Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждения высшего образования

«Национальный исследовательский технологический университет «МИСиС»

Лабораторная работа №3.4

по дисциплине

«Основы теории информации»

на тему:

«Основная задача кодирования»

Направление подготовки:

09.03.01 «Информатика и вычислительная техника»

Семестр 5

Выполнил: Проверил:

Попов Егор Куприянов Вячеслав

Борисович \_\_\_\_ Васильевич\_\_\_\_\_\_\_\_

(Ф.И.О. студента) (Ф.И.О преподавателя)

БИВТ-22-СП-4 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(№ группы) (оценка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата сдачи) (дата проверки)

Подпись: \_\_\_\_\_\_\_\_\_\_ Подпись: \_\_\_\_\_\_\_\_\_\_

Москва – 2022

**Оглавление**

[Цель работы 3](#_Toc12473)

[1. Теоретическая часть 4](#_Toc12474)

[2. Условие задачи 5](#_Toc12475)

[3. Практическая часть 5](#_Toc12476)

[Заключение 9](#_Toc12477)

[Приложение 9](#_Toc12478)

[Приложение 1: 9](#_Toc12479)

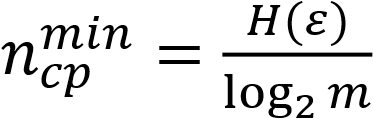
# Цель работы

Изучить тему «Основная задача кодирования» и реализовать решение задачи №3.1

(стр. 12) при помощи языка программирования Python 3.10 для закрепления материала.

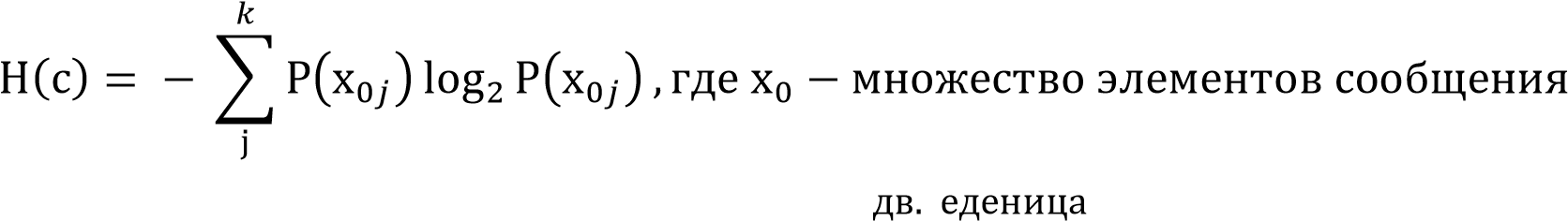
# Теоретическая часть

Средняя длина кодовой комбинации:

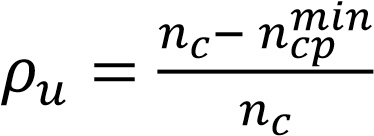
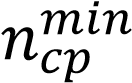
 , где 𝐻(𝜀) – энтропия сообщения, m – количество символов в

кодировке.

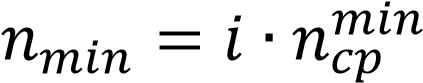
Энтропия сообщения – количество информации, содержащейся в любом осредненном сообщении.



Единица измерения энтропии сообщения: сообщение Величина избыточности источника:

, где 𝑛𝑐 – округленное в большую сторону значение .

Минимальная и необходимая длина кодовой комбинации:

, где I – количество букв элементарного сообщения.

Длина кодовой комбинации при заданной длине элементарного сообщения:

𝑛 = 𝑖 ∙ 𝑙𝑖, где I – количество букв элементарного сообщения, выбранная длина элементарного сообщения.

# Условие задачи

Имеется сообщение 𝜀 = (𝜀1, 𝜀2, … , 𝜀8) со значениями вероятности из появления.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 𝜀 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| P(𝜀) | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 |

Сообщение кодируется с помощью алфавита A = (1, 0) (т. е. m = 2), а каждое элементарное сообщение записывается тремя символами 𝑙𝑖 = 3, 𝑖 = 1, 2, … , 𝑛.

Кодовые слова имеют вид:

𝑉1 = 000; 𝑉2 = 001; 𝑉3 = 010;

𝑉4 = 011; 𝑉5 = 100; 𝑉6 = 101; 𝑉7 = 110; 𝑉8 = 111.

Какое качество используемого кода?

# Практическая часть

Для реализации решения задачи потребуется функции логарифма и округления в большую сторону, которые нужно подключить из библиотеки math командой from math import log и from math import ceil. Функция log принимает 2 аргумента: аргумент логарифма, основание логарифма. Функция ceil принимает на вход 1 параметр – число, которое нужно округлить.

Функция для вычисления длины кодовой комбинации:

def CodeCombination(length, amount):

    return length \* amount

Данная функция принимает 2 параметра: длину элементарного сообщения и количество сообщений, и возвращает их произведение.

Функция для вычисление средней длины кодовой комбинации:

def FindAverageLength(entropy, countSigns):

    return entropy / log(countSigns, 2)

Данная функция принимает 1 параметр – энтропию сообщения, и её возвращает отношение к логарифму с параметром countSigns по основанию 2.

Функция для нахождения избыточности кода:

def RedundancyOfCode(entropy, length, countSigns):

    ncp = FindAverageLength(entropy, countSigns)

    return round((length - ncp) / length \* 100, 3)

Данная функция принимает 3 параметр: энтропию сообщения, выбранную длину элементарного сообщения и количество знаков в эл. сообщении, затем запускает функцию FindAverageLength, для нахождения средней длины кода. В конце функция возвращает избыточность кода.

Функция для поиска энтропии сообщения:

def EntropyMessage(probabilityArray):

    result = 0

    for i in range(0, len(probabilityArray)):

        result += probabilityArray[i] \* log(probabilityArray[i], 2)

    if result != 0:

        result \*= -1

    print('Entropy: ', result)

    return result

Данная функция получает 2 аргумента: countMessages, хранящий в себе количество сообщений, и countSigns, хранящий в себе количество символов в алфавите кода в кодирования. Так как события равновероятны, то сразу вычисляем вероятность одного события и записываем результат в переменную p. Далее через цикл суммируем произведение вероятностей на логарифм от вероятности по основанию countSign. После чего умножаем результат на -1 и возвращаем его (в случае, когда result = 0, python выводит

«-0», что не совсем корректно, поэтому я добавил проверку).

Функция для ввода и валидации чисел:

def Validation():

    while True:

        try:

            data = int(input('Введите целое число более 0: '))

            if data > 0:

                return data

            print('Введенное значение не больше 0')

        except:

            print('Вы ввели не число')

            print('Попробуйте еще раз')

Данная функция реализует ввод данных и их проверку на условия:

1. Введенное значение является целым числом;
2. Введенное значение больше 0.

Первое условие проверяется через конструкцию try – except, которая обрабатывает ошибки, не прекращая выполнения программы. То есть при вводе набора символом, который нельзя преобразовать в целочисленные тип данных, управляющий поток перейдет в блок except и выведет сообщение «Вы ввели не число». А при вводе целого числа не больше 0, программа выведет сообщение «Введенное значение не больше 0».

В конце, если значение не удовлетворило условиям выводится сообщение «Попробуйте еще раз».

Функция для ввода списка дробей:

def InputProbability():

    print('Введите количество сообщений')

    count = Validation()

    probabilityArray = []

    while True:

        i = 0

        while i < count:

            prob = input('Введите вероятность появления сообщения в формате "1/2": ')

            try:

                first = int(prob.split('/')[0])

                second = int(prob.split('/')[1])

                probabilityArray.append(first / second)

                i += 1

            except:

                print('Данные введены не верно! Попробуйте еще раз.')

        if sum(probabilityArray) == 1:

            return probabilityArray

        print('Помните. Сумма вероятностей должна быть равна 1!')

Данная функция реализует ввод данных и их проверку на условия:

1. Введенное значение удовлетворяет формату «1/2»;
2. Полученное значение можно представить десятичной дробью;
3. Сумма вероятностей равна 1.

Первые условия проверяется через конструкцию try – except, которая обрабатывает ошибки, не прекращая выполнения программы. То есть при вводе набора символом, который нельзя преобразовать десятичную дробь, управляющий поток перейдет в блок except и выведет сообщение «Данные введены не верно! Попробуйте еще раз.». В случае, если сумма вероятностей будет не равна 1, программа выведет сообщение «Помните.

Сумма вероятностей должна быть равна 1!».

Основная функция: def

def Main():

    countOfSignForEncoding = 2

    print('Вычисление качества используемого кода')

    array = InputProbability()

    print('Введите длину записи закодированного сообщения')

    length = Validation()

    entropy = EntropyMessage(array)

    npc = FindAverageLength(entropy, countOfSignForEncoding)

    print(f"Средняя длина кодового слова: {npc:} буквы")

    print('Избыточность используемого кода: ', RedundancyOfCode(entropy, length, countOfSignForEncoding), '%')

    print('Минимально необходимая длина кодовой комбинации равна: ', CodeCombination(npc, len(array)))

    print('Длина кодовой комбинации при заданной длине равна: ', CodeCombination(length, len(array)))

    input("Нажмите Enter для завершения")

Данная функция реализует в себе основную логику работы программы: получает необходимые значения через функцию Validation и InputProbatility, получает энтропию сообщения через функцию EntropyMessage, величину средней длины сообщения FindAverageLength. Также выводит на экран сопровождающие сообщения для пользователя. Переменная countOfSignForEncoding отвечает за количество знаков в кодировке. Так как в задании оно по условию задано равным 2 и в подавляющем большинстве случаев также равно 2, то я решил не делать для неё отдельный ввод, а записать в переменную.

Последняя конструкция не является обязательной и нужна для возможности подключения данного python файла к другому. Программа начинает работу с этой конструкции.

if \_\_name\_\_ == '\_\_main\_\_':

Main()

Пример работы программы на значениях задачи представлен на рисунке 1.

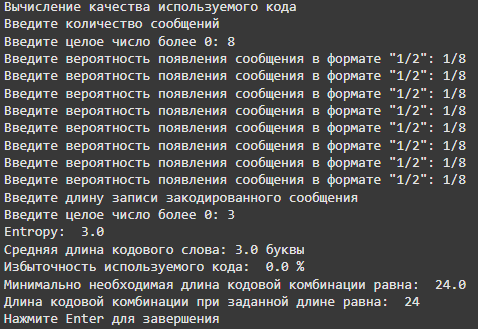


Рисунок 1 – пример работы программы.

# Заключение

В ходе работы была изучена тема «Основная задача кодирования», а также при помощи языка программирования Python 3.10 было реализовано решение задачи №3.4 (стр.

14) в общем виде (приложение 1) для закрепления материала.

# Приложение

## Приложение 1:

from math import log

# Вычисление длины кодовой комбинации

def CodeCombination(length, amount):

    return length \* amount

# Вычисление средней длины кодовой комбинации

def FindAverageLength(entropy, countSigns):

    return entropy / log(countSigns, 2)

# Нахождение избыточности кода

def RedundancyOfCode(entropy, length, countSigns):

    ncp = FindAverageLength(entropy, countSigns)

    return round((length - ncp) / length \* 100, 3)

# Функция для поиска энтропии сообщения

def EntropyMessage(probabilityArray):

    result = 0

    for i in range(0, len(probabilityArray)):

        result += probabilityArray[i] \* log(probabilityArray[i], 2)

    if result != 0:

        result \*= -1

    print('Entropy: ', result)

    return result

# Ввод и валидация данных

def Validation():

    while True:

        try:

            data = int(input('Введите целое число более 0: '))

            if data > 0:

                return data

            print('Введенное значение не больше 0')

        except:

            print('Вы ввели не число')

            print('Попробуйте еще раз')

# Ввод списка дробей

def InputProbability():

    print('Введите количество сообщений')

    count = Validation()

    probabilityArray = []

    while True:

        i = 0

        while i < count:

            prob = input('Введите вероятность появления сообщения в формате "1/2": ')

            try:

                first = int(prob.split('/')[0])

                second = int(prob.split('/')[1])

                probabilityArray.append(first / second)

                i += 1

            except:

                print('Данные введены не верно! Попробуйте еще раз.')

        if sum(probabilityArray) == 1:

            return probabilityArray

        print('Помните. Сумма вероятностей должна быть равна 1!')

def Main():

    countOfSignForEncoding = 2

    print('Вычисление качества используемого кода')

    array = InputProbability()

    print('Введите длину записи закодированного сообщения')

    length = Validation()

    entropy = EntropyMessage(array)

    npc = FindAverageLength(entropy, countOfSignForEncoding)

    print(f"Средняя длина кодового слова: {npc:} буквы")

    print('Избыточность используемого кода: ', RedundancyOfCode(entropy, length, countOfSignForEncoding), '%')

    print('Минимально необходимая длина кодовой комбинации равна: ', CodeCombination(npc, len(array)))

    print('Длина кодовой комбинации при заданной длине равна: ', CodeCombination(length, len(array)))

    input("Нажмите Enter для завершения")

if \_\_name\_\_ == '\_\_main\_\_':

    Main()