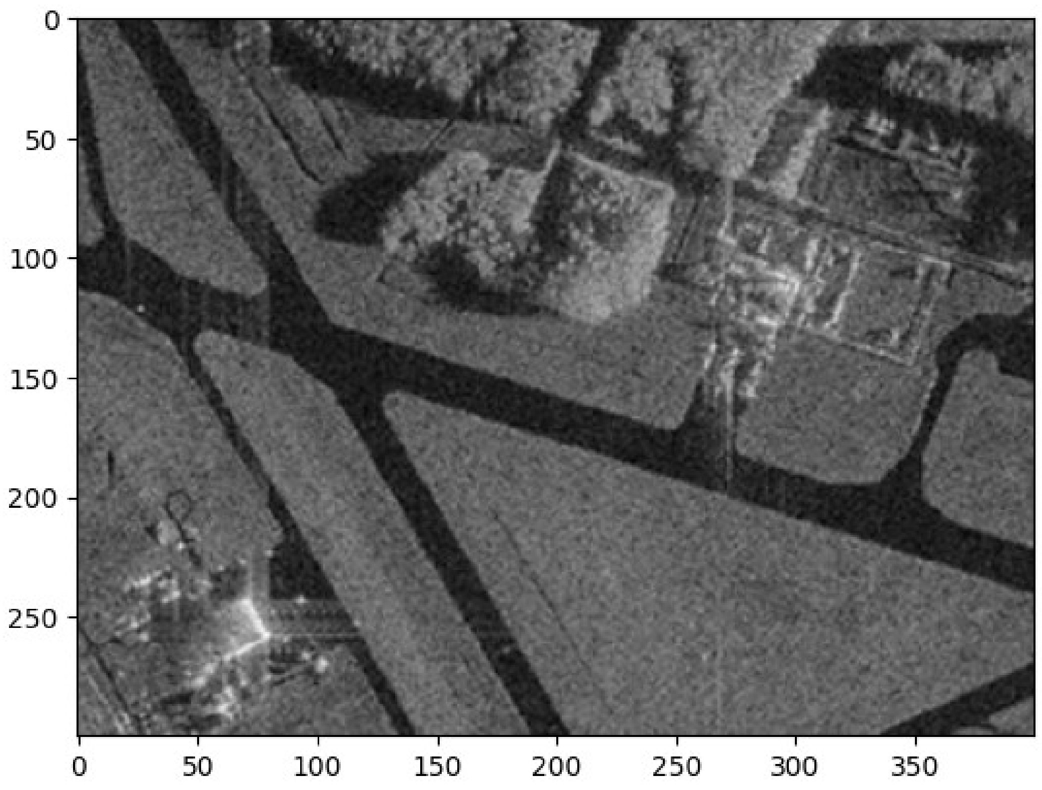
```
import numpy as np
import cv2
import matplotlib.pyplot as plt

image = cv2.imread('sar_1.jpg')
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

plt.imshow(image_gray, cmap="gray")
```

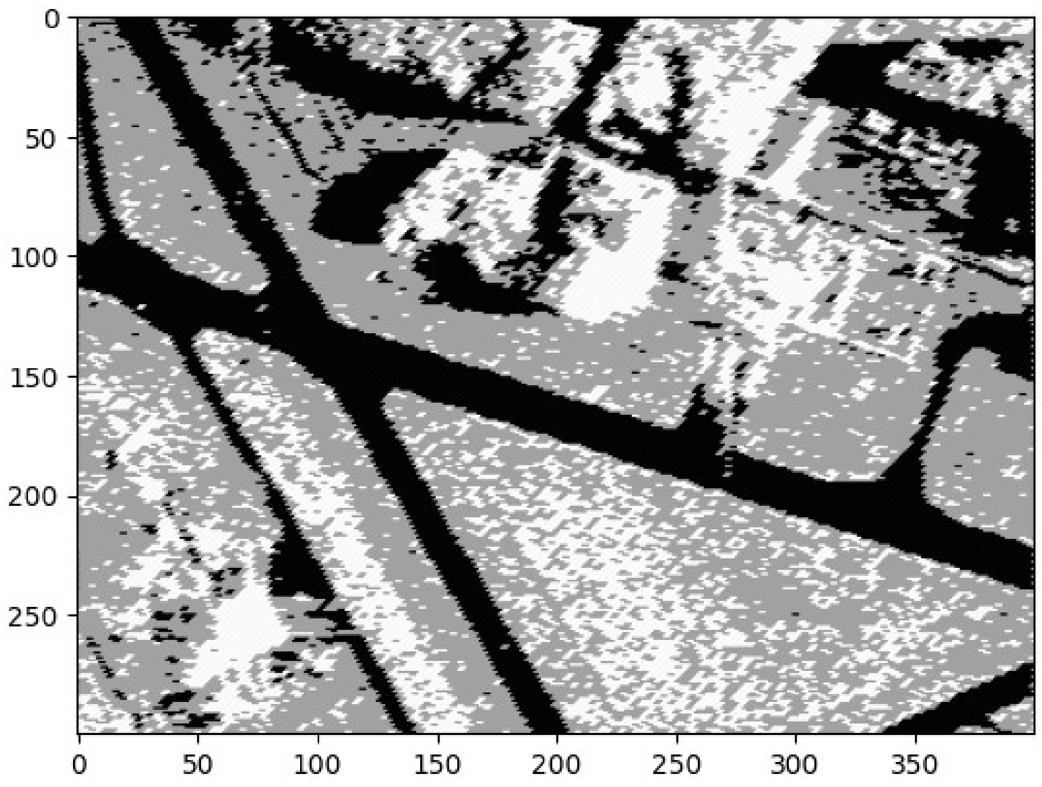
```
<matplotlib.image.AxesImage at 0x7cac7db2f490>
```



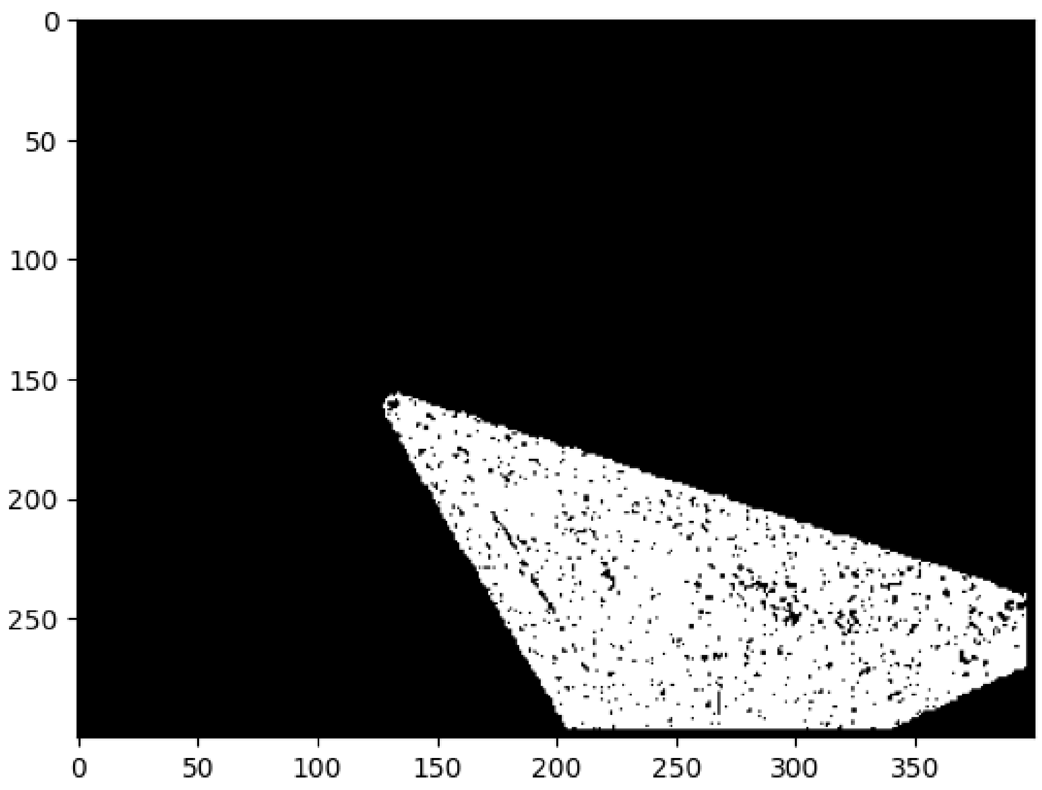
K-means

```
# Define criteria = ( type, max_iter = 10 , epsilon = 1.0 )
# criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 10,
1.0)

flags = cv2.KMEANS_RANDOM_CENTERS

z = image_gray.reshape((-1,3))
# convert to np.float32
z = np.float32(z)
# define criteria, number of clusters(K) and apply kmeans()
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10,
1.0)
K = 3
ret,label,center=cv2.kmeans(z,K,None,criteria,10,cv2.KMEANS_RANDOM_CEN
TERS)
# Now convert back into uint8, and make original image
center = np.uint8(center)
res = center[label.flatten()]
res2 = res.reshape((image_gray.shape))

plt.imshow(res2, cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7cac7dc19d20>
```

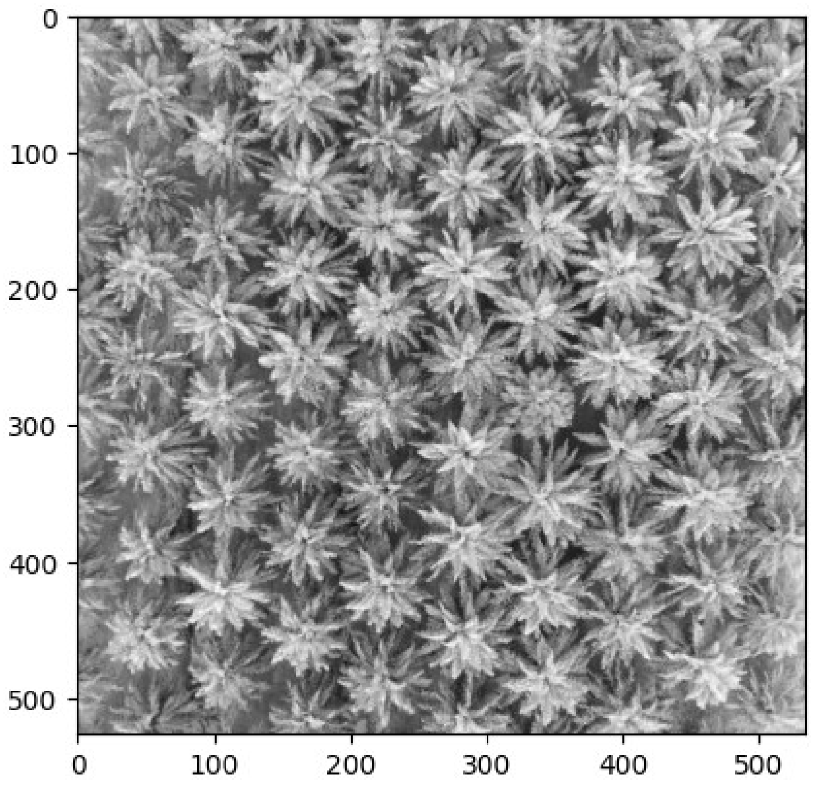


```
# 1. Подберите парамтеры алгоритма разрастания регионов так, чтобы был
выделен весь участок газона.
import numpy as np
import cv2
import matplotlib.pyplot as plt

def homo_average(img, mask, point, T):
    av_val = img[mask > 0].sum() / np.count_nonzero(img[mask > 0])

    if abs(av_val - img[point]) <= T:
        return True
    return False

def region_growing(image, seed_point, homo_fun, r, T):
    mask = np.zeros(image.shape, np.uint8)
    mask[seed_point] = 1
    count = 1
    while count > 0:
        count = 0
        local_mask = mask.copy()
        for i in range(r, image.shape[0] - r):
            for j in range(r, image.shape[1] - r):
                if mask[i, j] == 0 and mask[i - r:i + r, j - r:j +
r].sum() > 0:
                    if homo_fun(image, mask, (i, j), T):
                        local_mask[i, j] = 1
                        count += 1
        mask = np.logical_or(mask, local_mask)

    return mask * 255

image = cv2.imread('sar_1.jpg', cv2.IMREAD_GRAYSCALE)
seed_point = (250, 250)
mask = region_growing(image, seed_point, homo_average, 3, 15)

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

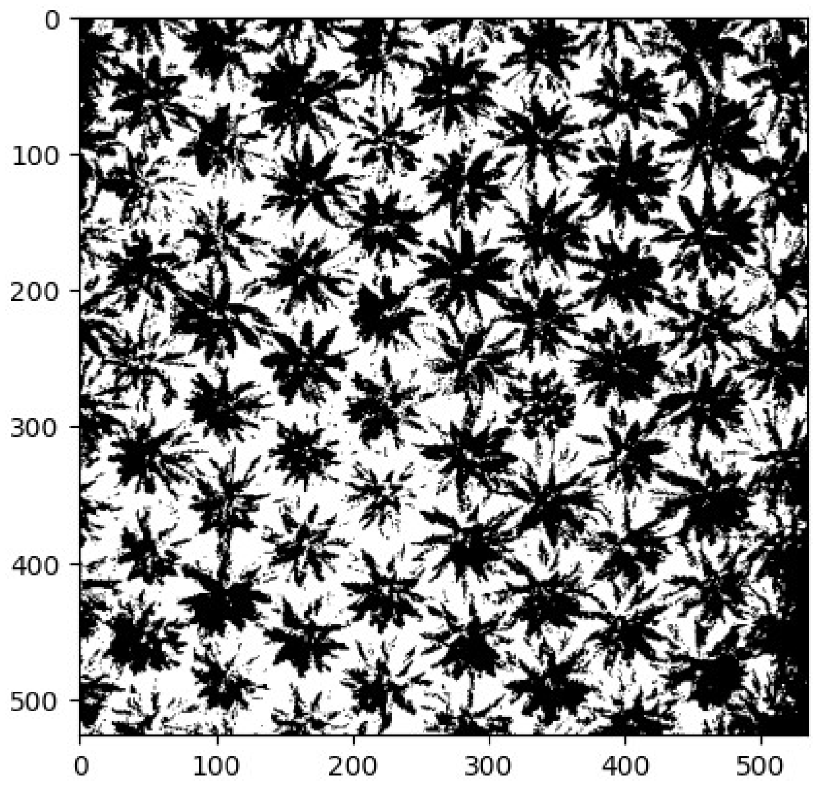


```
# 2. Реализуйте вычисление критерия однородности, отличного от
представленного.
import math
def homo_average(img, mask, point, T):
    av_val = img[mask > 0].sum() / np.count_nonzero(img[mask > 0])

    if abs(av_val - img[point]) <= T:
        return True

    return False

# 3. Применить алгоритм сегментации watershed+distance transform для
задачи подсчета пальмовых деревьев.
image = cv2.imread('palm_1.JPG')
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
plt.imshow(image_gray, cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7cac7dddd9f0>
```

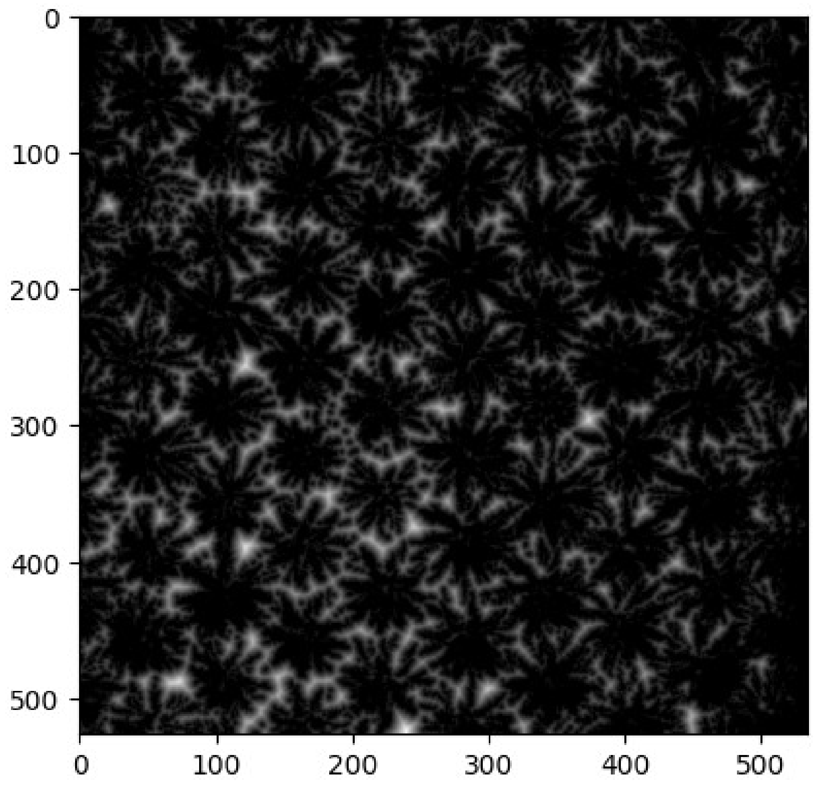


```
ret, thresh = cv2.threshold(image_gray,0,255,
cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
plt.imshow(thresh, cmap="gray")

<matplotlib.image.AxesImage at 0x7cac7dd4df60>
```

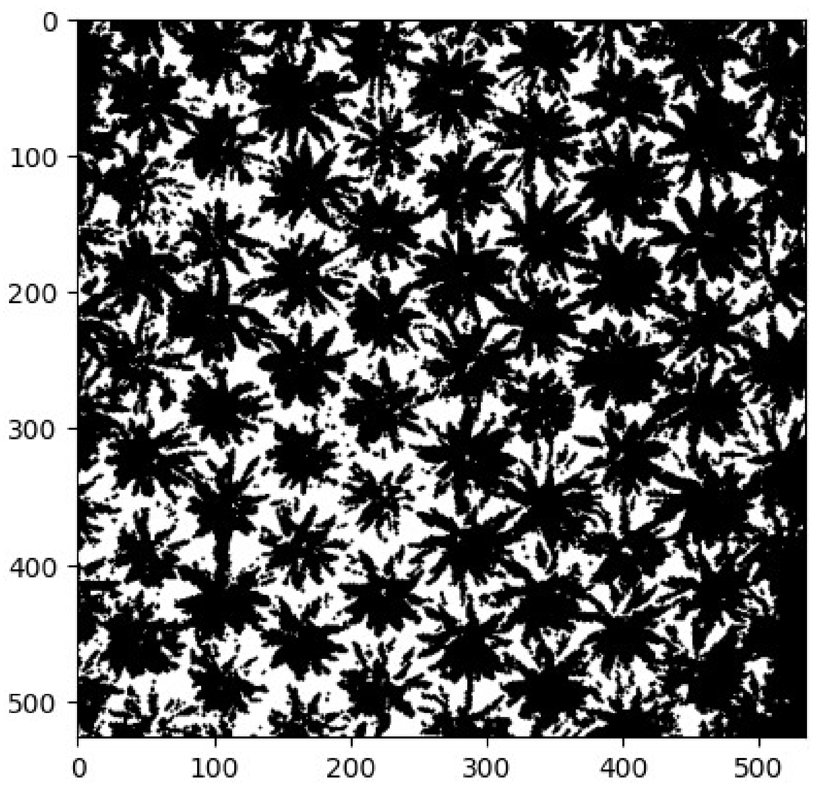


```
dist = cv2.distanceTransform(thresh, cv2.DIST_L2, 5)
plt.imshow(dist, cmap="gray")
```

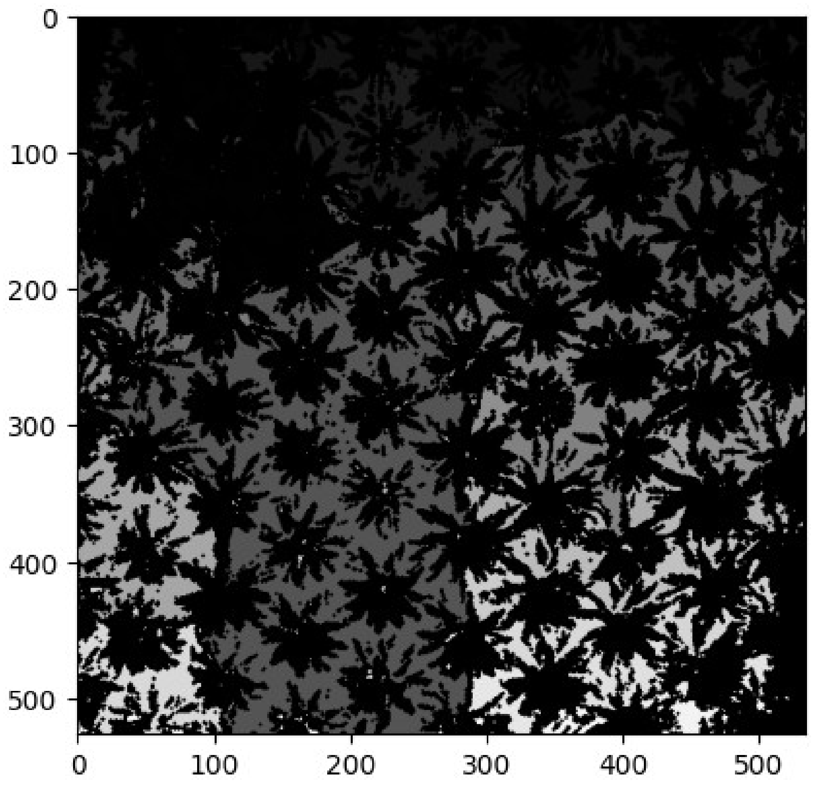
```
<matplotlib.image.AxesImage at 0x7cac7d428400>
```



```
ret, sure_fg = cv2.threshold(dist, 0.1 * dist.max(), 255,
cv2.THRESH_BINARY)
plt.imshow(sure_fg, cmap="gray")

<matplotlib.image.AxesImage at 0x7cac7d49d210>
```

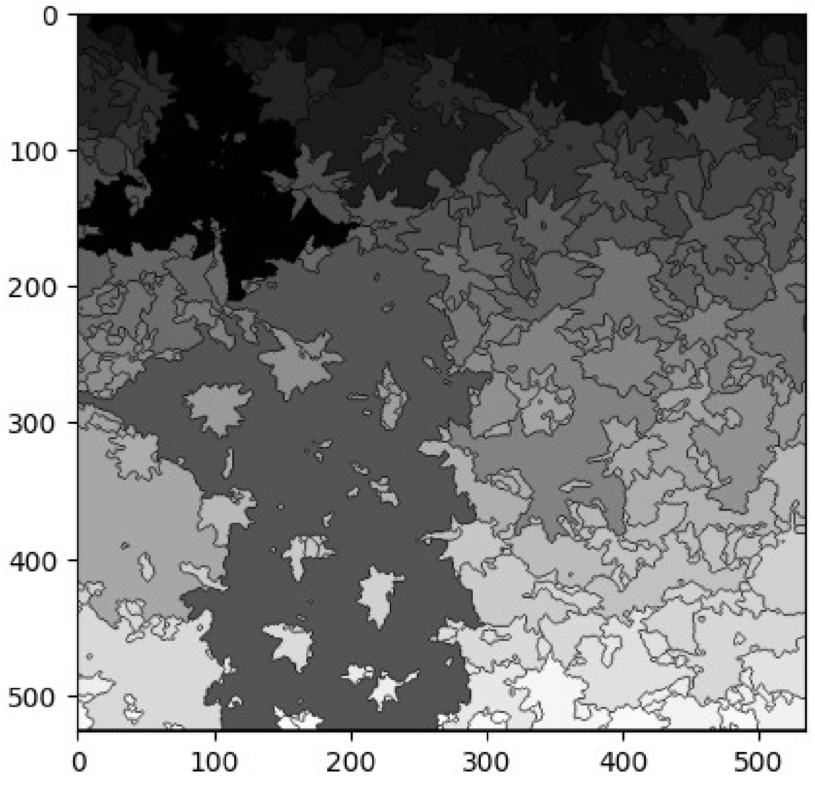


```
sure_fg = sure_fg.astype(np.uint8)
ret, markers = cv2.connectedComponents(sure_fg)
plt.imshow(markers, cmap="gray")

<matplotlib.image.AxesImage at 0x7cac7d4fe560>
```



```
markers = cv2.watershed(image, markers)
plt.imshow(markers, cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7cac7d372ad0>
```


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