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

Лабораторная работа №1
“Избыточное кодирование данных в информационных системах. Код Хемминга”

Выполнил:
студент 4 курса
группы ИИ-22
Клебанович В. Н.
Проверила:
Хацкевич А. С.

Брест 2024

Цель работы: приобретение практических навыков кодирования/декодирования двоичных данных при использовании кода Хемминга.

Ход работы

Вариант 7

1. Составить код Хемминга (классический алгоритм) (M + r, M), допустить ошибку в одном из разрядов и отыскать ее по алгоритму.
2. Составить код Хемминга (расширенный алгоритм) (первые 7 битов M + 3 проверочных, первые 7 битов M), допустить 2 или более ошибок в разрядах и отыскать их по алгоритму.

Для данного задания M = 611, r = 4.

Код программы:

```
binary_str = "1001100011"

binary_list = list(binary_str)

binary_list.insert(0, '0')
binary_list.insert(1, '0')
binary_list.insert(3, '0')
binary_list.insert(7, '0')

print("Число после добавления 0: ", ''.join(map(str, binary_list)), "\n")

binary_list = [int(x) for x in binary_list]

def calculate_control_bit(binary_list, indices, control_bit_index):
    control_sum = sum(binary_list[i] for i in indices)

    if control_sum % 2 == 0:
        binary_list[control_bit_index] = 0
        print(f"Сумма контрольных битов для индексов {[i + 1 for i in indices]}:
{control_sum}. Контрольный бит для данных индексов равен 0")
    else:
        binary_list[control_bit_index] = 1
        print(f"Сумма контрольных битов для индексов {[i + 1 for i in indices]}:
{control_sum}. Контрольный бит для данных индексов равен 1")

calculate_control_bit(binary_list, [0, 2, 4, 6, 8, 10, 12], 0)

calculate_control_bit(binary_list, [1, 2, 5, 6, 9, 10, 13], 1)

calculate_control_bit(binary_list, [3, 4, 5, 6, 11, 12, 13], 3)
```

```
calculate_control_bit(binary_list, [7, 8, 9, 10, 11, 12, 13], 7)

new_binary_str = ''.join(map(str, binary_list))

print("\nЧисло с контрольными битами:", new_binary_str)

error = int(input("\nВведите номер разряда, в котором будет допущена ошибка: "))

if binary_list[error - 1] == 1:
    binary_list[error - 1] = 0
else:
    binary_list[error - 1] = 1

binary_str_with_error = ''.join(map(str, binary_list))

print("Число с ошибочным битом:", binary_str_with_error)

calculate_control_bit(binary_list, [0, 2, 4, 6, 8, 10, 12], 0)

calculate_control_bit(binary_list, [1, 2, 5, 6, 9, 10, 13], 1)

calculate_control_bit(binary_list, [3, 4, 5, 6, 11, 12, 13], 3)

calculate_control_bit(binary_list, [7, 8, 9, 10, 11, 12, 13], 7)

positions = [7, 3, 1, 0]
selected_bits = [binary_list[i] for i in positions]
print("Данные 4 бита", selected_bits, "создают число в двоичном виде, которое
равно", error)
```

Вывод программы:

```
Число после добавления 0:  00100010100011

Сумма контрольных битов для индексов [1, 3, 5, 7, 9, 11, 13]: 4. Контрольный бит для данных индексов равен 0
Сумма контрольных битов для индексов [2, 3, 6, 7, 10, 11, 14]: 3. Контрольный бит для данных индексов равен 1
Сумма контрольных битов для индексов [4, 5, 6, 7, 12, 13, 14]: 3. Контрольный бит для данных индексов равен 1
Сумма контрольных битов для индексов [8, 9, 10, 11, 12, 13, 14]: 3. Контрольный бит для данных индексов равен 1

Число с контрольными битами: 01110011100011

Введите номер разряда, в котором будет допущена ошибка: 11
Число с ошибочным битом: 01110011101011
Сумма контрольных битов для индексов [1, 3, 5, 7, 9, 11, 13]: 5. Контрольный бит для данных индексов равен 1
Сумма контрольных битов для индексов [2, 3, 6, 7, 10, 11, 14]: 5. Контрольный бит для данных индексов равен 1
Сумма контрольных битов для индексов [4, 5, 6, 7, 12, 13, 14]: 4. Контрольный бит для данных индексов равен 0
Сумма контрольных битов для индексов [8, 9, 10, 11, 12, 13, 14]: 5. Контрольный бит для данных индексов равен 1
Данные 4 бита [1, 0, 1, 1] создают число в двоичном виде, которое равно 11
```

Вывод: приобрел практические навыки кодирования/декодирования двоичных данных при использовании кода Хемминга.