**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное автономное образовательное учреждение высшего образования

**Национальный исследовательский центр университет ИТМО**

*Мегафакультет трансляционных информационных технологий*

*Факультет информационных технологий и программирования*

**ЛАБОРАТОРНАЯ РАБОТА №2**

По дисциплине «Эффективное кодирование

Выполнил Чечулин Лев Олегович

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург, 2020г.

**Исходное изображение:**



**Фотография в заданном формате:**



**Полученная числовая последовательность:**

260 260 240 240 260 260 260 100 100 260 260 240 260 60 100 40

260 80 100 20 240 260 260 260 100 100 260 240 20 100 80 260

260 260 0 100 100 40 260 260 100 100 100 80 100 100 120 0

260 240 100 100 100 100 100 100 100 0 260 60 100 100 100 100

100 100 20 260 260 40 120 100 100 100 100 100 60 260 240 100

100 260 260 260 260 100 100 60 260 80 100 100 100 100 100 100

100 100 20 260 100 100 100 100 100 120 100 260 80 100 100 100

100 100 100 100 100 260 260 260 260 260 260 260 260 260 260 260

**Часть кода, отвечающая за считывание последовательности:**

from matplotlib import pyplot as plt

from skimage.io import imread, imshow, imsave

import math

// введение библиотек

img = imread('NC2\_2.jpg')

// считывание картинки

arr = []

// введение динамического массива arr[]

for i in range(128):

  arr.append(img[64][i])

// заполнение массива arr[] соответственными элементами средней строчки картинки

for i in range(128):

  arr[i]=int(round(arr[i]/20)\*20)

// округление элементов массива arr[] до кратных двадцати

for i in range(128):

  print(arr[i],end=' ')

  if i%16==15:

    print()

// вывод массива

**Упорядоченные символы первичного алфавита с указанием частоты встречаемости:**

0 meets 3 times out of 128

40 meets 3 times out of 128

120 meets 3 times out of 128

20 meets 4 times out of 128

60 meets 4 times out of 128

80 meets 5 times out of 128

240 meets 7 times out of 128

260 meets 41 times out of 128

100 meets 58 times out of 128

**Равномерный двоичный код:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 20 | 40 | 60 | 80 | 100 | 120 | 240 | 260 |
| 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 |

Количество символов алфавита: 9

Значение энтропии: 2.14884127076507

Расчётная длина двоичного кода: 4 бита \* 128 = 512 бит

**Код Шеннона-Фано:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A[i] | P(A[i]) | 1 шаг | 2 шаг | 3 шаг | 4 шаг | 5 шаг |
| A[1] = 0 | 3/128 | 0 | 00 | 000 | 0000 | 00000 |
| A[2] = 40 | 3/128 | 00001 |
| A[3] = 120 | 3/128 | 0001 | 00010 |
| A[4] = 20 | 4/128 | 00011 |
| A[5] = 60 | 4/128 | 001 | 0010 | 00100 |
| A[6] = 80 | 5/128 | 00101 |
| A[7] = 240 | 7/128 | 0011 | |
| A[8] = 260 | 41/128 | 01 | | | |
| A[9] = 100 | 58/128 | 1 | | | | |

**Кодовое дерево Хаффмана:**

**Код Хаффмана:**

Закодируем наш массив. Тогда алфавит будет A={0,40,120,20,60,80,240,260,100}, а набор весов (частота появления символов алфавита в кодируемом массиве) W={3,3,3,4,4,5,7,41,58}:

В дереве Хаффмана будет 9 узлов:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Узел** | **0** | **40** | **120** | **20** | **60** | **80** | **240** | **260** | **100** |
| Вес | 3 | 3 | 3 | 4 | 4 | 5 | 7 | 41 | 58 |

По алгоритму возьмем два символа с наименьшей частотой — это 0 и 40. Сформируем из них новый узел  весом 6 и добавим его к списку узлов:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Узел** | **120** | **20** | **60** | **80** | **0 и 40** | **240** | **260** | **100** |
| Вес | 3 | 4 | 4 | 5 | 6 | 7 | 41 | 58 |

Затем опять объединим в один узел два минимальных по весу узла — 120 и 20:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Узел** | **60** | **80** | **0 и 40** | **120 и 20** | **240** | **260** | **100** |
| Вес | 4 | 5 | 6 | 7 | 7 | 41 | 58 |

Затем опять объединим в один узел два минимальных по весу узла — 60 и 80:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Узел** | **0 и 40** | **120 и 20** | **240** | **60 и 80** | **260** | **100** |
| Вес | 6 | 7 | 7 | 9 | 41 | 58 |

Затем опять объединим в один узел два минимальных по весу узла — (0 и 40)

 и (120 и 20):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Узел** | **240** | **60 и 80** | **0, 20, 40 и 120** | **260** | **100** |
| Вес | 7 | 9 | 13 | 41 | 58 |

Затем опять объединим в один узел два минимальных по весу узла — 240

 и (60 и 80):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Узел** | **0, 20, 40 и 120** | **60, 80 и 240** | **260** | **100** |
| Вес | 13 | 16 | 41 | 58 |

Затем опять объединим в один узел два минимальных по весу узла —

(0, 20, 40 и 120) и (60, 80 и 240):

|  |  |  |  |
| --- | --- | --- | --- |
| **Узел** | **0, 20, 40, 60, 80, 120 и 240** | **260** | **100** |
| Вес | 29 | 41 | 58 |

Затем опять объединим в один узел два минимальных по весу узла —

(0, 20, 40, 60, 80, 120 и 240) и 260:

|  |  |  |  |
| --- | --- | --- | --- |
| **Узел** | **0, 20, 40, 60, 80, 120 и 240** | **260** | **100** |
| Вес | 29 | 41 | 58 |

Затем опять объединим в один узел два минимальных по весу узла:

|  |  |  |
| --- | --- | --- |
| **Узел** | **100** | **0, 20, 40, 60, 80, 120, 240 и 260** |
| Вес | 58 | 70 |

Объединяем последнюю пару и получаем один узел, значит, мы пришли к корню дерева Хаффмана (смотри рисуно­к). Теперь для каждого числа выберем кодовое слово (бинарная последовательность, обозначающая путь по дереву к этому символу от корня):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Символ** | **0** | **20** | **40** | **60** | **80** | **100** | **120** | **240** | **260** |
| Код | 10000 | 10011 | 10001 | 10110 | 10111 | 0 | 10010 | 1010 | 11 |

**Закодированное равномерным кодом сообщение:**

10001000011101111000100010000101010110001000011110000011010100101000010001010001011110001000100001010101100001110001010101001000100010000000010101010010100010000101010101010100010101010110000010000111010101010101010101010101010100001000001101010101010101010101010100011000100000100110010101010101010101010011100001110101010110001000100010000101010100111000010001010101010101010101010101010101000110000101010101010101010101100101100001000101010101010101010101010101010110001000100010001000100010001000100010001000

Длина кодового слова: 512 бит

Количество переданной информации: 128 пикселей

Степень сжатия сообщения: 1

**Закодированная полученным кодом Шеннона–Фано последовательность:**

01010011001101010111010100110100100100001010010110001100110101011101001100011100101010101000001100001010111100101110001000000010011111111100000010010011111100011010100001000101111100100010011110101010111001000100101111111110001101111110001010100101111111110101010101010101010101

Средняя длина кодового слова: 3,078125 бит

Количество переданной информации: 128 пикселей

Степень сжатия сообщения: 0,76953125

Относительная избыточность: 0,319

**Закодированная полученным кодом Хаффмана последовательность:**

11111010101011111100111110101110110010001111011101001110101111110011101010011010111111111100000010001111100010111001001010000111010000000010000111011000000010011111110001100100000010110111010001111111100101101110111000000001001111000001001001110111000000001111111111111111111111

Средняя длина кодового слова: 3,078125 бит

Количество переданной информации: 128 пикселей

Степень сжатия сообщения: 0,76953125

Относительная избыточность: 0,319

**Вывод:** существуют несколько возможностей сжать кодовое сообщение: путём кодирования Шеннона-Фано либо кодом Хаффмана. Это позволяет сжимать сообщение в среднем на 20-30%, что экономит затраченную память.