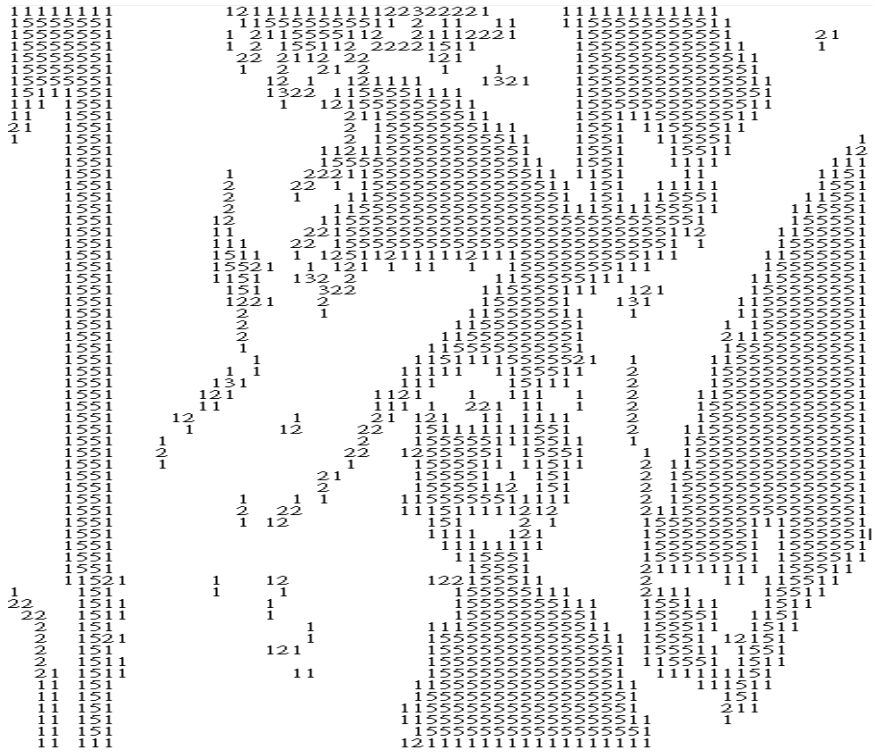
HW6 report

R10922082 林育駿

本次作業中使用 python opencv 幫助圖片的 I/O 一些處理部分，在有 lena.bmp 同目錄下執行 R10922082_HW6.py 會生成 lena_yokoi_connectivity_number.txt 存有 64*64 matrix 的資訊。

Write a program which counts the Yokoi connectivity number on a downsampled image(lena.bmp).



- Binarized

```python
def binarized_lena(img):


    # get image information
    (imgHeight, imgWidth) = img.shape[:2]

    # init a white canvas
    binimg = np.zeros(shape=(imgWidth,imgHeight,3))
    # manipulate pixels
    for i in range(imgHeight):
        for j in range(imgWidth):
            if  img[i][j].mean() >=128:
                binimg[i][j]= np.array([255,255,255])
            else:
                binimg[i][j]= np.array([0,0,0])

    binimg = binimg.astype(np.uint8) #set the right data type
    cv2.imwrite("binarized_lena.bmp", binimg) #write the output of image
    return binimg
```

和之前 HW2 相同作法，將原本像素值 0-255 以 128 為界轉成 binary 的形式。

- Downsample

```
def down_sample(img):
    # take every top-left one to represent 8 * 8 original is 512 * 512
    # get image information
    (imgHeight, imgWidth) = img.shape[:2]
    bs = 8 # block size
    # init a white canvas
    dimg = np.zeros(shape=(imgWidth//bs,imgHeight//bs,3))
    (imgHeight, imgWidth) = dimg.shape[:2] # new width and height
    # manipulate pixels
    for i in range(imgHeight):
        for j in range(imgWidth):

            dimg[i][j]= img[i*bs][j*bs]

    dimg = dimg.astype(np.uint8) #set the right data type
    # cv2.imwrite("down_sample_lena.bmp", dimg) #write the output of image
    # print(dimg.shape)
    return dimg
```

將原本 512*512 的圖以 8*8 為一個 block 單元取左上角值為 downsample 的值，輸出 64*64 的圖作為後續計算 connectivity 的矩陣。

- Yokoi connectivity matrix

```
def yokoiCnum(img):

    # image to label matrix
    global labelm
    (imgHeight, imgWidth) = img.shape[:2]
    labelm = np.zeros((imgWidth +2,imgHeight +2)) # for label with padding
    yokoim = np.zeros((imgWidth ,imgHeight)) # for result
    for i in range(imgHeight):
        for j in range(imgWidth):
            if img[i][j].mean() == 0:
                labelm[i+1][j+1] = 0
            else:
                labelm[i+1][j+1] = 1
    labelm = labelm.astype(np.int)
    # do algorithm
    for i in range(imgHeight):
        for j in range(imgWidth):
            yokoim[i][j] = f(i+1,j+1) #map to label matrix position

    yokoim = yokoim.astype(np.int)
```

先對 downsample 後的 label matrix 進行邊界的補零作為背景可以讓原本的邊界也能使用公式計算，再依照定 f 和 h 公式來對每點計算 connectivity number，最後輸出。

h function:

```
def h(y,x,c,d,e):
    global labelm
    (cx,cy) = index2pos(c)
    (dx,dy) = index2pos(d)
    (ex,ey) = index2pos(e)

    if labelm[y][x] == 1: # center exist
        # checl b != c
        if labelm[y][x] != labelm[y+cy][x+cx]:
            return 0 #s
        else:
            if labelm[y][x] ==labelm[y+ey][x+ex] and labelm[y+dy][x+dx] ==labelm[y][x]
                return 2 #r
            else:
                return 1 #q
```

f function:

```
def f(y,x):

    a1 = h(y,x,1,6,2)
    a2 = h(y,x,2,7,3)
    a3 = h(y,x,3,8,4)
    a4 = h(y,x,4,5,1)

    if isR(a1) and isR(a2) and isR(a3) and isR(a4):
        return 5 # internal
    else:
        return isQ(a1) + isQ(a2) + isQ(a3) + isQ(a4)
```