

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

**Отчет по лабораторной работе
Разбор задач на объектно-ориентированного программирования в языке
Python.**

по дисциплине «Объектно-ориентированное программирование»

Выполнила студентка группы ИВТ-б-о-20-1

Новикова В.С. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____

(подпись)

Ставрополь 2022

Цель работы: приобретение навыков по работе с жадными алгоритмами с помощью языка программирования Python версии 3х.

Ход работы:

Ссылка на репозиторий: https://github.com/Valentina1502/OOP_tasks

Задание 1.

Реализация жадных алгоритмов на примере Задачи о рюкзаке

Код:

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

'''
Реализация задачи про максимальную стоимость предметов рюкзака
'''

def knapsack(bag, goods, weights):
    items = []
    for i, item in enumerate(goods):
        itemInfo = {
            'vpw': goods[i] / weights[i],
            'weight': weights[i]
        }
        if len(items) == 0:
            items.append(itemInfo)
        else:
            k = 0
            while k < len(items) and items[k]['vpw'] > itemInfo['vpw']:
                k += 1
            items.insert(k, itemInfo)
    total = 0
    bag_left = bag
    for item in items:
        if bag_left - item['weight'] >= 0:
            total += item['weight'] * item['vpw']
            bag_left -= item['weight']
        elif bag_left > 0:
            total += item['vpw'] * bag_left
            bag_left = 0
    return total

if __name__ == '__main__':
    bag = int(input("Вместимость рюкзака: "))
    goods = tuple(map(int, input("Введите соответствующую стоимость
предметов: ").split()))
    weights = tuple(map(int, input("Введите соответствующие веса предметов:
").split()))
    print("\nМаксимальная стоимость рюкзака составит: ", knapsack(bag, goods,
weights))
```

```
C:\Users\Valentina\AppData\Local\Programs\Python\Python38\python.exe
Вместимость рюкзака: 25
Введите соответствующую стоимость предметов: 12 9 30
Введите соответствующие веса предметов: 3 2 5

Максимальная стоимость рюкзака составит: 51.0

Process finished with exit code 0
```

Рисунок 1 – Задание 1

Задание 2.

Реализация методов кодирования и декодирования по Хаффману.

Код:

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

'''
Реализация алгоритма Хаффмана
'''

from collections import Counter

class Node:
    def __init__(self, value, left=None, right=None):
        self.right = right
        self.left = left
        self.value = value

def get_code(root, codes=dict(), code=''):
    if root is None:
        return
    if isinstance(root.value, str):
        codes[root.value] = code
        return codes

    get_code(root.left, codes, code + '0')
    get_code(root.right, codes, code + '1')
    return codes

def get_tree(string):
    string_count = Counter(string)

    if len(string_count) <= 1:
        node = Node(None)
        if len(string_count) == 1:
            node.left = Node([key for key in string_count][0])
            node.right = Node(None)
        string_count = {node: 1}

    while len(string_count) != 1:
        node = Node(None)
```

```

        spam = string_count.most_common()[:-3:-1]

        if isinstance(spam[0][0], str):
            node.left = Node(spam[0][0])
        else:
            node.left = spam[0][0]
        if isinstance(spam[1][0], str):
            node.right = Node(spam[1][0])
        else:
            node.right = spam[1][0]

        del string_count[spam[0][0]]
        del string_count[spam[1][0]]
        string_count[node] = spam[0][1] + spam[1][1]
    return [key for key in string_count][0]

def coding(string, codes):
    res = ''
    for symbol in string:
        res += codes[symbol]
    return res

def decoding(string, codes):
    res = ''
    i = 0
    while i < len(string):
        for code in codes:
            if string[i:].find(codes[code]) == 0:
                res += code
                i += len(codes[code])
    return res

if __name__ == '__main__':
    my_string = input('Введите строку: ')
    tree = get_tree(my_string)

    codes = get_code(tree)
    print(f'Шифр: {codes}')

    coding_str = coding(my_string, codes)
    print('Сжатая строка: ', coding_str)

    decoding_str = decoding(coding_str, codes)
    print('Исходная строка: ', decoding_str)

    if my_string == decoding_str:
        print('Успешно!')
    else:
        print('Ошибка!')

```

```

C:\Users\Valentina\AppData\Local\Programs\Python\Python38\python.exe C:/Users/Valentina/Documents/GitHub/pythonProject/zd1.py
Введите строку для сжатия: Алгоритм Хаффмана
Шифр: {'н': '0000', 'х': '0001', 'ф': '001', 'и': '0100', 'р': '0101', ' ': '0110', 'т': '0111', 'л': '1000', 'а': '1001', 'о': '1010'}
Сжатая строка: 1001100010111010010101000111110011000011110010011101110000111
Исходная строка: Алгоритм Хаффмана
Успешно!

Process finished with exit code 0

```

Рисунок 2 – Задание 2

Вывод: при решении практических заданий были приобретены простейшие навыки по работе с жадными алгоритмами в языке программирования Python.