Практическая работа № 3

студента группы ИТз-221

Дмитриев Дмитрий Анатольевич

Выполнение:\_\_\_\_\_\_\_\_\_ Защита:\_\_\_\_\_\_\_\_\_\_\_

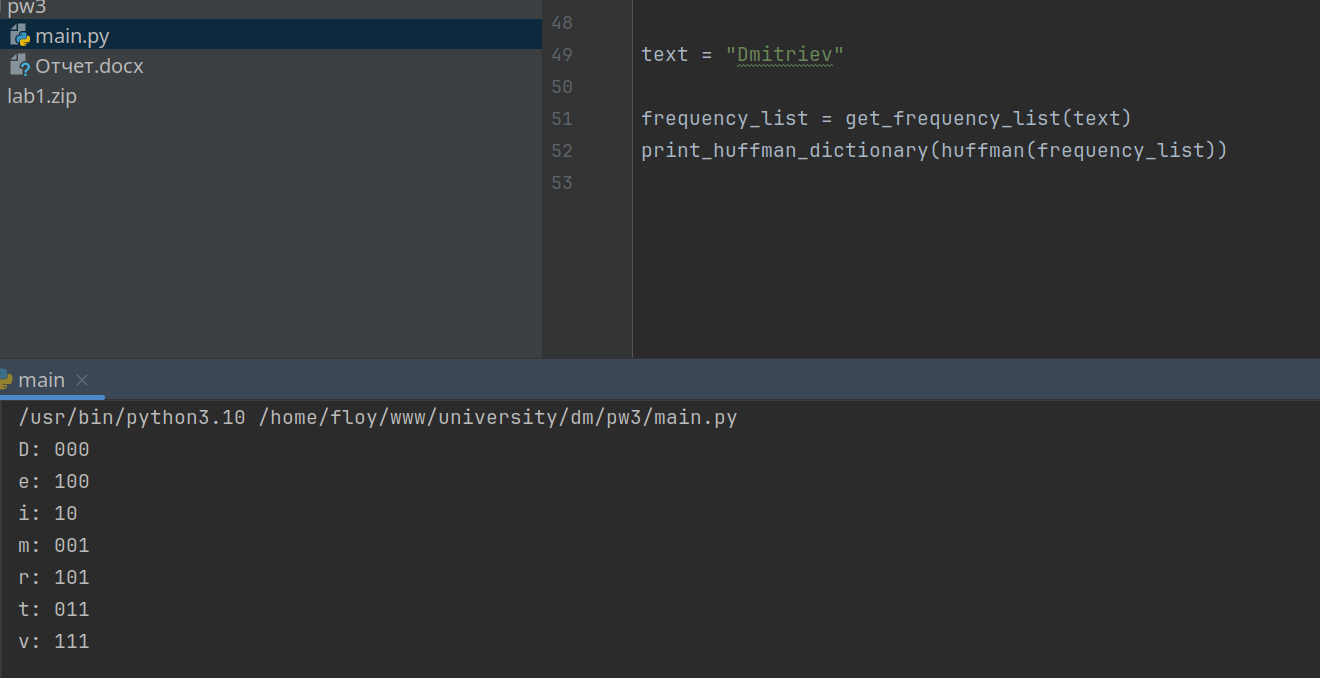
**Алгоритм Хаффмана.**

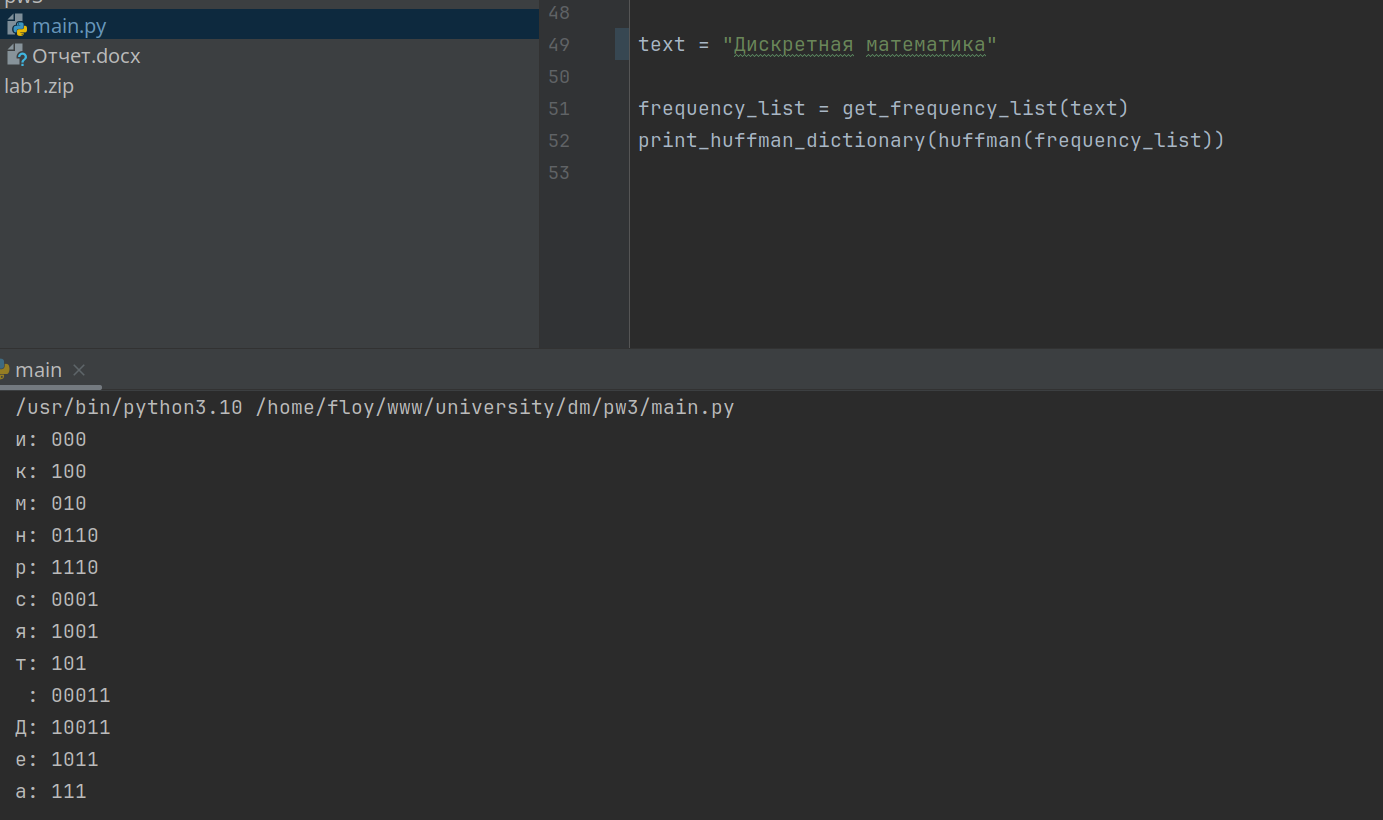
Цель работы: ознакомится с алгоритмами сжатия данных, научится реализовывать алгоритм Хаффмана

Содержание работы

Ход работы

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





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