In [ ]:
```
import cv2
import google.colab.patches as colab
import matplotlib.pyplot as plt
import numpy
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.moun
t("/content/drive", force_remount=True).
```

In [ ]:
```
in_mat = cv2.imread("/content/drive/MyDrive/Colab Notebooks/histogram_equalization/lena.
```

In [ ]:
```
colab.cv2_imshow(in_mat)
```

get the rows and columns of the input image

In [ ]:
```
rows = in_mat.shape[0]
cols = in_mat.shape[1]
```

Step1: compute the histogram of input image

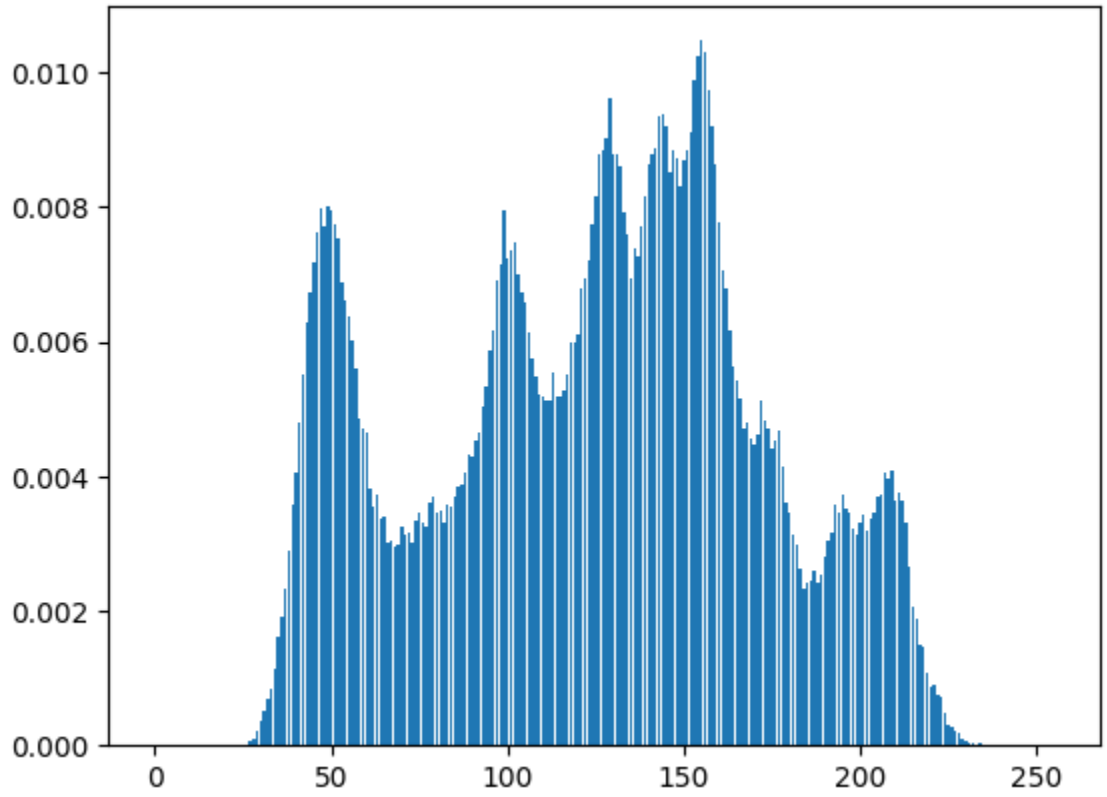
In [ ]:
```
in_hist = numpy.zeros((256), dtype=float)

for r in range(rows):
  for c in range(cols):

    gray_value = in_mat[r, c]
    in_hist[gray_value] = in_hist[gray_value] + 1
```

```
for i in range(256):
  in_hist[i] = in_hist[i] / (rows * cols)
```

display the histogram of the input image

In [ ]:
```
plt.bar(range(256), in_hist)
```

Out[ ]:
```
<BarContainer object of 256 artists>
```

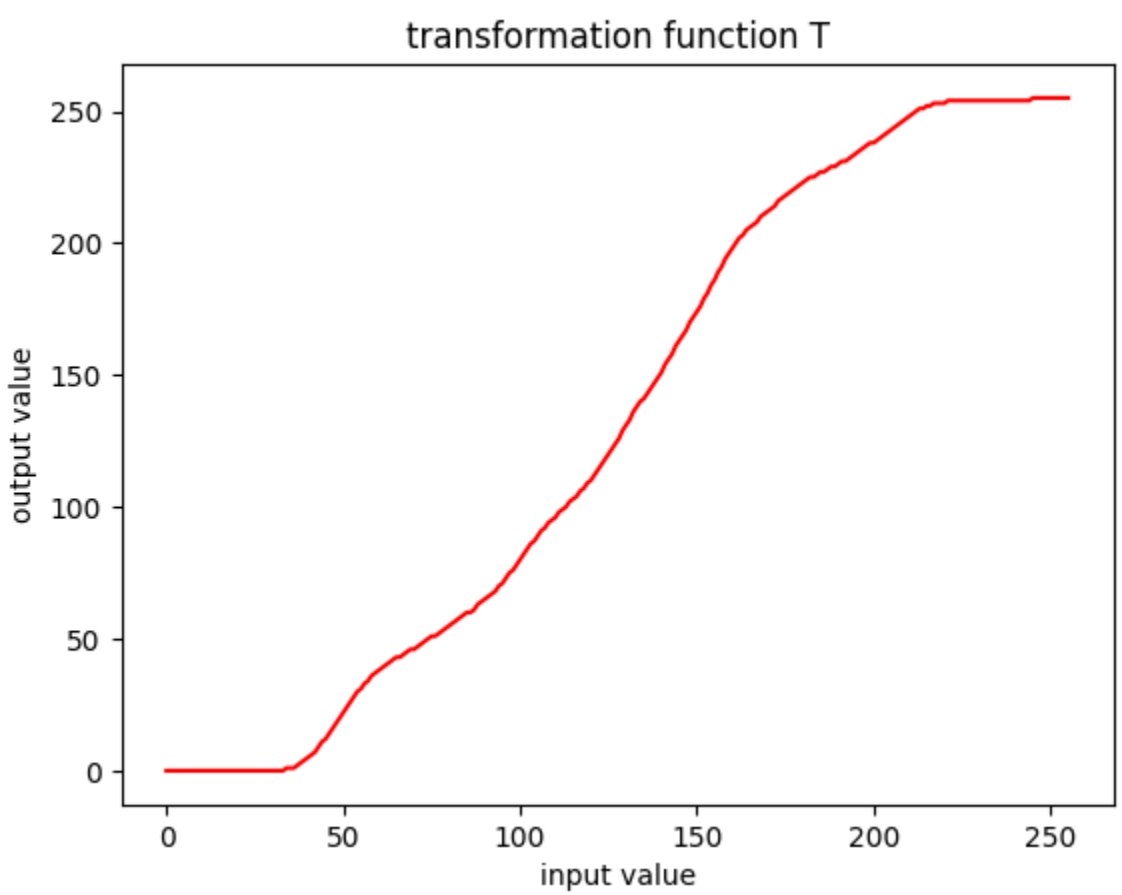


Step2: compute the transformation function $T(.)$

In [ ]:
```
# T: transformation function
T = numpy.zeros((256), dtype=int)

# acc_pr: the accumulated probability of the input histogram
acc_pr = 0.0;

for x in range(256):
  acc_pr = acc_pr + in_hist[x]
  T[x] = int(255.0 * acc_pr)
```

display the transformation function $T(.)$

In [ ]:
```
plt.title('transformation function T')
plt.xlabel('input value')
plt.ylabel('output value')
plt.plot(range(256), T, 'r')
plt.show()
```



Step 3: transform the value of each pixel

In [ ]:
```
# create an output image
out_mat = numpy.zeros((rows, cols), dtype=numpy.uint8)

for r in range(rows):
  for c in range(cols):
    gray_value = in_mat[r, c]
    out_value = T[gray_value]
    out_mat[r, c] = out_value
```

display the image after histogram equalization

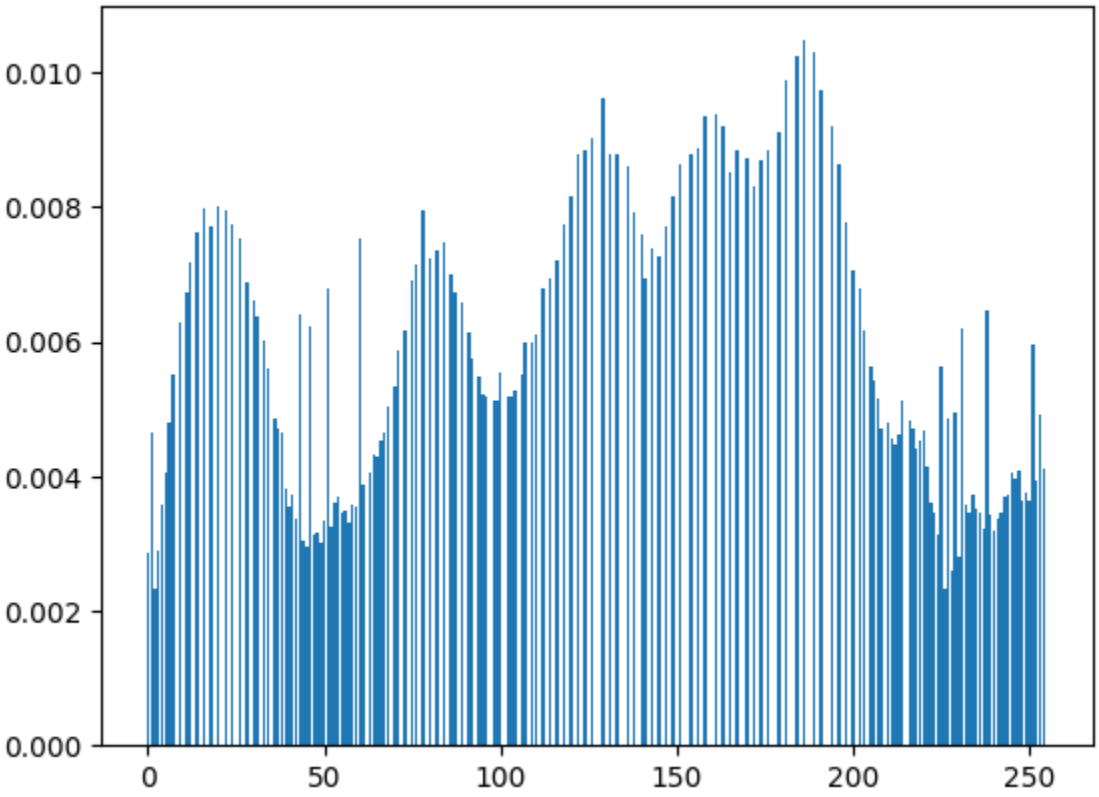
In [ ]:
```
colab.cv2_imshow(out_mat)
```

display the histogram of the image after histogram equalization

In [ ]:
```
out_hist = numpy.zeros((256), dtype=float)

for r in range(rows):
  for c in range(cols):
    gray_value = out_mat[r, c]
    out_hist[gray_value] = out_hist[gray_value] + 1

for i in range(256):
  out_hist[i] = out_hist[i] / (rows * cols)
```

In [ ]:
```
plt.bar(range(out_hist.shape[0]), out_hist)
```

Out[ ]:
```
<BarContainer object of 256 artists>
```


0.010
0.008
0.006
0.004
0.002
0.000
0
50
100
150
200
250