

Практическая работа № 3
студента группы ИТз-221
Дмитриев Дмитрий Анатольевич

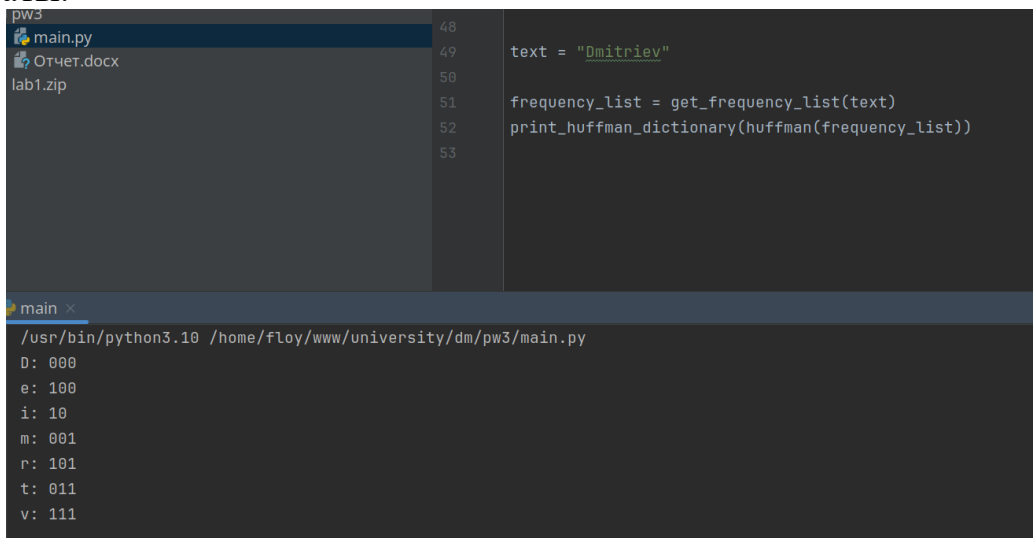
Выполнение:_____ Защита:_____

Алгоритм Хаффмана.

Цель работы: ознакомиться с алгоритмами сжатия данных, научиться реализовывать алгоритм Хаффмана

Содержание работы
Ход работы

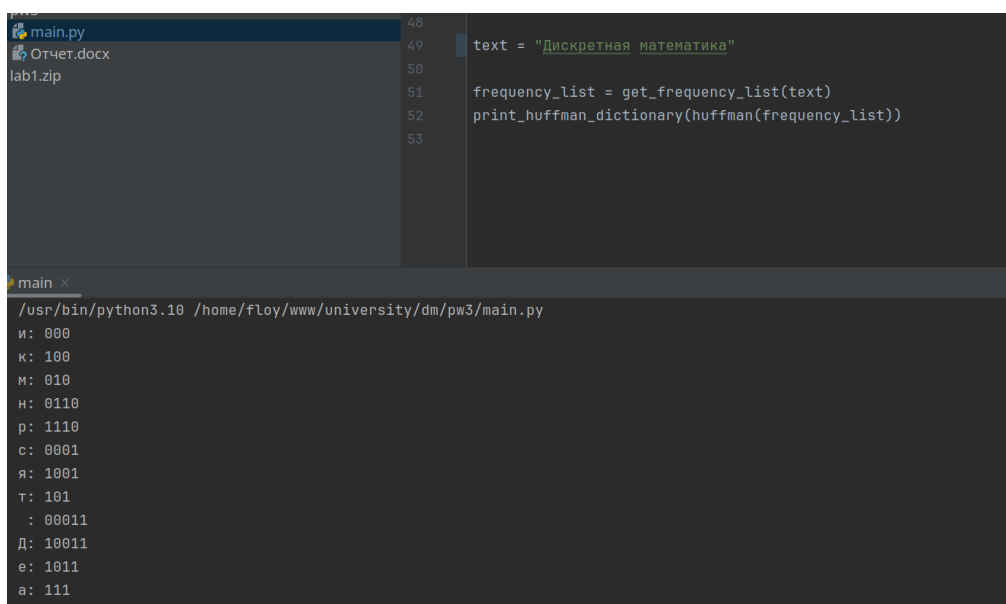
1) Реализовывал алгоритм Хаффмана, протестировал его, записал в отчет результаты.



```
pw3
main.py
Отчет.docx
lab1.zip

48
49 text = "Dmitriev"
50
51 frequency_list = get_frequency_list(text)
52 print_huffman_dictionary(huffman(frequency_list))
53

main x
/usr/bin/python3.10 /home/floy/www/university/dm/pw3/main.py
D: 000
e: 100
i: 10
m: 001
r: 101
t: 011
v: 111
```



```
pw3
main.py
Отчет.docx
lab1.zip

48
49 text = "Дискретная математика"
50
51 frequency_list = get_frequency_list(text)
52 print_huffman_dictionary(huffman(frequency_list))
53

main x
/usr/bin/python3.10 /home/floy/www/university/dm/pw3/main.py
и: 000
к: 100
м: 010
н: 0110
р: 1110
с: 0001
я: 1001
т: 101
: 00011
Д: 10011
е: 1011
а: 111
```

Код программы:

```
import heapq

def huffman(frequencies):
    heap = []
    for sym, freq in frequencies:
        heap.append((freq, [(sym, "")]))
    heapq.heapify(heap)

    while len(heap) > 1:
        low_freq, low_node = heapq.heappop(heap)
        high_freq, high_node = heapq.heappop(heap)

        for i in range(len(low_node)):
            pair_symbol, pair_code = low_node[i]
            low_node[i] = (pair_symbol, pair_code + '0')

        for i in range(len(high_node)):
            pair_symbol, pair_code = high_node[i]
            high_node[i] = (pair_symbol, pair_code + '1')

        merged_node = low_node + high_node
        heapq.heappush(heap, (low_freq + high_freq, merged_node))

    huffman_codes = { }
    for pair in heapq.heappop(heap)[1]:
        symbol, code = pair
        huffman_codes[symbol] = code

    return huffman_codes

def get_frequency_list(text):
    frequency_dict = { }
    for char in text:
        if char in frequency_dict:
            frequency_dict[char] += 1
        else:
            frequency_dict[char] = 1

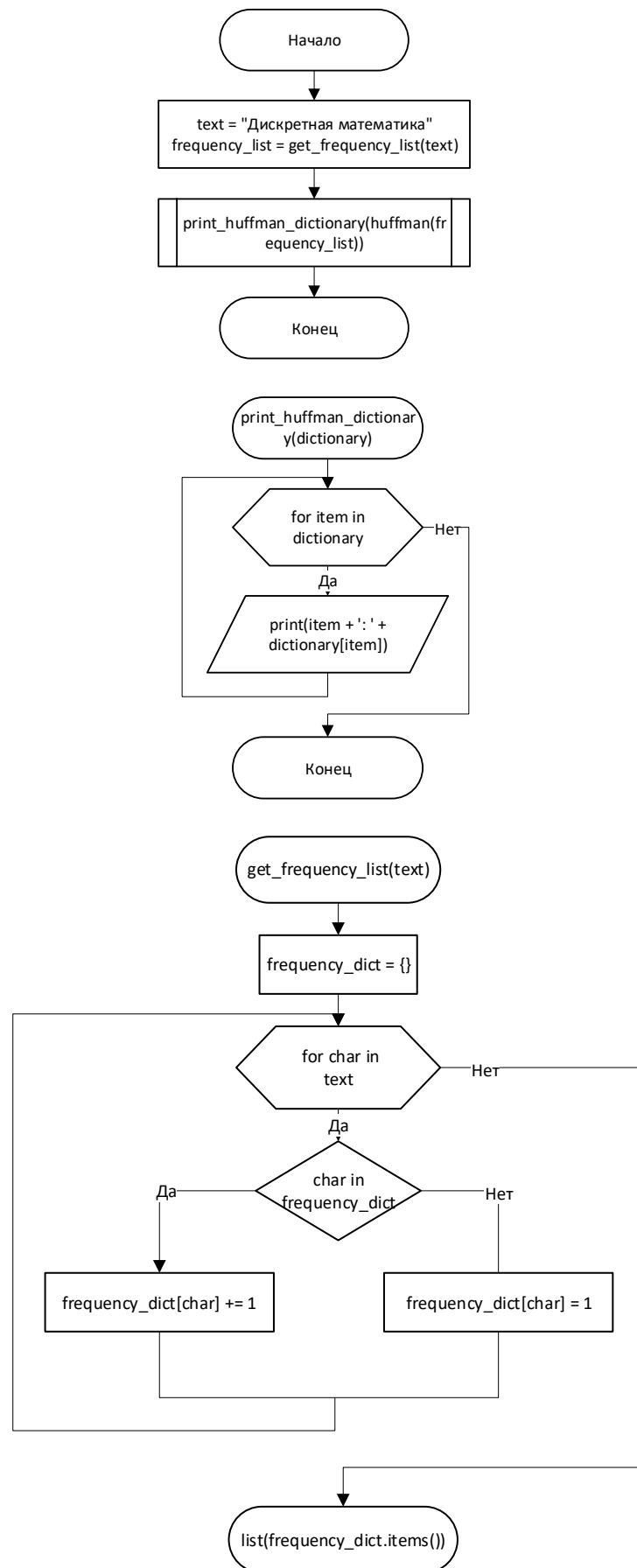
    return list(frequency_dict.items())

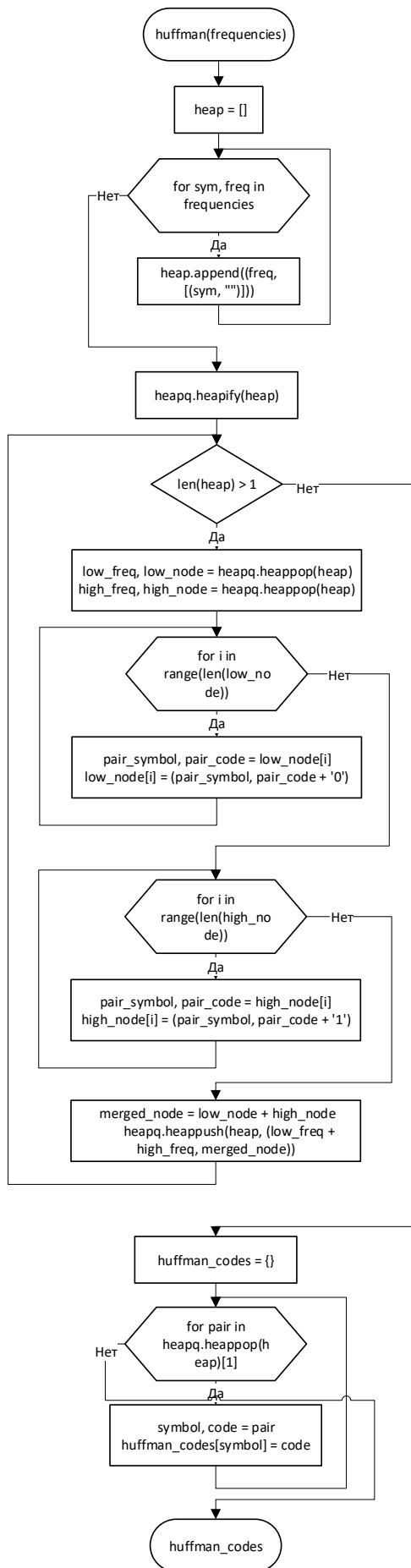
def print_huffman_dictionary(dictionary):
    for item in dictionary:
        print(item + ': ' + dictionary[item])

text = "Дискретная математика"

frequency_list = get_frequency_list(text)
print_huffman_dictionary(huffman(frequency_list))
```

Блок схема:





Вывод: ознакомился с алгоритмами сжатия данных, научился реализовывать алгоритм Хаффмана