МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Отчет по лабораторной работе №11**

**СЖАТИЕ/РАСПАКОВКА ДАННЫХ АРИФМЕТИЧЕСКИМ МЕТОДОМ**

Выполнил:

Cтудент 3 курса 1 группы

Парибок И. А.

Вариант 5

Минск 2022

**Цель**: приобретение практических навыков использования арифметических методов сжатия/распаковки данных.

**Практическое задание:**

1. Разработать авторское приложение в соответствии с целью лабораторной работы.

2. С помощью приложения выполнить прямое и обратное пре образования сообщений в соответствии с таблицей.

3. Дать оценку возможности переполнения при выполнении вычислений.

4. Сравнить характеристики арифметического сжатия с вероятностными алгоритмами.

5. Результаты оформить в виде отчета по установленным правилам.

**Выполнение работы:**

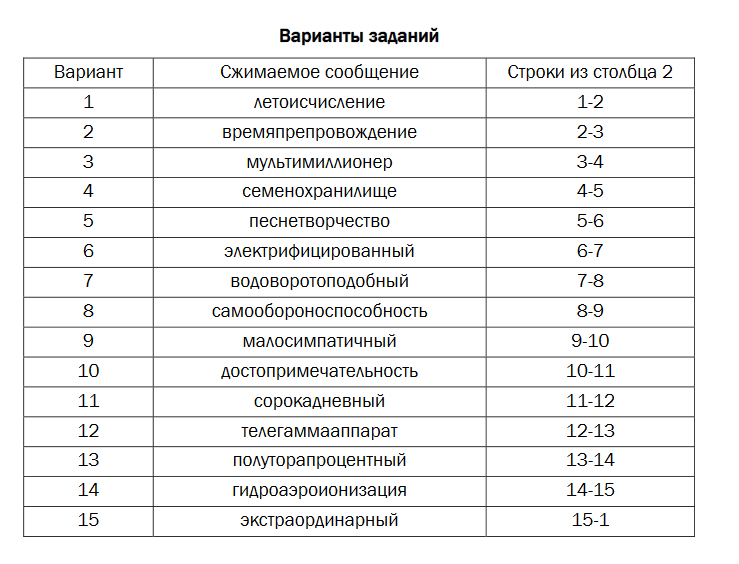


Рисунок 1 – ­Варианты заданий

Результаты выполнения задания представлены в рисунках 2-3. Листинг программы изложен в приложении.

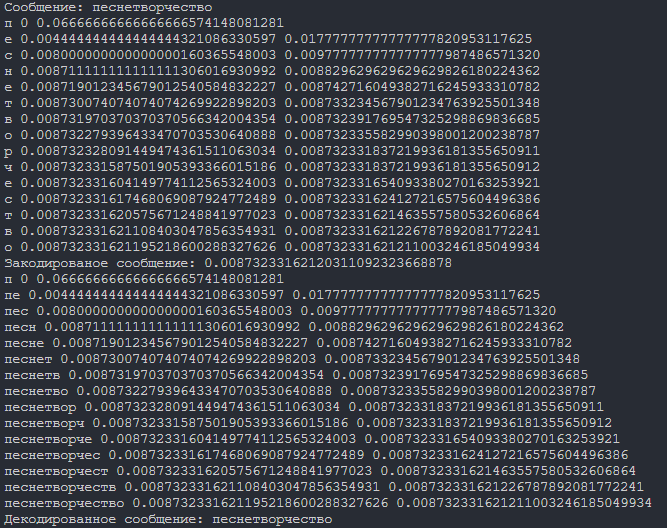


Рисунок 2 – Сжатие и распаковка сообщения «песнетворчество» аримфетическим методом

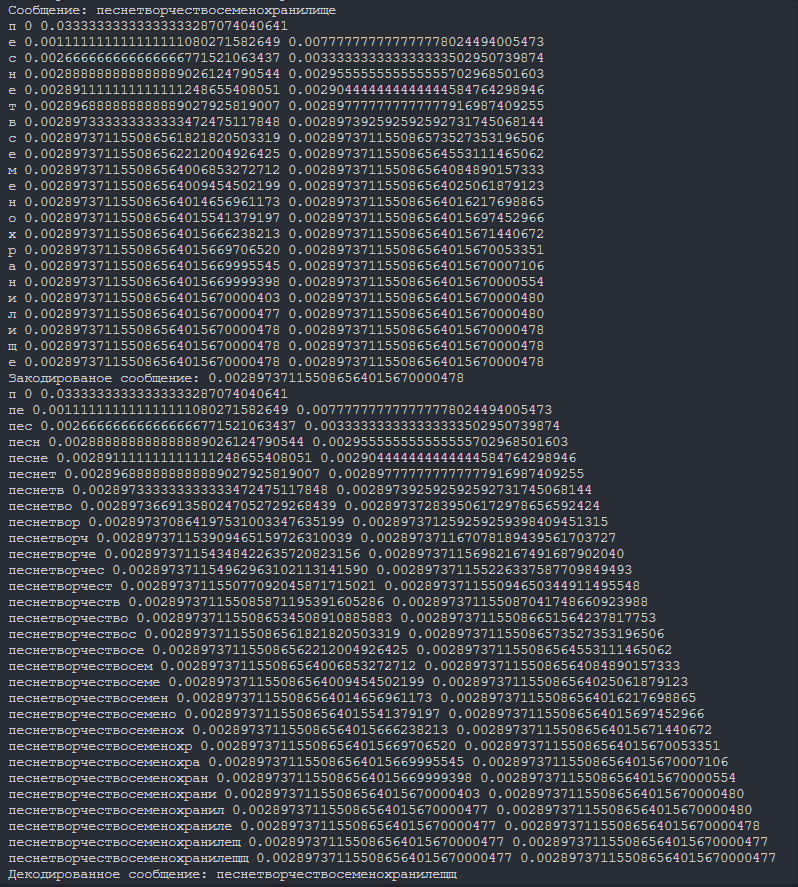


Рисунок 3 – Сжатие и распаковка сообщения «песнетворчествосеменохранилище» аримфетическим методом

**Вывод**: в результате данной лабораторной работы было разработано приложение для с использованием арифметического метода сжатия/распаковки данных. Арифметический метод обеспечивает почти оптимальную степень сжатия с точки зрения энтропийной оценки кодирования Шеннона. На каждый символ требуется почти *H* бит, где *H* — информационная энтропия источника.

В отличие от алгоритма Хаффмана, метод арифметического кодирования показывает высокую эффективность для дробных неравномерных интервалов распределения вероятностей кодируемых символов. Однако в случае равновероятного распределения символов, например, для строки бит 010101…0101 длины s метод арифметического кодирования приближается к префиксному коду Хаффмана и даже может занимать на один бит больше.

Арифметическое переполнение — специфичная для компьютерной арифметики ситуация, когда при арифметическом действии результат становится больше максимально возможного значения для переменной, использующейся для хранения результата..

Приложение

from decimal import Decimal

class ENCODER:

    def \_\_init\_\_(self, frequency\_table):

        self.probability\_table = self.get\_probability\_table(frequency\_table)

    def get\_probability\_table(self, frequency\_table):

        total\_frequency = sum(list(frequency\_table.values()))

        probability\_table = {}

        for key, value in frequency\_table.items():

            probability\_table[key] = value/total\_frequency

        return probability\_table

    def get\_encoded\_value(self, encoder):

        last\_stage = list(encoder[-1].values())

        last\_stage\_values = []

        for sublist in last\_stage:

            for element in sublist:

                last\_stage\_values.append(element)

        last\_stage\_min = min(last\_stage\_values)

        last\_stage\_max = max(last\_stage\_values)

        return (last\_stage\_min + last\_stage\_max)/2

    def process\_stage(self, probability\_table, stage\_min, stage\_max):

        stage\_probs = {}

        stage\_domain = stage\_max - stage\_min

        for term\_idx in range(len(probability\_table.items())):

            term = list(probability\_table.keys())[term\_idx]

            term\_prob = Decimal(probability\_table[term])

            cum\_prob = term\_prob \* stage\_domain + stage\_min

            stage\_probs[term] = [stage\_min, cum\_prob]

            stage\_min = cum\_prob

        return stage\_probs

    def encode(self, msg, probability\_table):

        encoder = []

        stage\_min = Decimal(0.0)

        stage\_max = Decimal(1.0)

        for msg\_term\_idx in range(len(msg)):

            stage\_probs = self.process\_stage(probability\_table, stage\_min, stage\_max)

            msg\_term = msg[msg\_term\_idx]

            stage\_min = stage\_probs[msg\_term][0]

            stage\_max = stage\_probs[msg\_term][1]

            print(msg\_term,stage\_min,stage\_max)

            encoder.append(stage\_probs)

        stage\_probs = self.process\_stage(probability\_table, stage\_min, stage\_max)

        encoder.append(stage\_probs)

        encoded\_msg = self.get\_encoded\_value(encoder)

        return encoder, encoded\_msg

    def decode(self, encoded\_msg, msg\_length, probability\_table):

        decoder = []

        decoded\_msg = ""

        stage\_min = Decimal(0.0)

        stage\_max = Decimal(1.0)

        for idx in range(msg\_length):

            stage\_probs = self.process\_stage(probability\_table, stage\_min, stage\_max)

            for msg\_term, value in stage\_probs.items():

                if encoded\_msg >= value[0] and encoded\_msg <= value[1]:

                    break

            decoded\_msg = decoded\_msg + msg\_term

            stage\_min = stage\_probs[msg\_term][0]

            stage\_max = stage\_probs[msg\_term][1]

            print(decoded\_msg,stage\_min,stage\_max)

            decoder.append(stage\_probs)

        stage\_probs = self.process\_stage(probability\_table, stage\_min, stage\_max)

        decoder.append(stage\_probs)

        return decoder, decoded\_msg

def get\_char\_frequency(s):

*# создаем словарь, где каждому символу ставим в соответствие его частоту*

    char\_frequency = {}

    for c in s:

        if c in char\_frequency:

            char\_frequency[c] += 1

        else:

            char\_frequency[c] = 1

    return char\_frequency

def results(msg):

    original\_msg = msg

    AE = ENCODER(get\_char\_frequency(original\_msg))

    print("Сообщение: {msg}".format(msg=original\_msg))

    encoder, encoded\_msg = AE.encode(msg=original\_msg,

                                    probability\_table=AE.probability\_table)

    print("Закодированое сообщение: {msg}".format(msg=encoded\_msg))

    decoder, decoded\_msg = AE.decode(encoded\_msg=encoded\_msg,

                                    msg\_length=len(original\_msg),

                                    probability\_table=AE.probability\_table)

    print("Декодированное сообщение: {msg}".format(msg=decoded\_msg))

results("песнетворчество")

results("песнетворчествосеменохранилище")