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

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.

1 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
```

2 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))
```

```
1420753508065110800530206048569962408 7/534528444574584263298399864800485834271 1
'в': (0; 1/27)
' ': (1/27; 11/54)
'ч': (11/54; 2/9)
'а': (2/9; 17/54)
'щ': (17/54; 1/3)
'х': (1/3; 19/54)
'ю': (19/54; 10/27)
'г': (10/27; 7/18)
'ж': (7/18; 11/27)
'и': (11/27; 25/54)
'л': (25/54; 14/27)
'б': (14/27; 29/54)
'ы': (29/54; 31/54)
'ц': (31/54; 16/27)
'т': (16/27; 11/18)
'р': (11/18; 35/54)
'у': (35/54; 2/3)
'с': (2/3; 37/54)
'?': (37/54; 19/27)
'д': (19/27; 13/18)
'н': (13/18; 20/27)
'о': (20/27; 41/54)
'ф': (41/54; 7/9)
'ь': (7/9; 43/54)
'ш': (43/54; 22/27)
'й': (22/27; 5/6)
'э': (5/6; 23/27)
'к': (23/27; 47/54)
'з': (47/54; 8/9)
'е': (8/9; 49/54)
'м': (49/54; 25/27)
'п': (25/27; 17/18)
'я': (17/18; 26/27)
'!': (26/27; 53/54)
'-': (53/54; 1)
в чащах юга жил бы цитрус? да но фальшивый экземпляр!
```

```
1144818266347707679088298029578232573486165/461003814801031900571344892112133572540714136 6
'V': (0; 1/69)
'i': (1/69; 4/69)
'c': (4/69; 2/23)
't': (2/23; 10/69)
'o': (10/69; 16/69)
'r': (16/69; 17/69)
'a': (17/69; 1/3)
' ': (1/3; 32/69)
'n': (32/69; 35/69)
'u': (35/69; 14/23)
'l': (14/23; 44/69)
'e': (44/69; 49/69)
's': (49/69; 56/69)
',': (56/69; 19/23)
'Q': (19/23; 58/69)
'm': (58/69; 20/23)
'q': (20/23; 21/23)
'f': (21/23; 64/69)
'b': (64/69; 65/69)
'j': (65/69; 22/23)
'g': (22/23; 67/69)
'h': (67/69; 68/69)
'«': (68/69; 1)
Victoria nulla est, Quam quae confessos animo quoque subjugat hostes
```

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

```
в чащах юга жил бы цитрус? да но фальшивый экземпляр!
1.325
```

```
Victoria nulla est, Quam quae confessos animo quoque subjugat hostes
1.5454545454545454
```

При выполнении второй лабораторной были получены следующие результаты:

```
Initial length: 318 --- Coded length: 251 --- Coefficient:
1.26693 --- Average code length: 4.73585 --- Dispersion:
1.55287
Decoded:
в чащах юга жил бы цитрус? да но фальшивый экземпляр!
```

```
Initial length: 340 --- Coded length: 278 --- Coefficient:
1.22302 --- Average code length: 4.08823 --- Dispersion:
1.13927
Decoded:
```

```
Victoria nulla est, Quam quae confessos animo quoque subjugat
hostes
```

Как видно, арифметическое кодирование сжало сообщения эффективнее метода Хаффмана