 **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технический университет**

**имени Н.Э. Баумана (национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления»**

**Кафедра «Системы обработки информации и управления»**

Домашнее задание №2

по дисциплине «Сети и телекоммуникации»

на тему:

«Разработка алгоритмов кодирования, декодирования и определения обнаруживающей и корректирующей способности кода в линейных протоколах»

Вариант №6

Выполнил:

студент группы ИУ5-53Б

Иванченко М.Д.

Проверил:

к.т.н., доц., Галкин В.А.

2023 г.

**Метод решения задачи**

Задача решена на языке Python 3.10.7 в среде разработки PyCharm Community Edition. Результат работы программы представлен в виде промежуточных значений, а также таблицы формата:

|  |  |  |  |
| --- | --- | --- | --- |
| k | No | Ckn | Cо |
|  |  |  |  |

Используякодирование кодом Хэмминга Х [7,4], определить обнаруживающую способность кода Со:

|  |  |  |  |
| --- | --- | --- | --- |
| № варианта | Информационный вектор | Код | Способность кода |
| 6 | 1110 | Х [7,4] | Со |

Обозначения:

Х [7,4] – код Хэмминга;

n – число разрядов кода: n = 7;

k – число информационных разрядов: k = 4.

Алгоритм кодирования:

Все номера позиций кода нумеруются в двоичной системе счисления, начиная с единицы, p-разрядным двоичным числом, причем

Проверочные разряды размещаются в позициях кода, кратных целой степени двойки. В нашем случае это позиции под номерами 1 (20), 2 (21) и 4 (22), то есть при i = 0, 1, 2. Остальные разряды – информационные, и заполняются разрядами исходного вектора.

Значение проверочного разряда определяется как сумма по mod2 тех разрядов кода, в номере которых двоичный разряд с (i)-ым весом равен единице.

Алгоритм декодирования:

Значение i-го разряда синдрома ошибки Eош = || hrhr-1 ...hi...h1 || определяется как сумма по mod2 тех разрядов принятого кода, включая проверочные, в номере которых вес двоичного разряда совпадает с весом разряда синдрома. Синдром ошибки определяет в двоичной системе номер разряда, в котором обнаружена однократная ошибка. Для исправления необходимо инвертировать разряд под этим номером.

Алгоритм вычисления обнаруживающей способности кода:

Для каждого возможного вектора ошибок 𝑒 данной кратности 𝑖 найти количество обнаруженных ошибок 𝑁0 (когда значение синдрома ошибки Eош равно нулю, то есть Eош = ||000||).

Вычисление обнаруживающей способности кода 𝐶0 для данной кратности ошибки 𝑖, которое определяется как отношение числа обнаруженных ошибок 𝑁0 к общему числу ошибок данной кратности. Общее число ошибок данной кратности определяется как число сочетаний из 𝑛 (длина кодовой комбинации) по 𝑖 (кратность ошибки – число единиц в векторе ошибок) – C𝑖n:

Реализация алгоритма кодирования:

def get\_checked\_pos(pattern, one\_pos):  
 num = 1  
 summa = 0  
 for symbol in list(reversed(pattern)):  
 if format(num, '03b')[-one\_pos] == '1' and symbol != ' ':  
 summa += int(symbol)  
 num += 1  
 summa %= 2  
 return summa  
  
  
def code\_vector(v):  
 res = list(v)  
 res.append(' ') *# проверочный разряд 001* res.insert(-1, ' ') *# проверочный разряд 010* res.insert(-3, ' ') *# проверочный разряд 100* res[-1] = str(get\_checked\_pos(res, 1))  
 res[-2] = str(get\_checked\_pos(res, 2))  
 res[-4] = str(get\_checked\_pos(res, 3))  
 return ''.join(res)

Реализация модели канала связи (генерация ошибок):

def is\_k\_error(num, k):  
 while k > 0 and num > 0:  
 num &= num - 1  
 k -= 1  
 if num == 0 and k == 0:  
 return True  
 return False  
  
  
def gen\_errors(n, k):  
 res = []  
 limit = factorial(n) / (factorial(n - k) \* factorial(k)) *# сочетание из n по k* num = 1  
 while len(res) < limit:  
 if is\_k\_error(num, k):  
 res.append(format(num, 'b'))  
 num += 1  
 return res

Реализация алгоритма декодирования:

def error\_syndrome(v):  
 ind = 3  
 res = []  
 while ind > 0:  
 summa = 0  
 cnt = 1  
 for symbol in v[::-1]:  
 if format(cnt, '3b')[-ind] == '1':  
 summa += int(symbol)  
 cnt += 1  
 summa %= 2  
 res.append(str(summa))  
 ind -= 1  
 return ''.join(res)  
  
  
def correct\_damaged\_vector\_1\_error(v):  
 err\_syndrome\_num = int(error\_syndrome(v), 2)  
 corrected\_vector = format(int(v, 2) ^ int('1' + '0' \* (err\_syndrome\_num - 1), 2), '07b')  
 return corrected\_vector  
  
def decode\_damaged\_vector(v):  
 corrected\_vector = correct\_damaged\_vector\_1\_error(v)  
 res = corrected\_vector[:3] + corrected\_vector[4]  
 return res

Реализация алгоритма вычисления обнаруживающей способности кода:

from gen\_errors import gen\_errors, get\_k\_errors\_dict, factorial  
from decode\_vector import error\_syndrome  
  
def info\_for\_k\_error(v, n, k):  
 err = 0  
 errors = gen\_errors(n, k)  
 total\_err = len(errors) *# кол-во ошибок равно числу сочетаний из n по k* for error in errors:  
 damaged\_vector = format(int(v, 2) ^ int(error, 2), '07b')  
 if error\_syndrome(damaged\_vector).find('1') != -1:  
 err += 1  
 return err, total\_err, (err \* 1.0 / total\_err)

def factorial(x):  
 if x == 0 or x == 1:  
 return 1  
 return x \* factorial(x - 1)  
  
def is\_k\_error(num, k):  
 while k > 0 and num > 0:  
 num &= num - 1  
 k -= 1  
 if num == 0 and k == 0:  
 return True  
 return False  
  
def gen\_errors(n, k):  
 res = []  
 limit = factorial(n) / (factorial(n - k) \* factorial(k)) *# сочетание из n по k* num = 1  
 while len(res) < limit:  
 if is\_k\_error(num, k):  
 res.append(format(num, 'b'))  
 num += 1  
 return res

Реализация вывода информации:

import pandas as pd  
from tabulate import tabulate  
from code\_vector import code\_vector  
from decode\_vector import decode\_damaged\_vector, correct\_damaged\_vector\_1\_error, error\_syndrome  
from detect\_ability import info\_for\_k\_error  
  
  
def main():  
 n = 7 *# длина кодовой комбинации Х[7,4]* vector = input('Enter info vector v : ') *# ввод информационного вектора* coded\_vector = code\_vector(vector)  
 print("Encoded vector v' = " + coded\_vector)  
 e = input('Enter error vector e: ')  
 damaged\_vector = format(int(coded\_vector, 2) ^ int(e, 2), '07b')  
 print("Now damaged vector v'' = " + damaged\_vector)  
 print("Error syndrome E = " + error\_syndrome(damaged\_vector))  
 print("Corrected vector v'' = " + correct\_damaged\_vector\_1\_error(damaged\_vector))  
 print("Decoded vector v = " + decode\_damaged\_vector(damaged\_vector))  
 print('------------------------------------------------------------------------------------------')  
 output\_table(coded\_vector, n)  
  
def output\_table(coded\_v, n):  
 table = []  
 for i in range(1, n + 1):  
 row = [i] + list(info\_for\_k\_error(coded\_v, n, i))  
 table.append(row)  
 pd.set\_option("display.precision", 4)  
 df = pd.DataFrame(table, columns=['Кратность, k', 'Число обнаруженных ошибок, N0', 'Общее число ошибок, Ckn',  
 'Обнаруживающая способность, C0'])  
 print(tabulate(df, showindex=False, headers=df.columns))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Результат работы программы:

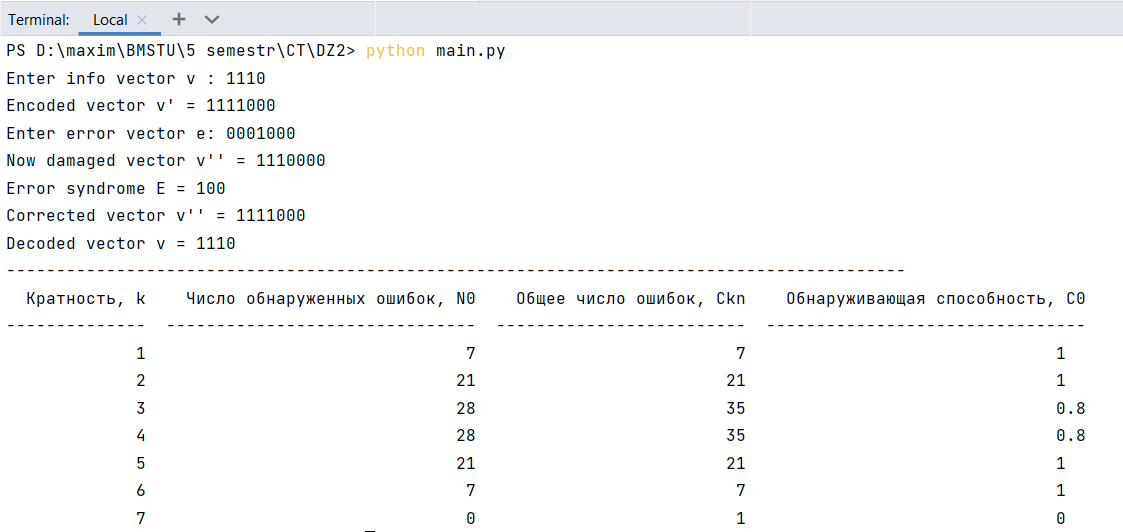


Таблица результатов:

|  |  |  |  |
| --- | --- | --- | --- |
| k | No | Ckn | Cо |
| 1 | 7 | 7 | 1 |
| 2 | 21 | 21 | 1 |
| 3 | 28 | 35 | 0.8 |
| 4 | 28 | 35 | 0.8 |
| 5 | 21 | 21 | 1 |
| 6 | 7 | 7 | 1 |
| 7 | 0 | 1 | 0 |

Вывод:

По таблице результатов видно, что только при кратности ошибки, равной 7, обнаруживающая способность кода Хэмминга [7,4]-кода равна 0, и он не способен обнаружить ошибку. При кратностях ошибки, равных 1, 2, 5 и 6, обнаруживающая способность кода равна 1 и ошибку всегда можно обнаружить. При остальных кратностях ошибки обнаруживающая способность кода близка к 1 и ошибку почти всегда можно обнаружить. Таким образом, код Хэмминга [7,4]-код подходит для обнаружения ошибок кратностью до 6.

Электронная копия отчета и решения:

<https://github.com/MaximIvanchenko18/Nets_5sem_DZ>

**Список используемой литературы**

1. Галкин В.А., Григорьев Ю.А. Телекоммуникации и сети: Учеб. Пособие для вузов.-М.: Изд-во МГТУ им. Н.Э.Баумана, 2003
2. <https://habr.com/ru/articles/140611/>
3. Конспект лекций по дисциплине “Сети и телекоммуникации”. М.: МГТУ им. Н.Э. Баумана. 2023 г.