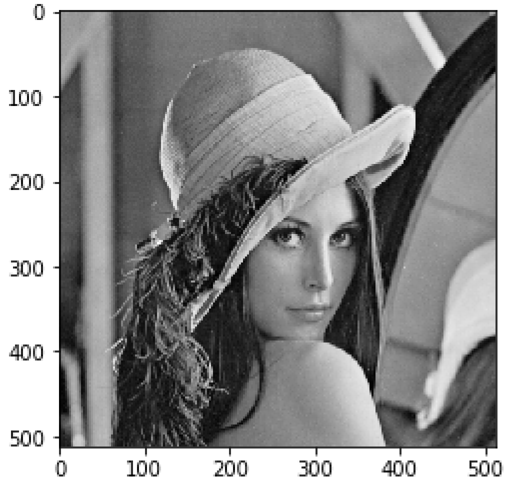
In [1]:

```
import sys
from tqdm import tqdm_notebook
import numpy as np
import cv2
import math
import cv2
import matplotlib.pyplot as plt
import numpy as np
import sys
img = cv2.imread('lena.bmp', 0)
%matplotlib inline
```

原圖

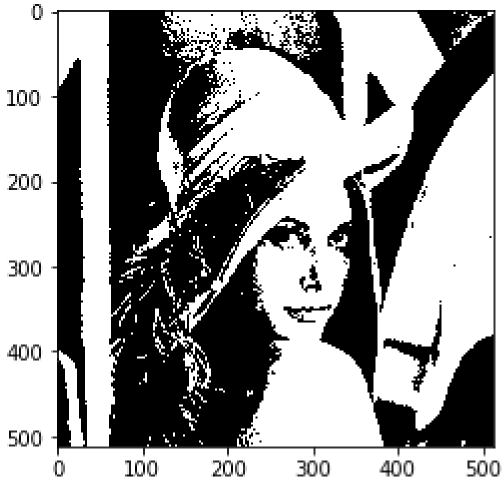
In [2]:

```
plt.imshow(img,cmap='gray')
plt.show()
```



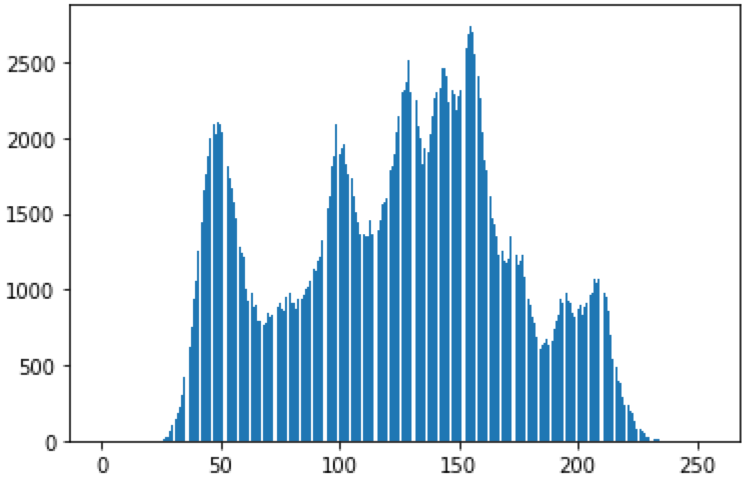
(a) a binary image

In [3]:
```
def binarize(image):
    return (img>127)*255
plt.imshow(binarize (img),cmap='gray')
plt.show()
```



(b) a histogram

In [4]:
```
def histogram(image):
    hist = np.zeros(256,np.int)
    for i in image.reshape(-1,):
        hist[i]+=1
    plt.bar(range(len(hist)), hist)
    plt.show()
histogram(img)
```



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

四連通

In [5]:

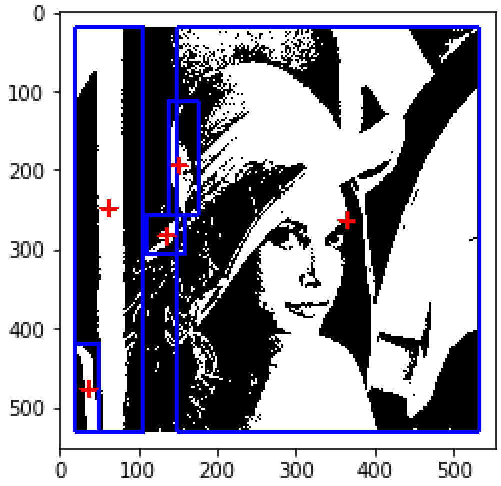
```
def connected_components(img):
    stacked_img = np.stack((img,)*3, axis=-1)
    image=(img>127)*1
    cur_idx=2
    mapping_dict={}
    count_dict={}
    for c in range(image.shape[0]):
        for r in range(image.shape[1]):
            if image[c][r]==1:
                up=0
                left=0
                if r-1<0 and c-1<0:
                    image[c][r]= cur_idx
                if r-1>=0:
                    up=image[c][r-1]
                if c-1>=0:
                    left=image[c-1][r]
                if up==0 and left==0:
                    cur_idx+=1
                    image[c][r]= cur_idx
                elif up==0 or left==0:
                    image[c][r]=max(up,left)
                else:
                    Max=max(up ,left )
                    Min=min(up ,left )
                    assert Min !=0
                    if Max!=Min:
                        while Min in mapping_dict:
                            Min=mapping_dict[Min]
                        mapping_dict[Max]=Min
                    image[c][r]=Min
    for c in range(image.shape[0]):
        for r in range(image.shape[1]):
            if image[c][r]!=0:
                if image[c][r] in mapping_dict:
                    image[c][r]=mapping_dict[image[c][r]]
                if image[c][r] in count_dict:
                    count_dict[image[c][r]]+=1
                else:
                    count_dict[image[c][r]]=1
    count_dict = {k: v for k, v in count_dict.items() if v >= 500}
    stacked_img=((stacked_img>127)*255).astype(np.int32)
    LEN=10
    WID=3
    MAR=20
    stacked_img=cv2.copyMakeBorder(stacked_img,MAR,MAR,MAR,MAR,cv2.BO
RDER_CONSTANT,value=[255,255,255])
    for i,k in enumerate(count_dict.keys()):
        z=np.where(image==k)
        c_max=max(z[0])+MAR
        c_min=min(z[0])+MAR
        r_max=max(z[1])+MAR
        r_min=min(z[1])+MAR
#         print((r_min,c_min),(r_max,c_max))
        stacked_img=cv2.rectangle(stacked_img,(r_min,c_min),(r_max,c_
max),(0,0,255),WID)
```

```
        c_cent=int(z[0].mean())+MAR
        r_cent=int(z[1].mean())+MAR

        stacked_img=cv2.line(stacked_img, (r_cent,c_cent - LEN ), (r_cent,c_cent + LEN), (255,0,0), WID)
        stacked_img=cv2.line(stacked_img, ( r_cent-LEN,c_cent), ( r_cent+LEN,c_cent), (255,0,0), WID)
    return stacked_img
plt.imshow(connected_components (img))
```

Out[5]:

```
<matplotlib.image.AxesImage at 0x7ff8722a0b00>
```


0
100
200
300
400
500
0
100
200
300
400
500