МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

**Лабораторная работа №2**

**По дисциплине «Введение в цифровую культуру и программирование»**

**Эффективное кодирование**

**Выполнил студент группы №M3101:**  
***Назаров Егор Александрович***

**Проверил:**  
***Хлопотов Максим Валерьевич***

**САНКТ-ПЕТЕРБУРГ**

**2019**

# введение

**Цель работы:** изучить метод оптимального кодирования Шеннона–Фано и Хаффмана.

**Задание:**

1. Взять фотографию размером 128х128 с глубиной цвета 256 градаций серого
2. Взять центральную строку пикселей T: X [x1, x2, ... xn] T и выполнить ее квантование по

формуле X = round(X / 20) \* 20. Эта строка будет представлять собой сообщение

1. Для каждого уникального символа вычислить его частоту появления в цифровой

последовательности (строке)

1. Проанализировать первичный алфавит: количество символов алфавита, значение энтропии,

среднюю минимальную длину двоичного кода

1. Построить двоичный равномерный односимвольный код
2. Сформировать коды Шеннона–Фано для выделенных символов
3. Сформировать коды Хаффмана для выделенных символов
4. Оценить среднюю длину кодовой комбинации для кодов Хаффмана и Шеннона-Фано
5. Оценить степень сжатия сообщения, закодированного равномерным односимвольным

кодом и кодом Шеннона-Фано, а также односимвольным кодом и кодом Хаффмана

1. Оценить избыточность для сформированных кодов Хаффмана и Шеннона-Фано

# Исходное и преобразованное изображение



Картинка 1. Исходная картинка



Картинка 2. Картинка 128х128 (8 bit grayscale)

# Получение цифровой последовательности

## Исходная последовательность:

[140, 120, 120, 120, 120, 120, 120, 120, 120, 120, 140, 140, 140, 120, 140, 140, 140, 140, 120, 140, 120, 120, 120, 120, 120, 120, 140, 140, 140, 140, 120, 140, 120, 100, 100, 100, 60, 80, 120, 120, 120, 100, 120, 140, 140, 140, 140, 140, 140, 140, 140, 140, 140, 140, 160, 160, 160, 160, 160, 140, 140, 140, 140, 140, 140, 140, 140, 140, 140, 140, 140, 140, 140, 140, 140, 140, 140, 120, 120, 120, 140, 140, 120, 120, 100, 80, 60, 60, 80, 80, 80, 120, 120, 60, 20, 40, 40, 80, 100, 120, 120, 120, 120, 80, 80, 80, 80, 80, 80, 100, 100, 100, 100, 100, 100, 100, 100, 120, 120, 120, 120, 120, 120, 120, 120, 120, 120, 120]

## Текст программы для получения исходной последовательности:

from skimage.io import imread # подключение библиотеки ввыода картинки  
  
line = imread("image.jpg", as\_gray=True)[64] # чтение картинки в grayscale и выдергивание из нее средней строки  
  
for i in range(128):  
 line[i] = round(int(round(line[i] \* 255))/20)\*20 # квантование строки  
  
print(str(line).replace(".", ",")) # вывод отквантованной строки

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Символ | 120 | 140 | 100 | 80 | 160 | 60 | 40 | 20 |
| Частота | 45 | 45 | 14 | 12 | 5 | 4 | 2 | 1 |
| Вероятность появления | 0,3515625 | | 0,109375 | 0,09375 | 0,0390625 | 0,03125 | 0,015625 | 0,0078125 |

Таблица 1. Частота появления каждого символа.

Длина алфавита: 8

Энтропия: 2.217196

Минимальная возможная длина строки: 284

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Символ | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 |
| Код | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |

Таблица 2. Пример двоичного равномерного кода.

Длина закодированного символа: 3

# Коды шеннона – фано

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Элемент сообщения | Вероятность появления | Деление сообщения на группы и подгруппы | | | | | | Код |
| 120 | 0,3515625 | 1 | 1 | | | | | 11 |
| 140 | 0,3515625 | 0 | | | | | 10 |
| 100 | 0,109375 | 0 | 1 | | | | | 01 |
| 80 | 0,09375 | 0 | 1 | | | | 001 |
| 160 | 0,0390625 | 0 | 1 | | | 0001 |
| 60 | 0,03125 | 0 | 1 | | 00001 |
| 40 | 0,015625 | 0 | 1 | 000001 |
| 20 | 0,0078125 | 0 | 000000 |

Таблица 3. Формирование кодов Шеннона-Фано.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Символ | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 |
| Код | 000000 | 000001 | 00001 | 001 | 01 | 11 | 10 | 0001 |

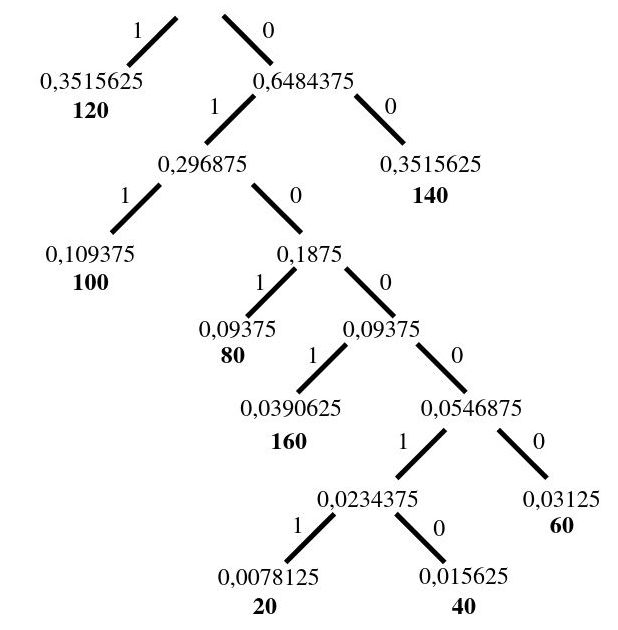
Таблица 4. Коды Шеннона-Фано.

# Код хаффмана

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Элемент сообщения | Вероятность появления (p) | P2 | P3 | P4 | P5 | P6 | P7 |
| 120 | 0,3515625 | 0,3515625 | 0,3515625 | 0,3515625 | 0,3515625 | 0,3515625 | 0,6484375 |
| 140 | 0,3515625 | 0,3515625 | 0,3515625 | 0,3515625 | 0,3515625 | 0,3515625 | 0,3515625 |
| 100 | 0,109375 | 0,109375 | 0,109375 | 0,109375 | 0,1875 | 0,296875 |
| 80 | 0,09375 | 0,09375 | 0,09375 | 0,09375 | 0,109375 |
| 160 | 0,0390625 | 0,0390625 | 0,0546875 | 0,09375 |
| 60 | 0,03125 | 0,03125 | 0,0390625 |
| 40 | 0,015625 | 0,0234375 |
| 20 | 0,0078125 |

Таблица 5. Формирование кодового дерева Хаффмана в таблице.

Примечание: в таблице желтым цветом обозначены ячейки, в которые была записана сумма двух минимальных вероятностей из предыдущего шага. Большей вероятности присваивается 0, меньшей – 1.

Рис. 1 Кодовое дерева Хаффмана

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Символ | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 |
| Код | 0100011 | 0100010 | 010000 | 0101 | 011 | 1 | 00 | 01001 |

Таблица 6. Коды Хаффмана.

# Закодированные сообщения

## Равномерный двоичный код

110101101101101101101101101101110110110101110110110110101110101101101101101101110110110110101110101100100100010011101101101100101110110110110110110110110110110110111111111111111110110110110110110110110110110110110110110110110110110101101101110110101101100011010010011011011101101010000001001011100101101101101011011011011011011100100100100100100100100101101101101101101101101101101101

Длина кода: 384 символа

Длина закодированного символа: 3

Относительная избыточность кода:

## Код Шеннона-Фано

10111111111111111111101010111010101011101111111111111010101011101101010100001001111111011110101010101010101010100001000100010001000110101010101010101010101010101010101011111110101111010010000100001001001001111100001000000000001000001001011111111100100100100100100101010101010101011111111111111111111111

Длина кода: 302 символа

Средняя длина закодированного символа: 2,359375

Степень сжатия:

Относительная избыточность кода:

## Код Хаффмана

001111111110000001000000001001111110000000010010110110110100000101111011100000000000000000000000100101001010010100101001000000000000000000000000000000000000111000011011010101000001000001010101010111010000010001010001001000100101011111101010101010101010101010101101101101101101101101111111111111

Длина кода: 294 символа

Средняя длина закодированного символа: 2,296875

Степень сжатия:

Относительная избыточность кода:

# Рассчеты кодов

Расчет кодов производился вручную с использованием таблиц 3 и 5 и рисунка 1. Ниже приведена полная программа, включая замену символов в сообщении.

from skimage.io import imread # подключение библиотеки ввыода картинки  
import collections # подключение библиотеки для определения частоты символов алфавита  
import math # подключение библиотеки с логарифмами  
  
line = imread("image 2.jpg", as\_gray=True)[64] # чтение картинки в grayscale и выдергивание из нее средней строки  
  
for i in range(128):  
 line[i] = round(int(round(line[i] \* 255)) / 20) \* 20 # квантование строки  
  
print(str(line).replace(".", ",")) # вывод отквантованной строки  
  
n\_items = list(collections.Counter(line).items()) # подсчет числа различных значений  
  
print(n\_items) # вывод алфаваита с количеством появлений  
  
entropy = 0): # Вычисление энтропии  
  
for i in range(8   
 entropy += (n\_items[i][1] / 128) \* math.log2(n\_items[i][1] / 128)  
  
entropy \*= -1  
  
print("Entropy: ", round(entropy, 6)) # Вывод энтропии  
  
line\_ShannonFano = str(line)  
line\_Haffman = str(line)  
line\_UniformBinaryCode = str(line)  
  
print(line\_ShannonFano.replace(",", "").replace(" ", "").replace("120", "11")  
.replace("140", "10").replace("100", "011").replace("80", "010").replace("160", "0001")  
.replace("60", "00001").replace("40", "000001").replace("20", "000000").replace(".", "")  
.replace("\n", "")) # замена символов на код Шеннона-Фано и вывод  
  
print(line\_Haffman.replace(",", "").replace(" ", "").replace("120", "1").replace("140", "00")  
.replace("100", "011").replace("80", "0101").replace("160", "01001").replace("60", "010000")  
.replace("40", "0100010").replace("20", "010001").replace(".", "").replace("\n", ""))  
# замена символов на код Хаффмана и вывод  
  
print(line\_UniformBinaryCode.replace(",", "").replace(" ", "").replace("120", "101")  
.replace("140", "110").replace("100", "100").replace("80", "011").replace("160", "111").replace("60", "010").replace("40", "001").replace("20", "000").replace(".", "")  
.replace("\n", "")) # замена символов на равномерный двоичный код и вывод

# Вывод

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тип кодирования | Длина кода | Средняя длина символа | Степень сжатия | Относительная избыточность |
| Равномерный код | 384 | 3 | 1 | 0,260935 |
| Код Шеннона-Фано | 302 | 2,359375 | 1,211523 | 0,060261 |
| Код Хаффмана | 294 | 2,296875 | 1,306122 | 0,03469 |

Таблица 7. Сравнение результатов применения методов кодирования.

Анализируя результаты работы, можно убедиться, что оба метода кодирования значительно эффективнее кодирования равномерным двоичным кодом. Так, алгоритм Шеннона-Фано сжал данные в 2,21 раза относительно равномерного двоичного кода, код Хаффмана оказался эффективнее кода Шеннона-Фано (коэффициент сжатия 1,31). Относительная избыточность кода Шеннона-Фано не превысила 10% и составила 6,03%, кода Хаффмана соответственно 5% и составила 3,47%. В данной работе алгоритмы сработали с приблизительно равной эффективностью, однако при больших объемах данных различие может быть выше. Это связано с более высокой оптимальностью алгоритма Хаффмана, который по сути является улучшенной версией алгоритма Шеннона-Фано.