

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»
Вариант 29

Выполнил:
Саенко Андрей Максимович
2 курс, группа ИВТ-б-о-22-1,
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., канд. технических
наук, доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____
Ставрополь, 2023 г.

Тема: алгоритм Хаффмана для кодирования и декодирования текста

Порядок выполнения работы:

1. Реализован алгоритм Хаффмана для кодирования и декодирования текста. Суть алгоритма заключается в следующем: для каждого символа подсчитывается количество вхождений в строку, после чего на основе полученных данных создаётся бинарное дерево, количество узлов которого равно количеству разных символов в строке. Первый узел формируется из двух символов с наименьшей частотой появления, после чего алгоритм создаёт новые узлы на основе объединения двух узлов с наименьшим весом, а когда остаётся один узел, алгоритм создания дерева останавливается. Код каждого символа отражает путь от корня дерева до этого символа в бинарном дереве. Известно, что алгоритм Хаффмана даёт оптимальный префиксный код.

Код программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from heapq import heappush, heappop

def add_tree(f):
    heap = []
    used = set()

    for i in f:
        heappush(heap, (f[i], i))

    while len(heap) > 1:
        f_first, i = heappop(heap)
        f_second, j = heappop(heap)
        fs = f_first + f_second
        ord_val = ord('a')
        fl = str(fs)

        while fl in used:
            letter = chr(ord_val)
            fl = str(fs) + " " + letter
            ord_val += 1

        used.add(fl)
        f[fl] = {f"{x}": f[x] for x in [i, j]}
```

```

        del f[i], f[j]
        heappush(heap, (fs, fl))

    return f

def add_dict(tree, codes, p=""):
    for i, (node, child) in enumerate(tree.items()):
        if isinstance(child, int):
            codes[node] = p[1:] + str(abs(i-1))
        else:
            add_dict(child, codes, p + str(abs(i-1)))

    return codes

def code(sentence, dictionary):
    replaced_s = ""

    for char in sentence:
        if char in dictionary:
            replaced_s += dictionary[char]
        else:
            replaced_s += char

    return replaced_s

def decode(encoded_text, tree):
    decoded_text = ""
    key = list(tree.keys())[0]
    temp = tree[key]

    for bit in encoded_text:
        for i, (node, child) in enumerate(temp.items()):
            if str(i) != bit:
                if isinstance(child, int):
                    decoded_text += node
                    temp = tree[key]
                    break

            temp = child
        break

    return decoded_text

def count_chars(s):
    chars = { }

    for char in s:
        if char in chars:
            chars[char] += 1

```

```

else:
    chars[char] = 1

return chars

if __name__ == "__main__":
    text = input("Введите текст: \n")

    chars = count_chars(text)
    for char, count in chars.items():
        print(f'Символ '{char}' встречается {count} раз(a)')

    tree = add_tree(chars)
    print(f'Дерево: {tree}')

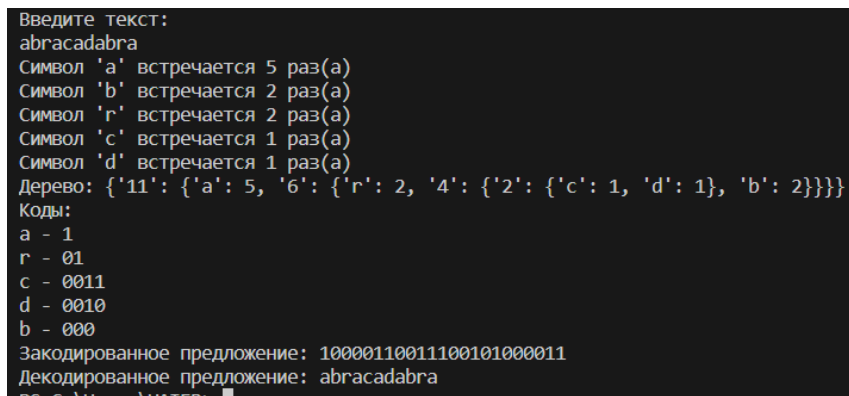
    codes = add_dict(tree, dict())
    print("Коды:")
    for i in codes:
        print(f'{i} - {codes[i]}')

    coding_s = str(code(text, codes))
    print(f'Закодированное предложение: {coding_s}')

    decoded_text = decode(coding_s, tree)
    print(f'Декодированное предложение: {decoded_text}')

```

Результат работы программы:



```

Введите текст:
abracadabra
Символ 'a' встречается 5 раз(a)
Символ 'b' встречается 2 раз(a)
Символ 'r' встречается 2 раз(a)
Символ 'c' встречается 1 раз(a)
Символ 'd' встречается 1 раз(a)
Дерево: {'11': {'a': 5, '6': {'r': 2, '4': {'2': {'c': 1, 'd': 1}, 'b': 2}}}}
Коды:
a - 1
r - 01
c - 0011
d - 0010
b - 000
Закодированное предложение: 10000110011100101000011
Декодированное предложение: abracadabra

```

Рисунок 1 – Результат работы программы

Вывод

В ходе выполнения работы изучен принцип работы алгоритма Хаффмана, который является алгоритмом оптимального префиксного кодирования алфавита.