Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №7
дисциплины «Алгоритмизация»

Выполнила:
Михеева Елена Александровна
2 курс, группа ИВТ-б-з-20-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной
техники и автоматизированных
систем», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А...,

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: Кодировка Хаффмана

Порядок выполнения работы.

1. Была написана программа для реализации алгоритма Хаффмана для кодирования строки. На вход программа принимает строку, затем идет подсчет частоты встречающихся букв. Далее запускается procedurehuffman(), которая принимает словарь с частотами и на его основе строит дерево Хаффмана(символы с низкой частотой идут в «глубину» дерева, а символы с высокой располагаются ближе к корню). Следующим этапом идет функция huffman_codes() – она рекурсивно обходит дерево и генерирует для каждого символа двоичный код. Сгенерированные коды хранятся в словаре “codes”. Для кодировки и декодировки используются функции huffman_encode() и huffman_decode() соответственно.

```python
import heapq


def procedurehuffman(f):
    heap = []
    buffer_fs = set()

    for symbol, frequency in f.items():
        heapq.heappush(heap, (frequency, symbol))
    while len(heap) > 1:
        f1, i = heapq.heappop(heap)
        f2, j = heapq.heappop(heap)
        fs = f1 + f2
        ord_val = ord('a')
        fl = str(fs)
        while fl in buffer_fs:
            symb = chr(ord_val)
            fl = str(fs) + " " + symb
            ord_val += 1
        buffer_fs.add(fl)
        f[fl] = {f"{x}": f[x] for x in [i, j]}
        del f[i], f[j]
        heapq.heappush(heap, (fs, fl))
    return f


def huffman_codes(tree, codes, path=''):
    for i, (node, child) in enumerate(tree.items()):
        # Если child - это лист, добавляем его в код
        if isinstance(child, int):
            codes[node] = path[1:] + str(abs(i-1))
        else:
            # Если child - это словарь, продолжаем рекурсивно
            huffman_codes(child, codes, path + str(abs(i-1)))
    return codes


def huffman_encode(data, codes):
    encoded_data = ''.join(codes[char] for char in data)
    return encoded_data


def huffman_decode(encoded_data, codes):
    decoded_data = ''
    current_code = ''
    for bit in encoded_data:
        current_code += bit
        for char, code in codes.items():
            if code == current_code:
                decoded_data += char
                current_code = ''
                break
    return decoded_data


if __name__ == "__main__":
    input_string = "Hello Word!"
    frequency = {}
    # Создание словаря с частотами букв
    for char in input_string:
        if char in frequency:
            frequency[char] += 1
        else:
            frequency[char] = 1
    for char, count in frequency.items():
        print(f"'{char}': {count} раз")

tree = procedurehuffman(frequency)
code = huffman_codes(tree, dict())
encoded_data = huffman_encode(input_string, code)
decoded_data = huffman_decode(encoded_data, code)

print(f"Encoded: {encoded_data}")
print(f"Decoded: {decoded_data}")
```

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ                Python Debug Console

'H': 1 раз
'e': 1 раз
'l': 2 раз
'o': 2 раз
' ': 1 раз
'W': 1 раз
'r': 1 раз
'd': 1 раз
'!': 1 раз
{'11': {'4 a': {'l': 2, 'o': 2}, '7': {'3': {'r': 1, '2': {' ': 1, '!': 1}}, '4': {'2 a': {'H': 1, 'W': 1}, '2 b': {'d': 1, 'e': 1}}}}}
code {'l': '11', 'o': '10', 'r': '011', ' ': '0101', '!': '0100', 'H': '0011', 'W': '0010', 'd': '0001', 'e': '0000'}
Original Data: Hello Word!
Encoded Data: 001100001111100101001010011000010100
Decoded Data: Hello Word!
```

Рисунок 1. Программа для кодировки Хаффмана