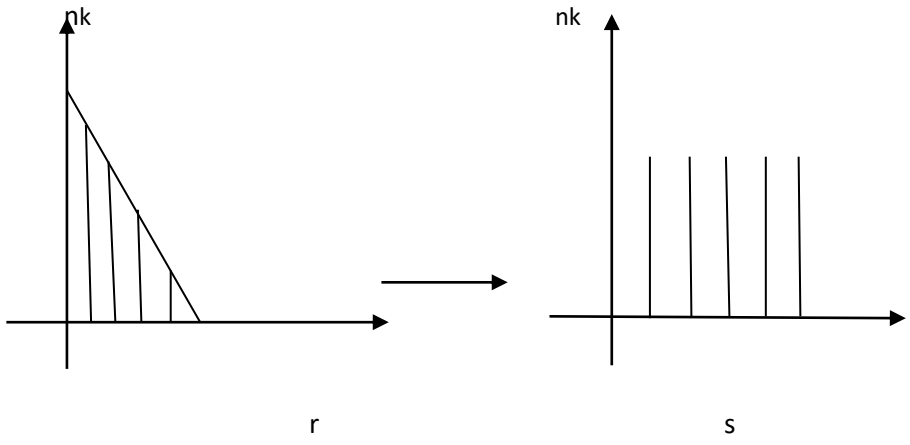
AIM: To Perform Histogram Equalization

THEORY:

Histogram Equalisation

There are many applications wherein we need a flat histogram. This cannot be achieved by histogram stretching. Hence histogram equalization was introduced.

A perfect image is one which has equal number of pixels in all its grey levels. Hence our objective is not only to spread the dynamic range, but also to have pixels in all grey levels. This technique is known as Histogram equalization.



The transformation must satisfy the two conditions:

- T (r) must be single valued and monotonically increasing in the interval $0 < r < 1$ and,
- $0 < T(r) < 1$ for $0 < r < 1$.

Hence the range of r is taken as [0,1]. This is called the normalized range. This range is taken for simplicity. So instead of r being in the range [0,255] we take [0,1].

RESULT:

1] The image initially had a lot of pixels with frequency < 150 and a very few frequencies of pixel > 200. Hence the initial image was more on the darker (black) side.

2] After histogram equalisation, the pixels were distributed evenly and we got nearly the same number of frequencies of pixels throughout the image.

3] Hence, the resulting image had good contrast.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
import pandas as pd

img = cv2.imread('/content/pic.jpg',0)

img = img[100:600,100:1000]

img.shape
```

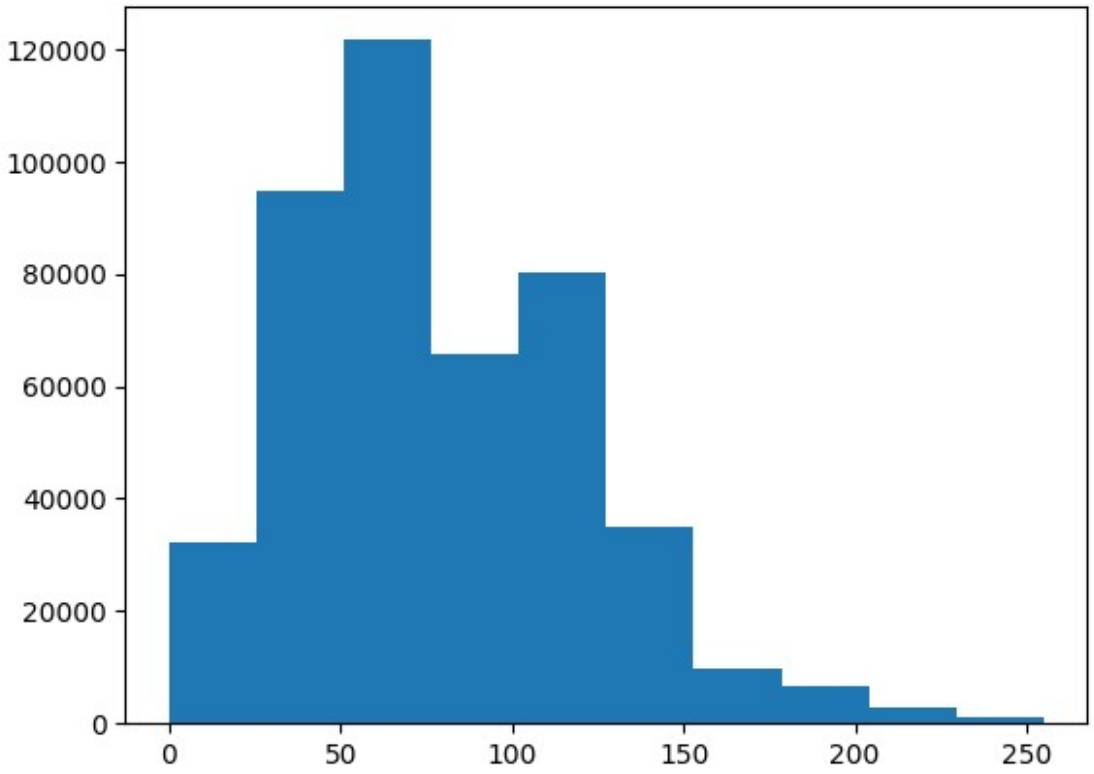
```
(500, 900)
```

```
cv2_imshow(img)
```

```
new = np.ravel(img)
diff = new

plt.hist(new)
```

```
(array([ 32278.,  94720., 121557.,  65825.,  80138.,  34878.,   9747.,
         6771.,   2957.,   1129.]),
 array([  0. ,  25.5,  51. ,  76.5, 102. , 127.5, 153. , 178.5, 204. ,
        229.5, 255. ]),
 <BarContainer object of 10 artists>)
```




```
df = pd.DataFrame(new,columns=["n"])

n = pd.unique(df["n"])

n.sort()

count = [0 for i in range(len(n))]

for i in range(len(new)):
  for j in range(len(n)):
    if n[j]==new[i]:
      count[j]+=1
      break

Table = pd.DataFrame(n,columns=["n"])

Table["nk"] = count

Table["pk"] = Table["nk"] / sum(Table["nk"])

cdf = [0 for i in range(len(n))]

cdf[0] = Table.loc[0,"pk"]

for i in range(1,len(n)):
    cdf[i] = cdf[i-1] + Table.loc[i,"pk"]

Table['cdf'] = cdf
```

```
Table['(n-1)*cdf'] = 255*Table["cdf"]

Table['Rounded'] = round(Table['(n-1)*cdf'])

Table
```

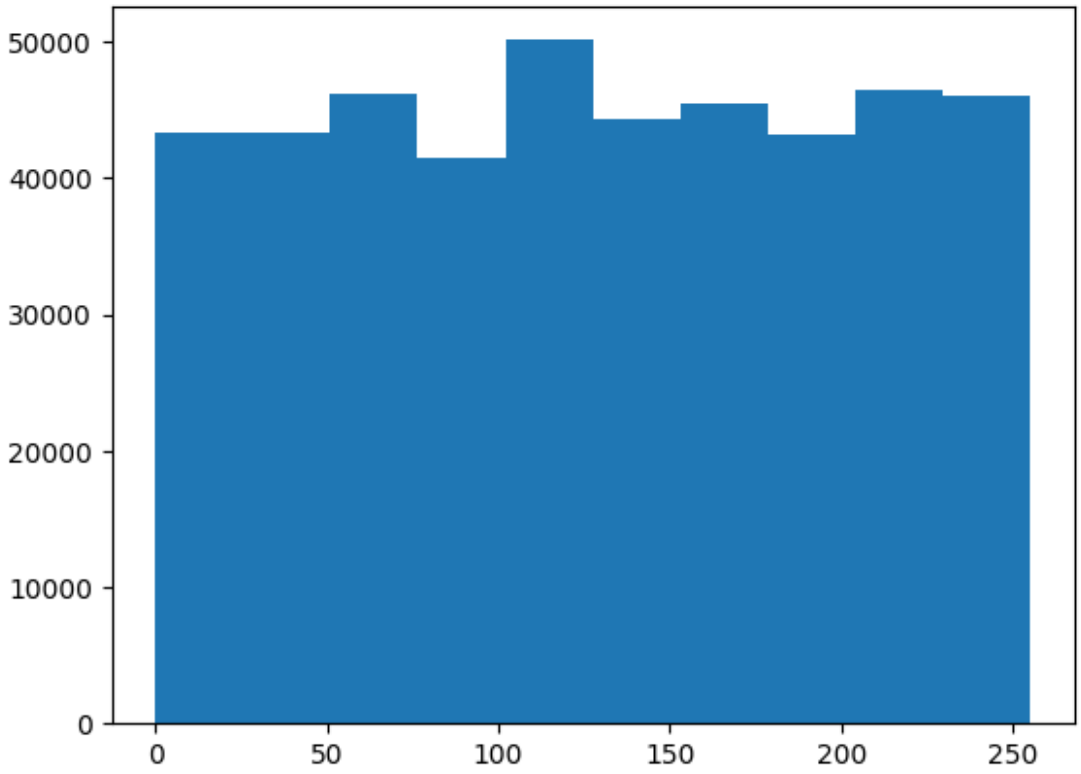
```
       n   nk        pk       cdf   (n-1)*cdf  Rounded
0      0  294  0.000653  0.000653    0.166600      0.0
1      1  144  0.000320  0.000973    0.248200      0.0
2      2  166  0.000369  0.001342    0.342267      0.0
3      3  272  0.000604  0.001947    0.496400      0.0
4      4  330  0.000733  0.002680    0.683400      1.0
..   ...  ...       ...       ...         ...      ...
251  251   20  0.000044  0.999720  254.928600    255.0
252  252   22  0.000049  0.999769  254.941067    255.0
253  253   19  0.000042  0.999811  254.951833    255.0
254  254   16  0.000036  0.999847  254.960900    255.0
255  255   69  0.000153  1.000000  255.000000    255.0

[256 rows x 6 columns]
```

```
for i in range(len(new)):
  diff[i] = int(Table.loc[new[i],"Rounded"])

plt.hist(diff)
```

```
(array([43393., 43386., 46203., 41452., 50133., 44378., 45435.,
43133.,
        46477., 46010.]),
 array([  0. ,  25.5,  51. ,  76.5, 102. , 127.5, 153. , 178.5, 204. ,
        229.5, 255. ]),
 <BarContainer object of 10 artists>)
```




```
diff = diff.reshape(500,900)
cv2_imshow(diff)
```