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»

Выполнил: ст. группы ПВ-211

Чувилко Илья Романович

Проверил:

Твердохлеб В.В.

Белгород 2023 г.

Тема: «Арифметическое кодирование»

Содержание отчета

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

Ход работы:

1. Построить обработчик, реализующий функцию арифметического кодирования

```
from sys import getsizeof
from fractions import Fraction

def codeMessage(message: str, x) -> tuple[Fraction, dict[str, tuple[Fraction, Fraction]]]:
    message += "\0"
    alphabet: dict[str, Fraction] = {}
    decodeTable: dict[str, tuple[Fraction, Fraction]] = {}
    maxDenominator = 10 ** int(len(message) / x)

    for i in message:
        alphabet[i] = alphabet[i] + 1 if alphabet.__contains__(i) else 1

    old: tuple[Fraction, Fraction] = (Fraction(0), Fraction(0))
    for i in alphabet.keys():
        alphabet[i] /= Fraction(len(message))
        decodeTable[i] = (old[1], old[1] + alphabet[i])
        old = decodeTable[i]

    current: tuple[Fraction, Fraction] = (Fraction(0), Fraction(0))
    for i in message:
        t: tuple[Fraction, Fraction] = decodeTable[i]
        if current == (0, 0):
            current = t
        else:
            left = current[0].limit_denominator(maxDenominator)
            right = current[1].limit_denominator(maxDenominator)
            current = (left + (right - left) * t[0], left + (right - left) * t[1])

    return ((current[0] + current[1]) / 2).limit_denominator(maxDenominator), decodeTable


def decodeMessage(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 decodeMessage(code, table) == s:
        code, table = codeMessage(s, x)
        if decodeMessage(code, table) != s:
            break
        last_x = x
        x *= 1.1
        undef = False
    code, table = codeMessage(s, last_x)
    print(code)
    for ch in table:
        print(f"\'{ch}\': ({table[ch][0]}; {table[ch][1]})")
    print(decodeMessage(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!
1.325
```

```
184820990658886881450107342407787950195941 7/72111432844558866783070035181700415021555847 07
'V': (0; 1/68)
'i': (1/68; 1/17)
'c': (1/17; 3/34)
't': (3/34; 5/34)
'o': (5/34; 4/17)
'r': (4/17; 1/4)
'a': (1/4; 23/68)
' ': (23/68; 31/68)
'n': (31/68; 1/2)
'u': (1/2; 41/68)
'l': (41/68; 43/68)
'e': (43/68; 12/17)
's': (12/17; 55/68)
',': (55/68; 14/17)
'Q': (14/17; 57/68)
'm': (57/68; 59/68)
'q': (59/68; 31/34)
'f': (31/34; 63/68)
'b': (63/68; 16/17)
'j': (16/17; 65/68)
'g': (65/68; 33/34)
'h': (33/34; 67/68)
'‥': (67/68; 1)
Victoria nullaest, Quam quae confessos animo quoque subjugat hostes
1.5227272727272727
```

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 nullaest, Quam quae confessos animo quoque subjugat hostes
1.5227272727272727
```

	Размер в биитах	
в чащах юга жил бы цитрус?		

Вес сообщений после сжатия (в битах):

	ЛР№2	Арифметического кодирование
в чащах юга жил бы цитрус?..	396	48
Victoria nulla est...	278	48

Как видно, арифметическое кодирование сжало сообщения эффективнее метода Хаффмана