

In [120]:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from mylib import *
```

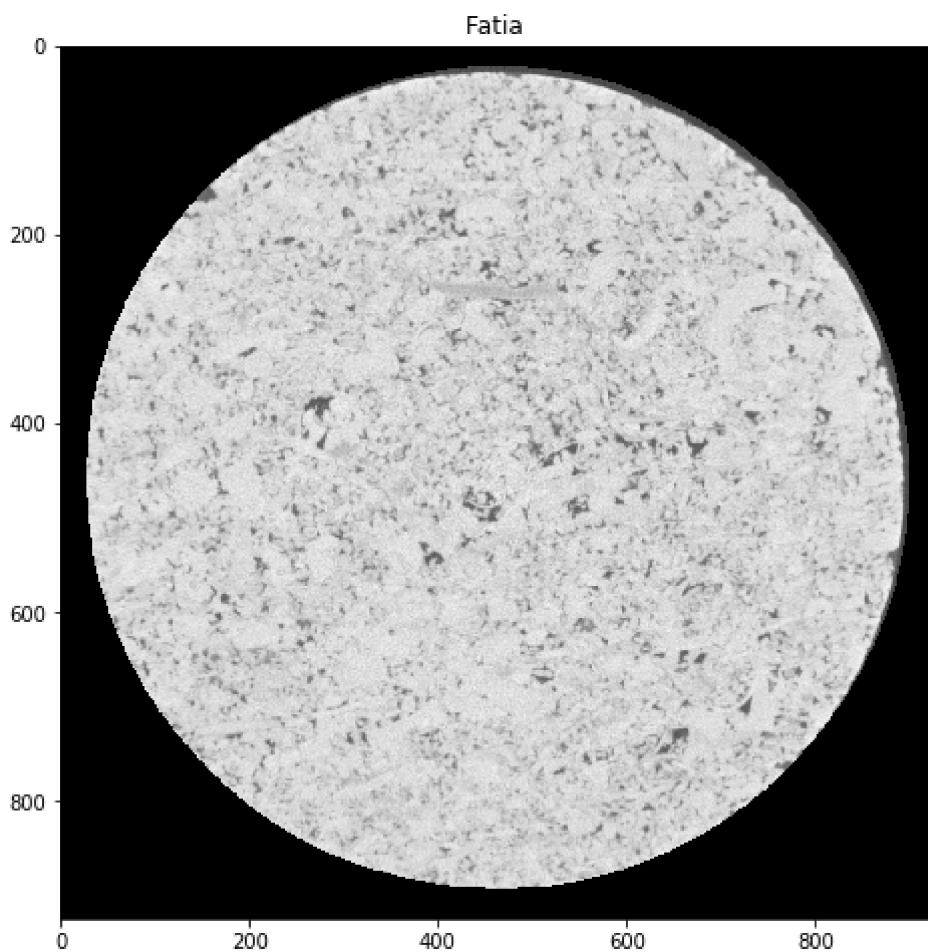
In [121]:

```
1 fatia = np.load("secao_do_plug.npy")
2 show_npy(fatia)
```

shape=(925, 920), type=uint16  
min= 0, max=10184

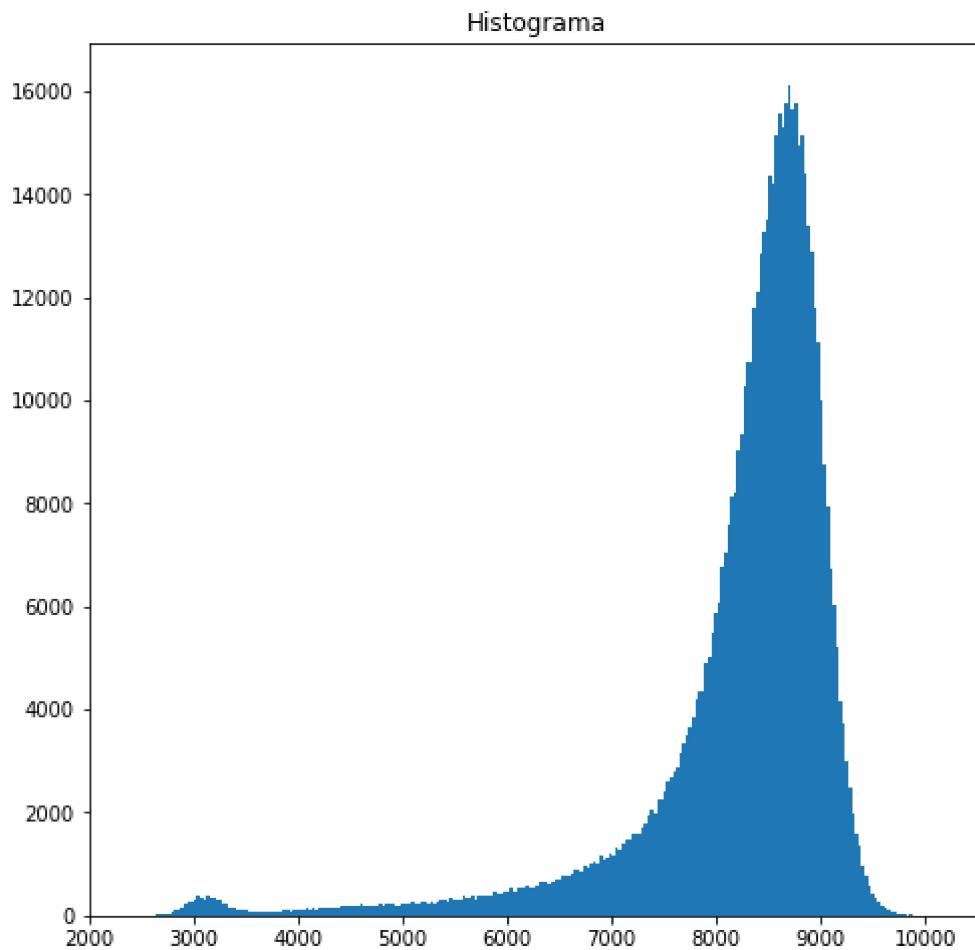
In [122]:

```
1 show_gray(fatia, "Fatia")
```



In [123]:

```
1 data = fatia[fatia>0]
2 show_hist(data.ravel(),"Histograma")
```



## 0.1 Algoritmo de KMeans

1. Determine a posição de k representantes
2. Agrupe todos os pontos em torno dos seus representantes
3. Recalcule a posição dos representantes
4. Repita passos 1 e 2 até convergir

In [124]:

```

1  def KMeans(k, dados, erro_max, max_iteracoes):
2      # inicializacao
3      n = dados.size
4      valor_min = np.amin(dados)
5      valor_max = np.amax(dados)
6      delta = (valor_max - valor_min) / k
7      c0 = valor_min + 0.5 * delta
8      centros = np.array([int(c0 + i * delta) for i in range(k)])
9      grupos = np.zeros(n, dtype=np.int)
10     distancias = np.zeros(k, dtype=np.int)
11     novos_centros = np.zeros(k, dtype=np.int)
12     erro = erro_max + 1
13     iteracao = 1
14     while (iteracao <= max_iteracoes and erro > erro_max):
15         # reatribui as amostras para os grupos
16         for i, v in enumerate(dados):
17             distancias = np.abs(centros - v)
18             grupos[i] = np.argmin(distancias)
19         # recalcula a posicao dos centros
20         for j in range(k):
21             novos_centros[j] = np.mean(dados[grupos == j]).astype(int)
22             erro = np.amax(np.abs(centros - novos_centros))
23             centros = novos_centros.copy()
24             print(f'iteracao={iteracao}, erro={erro}, centros={centros}')
25             iteracao += 1
26     return centros, erro, grupos

```

In [125]:

```

1  k = 4
2  erro_max = 10
3  max_iteracoes = 10
4  c, e, g = KMeans(k, data, erro_max, max_iteracoes)

```

```

iteracao=1, erro=495, centros=[3406 5521 7684 8713]
iteracao=2, erro=287, centros=[3485 5808 7709 8701]
iteracao=3, erro=180, centros=[3601 5988 7747 8703]
iteracao=4, erro=128, centros=[3694 6116 7787 8710]
iteracao=5, erro=99, centros=[3771 6215 7826 8719]
iteracao=6, erro=76, centros=[3826 6291 7861 8728]
iteracao=7, erro=62, centros=[3872 6353 7891 8736]
iteracao=8, erro=51, centros=[3910 6404 7917 8744]
iteracao=9, erro=43, centros=[3941 6447 7940 8751]
iteracao=10, erro=37, centros=[3968 6484 7959 8757]

```

In [126]:

```

1  def show_image(img, title):
2      plt.figure(figsize=(8,8))
3      plt.imshow(img, cmap='gray')
4      plt.title(title)
5      plt.show()

```

In [127]:

```
1 def pinta_imagem(rgb, img):
2     return np.dstack((img * rgb[0], img * rgb[1], img * rgb[2])).astype(np.uint8)
```

In [128]:

```
1 def get_values(k, g, img):
2     values = []
3     for i in range(k):
4         values.append(np.zeros(img.shape))
5     k = 0
6     for i in range(img.shape[0]):
7         for j in range(img.shape[1]):
8             if img[i, j] != 0:
9                 values[g[k]][i, j] = 1
10            k += 1
11    return values
```

In [129]:

```
1 vermelho = [255, 0, 0]
2 laranja = [255, 165, 0]
3 azul = [0, 255, 0]
4 verde = [0, 0, 255]
5 branco = [255, 255, 255]
```

In [130]:

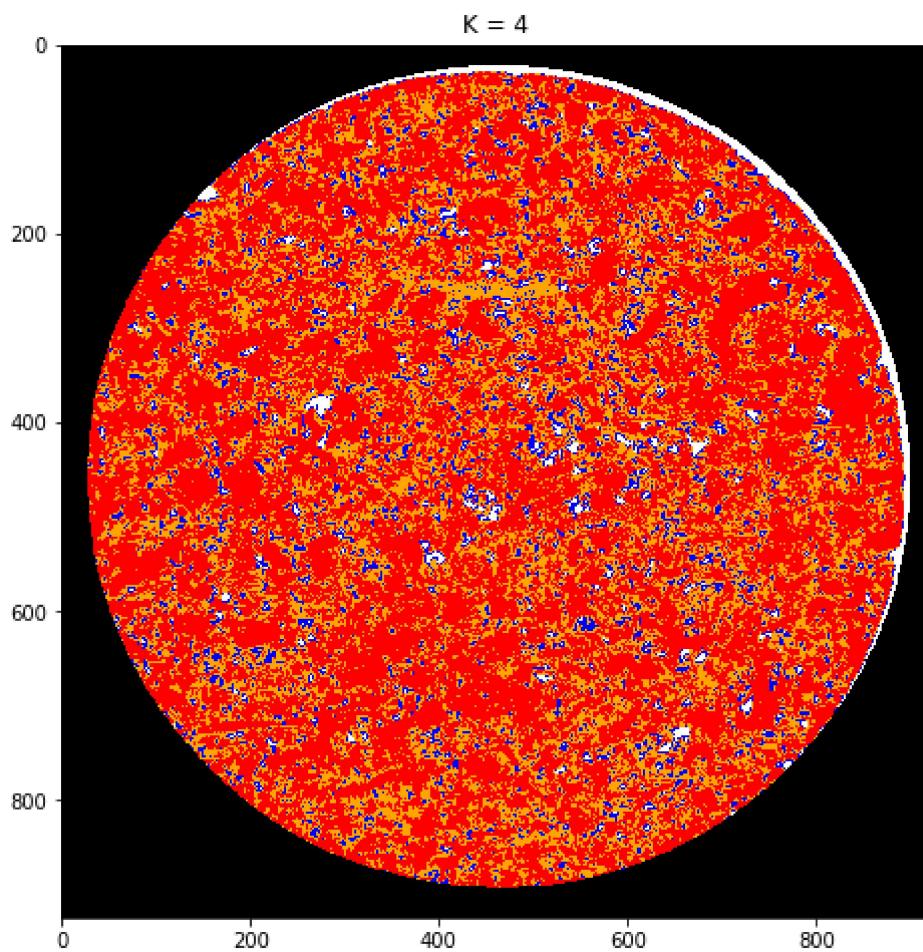
```
1 values = get_values(k, g, fatia)
```

In [131]:

```
1 imagem = pinta_imagem(branco, values[0]) + pinta_imagem(
2     verde, values[1]) + pinta_imagem(laranja, values[2]) + pinta_imagem(
3     vermelho, values[3])
```

In [132]:

```
1 show_image(imagem, "K = 4")
```



In [133]:

```
1 k = 3
2 c, e, g = KMeans(k, data, erro_max, max_iteracoes)
```

```
iteracao=1, erro=510, centros=[3818 6792 8551]
iteracao=2, erro=228, centros=[4046 6940 8567]
iteracao=3, erro=150, centros=[4196 7059 8582]
iteracao=4, erro=110, centros=[4306 7152 8597]
iteracao=5, erro=85, centros=[4391 7225 8609]
iteracao=6, erro=66, centros=[4457 7282 8618]
iteracao=7, erro=51, centros=[4508 7326 8627]
iteracao=8, erro=43, centros=[4551 7361 8633]
iteracao=9, erro=36, centros=[4587 7390 8639]
iteracao=10, erro=27, centros=[4614 7412 8643]
```

In [134]:

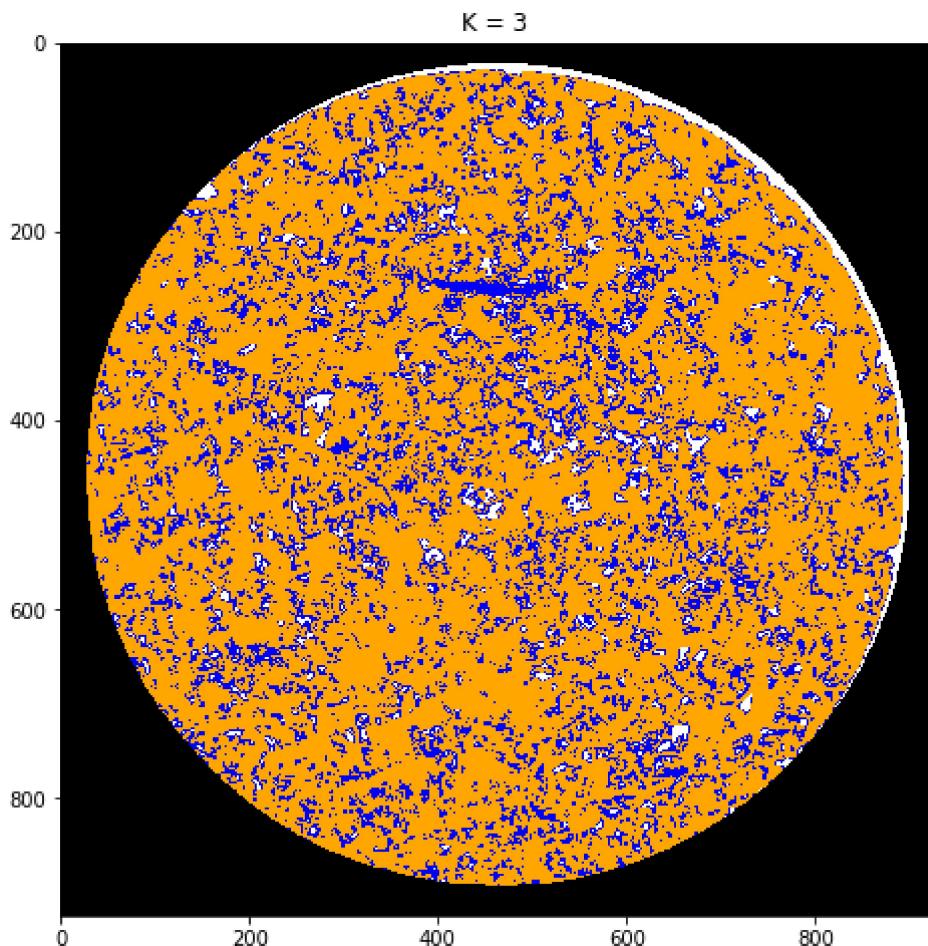
```
1 values = get_values(k, g, fatia)
```

In [135]:

```
1 imagem = pinta_imagem(branco, values[0]) + pinta_imagem(verde, values[1]) + pinta_imagem(azul, values[2])
```

In [136]:

```
1 show_image(imagem, "K = 3")
```



In [137]:

```
1 k = 5
2 c, e, g = KMeans(k, data, erro_max, max_iteracoes)
```

```
iteracao=1, erro=509, centros=[3222 4845 6462 8156 8894]
iteracao=2, erro=235, centros=[3258 4979 6697 8124 8841]
iteracao=3, erro=152, centros=[3297 5131 6828 8112 8819]
iteracao=4, erro=129, centros=[3343 5260 6915 8113 8811]
iteracao=5, erro=109, centros=[3391 5369 6981 8121 8810]
iteracao=6, erro=91, centros=[3434 5460 7037 8132 8811]
iteracao=7, erro=73, centros=[3473 5533 7084 8145 8814]
iteracao=8, erro=63, centros=[3509 5596 7125 8157 8818]
iteracao=9, erro=55, centros=[3539 5651 7162 8170 8822]
iteracao=10, erro=48, centros=[3567 5699 7195 8183 8826]
```

In [138]:

```
1 values = get_values(k, g, fatia)
```

In [139]:

```
1 imagem = pinta_imagem(branco, values[0]) + pinta_imagem(
2 verde, values[1]) + pinta_imagem(laranja, values[2]) + pinta_imagem(
3 vermelho, values[3]) + pinta_imagem(verde, values[4])
```

In [140]:

```
1 show_image(imagem, "K = 5")
2 # Com 5 clusters, existem grupos com muito poucos pontos. Quase não é possível enxergar
3 # A opção com 4 clusters parece ser a melhor.
```

