МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИЙ

Федеральное государственное автономное образовательное учреждение высшего образования

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

Дополнительная лабораторная работа №2

По дисциплине «Введение в цифровую культуру и программирования»

Выполнил студент группы: №3112

Баатарцогт Анужин

Проверил:

Хлопотов Максим Вальеревич

САНКТ-ПЕТЕРБУРГ

2019

1. Взять цветную картинку. Это должна быть фотография.

```
In [1]: from heapq import heappush, heapify, heappop
        from collections import Counter
        from itertools import count
        from PIL import Image
        import numpy as np
        import math
```

Question 1

```
In [3]: # Take a color picture. It must be a photograph.
        pic = Image.open('zootopia.png')
        pic
```

2. *Перевести картинку в цветовое простанство greyscale.*

Question 2

```
In [6]: pic = pic.convert('L')
        pic
```

3. *Реализовать квантование с шагом 16 или 32. Шаг нужно подобрать такой, чтобы визуально изменения были не слишком заметны.*

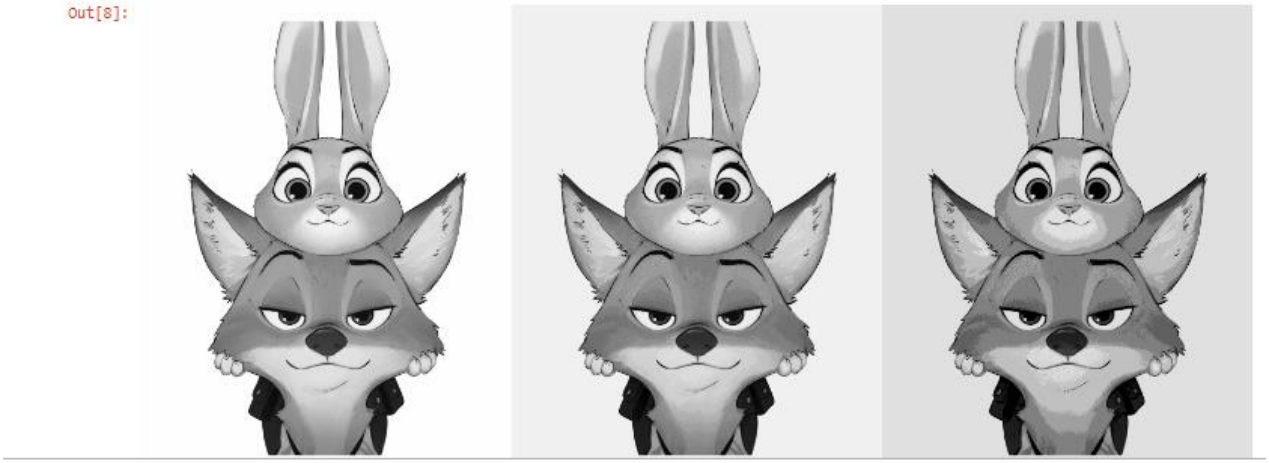
Question 3

```
In [7]: width, height = pic.size
        pic1 = Image.fromarray((np.asarray(pic) // 16) * 16)
        pic2 = Image.fromarray((np.asarray(pic) // 32) * 32)
        new_im = Image.new('LA', (width * 3, height))
        images = [pic, pic1, pic2]

        x_offset = 0
        for im in images:
            new_im.paste(im, (x_offset, 0))
            x_offset += im.size[0]
```

```
In [8]: new_im
```

Out[8]:



4. *Оценить энтропию картинки после квантования.*

$$H = -\sum p(x)\ log_2\, p(x)$$

Question 4

In [11]:
```
arr = np.asarray(pic1)
cnr = Counter()
for i in arr:
    cnr += Counter(i)
```

In [12]:
```
prob = np.array(list(cnr.values())) / sum(cnr.values())
entropy = sum([-i * math.log(i, 2) for i in prob])
entropy
```

Out[12]:
```
2.3476662620412694
```

5. *С учётом того, что до квантования на каждый пиксель отводится 8 бит, оценить максимальную степень сжатия.*

Question 5

In [15]:
```
width, height = pic.size
images = [Image.fromarray((np.asarray(pic) // 2**i) * 2**i) for i in range(9)]
new_im = Image.new('LA', (width * 9, height))

x_offset = 0
for im in images:
    new_im.paste(im, (x_offset, 0))
    x_offset += im.size[0]
```

In [16]:
```
new_im
```

Out[16]:

6. *Взять из середины картинки квадрат размером 12 на 12.*

Question 6

In [18]:
```
w, h = arr.shape
w1, h1 = w // 2, h // 2
s = arr[(w1 - 6):(w1 + 6), (h1 - 6):(h1 + 6)]
s
```

Out[18]:
```
array([[224, 224, 224, 224, 240, 224, 224, 224, 224, 224, 224, 224],
       [224, 224, 224, 224, 224, 224, 224, 224, 224, 224, 224, 224],
       [224, 224, 224, 224, 224, 224, 224, 224, 224, 224, 224, 224],
       [224, 224, 224, 224, 224, 224, 224, 224, 224, 224, 224, 224],
       [224, 208, 224, 224, 224, 224, 224, 224, 224, 224, 224, 224],
       [224, 224, 224, 208, 224, 224, 224, 224, 224, 224, 224, 224],
       [224, 224, 224, 224, 224, 224, 208, 208, 208, 224, 208, 224],
       [224, 224, 224, 224, 224, 224, 224, 208, 224, 224, 224, 224],
       [208, 224, 224, 224, 208, 208, 224, 224, 224, 224, 224, 208],
       [224, 224, 208, 208, 208, 208, 224, 224, 224, 224, 224, 224],
       [208, 224, 224, 208, 224, 208, 224, 224, 224, 224, 208, 208],
       [224, 208, 224, 208, 224, 224, 208, 208, 224, 208, 224, 208]],
      dtype=uint8)
```

7. *Закодировать пиксели кодом Хаффмана и кодом Шеннона-Фано. Сделать выводы.*

In [20]:
```
def huffman(seq, frq):
    num = count()
    trees = list(zip(frq, num, seq))
    heapify(trees)
    while len(trees) > 1:
        fa, _, a = heappop(trees)
        fb, _, b = heappop(trees)
        n = next(num)
        heappush(trees, (fa+fb, n, [a, b]))
    return trees[0][-1]
```

In [25]:
```
arr = np.asarray(s)
cnr = Counter()
for i in arr:
    cnr += Counter(i)
prob = np.array(list(cnr.values())) / sum(cnr.values())
list(cnr.keys()), list(cnr.values())
```

Out[25]:
```
([224, 240, 208], [116, 1, 27])
```

In [26]:
```
huffman(list(cnr.keys()), list(cnr.values()))
```

Out[26]:
```
[[240, 208], 224]
```

8. *Перевести исходную цветную картику в формат YUV по следующим формулам.*

RGB to YUV

$$Y = 0.299 * R + 0.587 * G + 0.114 * B$$

$$Cb = -0.1687 * R - 0.3313 * G + 0.5 * B + 128$$

$$Cr = 0.5 * R - 0.4187 * G - 0.0813 * B + 128$$

Question 8

In [32]:
```
pic_ = Image.open('zootopia.png').convert('RGB')
arr1 = np.asarray(pic_)
arr2 = np.zeros(arr1.shape, dtype = np.uint8)
```

In [34]:
```
for i in range(arr1.shape[0]):
    for j in range(arr1.shape[1]):
        R, G, B = arr1[i][j]
        Y = 0.299 * R + 0.587 * G + 0.114 * B
        Cb = -0.1687 * R - 0.3313 * G + 0.5 * B + 128
        Cr = 0.5 * R - 0.4187 * G - 0.0813 * B + 128
        arr2[i][j] = [Y, Cb, Cr]
```

In [35]:
```
Image.fromarray(arr2)
```

9. *На каналах U и V сделать прореживание (децимацию). Заменив блоки размером 2 на 2 пикселя блоками размером 1 на 1 пиксель. Для этого нужно усреднить значения пикселей в блоке 2 на 2*
10. *Оценить степень сжатия.*
11. *11. Восстановить картинку (размеры каналов U и V должны снова стать такими же как у канала Y)*
12. *Перевести обратно в формат YUV to RGB*

    $R = Y + 1.402 * (Cr - 128)$

    $G = Y - 0.34414 * (Cb - 128) - 0.71414 * (Cr - 128)$

    $B = Y + 1.772 * (Cb - 128)$
13. *Найти MSE (https://en.wikipedia.org/wiki/Mean_squared_error)*