In [1]:

```
import numpy as np
from PIL import Image
from scipy import ndimage
from matplotlib import pyplot as plt
```

In [2]:

```
# Set directory where you store the data

data_dir = '/Users/daiyun/Documents/Chromosome/abnormal/'
```

In [3]:

```
# Now we load one image as sample to show how we can extract chromosomes in Python
# Image loaded by PIL.Image may not be writable, hence here we create a copy of our int
erested image

sample_img = Image.open(data_dir + '101D.tif')
sample_img = np.asarray(sample_img)
img = sample_img.copy()
img.flags
```

Out[3]:

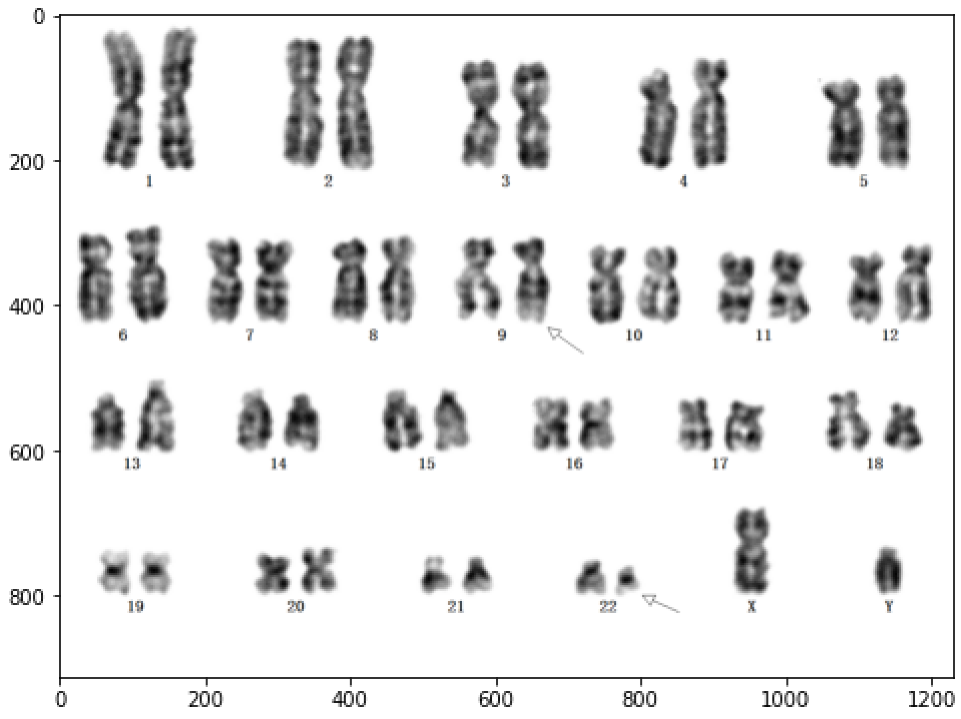
```
  C_CONTIGUOUS : True
  F_CONTIGUOUS : False
  OWNDATA : True
  WRITEABLE : True
  ALIGNED : True
  WRITEBACKIFCOPY : False
  UPDATEIFCOPY : False
```

In [4]:

```
# Plot original image

plt.figure(figsize=(8, 8))
plt.imshow(img)
```

Out[4]:

```
<matplotlib.image.AxesImage at 0x104414d90>
```

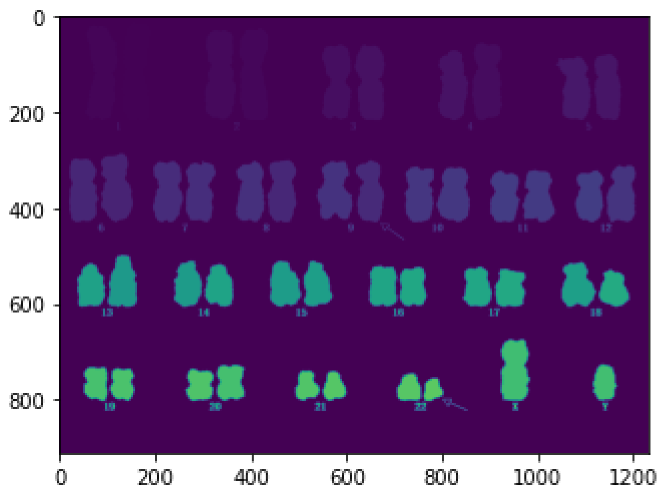


In [5]:

```
# Although the original image has 3 channels (RGB), we shall find all chromosomes are g
ray-scaled
# that means R,G,B values of those chromosomes are same, hence we only need to extract
 one channel from original image
# We directly apply ndimage.label from scipy package to extract objects from one image

gray_img = img[:, :, 0]
labeled, nb_objects = ndimage.label(gray_img < 255)
plt.imshow(labeled)
```

Out[5]:

```
<matplotlib.image.AxesImage at 0x1057bde50>
```

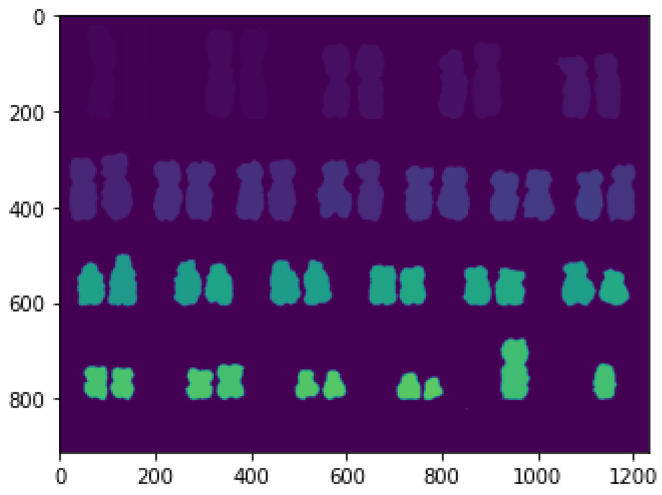


In [6]:

```python
# We can see from above that those numbers and arrows were also recognized as objects
# Fortunately, those unwanted items are relatively small compared to chromosomes,
# Hence, here we set a threshold (of areas) to filter those small objects

chr_img = labeled.copy()
threshold = 500

for i in range(nb_objects):
    if len(np.where(chr_img == i)[0]) < threshold:
        chr_img[chr_img == i] = 0

plt.imshow(chr_img)
```

Out[6]:

```
<matplotlib.image.AxesImage at 0x11620e350>
```

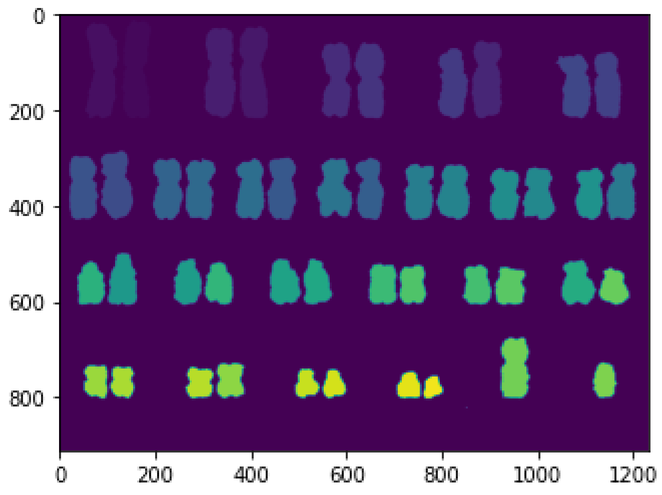


In [7]:

```
# Let us see what happend after we removed those items whose areas are less than 500

new_labeled, new_nb_objects = ndimage.label(chr_img)
plt.imshow(new_labeled)
```

Out[7]:

```
<matplotlib.image.AxesImage at 0x113bfc750>
```

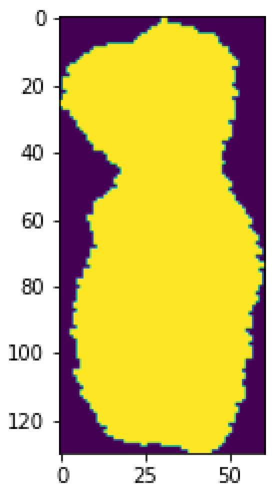


In [8]:

```
# Now we successfully extracted those chromosomes, by looping over all objects we can f
ind the chromosome we want
# Here is an example of chromosome 9 and how we get its bounding box

sample_chr9 = new_labeled == 14
bbox = ndimage.find_objects(sample_chr9)
slice1 = bbox[0][0]
slice2 = bbox[0][1]
plt.imshow(sample_chr9[slice1, slice2])
```

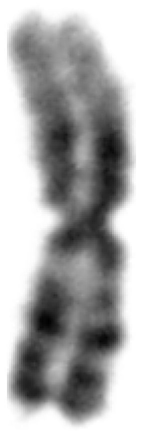
Out[8]:

```
<matplotlib.image.AxesImage at 0x113c659d0>
```



In [9]:

```
# Above example may not work for all image, you need to check all of your data carefull
y
# In case, above procedure do not work on some images,
# we show interested observation you may use to discriminate unwanted objects from chro
mosomes

plt.imshow(img[15:220, 60:120])
plt.axis('off')
plt.show()
```



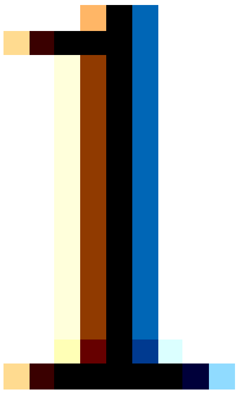
In [10]:

```
print(np.any(img[15:220, 60:120, 0] == 0),
      np.any(img[15:220, 60:120, 1] == 0),
      np.any(img[15:220, 60:120, 2] == 0))
```

```
False False False
```

In [11]:

```
# You may see, unwanted number contains pixels that have different R, G, B values
# But you shall also be careful about those pure black pixels (R, G, B) == (255, 255, 2
55)
# Since some chromosomes may also contain such pixels

plt.imshow(img[221:238, 119:130])
plt.axis('off')
plt.show()
```



In [ ]:

```

```