HW6

- **A discription of your homework**

  Programming language used: Python 2.7

  Library used: Numpy, PIL

- **Your parameters**

  i: row

  j: column

  tem: 用於儲存每個像素的灰階數值 0~255

  a: 計算 Yokoi connectivity number 時的a_1, a_2, a_3, a_4

- **Function**

  bi: binarize Lena

  unit: downsample Lena from 512x512 to 64x64

  yokoi: calculate the Yokoi connectivity number

- **The algorithm you used**

1. Binarize Lena (512x512)

   Threshold: 128

   gray scale >= 128 → white(1)

   gray scale < 128 → black(0)

2. Using 8x8 blocks as a unit, take the topmost-left pixel as the downsampled data

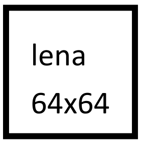
   → Downsampling Lena from 512x512 to 64x64

   取原圖 i,j 為 8 的倍數的像素

   & add one pixel of background to the edge of the 64x64 image

   → for convenience of calculating the yokoi number

   i,j 為 0 或 65 時，灰階值給 0



3. Calculate the Yokoi connectivity number

   for 4-connectivity:

$$h(b,c,d,e) = \begin{cases} q & if\ b = c\ and\ (d \neq b\ \vee e \neq b) \\ r & if\ b = c\ and\ (d = b \wedge e = b) \\ s & if\ b \neq c\ and\ (d = b \wedge e = b) \end{cases}$$

$$f(b,c,d,e) = \begin{cases} 5 & if\ a_1 = a_2 = a_3 = a_4 = r \\ n & where\ n = \#\ \{a_k | a_k = q\}\ ,\ otherwise \end{cases}$$

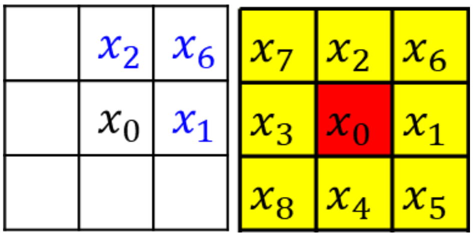
- $a_1 = h(x_0, x_1, x_6, x_2)$

  $a_2 = h(x_0, x_2, x_7, x_3)$

  $a_3 = h(x_0, x_3, x_8, x_4)$

  $a_4 = h(x_0, x_4, x_5, x_1)$

- $output = f(a_1, a_2, a_3, a_4)$



	x_2	x_6
	x_0	x_1

x_7	x_2	x_6
x_3	x_0	x_1
x_8	x_4	x_5

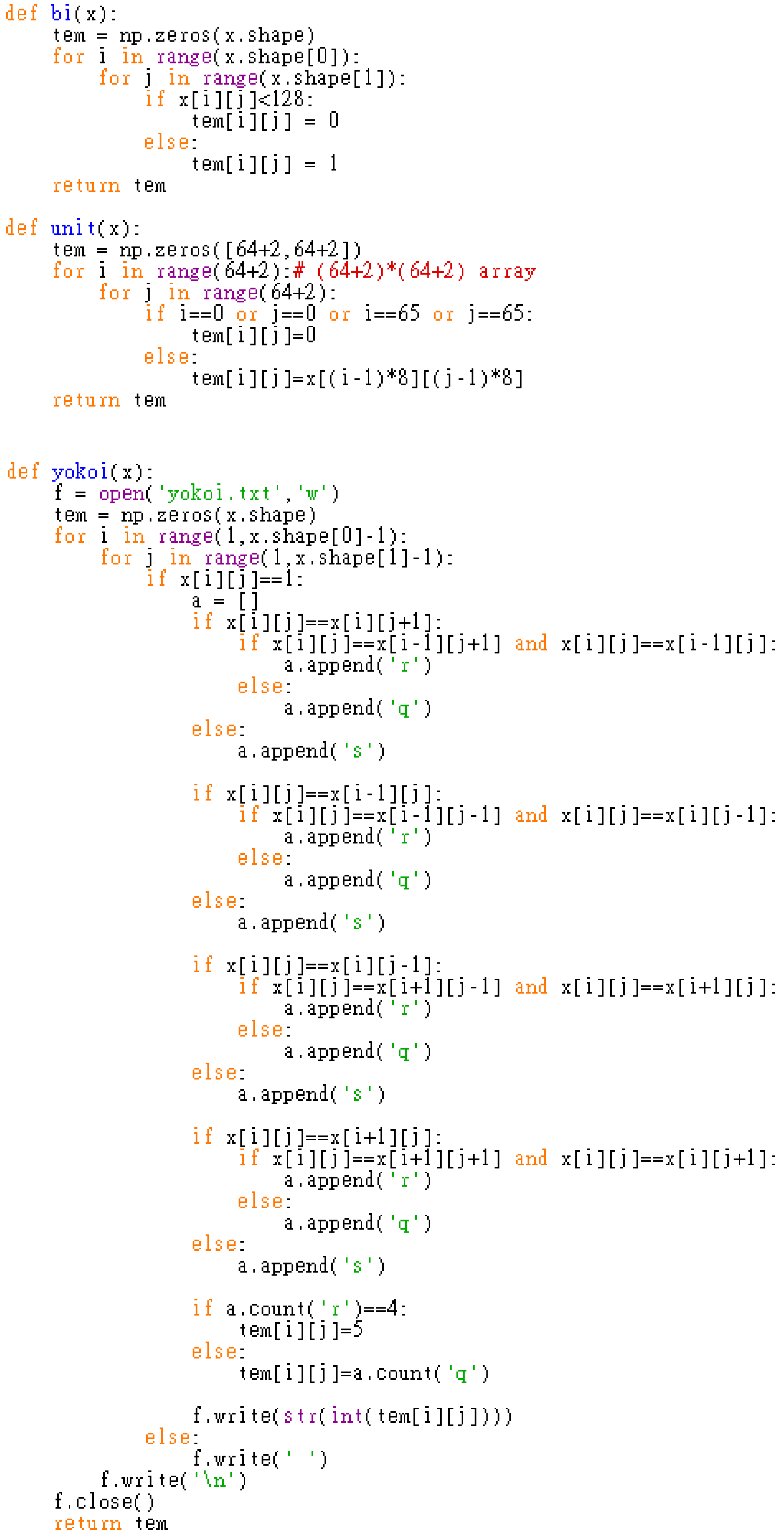
(1) 若該像素為物件，則按照其相鄰像素的異同，依次計算$\mathrm{a}_1, \mathrm{a}_2, \mathrm{a}_3, \mathrm{a}_4$(給出 r,q,s)，而$x_0$~$x_8$對應到下表的 i,j。

(2) 最後計算$f(a_1, a_2, a_3, a_4)$，若 r=4，則在該像素給出 Yokoi connectivity number = 5，其他情況則輸出 q 的數量。

(3) 輸出結果到 yokoi.txt。

(i-1,j-1)	(i-1,j)	(i-1,j+1)
(i,j-1)	(i,j)	(i,j+1)
(i+1,j-1)	(i+1,j)	(i+1,j+1)

- **Principal code fragment**



```
def bi(x):
    tem = np.zeros(x.shape)
    for i in range(x.shape[0]):
        for j in range(x.shape[1]):
            if x[i][j]<128:
                tem[i][j] = 0
            else:
                tem[i][j] = 1
    return tem

def unit(x):
    tem = np.zeros([64+2,64+2])
    for i in range(64+2):# (64+2)*(64+2) array
        for j in range(64+2):
            if i==0 or j==0 or i==65 or j==65:
                tem[i][j]=0
            else:
                tem[i][j]=x[(i-1)*8][(j-1)*8]
    return tem


def yokoi(x):
    f = open('yokoi.txt','w')
    tem = np.zeros(x.shape)
    for i in range(1,x.shape[0]-1):
        for j in range(1,x.shape[1]-1):
            if x[i][j]==1:
                a = []
                if x[i][j]==x[i][j+1]:
                    if x[i][j]==x[i-1][j+1] and x[i][j]==x[i-1][j]:
                        a.append('r')
                    else:
                        a.append('q')
                else:
                    a.append('s')

                if x[i][j]==x[i-1][j]:
                    if x[i][j]==x[i-1][j-1] and x[i][j]==x[i][j-1]:
                        a.append('r')
                    else:
                        a.append('q')
                else:
                    a.append('s')

                if x[i][j]==x[i][j-1]:
                    if x[i][j]==x[i+1][j-1] and x[i][j]==x[i+1][j]:
                        a.append('r')
                    else:
                        a.append('q')
                else:
                    a.append('s')

                if x[i][j]==x[i+1][j]:
                    if x[i][j]==x[i+1][j+1] and x[i][j]==x[i][j+1]:
                        a.append('r')
                    else:
                        a.append('q')
                else:
                    a.append('s')

                if a.count('r')==4:
                    tem[i][j]=5
                else:
                    tem[i][j]=a.count('q')

                f.write(str(int(tem[i][j])))
            else:
                f.write(' ')
        f.write('\n')
    f.close()
    return tem
```

- **Resulting images**

```
11111111        1211111111122322221      111111111111        0  0
15555551         115555555511 2 11  11  1155555555511         0
15555551        1 2115555112   21112221   155555555551      21
15555551        1 2 155112 22221511      1555555555511     1
15555551         22 2112 22   121 0 0   15555555555511   0
15555551         1  2  21 2     1   1     15555555555551  0
15555551           12 1  121111    1321   155555555555511
15111551           1322 1155551111        155555555555551
111 1551            1  121555555511       155555555555511
11  1551                 21155555511      1551115555511
21  1551                 2 15555555111    1551 11555511
1   1551                 2 155555555511   1551  115551         1
    1551               1121155555555551   1551   15511        12
    1551               15555555555555511  1551   1111        111
    1551        1     2221155555555555511 1151    11        1151
    1551        2    22 1 155555555555511 151  11111       1551
    1551        2    1   115555555555555551 151 115551    11551
    1551        2       11555555555555555511511155511      115551
    1551       12      1155555555555555555555555551        155551
    1551       11   0 2215555555555555555555555112       1155551
    1551       111  22 155555555555555555555551 1       1555551
    1551       1511  1 125112111121115555555555111      11555551
    1551       15521  1 121 1 11 1  15555555111  0     15555551
    1551       1151  132 2          1155555111   0     115555551
    1551        151 0  322         115555111  121      155555551
    1551        1221   2           1555551   131      1155555551
    1551         2  0  1          115555511   1       1155555551
    1551         2   0      0    1155555551  0       1 155555551
    1551         2              11555555551          21155555551
    1551         1  0          115555555551          15555555551
    1551          1           11511115555521  1     115555555551
    1551        1 1          11111  1155511   2     155555555551
    1551       131           111     15111    2     155555555551
    1551      121 0        1121   1  111  1   2    1155555555551
    1551      11           111 1  221 11 1   2    1555555555551
    1551    12  0    1     21 121  11 1111   2    1555555555551
    1551     1      12    22  151111111551    2   11555555555551
    1551   1              2   1555551115511   1   15555555555551
    1551   2    0     0  22  12555551 15551    1  15555555555551
    1551   1             1    1555511 11511    2 115555555555551
    1551      0 0      21     155551 1 151    2 155555555555551
    1551               2      15555112 151     2 155555555555551
    1551         1   1 1     1155555511111     2 155555555555551
    1551         2  22       111511111212      2115555555555555551
    1551  0      1 12          151    2 1     15555555111555551
    1551       0  0  0         1111  121       155555551 1555551
    1551            0           11111111       155555551 1555551
    1551        0                115551        155555551 1555511
    1551                          15551        211111111 155511
    11521      1   12          122155511       2     11 115511
1    151 0     1    1            155555111     2111     15511
22   1511          1             15555555111   155111    1511
 22  1511          1             15555555551   155551  1151
  2  151            0 1        11155555555511  155511  1511
  2  1521    0        1        155555555555511 15551 12151
  2  151           121         155555555555551 155511 1551
  2  1511                   0  155555555555551 115551 1511
  21 1511            11        155555555555551  111111151
  11 151          0           11555555555555511    111511
  11 151                      15555555555555551      151
  11 151            0        115555555555555551      211
  11 151                     115555555555555511      1
  11 151                    0 155555555555555551
  11 111        0            121111111111111111
```