Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____Информатика и системы управления_____

КАФЕДРА _____Системы обработки информации и управления_____

Отчёт по домашнему заданию по курсу

«Сети и телекоммуникации»

Выполнил: Студент группы **РТ5-51** _____ **Астанов Э.М.**

(Подпись, дата) (Фамилия И.О.)

Проверил: Преподаватель. _____ **Галкин В.А.**

(Подпись, дата) (Фамилия И.О.)

Москва, 2018

Постановка задачи.

Имеется дискретный канал связи, на вход которого подается закодированная в соответствии с вариантом задания кодовая последовательность. В канале возможны ошибки любой кратности. Вектор ошибки может принимать значения от единицы в младшем разряде до единицы во всех разрядах кодового вектора. Для каждого значения вектора ошибки на выходе канала после декодирования определяется факт наличия ошибки и предпринимается попытка ее исправления.

Обнаруживающая способность кода C_o определяется как отношение числа обнаруженных ошибок N_o к общему числу ошибок данной кратности, которое определяется как число сочетаний из n (длина кодовой комбинации) по i (кратность ошибки – число единиц в векторе ошибок) - C^i_n.

Описание алгоритмов кодирования и декодирования.

1. Алгоритм кодирования.

№ варианта	Информационный вектор	Код	Способность кода
4	1010	X [7,4]	C_k

Информационный кодовый вектор v = 1010. В коде Хемминга этот вектор будет занимать позиции C_3, C_5, C_6 и C_7, начиная с младшего разряда, а позиции C_1, C_2, C_4 отводятся под проверочные разряды кода.

Пронумеруем все позиции кода в двоичной системе счисления:

$$C_{111}C_{110}C_{101}[C_{100}]C_{011}[C_{010}][C_{001}]$$

и выделим позиции для размещения проверочных разрядов. Определим значения проверочных разрядов кода суммированием по mod2 тех разрядов кода, в номере которых двоичный разряд с (i)-ым весом равен единице.

C_{111}	C_{110}	C_{101}	C_{100}	C_{011}	C_{010}	C_{001}
1	0	1	0	0	1	0

$$i = 0 \to 2^0 \to c_{001} = c_{011} \oplus c_{101} \oplus c_{111}$$
$$i = 1 \to 2^1 \to c_{010} = c_{011} \oplus c_{110} \oplus c_{111}$$
$$i = 2 \to 2^2 \to c_{100} = c_{101} \oplus c_{110} \oplus c_{111}$$

Таким образом получен кодовый вектор v’ = 1010010, который передается по каналу, подверженному влиянию помех.

2. Алгоритм декодирования.

Имеем код Хемминга v’ = 1010010.

Пусть код имеет ошибку в 5 разряде.

$$\text{v}'' = 1110010$$

Вычислим значение синдрома ошибки:

$$\text{E}_{\text{ош}} = \|h_r h_{r-1} \dots h_i \dots h_1\|$$

$$h_1 = c_{001} \oplus c_{011} \oplus c_{101} \oplus c_{111}$$
$$h_2 = c_{010} \oplus c_{011} \oplus c_{110} \oplus c_{111}$$
$$h_3 = c_{100} \oplus c_{101} \oplus c_{110} \oplus c_{111}$$

Для нашего вектора v’’ = 1110010

$$h_1 = 0$$

$$h_2 = 1$$

$$h_3 = 1$$

Получаем, что $\text{E}_{\text{ош}} = ||110||$. Следовательно ошибка в 5 разряде. Для ее исправление необходимо инвертировать этот разряд v’’ = 1010010.

Остается убрать $\text{C}_1, \text{C}_2, \text{C}_4$. Получаем:

$$\text{v} = 1010$$

Исходный код программы.

```
# Подключаем необходимые библиотеки
import itertools
import numpy as np
import pandas as pd
from math import log2


# Проверка числа, является ли оно степенью двойки
def is_pow_2(number):
    return int(log2(number)) == float(log2(number))

class OTK:
    def __init__(self, word):
        self.word = word
        self.encoded_word = None

    # Функция для представления кодового слова в виде 101?0??
    def getCode(self,word):
        word_iterator = 0
        bit_iterator = 1
        q_code = ''
        while word_iterator < len(word):
            if is_pow_2(bit_iterator):
                q_code += '?'
            else:
                q_code += word[word_iterator]
                word_iterator += 1
            bit_iterator += 1
        return q_code

    # Функция создания матрицы, по которой будет осуществлена подстановка 1
или 0 вместо "?"
    def initMatrix(self,column_indexes, row_indexes):
        matrix = np.zeros([len(row_indexes), len(column_indexes)])
        for c in range(0, len(column_indexes)):
            for r in range(0, len(row_indexes)):
                if column_indexes[c] <= row_indexes[r]:
                    if matrix[r].sum() + column_indexes[c] \
                            <= row_indexes[r]:
                        matrix[r][c] = column_indexes[c]
        return matrix

    # Функция кодирования
    def encodeHamming(self,word):
        q_code = self.getCode(word)
        column_indexes = []
        row_indexes = []
        for i in range(0, len(q_code)):
            if q_code[i] == '?':
                column_indexes.append(i + 1)
            elif q_code[i] == '1':
                row_indexes.append(i + 1)
```

```
        column_indexes = np.array(column_indexes)[::-1]
        matrix = self.initMatrix(column_indexes, row_indexes)
        encoded_code = list(q_code)
        for c in range(0, len(column_indexes)):
            matrix[:, c] /= column_indexes[c]
            _sum = matrix[:, c].sum()
            if _sum % 2 == 0:
                encoded_code[column_indexes[c] - 1] = '0'
            else:
                encoded_code[column_indexes[c] - 1] = '1'
        return ''.join(encoded_code)

    # Функция декодирования
    def decodeHamming(self,word):
        column_indexes = []
        row_indexes = []
        for i in range(0, len(word)):
            if is_pow_2(i + 1):
                column_indexes.append(i + 1)
            if word[i] == '1':
                row_indexes.append(i + 1)
        column_indexes = np.array(column_indexes)[::-1]
        matrix = self.initMatrix(column_indexes, row_indexes)
        errors = []
        decoded_code = []
        for c in range(0, len(column_indexes)):
            matrix[:, c] /= column_indexes[c]
            _sum = matrix[:, c].sum()
            if _sum % 2 != 0:
                errors.append(column_indexes[c])
        if sum(errors) == 0:
            for i in range(0, len(word)):
                if not is_pow_2(i + 1):
                    decoded_code.append(word[i])
            return ''.join(decoded_code)
        try:
            pre_decoded_code = list(word)
            decoded_code = []
            error_bit = sum(errors) - 1
            pre_decoded_code[error_bit] = str(1 -
int(pre_decoded_code[error_bit]))
            for i in range(0, len(pre_decoded_code)):
                if not is_pow_2(i + 1):
                    decoded_code.append(pre_decoded_code[i])
            return ''.join(decoded_code)
        except Exception as err:
            print(err)
            return None

    # Создание таблицы
    def createTable(self,encoded_word):
        N = len(encoded_word)
        table_data = []
        for i in range(len(encoded_word)):
```

```
            errors = list(itertools.combinations(range(N), i))
            C = len(errors)
            decoded_right = 0
            for error in errors:
                error_combinations = list(error)
                err_vector = list(encoded_word)
                if len(error_combinations) == 0 : continue
                for ec in range(0,len(error_combinations)):
                    err_vector[error_combinations[ec]] = str(1 -
int(err_vector[error_combinations[ec]]))
                if self.decodeHamming(''.join(err_vector)) == self.word:
                    if i == 2 : print(''.join(err_vector))
                    decoded_right+=1

table_data.append({"i":i,"C":C,"Nk":decoded_right,"Ck":decoded_right/C},)
        return
pd.DataFrame.from_records(table_data,columns=['i','C','Nk','Ck'])

    # Отображение результата
    def getResult(self):
        self.encoded_word = self.encodeHamming(self.word)
        df = self.createTable(self.encoded_word)

        print('Исходный код: {0} | Код Хемминга:
{1}\n'.format(self.word,self.encoded_word))
        print(df)
```

Результат выполнения.

```
if __name__ == '__main__':
    run = OTK('1010')
    run.getResult()
```

```
Исходный код: 1010 | Код Хемминга: 1011010

   i   C  Nk   Ck
0  0   1   0  0.0
1  1   7   7  1.0
2  2  21   0  0.0
3  3  35   0  0.0
4  4  35   0  0.0
5  5  21   0  0.0
6  6   7   0  0.0
```

	I	C	N_K	C_K
0	0	1	0	0
1	1	7	7	1
2	2	21	0	0
3	3	35	0	0
4	4	35	0	0
5	5	21	0	0
6	6	7	0	0

Выводы

Из итоговой таблицы можно сделать следующие выводы:

1) Программа не смогла дешифровать коды с кратностью ошибки больше 1 – так как алгоритм может исправлять не более одной ошибки.

2) При кратности ошибки = 0, программа ничего не исправила.

3) При кратности ошибки = 1, программа смогла дешифровать все значения.

Из 1), 2) и 3) следует, что программа работает верно.

Литература

● Методические указания к выполнению данного ДЗ

● Галкин В.А., Григорьев Ю.А. Телекоммуникации и сети: Учеб. Пособие для вузов.-М.: Изд-во МГТУ им.Н.Э.Баумана, 2003

● Код Хэмминга. Пример работы алгоритма – Хабрахабр. https://habr.com/post/140611/

● https://www.youtube.com/watch?v=ekCzFnBHAnc&t=2s - Визуализация алгоритма по шифрованию и декодированию кода Хэмминга

● http://all-ht.ru/inf/systems/p_0_14.html