МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»

(БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа 6

по дисциплине: Теория информации

тема: «Арифметическое кодирование»

Выполнил: ст. группы ПВ-211

Шамраев Александр Анатольевич

Проверил:

Твердохлеб Виталий Викторович

Белгород 2023 г.

содержание отчета

[Задание 3](#_Toc133217021)

[1 Построить обработчик, реализующий функцию арифметического кодирования. 4](#_Toc133217022)

[2 В качестве исходных данных, подлежащих обработке, использовать последовательности из работы №2. 5](#_Toc133217023)

[3 Для полученных результатов рассчитать показатели сжатия. Сравнить с полученными в работе №2. 8](#_Toc133217024)

Задание

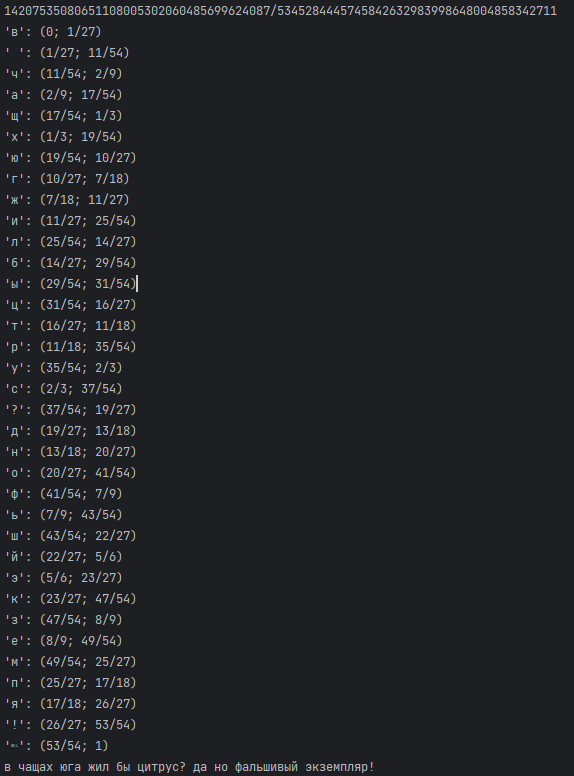
1. Построить обработчик, реализующий функцию арифметического кодирования.
2. В качестве исходных данных, подлежащих обработке, использовать последовательности из работы №2.
3. Для полученных результатов рассчитать показатели сжатия. Сравнить с полученными в работе №2.

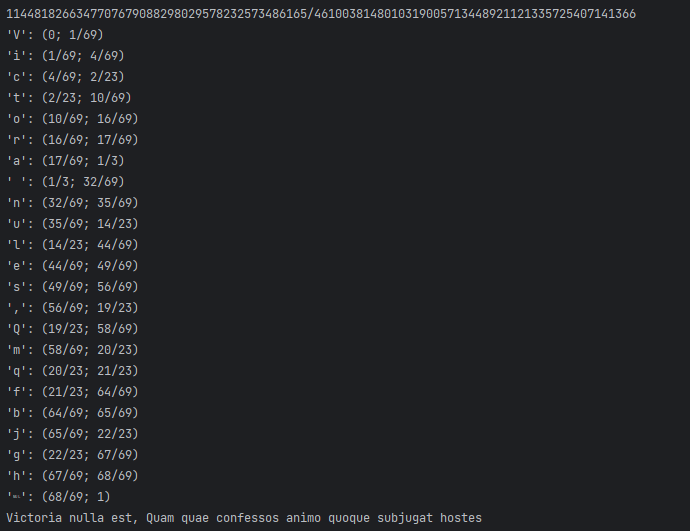
# Построить обработчик, реализующий функцию арифметического кодирования.

from sys import getsizeof  
from fractions import Fraction  
  
  
def code\_massage(message: str, x) -> tuple[Fraction, dict[str, tuple[Fraction, Fraction]]]:  
 message += "\0"  
 alphabet: dict[str, Fraction] = {}  
 decode\_table: dict[str, tuple[Fraction, Fraction]] = {}  
 max\_denominator = 10\*\*int(len(message) / x)  
  
 for ch in message:  
 alphabet[ch] = alphabet[ch] + 1 if alphabet.\_\_contains\_\_(ch) else 1  
  
 old: tuple[Fraction, Fraction] = (Fraction(0), Fraction(0))  
 for ch in alphabet.keys():  
 alphabet[ch] /= Fraction(len(message))  
 decode\_table[ch] = (old[1], old[1] + alphabet[ch])  
 old = decode\_table[ch]  
  
 current: tuple[Fraction, Fraction] = (Fraction(0), Fraction(0))  
 for ch in message:  
 t: tuple[Fraction, Fraction] = decode\_table[ch]  
 if current == (0, 0):  
 current = t  
 else:  
 left = current[0].limit\_denominator(max\_denominator)  
 right = current[1].limit\_denominator(max\_denominator)  
 current = (left + (right - left) \* t[0], left + (right - left) \* t[1])  
  
 return ((current[0] + current[1]) / 2).limit\_denominator(max\_denominator), decode\_table  
  
  
def decode\_message(coded\_message: Fraction, decode\_table: dict[str, tuple[Fraction, Fraction]]) -> str:  
 message: str = ""  
 while True:  
 t: tuple[str, tuple[Fraction, Fraction]] = ("",(Fraction(0), Fraction(0)))  
 for ch in decode\_table.keys():  
 if decode\_table[ch][0] <= coded\_message <= decode\_table[ch][1]:  
 t = (ch, decode\_table[ch])  
  
 if t[0] == "\0":  
 break  
  
 message += t[0]  
 coded\_message = (coded\_message - t[1][0]) / (t[1][1] - t[1][0])  
  
 return message

# В качестве исходных данных, подлежащих обработке, использовать последовательности из работы №2.

if \_\_name\_\_ == '\_\_main\_\_':  
 s = "в чащах юга жил бы цитрус? да но фальшивый экземпляр!"  
 undef = True  
 last\_x = x = 1  
 while undef or decode\_message(code, table) == s:  
 code, table = code\_massage(s, x)  
 if decode\_message(code, table) != s:  
 break  
 last\_x = x  
 x \*= 1.1  
 undef = False  
  
 code, table = code\_massage(s, last\_x)  
 print(code)  
 for ch in table:  
 print("\'{0}\': ({1}; {2})".format(ch, table[ch][0], table[ch][1]))  
 print(decode\_message(code, table))  
 print(len(s) / (getsizeof(code.numerator) + getsizeof(code.denominator) - 48))



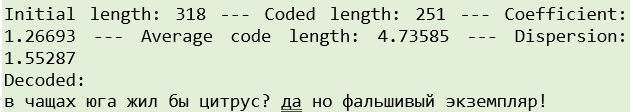


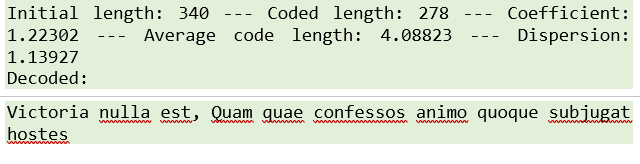
# Для полученных результатов рассчитать показатели сжатия. Сравнить с полученными в работе №2.





При выполнении второй лабораторной были получены следующие результаты:





Как видно, арифметическое кодирование сжало сообщения эффективнее метода Хаффмана