Homework 2

Basic Image Manipulation

Programming language: Python 3.7.3

Library used for this homework:

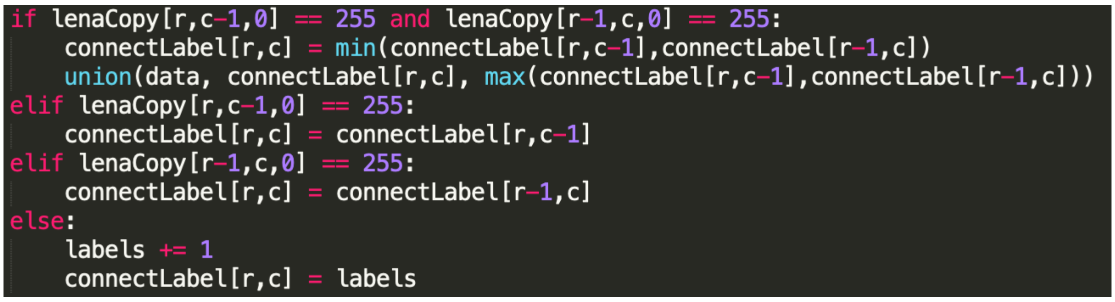
- ✦ Numpy
- ✦ OpenCv: to read and write the image file

Image info: lena.bmp [512(width),512(height),3(RGB)]

Code explanation:

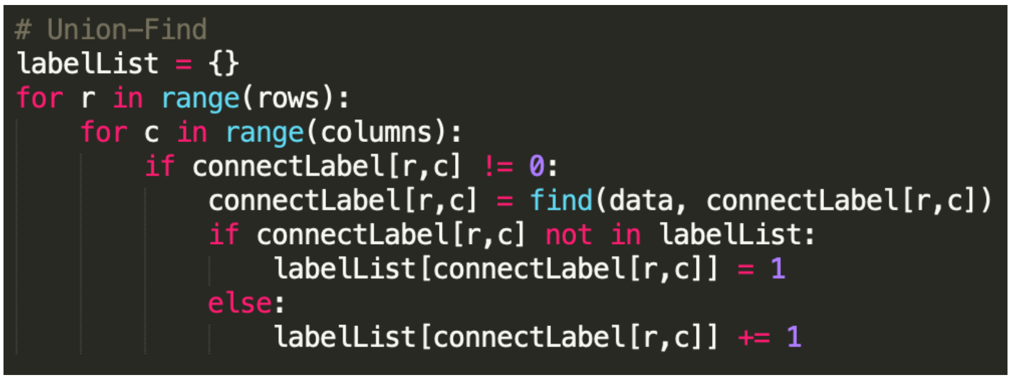
Use “two-pass algorithm”.

1. Check if the left pixel and upper pixel are both white. If it is true, label the current pixel as the smaller label among those two pixels. Record that the two labels are connected.
2. Check if the left pixel is both white. If it is true, label the current pixel as the label of the left pixel.
3. Check if the upper pixel is both white. If it is true, label the current pixel as the label of the upper pixel.
4. If none of the conditions match, create a new label for the current pixel.



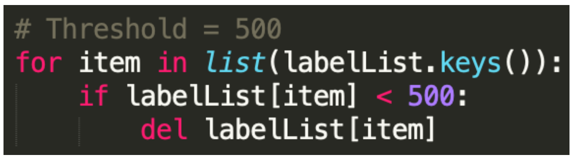
```
if lenaCopy[r,c-1,0] == 255 and lenaCopy[r-1,c,0] == 255:
    connectLabel[r,c] = min(connectLabel[r,c-1],connectLabel[r-1,c])
    union(data, connectLabel[r,c], max(connectLabel[r,c-1],connectLabel[r-1,c]))
elif lenaCopy[r,c-1,0] == 255:
    connectLabel[r,c] = connectLabel[r,c-1]
elif lenaCopy[r-1,c,0] == 255:
    connectLabel[r,c] = connectLabel[r-1,c]
else:
    labels += 1
    connectLabel[r,c] = labels
```

Use “union-find” to find the smallest label number among the connected labels and change the current label to it.



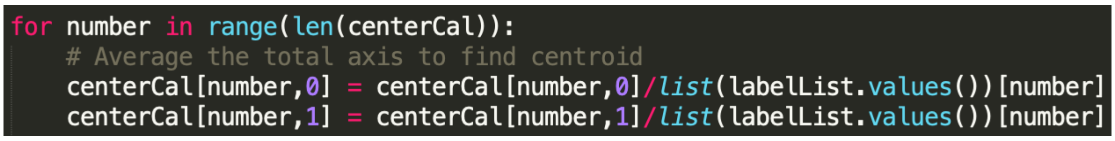
```
# Union-Find
labelList = {}
for r in range(rows):
    for c in range(columns):
        if connectLabel[r,c] != 0:
            connectLabel[r,c] = find(data, connectLabel[r,c])
            if connectLabel[r,c] not in labelList:
                labelList[connectLabel[r,c]] = 1
            else:
                labelList[connectLabel[r,c]] += 1
```

Set a threshold which only accept connected components that are larger then 500 pixels.



```python
# Threshold = 500
for item in list(labelList.keys()):
    if labelList[item] < 500:
        del labelList[item]
```

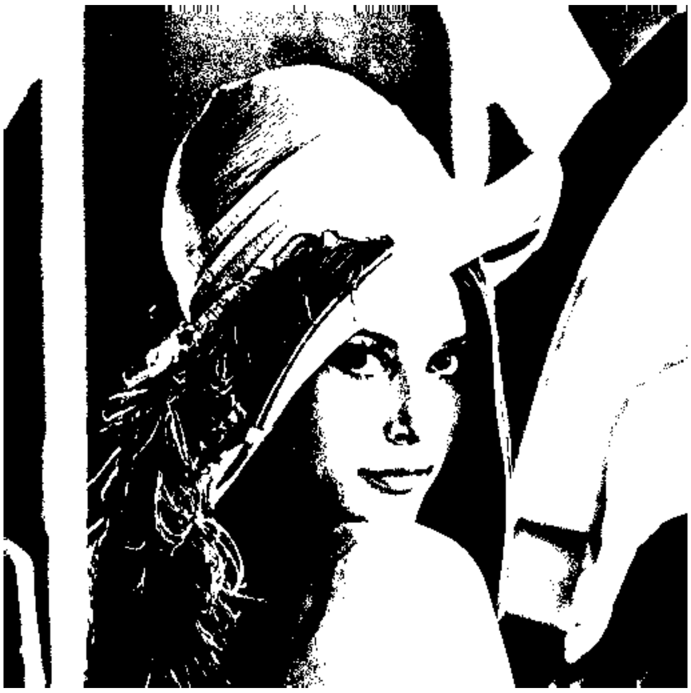
Finally, add up all the x and y positions with the same label, and average the number to get centroid.
Draw a box from the top left to the bottom right of each connected region, and draw a dot on each centroid.



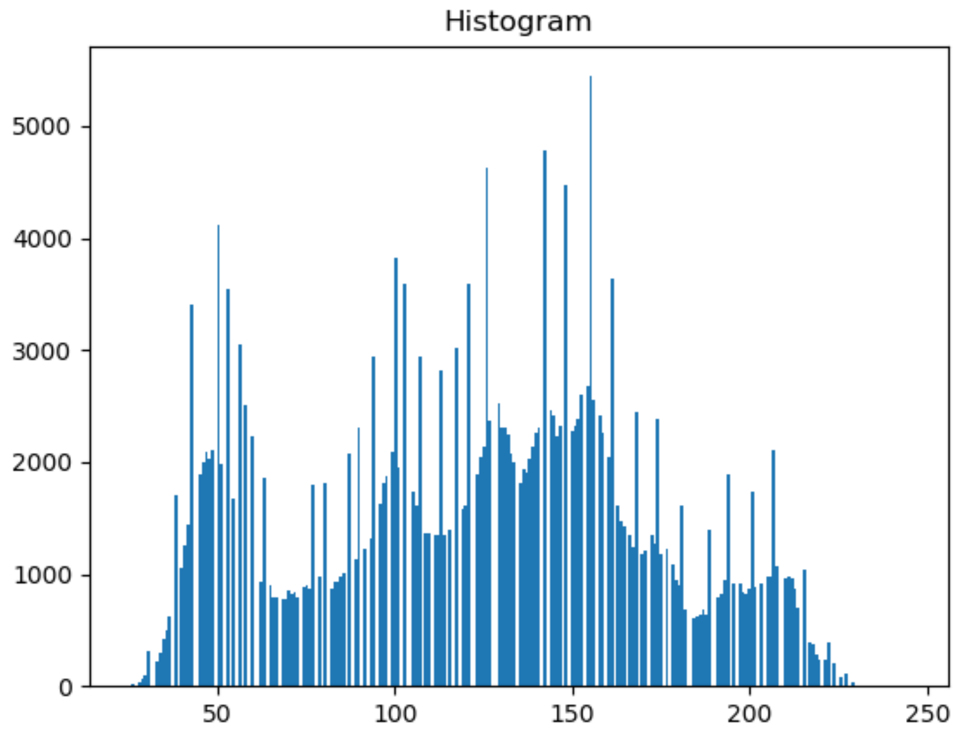
```python
for number in range(len(centerCal)):
    # Average the total axis to find centroid
    centerCal[number,0] = centerCal[number,0]/list(labelList.values())[number]
    centerCal[number,1] = centerCal[number,1]/list(labelList.values())[number]
```

Result:

(a)



(b)



(c)

Use 4-connected neighborhood

Red dot: centroid points

Light blue line: bounding boxes