#### Отчет

#### Лабораторная работа 2 (дополнительное задание)

Исследование статистических характеристик исходных текстов (как бинарных файлов, так и файлов в формате простого текста). Работа с кодовыми таблицами русского языка

Операционная система ОС Windows, 64-разрядная ОС

Задание Л2.№4. Бонус +7 (добавляется к л/р, а не к общему бонусу) — источник с памятью.

Разработайте программу, которая по заданному файлу Q рассчитывает:

- $-count(a_ia_k)$  количество вхождений подстрок  $a_ia_k$ ;
- $count(a_j^*)$  общее количество вхождений любых двухсимвольных подстрок, начинающихся с  $a_j$  (для всех символов, кроме последнего символа файла,  $count(a_j^*) = count(a_j^*)$ ;

и оценивает, строя модель источника символов по файлу  $Q = c_1 \dots c_n$  в виде стационарного источника Маркова первого порядка:

- условную вероятность  $p(a_k|a_j) = count(a_ja_k) / \sum count(a_ja_k) = count(a_ja_k) / count(a_j^*)$  каждой пары символов  $a_j$ ,  $a_k \in A_1$ ;
- суммарное количество информации ICM1(Q) в файле Q в битах и байтах (безусловную вероятность p(c1 = aj) считайте равной 1 / 256); аналогично  $\Pi = 1$  (символ кодирования = байт, как всегда).

Код: https://pastebin.com/wsbPEuEP

```
import math
from collections import Counter
import pandas as pd

def open_file(path, open_mode):
    with open(path, **open_mode) as file:
        return file.read()

def create_markov_table(file_data):
    octats = [hex(item)[2:] for item in list(Counter(file_data).keys())]
    df = pd.DataFrame(columns=octats, index=octats)
    for index in df.index:
        df.loc[index] = [0] * len(list(Counter(file_data).keys()))
    df.loc['-'] = [1 / 256] * len(df.index)
```

```
for i, item in enumerate(file_data):
        if i != 0:
            cur = hex(file_data[i])[2:]
            prev = hex(file_data[i - 1])[2:]
            df.loc[prev, cur] += 1
    df = df.assign(sum=df.sum(axis=1))
    return df
def calculate_prob(file_data, df):
    prob = 1
    for i, item in enumerate(file_data):
        cur = hex(file_data[i])[2:]
        if i == 0:
            prob *= df.loc["-", str(cur)]
        else:
            prev = hex(file_data[i - 1])[2:]
            if df.loc[str(prev), "sum"] != 0:
                prob *= df.loc[str(prev), str(cur)] / df.loc[str(prev), "sum"]
    return prob
def calculate_log(file_data, df):
    # calculating amount of information directly
    # use where probability of message is very small to store
    log_ret = 0
    for i, item in enumerate(file_data):
       cur = hex(file_data[i])[2:]
        if i == 0:
            prob = df.loc["-", str(cur)]
            prev = hex(file_data[i - 1])[2:]
            if df.loc[str(prev), "sum"] != 0:
                prob = df.loc[str(prev), str(cur)] / df.loc[str(prev), "sum"]
            else:
                prob = 1
        log_ret += math.log2(prob)
    return -log_ret
FILE_PATH = r"C:\Users\1\Downloads\otik-master\otik-master\labs-files\Файлы в формате простого текста
– кодировки разные\gbfd"
FILE_OPEN_MODE = {'mode': 'rb'}
file_data = open_file(FILE_PATH, FILE_OPEN_MODE)
df = create_markov_table(file_data)
prob = calculate_prob(file_data, df)
print(f"Таблица:\n {df}")
```

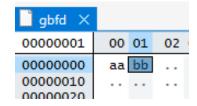
```
print(f"\nВероятность сообщения = {prob}")
print(f"Количество информации = {-math.log2(prob)} бит")
```

Для проверки корректности используйте файлы с заранее известным  $I_{\text{CM1}}(Q)$ :  $I_{\text{CM1}}(a) = I_{\text{CM1}}(ab) = I_{\text{CM1}}(abcd) = I_{\text{CM1}}(abab) = 8$  бит (1 байт х86);  $I_{\text{CM1}}(abac) = 10$  бит (1,25 байта х86);  $I_{\text{CM1}}(aabacad) = 16$  бит (2 байта х86).

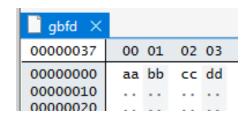
### 1) $I_{CM1}(a)$



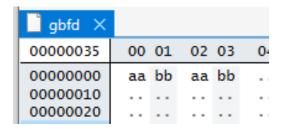
## 2) $I_{CM1}(ab)$



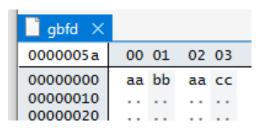
# 3) $I_{CM1}(abcd)$



## 4) $I_{CM1}(abab)$



## 5) $I_{CM1}(abac)$



Ta6	лица:								
	aa	bb	cc	sum					
aa	Θ	1	1	2					
bb	1	Θ	Θ	1					
cc	Θ	Θ	Θ	Θ					
-	0.003906 0.0	03906 0.0	003906 0.	011719					
	Вероятность сообщения = 0.0009765625 Количество информации = 10.0 бит								

# 6) $I_{CM1}(aabacad)$

☐ gbfd ×								
0000006b	00	01	02	03	04	05	06	07
00000000	aa	aa	bb	aa	cc	aa	dd	
00000010								
00000020								
00000030								
00000040								
00000000								

Таблица:									
	aa	bb	CC	dd	sum				
aa	1	1	1	1	4				
bb	1	Θ	Θ	0	1				
cc	1	Θ	Θ	0	1				
dd	Θ	Θ	0	0	0				
-	0.003906 0.0	03906 0.00	93906 0.0	03906 0.0	15625				
Вероятность сообщения = 1.52587890625e-05									
Количество информации = 16.0 бит									

Проверьте разработанную программу на файлах различного формата (не только простом тексте; в том числе и на бинарных). Сопоставьте результат с Л2.№1.

1) Бинарный файл: otik-master\labs-files\Файлы в разных форматах\hexdump Стационарный источник Маркова 1 порядка:

Ta	блица:										
	7f	45	4c	46		95	4a	d5	sum		
7f	Θ	1	Θ	Θ		Θ	0	Θ	8		
45	0	0	1	0		Θ	Θ	Θ	141		
4c	0	1	0	2		Θ	Θ	Θ	71		
46	0	0	0	0		Θ	Θ	Θ	70		
1	Θ	2	1	3		Θ	Θ	Θ	436		
8a	Θ	Θ	Θ	Θ		Θ	Θ	Θ	18		
95	Θ	Θ	Θ	Θ		Θ	Θ	Θ	4		
4a	Θ	Θ	Θ	Θ		Θ	9	Θ	21		
d5	0	Θ	Θ	Θ		Θ	Θ	Θ	11		
-	0.003906	0.003906	0.003906	0.003906		0.003906	0.003906	0.003906	1.0		
[25	[257 rows x 257 columns]										
Кол	Количество информации = 93805.06103711652 бит										

#### Источник без памяти:

```
Длина файла Q в битах: 208992
Суммарное количество информации в файле в битах: 164217.14
Дробная часть в экспон. форме: 1.41e-01
```

2) Простой текст: otik-master\labs-files\Файлы в формате простого текста — utf8\The Secret Adversary, by Agatha Christie.txt

Стационарный источник Маркова 1 порядка:

Таблица:									
	ef	bb	bf		40	24	sum		
ef	0	1	0		0	Θ	1		
bb	0	Θ	1		0	Θ	1		
bf	0	Θ	Θ		Θ	Θ	1		
54	0	Θ	0		0	Θ	2788		
68	Θ	Θ	Θ		Θ	Θ	19300		
2f	0	Θ	Θ		Θ	Θ	26		
25	0	Θ	Θ		Θ	Θ	1		
40	Θ	Θ	Θ		Θ	Θ	2		
24	Θ	0	Θ		Θ	Θ	2		
- 0.00	3906 0.	003906 0.0	003906		0.003906	0.003906	0.335938		
[87 rows x 87 columns]									
Количест	Количество информации = 1596594.8210852528 бит								

#### Источник без памяти:

Длина файла Q в битах: 3563952 Суммарное количество информации в файле в битах: 2057918.11 Дробная часть в экспон. форме: 1.15e-01 3) Простой текст: otik-master\labs-files\Файлы в формате простого текста — utf8\ Лев Николаевич Толстой. Война и мир 1.txt

## Стационарный источник Маркова 1 порядка:

Ta6	Таблица:									
	а	d0	9b		5d	57	' sum			
а	3648	4669	Θ		Θ	0	12274			
dΘ	0	0	145		Θ	0	383048			
9b	0	120	0		Θ	0	145			
2e	2918	17	0		36	Θ	8389			
20	202	73182	0		1	8	106575			
5a	Θ	Θ	Θ		Θ	Θ	4			
5b	Θ	150	Θ		Θ	Θ	165			
5d	163	0	0		Θ	Θ	166			
57	0	Θ	0		Θ	Θ	10			
-	0.003906	0.003906	0.003906		0.003906	0.003906	0.566406			
[14	[146 rows x 146 columns]									
Кол	Количество информации = 3215655.6380596594 бит									

#### Источник без памяти:

Длина файла Q в битах: 10198496

Суммарное количество информации в файле в битах: 5394161.66

Дробная часть в экспон. форме: 6.57е-01

### Лабораторная работа 4 (упрощённое задание)

## Сжатие данных без учёта контекста (энтропийное сжатие)

Операционная система ОС Windows, 64-разрядная ОС

#### Л4.1. Упрощённое задание (не более 7 баллов за работу)

Используйте демонстрационные программы любой свободной библиотеки для сжатия методом Хаффмана, Шеннона—Фано, Шеннона либо целочисленным арифметическим (интервальным) методом.

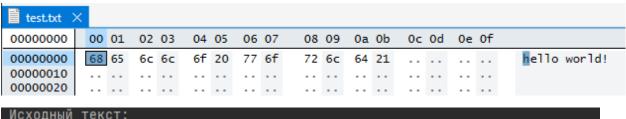
Код: <a href="https://pastebin.com/bzErWUcP">https://pastebin.com/bzErWUcP</a>

```
import collections
import math
import huffman
from collections import Counter
def get_total_information(file_path):
   def calculate_information(probability):
       if probability == 0:
           return 0
       return -math.log2(probability)
   with open(file_path, 'rb') as file:
       file_content = file.read()
   file_length = len(file_content)
    byte_frequencies = collections.Counter(file_content)
    byte_probabilities = {byte: freq / file_length for byte, freq in byte_frequencies.items()}
    byte_information = {byte: calculate_information(prob) for byte, prob in byte_probabilities.items()}
    total_information = sum(freq * byte_information[byte] for byte, freq in byte_frequencies.items())
    return total_information
def read_file(file_path):
    with open(file_path, 'rb') as infile:
       return infile.read()
def huffman_compress(data):
    freq_counter = Counter(data)
    huffman_tree = huffman.codebook(freq_counter.items())
    encoded_data = ''.join(str(huffman_tree[char]) for char in data)
    return huffman_tree, encoded_data
def huffman_decompress(huffman_tree, encoded_data):
   reverse_tree = {v: k for k, v in huffman_tree.items()}
    decoded_data = ""
    buffer = ""
    for bit in encoded_data:
       buffer += bit
       if buffer in reverse_tree:
```

```
decoded_data += chr(reverse_tree[buffer])
           buffer = ""
   return decoded_data
input_file = "test.txt"
data = read_file(input_file)
# print(f"Исходный текст:\n{data}\n")
print(f"Длина исходных данных: {len(data)} символов")
# Сжатие данных
huffman_tree, compressed_data = huffman_compress(data)
print(f"Таблица кодирования:")
for key, value in huffman_tree.items():
    print(f"{chr(key)}:{value}", end="; ")
print(f"\nСжатые данные:\n{compressed_data}\n")
print(f"Размер сжатых данных: {len(compressed_data)} бит")
print(f"Оценка количества информации исходного файла {input_file}: {get_total_information(input_file)}")
# Декодирование сжатых данных
decompressed_data = huffman_decompress(huffman_tree, compressed_data)
# print(f"Декодированные данные:\n{decompressed_data}\n")
print(f"Длина декодированных данных: {len(decompressed_data)} символов")
```

#### Продемонстрируйте работу кодера и декодера.

#### 1) test.txt



```
Исходный текст:
b'hello world!'

Длина исходных данных: 12 символов
Таблица кодирования:
h:1110; e:1101; l:01; o:101; :1100; w:001; r:000; d:1111; !:100;
Сжатые данные:
1110110101011011100001101000011111100

Размер сжатых данных: 37 бит
Оценка количества информации исходного файла test.txt: 36.264662506490396
Декодированные данные:
hello world!
Длина декодированных данных: 12 символов
```

### 2) chess16.jpg

```
Длина мсходных данных: 575 символов
Таблица мсходных данных: 575 символов
```

# Сопоставьте длину сжатых данных с оценкой количества информации в исходном файле (Л2.№1).

1) Бинарный файл: otik-master\labs-files\Файлы в разных форматах\hexdump

```
Размер сжатых данных: 165450 бит
Оценка количества информации исходного файла C:\Users\1\Downloads\otik-master\otik-master\labs-files\Файлы в разных форматах\hexdump: 164217.14076973282
```

#### Источник без памяти:

```
Длина файла Q в битах: 208992
Суммарное количество информации в файле в битах: 164217.14
Дробная часть в экспон. форме: 1.41e-01
```

 Простой текст: otik-master\labs-files\Файлы в формате простого текста utf8\The Secret Adversary, by Agatha Christie.txt

```
Размер сжатых данных: 2073883 бит
Оценка количества информации исходного файла C:\Users\1\Downloads\otik-master\otik-master\labs-files\Файлы в формате простого текста — utf8\The Secret Adversary, by Agatha Christie.txt: 2057918.1146295287
```

#### Источник без памяти:

```
Длина файла Q в битах: 3563952
Суммарное количество информации в файле в битах: 2057918.11
Дробная часть в экспон. форме: 1.15e-01
```

 Простой текст: otik-master\labs-files\Файлы в формате простого текста utf8\ Лев Николаевич Толстой. Война и мир 1.txt

```
Pasmep сжатых данных: 5433149 бит
Dценка количества информации исходного файла C:\Users\1\Downloads\otik-master\otik-master\labs-files\Файлы в формате простого текста — utf8\Лев Николаевич Толстой. Война и мир 1.txt: 5394161.657836878
```

#### Источник без памяти:

```
Длина файла Q в битах: 10198496
Суммарное количество информации в файле в битах: 5394161.66
Дробная часть в экспон. форме: 6.57e-01
```