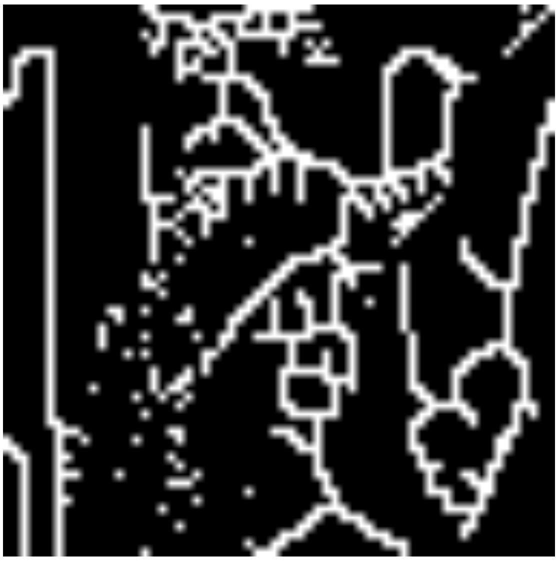
Computer Vision Homework #7

Student-ID: r10944020; Name: 林顥倫; Department: GINM

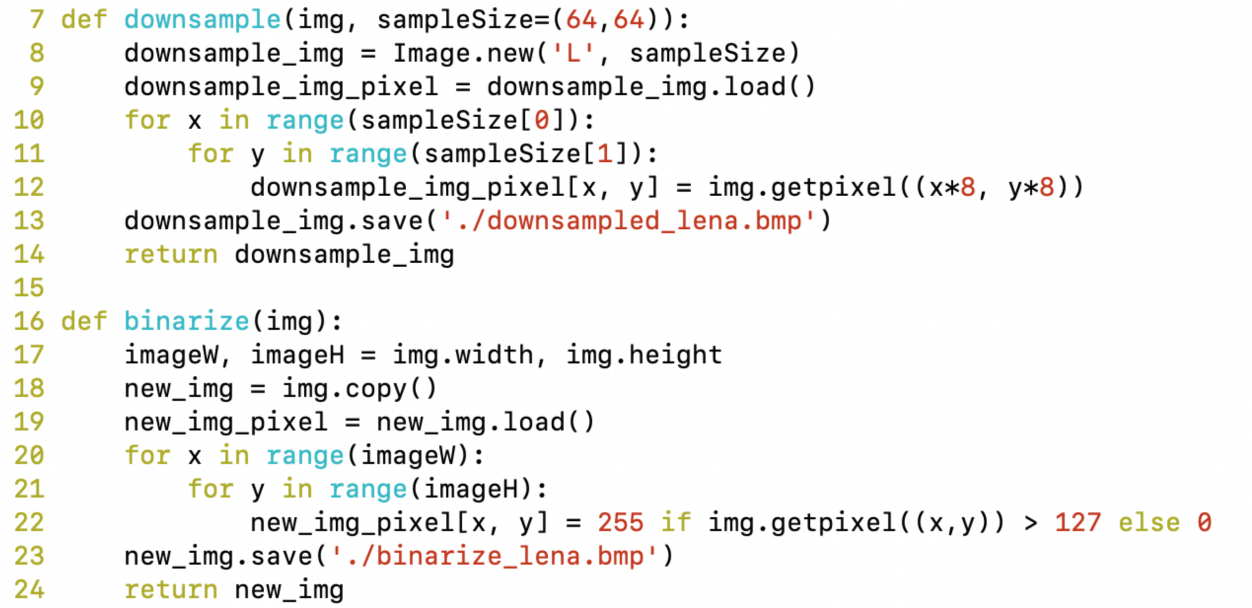
**[Results] Thinning Operation**

(此圖片結果經過放大，可能造成解析度模糊，建議參考程式碼運行結果)



**[Code Fragment & Explanation]**

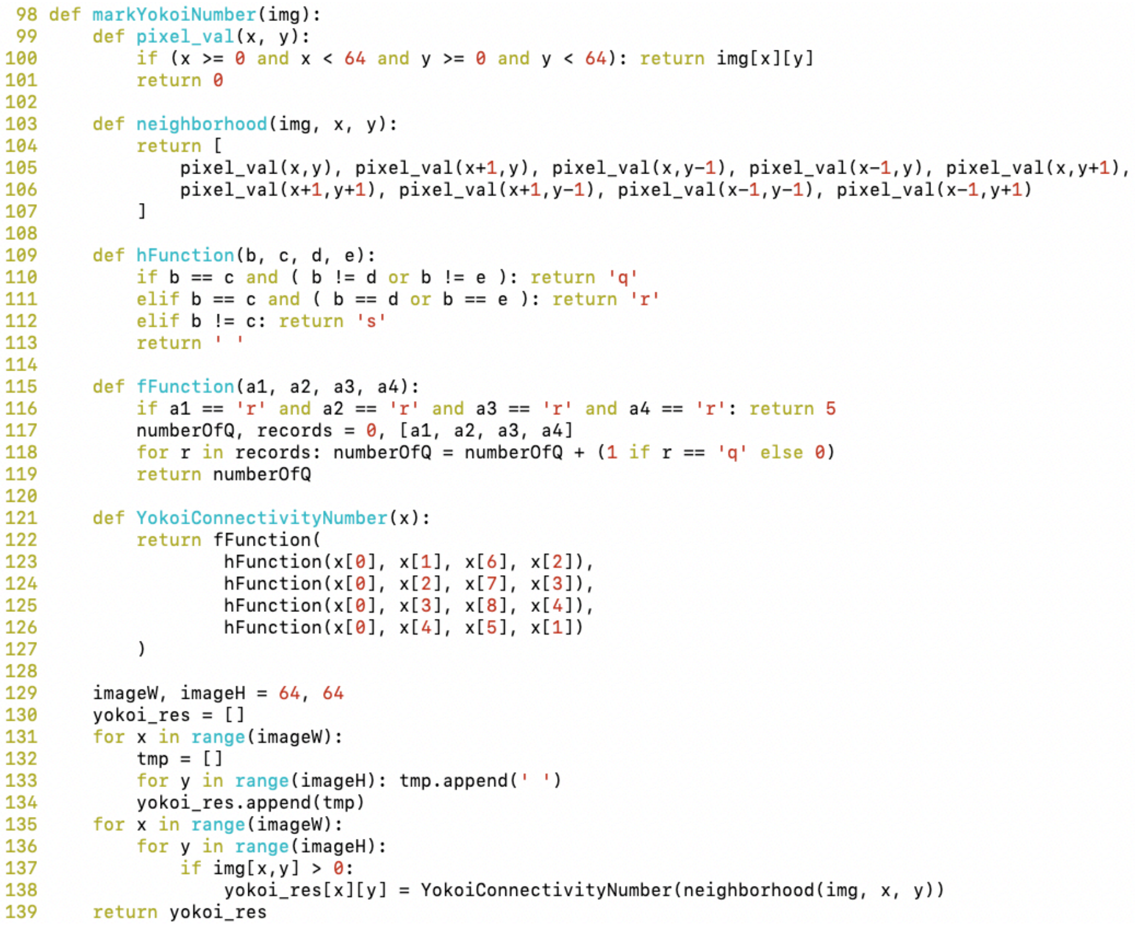
Part1. 先將原始影像進行二值化的處理，並且downsample到64*64的大小



```python
def downsample(img, sampleSize=(64,64)):
    downsample_img = Image.new('L', sampleSize)
    downsample_img_pixel = downsample_img.load()
    for x in range(sampleSize[0]):
        for y in range(sampleSize[1]):
            downsample_img_pixel[x, y] = img.getpixel((x*8, y*8))
    downsample_img.save('./downsampled_lena.bmp')
    return downsample_img

def binarize(img):
    imageW, imageH = img.width, img.height
    new_img = img.copy()
    new_img_pixel = new_img.load()
    for x in range(imageW):
        for y in range(imageH):
            new_img_pixel[x, y] = 255 if img.getpixel((x,y)) > 127 else 0
    new_img.save('./binarize_lena.bmp')
    return new_img
```

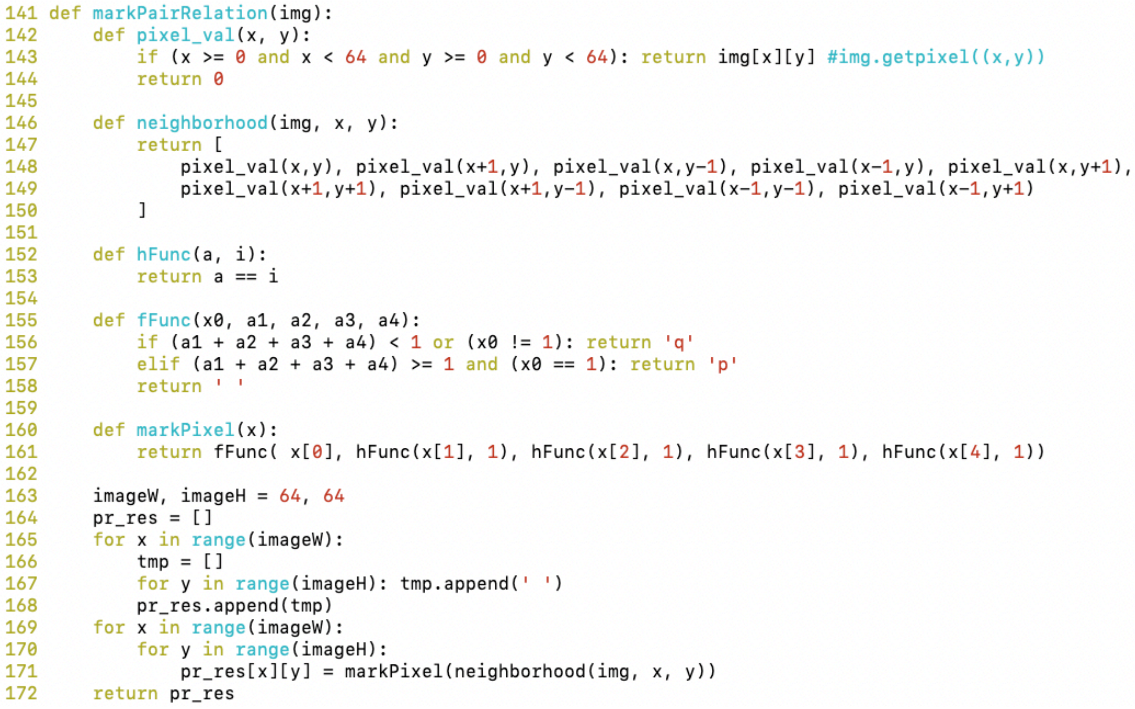
Part2. Thinning Operator的三個部分

[Step1] 實作公式參考課程投影片 Mark Yokoi Number



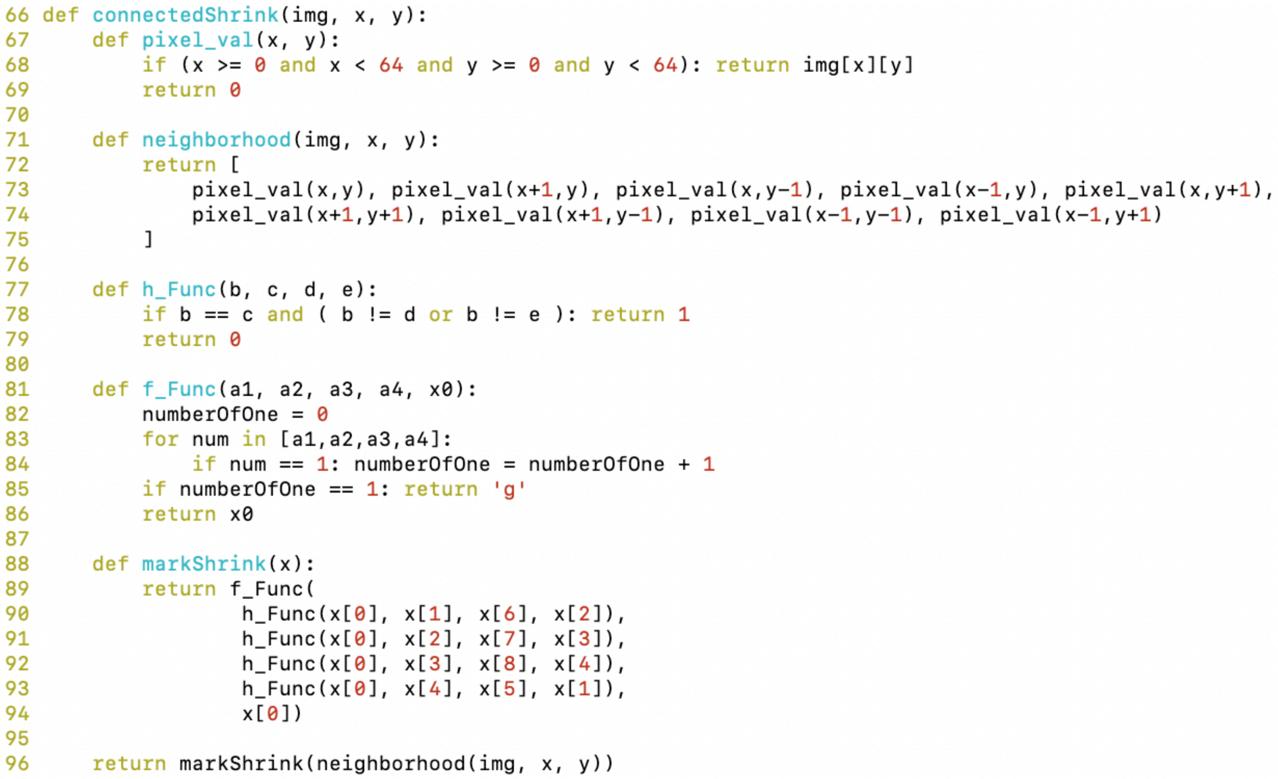
```python
def markYokoiNumber(img):
    def pixel_val(x, y):
        if (x >= 0 and x < 64 and y >= 0 and y < 64): return img[x][y]
        return 0

    def neighborhood(img, x, y):
        return [
            pixel_val(x,y), pixel_val(x+1,y), pixel_val(x,y-1), pixel_val(x-1,y), pixel_val(x,y+1),
            pixel_val(x+1,y+1), pixel_val(x+1,y-1), pixel_val(x-1,y-1), pixel_val(x-1,y+1)
        ]

    def hFunction(b, c, d, e):
        if b == c and ( b != d or b != e ): return 'q'
        elif b == c and ( b == d or b == e ): return 'r'
        elif b != c: return 's'
        return ' '

    def fFunction(a1, a2, a3, a4):
        if a1 == 'r' and a2 == 'r' and a3 == 'r' and a4 == 'r': return 5
        numberOfQ, records = 0, [a1, a2, a3, a4]
        for r in records: numberOfQ = numberOfQ + (1 if r == 'q' else 0)
        return numberOfQ

    def YokoiConnectivityNumber(x):
        return fFunction(
                hFunction(x[0], x[1], x[6], x[2]),
                hFunction(x[0], x[2], x[7], x[3]),
                hFunction(x[0], x[3], x[8], x[4]),
                hFunction(x[0], x[4], x[5], x[1])
        )

    imageW, imageH = 64, 64
    yokoi_res = []
    for x in range(imageW):
        tmp = []
        for y in range(imageH): tmp.append(' ')
        yokoi_res.append(tmp)
    for x in range(imageW):
        for y in range(imageH):
            if img[x,y] > 0:
                yokoi_res[x][y] = YokoiConnectivityNumber(neighborhood(img, x, y))
    return yokoi_res
```

[Step2] 實作公式參考課程投影片 Mark pair relationship operator (把Yokoi結果傳入)



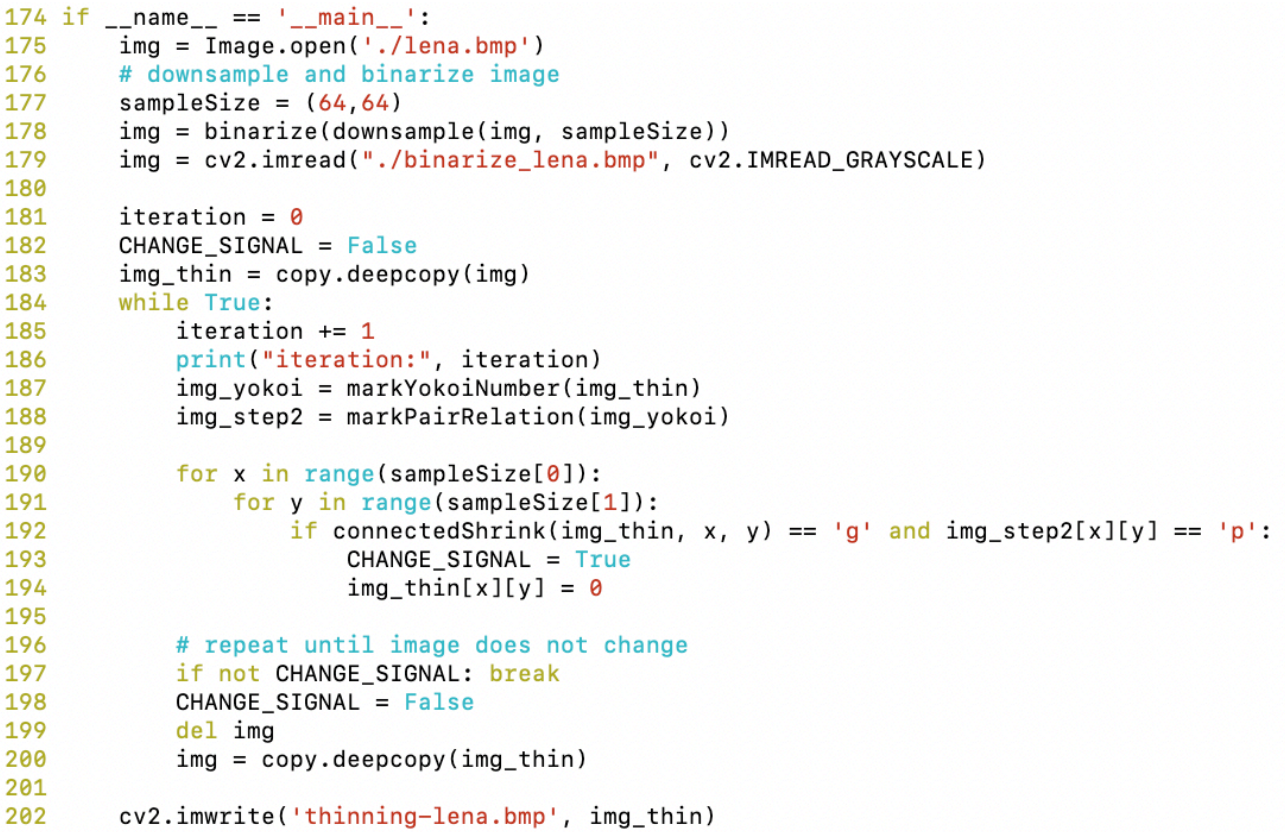
```python
def markPairRelation(img):
    def pixel_val(x, y):
        if (x >= 0 and x < 64 and y >= 0 and y < 64): return img[x][y] #img.getpixel((x,y))
        return 0

    def neighborhood(img, x, y):
        return [
            pixel_val(x,y), pixel_val(x+1,y), pixel_val(x,y-1), pixel_val(x-1,y), pixel_val(x,y+1),
            pixel_val(x+1,y+1), pixel_val(x+1,y-1), pixel_val(x-1,y-1), pixel_val(x-1,y+1)
        ]

    def hFunc(a, i):
        return a == i

    def fFunc(x0, a1, a2, a3, a4):
        if (a1 + a2 + a3 + a4) < 1 or (x0 != 1): return 'q'
        elif (a1 + a2 + a3 + a4) >= 1 and (x0 == 1): return 'p'
        return ' '

    def markPixel(x):
        return fFunc( x[0], hFunc(x[1], 1), hFunc(x[2], 1), hFunc(x[3], 1), hFunc(x[4], 1))

    imageW, imageH = 64, 64
    pr_res = []
    for x in range(imageW):
        tmp = []
        for y in range(imageH): tmp.append(' ')
        pr_res.append(tmp)
    for x in range(imageW):
        for y in range(imageH):
            pr_res[x][y] = markPixel(neighborhood(img, x, y))
    return pr_res
```

[Step3] 這邊要做 Connected Shrinking Operator，根據作業投影片，實作方式為：將原始影像傳入Yokoi Function計算Yokoi Number，在把他經過Mark Pair Relationship 運算的結果，與原圖經過Connected Shrink的結果比較，看此pixel是否為可刪除的。如果此pixel的Pair Relationship運算的結果是’p’，且Connected Shrink運算的結果為 ‘g’，那這樣代表他是可以刪除的。



```python
def connectedShrink(img, x, y):
    def pixel_val(x, y):
        if (x >= 0 and x < 64 and y >= 0 and y < 64): return img[x][y]
        return 0

    def neighborhood(img, x, y):
        return [
            pixel_val(x,y), pixel_val(x+1,y), pixel_val(x,y-1), pixel_val(x-1,y), pixel_val(x,y+1),
            pixel_val(x+1,y+1), pixel_val(x+1,y-1), pixel_val(x-1,y-1), pixel_val(x-1,y+1)
        ]

    def h_Func(b, c, d, e):
        if b == c and ( b != d or b != e ): return 1
        return 0

    def f_Func(a1, a2, a3, a4, x0):
        numberOfOne = 0
        for num in [a1,a2,a3,a4]:
            if num == 1: numberOfOne = numberOfOne + 1
        if numberOfOne == 1: return 'g'
        return x0

    def markShrink(x):
        return f_Func(
                h_Func(x[0], x[1], x[6], x[2]),
                h_Func(x[0], x[2], x[7], x[3]),
                h_Func(x[0], x[3], x[8], x[4]),
                h_Func(x[0], x[4], x[5], x[1]),
                x[0])

    return markShrink(neighborhood(img, x, y))
```

Part3. 最後，不斷執行Thinning Operation直到影像不變化為



```python
if __name__ == '__main__':
    img = Image.open('./lena.bmp')
    # downsample and binarize image
    sampleSize = (64,64)
    img = binarize(downsample(img, sampleSize))
    img = cv2.imread("./binarize_lena.bmp", cv2.IMREAD_GRAYSCALE)

    iteration = 0
    CHANGE_SIGNAL = False
    img_thin = copy.deepcopy(img)
    while True:
        iteration += 1
        print("iteration:", iteration)
        img_yokoi = markYokoiNumber(img_thin)
        img_step2 = markPairRelation(img_yokoi)

        for x in range(sampleSize[0]):
            for y in range(sampleSize[1]):
                if connectedShrink(img_thin, x, y) == 'g' and img_step2[x][y] == 'p':
                    CHANGE_SIGNAL = True
                    img_thin[x][y] = 0

        # repeat until image does not change
        if not CHANGE_SIGNAL: break
        CHANGE_SIGNAL = False
        del img
        img = copy.deepcopy(img_thin)

    cv2.imwrite('thinning-lena.bmp', img_thin)
```