МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В. Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа № 7**

по дисциплине: Теория информации

тема: «Исследование помехоустойчивых кодов на примере алгоритма Хэмминга»

Выполнил: ст. группы ПВ-223

Игнатьев Артур Олегович

Проверил:

Твердохлеб Виталий Викторович

Белгород 2024г.

**Лабораторная работа №7**

**«Исследование помехоустойчивых кодов на примере алгоритма Хэмминга»**

**Задание 1**

Закодировать по Хэммингу произвольно сформированные последовательности двоичных символов длиной:

- 18 бит;

- 48 бит.

Сделать выводы относительно эффективности каждого их сообщений.

import random  
  
# Длина блока кодирования  
CHUNK\_LENGTH = 8  
  
# Проверка длины блока  
assert CHUNK\_LENGTH > 0, 'CHUNK\_LENGTH должен быть больше 0'  
  
# Вычисление контрольных бит  
CHECK\_BITS = [i for i in range(1, CHUNK\_LENGTH + 1) if not i & (i - 1)]  
  
def chars\_to\_bin(chars):  
 *"""  
 Преобразование символов в бинарный формат  
 """* assert not len(chars) \* 8 % CHUNK\_LENGTH, 'Длина кодируемых данных должна быть кратна длине блока кодирования'  
 return ''.join([bin(ord(c))[2:].zfill(8) for c in chars])  
  
def chunk\_iterator(text\_bin, chunk\_size=CHUNK\_LENGTH):  
 *"""  
 Поблочный вывод бинарных данных  
 """* for i in range(0, len(text\_bin), chunk\_size):  
 yield text\_bin[i:i + chunk\_size]  
  
def get\_check\_bits\_data(value\_bin):  
 *"""  
 Получение информации о контрольных битах из бинарного блока данных  
 """* check\_bits\_count\_map = {k: 0 for k in CHECK\_BITS}  
 for index, value in enumerate(value\_bin, 1):  
 if int(value):  
 bin\_char\_list = list(bin(index)[2:].zfill(8))  
 bin\_char\_list.reverse()  
 for degree in [2 \*\* int(i) for i, value in enumerate(bin\_char\_list) if int(value)]:  
 check\_bits\_count\_map[degree] += 1  
 check\_bits\_value\_map = {}  
 for check\_bit, count in check\_bits\_count\_map.items():  
 check\_bits\_value\_map[check\_bit] = 0 if not count % 2 else 1  
 return check\_bits\_value\_map  
  
def set\_empty\_check\_bits(value\_bin):  
 *"""  
 Добавить в бинарный блок "пустые" контрольные биты  
 """* for bit in CHECK\_BITS:  
 value\_bin = value\_bin[:bit - 1] + '0' + value\_bin[bit - 1:]  
 return value\_bin  
  
def set\_check\_bits(value\_bin):  
 *"""  
 Установить значения контрольных бит  
 """* value\_bin = set\_empty\_check\_bits(value\_bin)  
 check\_bits\_data = get\_check\_bits\_data(value\_bin)  
 for check\_bit, bit\_value in check\_bits\_data.items():  
 value\_bin = '{0}{1}{2}'.format(value\_bin[:check\_bit - 1], bit\_value, value\_bin[check\_bit:])  
 return value\_bin  
  
def get\_check\_bits(value\_bin):  
 *"""  
 Получить информацию о контрольных битах из блока бинарных данных  
 """* check\_bits = {}  
 for index, value in enumerate(value\_bin, 1):  
 if index in CHECK\_BITS:  
 check\_bits[index] = int(value)  
 return check\_bits  
  
def exclude\_check\_bits(value\_bin):  
 *"""  
 Исключить информацию о контрольных битах из блока бинарных данных  
 """* clean\_value\_bin = ''  
 for index, char\_bin in enumerate(list(value\_bin), 1):  
 if index not in CHECK\_BITS:  
 clean\_value\_bin += char\_bin  
 return clean\_value\_bin  
  
def set\_errors(encoded):  
 *"""  
 Допустить ошибку в блоках бинарных данных  
 """* result = ''  
 for chunk in chunk\_iterator(encoded, CHUNK\_LENGTH + len(CHECK\_BITS)):  
 num\_bit = random.randint(1, len(chunk))  
 chunk = '{0}{1}{2}'.format(chunk[:num\_bit - 1], int(chunk[num\_bit - 1]) ^ 1, chunk[num\_bit:])  
 result += chunk  
 return result  
  
def check\_and\_fix\_error(encoded\_chunk):  
 *"""  
 Проверка и исправление ошибки в блоке бинарных данных  
 """* check\_bits\_encoded = get\_check\_bits(encoded\_chunk)  
 check\_item = exclude\_check\_bits(encoded\_chunk)  
 check\_item = set\_check\_bits(check\_item)  
 check\_bits = get\_check\_bits(check\_item)  
 if check\_bits\_encoded != check\_bits:  
 invalid\_bits = []  
 for check\_bit\_encoded, value in check\_bits\_encoded.items():  
 if check\_bits[check\_bit\_encoded] != value:  
 invalid\_bits.append(check\_bit\_encoded)  
 num\_bit = sum(invalid\_bits)  
 encoded\_chunk = '{0}{1}{2}'.format(encoded\_chunk[:num\_bit - 1], int(encoded\_chunk[num\_bit - 1]) ^ 1, encoded\_chunk[num\_bit:])  
 return encoded\_chunk  
  
def get\_diff\_index\_list(value\_bin1, value\_bin2):  
 *"""  
 Получить список индексов различающихся битов  
 """* diff\_index\_list = []  
 for index, char\_bin\_items in enumerate(zip(list(value\_bin1), list(value\_bin2)), 1):  
 if char\_bin\_items[0] != char\_bin\_items[1]:  
 diff\_index\_list.append(index)  
 return diff\_index\_list  
  
def encode(source):  
 *"""  
 Кодирование данных  
 """* text\_bin = chars\_to\_bin(source)  
 result = ''  
 for chunk\_bin in chunk\_iterator(text\_bin):  
 chunk\_bin = set\_check\_bits(chunk\_bin)  
 result += chunk\_bin  
 return result  
  
def decode(encoded, fix\_errors=True):  
 *"""  
 Декодирование данных  
 """* decoded\_value = ''  
 fixed\_encoded\_list = []  
 for encoded\_chunk in chunk\_iterator(encoded, CHUNK\_LENGTH + len(CHECK\_BITS)):  
 if fix\_errors:  
 encoded\_chunk = check\_and\_fix\_error(encoded\_chunk)  
 fixed\_encoded\_list.append(encoded\_chunk)  
 clean\_chunk\_list = []  
 for encoded\_chunk in fixed\_encoded\_list:  
 encoded\_chunk = exclude\_check\_bits(encoded\_chunk)  
 clean\_chunk\_list.append(encoded\_chunk)  
 for clean\_chunk in clean\_chunk\_list:  
 for clean\_char in [clean\_chunk[i:i + 8] for i in range(0, len(clean\_chunk), 8)]:  
 decoded\_value += chr(int(clean\_char, 2))  
 return decoded\_value  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 source = input('Укажите текст для кодирования/декодирования:')  
 print('Длина блока кодирования: {0}'.format(CHUNK\_LENGTH))  
 print('Контрольные биты: {0}'.format(CHECK\_BITS))  
 encoded = encode(source)  
 print('Закодированные данные: {0}'.format(encoded))  
 decoded = decode(encoded)  
 print('Результат декодирования: {0}'.format(decoded))  
 encoded\_with\_error = set\_errors(encoded)  
 print('Допускаем ошибки в закодированных данных: {0}'.format(encoded\_with\_error))  
 diff\_index\_list = get\_diff\_index\_list(encoded, encoded\_with\_error)  
 print('Допущены ошибки в битах: {0}'.format(diff\_index\_list))  
 decoded = decode(encoded\_with\_error, fix\_errors=False)  
 print('Результат декодирования ошибочных данных без исправления ошибок: {0}'.format(decoded))  
 decoded = decode(encoded\_with\_error)  
 print('Результат декодирования ошибочных данных с исправлением ошибок: {0}'.format(decoded))

Для сообщения длинной 18 бит:

Укажите текст для кодирования/декодирования:111000110000100100  
Длина блока кодирования: 0  
Контрольные биты: [1, 2, 4, 8]  
Закодированные данные: 100101110001100101110001100101110001100001100000100001100000100001100000100101110001100101110001100001100000100001100000100001100000100001100000100101110001100001100000100001100000100101110001100001100000100001100000  
Результат декодирования: 111000110000100100

Для сообщения длинной 48 бит:

Укажите текст для кодирования/декодирования:110111000100101010100000111011000001000011100110  
Длина блока кодирования: 0  
Контрольные биты: [1, 2, 4, 8]  
Закодированные данные: 100101110001100101110001100001100000100101110001100101110001100101110001100001100000100001100000100001100000100101110001100001100000100001100000100101110001100001100000100101110001100001100000100101110001100001100000100101110001100001100000100001100000100001100000100001100000100001100000100101110001100101110001100101110001100001100000100101110001100101110001100001100000100001100000100001100000100001100000100001100000100101110001100001100000100001100000100001100000100001100000100101110001100101110001100101110001100001100000100001100000100101110001100101110001100001100000  
Результат декодирования: 110111000100101010100000111011000001000011100110

**Задание 2**

Внести одиночную ошибку и устранить ее, используя механизм восстановления.

Для 18 бит:

Допускаем ошибки в закодированных данных: 100101100001100101010001100101111001100001000000100001000000100001100001100101110101100101100001100011100000100001101000100001100100100001110000100111110001000001100000100001100010000101110001110001100000110001100000  
Допущены ошибки в битах: [8, 19, 33, 43, 55, 72, 82, 92, 101, 117, 130, 140, 149, 157, 179, 181, 194, 206]  
Результат декодирования ошибочных данных без исправления ошибок: 1!9 151p840q02100  
Результат декодирования ошибочных данных с исправлением ошибок: 111000110000100100

Для 48 бит:

Допускаем ошибки в закодированных данных: 100100110001100101111001101001100000100100110001101101110001100101110011110001100000100011100000000001100000100001110001000001100000100001101000000101110001100001100100100100110001100101100000100111110001100001110000101101110001100011100000100001100010100001110000000001100000100001101000100101110011100101110011100101110101000001100000100101111001100111110001110001100000100000100000110001100000100001100001100000100000100101110101101001100000100101100000100001110000100001000000101101110001110101110001101101110001100001110000100101100000100101110101100101010001100001110000  
Допущены ошибки в битах: [6, 21, 27, 42, 51, 71, 74, 89, 97, 112, 121, 141, 145, 166, 174, 184, 197, 212, 219, 233, 251, 260, 265, 285, 299, 311, 322, 325, 345, 353, 362, 378, 386, 408, 414, 430, 435, 448, 464, 475, 483, 494, 507, 524, 532, 550, 559, 572]  
Результат декодирования ошибочных данных без исправления ошибок: 9°±30p0108140q0±p200833509q0015°00 ±1±005!0  
Результат декодирования ошибочных данных с исправлением ошибок: 110111000100101010100000111011000001000011100110

**Задание 3**

Рассмотреть вариант кодирования сообщения из 48 бит с предварительным сегментированием на блоки (размерность блоков выбрать самостоятельно).

Длинна блока равна 8:

Укажите текст для кодирования/декодирования:001011111010100100111100111000010101001110101101  
Длина блока кодирования: 8  
Контрольные биты: [1, 2, 4, 8]  
Закодированные данные: 100001100000100001100000100101110001100001100000100101110001100101110001100101110001100101110001100101110001100001100000100101110001100001100000100101110001100001100000100001100000100101110001100001100000100001100000100101110001100101110001100101110001100101110001100001100000100001100000100101110001100101110001100101110001100001100000100001100000100001100000100001100000100101110001100001100000100101110001100001100000100101110001100001100000100001100000100101110001100101110001100101110001100001100000100101110001100001100000100101110001100101110001100001100000100101110001  
Результат декодирования: 001011111010100100111100111000010101001110101101  
Допускаем ошибки в закодированных данных: 100001100001100001100001100101100001100001100010100100110001100101010001100001110001100101010001100101110000100011100000100001110001100011100000110101110001100001100001100000100000000101110001100001000000000001100000100101111001101101110001101101110001000101110001100001110000100001101000100101010001100101010001110101110001101001100000110001100000100001100001100001101000100001110001100001100010100101111001100001100100110101110001110001100000000001100000100101100001110101110001000101110001000001100000100100110001100001100100100101110101100101010001100001100001110101110001  
Допущены ошибки в битах: [12, 24, 32, 47, 54, 67, 76, 91, 108, 113, 124, 137, 146, 168, 174, 181, 199, 205, 225, 231, 243, 253, 272, 285, 295, 307, 314, 327, 338, 360, 369, 376, 395, 405, 418, 422, 434, 445, 464, 470, 481, 493, 510, 526, 538, 547, 564, 566]  
Результат декодирования ошибочных данных без исправления ошибок: 1112!1!0p1p111 09±±108!!1°0181294100111045!11  
Результат декодирования ошибочных данных с исправлением ошибок: 001011111010100100111100111000010101001110101101

**Задание 4**

Сравнить режимы кодирования с сегментацией и без.

- Режим кодирования Хэмминга без сегментации предоставляет простой и надежный способ обнаружения и исправления ошибок в одном блоке данных.

- Режим кодирования Хэмминга с сегментацией улучшает способность обработки больших объемов данных, обеспечивая более надежную коррекцию ошибок при передаче информации по частям.

- Выбор между этими режимами зависит от потребностей конкретной системы передачи данных и требований к эффективности исправления ошибок.

**Вывод:** В ходе выполнения лабораторной работы были изучены и сравнены два режима кодирования методом Хэмминга: с сегментацией данных и без неё. Кодирование Хэмминга без сегментации обеспечивает простой и надежный способ обнаружения и исправления ошибок в пределах одного блока данных, но ограничено в способности исправлять ошибки и неэффективно для больших объемов данных. В отличие от этого, кодирование Хэмминга с сегментацией улучшает обработку больших объемов данных и обеспечивает более надежную коррекцию ошибок при передаче информации частями. Выбор между этими методами зависит от потребностей конкретной системы передачи данных и требований к эффективности исправления ошибок.