

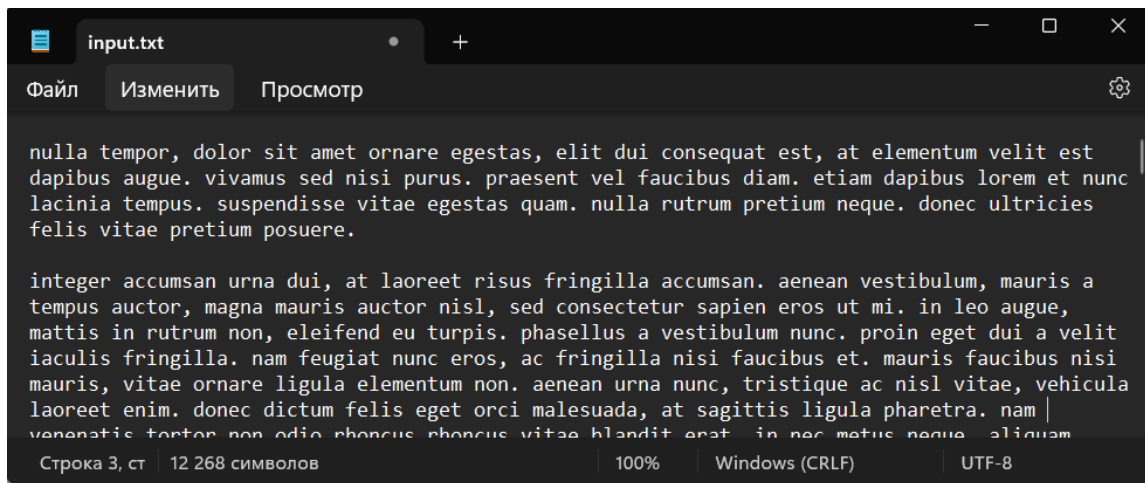
*Дискретная математика для программистов.
Лабораторная работа № 4. (февраль 2024)
Предметом исследования являются методы
кодирования. (Выполняется группой 3 человека, у
разных групп разные тексты!)*

Вариант 2. Задание:

0. Подобрать англоязычный текст длиной примерно три страницы (около 12000 символов), при этом для удобства все буквы переделать в строчные, оставив не более 32 различных символов.

Соколов Арсений

Имеем файл input.txt на 12.268 символов:



The screenshot shows a text editor window with a dark theme. The title bar at the top says "input.txt". Below the title bar is a menu bar with "Файл", "Изменить", and "Просмотр". The main text area contains two paragraphs of Latin text. The first paragraph is: "nulla tempor, dolor sit amet ornare egestas, elit dui consequat est, at elementum velit est dapibus augue. vivamus sed nisi purus. praesent vel faucibus diam. etiam dapibus lorem et nunc lacinia tempus. suspendisse vitae egestas quam. nulla rutrum pretium neque. donec ultricies felis vitae pretium posuere." The second paragraph is: "integer accumsan urna dui, at laoreet risus fringilla accumsan. aenean vestibulum, mauris a tempus auctor, magna mauris auctor nisl, sed consectetur sapien eros ut mi. in leo augue, mattis in rutrum non, eleifend eu turpis. phasellus a vestibulum nunc. proin eget dui a velit iaculis fringilla. nam feugiat nunc eros, ac fringilla nisi faucibus et. mauris faucibus nisi mauris, vitae ornare ligula elementum non. aenean urna nunc, tristique ac nisl vitae, vehicula laoreet enim. donec dictum felis eget orci malesuada, at sagittis ligula pharetra. nam | venenatis tortor non odio rhoncus rhoncus vitae blandit erat in nec metus neque aliquam". At the bottom of the window, there is a status bar that says "Строка 3, ст 12 268 символов", "100%", "Windows (CRLF)", and "UTF-8".

1. Провести статистический анализ по частоте букв и по частоте пар букв.

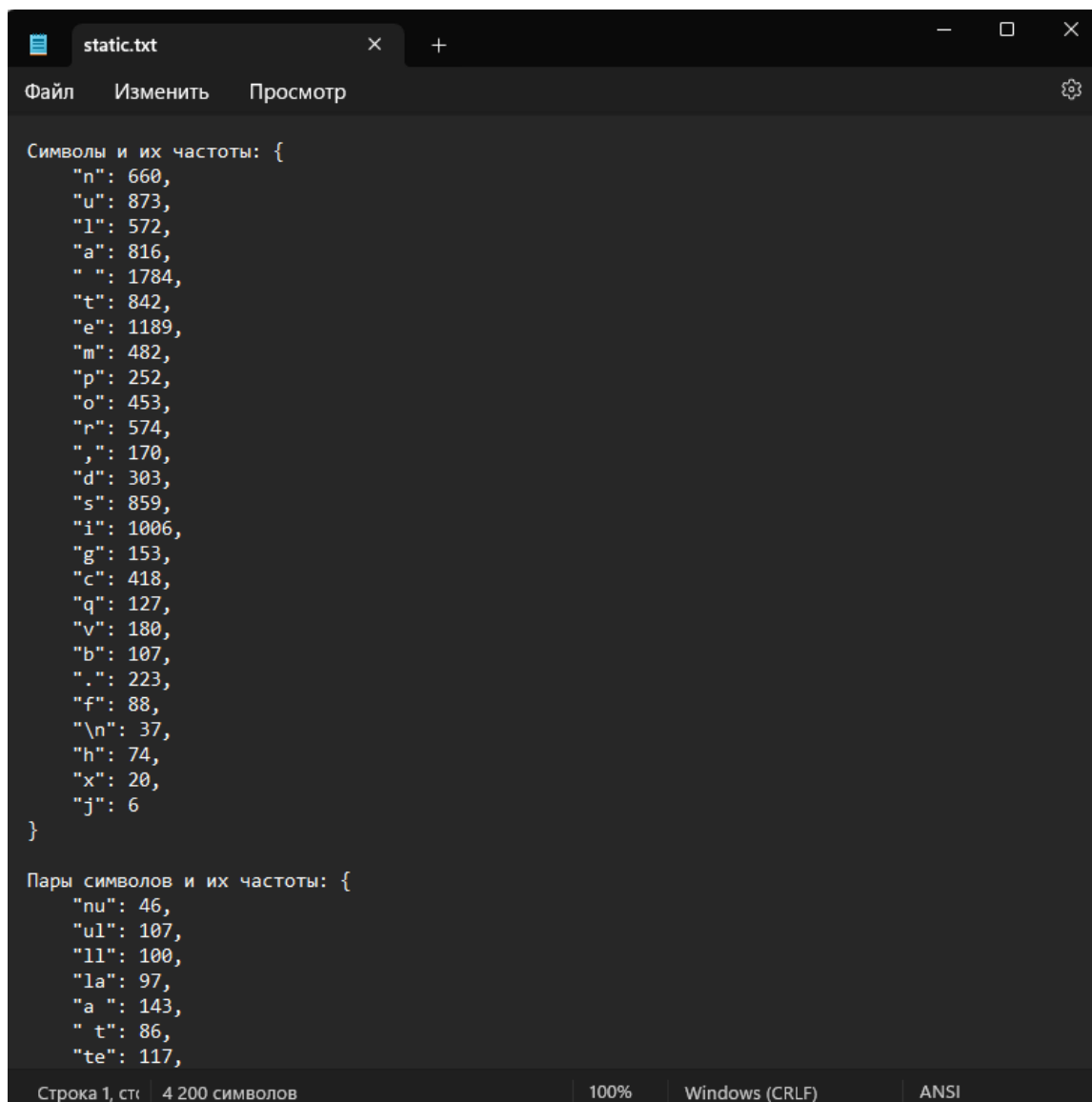
```
1 from collections import defaultdict

41 # Задание 1 статический анализ
42 freq_uniq = defaultdict(int) # частоты уникальных символов
43 freq_pair = defaultdict(int) # частоты пар символов
44
45 with open("input.txt", "r") as file:
46     text = file.read()
47 for ch in text:
48     freq_uniq[ch] += 1
49 for i in range(0, len(text) - 1):
50     freq_pair[text[i] + text[i + 1]] += 1
51
52 print("Задание 1")
53 print("Кол-во символов в исходном тексте: ", len(text))
54 print("Кол-во уникальных символов: ", len(freq_uniq))
55 print()
56
57 with open('static.txt', 'w') as file:
58     file.write("Символы и их частоты: ")
59     json.dump(freq_uniq, file, indent=4)
60     file.write("\n\n")
61     file.write("Пары символов и их частоты: ")
62     json.dump(freq_pair, file, indent=4)
63
```

В консоль выводится:

```
Run lab_4 x
C:\Users\User\AppData\Local\Programs\Python\Python310
Задание 1
Кол-во символов в исходном тексте: 12268
Кол-во уникальных символов: 26
```

В файл static.txt выводится:



```
static.txt
Файл  Изменить  Просмотр  [Settings Icon]

Символы и их частоты: {
  "n": 660,
  "u": 873,
  "l": 572,
  "a": 816,
  " ": 1784,
  "t": 842,
  "e": 1189,
  "m": 482,
  "p": 252,
  "o": 453,
  "r": 574,
  ",": 170,
  "d": 303,
  "s": 859,
  "i": 1006,
  "g": 153,
  "c": 418,
  "q": 127,
  "v": 180,
  "b": 107,
  ".": 223,
  "f": 88,
  "\n": 37,
  "h": 74,
  "x": 20,
  "j": 6
}

Пары символов и их частоты: {
  "nu": 46,
  "ul": 107,
  "ll": 100,
  "la": 97,
  "a ": 143,
  " t": 86,
  "te": 117,
}

Строка 1, столбец 4 200 символов | 100% | Windows (CRLF) | ANSI
```

2. На основе этих статистик построить коды Хаффмана, закодировать текст этими кодами, сравнить количество бит закодированного текста с равномерными (5-ти битовыми) кодами, по формуле Шенона найти количество информации и сравнить с кодами Хаффмана.

Код:

```
2 usages
6 class Node:
7     def __init__(self, label, frequency=0):
8         self.label = label
9         self.frequency = frequency
10        self.left = None
11        self.right = None
12        self.code = str()
13
14    def __repr__(self):
15        return f"{self.label}: {self.frequency}"
16
27 def encoding(text, codes): # кодирование текста по дереву Хаффмана
28     encoded_text = str()
29     for ch in text:
30         encoded_text += codes[ch]
31     return encoded_text
32
66 # Задание 2 кодировка Хаффмана
67 nodes = []
68 for k, v in freq_uniq.items():
69     nodes.append(Node(k, v))
70
71 nodes.sort(key=lambda x: x.frequency) # сортируется по возрастанию частот
72 end_nodes = nodes.copy() # словарь кодов Хаффмана
73
74 while len(nodes) != 1:
75     left = nodes.pop(0)
76     right = nodes.pop(0)
77     node = Node(left.label + right.label, left.frequency + right.frequency)
78     node.left = left
79     node.right = right
80     nodes.insert(0, node)
81     nodes.sort(key=lambda x: x.frequency)
82
83 codewords_making(nodes[0])
84 codes = dict() # ключи - символы, значения - коды Хаффмана
```

```

85
86 for node in end_nodes:
87     codes[node.label] = node.code
88
89 encoded = encoding(text, codes) # кодирование
90
91 print("Задание 2")
92 with open('huffman_coding.txt', 'w') as file:
93     file.write("Символы и их коды Хаффмана:\n")
94     json.dump(codes, file, indent=4)
95     file.write("\nЗакодированная строка: " + encoded)
96
97 print("Длина закодированного текста (алгоритм Хаффмана):", len(encoded))
98 print("Длина при равномерном (5-ти битовом) кодировании:", 5 * len(text))
99 print("Степень сжатия по сравнению с равномерным (5-ти битовом) кодированием:",
100       round(100 - (len(encoded) / (5 * len(text))) * 100, 2), "%")
101 print("Энтропия Шеннона:", Shannon_formula(freq_uniq, len(text)), "бит")
102
2 from math import ceil, log2
3 import json

```

Среднее количество информации (Энтропия Шеннона)

$$I = -\sum_{i=1}^N p_i \log_2 p_i$$

```

1 usage
34 def Shannon_formula(frequencies, text_length):
35     shannon = 0 # значение энтропии Шеннона
36     for char, prob in frequencies.items():
37         shannon += (prob / text_length) * log2(prob / text_length)
38     return round(-1 * shannon, 2)
39

```

В консоль выводится:

```

Задание 2
Длина закодированного текста (алгоритм Хаффмана): 51050
Длина при равномерном (5-ти битовом) кодировании: 61340
Степень сжатия по сравнению с равномерным (5-ти битовом) кодированием: 16.78 %
Энтропия Шеннона: 4.13 бит

```

В файл huffman.txt выводится:

```
huffman_coding.txt
Файл  Изменить  Просмотр
Символы и их коды Хаффмана:
{
  "j": "000001000",
  "x": "000001001",
  "\n": "00000101",
  "h": "0000011",
  "f": "0110110",
  "b": "0110111",
  "q": "000000",
  "g": "011000",
  ",": "011001",
  "v": "011010",
  ".": "111000",
  "p": "111001",
  "d": "00001",
  "c": "10100",
  "o": "10101",
  "m": "11101",
  "l": "0001",
  "r": "0100",
  "n": "0101",
  "a": "0111",
  "t": "1000",
  "s": "1001",
  "u": "1011",
  "i": "1111",
  "e": "001",
  " ": "110"
}
Закодированная строка:
01011011000100010111101000001110111100110101000110011100000110101000110101001101001111100011001111101001100011010101000101011101000
1110001011000001100110000111100101100111000100011111000110000011011111101010010101010010000001011011110001100011001100001100111001111
000110001000100111101010101100010111101110011010001000111110001100011001100010111110011111011011100111001111011011000101100111
100011001101011110110011111011011001110100100100001110010111111001111110111001101100101110011110001101110001101100110001100110010010101100011
0011010001000111001101100111101110100111101111001110000011111011111011100011000110001111011111011100000101111100111110111101111011100
1110000110101010000111101110001100011001011011011011001100001011110100111101111101111101000001110111100110111001111000110110011100111100
10010101000011111100110010011100111110000110011100011100111000000101101111101111000110011011011000100010111110011001
0111000010010111110111011100101000011000111110111101110010100100000101101101100110110110001100001001111101001111001
100111001101100010001111110011100110101111100001110011101110010100001100011110111110111011100110110011011001101100001111000000001010000010111
11010110000101100000101001100111101001010010111101100101110101110101101001010111110000110111110110011110011100011011101010100001001
```

Строка 1, стр | 51 584 символа | 100% | Windows (CRLF) | ANSI

Дискретная математика
Лабораторная работа №4
Гвозденко Демид

Код:

```
104 # Задание 3 кодировка LZW
105 LZW_dict = {} # словарь для кодирования текста
106 dictionary_size = 0
107 init_bits = 0
108 string = str() # текущая строка, которая будет использоваться для построения новых кодовых послед
109 LZW_encoded_res = str() # закодированная строка в двоичном формате.
110 LZW_encoded = [] # закодированная строка в виде списка кодовых последовательностей.
111
112 # присваиваем каждому символу уникальный код
113 for char in codes:
114     LZW_dict[char] = dictionary_size
115     dictionary_size += 1
116
117 # количество бит, необходимое для представления кодовых последовательностей в начальном словаре.
118 init_bits = ceil(log2(dictionary_size))
119
120 for char in text:
121     new_string = string + char
122     if new_string in LZW_dict: # Если new_string уже есть в словаре, то текущая строка string обнов
123         string = new_string
124     else: # Если new_string отсутствует в словаре, то текущая кодовая последовательность LZW_dict[s
125         LZW_encoded.append(LZW_dict[string])
126         LZW_dict[new_string] = dictionary_size
127         dictionary_size += 1
128         string = char
129
130 if string in LZW_dict: # проверяется, содержится ли последняя строка string в словаре LZW
131     LZW_encoded.append(LZW_dict[string])
132
133 for seq in LZW_encoded: #
134     bits = init_bits if seq == 0 or ceil(log2(seq)) < init_bits else ceil(log2(seq))
135     LZW_encoded_res += format(seq, f'0{bits}b')
136
137 with open('LZW_coding.txt', 'w') as file:
138     file.write("Dictor:\n")
139     json.dump(LZW_dict, file, indent=4)
140     file.write("\nEncoded string (bit): " + LZW_encoded_res)
141     file.write("\n\n")
142     file.write("Encoded string (code): " + str(LZW_encoded))
143
144 print()
145 print("Задание 3")
146 print("Длина закодированного текста (метод LZW): ", len(LZW_encoded_res))
147 print("Степень сжатия по сравнению с равномерным (пятибитовым) кодированием:",
148       round(100 - (len(LZW_encoded_res) / (5 * len(text))) * 100, 5), "%")
149 print("Степень сжатия по сравнению с кодами Хаффмана:",
150       round(100 - (len(LZW_encoded_res) / len(encoded)) * 100, 5), "%")
151
```


Вывод в консоль:

Задание 3

Длина закодированного текста (метод LZW): 34541

Степень сжатия по сравнению с равномерным (пятибитовым) кодированием: 43.68927 %

Степень сжатия по сравнению с кодами Хаффмана: 32.33888 %

Файл:

```
1 Dictor:
2 {
3     "j": 0,
4     "x": 1,
5     "\n": 2,
6     "h": 3,
7     "f": 4,
8     "b": 5,
9     "q": 6,
10    "g": 7,
11    ",": 8,
12    "v": 9,
13    ".": 10,
14    "p": 11,
15    "d": 12,
16    "c": 13,
17    "o": 14,
18    "m": 15,
19    "l": 16,
20    "r": 17,
21    "n": 18,
22    "a": 19,
23    "t": 20,
24    "s": 21,
25    "u": 22,
26    "i": 23,
3732 }
3733 Encoded string (bit): 1001010110100001000010011110011010011000011110101101110100010100011001011000111010000100
3734
3735 Encoded string (code): [18, 22, 16, 16, 19, 25, 20, 24, 15, 11, 14, 17, 8, 25, 12, 14, 16, 36, 25, 21, 23, 20,
```