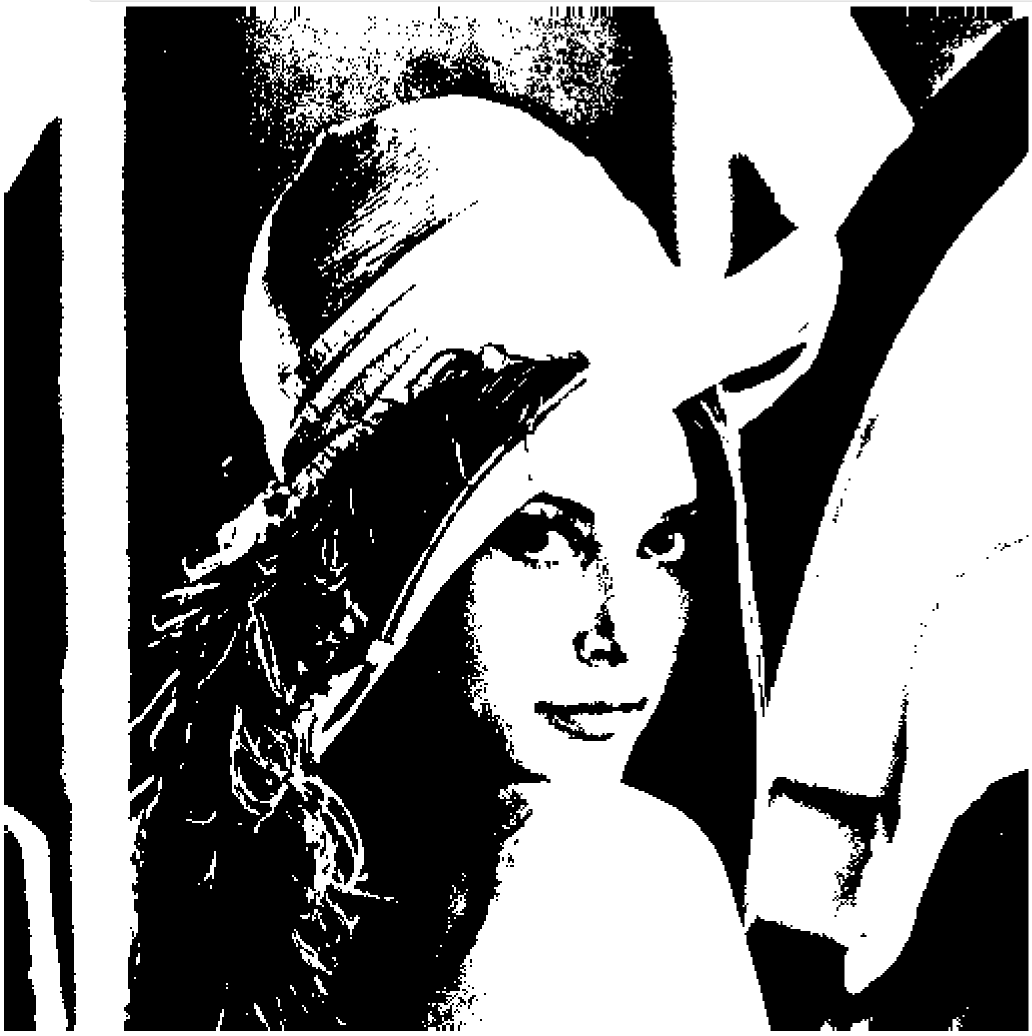
- (a) a binary image (threshold at 128)

```
In [2]: image = cv2.imread('lena.bmp', cv2.IMREAD_GRAYSCALE)
        threshold = 128
        binary_image = np.zeros_like(image, dtype=np.uint8)
        height, width = image.shape

        # Thresholding
        for y in range(height):
            for x in range(width):
                if image[y, x] >= threshold:
                    binary_image[y, x] = 255

        plt.axis('off')
        plt.imshow(binary_image, cmap='gray')
        cv2.imwrite('1a.bmp', binary_image)
        binary = binary_image #for (c)
```



使用 for 迴圈遍歷每一 pixel，當亮度大於 128 時，設為白色(255)，反之設為黑色，即獲得 binary image。

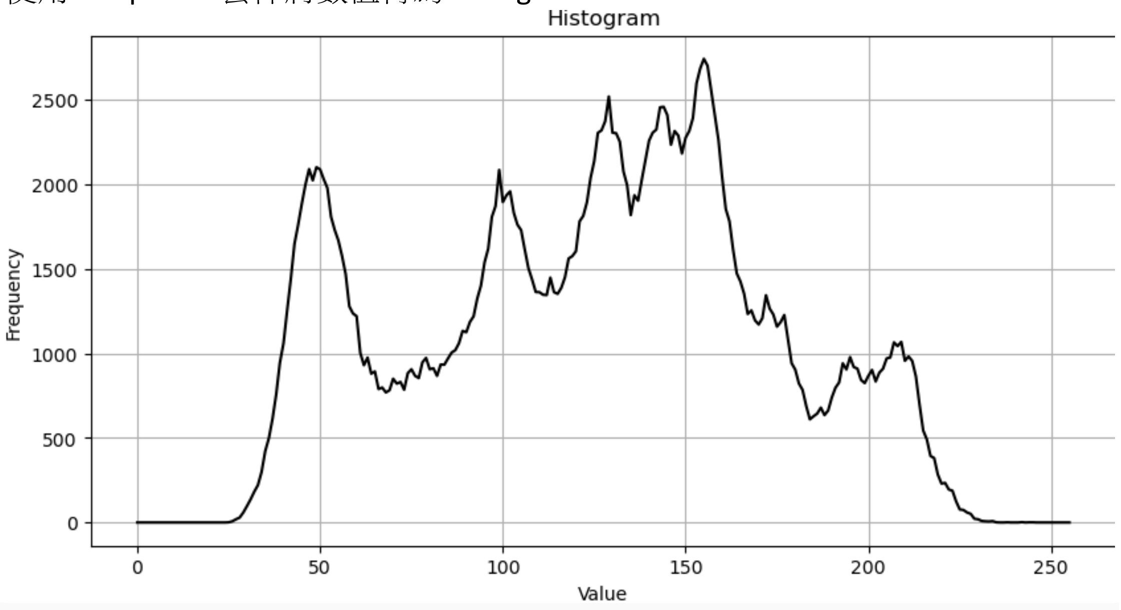
- (b) a histogram

```
In [3]: from PIL import Image

        image_path = 'lena.bmp'
        image = Image.open(image_path).convert('L')

        pixels = list(image.getdata())

        hist = [0] * 256
        for pixel in pixels:
            hist[pixel] += 1

        print(hist)
```

先統計每一 pixel 灰階時的值。

```
In [4]: plt.figure(figsize=(10, 5))
        plt.plot(hist, color='black')
        plt.title('Histogram')
        plt.xlabel('Value')
        plt.ylabel('Frequency')
        plt.grid(True)
        plt.savefig('1b.png', bbox_inches='tight', pad_inches=0)
        plt.show()
```

使用 matplotlib 套件將數值轉為 histogram。



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

從上到下由左到右遍歷所有 pixel，當上或左為被標記時，此 pixel 標記為與前者同一 component，當兩者皆被標記過，與前兩者合為同一 component，以上皆無，標記為新區塊。

In [5]:
```python
#前處理
lena = Image.open('lena.bmp').convert('L')
lena_arr = np.array(lena)
threshold = 128
binary_image = (lena_arr >= threshold).astype(int) #trasfer to 01
height, width = binary_image.shape
mark = 2
bounding_boxes = {}

#find connecting components
for y in range(height):
    for x in range(width):
        if binary_image[y, x] == 1:
            top_label = binary_image[y-1, x] if y > 0 else 0
            left_label = binary_image[y, x-1] if x > 0 else 0

            if top_label > 1 and left_label > 1: #if both marked, merge two components
                if top_label != left_label:
                    min_label = min(top_label, left_label)
                    max_label = max(top_label, left_label)
                    # merge
                    binary_image[binary_image == max_label] = min_label
                    min_box = bounding_boxes[min_label]
                    max_box = bounding_boxes[max_label]
                    bounding_boxes[min_label] = (
                        min(min_box[0], max_box[0]),  # Top
                        max(min_box[1], max_box[1]),  # Bottom
                        min(min_box[2], max_box[2]),  # Left
                        max(min_box[3], max_box[3]),  # Right
                        min_box[4] + max_box[4],      # 面積
                        min_box[5] + max_box[5],      # x權重
                        min_box[6] + max_box[6]       # y權重
                    )
                    del bounding_boxes[max_label]

                binary_image[y, x] = min(top_label, left_label) #避免遍歷時未合併完全
                label = binary_image[y, x]
            elif top_label > 1:
                binary_image[y, x] = top_label
                label = top_label
            elif left_label > 1:
                binary_image[y, x] = left_label
                label = left_label
            else:
                binary_image[y, x] = mark
                bounding_boxes[mark] = (y, y, x, x, 1, x, y)  # (top, bottom, left, right, 面積, x權重,
                mark += 1
                label = binary_image[y, x]

            #Reset component information
            top, bottom, left, right, area, sum_x, sum_y = bounding_boxes[label]
            bounding_boxes[label] = (
                min(top, y), max(bottom, y), min(left, x), max(right, x), area + 1,
                sum_x + x, sum_y + y
            )

# Calculate centroids
for label in bounding_boxes.keys():
    bbox = bounding_boxes[label]
    if bbox[4] > 0:
        centroid_x = bbox[5] // bbox[4] #sum_x/area
        centroid_y = bbox[6] // bbox[4] #sum_y/area
        bounding_boxes[label] = (*bbox[:4], bbox[4], centroid_x, centroid_y)
```

完成計算後，將結果貼在圖一的 binary image 上。

In [6]:
```python
HW2c=np.array([[[i]*3 for i in j] for j in binary],dtype='uint8') #turn result in (a) to 512,512,3
for label in bounding_boxes.keys():
    bbox = bounding_boxes[label]
    if bbox[4] > 500:
        #draw bounding box
        cv2.rectangle(HW2c,(bbox[2],bbox[0]),(bbox[3],bbox[1]),(0,255,0),2)
        centroid_x = bbox[5]
        centroid_y = bbox[6]
        # draw +
        line_length = 5
        cv2.line(HW2c, (centroid_x - line_length, centroid_y),
                 (centroid_x + line_length, centroid_y), (255, 0, 0), 2)
        cv2.line(HW2c, (centroid_x, centroid_y - line_length),
                 (centroid_x, centroid_y + line_length), (255, 0, 0), 2)
Image.fromarray(HW2c).save("1c.bmp")
```