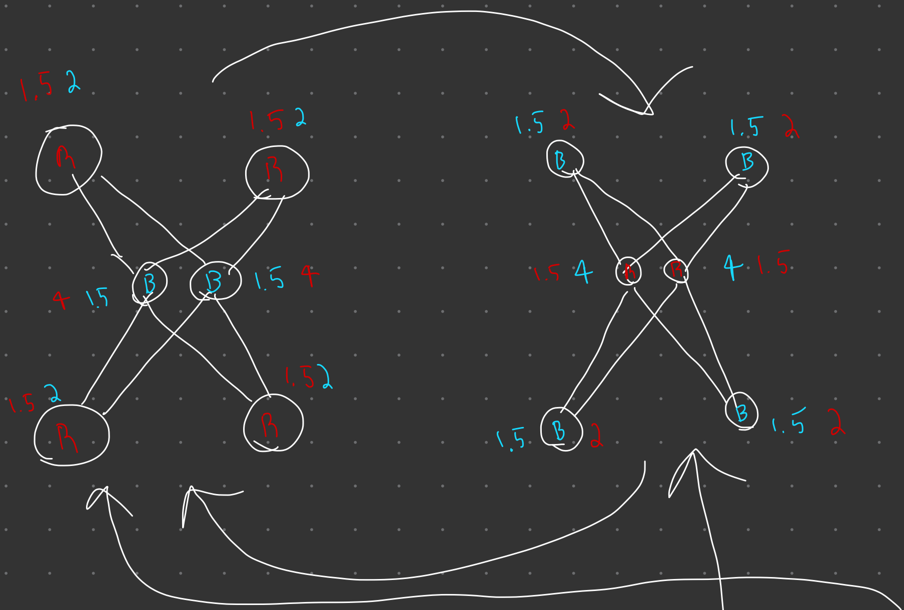
Set 5.

1.



Say we have this Subgraph Coloring or .

Say to start Blue or red have nodes on some other, unconnected part of the graph s.t. both chose k starting nodes. This pattern above would cycle infinitely, causing the sim to not converge and stop after max iters.

In [29]:

```
import networkx as nx
import numpy as np
from sklearn.cluster import KMeans, SpectralClustering
from sklearn.datasets import make_circles
import matplotlib.pyplot as plt
```

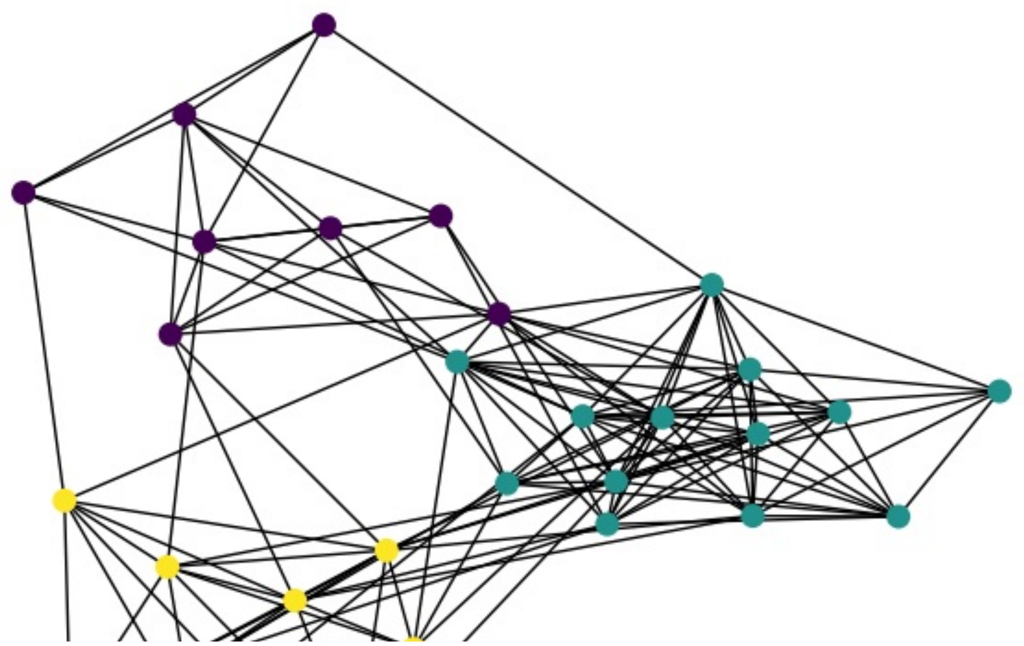
In [8]:

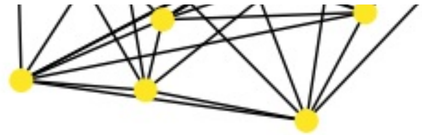
```
# SSBM Graph
n = 30
k = 3
A = 0.7
B = 0.1
W = [[A, B, B], [B, A, B], [B, B, A]]

G = nx.Graph()
labels = [0, 1, 2]

for i in range(n):
    label = np.random.choice(labels)
    G.add_node(str(i) + " " + str(label))

nodes = list(G.nodes)
for i in range(n):
    for j in range(i + 1, n, 1):
        node1 = nodes[i]
        node2 = nodes[j]
        label1 = node1.split(" ")[1]
        label2 = node2.split(" ")[1]
        p = W[int(label1)][int(label2)]
        if np.random.random() < p:
            G.add_edge(node1, node2)
```

In [15]:

```
spectral = SpectralClustering(k)
clusters = spectral.fit_predict(nx.to_numpy_array(G))
nx.draw(G, node_color=clusters, node_size=70)
```

```
/opt/homebrew/lib/python3.10/site-packages/sklearn/cluster/_spectral.py:717: UserWarning:
The spectral clustering API has changed. ``fit``now constructs an affinity matrix from da
ta. To use a custom affinity matrix, set ``affinity=precomputed``.
  warnings.warn(
```

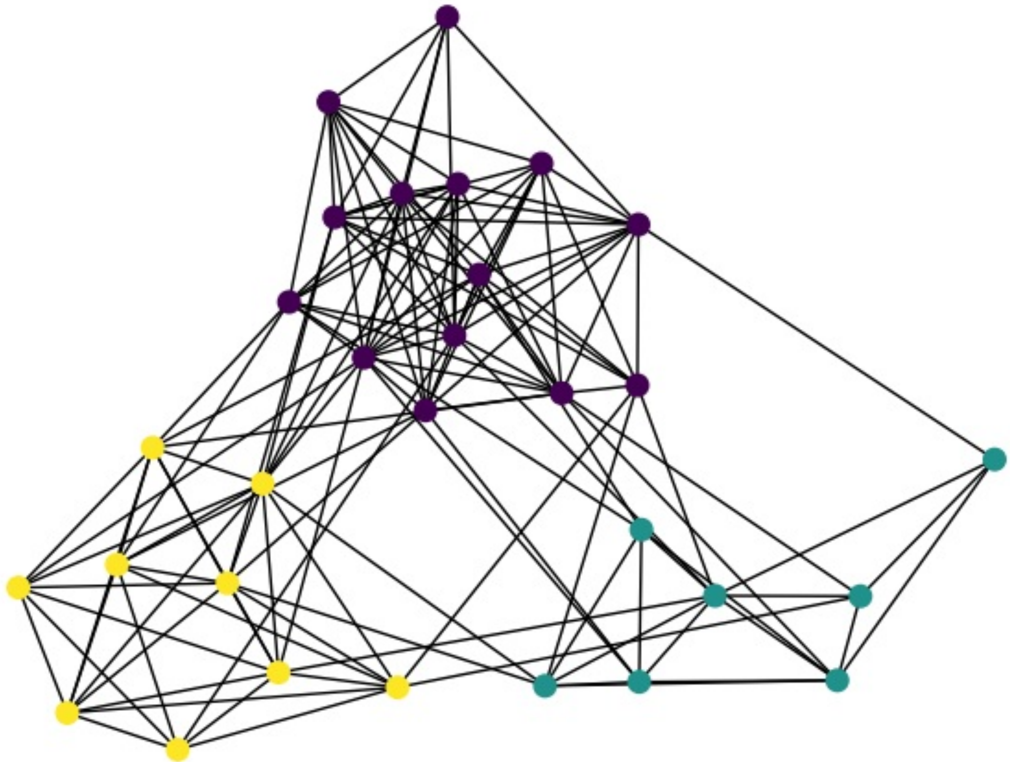




In [16]:

```
kmeans = KMeans(k)
clusters = kmeans.fit_predict(nx.to_numpy_array(G))
nx.draw(G, node_color=clusters, node_size=70)
```

```
/opt/homebrew/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```



There is a slight difference with 2 of the clusters containing differing numbers of nodes, but overall both algorithms capture very similar clusters. Both algorithms seem to capture the SSBM clusters pretty well as well.

In [19]:

```
circles = make_circles(500, noise=0.01)
```

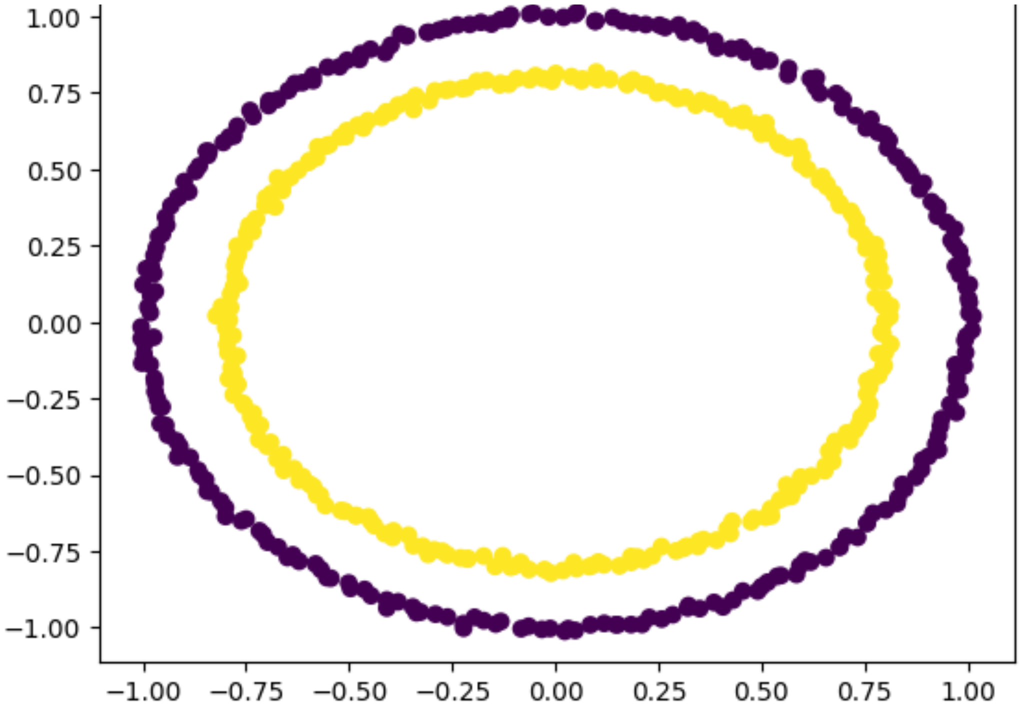
In [34]:

```
spectral = SpectralClustering(2, affinity="nearest_neighbors")
clusters = spectral.fit_predict(circles[0])
plt.scatter(circles[0][:, 0], circles[0][:, 1], c=clusters)
```

```
/opt/homebrew/lib/python3.10/site-packages/sklearn/manifold/_spectral_embedding.py:274: UserWarning: Graph is not fully connected, spectral embedding may not work as expected.
  warnings.warn(
```

Out[34]:

```
<matplotlib.collections.PathCollection at 0x17fe87190>
```



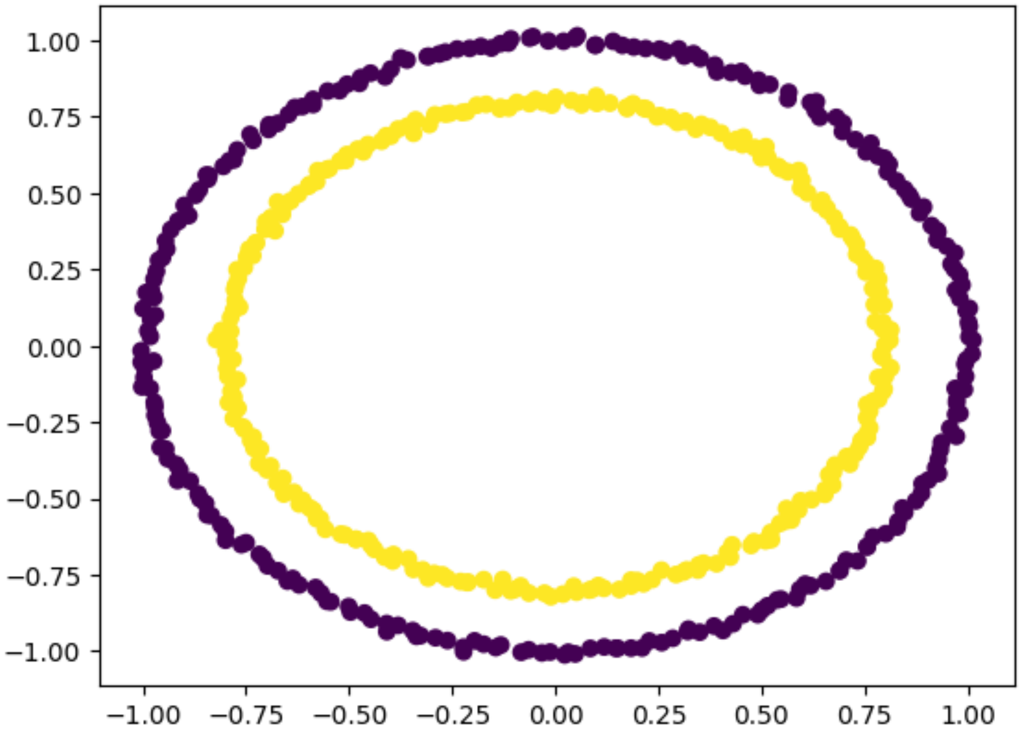
In [37]:

```
kmeans = KMeans(2)
clusters = spectral.fit_predict(circles[0])
plt.scatter(circles[0][:, 0], circles[0][:, 1], c=clusters)
```

```
/opt/homebrew/lib/python3.10/site-packages/sklearn/manifold/_spectral_embedding.py:274: U
serWarning: Graph is not fully connected, spectral embedding may not work as expected.
  warnings.warn(
```

Out[37]:

<matplotlib.collections.PathCollection at 0x2a794c280>



Both algorithms perform essentially the same, separating the inside vs the outside circles well.

3.

A.

Player 1 \ Player 2	a	b	c	d	e
w	(3, 1)	(1, 1)	(6, 1)	(5, 2)	(2, 2)
x	(10, 4)	(1, 5)	(7, 3)	(3, 3)	(0, 0)
y	(0, 2)	(2, 5)	(2, 4)	(3, 3)	(1, 0)
z	(0, 0)	(0, 0)	(8, 0)	(0, 1)	(3, 3)
t	(4, 3)	(1, 2)	(1, 4)	(4, 6)	(1, 2)

	b	d	e
w	(1,1)	(5,2)	(2,2)
y	(2,5)	(3,3)	(1,0)
z	(0,0)	(0,1)	(3,3)

player 2 has no incentive to play A or C,
Thus player 1 then has no incentive to play X.
player 1 has no incentive to play t either.

Thus, bde survive for p2, wyz for p1

B.

$p(w) = p$ $\qquad$ $p(b) = x$

$p(y) = q$ $\qquad$ $p(d) = y$

$p(z) = 1-p-q$ $\qquad$ $p(e) = 1-x-y$

indiff. w,y: $p + 5q + 2(1-p-q) = 2p + 3q + 1 - p - q$

y,z: $2p + 3q + 1 - p - q = 3(1-p-q)$

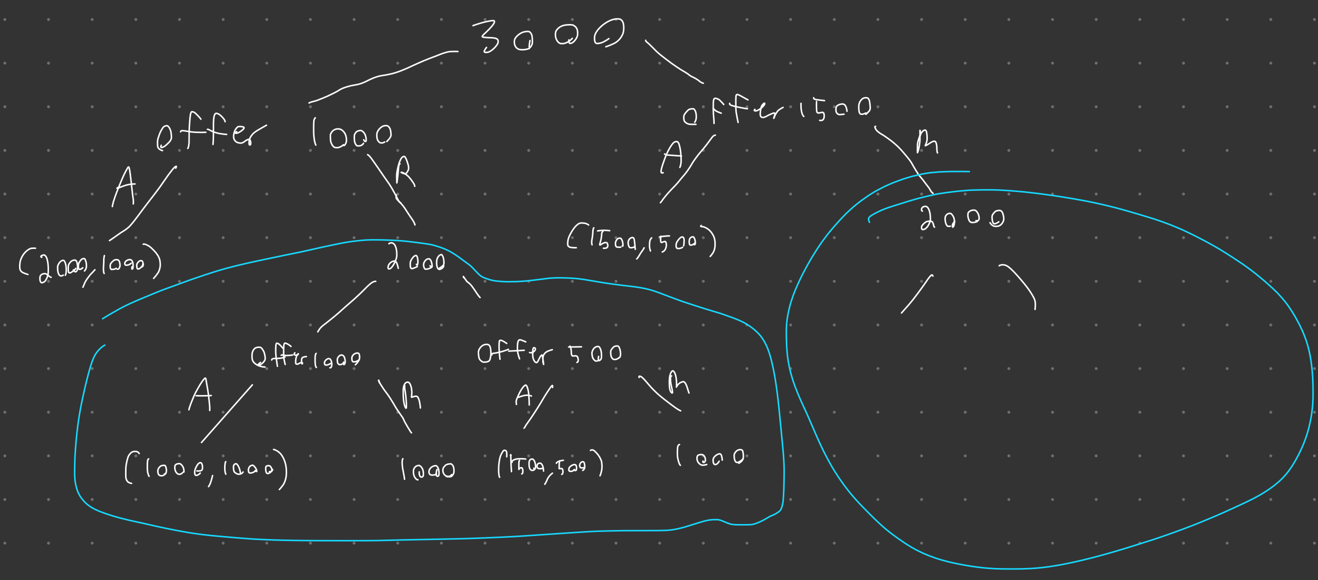
$p = \frac{1}{2} \quad q = 0 \qquad 1-p-q = \frac{1}{2}$

indiff b,d: $x + 5y = 2x + 3y + 1 - x - y$

d,e: $2x + 3y + 1 - x - y = 2x + 3(1-x-y)$

$x = \frac{1}{6} \quad y = \frac{1}{3}, \quad 1-x-y = \frac{1}{2}$

So: p1: $(\frac{1}{2}, 0, 0, \frac{1}{2}, 0)$ $\quad$ p2: $(0, \frac{1}{6}, 0, \frac{1}{3}, \frac{1}{2})$



M wants 1500, so he prefers if A offers 1500, but A wants the 2000 payoff so he would offer 1000. if M rejects then he can offer 1000 to be indifferent, but runs the risk of A rejecting resulting in less money. Ultimately A would offer 1001, since he gets 1999, and M is likely to accept since if he rejects he can really only get 1000 vs. 1001, the offer is 1001 instead of 1000 to avoid indifference by Mohan.

B. A offers 1500. This is the most he can get without M instantly rejecting. if rejection happens, M would likely make less money as A would reject offers of less than half the money because he can reject and get more next iteration. M would not spitefully reject 1500, and any bigger an offer means A gets less.

5. Consider:

		q	$1-q$
		A	B
p	A	(2,2)	(0,3)
$1-p$	B	(3,0)	(1,1)

All payoffs are different.

$$2pq + 3(1-p)q + (1-p)(1-q) = 2pq + 3(1-q)p + (1-p)(1-q)$$

$q = p$ is Soln.

infinite equilibria!

if $m = 1$, each player only has 1 strategy, So necessarily there is only 1 nash eq. So finite equilibria. Thus, $m = 1$ is an upper bound where Ann is right.