assignment06

May 9, 2019

In [1]:
```
import PIL.Image as pilimg
import numpy as np
import matplotlib.pyplot as plt
import random
import copy

#read my image(pixel)
#image = pilimg.open("C:\\Users\\recognize_data\\image.jpg")
#image = pilimg.open("C:\\Users\\recognize_data\\color.png")

#image pixel data to array
#315*420
#image_pixel[0][0] = [74 112 175]
#image_pixel = np.array(image)
#image size
#size_col = image.size[0]    # width of the image 420
#size_row = image.size[1]    # height of the image 315
#image_size = size_col * size_row

size_col = 20
size_row = 20
#time
t = 0
```

1 init x, y matrix

In [2]:
```
tempValue = 0;
x_pos =  np.empty((size_row, size_col), dtype = int)
y_pos = np.empty((size_row, size_col), dtype = int)
for i in range(0,size_row):
    for j in range(0,size_col):
        x_pos[i][j] = tempValue
    tempValue += 1

tempValue = 0;
for i in range(0,size_col):
```

```
            for j in range(0,size_row):
                y_pos[j][i] = tempValue
            tempValue += 1
```

In [3]:
```
class AllProcess:

    def _init_(self, k, image_label, centroid, count, store_distance, output_image, ini

        self.k = k
        self.image_label = image_label

        self.centroid = centroid
        self.count = count
        self.store_distance = store_distance
        self.output_image = output_image
        self.init_label = init_label
        self.track_centroid = track_centroid
        self.track_centroid1 = track_centroid1
    def setdata(self,k):
        self.k = k
        self.centroid = np.zeros((5, k), dtype = int) #x,y,r,g,b
        self.count = np.empty(k, dtype = int)
        self.store_distance = np.empty(k)

        self.image_label = np.empty((size_row, size_col), dtype = int)
        self.init_label = np.empty((size_row, size_col), dtype = int)

        self.output_image = np.empty((size_row, size_col,3), dtype = int)
        #self.track_centroid = [np.zeros((2, k), dtype = int) for depth in range(0)]
        self.track_centroid = [[[0 for col in range(k)] for row in range(2)] for depth
        self.track_centroid1 = [[[0 for col in range(k)] for row in range(2)] for deptl
    def random_labeling(self):
        for i in range(0,size_row):
            for j in range(0,size_col):
                self.image_label[i][j] = random.randrange(0,self.k)

    def update_centroid(self):
        #initialize the centroid
        for i in range(0,self.k):
            self.centroid[0][i] = 0
            self.centroid[1][i] = 0
            self.count[i] = 0

        for m in range(0,self.k):
            for i in range(0,size_row):
                for j in range(0,size_col):
                    if(self.image_label[i][j] == m):
                        self.centroid[0][m] += x_pos[i][j]
```

```
                    self.centroid[1][m] += y_pos[i][j]
                    self.count[m] += 1

        # average center
        for i in range(0,self.k):
            if(self.count[i]==0):
                self.count[i] = 1
            self.centroid[0][i] = self.centroid[0][i]/self.count[i]
            self.centroid[1][i] = self.centroid[1][i]/self.count[i]



       #distancel2
    def distance2(self, x,cx,y,cy):
        d = (x-cx) ** 2 + (y-cy)**2
        s = np.sqrt(d)
        return(s)


    def find_nearest2(self, x, y):
        for i in range(0,self.k):
            self.store_distance[i] = self.distance2(x,self.centroid[0][i],y,self.centr
        r = self.store_distance.argmin()
        return r
    #update label
    def update_label2(self):
        for i in range(0,size_row):
            for j in range(0,size_col):
                self.image_label[i][j] = self.find_nearest2(x_pos[i][j], y_pos[i][j])

    #distancel1
    def distance1(self, x,cx,y,cy):
        d = abs(x-cx) + abs(y-cy)
        return(d)


    def find_nearest1(self, x, y):
        for i in range(0,self.k):
            self.store_distance[i] = self.distance1(x,self.centroid[0][i],y,self.centr
        r = self.store_distance.argmin()
        return r
    #update label
    def update_label1(self):
        for i in range(0,size_row):
            for j in range(0,size_col):
                self.image_label[i][j] = self.find_nearest1(x_pos[i][j], y_pos[i][j])

    def print_color(self):
```

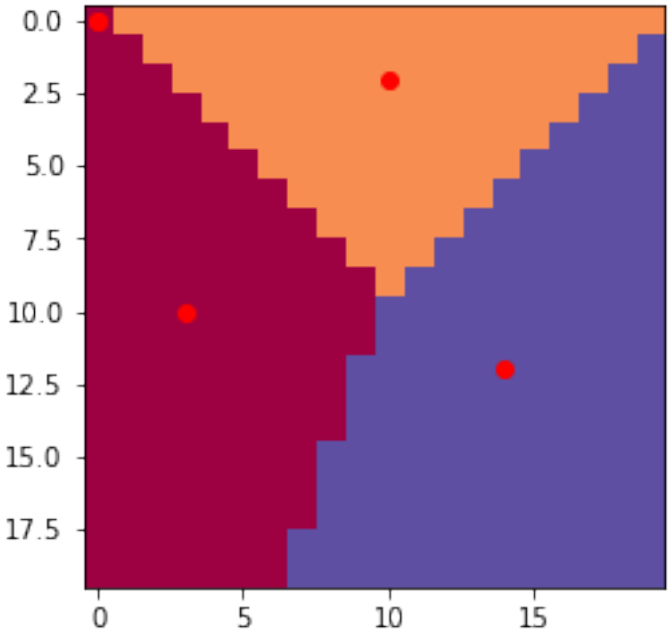
```
                for i in range(0,size_row):
                    for j in range(0,size_col):
                        a = self.image_label[i][j]
                        self.output_image[i][j][0] = self.centroid[a][2] #r
                        self.output_image[i][j][1] = self.centroid[a][3] #g
                        self.output_image[i][j][2] = self.centroid[a][4] #b
                plt.title(t)
                plt.imshow(self.output_image)
```

2 k = 5

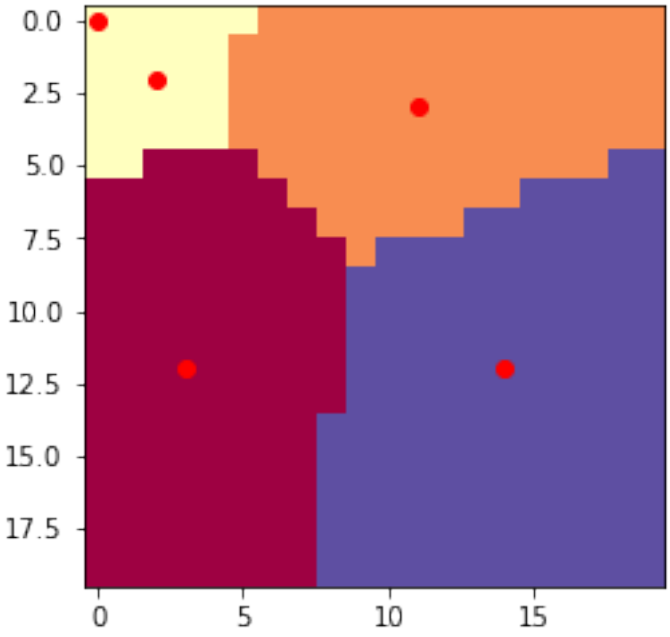
3 L2

```
In [4]: p1 = AllProcess()
        # k = 5
        p1.setdata(5)
        p1.random_labeling()
        p1.init_label = copy.deepcopy(p1.image_label)
        p1.update_centroid()
        a = 0
        while True:

            p1.track_centroid.append([copy.deepcopy(p1.centroid[0]), copy.deepcopy(p1.centroid
            a += 1

            p1.update_label2()
            old_centroid = copy.deepcopy(p1.centroid)
            p1.update_centroid()
            plt.imshow(p1.image_label, cmap=plt.cm.get_cmap('Spectral', p1.k))
            print(p1.centroid[0])
            print(p1.centroid[1])

            plt.plot(p1.centroid[1], p1.centroid[0], 'ro')
            plt.show()
            if np.array_equal(old_centroid, p1.centroid):
                break
```

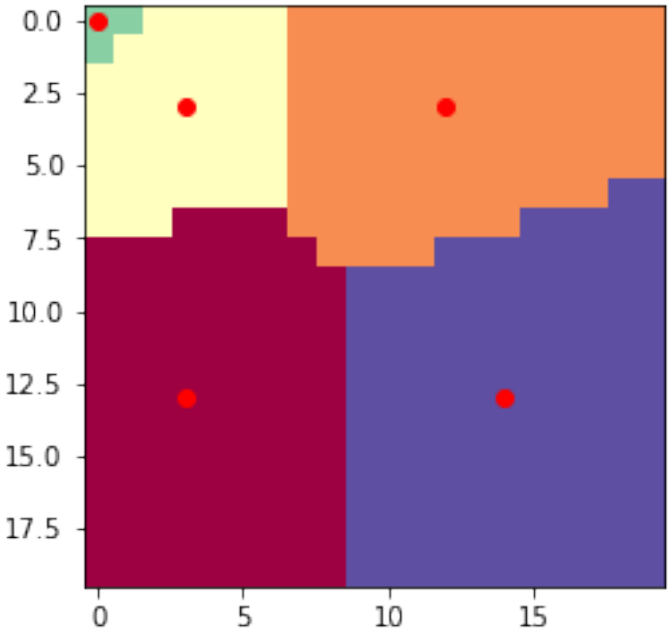
```
[10  2  0  0 12]
[ 3 10  0  0 14]
```



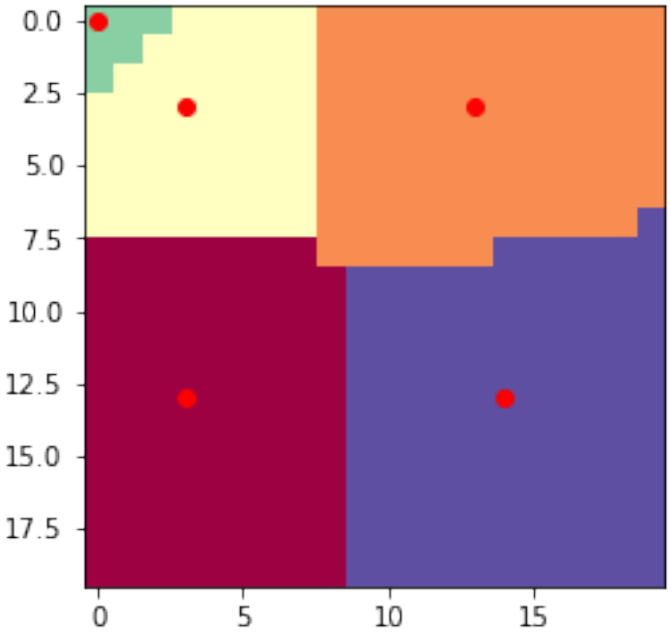

```
[12  3  2  0 12]
[ 3 11  2  0 14]
```



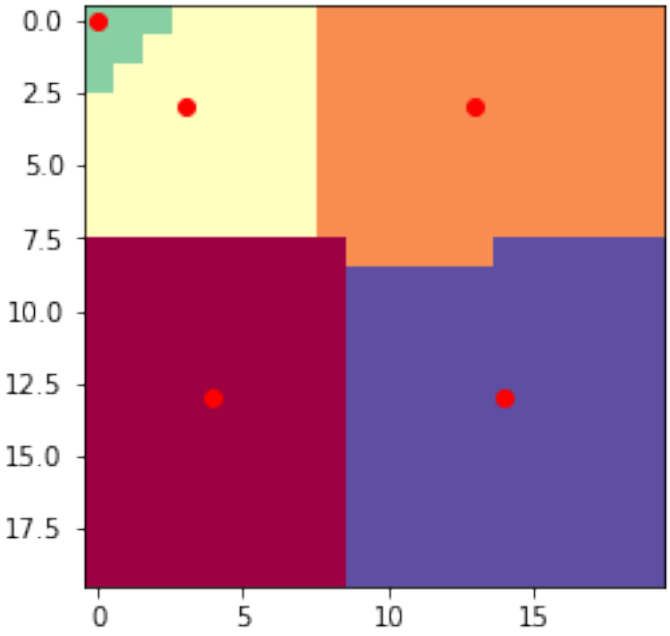
```
[13  3  3  0 13]
[ 3 12  3  0 14]
```



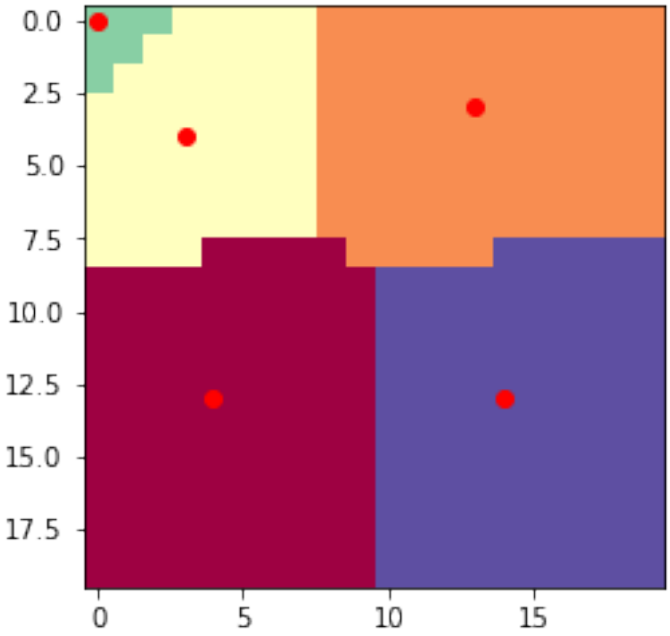

```
[13  3  3  0 13]
[ 3 13  3  0 14]
```



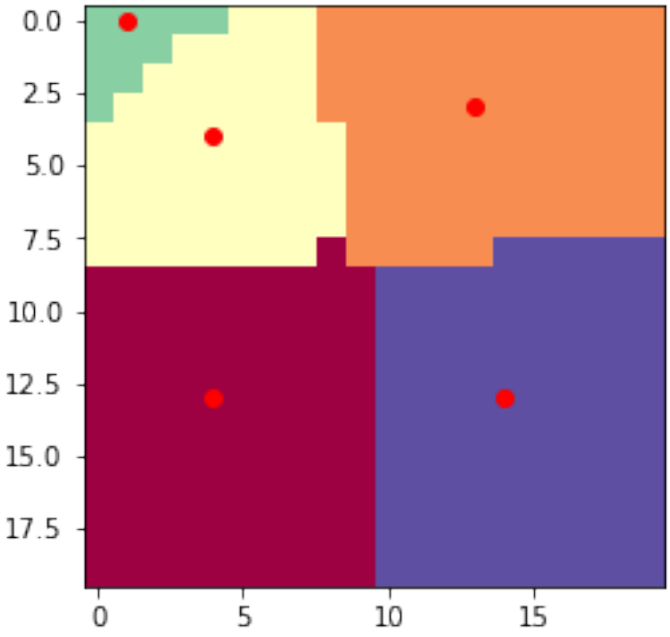
```
[13  3  3  0 13]
[ 4 13  3  0 14]
```



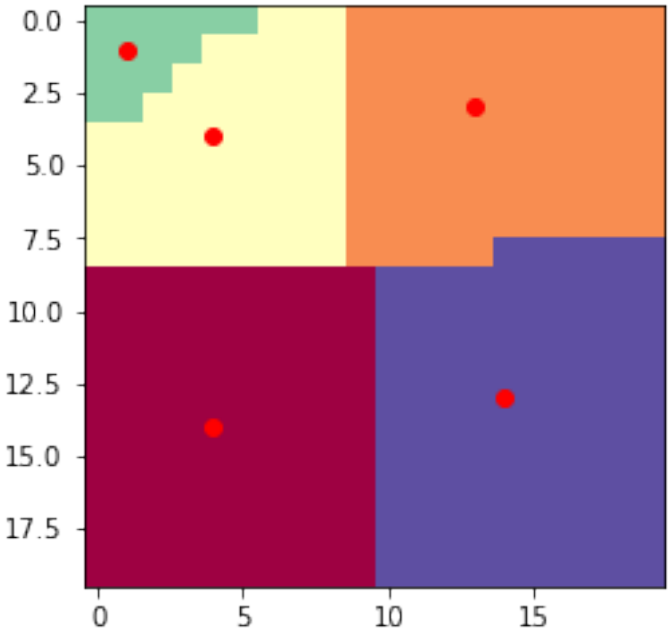

```
[13  3  4  0 13]
[ 4 13  3  0 14]
```



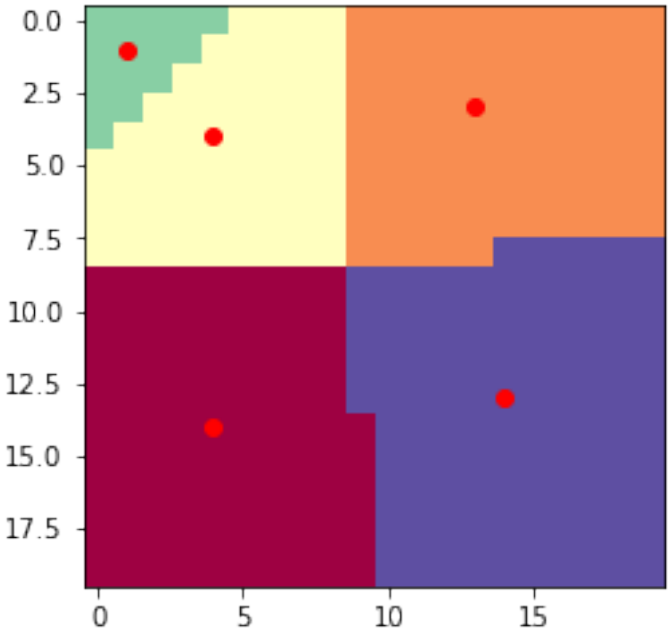

```
[13  3  4  0 13]
[ 4 13  4  1 14]
```



```
[14  3  4  1 13]
[ 4 13  4  1 14]
```



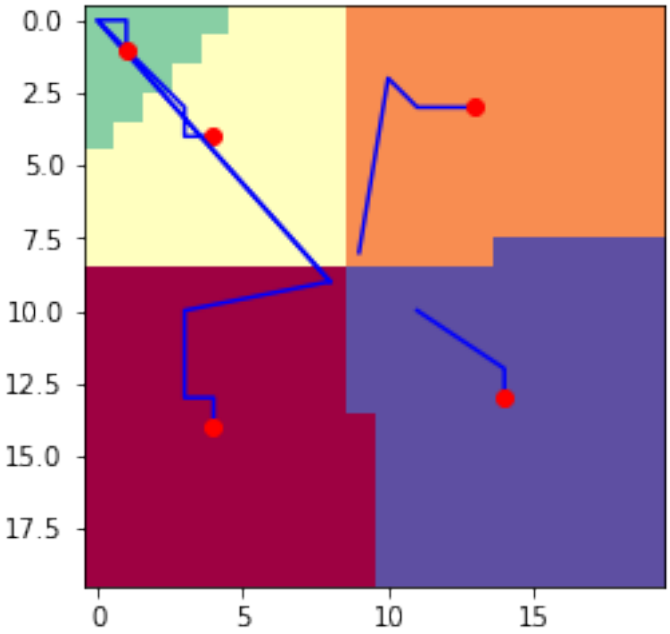

```
[14  3  4  1 13]
[ 4 13  4  1 14]
```

4 trajectory of centroid

endpoint represents red dot.

```
In [5]: one = np.zeros((a,2), dtype = int)

        for j in range(0,p1.k):
            for i in range(0,a):
                one[i] = [p1.track_centroid[i][0][j], p1.track_centroid[i][1][j]]

            plt.plot(one.T[1], one.T[0], 'b-',one.T[1][a-1], one.T[0][a-1], 'ro')

        plt.imshow(p1.image_label, cmap=plt.cm.get_cmap('Spectral', p1.k))

Out[5]: <matplotlib.image.AxesImage at 0x277b5d6e358>
```



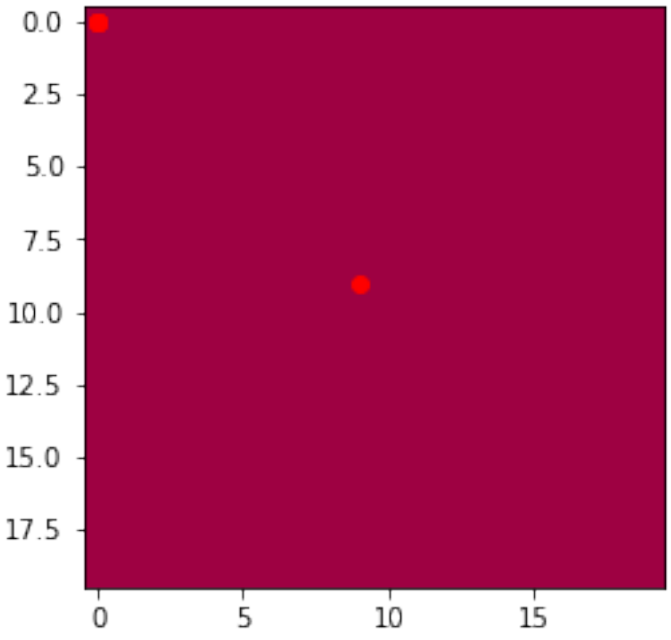
5 k = 5

6 L1

```
In [6]: p1.image_label = copy.deepcopy(p1.init_label)
        for i in range(0,p1.k):
```

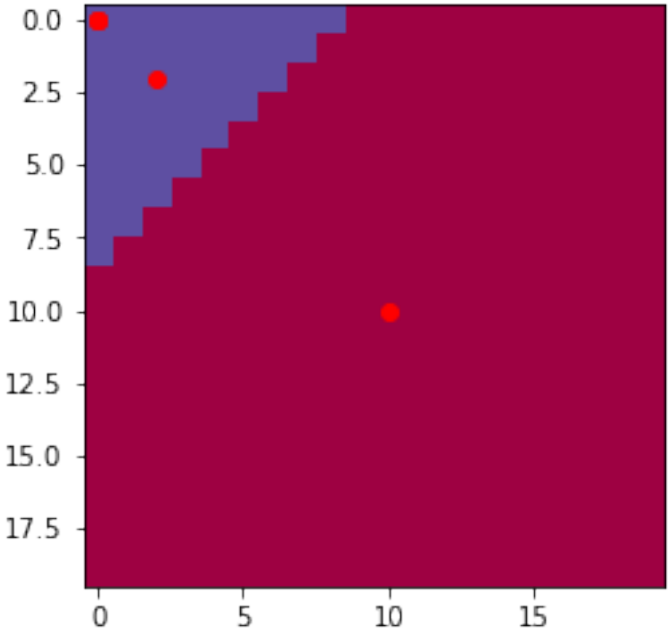
```
            p1.centroid[0][i] = 0
            p1.centroid[1][i] = 0


        a = 0
        while True:

            p1.track_centroid1.append([copy.deepcopy(p1.centroid[0]), copy.deepcopy(p1.centroi
            a += 1
            p1.update_label1()
            old_centroid = copy.deepcopy(p1.centroid)
            p1.update_centroid()
            plt.imshow(p1.image_label, cmap=plt.cm.get_cmap('Spectral', p1.k))
            print(p1.centroid[0])
            print(p1.centroid[1])

            plt.plot(p1.centroid[1], p1.centroid[0], 'ro')
            plt.show()
            if np.array_equal(old_centroid, p1.centroid):
                break
```

```
[9 0 0 0 0]
[9 0 0 0 0]
```

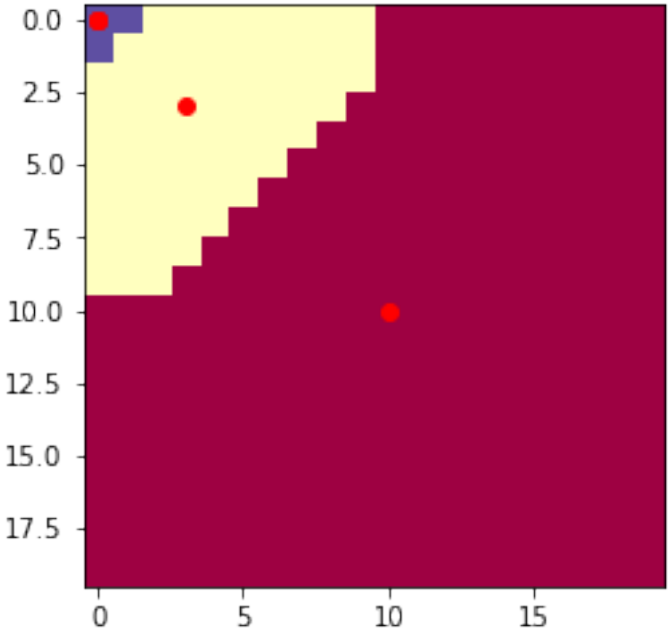



```
[10  2  0  0  0]
```

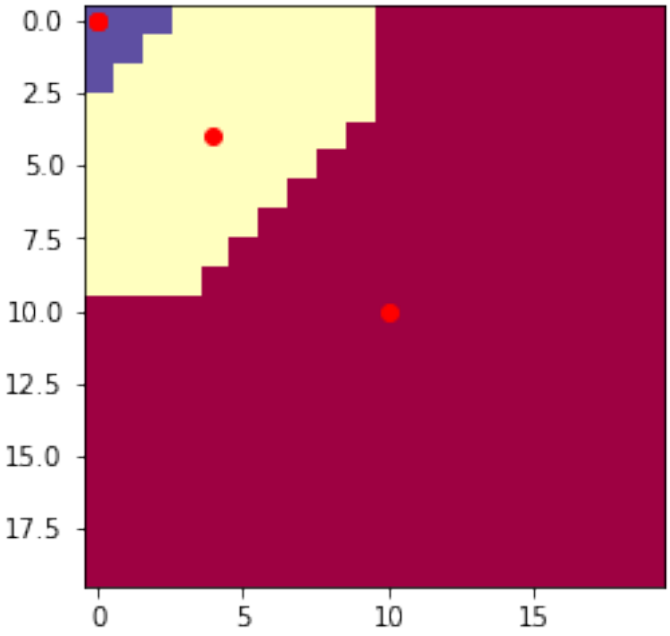
```
[10   2   0   0   0]
```



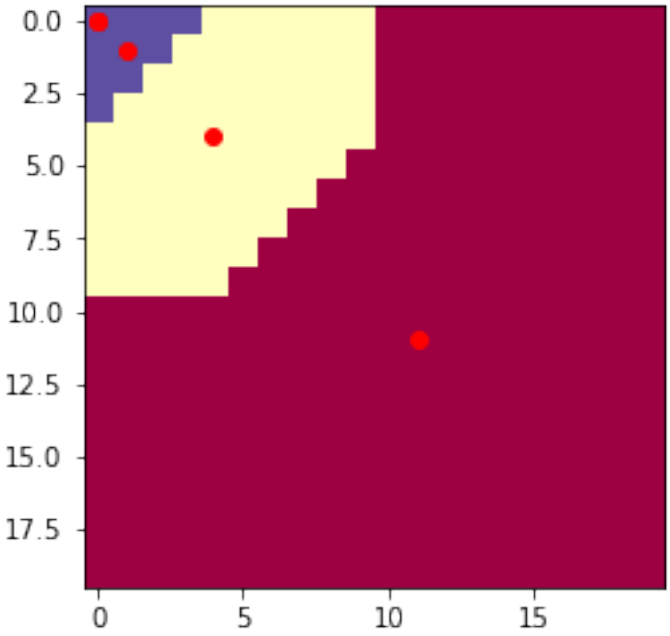

```
[10   3   0   0   0]
[10   3   0   0   0]
```



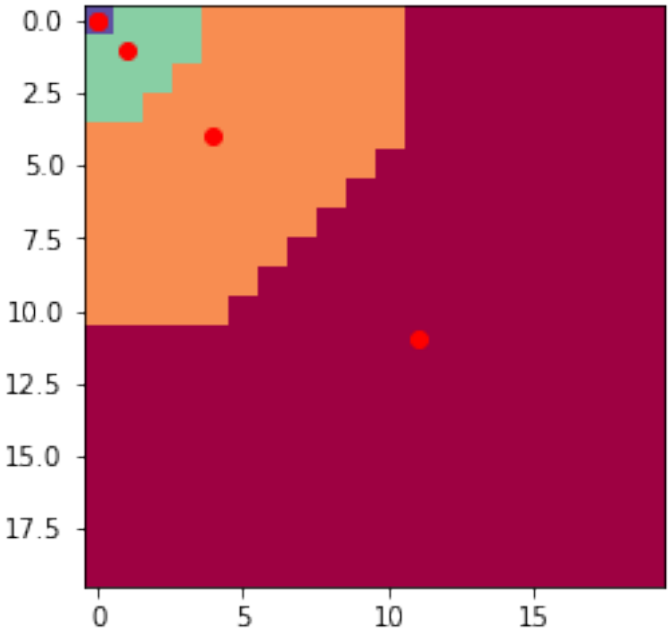
```
[10  4  0  0  0]
[10  4  0  0  0]
```




```
[11  4  1  0  0]
[11  4  1  0  0]
```

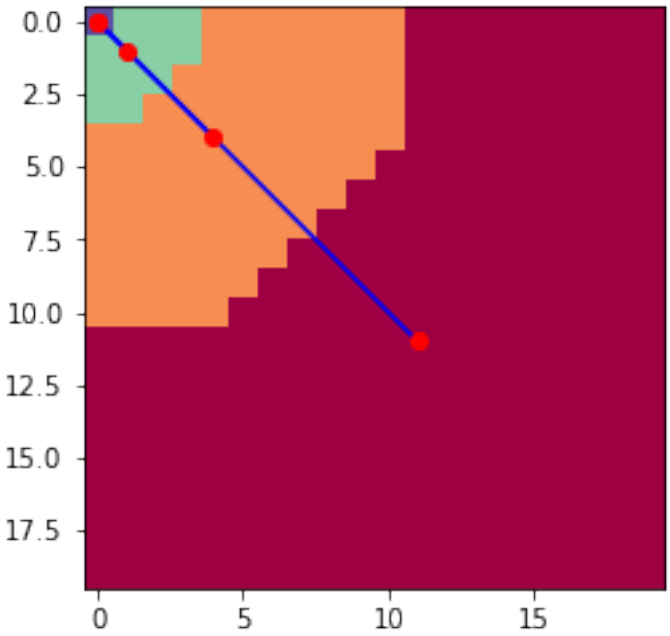



```
[11  4  1  0  0]
[11  4  1  0  0]
```



In [7]:
```
one = np.zeros((a,2), dtype = int)

for j in range(0,p1.k):
    for i in range(0,a):
        one[i] = [p1.track_centroid1[i][0][j], p1.track_centroid1[i][1][j]]

    plt.plot(one.T[1], one.T[0], 'b-',one.T[1][a-1], one.T[0][a-1], 'ro')

plt.imshow(p1.image_label, cmap=plt.cm.get_cmap('Spectral', p1.k))
```

Out[7]: <matplotlib.image.AxesImage at 0x277b56d43c8>



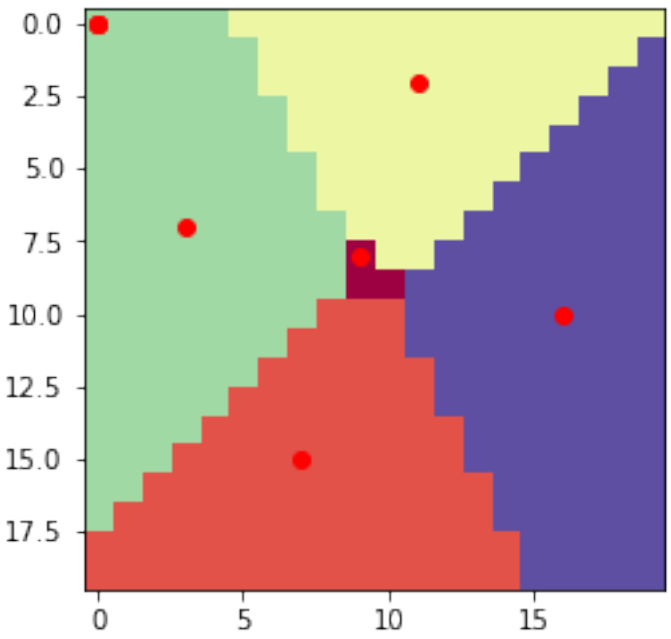
7 k = 8

8 L2

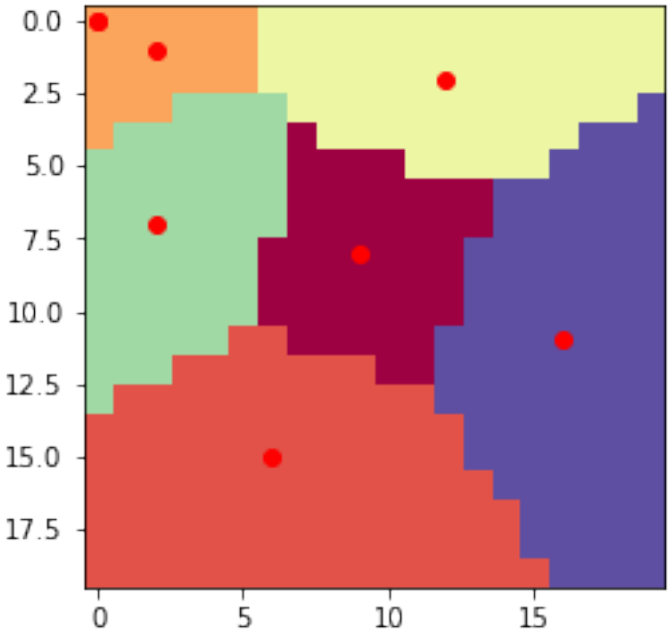
In [8]:
```
p2 = AllProcess()
# k = 8
p2.setdata(8)
p2.random_labeling()
p2.init_label = copy.deepcopy(p2.image_label)
p2.update_centroid()

a = 0
while True:
```

```
            p2.track_centroid.append([copy.deepcopy(p2.centroid[0]), copy.deepcopy(p2.centroid
            a += 1
            p2.update_label2()
            old_centroid = copy.deepcopy(p2.centroid)
            p2.update_centroid()
            plt.imshow(p2.image_label, cmap=plt.cm.get_cmap('Spectral', p2.k))
            print(p2.centroid[0])
            print(p2.centroid[1])

            plt.plot(p2.centroid[1], p2.centroid[0], 'ro')
            plt.show()
            if np.array_equal(old_centroid, p2.centroid):
                break
```

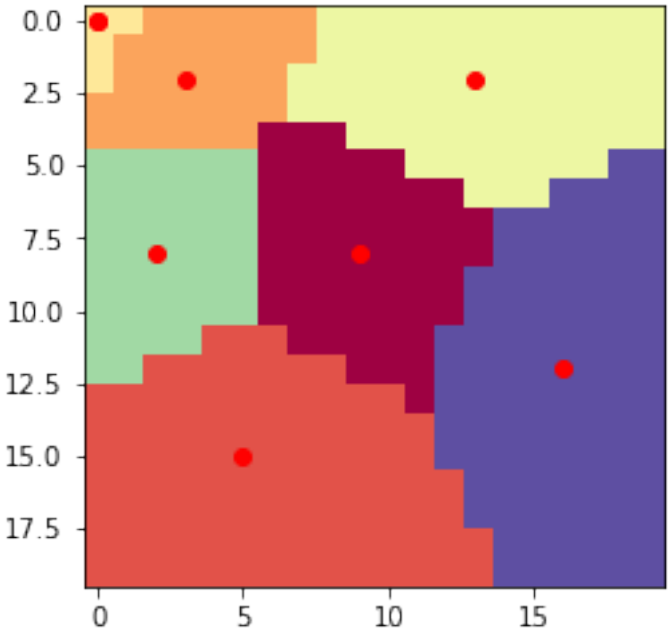
```
[ 8 15  0  0  2  7  0 10]
[ 9  7  0  0 11  3  0 16]
```



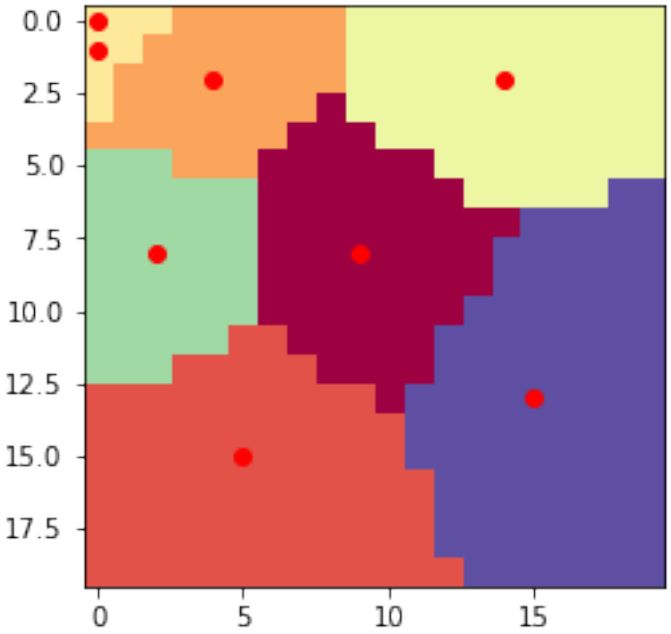

```
[ 8 15  1  0  2  7  0 11]
[ 9  6  2  0 12  2  0 16]
```



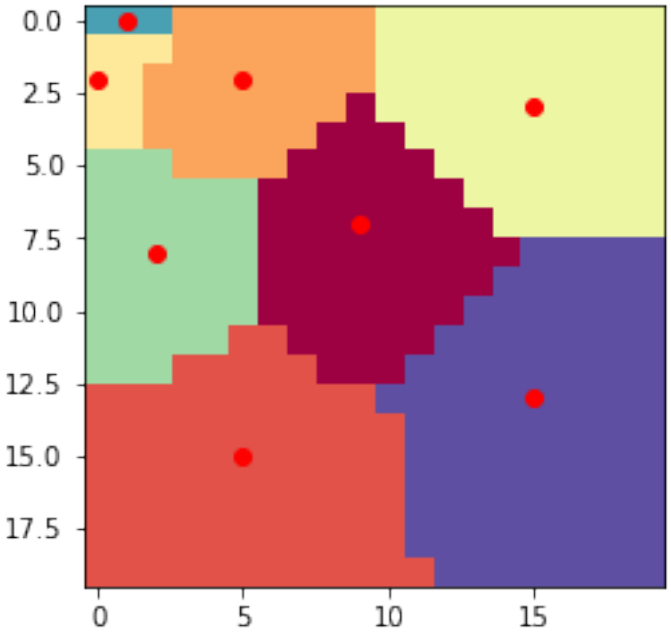

```
[ 8 15  2  0  2  8  0 12]
[ 9  5  3  0 13  2  0 16]
```



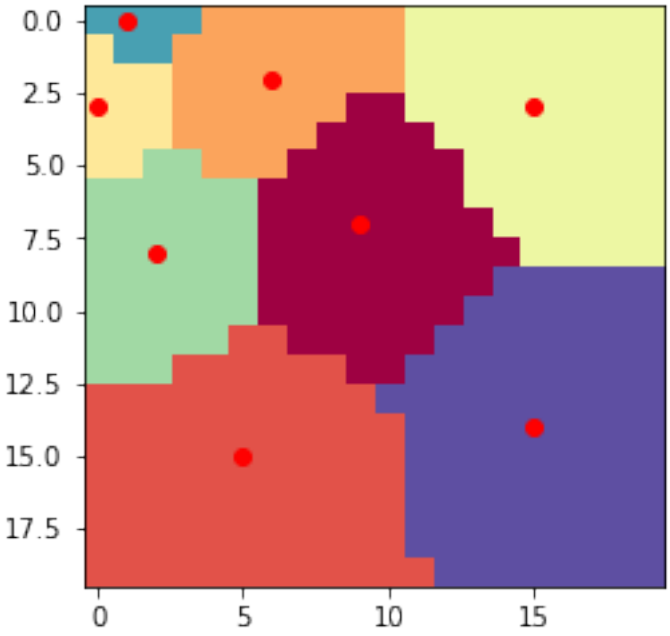
```
[ 8 15  2  1  2  8  0 13]
[ 9  5  4  0 14  2  0 15]
```



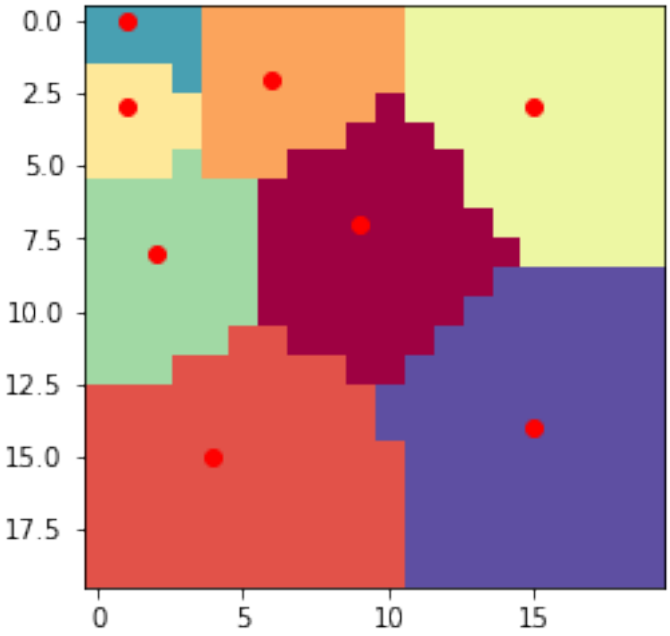

```
[ 7 15  2  2  3  8  0 13]
[ 9  5  5  0 15  2  1 15]
```



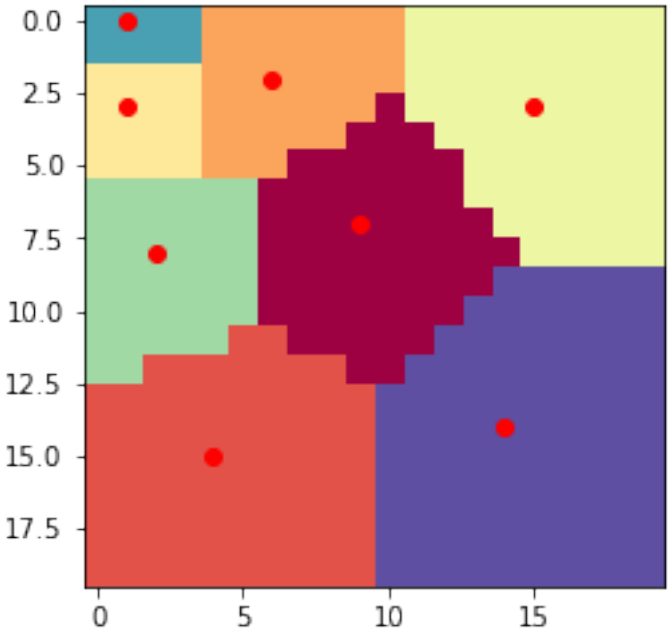
```
[ 7 15  2  3  3  8  0 14]
[ 9  5  6  0 15  2  1 15]
```



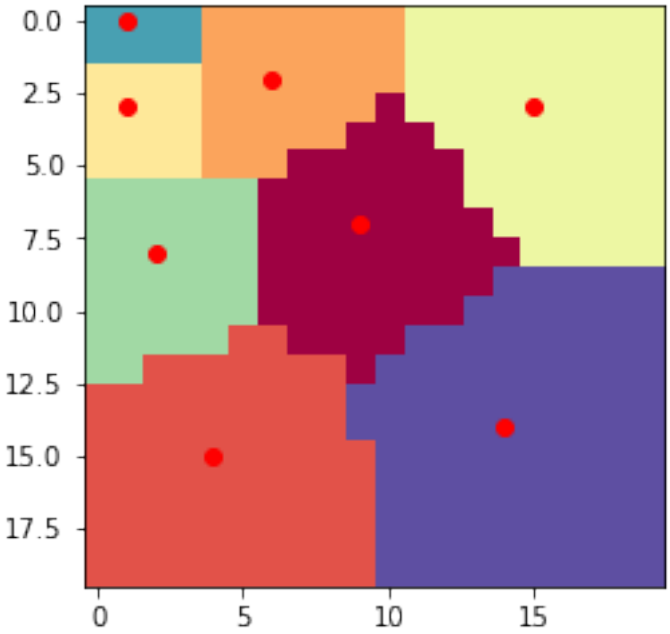

```
[ 7 15  2  3  3  8  0 14]
[ 9  4  6  1 15  2  1 15]
```




```
[ 7 15  2  3  3  8  0 14]
[ 9  4  6  1 15  2  1 14]
```

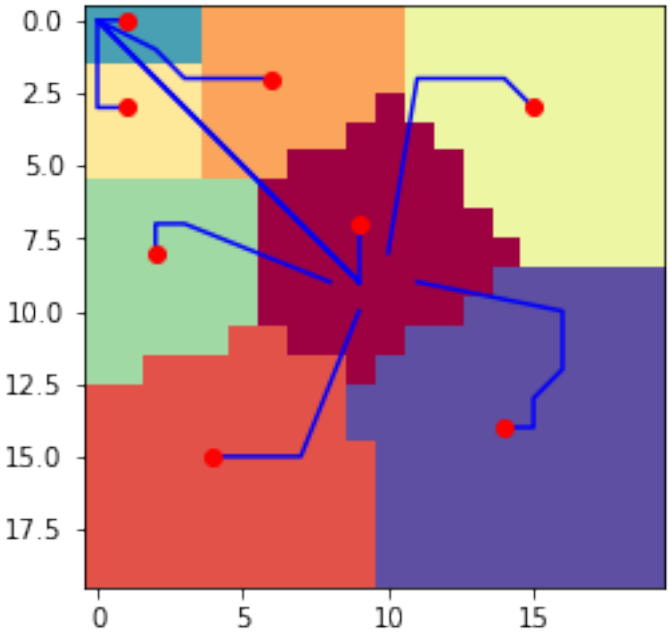


```
[ 7 15  2  3  3  8  0 14]
[ 9  4  6  1 15  2  1 14]
```



In [9]:
```
one = np.zeros((a,2), dtype = int)

        for j in range(0,p2.k):
            for i in range(0,a):
                one[i] = [p2.track_centroid[i][0][j], p2.track_centroid[i][1][j]]

            plt.plot(one.T[1], one.T[0], 'b-',one.T[1][a-1], one.T[0][a-1], 'ro')

        plt.imshow(p2.image_label, cmap=plt.cm.get_cmap('Spectral', p2.k))
```

Out[9]: <matplotlib.image.AxesImage at 0x277b5d32748>

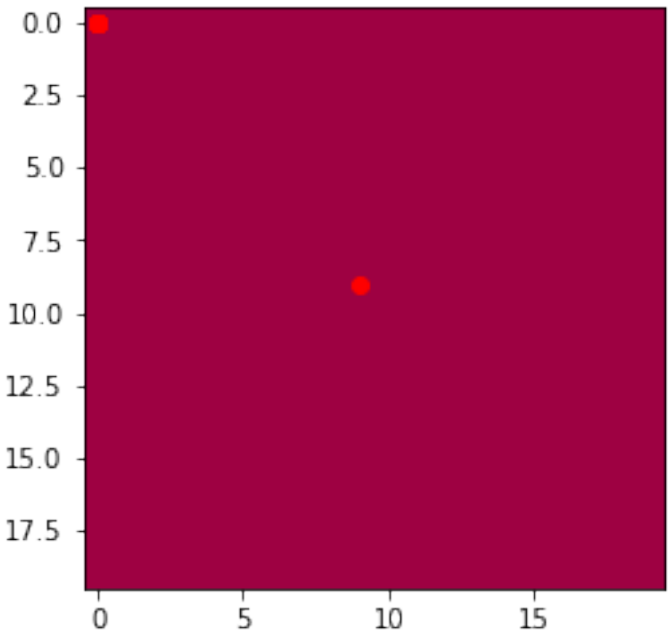


9 k = 8

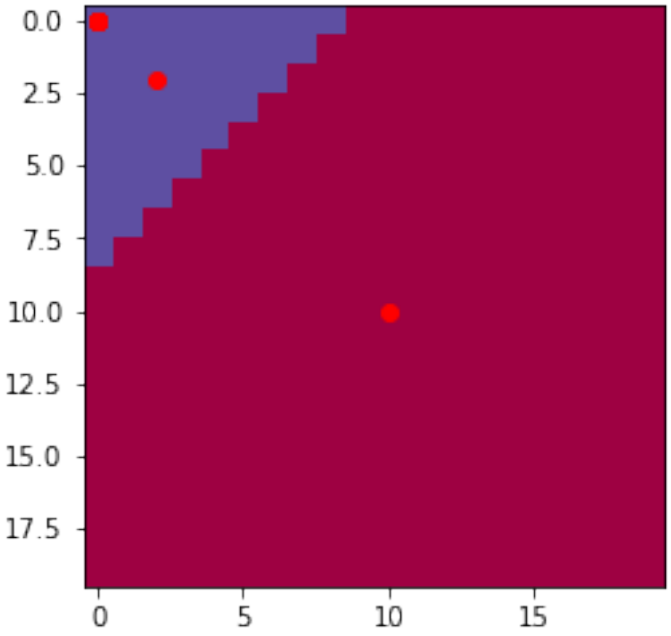
10 L1

```
In [10]: p2.image_label = copy.deepcopy(p2.init_label)
         for i in range(0,p2.k):
             p2.centroid[0][i] = 0
             p2.centroid[1][i] = 0

         a = 0
         while True:

             p2.track_centroid1.append([copy.deepcopy(p2.centroid[0]), copy.deepcopy(p2.centro
             a += 1
             p2.update_label1()
             old_centroid = copy.deepcopy(p2.centroid)
             p2.update_centroid()
             plt.imshow(p2.image_label, cmap=plt.cm.get_cmap('Spectral', p2.k))
             print(p2.centroid[0])
             print(p2.centroid[1])

             plt.plot(p2.centroid[1], p2.centroid[0], 'ro')
             plt.show()
             if np.array_equal(old_centroid, p2.centroid):
                 break
```

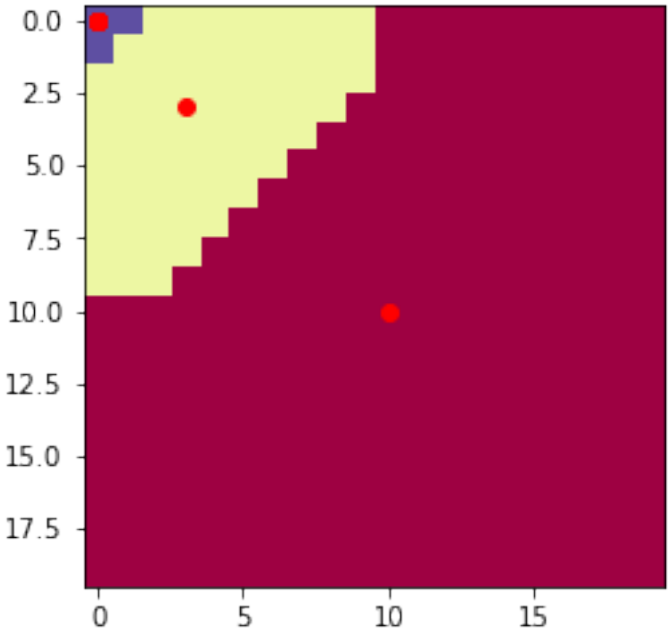
```
[9 0 0 0 0 0 0 0]
[9 0 0 0 0 0 0 0]
```



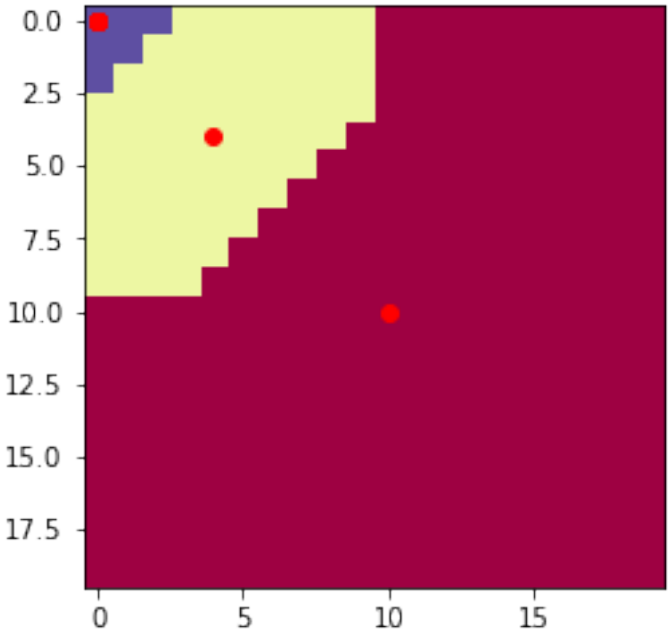

```
[10  2  0  0  0  0  0  0]
[10  2  0  0  0  0  0  0]
```



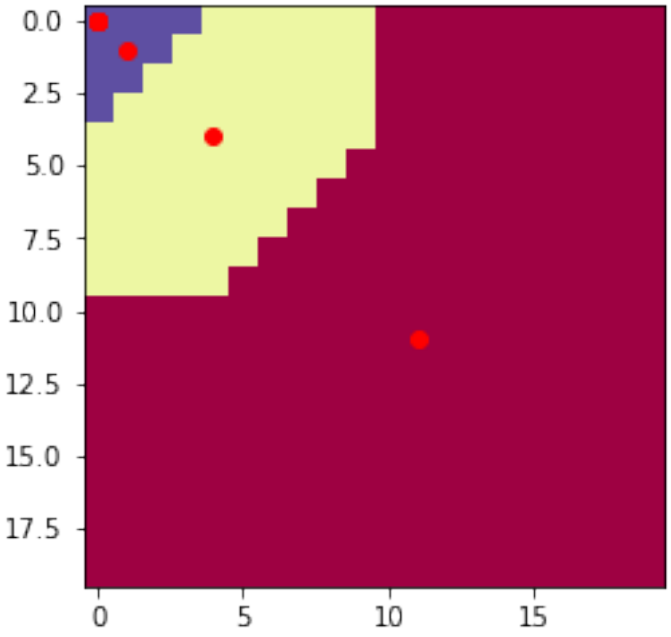
```
[10  3  0  0  0  0  0  0]
[10  3  0  0  0  0  0  0]
```



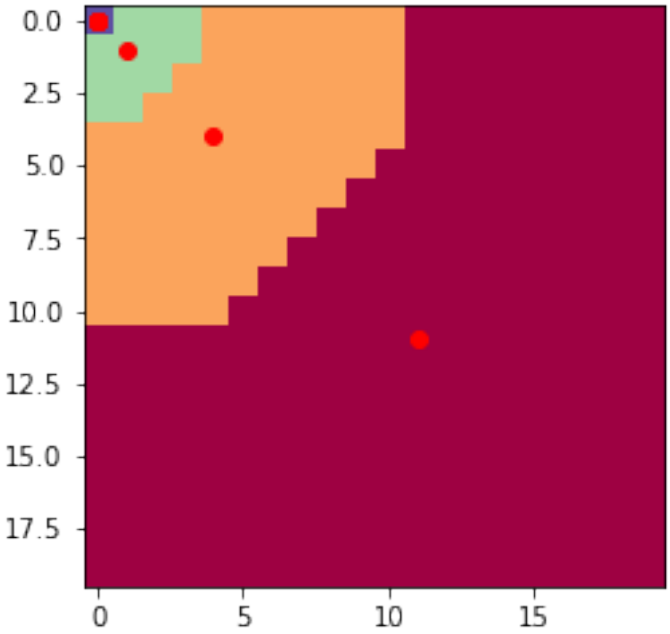

```
[10  4  0  0  0  0  0  0]
[10  4  0  0  0  0  0  0]
```



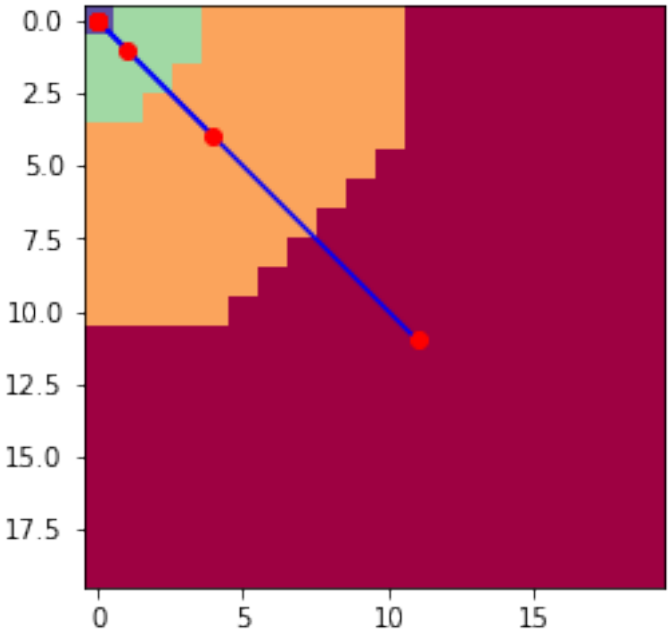

```
[11  4  1  0  0  0  0  0]
[11  4  1  0  0  0  0  0]
```



```
[11  4  1  0  0  0  0  0]
[11  4  1  0  0  0  0  0]
```




```
In [11]: one = np.zeros((a,2), dtype = int)

         for j in range(0,p2.k):
             for i in range(0,a):
                 one[i] = [p2.track_centroid1[i][0][j], p2.track_centroid1[i][1][j]]

             plt.plot(one.T[1], one.T[0], 'b-',one.T[1][a-1], one.T[0][a-1], 'ro')

         plt.imshow(p2.image_label, cmap=plt.cm.get_cmap('Spectral', p2.k))

Out[11]: <matplotlib.image.AxesImage at 0x277b5b77cf8>
```



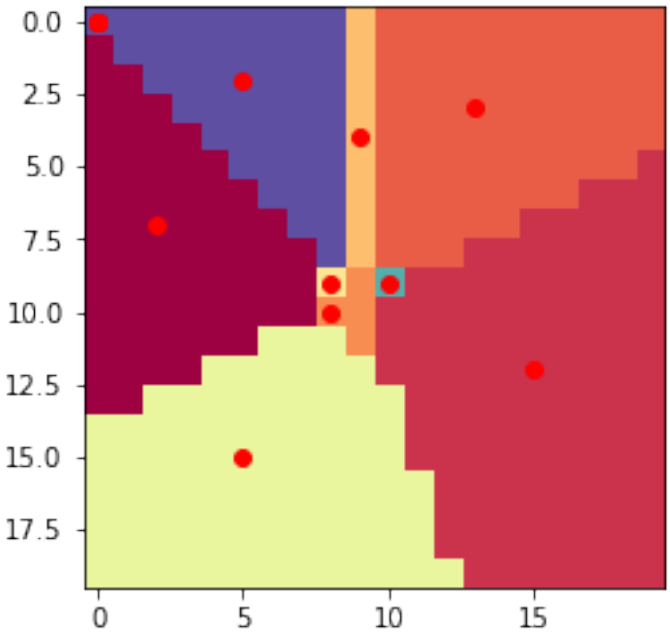
11 k = 13

12 L2

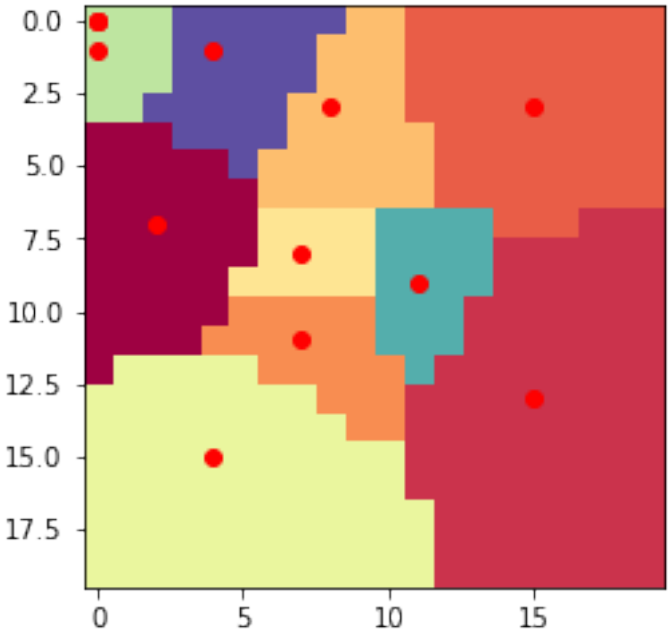
```
In [12]: p3 = AllProcess()
         # k = 13
         p3.setdata(13)
         p3.random_labeling()
         p3.init_label = copy.deepcopy(p3.image_label)
         p3.update_centroid()

         a = 0
         while True:

             p3.track_centroid.append([copy.deepcopy(p3.centroid[0]), copy.deepcopy(p3.centroi
             a += 1
             p3.update_label2()
             old_centroid = copy.deepcopy(p3.centroid)
             p3.update_centroid()
             plt.imshow(p3.image_label, cmap=plt.cm.get_cmap('Spectral', p3.k))
             print(p3.centroid[0])
             print(p3.centroid[1])

             plt.plot(p3.centroid[1], p3.centroid[0], 'ro')
```

```
            plt.show()
            if np.array_equal(old_centroid, p3.centroid):
                break
```

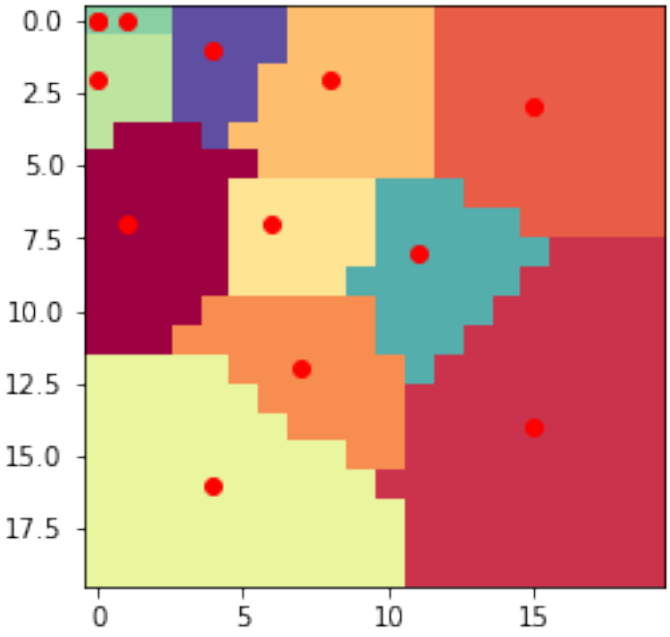
```
[ 7 12  3 10  4  9 15  0  0  9  0  2  0]
[ 2 15 13  8  9  8  5  0  0 10  0  5  0]
```



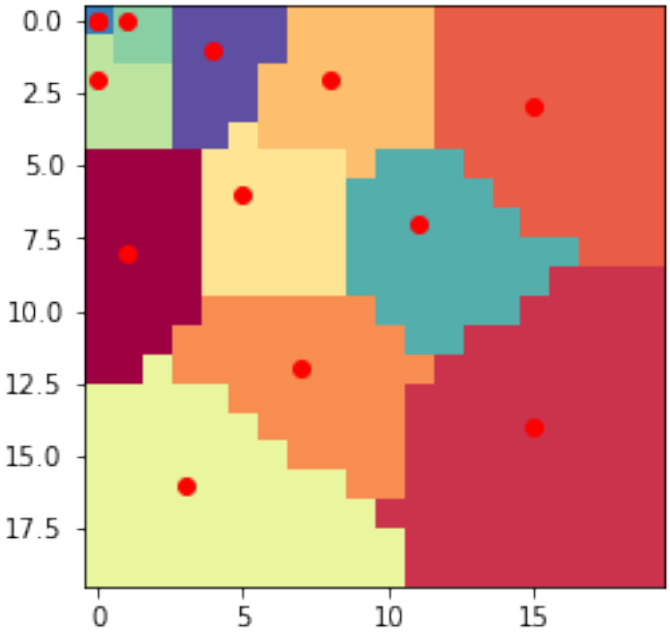

```
[ 7 13  3 11  3  8 15  1  0  9  0  1  0]
[ 2 15 15  7  8  7  4  0  0 11  0  4  0]
```



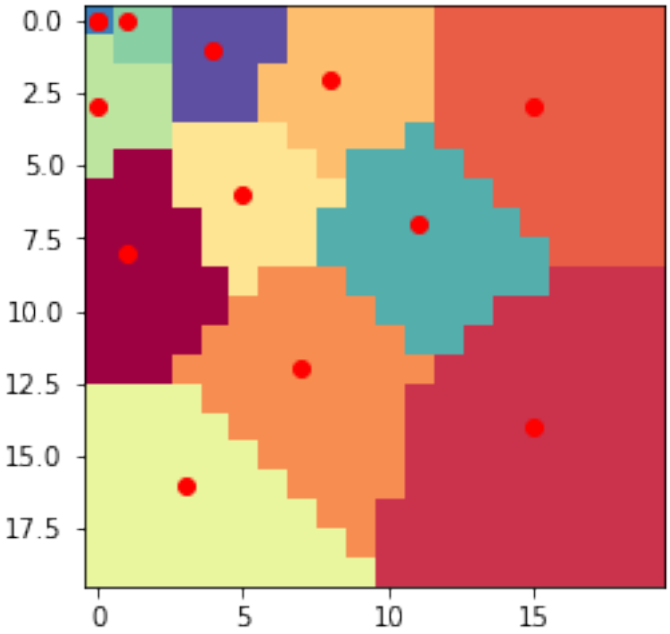

```
[ 7 14  3 12  2  7 16  2  0  8  0  1  0]
[ 1 15 15  7  8  6  4  0  1 11  0  4  0]
```



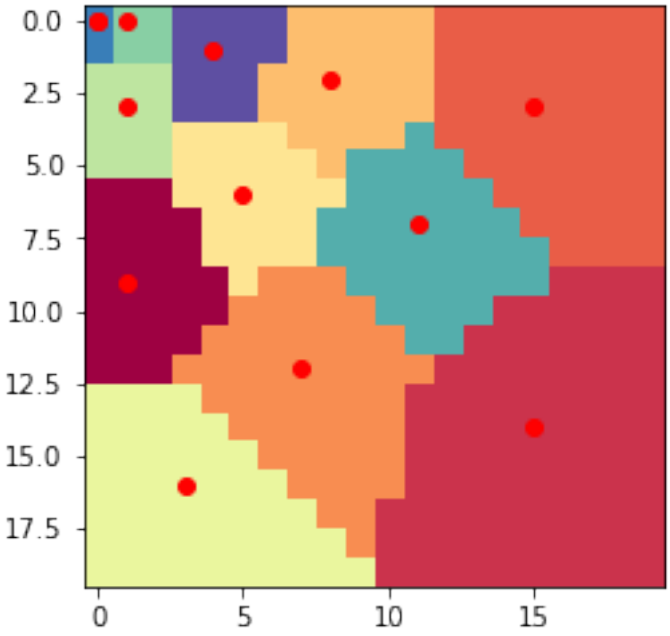
```
[ 8 14  3 12  2  6 16  2  0  7  0  1  0]
[ 1 15 15  7  8  5  3  0  1 11  0  4  0]
```



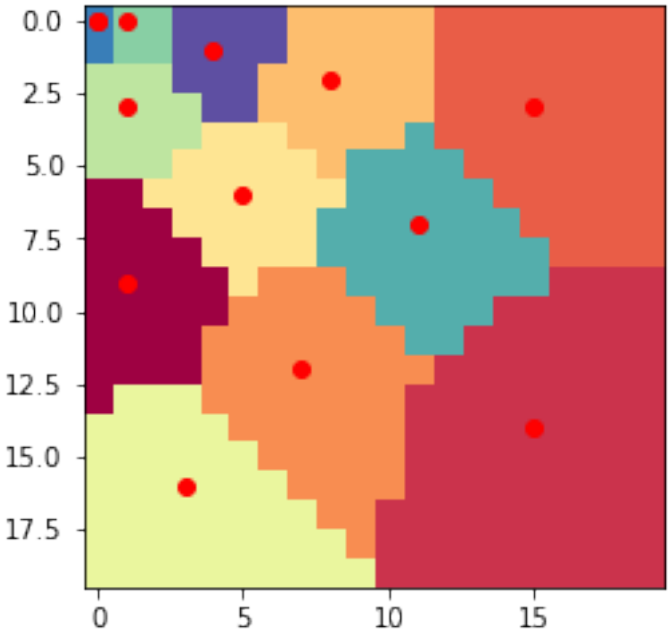

```
[ 8 14  3 12  2  6 16  3  0  7  0  1  0]
[ 1 15 15  7  8  5  3  0  1 11  0  4  0]
```



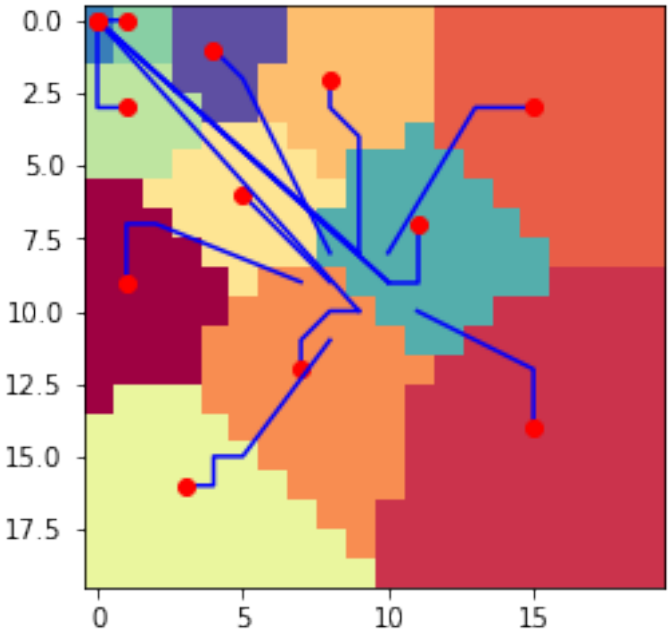
```
[ 9 14  3 12  2  6 16  3  0  7  0  1  0]
[ 1 15 15  7  8  5  3  1  1 11  0  4  0]
```




```
[ 9 14  3 12  2  6 16  3  0  7  0  1  0]
[ 1 15 15  7  8  5  3  1  1 11  0  4  0]
```




```
In [13]: one = np.zeros((a,2), dtype = int)

         for j in range(0,p3.k):
             for i in range(0,a):
                 one[i] = [p3.track_centroid[i][0][j], p3.track_centroid[i][1][j]]

             plt.plot(one.T[1], one.T[0], 'b-',one.T[1][a-1], one.T[0][a-1], 'ro')

         plt.imshow(p3.image_label, cmap=plt.cm.get_cmap('Spectral', p3.k))

Out[13]: <matplotlib.image.AxesImage at 0x277b5bc6ef0>
```

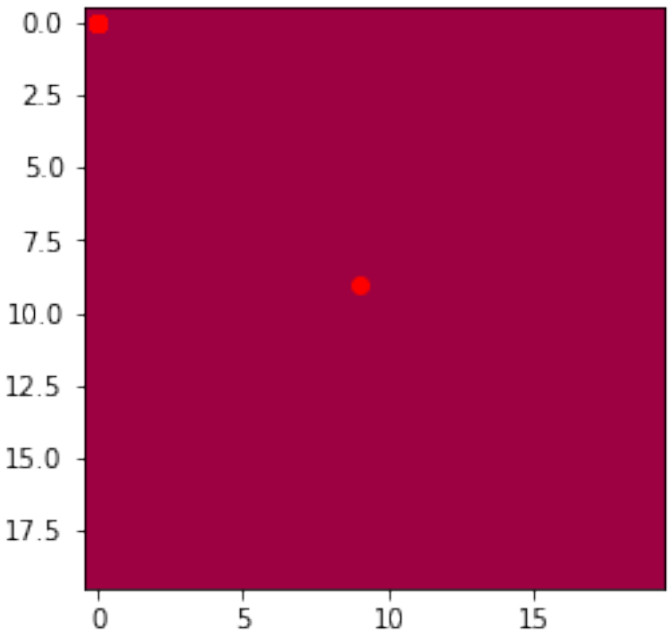


13 k = 13

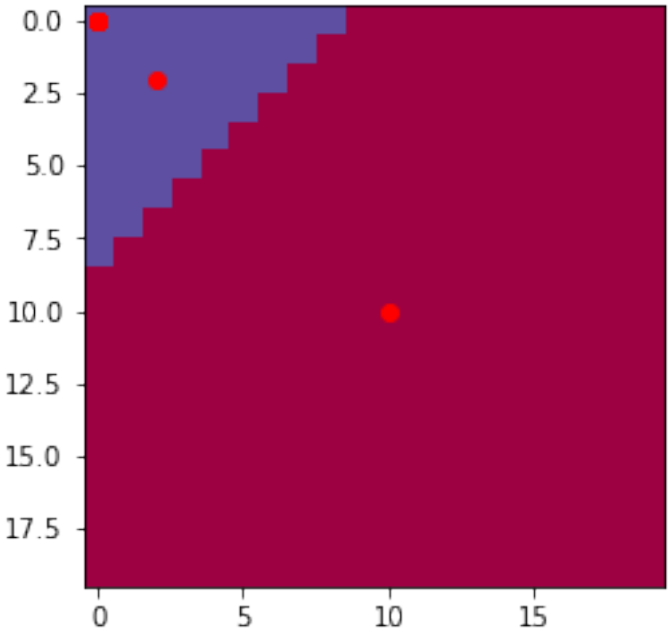
14 L1

```
In [14]: p3.image_label = copy.deepcopy(p3.init_label)
         for i in range(0,p3.k):
             p3.centroid[0][i] = 0
             p3.centroid[1][i] = 0

         a = 0
         while True:

             p3.track_centroid1.append([copy.deepcopy(p3.centroid[0]), copy.deepcopy(p3.centro
             a += 1
             p3.update_label1()
             old_centroid = copy.deepcopy(p3.centroid)
             p3.update_centroid()
             plt.imshow(p3.image_label, cmap=plt.cm.get_cmap('Spectral', p3.k))
             print(p3.centroid[0])
             print(p3.centroid[1])

             plt.plot(p3.centroid[1], p3.centroid[0], 'ro')
             plt.show()
             if np.array_equal(old_centroid, p3.centroid):
                 break
```

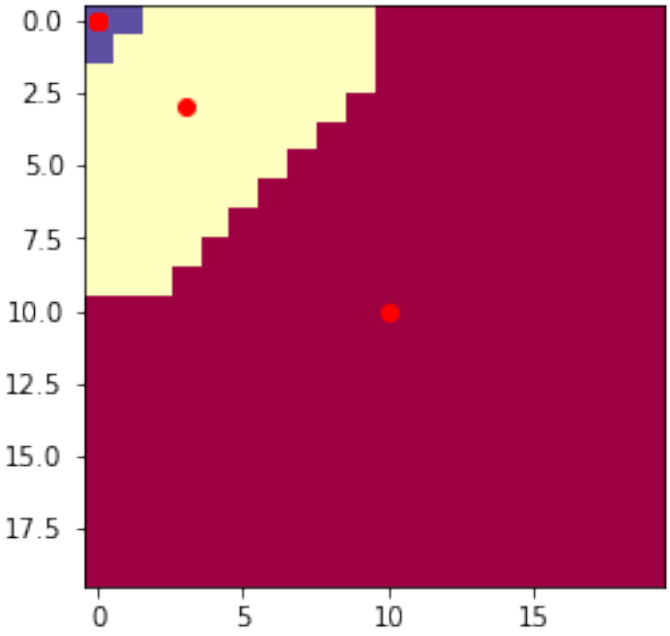
```
[9 0 0 0 0 0 0 0 0 0 0 0 0]
[9 0 0 0 0 0 0 0 0 0 0 0 0]
```



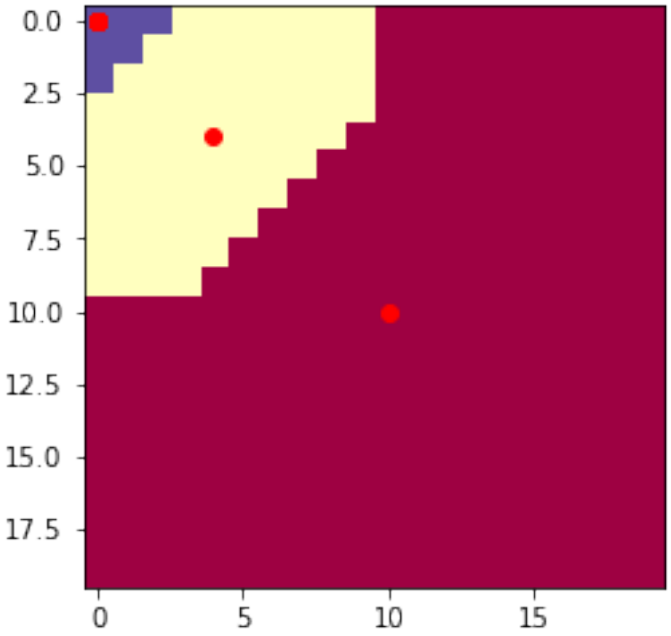

```
[10  2  0  0  0  0  0  0  0  0  0  0  0]
[10  2  0  0  0  0  0  0  0  0  0  0  0]
```



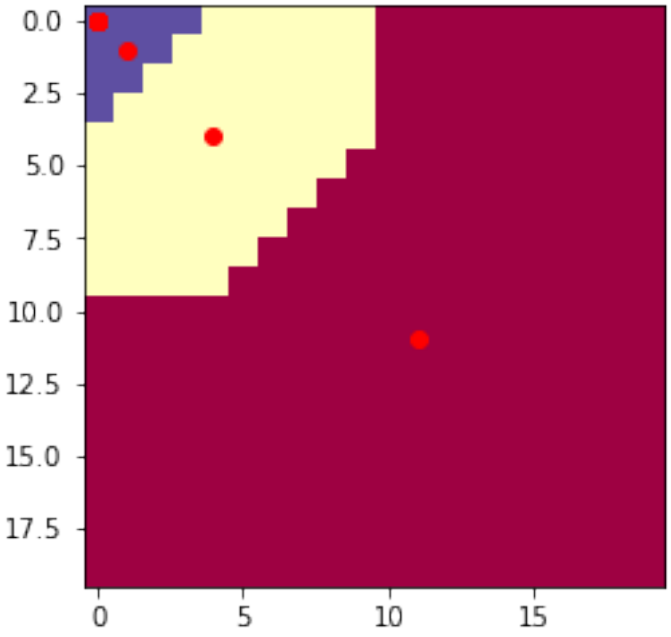
```
[10  3  0  0  0  0  0  0  0  0  0  0  0]
[10  3  0  0  0  0  0  0  0  0  0  0  0]
```



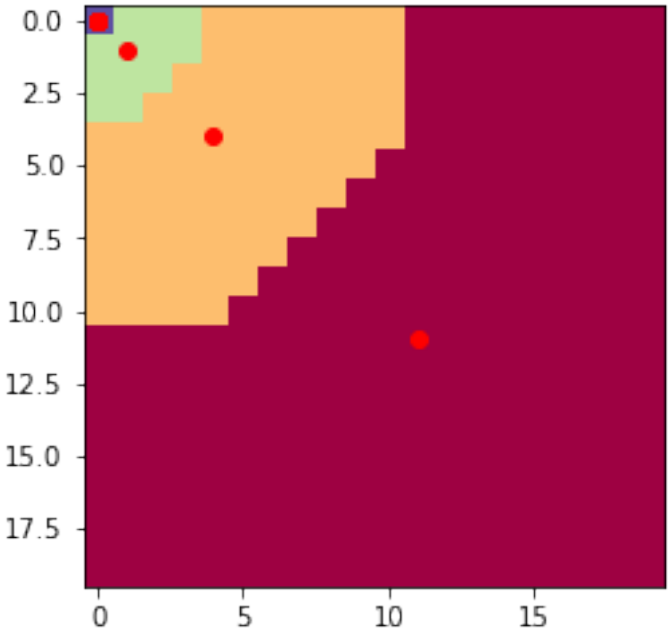

```
[10  4  0  0  0  0  0  0  0  0  0  0  0]
[10  4  0  0  0  0  0  0  0  0  0  0  0]
```



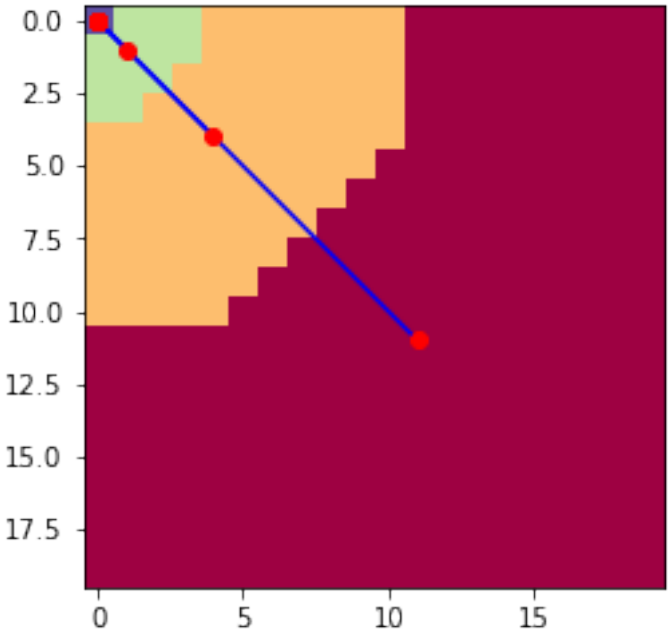

```
[11  4  1  0  0  0  0  0  0  0  0  0  0]
[11  4  1  0  0  0  0  0  0  0  0  0  0]
```



```
[11  4  1  0  0  0  0  0  0  0  0  0  0]
[11  4  1  0  0  0  0  0  0  0  0  0  0]
```




```
In [15]: one = np.zeros((a,2), dtype = int)

         for j in range(0,p3.k):
             for i in range(0,a):
                 one[i] = [p3.track_centroid1[i][0][j], p3.track_centroid1[i][1][j]]

             plt.plot(one.T[1], one.T[0], 'b-',one.T[1][a-1], one.T[0][a-1], 'ro')

         plt.imshow(p3.image_label, cmap=plt.cm.get_cmap('Spectral', p3.k))

Out[15]: <matplotlib.image.AxesImage at 0x277b5cfecc0>
```



In [ ]:

In [ ]: