A logo of a university

Description automatically generated

International University of Information Technology

Department of Computer Engineering

**Laborotoy Work №6**

Complete by a student of the group: Ospan Ramazan it2-2310

Checked by senior lector of the Department of Computer Engineering:

Daurenbaeva Nurkamilya

Almaty 2024

**Arithmetic coding** is a method of data compression that converts an entire message into a single number, typically a fractional value between 0 and 1. Unlike traditional methods (like Huffman coding) that assign a fixed-length or variable-length code to each character, arithmetic coding processes the entire string at once and narrows down the range for the encoded value based on the probabilities of characters.

### Basic Principles:

1. **Probability Calculation**:
   * The first step is to calculate the probability of each character based on its frequency in the text. Characters that appear more often will have larger probabilities and, therefore, will occupy a larger portion of the interval.
2. **Range Initialization**:
   * Start with the interval **[0, 1)**.
   * The goal is to narrow this interval down as each character in the text is processed.
3. **Interval Division**:
   * For each character, the interval is divided into sub-intervals according to their probabilities. The size of each sub-interval corresponds to the probability of that character.
   * The current interval is then updated to the sub-interval that corresponds to the current character.
4. **Final Encoding**:
   * After processing all characters, the result is a single number that falls within the final interval. This number uniquely represents the original message.

Example:

Let's encode the word **"ABCABEAAFB"**.

1. **Step 1: Calculate Probabilities**
   * Character frequencies:
     + **A**: 4 times
     + **B**: 3 times
     + **C**: 1 time
     + **E**: 1 time
     + **F**: 1 time
   * Probabilities:
     + **P(A) = 4 / 10 = 0.4**
     + **P(B) = 3 / 10 = 0.3**
     + **P(C) = 1 / 10 = 0.1**
     + **P(E) = 1 / 10 = 0.1**
     + **P(F) = 1 / 10 = 0.1**
2. **Step 2: Define Ranges Based on Probabilities**
   * **A**: **[0.0, 0.4)**
   * **B**: **[0.4, 0.7)**
   * **C**: **[0.7, 0.8)**
   * **E**: **[0.8, 0.9)**
   * **F**: **[0.9, 1.0)**
3. **Step 3: Encode Each Character**
   * Start with **[0.0, 1.0)**
   * **First character "A"**: Range becomes **[0.0, 0.4)**
   * **Second character "B"**: Range becomes **[0.16, 0.28)**
   * **Third character "C"**: Range becomes **[0.244, 0.256)**
   * Continue this process for all characters in the string.
   * Final interval after processing all characters: **[0.2470936576, 0.2470944288)**
4. **Final Encoded Value**:
   * The final encoded value can be any number within the interval **[0.2470936576, 0.2470944288)**

**Advantages:**

1. **High Compression Efficiency**: Arithmetic coding can reach near-optimal compression ratios, especially for texts with characters of varying frequencies.
2. **Adaptive**: It can adapt to different input data sets and does not require a predefined code for each symbol.

**Algorithm**

# Импорт необходимых библиотек

import sys

# Функция для арифметического кодирования строки `text` с использованием словаря вероятностей `probabilities`.

def arithmeticEncoding(thisText, thisProbabilities):

    low = 0.0

    high = 1.0

    for thisChar in thisText:

        range\_ = high - low

        # Округление до 6 знаков после запятой

        high = round(low + range\_ \* thisProbabilities[thisChar]['high'], 6)

        low = round(low + range\_ \* thisProbabilities[thisChar]['low'], 6)

    # Возвращаем среднее значение последнего диапазона как закодированное число

    return round((low + high) / 2, 6)

# Функция для расчёта вероятностей для каждого символа в тексте на основе частоты их появления

def calculateProbabilities(thisText):

    thisProbabilities = {}

    totalChars = len(thisText)

    for thisChar in thisText:

        if thisChar in thisProbabilities:

            thisProbabilities[thisChar] += 1

        else:

            thisProbabilities[thisChar] = 1

    # Преобразование частот в вероятности и расчет накопленных вероятностей

    sortedProbs = sorted(thisProbabilities.items(), key=lambda item: item[1], reverse=True)

    low = 0.0

    cumProb = {}

    for thisChar, freq in sortedProbs:

        high = round(low + (freq / totalChars), 6)

        cumProb[thisChar] = {'low': round(low, 6), 'high': round(high, 6)}

        low = high

    return cumProb

# Основная функция для ввода текста и вывода результатов

def main():

    inputText = input("Введите строку для кодирования: ")

    thisProbabilities = calculateProbabilities(inputText)

    encodedValue = arithmeticEncoding(inputText, thisProbabilities)

    encodedLength = sys.getsizeof(encodedValue)

    originalLength = sys.getsizeof(inputText)

    compressionRatio = originalLength / encodedLength

    print(f"Вероятности и интервалы: {thisProbabilities}")

    print(f"Закодированное значение: {encodedValue}")

    print(f"Исходная длина: {originalLength} байт")

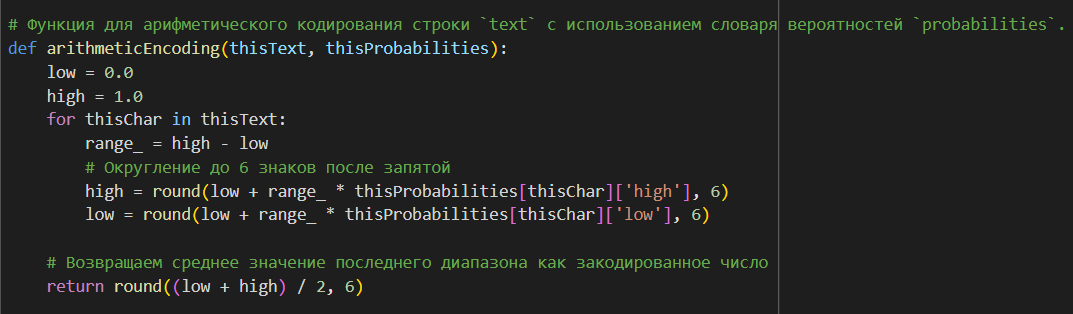
    print(f"Длина закодированного значения: {encodedLength} байт")

    print(f"Отношение сжатия: {compressionRatio:.2f}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Explanation:**

****

 **Purpose**: This function performs arithmetic encoding of the string thisText using precomputed probabilities thisProbabilities.

 **Interval Initialization**:

* The variables low and high are initially set to 0 and 1, which defines the starting range **[0, 1)**.

 **Processing Each Character**:

* For each character in the string, it calculates the current range range\_.
* The new interval is determined based on the character's probability range; low and high are adjusted accordingly.
* Values are rounded to 6 decimal places to avoid excessive precision and make the results clearer.

 **Final Value**:

* After processing all characters, the function returns the middle value of the last interval, which serves as the encoded number.

****

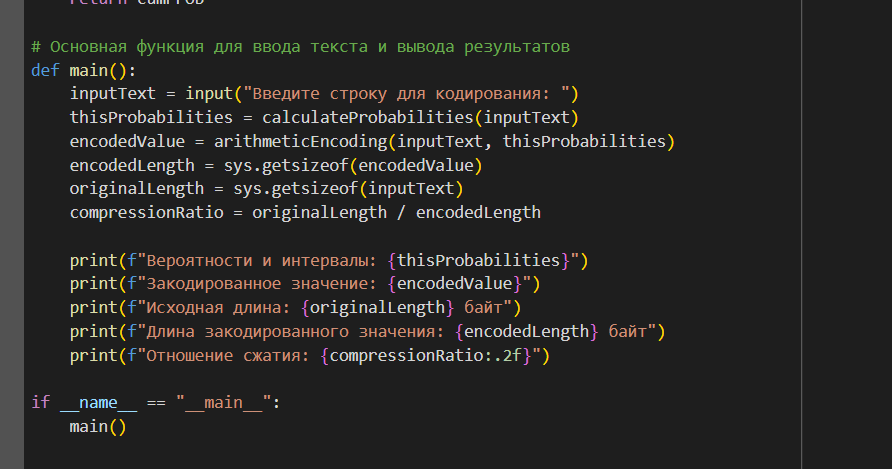
 **Purpose**: This function calculates the probabilities of characters in the string thisText and creates intervals based on their frequency of appearance.

 **Counting Character Frequencies**:

* Each character is counted, creating a dictionary thisProbabilities, where the key is the character, and the value is its occurrence count.

 **Converting Frequencies to Probabilities**:

* Frequencies are converted to probabilities by dividing by the total number of characters.
* For each character, intervals low and high are calculated and stored in a new dictionary cumProb.
* Intervals are determined such that the cumulative range covers **[0, 1)**.

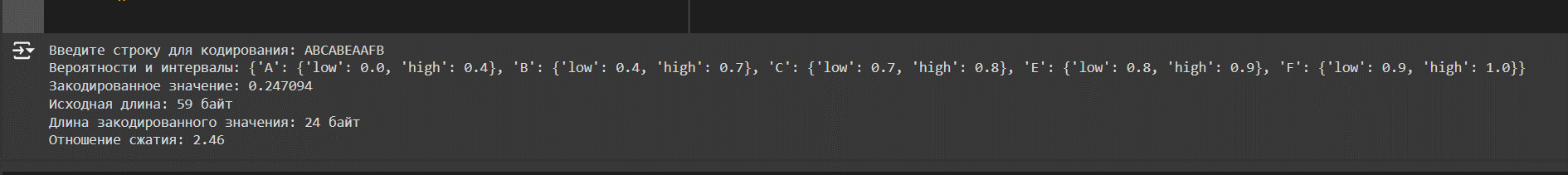
****

**Explanation:**

* **Purpose**: The main function integrates all parts of the process — from entering the string to displaying the encoded value and compression results.
* **String Input**: The user enters the string to be encoded.
* **Probability Calculation**: The function calculateProbabilities is called to compute character probabilities and their intervals.
* **String Encoding**: The arithmeticEncoding function is used to get the encoded value based on the calculated intervals.
* **Data Size Calculation**:
  + The function sys.getsizeof is used to determine the size of the original text and the encoded value.
  + Based on these, the compression ratio is calculated, showing how effective the compression was.
* **Output Results**: The probabilities, encoded value, original length, encoded length, and compression ratio are displayed.

**Test:**

**word – ‘*ABCABEAAFB*’**



**Conclusion:**

Arithmetic coding is a powerful method of compression that represents an entire string as a single number within a defined range. By using probabilities and oranges, it efficiently narrows down the interval, producing a compact and unique encoded value. For example, the word "ABCABEAAFB" was compressed into the value 0.247... which can later be decoded to retrieve the original text.