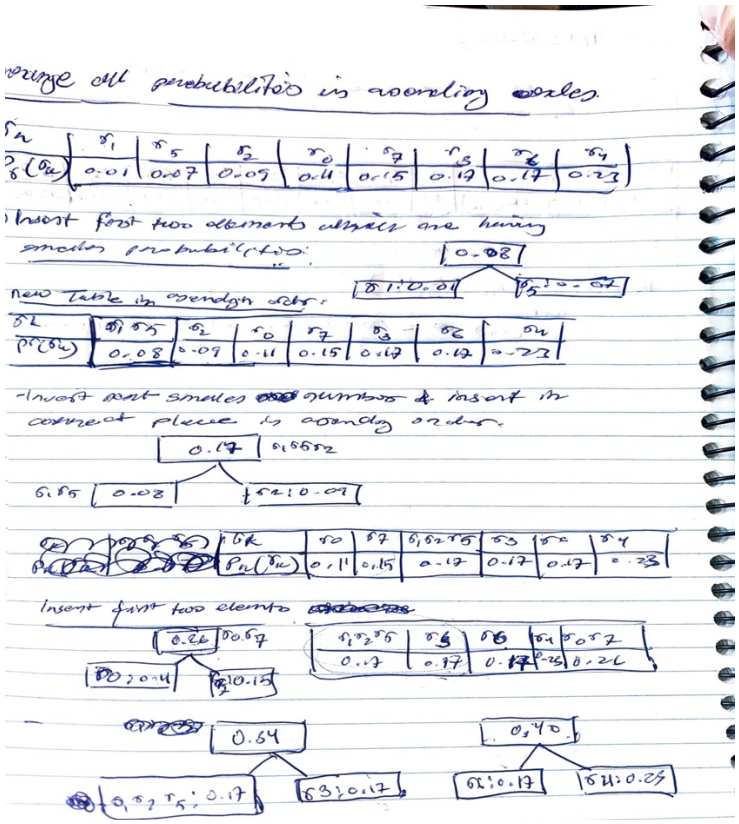
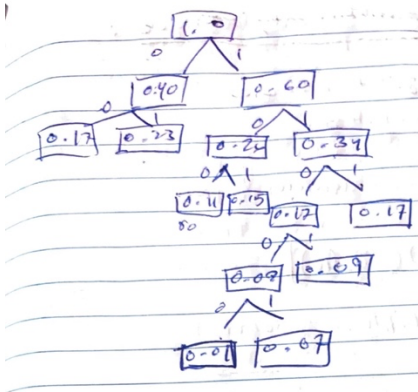
1. Construct the Huffman code of the source whose symbol probabilities are defined below. [3 points]

r_k	$p_r(r_k)$
$r_0 = 0$	0.11
$r_1 = 1/7$	0.01
$r_2 = 2/7$	0.09
$r_3 = 3/7$	0.17
$r_4 = 4/7$	0.23
$r_5 = 5/7$	0.07
$r_6 = 6/7$	0.17
$r_7 = 1$	0.15

Huffman Coding



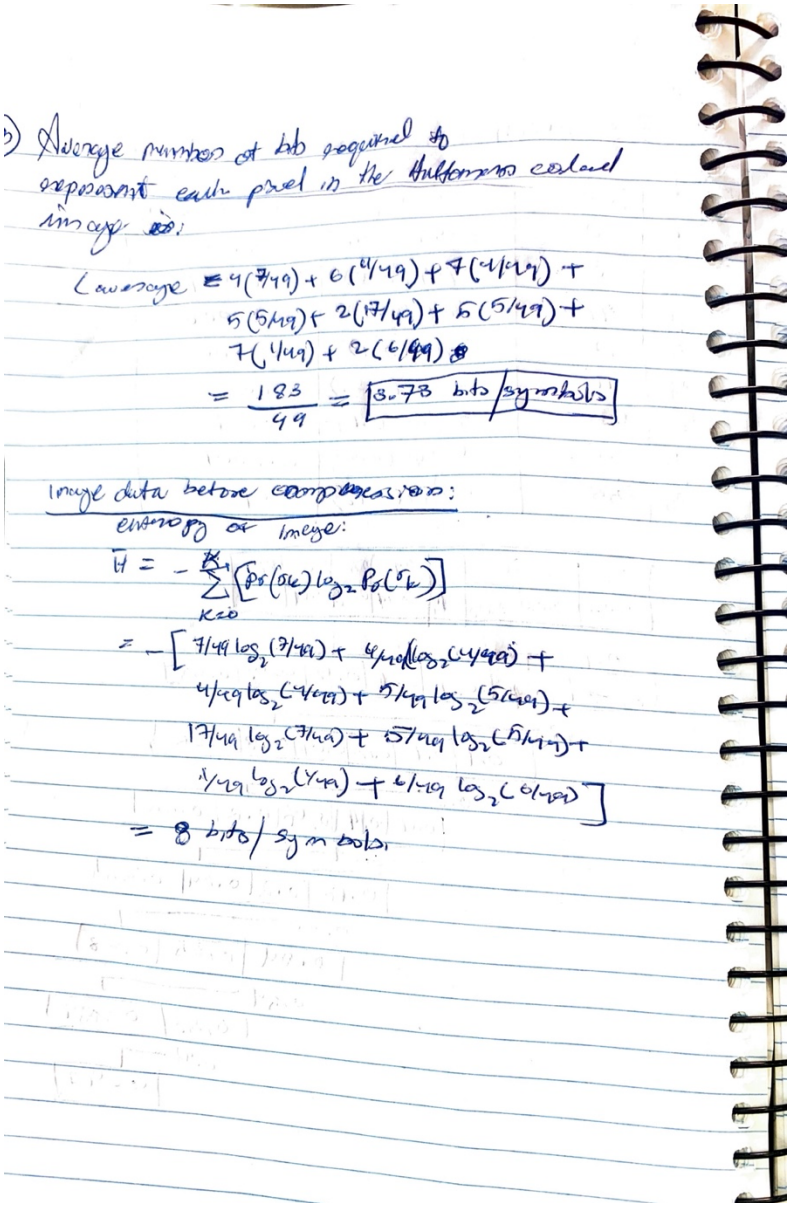


s_k	$P_S(s_k)$	code
s_0	0.11	100
s_1	0.01	11000
s_2	0.09	1101
s_3	0.17	111
s_4	0.23	01
s_5	0.07	11001
s_6	0.17	00
s_7	0.15	101

2. Consider the simple 7×7, 8-bit image: [6 points]

77	122	111	0	0	255	255
0	111	111	77	111	77	77
111	111	0	111	111	0	255
122	0	111	111	122	255	0
111	111	77	111	111	255	255
49	49	3	49	49	3	3
3	122	111	200	122	111	111

a. Compress the image using Huffman coding



b) Average number of bits required to represent each pixel in the Huffman coded image will be:

Average = 4(7/49) + 6(4/49) + 7(4/49) + 5(5/49) + 2(17/49) + 5(5/49) + 7(4/49) + 2(6/49)

= 183/49 = 3.73 bits/symbols

Image data before compression:

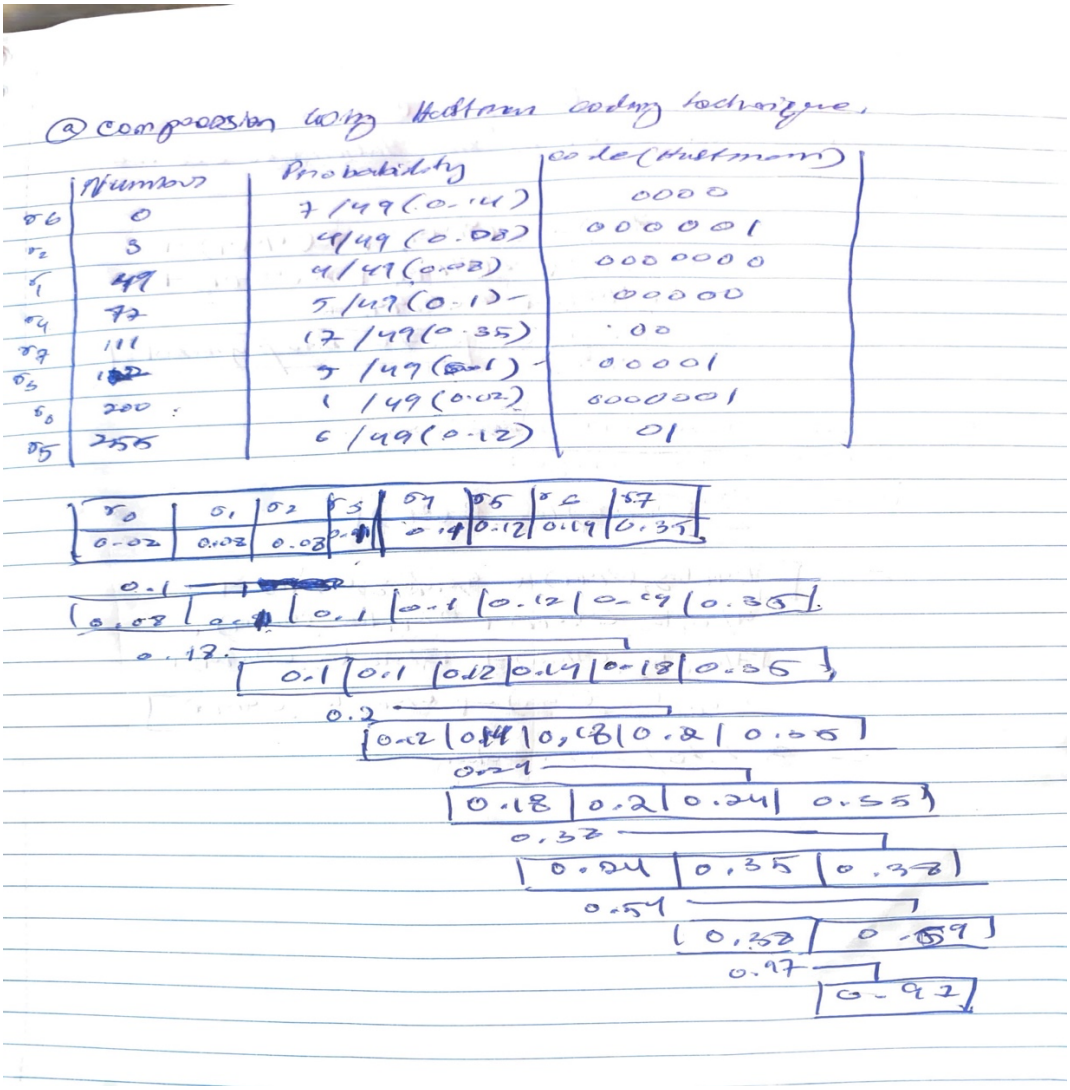
entropy of image:

$$\bar{H} = -\sum_{k=0}^{K-1} [p_s(s_k) \log_2 p_s(s_k)]$$

= −[7/49 log₂(7/49) + 4/49 log₂(4/49) + 4/49 log₂(4/49) + 5/49 log₂(5/49) + 17/49 log₂(17/49) + 5/49 log₂(5/49) + 4/49 log₂(4/49) + 4/49 log₂(6/49)]

= 8 bits/symbols

b. Compute the compression achieved and the effectiveness of Huffman coding.



Original image data was : 8-bits/symbols

Data compression using Huffman coding technique reduced the original image to

Average = 3.73 bits/symbols

Reducing the image by 4.27 bits/symbols.

3. **Write** programs to perform the following transformations on the grayscale images (you may *pick any image)*. **Display** the output images along with the respective input images. [11 points]

$$g(x,y) = \begin{cases} 0 & if\ f(x,y) < t \\ L-1 & if\ f(x,y) \geq t \end{cases} \quad \text{where } t = 70 \text{ and } 170$$

$$g(x,y) = \begin{cases} 0 & if\ f(x,y) < t1 \\ f(x,y) & if\ t1 \leq f(x,y) \leq t2 \\ 0 & if\ f(x,y) > t2 \end{cases} \quad \text{where } t1 = 70 \text{ and } t2 = 170$$

$$s = c\ \log(1+|r|)$$

$$s = c\ r^{\eta}$$

Output for:

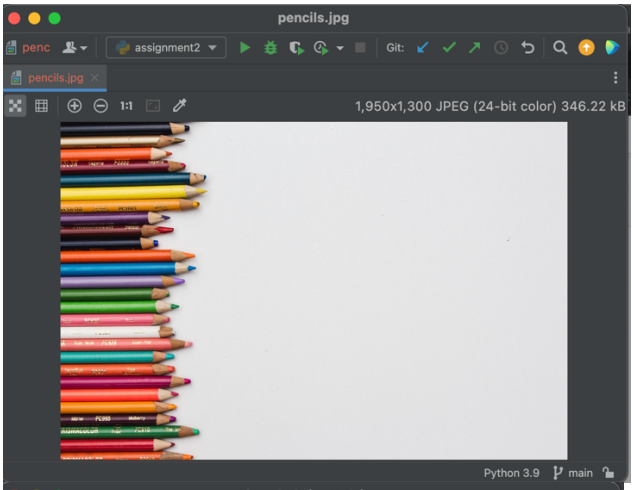
$$g(x,y) = \begin{cases} 0 & if\ f(x,y) < t \\ L-1 & if\ f(x,y) \geq t \end{cases} \quad \text{where } t = 70 \text{ and } 170$$

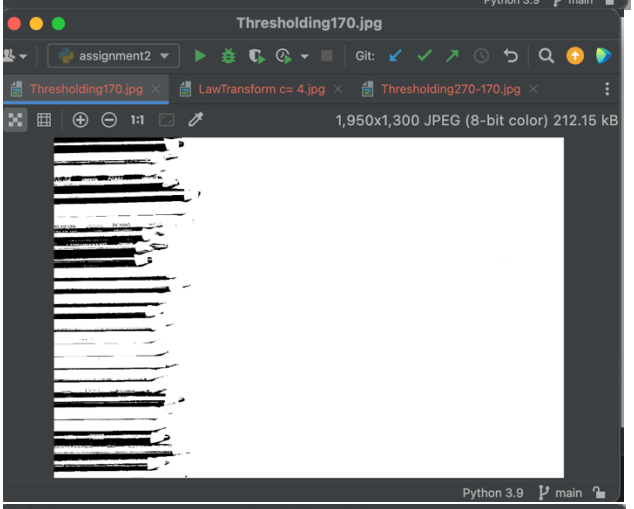
cis465 – assignment2.py

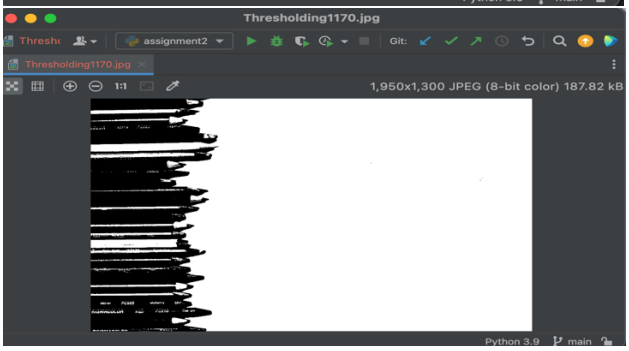
assignment2.py

```
import cv2
import numpy as np
image = cv2.imread("pencils.jpg")
grayscale = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

height, width = grayscale.shape
def Thresholding1(t):
    ouput_image = np.zeros((height, width))
    for i in range(0, height - 1):
        for j in range(0, width - 1):
            if (grayscale[i][j] < t):
                ouput_image[i][j] = 0
            elif (grayscale[i][j] >= t):
                ouput_image[i][j] = 255
    filename = "Thresholding1" + str(t) + ".jpg"
    cv2.imwrite(filename, ouput_image)
```

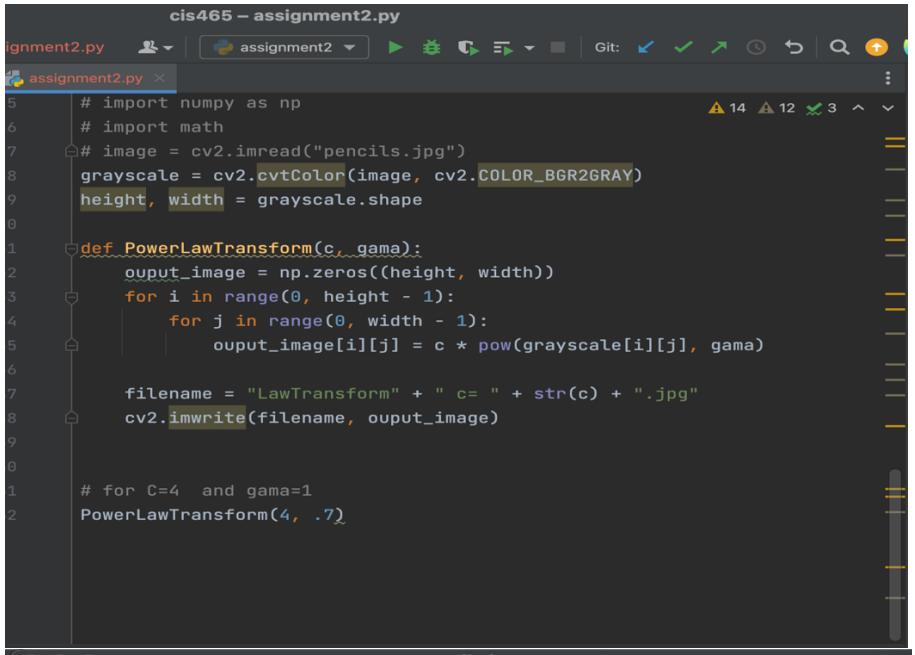

pencils.jpg
assignment2
Git:
pencils.jpg
1:1
1,950x1,300 JPEG (24-bit color) 346.22 kB
Python 3.9
main

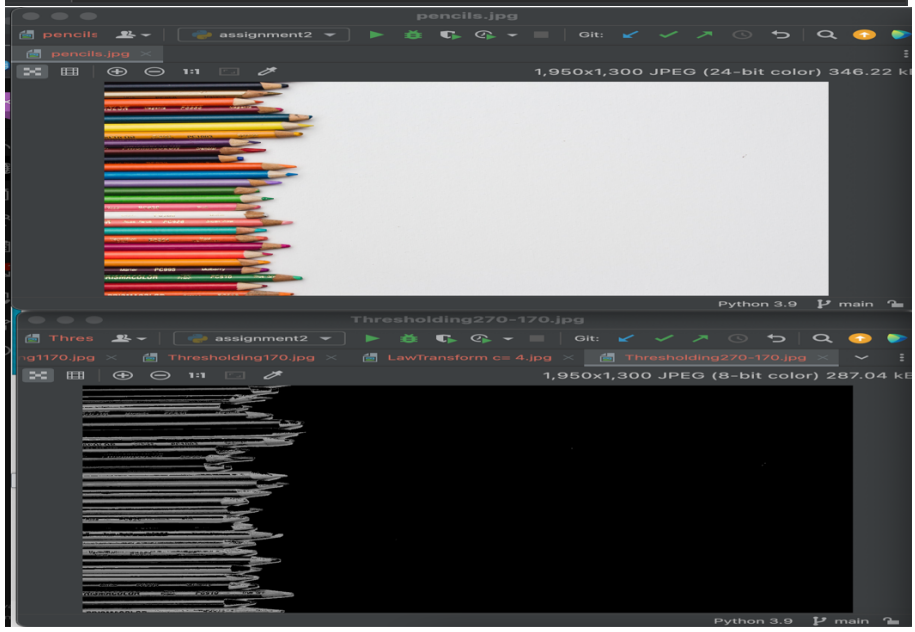

Thresholding170.jpg
assignment2
Git:
Thresholding170.jpg
LawTransform c= 4.jpg
Thresholding270-170.jpg
1:1
1,950x1,300 JPEG (8-bit color) 212.15 kB
Python 3.9
main


Thresholding1170.jpg
assignment2
Git:
Thresholding1170.jpg
1:1
1,950x1,300 JPEG (8-bit color) 187.82 kB
Python 3.9
main

Output for:

$$g(x,y)=\begin{cases}0 & if\ f(x,y)<t1\\ f(x,y) & if\ t1\le f(x,y)\le t2\\ 0 & if\ f(x,y)>t2\end{cases}\qquad \text{where } t1=70 \text{ and } t2=170$$





Output for:

$$s = c \log(1+|r|)$$

$$s = c\ r^{\eta}$$

cis465 – assignment2.py

```
grayscale = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

height, width = grayscale.shape


def Thresholding2(t1, t2):
    ouput_image = np.zeros((height, width))
    for i in range(0, height - 1):
        for j in range(0, width - 1):
            if (grayscale[i][j] < t1):
                ouput_image[i][j] = 0
            elif (t1 <= grayscale[i][j] <= t2):
                ouput_image[i][j] = grayscale[i][j]
            elif (grayscale[i][j] > t2):
                ouput_image[i][j] = 0

    filename = "Thresholding2" + str(t1) + "-" + str(t2) + ".jpg"
    cv2.imwrite(filename, ouput_image)


Thresholding2(70, 170)
# import cv2
```