CSCI576 Homework#3

Yifan Wang

3038184983

wang608@usc.edu

Question 1: DCT Coding *(15 points)*

In this question you will try to understand the working of DCT in the context of JPEG. Below is an 8x8 luminance block of pixel values and its corresponding DCT coefficients.

188	180	155	149	179	116	86	96
168	179	168	174	180	111	86	95
150	166	175	189	165	101	88	97
163	165	179	184	135	90	91	96
170	180	178	144	102	87	91	98
175	174	141	104	85	83	88	96
153	134	105	82	83	87	92	96
117	104	86	80	86	90	92	103

- Using the 2D DCT formula, compute the 64 DCT values. Assume that you quantize your DCT coefficients uniformly with Q=100. What does your table look like after quantization? *(2 points)*

In [1]:
```
import numpy as np
from scipy.fftpack import dct, idct
import matplotlib.pyplot as plt

a = np.array([[188, 180, 155, 149, 179, 116, 86, 96,
               [168, 179, 168, 174, 180, 111, 86, 95,
               [150, 166, 175, 189, 165, 101, 88, 97,
               [163, 165, 179, 184, 135,  90, 91, 96,
               [170, 180, 178, 144, 102,  87, 91, 98,
               [175, 174, 141, 104,  85,  83, 88, 96,
               [153, 134, 105,  82,  83,  87, 92, 96,
               [117, 104,  86,  80,  86,  90, 92, 103]])

def dct2d(x):
    return dct(dct(x,norm='ortho', axis=0),norm='ortho',axis=1)
def idct2d(x):
    return idct(idct(x,norm='ortho', axis=0),norm='ortho',axis=1)

a = a.reshape(8,8)
ta = dct2d(a)
ta = np.round(ta/100).astype('int16')
print('quantized coef')
print(ta)
```

```
quantized coef
[[10  2  0  0  0  0  0  0]
 [ 1  1 -1  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]]
```

- In the JPEG pipeline, the quantized DCT values are then further scanned in a zigzag order. Show the resulting zigzag scan when Q=100, *(1 point)*.

In [2]:
```
class ZigZag():
    def __init__(self):
        self.idx = []

    def zig_zag(self, i, j, n):
        if i + j >= n:
            return n * n - 1 - self.zig_zag(n - 1 - i, n - 1 - j, n)
        k = (i + j) * (i + j + 1) // 2
        return k + i if (i + j) & 1 else k + j

    def zig_zag_getIdx(self, N):
        idx = np.zeros((N, N))
        for i in range(N):
            for j in range(N):
                idx[i, j] = self.zig_zag(i, j, N)
        return idx.reshape(-1)

    def transform(self, X):
        self.idx = self.zig_zag_getIdx((int)(np.sqrt(X.shape[-1]))).astype('int32')
        S = list(X.shape)
        X = X.reshape(-1, X.shape[-1])
        return X[:, np.argsort(self.idx)].reshape(S)

    def inverse_transform(self, X):
        self.idx = self.zig_zag_getIdx((int)(np.sqrt(X.shape[-1]))).astype('int32')
        S = list(X.shape)
        X = X.reshape(-1, X.shape[-1])
        return X[:, self.idx].reshape(S)
b = ZigZag().transform(ta.reshape(-1))
print("after zig-zag")
print(b)
```

```
after zig-zag
[10  2  1  0  1  0 -1  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
```

- For this zigzag AC sequence, write down the intermediary notation *(2 points)*

`<0,2>(2), <0,1>(1), <1,1>(1), <2,1>(-1), <9,1>(1), <0,0>`

- Assuming these are luminance values, write down the resulting JPEG bit stream. For the bit stream you may consult standard JPEG VLC and VLI code tables. You will need to refer to the code tables from the ITU-T JPEG standard which also uploaded with your assignment. *(2 points)*

```
<0,2> : 01
2     : 10
<0,1> : 00
1     : 1
<1,1> : 1100
1     : 1
<2,1> : 11011
-1    : 0
<9,1> : 111111000
1     : 1
<0,0> : 1010
```

Then, the total bit stream: `01 10 00 1 1100 1 11011 0 111111000 1 1010`

- What compression ratio do you get for this luminance block ?

if we ignore the DC, the compression ratio:

$$63 * 8 : 32 = 15.75 : 1$$

Question 2: Image Dithering *(25 points)*

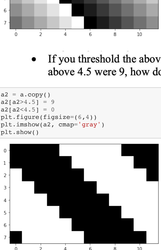
Let's say that we have an original image of size mxn with 8 bits per pixel. A certain sub section of this image matrix is represented in a 12x8 sub image below and the values are normalized. In the normalized representation shown below, assume that 0 corresponds to white and 9 corresponds to black.

1	2	3	4	5	6	7	8	9	0	1	2
0	1	2	3	4	5	6	7	8	9	0	1
9	0	1	2	3	4	5	6	7	8	9	0
8	9	0	1	2	3	4	5	6	7	8	9
7	8	9	0	1	2	3	4	5	6	7	8
6	7	8	9	0	1	2	3	4	5	6	7
5	6	7	8	9	0	1	2	3	4	5	6
4	5	6	7	8	9	0	1	2	3	4	5

Answer the following questions You may want to code/script a process to generate the final outputs, but only final outputs are expected. Also, we have asked you to plot this 12x8 image block as a gray color image, and its processed outputs as binary black/white images. For reasons of visible clarity (because a 12x8 image block is very small) you may want to show a zoomed or magnified picture.

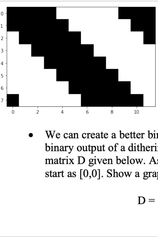
- Plot the image as an 8-bit gray scale map, that is - create a 12x8 image to show the original gray image block. *(2 points)*

In [3]:
```
a = np.array([[1,2,3,4,5,6,7,8,9,0,1,2],
              [0,1,2,3,4,5,6,7,8,9,0,1],
              [9,0,1,2,3,4,5,6,7,8,9,0],
              [8,9,0,1,2,3,4,5,6,7,8,9],
              [7,8,9,0,1,2,3,4,5,6,7,8],
              [6,7,8,9,0,1,2,3,4,5,6,7],
              [5,6,7,8,9,0,1,2,3,4,5,6],
              [4,5,6,7,8,9,0,1,2,3,4,5]])
na = (a-np.min(a))/(np.max(a)-np.min(a))*255
plt.figure(figsize=(6,4))
plt.imshow(na, cmap='gray')
plt.show()
```



- If you threshold the above 12x8 image block such that all values below 4.5 were 0 and above 4.5 were 9, how does your output image B/W block look? Plot an image (*3 points*)

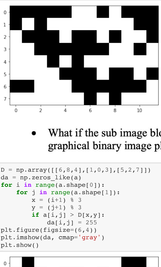
In [4]:
```
a2 = a.copy()
a2[a2>4.5] = 9
a2[a2<4.5] = 0
plt.figure(figsize=(6,4))
plt.imshow(a2, cmap='gray')
plt.show()
```



- We can create a better binary image output by using a dithering matrix. Compute the binary output of a dithering operation on the gray level 12x8 image using the dithering matrix D given below. Assume that the sub image block's top left coordinates indexes start as [0,0]. Show a graphical binary image plot of the dithered output. *(5 points)*

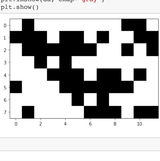
$$D = \begin{bmatrix} 6 & 8 & 4 \\ 1 & 0 & 3 \\ 5 & 2 & 7 \end{bmatrix}$$

In [5]:
```
D = np.array([[6,8,4],[1,0,3],[5,2,7]])
da = np.zeros_like(a)
for i in range(a.shape[0]):
    for j in range(a.shape[1]):
        x = i % 3
        y = j % 3
        if a[i,j] > D[x,y]:
            da[i,j] = 255
plt.figure(figsize=(6,4))
plt.imshow(da, cmap='gray')
plt.show()
```



- What if the sub image block's top left coordinate indexes start with [1,1]. Show a graphical binary image plot of the dithered output. *(5 points)*

In [6]:
```
D = np.array([[6,8,4],[1,0,3],[5,2,7]])
da = np.zeros_like(a)
for i in range(a.shape[0]):
    for j in range(a.shape[1]):
        x = (i+1) % 3
        y = (j+1) % 3
        if a[i,j] > D[x,y]:
            da[i,j] = 255
plt.figure(figsize=(6,4))
plt.imshow(da, cmap='gray')
plt.show()
```



In [ ]:
```
```