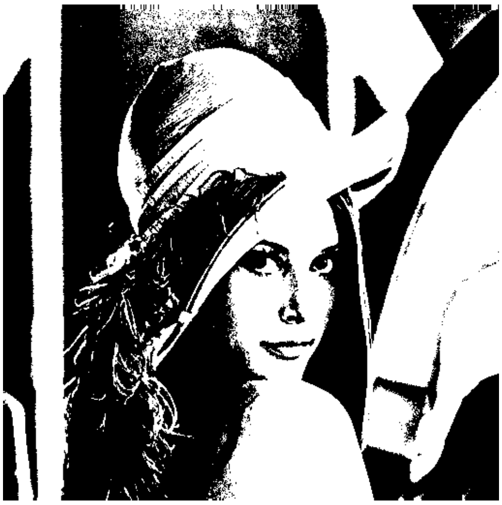
Computer vision homework 2

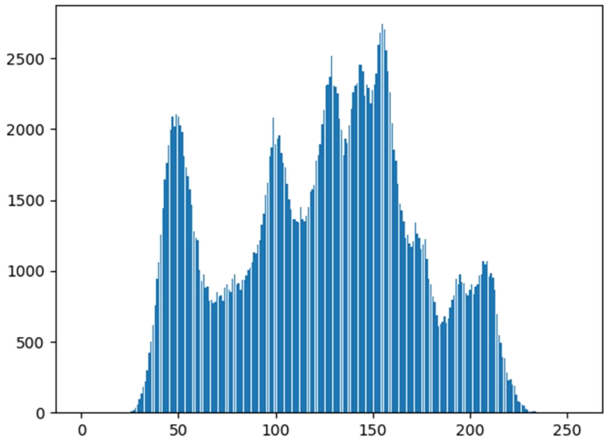
B04902028 資工三 洪浩翔

1. a binary image:



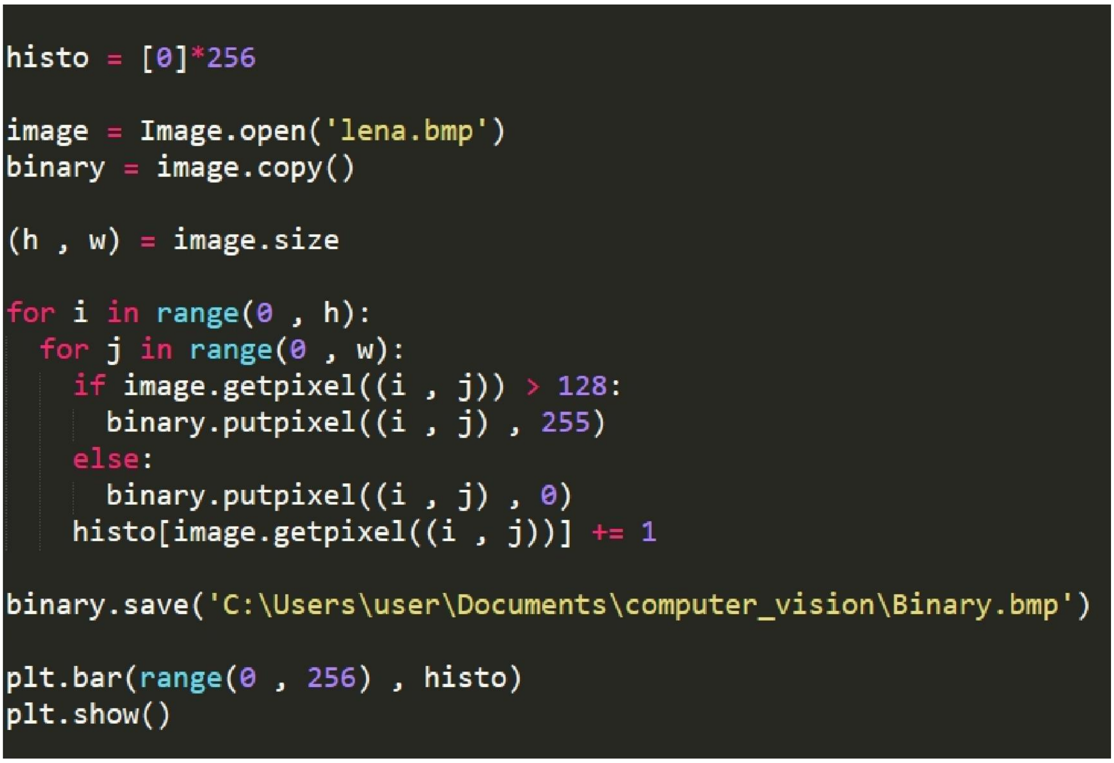
Binarized result

2. a histogram:



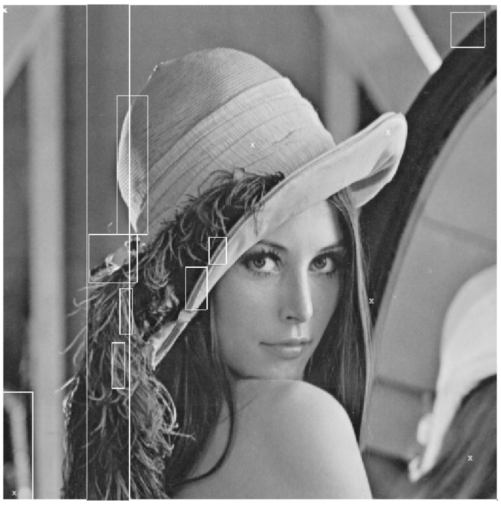
Histogram

Main code of binary image and histogram:



```
histo = [0]*256

image = Image.open('lena.bmp')
binary = image.copy()

(h , w) = image.size

for i in range(0 , h):
  for j in range(0 , w):
    if image.getpixel((i , j)) > 128:
      binary.putpixel((i , j) , 255)
    else:
      binary.putpixel((i , j) , 0)
    histo[image.getpixel((i , j))] += 1

binary.save('C:\Users\user\Documents\computer_vision\Binary.bmp')

plt.bar(range(0 , 256) , histo)
plt.show()
```

3. connected components with bounding box and centroid:



Result of connected components

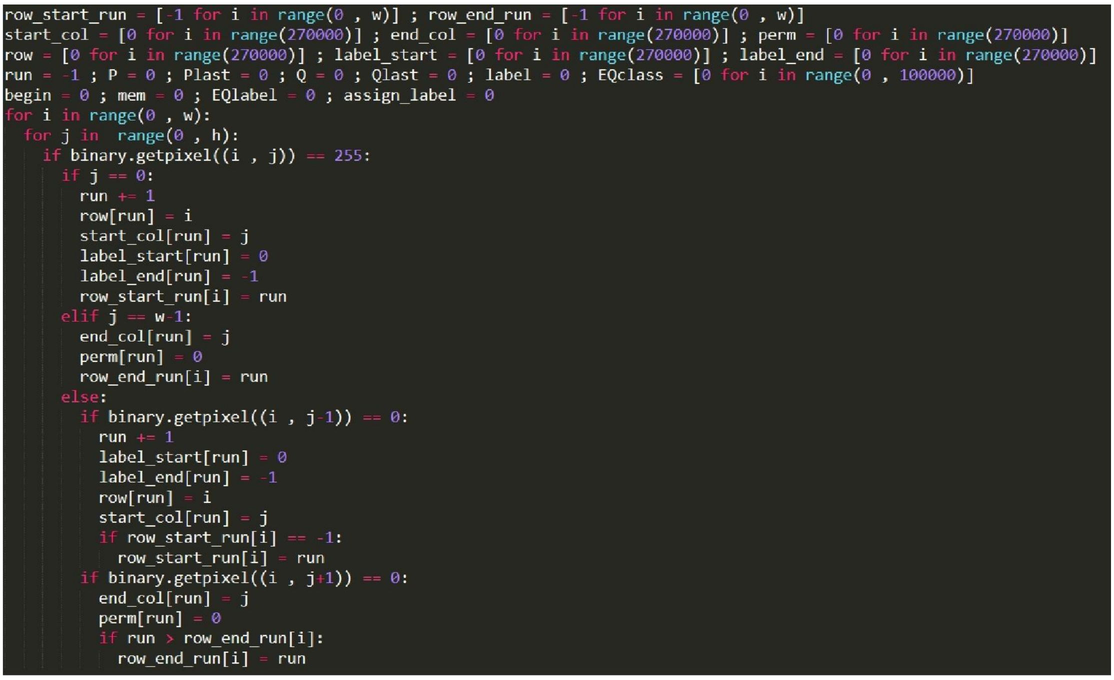
Algorithm:

Run-Length Implementation of the Local Table Method

With using 4-connected

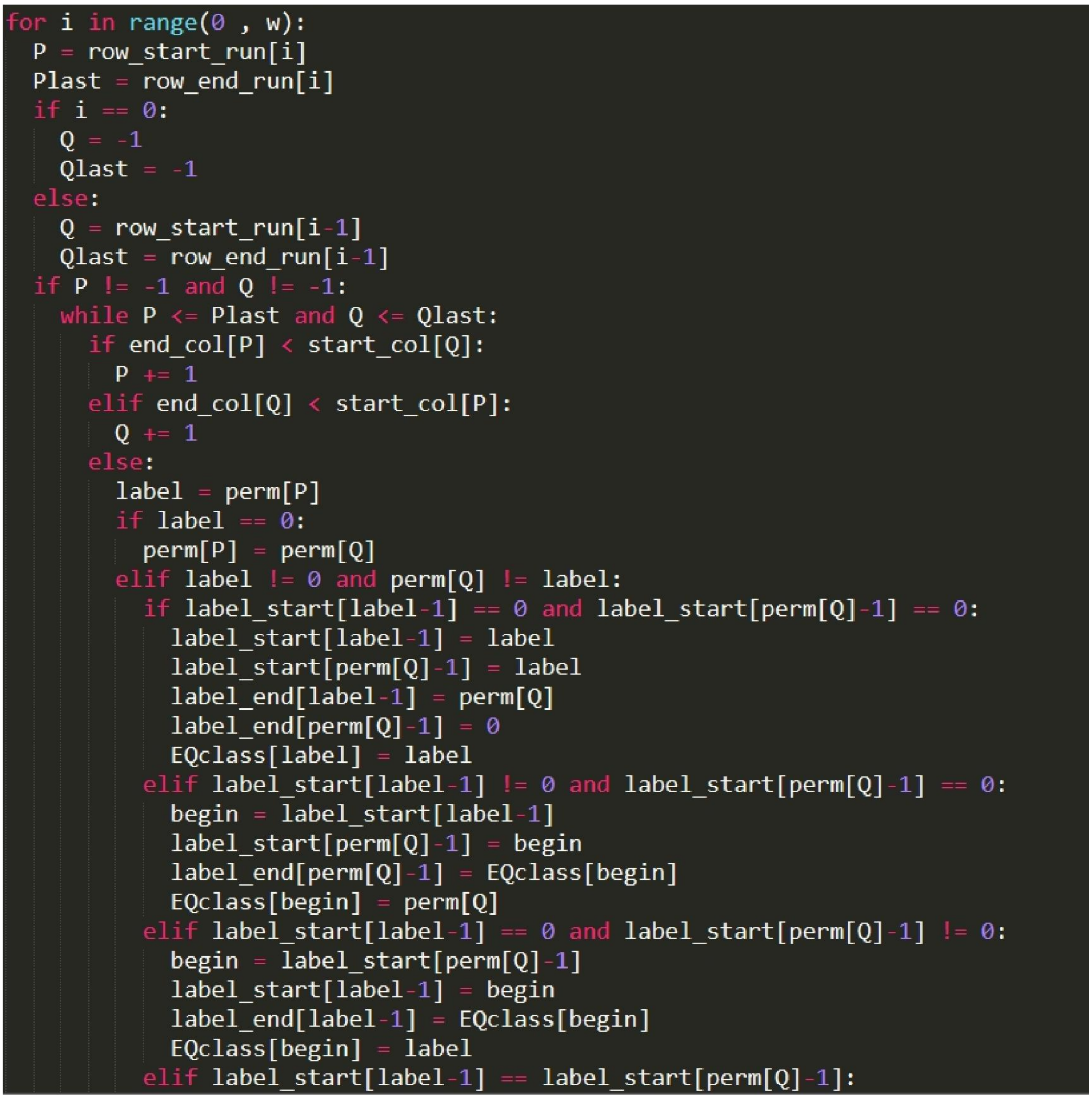
Main code of implementation:

1. Create and set all the table we need

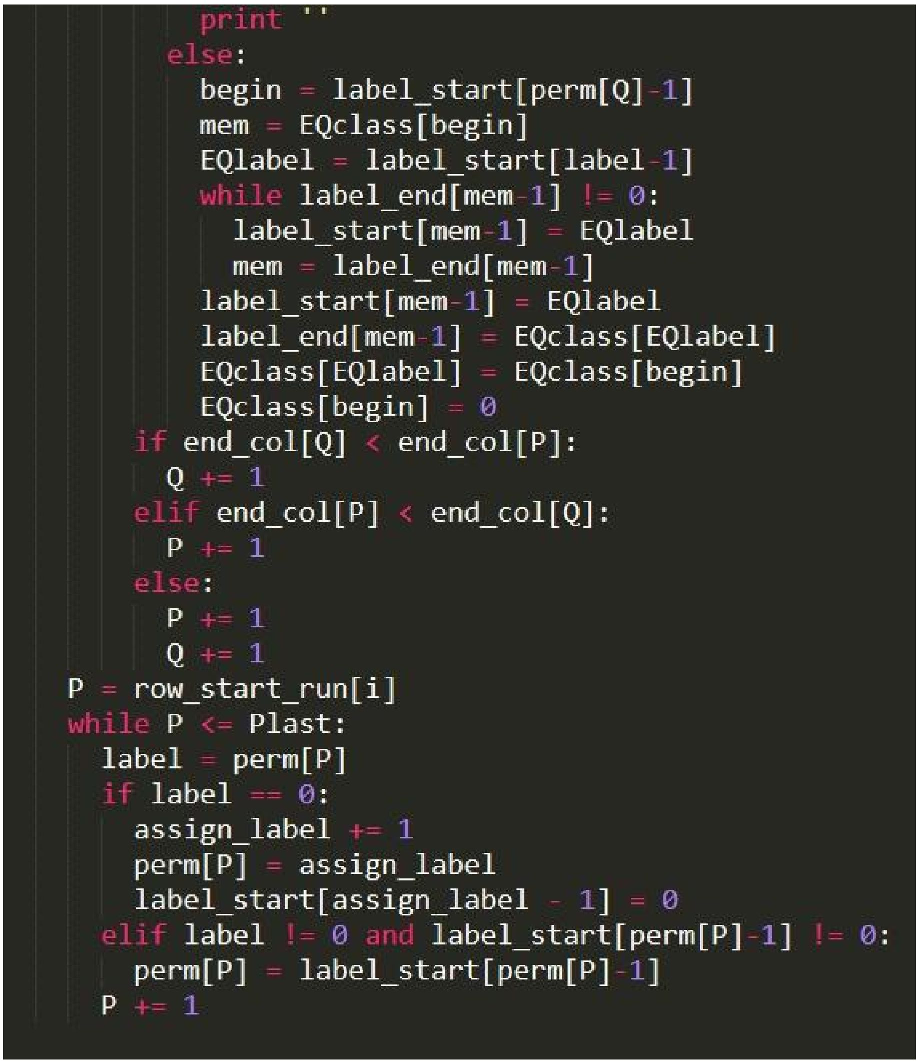


```
row_start_run = [-1 for i in range(0 , w)] ; row_end_run = [-1 for i in range(0 , w)]
start_col = [0 for i in range(270000)] ; end_col = [0 for i in range(270000)] ; perm = [0 for i in range(270000)]
row = [0 for i in range(270000)] ; label_start = [0 for i in range(270000)] ; label_end = [0 for i in range(270000)]
run = -1 ; P = 0 ; Plast = 0 ; Q = 0 ; Qlast = 0 ; label = 0 ; EQclass = [0 for i in range(0 , 100000)]
begin = 0 ; mem = 0 ; EQlabel = 0 ; assign_label = 0
for i in range(0 , w):
  for j in  range(0 , h):
    if binary.getpixel((i , j)) == 255:
      if j == 0:
        run += 1
        row[run] = i
        start_col[run] = j
        label_start[run] = 0
        label_end[run] = -1
        row_start_run[i] = run
      elif j == w-1:
        end_col[run] = j
        perm[run] = 0
        row_end_run[i] = run
      else:
        if binary.getpixel((i , j-1)) == 0:
          run += 1
          label_start[run] = 0
          label_end[run] = -1
          row[run] = i
          start_col[run] = j
          if row_start_run[i] == -1:
            row_start_run[i] = run
        if binary.getpixel((i , j+1)) == 0:
          end_col[run] = j
          perm[run] = 0
          if run > row_end_run[i]:
            row_end_run[i] = run
```

2. Top-down process:

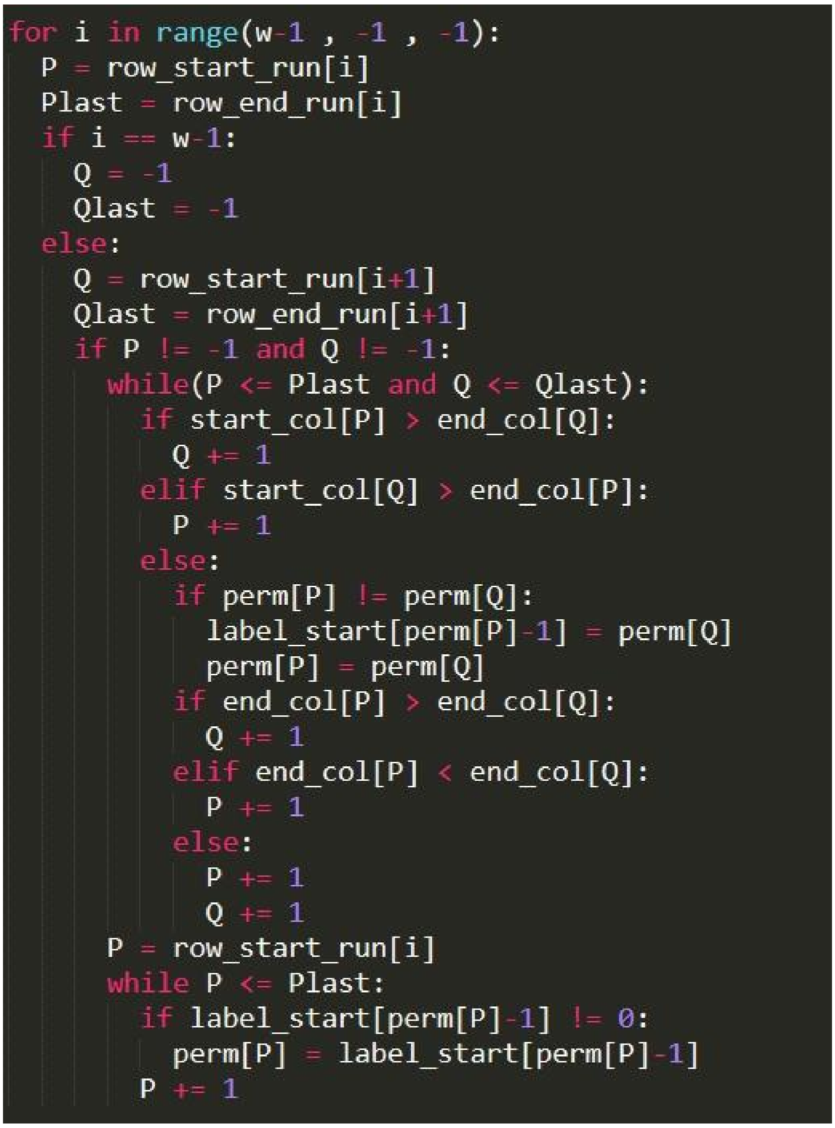


```
for i in range(0 , w):
  P = row_start_run[i]
  Plast = row_end_run[i]
  if i == 0:
    Q = -1
    Qlast = -1
  else:
    Q = row_start_run[i-1]
    Qlast = row_end_run[i-1]
  if P != -1 and Q != -1:
    while P <= Plast and Q <= Qlast:
      if end_col[P] < start_col[Q]:
        P += 1
      elif end_col[Q] < start_col[P]:
        Q += 1
      else:
        label = perm[P]
        if label == 0:
          perm[P] = perm[Q]
        elif label != 0 and perm[Q] != label:
          if label_start[label-1] == 0 and label_start[perm[Q]-1] == 0:
            label_start[label-1] = label
            label_start[perm[Q]-1] = label
            label_end[label-1] = perm[Q]
            label_end[perm[Q]-1] = 0
            EQclass[label] = label
          elif label_start[label-1] != 0 and label_start[perm[Q]-1] == 0:
            begin = label_start[label-1]
            label_start[perm[Q]-1] = begin
            label_end[perm[Q]-1] = EQclass[begin]
            EQclass[begin] = perm[Q]
          elif label_start[label-1] == 0 and label_start[perm[Q]-1] != 0:
            begin = label_start[perm[Q]-1]
            label_start[label-1] = begin
            label_end[label-1] = EQclass[begin]
            EQclass[begin] = label
          elif label_start[label-1] == label_start[perm[Q]-1]:
```



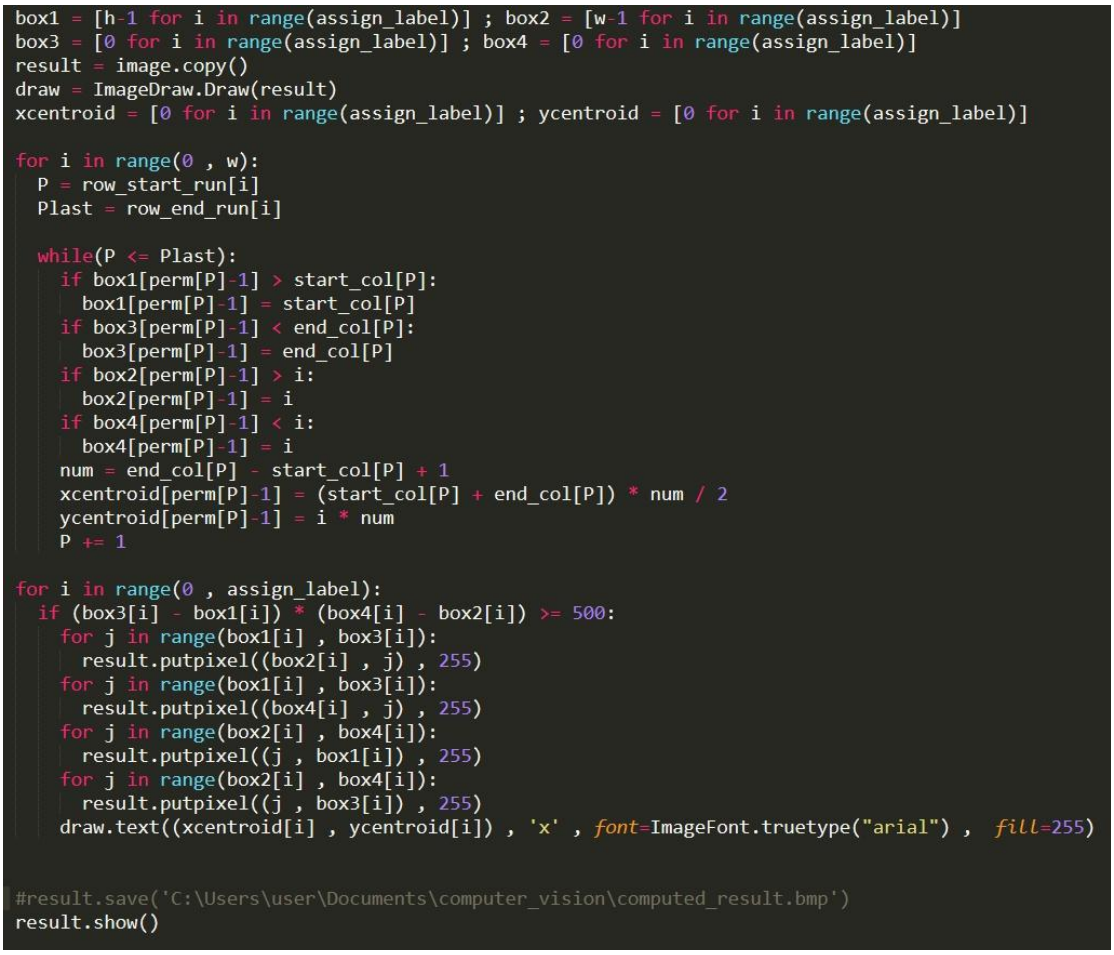
```
          print ''
        else:
          begin = label_start[perm[Q]-1]
          mem = EQclass[begin]
          EQlabel = label_start[label-1]
          while label_end[mem-1] != 0:
            label_start[mem-1] = EQlabel
            mem = label_end[mem-1]
          label_start[mem-1] = EQlabel
          label_end[mem-1] = EQclass[EQlabel]
          EQclass[EQlabel] = EQclass[begin]
          EQclass[begin] = 0
      if end_col[Q] < end_col[P]:
        Q += 1
      elif end_col[P] < end_col[Q]:
        P += 1
      else:
        P += 1
        Q += 1
  P = row_start_run[i]
  while P <= Plast:
    label = perm[P]
    if label == 0:
      assign_label += 1
      perm[P] = assign_label
      label_start[assign_label - 1] = 0
    elif label != 0 and label_start[perm[P]-1] != 0:
      perm[P] = label_start[perm[P]-1]
    P += 1
```

3. Bottom-up process:



```
for i in range(w-1 , -1 , -1):
  P = row_start_run[i]
  Plast = row_end_run[i]
  if i == w-1:
    Q = -1
    Qlast = -1
  else:
    Q = row_start_run[i+1]
    Qlast = row_end_run[i+1]
    if P != -1 and Q != -1:
      while(P <= Plast and Q <= Qlast):
        if start_col[P] > end_col[Q]:
          Q += 1
        elif start_col[Q] > end_col[P]:
          P += 1
        else:
          if perm[P] != perm[Q]:
            label_start[perm[P]-1] = perm[Q]
            perm[P] = perm[Q]
          if end_col[P] > end_col[Q]:
            Q += 1
          elif end_col[P] < end_col[Q]:
            P += 1
          else:
            P += 1
            Q += 1
      P = row_start_run[i]
      while P <= Plast:
        if label_start[perm[P]-1] != 0:
          perm[P] = label_start[perm[P]-1]
        P += 1
```

4. Bounding box and centroid process:



```
box1 = [h-1 for i in range(assign_label)] ; box2 = [w-1 for i in range(assign_label)]
box3 = [0 for i in range(assign_label)] ; box4 = [0 for i in range(assign_label)]
result = image.copy()
draw = ImageDraw.Draw(result)
xcentroid = [0 for i in range(assign_label)] ; ycentroid = [0 for i in range(assign_label)]

for i in range(0 , w):
  P = row_start_run[i]
  Plast = row_end_run[i]

  while(P <= Plast):
    if box1[perm[P]-1] > start_col[P]:
      box1[perm[P]-1] = start_col[P]
    if box3[perm[P]-1] < end_col[P]:
      box3[perm[P]-1] = end_col[P]
    if box2[perm[P]-1] > i:
      box2[perm[P]-1] = i
    if box4[perm[P]-1] < i:
      box4[perm[P]-1] = i
    num = end_col[P] - start_col[P] + 1
    xcentroid[perm[P]-1] = (start_col[P] + end_col[P]) * num / 2
    ycentroid[perm[P]-1] = i * num
    P += 1

for i in range(0 , assign_label):
  if (box3[i] - box1[i]) * (box4[i] - box2[i]) >= 500:
    for j in range(box1[i] , box3[i]):
      result.putpixel((box2[i] , j) , 255)
    for j in range(box1[i] , box3[i]):
      result.putpixel((box4[i] , j) , 255)
    for j in range(box2[i] , box4[i]):
      result.putpixel((j , box1[i]) , 255)
    for j in range(box2[i] , box4[i]):
      result.putpixel((j , box3[i]) , 255)
    draw.text((xcentroid[i] , ycentroid[i]) , 'x' , font=ImageFont.truetype("arial") ,  fill=255)


#result.save('C:\Users\user\Documents\computer_vision\computed_result.bmp')
result.show()
```