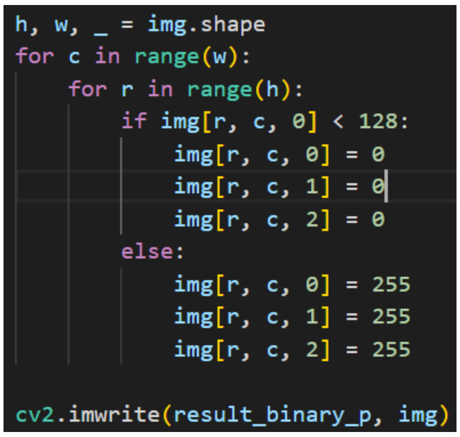
Homework 2 網媒所碩一 何明倩 R11944009

(a) a binary image (threshold at 128)

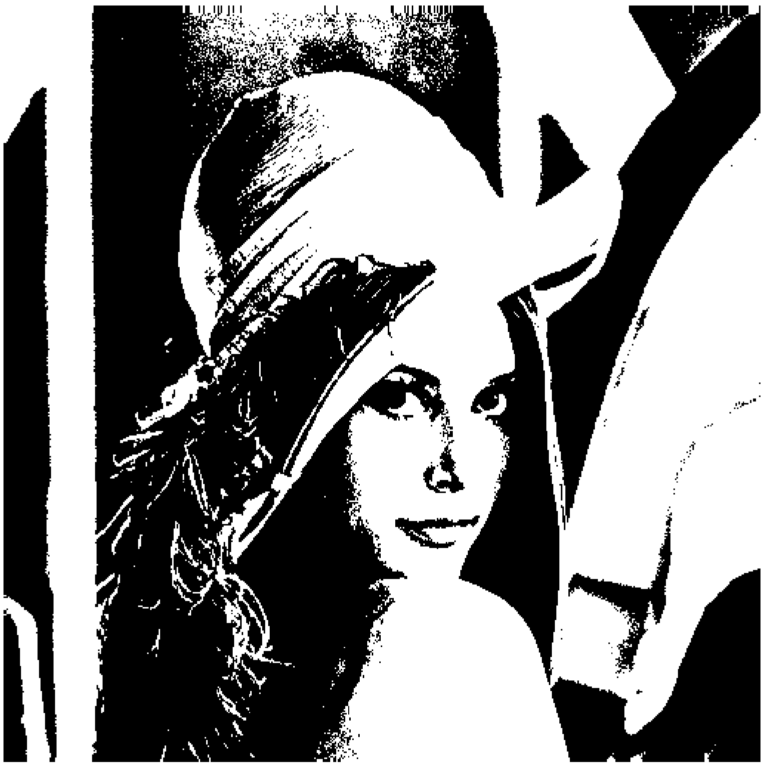
- description:
  若 pixel 的數值小於 128，則數值改為 255；其餘則為 0
- algorithm:
  拜訪每個 pixel，若 pixel 的數值小於 128，則數值改為 255；其餘則為 0
- principal code fragment:



```python
h, w, _ = img.shape
for c in range(w):
    for r in range(h):
        if img[r, c, 0] < 128:
            img[r, c, 0] = 0
            img[r, c, 1] = 0
            img[r, c, 2] = 0
        else:
            img[r, c, 0] = 255
            img[r, c, 1] = 255
            img[r, c, 2] = 255

cv2.imwrite(result_binary_p, img)
```

- result:

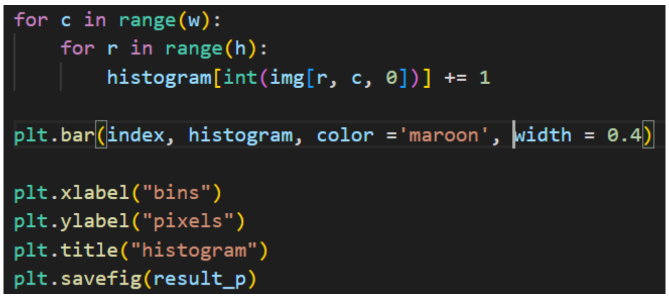


(b) a histogram

- description:
  劃出 0-255 的分布圖

- algorithm:

  拜訪每個 pixel，計算 0-255 各自的 pixel 數量並劃出 bar graph
- principal code fragment:

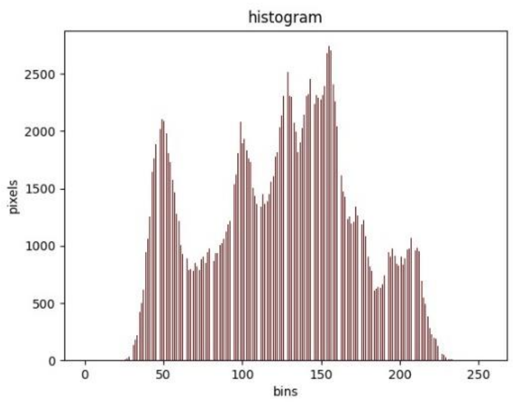


```
for c in range(w):
    for r in range(h):
        histogram[int(img[r, c, 0])] += 1

plt.bar(index, histogram, color ='maroon', width = 0.4)

plt.xlabel("bins")
plt.ylabel("pixels")
plt.title("histogram")
plt.savefig(result_p)
```

- result:

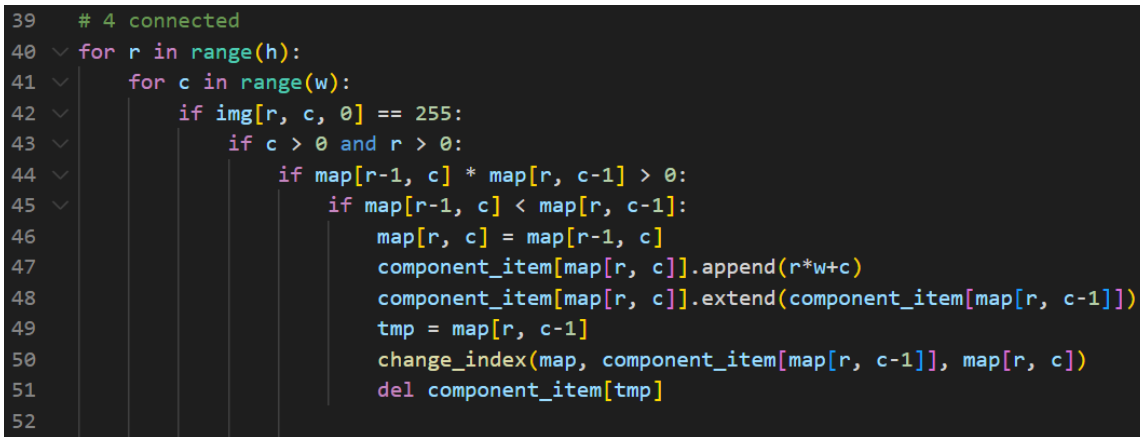


(c) connected components(regions with + at centroid, bounding box)

- description:

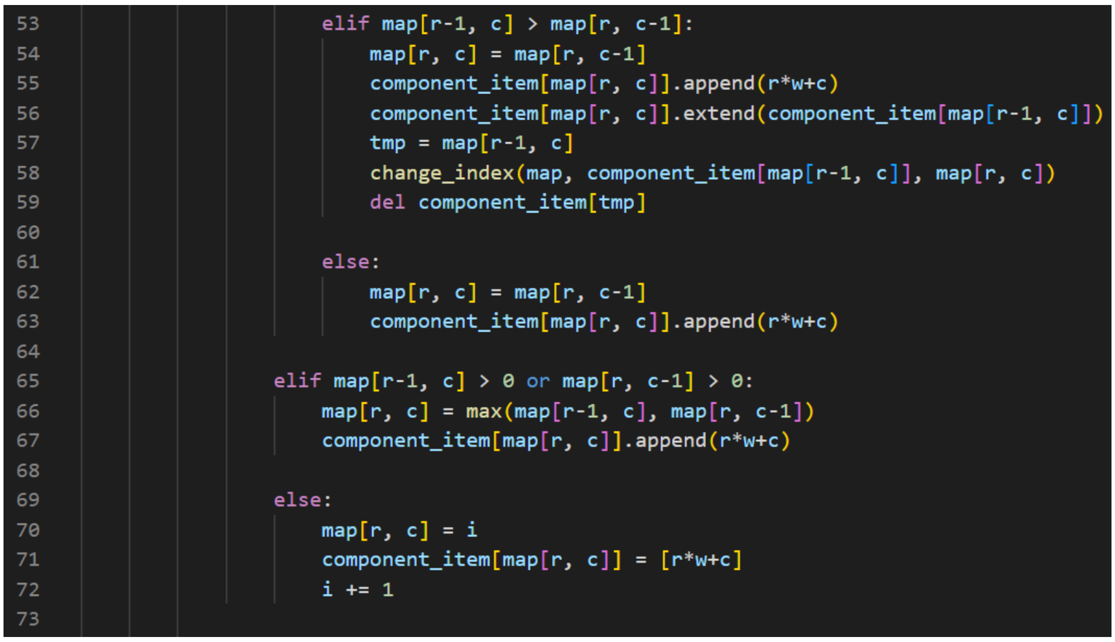
  利用 4-connected 來找出對應 component，並劃出面積>=500 的 bounding box 和 centroid
- algorithm:
  1. 先將影像轉成 binary image
  2. 拜訪每個 pixel，確認該 pixel 上面和左邊 pixel 所屬的 label，若上面和左邊的 label 大於 0，將該 pixel 標示為最小的 label，較大 label 的 component 將其 label 改成較小 label（因為這兩個 component 被連接在一起）。若上面左邊其中一個 label 是 0，則該 pixel 標記成非 0 的 label。若上面左邊皆為 0，則設該 pixel 為新的 label
  3. 拜訪所有紀錄的 components，將該 component 的每個 pixel 的 row 集合取最大最小值為 y_max, y_min，將該 component 的每個 pixel 的 column 集合取最大最小值為 x_max, x_min，即可算出 bounding box 的對稱角(x_max, y_max)，(x_min, y_min)。將該 component 的每個 pixel 的 row 相加取平均為 y，每個 pixel 的 column 相加取平均為 x，(x, y)即為重心

• principal code fragment:



```python
# 4 connected
for r in range(h):
    for c in range(w):
        if img[r, c, 0] == 255:
            if c > 0 and r > 0:
                if map[r-1, c] * map[r, c-1] > 0:
                    if map[r-1, c] < map[r, c-1]:
                        map[r, c] = map[r-1, c]
                        component_item[map[r, c]].append(r*w+c)
                        component_item[map[r, c]].extend(component_item[map[r, c-1]])
                        tmp = map[r, c-1]
                        change_index(map, component_item[map[r, c-1]], map[r, c])
                        del component_item[tmp]

```



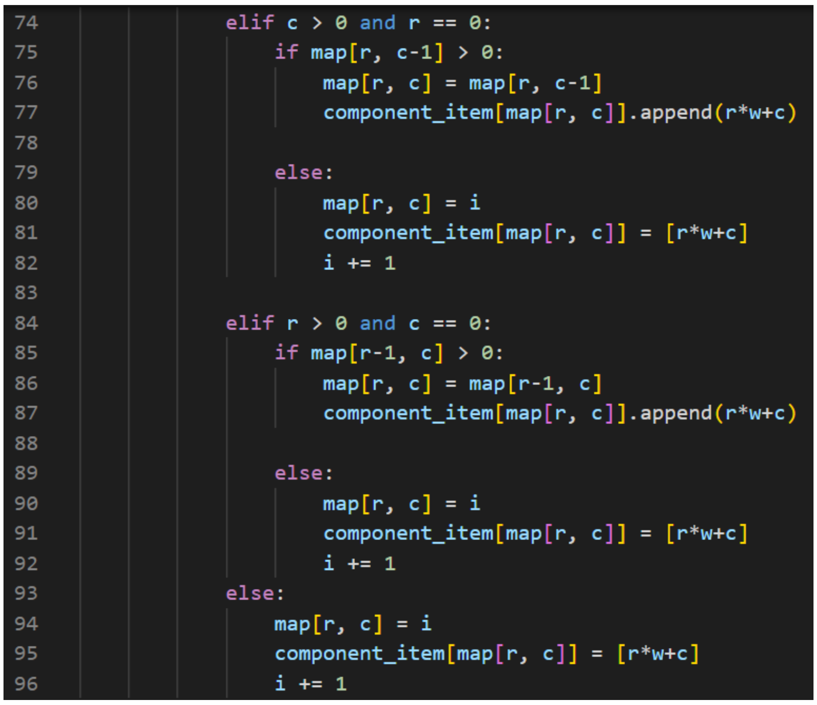
```python
                    elif map[r-1, c] > map[r, c-1]:
                        map[r, c] = map[r, c-1]
                        component_item[map[r, c]].append(r*w+c)
                        component_item[map[r, c]].extend(component_item[map[r-1, c]])
                        tmp = map[r-1, c]
                        change_index(map, component_item[map[r-1, c]], map[r, c])
                        del component_item[tmp]

                    else:
                        map[r, c] = map[r, c-1]
                        component_item[map[r, c]].append(r*w+c)

                elif map[r-1, c] > 0 or map[r, c-1] > 0:
                    map[r, c] = max(map[r-1, c], map[r, c-1])
                    component_item[map[r, c]].append(r*w+c)

                else:
                    map[r, c] = i
                    component_item[map[r, c]] = [r*w+c]
                    i += 1

```

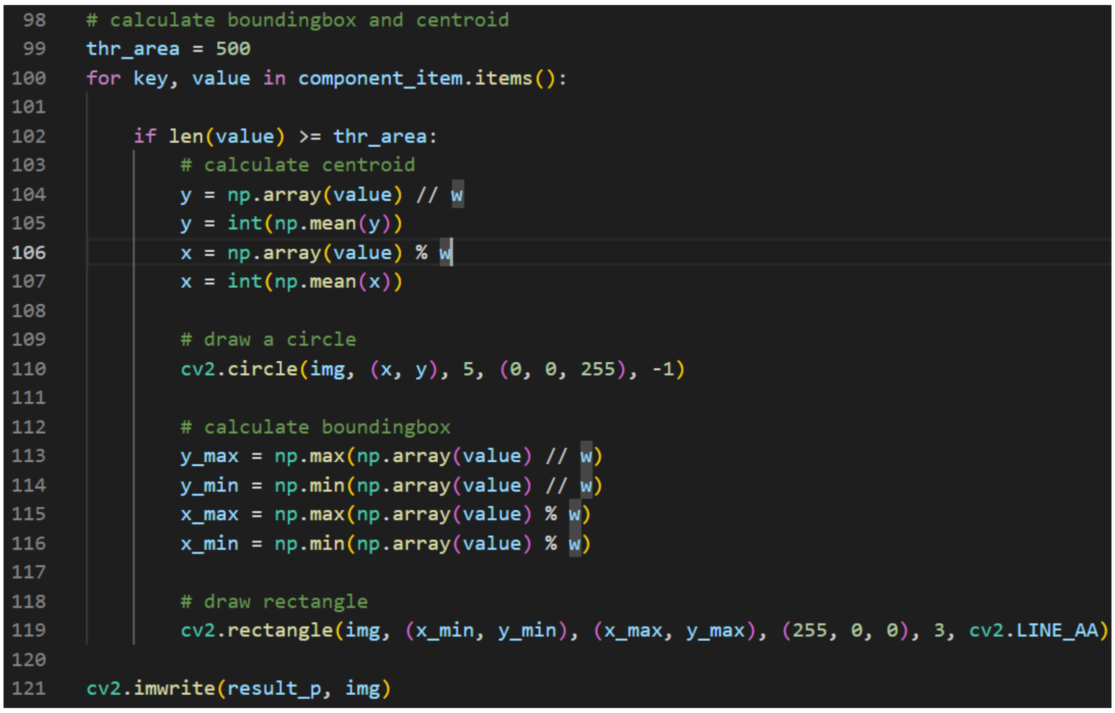


```python
            elif c > 0 and r == 0:
                if map[r, c-1] > 0:
                    map[r, c] = map[r, c-1]
                    component_item[map[r, c]].append(r*w+c)

                else:
                    map[r, c] = i
                    component_item[map[r, c]] = [r*w+c]
                    i += 1

            elif r > 0 and c == 0:
                if map[r-1, c] > 0:
                    map[r, c] = map[r-1, c]
                    component_item[map[r, c]].append(r*w+c)

                else:
                    map[r, c] = i
                    component_item[map[r, c]] = [r*w+c]
                    i += 1
            else:
                map[r, c] = i
                component_item[map[r, c]] = [r*w+c]
                i += 1
```



```python
# calculate boundingbox and centroid
thr_area = 500
for key, value in component_item.items():

    if len(value) >= thr_area:
        # calculate centroid
        y = np.array(value) // w
        y = int(np.mean(y))
        x = np.array(value) % w
        x = int(np.mean(x))

        # draw a circle
        cv2.circle(img, (x, y), 5, (0, 0, 255), -1)

        # calculate boundingbox
        y_max = np.max(np.array(value) // w)
        y_min = np.min(np.array(value) // w)
        x_max = np.max(np.array(value) % w)
        x_min = np.min(np.array(value) % w)

        # draw rectangle
        cv2.rectangle(img, (x_min, y_min), (x_max, y_max), (255, 0, 0), 3, cv2.LINE_AA)

cv2.imwrite(result_p, img)
```

- result: