

}

}

syndrome = parity[power] + syndrome;

}

// This gives us the parity check equation values.

// Using these values, we will now check if there is a single bit error and then correct it.

int error\_location = Integer.parseInt(syndrome, 2);

if(error\_location != 0) {

System.out.println("Error is at location " + error\_location + ".");

a[error\_location-1] = (a[error\_location-1]+1)%2;

System.out.println("Corrected code is:");

for(int i=0 ; i < a.length ; i++) {

System.out.print(a[a.length-i-1]);

}

System.out.println();

}

else {

System.out.println("There is no error in the received data.");

}

// Finally, we shall extract the original data from the received (and corrected) code:

System.out.println("Original data sent was:");

power = parity\_count-1;

for(int i=a.length ; i > 0 ; i--) {

if(Math.pow(2, power) != i) {

System.out.print(a[i-1]);

}

else {

power--;

}

}

System.out.println();

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Output:

Enter the number of bits for the Hamming data:

7

Enter bit no. 7:

1

Enter bit no. 6:

0

Enter bit no. 5:

1

Enter bit no. 4:

0

Enter bit no. 3:

1

Enter bit no. 2:

0

Enter bit no. 1:

1

You entered:

1010101

Generated code is:

10100101111

Enter position of a bit to alter to check for error detection at the receiver end (0 for no error):

5

Sent code is:

10100111111

Error is at location 5.

Corrected code is:

10100101111

Original data sent was:

1010101

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Another code\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* hamming\_code =input(**"Enter the code of 7 symbols in hamming coding "**)  
  index\_of\_error=0  
  **if** int(hamming\_code[0])!= ((int(hamming\_code[2])+int(hamming\_code[4])+int(hamming\_code[6]))%2):  
   index\_of\_error+=1  
  **if** int(hamming\_code[1])!= ((int(hamming\_code[2])+int(hamming\_code[5])+int(hamming\_code[6]))%2):  
   index\_of\_error += 2  
  **if** int(hamming\_code[3])!= ((int(hamming\_code[4])+int(hamming\_code[5])+int(hamming\_code[6]))%2):  
   index\_of\_error += 4  
    
  **if** index\_of\_error!=0:  
   **if** hamming\_code[index\_of\_error-1]==**'1'**:  
   hamming\_code=hamming\_code[:(index\_of\_error-1 )]+**"0"**+hamming\_code[(index\_of\_error):]  
   **else**:  
   hamming\_code = hamming\_code[:(index\_of\_error-1)] + **"1"** + hamming\_code[(index\_of\_error):]  
    
   print(**f"the mistake is in digit number {index\_of\_error} \n the correct code is {hamming\_code}"**)  
    
  **else**:  
   print(**f"the code {hamming\_code} is correct "**)
* **Вывод:**

Вывод о проделанной работе.

в этой лаборатории я применил теоретические предыдущие знания, которые у меня были о коде Хэмминга, и применил их на практике