Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный технический университет”
Кафедра интеллектуально-информационных технологий

Лабораторная работа №2
“Избыточное кодирование данных в информационных системах. Итеративные коды”

Выполнил:
студент 4 курса
группы ИИ-22
Сидоренко А.А.
Проверила:
Хацкевич А. С.

Брест 2024

Цель работы: приобретение практических навыков кодирования/декодирования двоичных данных при использовании итеративных кодов.

Ход работы:

Задание

Разработать собственное приложение, которое позволяет выполнять следующие операции:

1) вписывать произвольное двоичное представление информационного слова Xk (кодируемой информации) длиной k битов в двумерную матрицу размерностью в соответствии с вариантом;

2) вычислить проверочные биты (биты паритетов): а) по двум; б) по трем; в) по четырем направлениям (группам паритетов);

3) формировать кодовое слово Xn присоединением избыточных символов к информационному слову;

4) генерировать ошибку произвольной кратности (i, i > 0), распределенную случайным образом среди символов слова Xn, в результате чего формируется кодовое слово Yn;

5) определять местоположение ошибочных символов итеративным кодом в слове Yn в соответствии с используемыми группами паритетов и исправлять ошибочные символы (результат исправления - слово Yn');

6) выполнять анализ корректирующей способности используемого кода (количественная оценка) путем сравнения соответствующих слов Xn и Yn'; результат анализа может быть представлен в виде отношения общего числа сгенерированных кодовых слов с ошибками определенной одинаковой кратности (с одной ошибкой, с двумя ошибками и т. д.) к числу кодовых слов, содержащих ошибки этой кратности, которые правильно обнаружены и которые правильно скорректированы.

Вариант 2

Варианты заданий

Вариант	Длина информационного слова (бит), k	k_1	k_2	z	Количество групп паритетов
1	16	4 8 4 2	4 2 2 4	– - 2 2	2; 3 2; 3 2; 3; 4; 5 2; 3; 4; 5
2	20	4 2 2 2	5 10 5 2	- - 2 5	2; 3 2; 3 2; 3; 4; 5 2; 3; 4; 5

Код программы без применения:

```
import numpy as np


class Code:
    def __init__(self, k1, k2, z, parity_groups):
        self.k1 = k1
        self.k2 = k2
        self.z = z
        self.parity_groups = parity_groups  #

    def encode(self, message):

        message_bits = np.array([int(bit) for bit in message])

        if len(message_bits) != 20:
            raise ValueError("Длина сообщения должна быть 20 бит.")

        parity_bits = []
        for group in self.parity_groups:
            parity = 0
            for i in range(0, len(message_bits), group):
                parity ^= message_bits[i:i + group].sum() % 2
            parity_bits.append(parity)

        coded_message = np.concatenate((message_bits, parity_bits))
        return coded_message

    def decode(self, received):

        message_bits = received[:20]
        received_parity_bits = received[20:]

        error_detected = False
        for i, group in enumerate(self.parity_groups):
            expected_parity = 0
            for j in range(0, len(message_bits), group):
                expected_parity ^= message_bits[j:j + group].sum() % 2

            if received_parity_bits[i] != expected_parity:
                error_detected = True
                print(f"Ошибка в группе паритета {i + 1}")

        if not error_detected:
            print("Ошибок не обнаружено.")
        else:
            print("Обнаружены ошибки в принятых данных.")

        return message_bits


k1 = 4
k2 = 2
z = 10
parity_groups = [2, 3, 4, 5]

codec = Code(k1, k2, z, parity_groups)

message = "11010111011101010101"

encoded_message = codec.encode(message)
print("Закодированное сообщение:", encoded_message)
```

```
received_message = np.copy(encoded_message)
received_message[3] = (received_message[3] + 1) % 2

decoded_message = codec.decode(received_message)
print("Декодированное сообщение:", decoded_message)
```

Вывод программы:

```
C:\Users\Xiaomi\AppData\Local\Programs\Python\Python312\python.exe C:\Users\Xiaomi\Desktop\prjs\SMZKS-
Закодированное сообщение: [1 1 0 1 0 1 1 1 0 1 1 1 0 1 0 1 0 1 0 1 1 1 1 1]
Ошибка в группе паритета 1
Ошибка в группе паритета 2
Ошибка в группе паритета 3
Ошибка в группе паритета 4
Обнаружены ошибки в принятых данных.
Декодированное сообщение: [1 1 0 0 0 1 1 1 0 1 1 1 0 1 0 1 0 1 0 1]
```