МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»

(БГТУ им. В. Г. ШУХОВА)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа № 8

По дисциплине: Теория информации

Тема: «Исследование помехоустойчивых кодов на примере алгоритма Хэмминга»

Выполнил: ст. группы ПВ-211
Чувилко Илья Романович
Проверил:
Твердохлеб Виталий Викторович

Белгород 2023 г.

Цель работы: исследование помехоустойчивых кодов на примере алгоритма Хэмминга.

Ход работы

1. Закодировать по Хэммингу произвольно сформированные последовательности двоичных символов длиной:

- 18 бит;

- 48 бит.

Сделать выводы относительно эффективности каждого их сообщений.

Реализация функции кодирования по Хэммингу:

```
def generate_hamming_code(message):
    # Определение количества проверочных битов
    r = 0

    while 2 ** r < len(message) + r + 1:
        r += 1

    # Создание закодированного сообщения с заполнением
проверочных битов нулями
    encoded_message = [0] * (len(message) + r)
    encoded_message[r:] = message

    # Вычисление значений проверочных битов
    for i in range(r):
        parity = 0
        for j in range(1, len(encoded_message) + 1):
            if (j >> i) & 1:
                parity ^= encoded_message[-j]
        encoded_message[2 ** i - 1] = parity

    return encoded_message



# Задача 1: Закодировать исходные сообщения
message_18 = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
0]
encoded_18 = generate_hamming_code(message_18)
print("Закодированное сообщение (18 бит):", encoded_18)
message_48 = [1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0,
1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0]
encoded_48 = generate_hamming_code(message_48)
print("Закодированное сообщение (48 бит):", encoded_48)
```

Программный вывод

```
Закодированное сообщение (18 бит): [0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]
Закодированное сообщение (48 бит): [0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0,
```

2. Внести одиночную ошибку и устранить ее, используя механизм восстановления.

Реализация функции для восстановления сообщения:

```
def decode_hamming_code(encoded_message):
    r = 0

    while 2 ** r < len(encoded_message):
        r += 1

    # Поиск и исправление ошибок
    error_index = 0

    for i in range(r):
        parity = 0

        for j in range(1, len(encoded_message) + 1):
            if (j >> i) & 1:
                parity ^= encoded_message[-j]

        error_index += parity << i

    if error_index:
        encoded_message[-error_index] = int(not
encoded_message[-error_index])

    # Извлечение исходного сообщения
    decoded_message = encoded_message[::2]

    return decoded_message



# Задача 2: Внести и исправить ошибку
encoded_with_error = list(encoded_18)  # Выбираем закодированное
сообщение для внесения ошибки
encoded_with_error[5] = 1  # Вносим ошибку
decoded_with_error = decode_hamming_code(encoded_with_error)
```

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

Функция для реализации кодирования с сегментацией:

```
def encode_segmented_hamming_code(message, block_size):
    encoded_message = []
```

```
    for i in range(0, len(message), block_size):
        block = message[i:i+block_size]
        encoded_block = generate_hamming_code(block)
        encoded_message.extend(encoded_block)

    return encoded_message



# Задача 3: Кодирование сообщения из 48 бит с предварительным
# сегментированием на блоки
message_48 = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
1, 0, 1, 0, 1, 0, 1, 0, 1, 0]
block_size = 8
encoded_segmented = encode_segmented_hamming_code(message_48,
block_size)
print("Закодированное сообщение с сегментацией (48 бит):",
encoded_segmented)

# Задача 4: Сравнение режимов кодирования
print("Исходное сообщение (48 бит):", message_48)
print("Закодированное сообщение (48 бит):", encoded_48)
print("Закодированное сообщение с сегментацией (48 бит):",
encoded_segmented)

# Сравнение размеров закодированных данных
print("Размер исходного сообщения (48 бит):", len(message_48))
print("Размер закодированного сообщения (48 бит)):",
len(encoded_48))
print("Размер закодированного сообщения с сегментацией (48
бит):", len(encoded_segmented))

# Сравнение степени сжатия
compression_ratio = len(message_48) / len(encoded_48)
segmented_compression_ratio = len(message_48) /
len(encoded_segmented)
print("Коэффициент сжатия (без сегментации):",
compression_ratio)
print("Коэффициент сжатия (с сегментацией):",
segmented_compression_ratio)
```

Программный вывод:

```
Закодированное сообщение с сегментацией (48 бит): [0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
Исходное сообщение (48 бит): [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
Закодированное сообщение (48 бит): [0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0,
Закодированное сообщение с сегментацией (48 бит): [0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
Размер исходного сообщения (48 бит): 48
Размер закодированного сообщения (48 бит)): 56
Размер закодированного сообщения с сегментацией (48 бит): 72
Коэффициент сжатия (без сегментации): 0.8571428571428571
Коэффициент сжатия (с сегментацией): 0.6666666666666666
```

4. Сравнить режимы кодирования с сегментацией и без.

• Режим кодирования без сегментации позволяет закодировать исходное сообщение из 48 бит в закодированное сообщение такой же длины (48 бит).

• Режим кодирования с сегментацией позволяет разделить исходное сообщение на блоки и закодировать каждый блок отдельно. Закодированное сообщение с сегментацией имеет больший размер (больше 48 бит), но при этом можно выбрать размер блоков и добиться более эффективной сжатия в зависимости от конкретных данных.

• Сравнение степени сжатия показывает, что режим кодирования без сегментации обеспечивает лучшую степень сжатия, так как размер закодированного сообщения остается таким же как и исходного. Однако режим кодирования с сегментацией может быть полезен в случаях, когда есть определенная структура в исходных данных и можно достичь большей степени сжатия за счет выбора оптимального размера блоков.

Вывод: Таким образом, на этом лабораторном занятии я исследовал помехоустойчивые коды на примере алгоритма Хэмминга. Так же, научился программно реализовывать алгоритм кодирования по Хэммингу.