In [7]:

```
#Part 1:
#This is our intitial image which we uploaded from my file.
from PIL import Image

file = "/Users/mehradhq/Computer_Vision/Project_1/Image_project.jpeg"
img = Image.open(file)
#below we have the image on which we want to examine
img
```

Out[7]:

In [10]:

```
#question 1:
#This is the part where we half the R values of our pixels.
#It is better to each time load your intitial image so that the changes you make do'nt build up on each other.
from PIL import Image

file = "/Users/mehradhq/Computer_Vision/Project_1/Image_project.jpeg"
img = Image.open(file)
#below load the RGB values of the image.
im=img.load()
#below is the image dimension: x is the number of columns and y is the number rows.
[x,y]=img.size
#The following gives the RGB value of pixel with x=a and y=b: im [a,b]

for i in range (x):
    for j in range (y):
        [r,g,b]=im [i,j]
        r_new=r//2
        value=(r_new,g,b)
        img.putpixel((i, j), value)
img
#by using the code below we saved our image in the named file.
#img.save('question_1_red_halved.jpg')
```

In [11]:

```
#question 1:
#This is the part where we set the R values to zero.
from PIL import Image

file = "/Users/mehradhq/Computer_Vision/Project_1/Image_project.jpeg"
img = Image.open(file)
im=img.load()
for i in range (x):
    for j in range (y):
        [r,g,b]=im [i,j]
        r_new=0
        value=(r_new,g,b)
        img.putpixel((i, j), value)
img
#by using the code below we saved our image in the named file.
#img.save('question_1_red_Zero.jpg')
```

In [12]:

```
#question 1:
#This is the part where we halve both R and G values
from PIL import Image

file = "/Users/mehradhq/Computer_Vision/Project_1/Image_project.jpeg"
img = Image.open(file)
im=img.load()
for i in range (x):
    for j in range (y):
        [r,g,b]=im [i,j]
        r//=2
        g//=2
        value=(r,g,b)
        img.putpixel((i, j), value)
img
#by using the code below we saved our image in the named file.
#img.save('question_1_red_green_halved.jpg')
```

In [13]:

```
#question 1:
#This is the part where we set both R and G values to zero.
from PIL import Image

file = "/Users/mehradhq/Computer_Vision/Project_1/Image_project.jpeg"
img = Image.open(file)
im=img.load()
for i in range (x):
    for j in range (y):
        [r,g,b]=im [i,j]
        r=0
        g=0
        value=(r,g,b)
        img.putpixel((i, j), value)
img
#by using the code below we saved our image in the named file.
#img.save('question_1_red_green_Zero.jpg')
```

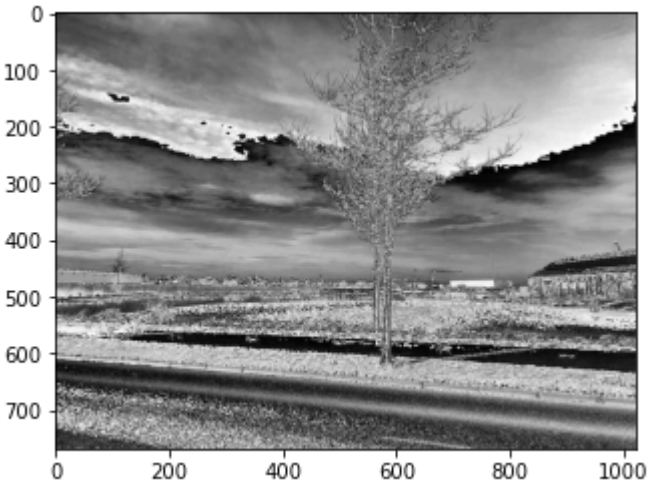
In [14]:

```
#question 2:
#This is the python in-built function of turning to grey scale image
from PIL import Image
import matplotlib.pyplot as plt
file = "/Users/mehradhq/Computer_Vision/Project_1/Image_project.jpeg"
img = Image.open(file)
imgGray = img.convert('L')
imgGray
#using the below command we saved our gray scale image in a new file.
#imgGray.save('test_gray.jpg')
```

Out[14]:

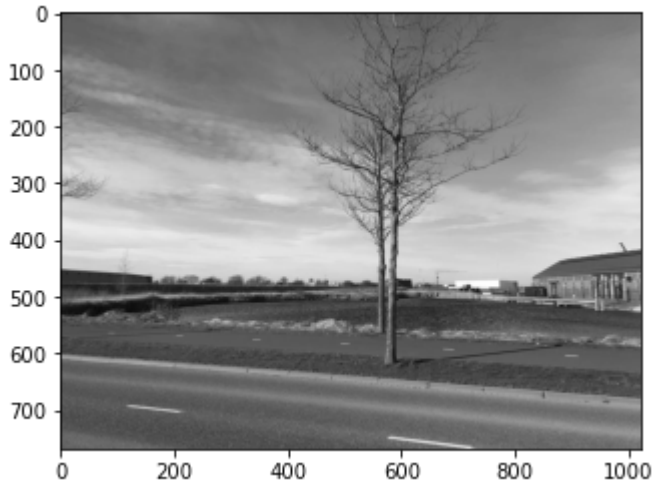
In [16]:

```
#question 2:
#This is turning the image to a grey scale image using weighted sums of 1/3. (As a result the gray scale image is too dark)
from matplotlib import pyplot as plt
import matplotlib.image as mpimg

img = mpimg.imread("/Users/mehradhq/Computer_Vision/Project_1/Image_project.jpeg")

R, G, B = img[:,:,0], img[:,:,1], img[:,:,2]
imgGray = (1/3)*( R + G + B)
plt.imshow(imgGray, cmap='gray')
plt.show()
```



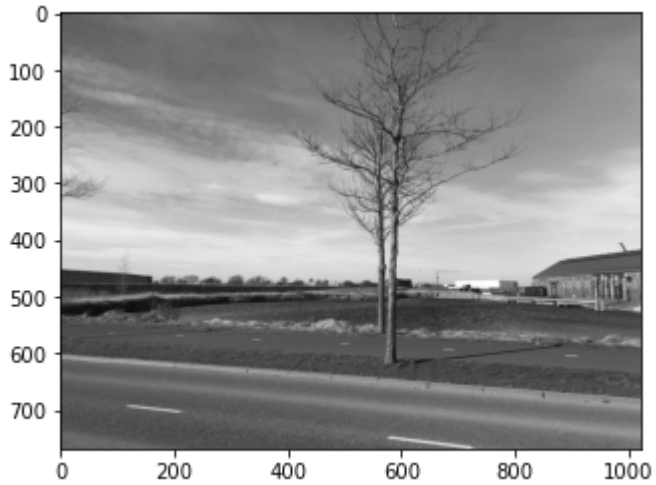
In [18]:

```
#question 2:
#This is turning the image to a grey scale image using weighted sums of wR = 0.299, wG = 0.587, wB = 0.114 (analog color
from matplotlib import pyplot as plt
import matplotlib.image as mpimg

img = mpimg.imread("/Users/mehradhq/Computer_Vision/Project_1/Image_project.jpeg")

R, G, B = img[:,:,0], img[:,:,1], img[:,:,2]
imgGray = 0.299*R +0.587*G + 0.114*B
plt.imshow(imgGray, cmap='gray')
plt.show()
```



In [70]:

```
#question 2:
#This is turning the image to grey scale using weighted sums of wR = 0.2125, wG = 0.7154, wB = 0.072 (HDTV).
#There is not much difference between this version and the previous one.
from matplotlib import pyplot as plt
import matplotlib.image as mpimg

img = mpimg.imread("Image_project.jpeg")

R, G, B = img[:,:,0], img[:,:,1], img[:,:,2]
imgGray = 0.2125*R +0.7154*G + 0.072*B
plt.imshow(imgGray, cmap='gray')
plt.show()
```



In [1]:

```
#question3
#below we get each pixel and calculate the the average of the three pixels neighbouring it along with the main pixel(the four p
#assign the average amount to one pixel in the blank image. We go to the next 2by2 grid and do the same. It is like mapping 2
#as a result the image becomes kind of blur but still good.
from PIL import Image
import cv2 as cv
import numpy as np
file = "/Users/mehradhq/Computer_Vision/Project_1/test_gray.jpg" #this is the gray-scale image we saved earlier and we are
img = Image.open(file)
im=img.load()
[x,y]=img.size
#we create a blank image half the height and half the width. The 1 is because it is gray scale.
blank=np.zeros((y//2,x//2,1), dtype="uint8")
#The step 2 is because of the 2by2 grid.
for i in range (1,x-1,2):
    for j in range (1,y-1,2):
        #the coments below were for controlling my loop and whether it was functioning okay or not.
        #print (i,j)
        #print (im[i,j])
        #here we get the four neighbouring pixels.
        main_pixel=im[i,j]
        left_pixel=im[i-1,j]
        above_pixel=im[i,j-1]
        above_left_pixel=im[i-1,j-1]
        #print (main_pixel,left_pixel,above_pixel,above_left_pixel)
        #below we calculate the average
        value=(main_pixel+left_pixel+above_pixel+above_left_pixel)//4
        blank[j//2, i//2]=value
        #print (im[i,j])
cv.imshow("blank",blank)
cv.imwrite('/Users/mehradhq/Computer_Vision/Project_1/question3_average.png',blank)
cv.waitKey(0)
```

Out[1]:

48

In [2]:

```
#question3
#process like above but two times.
#below we get each pixel and calculate the the average of the three pixels neighbouring it along with the main pixel(the four p
#assign the average amount to one pixel in the blank image. We go to the next 2by2 grid and do the same. It is like mapping 2
#as a result the image becomes kind of blur but still good.
from PIL import Image
import cv2 as cv
import numpy as np
file = "/Users/mehradhq/Computer_Vision/Project_1/question3_average.png" #this is the decreased resolution gray-scale ima
img = Image.open(file)
im=img.load()
[x,y]=img.size
#we create a blank image half the height and half the width. The 1 is because it is gray scale.
blank=np.zeros((y//2,x//2,1), dtype="uint8")
#The step 2 is because of the 2by2 grid.
for i in range (1,x-1,2):
    for j in range (1,y-1,2):
        #the coments below were for controlling my loop and whether it was functioning okay or not.
        #print (i,j)
        #print (im[i,j])
        #here we get the four neighbouring pixels.
        main_pixel=im[i,j]
        left_pixel=im[i-1,j]
        above_pixel=im[i,j-1]
        above_left_pixel=im[i-1,j-1]
        #print (main_pixel,left_pixel,above_pixel,above_left_pixel)
        #below we calculate the average
        value=(main_pixel+left_pixel+above_pixel+above_left_pixel)//4
        blank[j//2, i//2]=value
        #print (im[i,j])
cv.imshow("blank",blank)
cv.imwrite('/Users/mehradhq/Computer_Vision/Project_1/question3_average_two_times.png',blank)
cv.waitKey(0)
```

Out[2]:

48

In [1]:

```
#question3
#below we get each pixel and calculate the the maximum of the three pixels neighbouring it along with the main pixel(the fou
#assign the maximum amount to one pixel in the blank image. We go to the next 2by2 grid and do the same. It is like mapping
#as a result the image becomes kind of blur. The image gets brighter.
from PIL import Image
import cv2 as cv
import numpy as np
file = "/Users/mehradhq/Computer_Vision/Project_1/test_gray.jpg" #this is the gray-scale image we saved earlier and we are
img = Image.open(file)
im=img.load()
[x,y]=img.size
#we create a blank image half the height and half the width. The 1 is because it is gray scale.
blank=np.zeros((y//2,x//2,1), dtype="uint8")
#The step 2 is because of the 2by2 grid.
for i in range (1,x-1,2):
    for j in range (1,y-1,2):
        #the coments below were for controlling my loop and whether it was functioning okay or not.
        #print (i,j)
        #print (im[i,j])
        #here we get the four neighbouring pixels.
        main_pixel=im[i,j]
        left_pixel=im[i-1,j]
        above_pixel=im[i,j-1]
        above_left_pixel=im[i-1,j-1]
        #print (main_pixel,left_pixel,above_pixel,above_left_pixel)
        #below we calculate the average
        value=max(main_pixel,left_pixel,above_pixel,above_left_pixel)
        blank[j//2, i//2]=value
        #print (im[i,j])
cv.imshow("blank",blank)
cv.imwrite('/Users/mehradhq/Computer_Vision/Project_1/question3_maximum.png',blank)
cv.waitKey(0)
```

Out[1]:

48

In [1]:

```
#question3
#process like above but aplied on the initial two times. that is it is applied on the result image of the previous program one tim
#below we get each pixel and calculate the the maximum of the three pixels neighbouring it along with the main pixel(the fou
#assign the maximum amount to one pixel in the blank image. We go to the next 2by2 grid and do the same. It is like mapping
#as a result the image becomes kind of blur. the image gets more brighter.
from PIL import Image
import cv2 as cv
import numpy as np
file = "/Users/mehradhq/Computer_Vision/Project_1/question3_maximum.png"
img = Image.open(file)
im=img.load()
[x,y]=img.size
#we create a blank image half the height and half the width. The 1 is because it is gray scale.
blank=np.zeros((y//2,x//2,1), dtype="uint8")
#The step 2 is because of the 2by2 grid.
for i in range (1,x-1,2):
    for j in range (1,y-1,2):
        #the coments below were for controlling my loop and whether it was functioning okay or not.
        #print (i,j)
        #print (im[i,j])
        #here we get the four neighbouring pixels.
        main_pixel=im[i,j]
        left_pixel=im[i-1,j]
        above_pixel=im[i,j-1]
        above_left_pixel=im[i-1,j-1]
        #print (main_pixel,left_pixel,above_pixel,above_left_pixel)
        #below we calculate the average
        value=max(main_pixel,left_pixel,above_pixel,above_left_pixel)
        blank[j//2, i//2]=value
        #print (im[i,j])
cv.imshow("blank",blank)
cv.imwrite('/Users/mehradhq/Computer_Vision/Project_1/question3_maximum_two_times.png',blank)
cv.waitKey(0)
```

Out[1]:

48

In [2]:

```
#question3
#below we get each pixel and calculate the the maximum of the three pixels neighbouring it along with the main pixel(the fou
#assign the maximum amount to one pixel in the blank image. We go to the next 2by2 grid and do the same. It is like mapping
#as a result the image becomes kind of blur. The image gets darker.
from PIL import Image
import cv2 as cv
import numpy as np
file = "/Users/mehradhq/Computer_Vision/Project_1/test_gray.jpg" #this is the gray-scale image we saved earlier and we are
img = Image.open(file)
im=img.load()
[x,y]=img.size
#we create a blank image half the height and half the width. The 1 is because it is gray scale.
blank=np.zeros((y//2,x//2,1), dtype="uint8")
#The step 2 is because of the 2by2 grid.
for i in range (1,x-1,2):
    for j in range (1,y-1,2):
        #the coments below were for controlling my loop and whether it was functioning okay or not.
        #print (i,j)
        #print (im[i,j])
        #here we get the four neighbouring pixels.
        main_pixel=im[i,j]
        left_pixel=im[i-1,j]
        above_pixel=im[i,j-1]
        above_left_pixel=im[i-1,j-1]
        #print (main_pixel,left_pixel,above_pixel,above_left_pixel)
        #below we calculate the average
        value=min(main_pixel,left_pixel,above_pixel,above_left_pixel)
        blank[j//2, i//2]=value
        #print (im[i,j])
cv.imshow("blank",blank)
cv.imwrite('/Users/mehradhq/Computer_Vision/Project_1/question3_minimum.png',blank)
cv.waitKey(0)
```

Out[2]:

48

In [3]:

```
#question3
#process like above but aplied on the initial two times. that is it is applied on the result image of the previous program one tim
#below we get each pixel and calculate the the maximum of the three pixels neighbouring it along with the main pixel(the fou
#assign the maximum amount to one pixel in the blank image. We go to the next 2by2 grid and do the same. It is like mapping
#as a result the image becomes kind of blur. the image gets more darker.
from PIL import Image
import cv2 as cv
import numpy as np
file = "/Users/mehradhq/Computer_Vision/Project_1/question3_minimum.png"
img = Image.open(file)
im=img.load()
[x,y]=img.size
#we create a blank image half the height and half the width. The 1 is because it is gray scale.
blank=np.zeros((y//2,x//2,1), dtype="uint8")
#The step 2 is because of the 2by2 grid.
for i in range (1,x-1,2):
    for j in range (1,y-1,2):
        #the coments below were for controlling my loop and whether it was functioning okay or not.
        #print (i,j)
        #print (im[i,j])
        #here we get the four neighbouring pixels.
        main_pixel=im[i,j]
        left_pixel=im[i-1,j]
        above_pixel=im[i,j-1]
        above_left_pixel=im[i-1,j-1]
        #print (main_pixel,left_pixel,above_pixel,above_left_pixel)
        #below we calculate the average
        value=min(main_pixel,left_pixel,above_pixel,above_left_pixel)
        blank[j//2, i//2]=value
        #print (im[i,j])
cv.imshow("blank",blank)
cv.imwrite('/Users/mehradhq/Computer_Vision/Project_1/question3_minimum_two_times.png',blank)
cv.waitKey(0)
```

Out[3]:

48

In [4]:

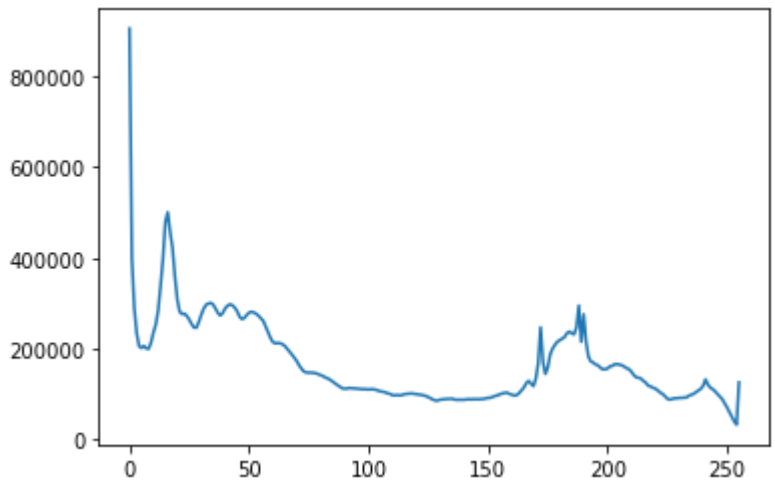
```
#the best possible strategy is the average among the three above examined methods.
```

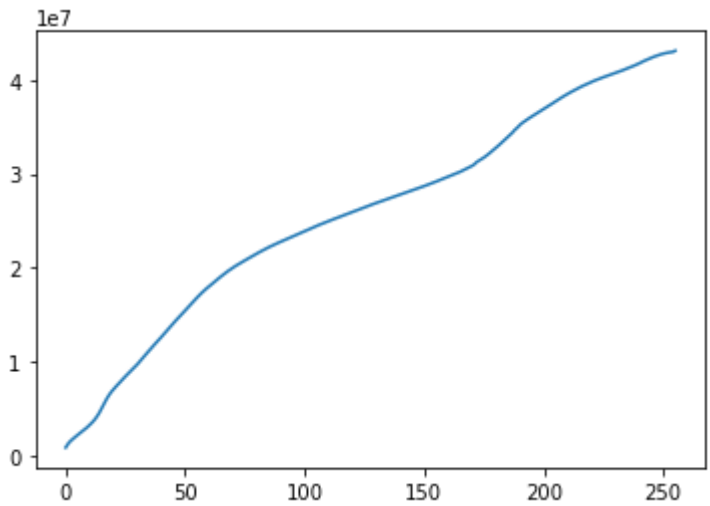
In [58]:

```
#the below codes are mine. it does work but the runtime is very high. so I am putting more efficient codes which I have found
# #part 2:
# #histogram
# #the below image is the image of Red Lion Square Park in Holborn, London, UK, January 2020. Road, street, buildings, tre
# import matplotlib.pyplot as plt
# import cv2 as cv
# from PIL import Image
# #the file address
# file = "/Users/mehradhq/Computer_Vision/Part2_question1.jpg"
# #reads the file
# img = Image.open(file)
# #loads the file pixel values
# im=img.load()
# #empty list for the values.
# li=[]
# [x,y]=img.size
# for i in range (x):
#     for j in range (y):
#         li.append(im[i,j])
# result=[]
# for i in li:
#     con=True
#     while con==True:
#         for j in i:
#             result.append(j)
#             con=False
# result
# #we set result in an ascending order so that we use the following loop. if the list is not an acsending order you cannot use th
# result.sort()
# hist=[]
# temporary=0
# for i in result:
#     if temporary==i:
#         continue
#     temporary=i
#     C=result.count(i)
#     hist.append(C)
# hist
# #hist is a list of the number of repititions of the value, that is how many pixels have a ceratin value.
```

In [7]:

```
#part 2: question1: Uniform image from unsplash.com. roughly uniform except for very very low values.
#histogram
#the below image is the image of Red Lion Square Park in Holborn, London, UK, January 2020. Road, street, buildings, trees
import matplotlib.pyplot as plt
import cv2 as cv
import numpy as np
file = "/Users/mehradhq/Computer_Vision/Part2_question1.jpg"
img = cv.imread(file)

hist = cv.calcHist([img],[0],None,[256],[0,256])
plt.plot(hist)
plt.show()

histogram_cumulative=np.array(hist)
histogram_cumulative=np.cumsum(hist)
plt.plot(histogram_cumulative)
plt.show()
```





In [8]:

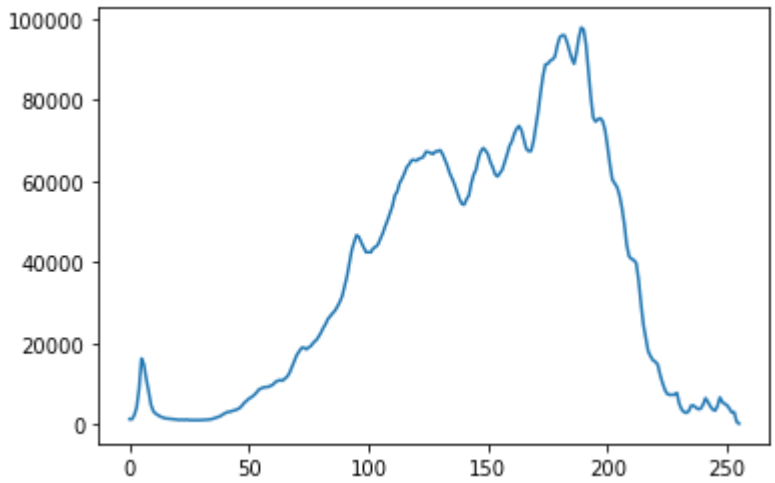
```
#part 2: question1: inverting image
import cv2 as cv


image = cv.imread("/Users/mehradhq/Computer_Vision/Part2_question1.jpg")
invert = cv.bitwise_not(image)
cv.imshow("invert", invert)
cv.imwrite("/Users/mehradhq/Computer_Vision/Part2_question1_invert.jpg",invert)
cv.waitKey(0)
```

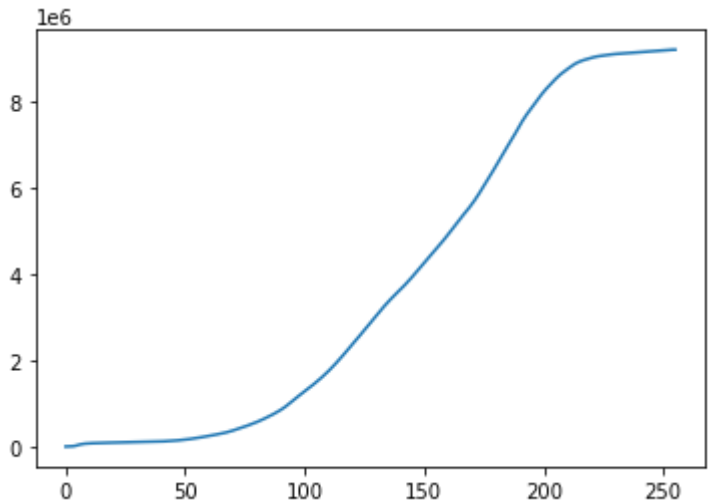
Out[8]:

48

In [21]:

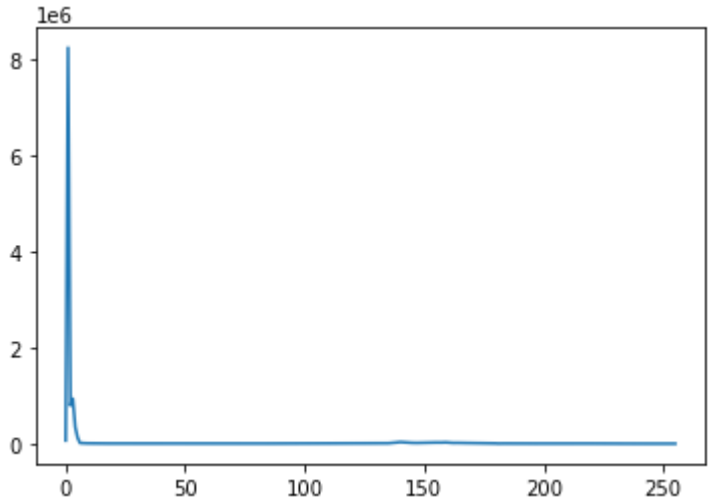
```
#part 2: question2: bias towards high intensity values. histogram and cumulative histogram.
import matplotlib.pyplot as plt
import cv2 as cv

file = "/Users/mehradhq/Computer_Vision/Part2_question2_light.jpg"
img = cv.imread(file)

hist = cv.calcHist([img],[0],None,[256],[0,256])
plt.plot(hist)
plt.show()

histogram_cumulative=np.array(hist)
histogram_cumulative=np.cumsum(hist)
plt.plot(histogram_cumulative)
plt.show()
```

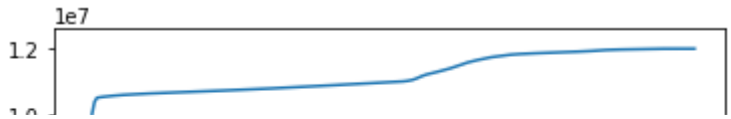




In [15]:

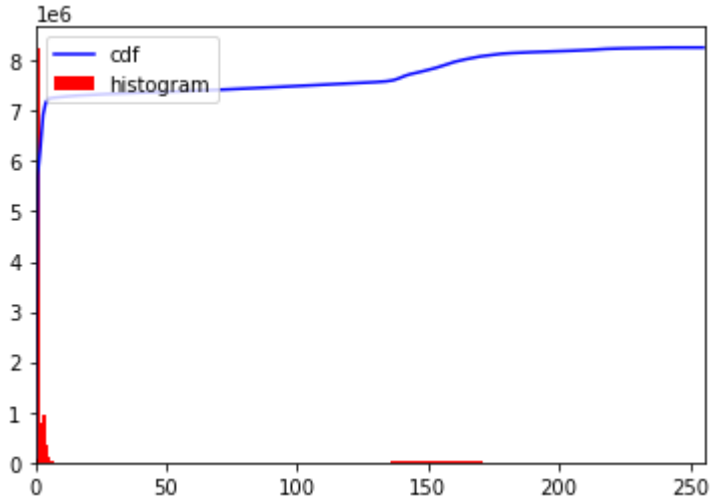
```
#part 2: question2: bias towards low intensity values. histogram and cumulative histogram.
import matplotlib.pyplot as plt
import cv2 as cv

file = "/Users/mehradhq/Computer_Vision/Part2_question2_dark.jpg"
img = cv.imread(file)

hist = cv.calcHist([img],[0],None,[256],[0,256])
plt.plot(hist)
plt.show()

histogram_cumulative=np.array(hist)
histogram_cumulative=np.cumsum(hist)
plt.plot(histogram_cumulative)
plt.show()
```





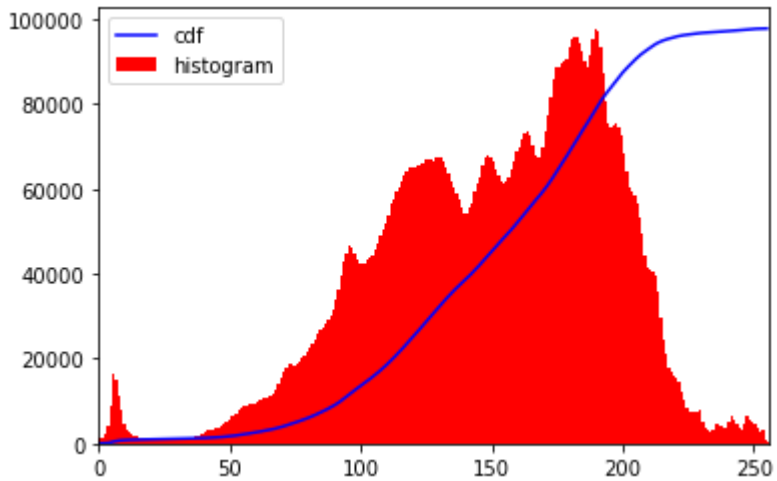
In [13]:

```
#part2: question3 : dark
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

file = "/Users/mehradhq/Computer_Vision/Part2_question2_dark.jpg"
img = cv.imread(file)
grayimg = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
cv.imshow('gray_image',grayimg)
cv.waitKey(0)
cv.destroyAllWindows()

hist,bins = np.histogram(grayimg.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(grayimg.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()


equ_dark = cv.equalizeHist(grayimg)
cv.imshow('equ_dark.png',equ_dark)
cv.imwrite('/Users/mehradhq/Computer_Vision/Project_1/equal_dark_part2_q3.png',equ_dark)
cv.waitKey(0)
cv.destroyAllWindows()
```



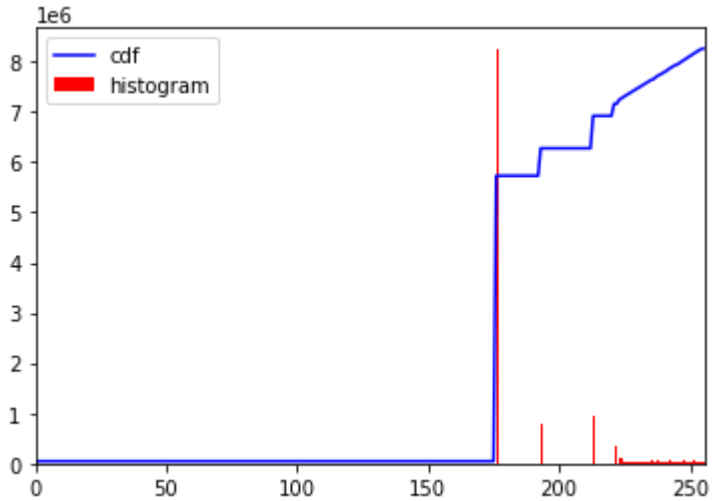
In [14]:

```
#part2: question3 : light
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

file = "/Users/mehradhq/Computer_Vision/Part2_question2_light.jpg"
img = cv.imread(file)
grayimg = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
cv.imshow('gray_image',grayimg)
cv.waitKey(0)
cv.destroyAllWindows()

hist,bins = np.histogram(grayimg.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(grayimg.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()


equ_light = cv.equalizeHist(grayimg)
cv.imshow('equ_light.png',equ_light)
cv.imwrite('/Users/mehradhq/Computer_Vision/Project_1/equal_light_part2_q3.png',equ_light)
cv.waitKey(0)
cv.destroyAllWindows()
```



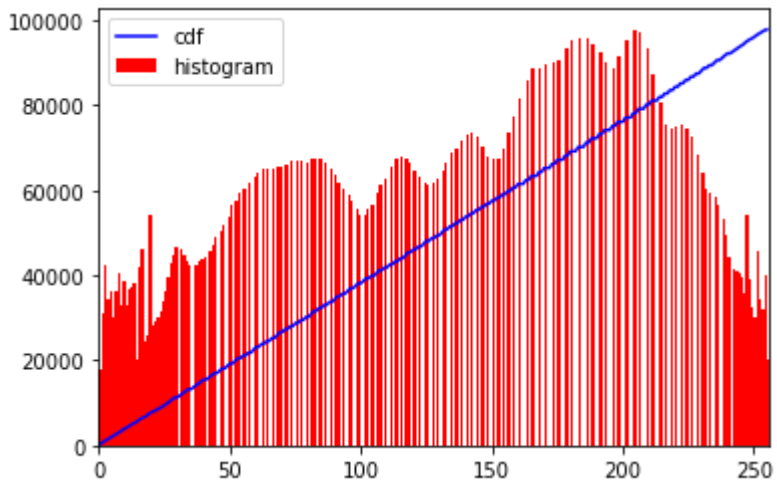
In [15]:

```
#histogram and cumulative histogram for the equalization of the dark image
hist,bins = np.histogram(equ_dark.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(equ_dark.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```



In [16]:

```
#histogram and cumulative histogram for the equalization of the light image
hist,bins = np.histogram(equ_light.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(equ_light.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```



In [1]:

```
#part 2: question4: the images has roughly a low range. increasing brightness.
import matplotlib.pyplot as plt
import cv2 as cv

file = "/Users/mehradhq/Computer_Vision/Part2_q4.jpg"
img = cv.imread(file)
#you can change the threshold 30
def increase_brightness(img, value=30):
    #the image is gray scale
    h, s, v = cv.split(img)

    lim = 255 - value
    #you can change 230
    v[v > lim] = 230
    v[v <= lim] += value

    final_img = cv.merge((h, s, v))
    return final_img
bright=increase_brightness(img)
cv.imshow("brighter",bright)
cv.imwrite("/Users/mehradhq/Computer_Vision/Part2_q4_bright.jpg",bright)
cv.waitKey(0)
```

Out[1]:

48

In [1]:

```
#part3: question1: This is an average filter. if we use 3by3 smoothing filter the image will get darker. Using 5by5 filter
#the image will not change that much. if we increase the size of our filter from 3by3 to 10by10 for instance, we will see the pi
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread("/Users/mehradhq/Computer_Vision/part3.jpg")
kernel = np.ones((3,3),np.float32)/25
dst = cv.filter2D(img,-1,kernel)
cv.imshow("original", img)
cv.imshow("averaging",dst)
cv.imwrite("/Users/mehradhq/Computer_Vision/part3_q1_smoothing_average_filter.jpg",dst)
cv.waitKey(0)
```

Out[1]:

48

In [1]:

```
#part3:question1:gaussian kernel. not that much difference is seen compared to the original image.
import cv2 as cv
img = cv.imread("/Users/mehradhq/Computer_Vision/part3.jpg")
blur = cv.GaussianBlur(img,(3,3),0)
cv.imshow("gaussian",blur)
cv.imwrite("/Users/mehradhq/Computer_Vision/part3_q1_gaussian_filter.jpg",blur)
cv.waitKey(0)
```

Out[1]:

48

In [2]:

```
#part3:question1:median kernel. again not that much difference is seen compared to the original image.
import cv2 as cv
img = cv.imread("/Users/mehradhq/Computer_Vision/part3.jpg")
median = cv.medianBlur(img,5)
cv.imshow("median",median)
cv.imwrite("/Users/mehradhq/Computer_Vision/part3_q1_median_filter.jpg",median)
cv.waitKey(0)
```

Out[2]:

48

In [3]:

```
#part3:question1:bilateral filter. not that much difference is seen compared to the original image.
import cv2 as cv
img = cv.imread("/Users/mehradhq/Computer_Vision/part3.jpg")
blur = cv.bilateralFilter(img,9,75,75)
cv.imshow("bilateralfilter",blur)
cv.imwrite("/Users/mehradhq/Computer_Vision/part3_q1_bilateral_filter.jpg",blur)
cv.waitKey(0)
```

Out[3]:

48

In [1]:

```
#part3:question2: adding s&p noise
"""
the noise covered in the below codes:
   's&p'       Replaces random pixels with 0 or 1.
"""

import numpy as np
import random as rnd
import cv2 as cv

def sp_noise(image,prob):
   '''
   Add salt and pepper noise to image
   prob: Probability of the noise
   '''
   output = np.zeros(image.shape,np.uint8)
   thres = 1 - prob
   for i in range(image.shape[0]):
      for j in range(image.shape[1]):
         rdn = rnd.random()
         if rdn < prob:
            output[i][j] = 0
         elif rdn > thres:
            output[i][j] = 255
         else:
            output[i][j] = image[i][j]
   return output

image = cv.imread('/Users/mehradhq/Computer_Vision/part3.jpg',0) # Only for grayscale image
noise_img = sp_noise(image,0.05)
cv.imwrite('sp_noise_part3_q2.jpg', noise_img)
```

Out[1]:

True

In [2]:

```
#part3:question2: applying 3by3 median filter. it roughly gives the original image back.
import cv2 as cv
img = cv.imread("/Users/mehradhq/Computer_Vision/sp_noise_part3_q2.jpg")
median = cv.medianBlur(img,3)
cv.imshow("median",median)
cv.imwrite("/Users/mehradhq/Computer_Vision/part3_q2_median_filter_s&p_noise.jpg",median)
cv.waitKey(0)
```

Out[2]:

48

In [1]:

```
#part3:question2: applying 5by5 median filter. it roughly gives the original image back.
import cv2 as cv
img = cv.imread("/Users/mehradhq/Computer_Vision/sp_noise_part3_q2.jpg")
median = cv.medianBlur(img,5)
cv.imshow("median",median)
cv.imwrite("/Users/mehradhq/Computer_Vision/part3_q2_median_filter(5by5)_s&p_noise.jpg",median)
cv.waitKey(0)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/var/folders/jr/fkn85rwj2l78jxbdpgdz7j840000gn/T/ipykernel_3705/2914666368.py in <module>
      5 cv.imshow("median",median)
      6 cv.imwrite("/Users/mehradhq/Computer_Vision/part3_q2_median_filter(5by5)_s&p_noise.jpg",me
dian)
----> 7 cv.waitKey(0)

KeyboardInterrupt:
```

In [1]:

```
#part3:question2: adding s&p 2by2 grid noise
"""
the noise covered in the below codes:
   's&p'       Replaces random pixels with 0 or 1.
"""

import numpy as np
import random as rnd
import cv2 as cv

def sp_noise(image,prob):
   '''
   Add salt and pepper noise to image
   prob: Probability of the noise
   '''
   output = np.zeros(image.shape,np.uint8)
   thres = 1 - prob
   for i in range(1,image.shape[0],2):
      for j in range(1,image.shape[1],2):
         rdn = rnd.random()
         if rdn < prob:
            output[i][j] = 0
            output[i-1][j-1] = 0
            output[i][j-1] = 0
            output[i-1][j] = 0
         elif rdn > thres:
            output[i][j] = 255
            output[i-1][j-1] = 255
            output[i][j-1] = 255
            output[i-1][j] = 255
         else:
            output[i][j] = image[i][j]
            output[i-1][j-1] = image[i-1][j-1]
            output[i-1][j] = image[i-1][j]
            output[i][j-1] = image[i][j-1]
   return output

image = cv.imread('/Users/mehradhq/Computer_Vision/part3.jpg',0) # Only for grayscale image
noise_img = sp_noise(image,0.05)
cv.imwrite('sp_noise_part3_q2_2by2_gridNoise.jpg', noise_img)
```

Out[1]:

```
True
```

In [5]:

```
#part3:question2: applying 3by3 median filter. inthis version even applying the 3by3 median filter does not give our initial im
import cv2 as cv
img = cv.imread("/Users/mehradhq/Computer_Vision/sp_noise_part3_q2_2by2_gridNoise.jpg")
median = cv.medianBlur(img,3)
cv.imshow("median",median)
cv.imwrite("/Users/mehradhq/Computer_Vision/part3_q2_median_filter_s&p_2by2grid_noise.jpg",median)
cv.waitKey(0)
```

Out[5]:

```
48
```

In [1]:

```
#part3:question2: applying 5by5 median filter. inthis version even applying the 5by5 median filter does not give our initial im
#applyig the 5by5 median filter in the previous version. However, the 5by5 median filter operates better than the 3by3 median
import cv2 as cv
img = cv.imread("/Users/mehradhq/Computer_Vision/sp_noise_part3_q2_2by2_gridNoise.jpg")
median = cv.medianBlur(img,5)
cv.imshow("median",median)
cv.imwrite("/Users/mehradhq/Computer_Vision/part3_q2_median_filter5by5_s&p_2by2grid_noise.jpg",median)
cv.waitKey(0)
```

Out[1]:

48

In [21]:

```
#part3:q3:sharping a greyscale image
import numpy as np
import matplotlib.pylab as plt
from PIL import Image, ImageFilter
from skimage.io import imread
from skimage.filters import unsharp_mask
# with scikit-image
im = imread("/Users/mehradhq/Computer_Vision/Project_1/Image_project.jpeg")
im1 = np.copy(im).astype(np.float)
for i in range(3):
    im1[...,i] = unsharp_mask(im[...,i], radius=2, amount=2)
# with PIL
im = Image.open("/Users/mehradhq/Computer_Vision/Project_1/Image_project.jpeg")
im2 = im.filter(ImageFilter.UnsharpMask(radius=2, percent=150))
# plot
plt.figure(figsize=(20,7))
plt.subplot(131), plt.imshow(im), plt.axis('off'), plt.title('Original', size=20)
plt.subplot(132), plt.imshow(im1), plt.axis('off'), plt.title('Sharpened (skimage)', size=20)
plt.subplot(133), plt.imshow(im2), plt.axis('off'), plt.title('Sharpened (PIL)', size=20)
plt.show()
```

```
/var/folders/jr/fkn85rwj2l78jxbdpgdz7j840000gn/T/ipykernel_3769/420902829.py:8: DeprecationWarnin
g: `np.float` is a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing
this will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.fl
oat64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-note
s.html#deprecations (https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations)
  im1 = np.copy(im).astype(np.float)
```