HW7

August 14, 2019

1 HW7

Author: Luis Perez
Last Updated: Aug. 10th, 2019

[37]:
```
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
```

1.1 Problem 2: Drawing a representation of a graph

[10]:
```
def getProblem2Inputs():
    n = 60
    m = 73
    A = np.array([
    [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,␣
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,␣
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
 ↪0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,␣
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,␣
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,␣
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
 ↪0, 0, 0, 0, 1, 0, 0, 0, 0, 0, ],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
 ↪0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
```

```
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 1, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,␣
↪0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 1, 0, 0, 0, 0, 1, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ],
```

```
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 1, 1, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪1, 0, 1, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 1, 0, 0, ],
    [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 1, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 1, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,␣
↪0, 1, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
```

```
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ],
    [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,␣
↪0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 1, 0, 0, 0, ],
    [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 1, 0, 0, 0, 0, 0, 1, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 1, 0, 0, 0, ],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,␣
↪1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,␣
↪0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 1, 0, 0, 0, 0, 0, 0, 0, ],
```

```
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, ],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ],
    ])
    x_circ = np.array([
        [0.1816],
        [0.1786],
        [0.1736],
        [0.1668],
        [0.1581],
        [0.1477],
        [0.1357],
        [0.1222],
        [0.1073],
        [0.0913],
        [0.0743],
        [0.0564],
        [0.0380],
        [0.0191],
        [0.0000],
        [-0.0191],
        [-0.0380],
        [-0.0564],
        [-0.0743],
        [-0.0913],
```

```
        [-0.1073],
        [-0.1222],
        [-0.1357],
        [-0.1477],
        [-0.1581],
        [-0.1668],
        [-0.1736],
        [-0.1786],
        [-0.1816],
        [-0.1826],
        [-0.1816],
        [-0.1786],
        [-0.1736],
        [-0.1668],
        [-0.1581],
        [-0.1477],
        [-0.1357],
        [-0.1222],
        [-0.1073],
        [-0.0913],
        [-0.0743],
        [-0.0564],
        [-0.0380],
        [-0.0191],
        [-0.0000],
        [0.0191],
        [0.0380],
        [0.0564],
        [0.0743],
        [0.0913],
        [0.1073],
        [0.1222],
        [0.1357],
        [0.1477],
        [0.1581],
        [0.1668],
        [0.1736],
        [0.1786],
        [0.1816],
        [0.182],
      ])

y_circ = np.array([
        [0.0191],
        [0.0380],
        [0.0564],
        [0.0743],
```

```
        [0.0913],
        [0.1073],
        [0.1222],
        [0.1357],
        [0.1477],
        [0.1581],
        [0.1668],
        [0.1736],
        [0.1786],
        [0.1816],
        [0.1826],
        [0.1816],
        [0.1786],
        [0.1736],
        [0.1668],
        [0.1581],
        [0.1477],
        [0.1357],
        [0.1222],
        [0.1073],
        [0.0913],
        [0.0743],
        [0.0564],
        [0.0380],
        [0.0191],
        [0.0000],
        [-0.0191],
        [-0.0380],
        [-0.0564],
        [-0.0743],
        [-0.0913],
        [-0.1073],
        [-0.1222],
        [-0.1357],
        [-0.1477],
        [-0.1581],
        [-0.1668],
        [-0.1736],
        [-0.1786],
        [-0.1816],
        [-0.1826],
        [-0.1816],
        [-0.1786],
        [-0.1736],
        [-0.1668],
        [-0.1581],
        [-0.1477],
```

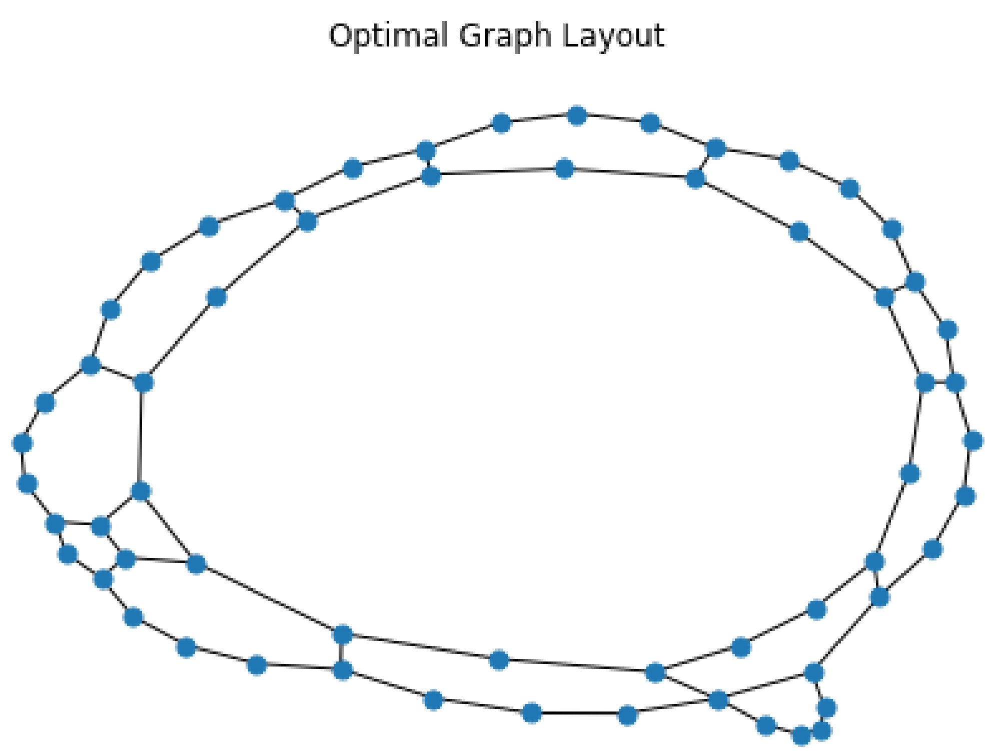
```
        [-0.1357],
        [-0.1222],
        [-0.1073],
        [-0.0913],
        [-0.0743],
        [-0.0564],
        [-0.0380],
        [-0.0191],
        [-0.000],
    ])
    return A, x_circ, y_circ, n, m
```

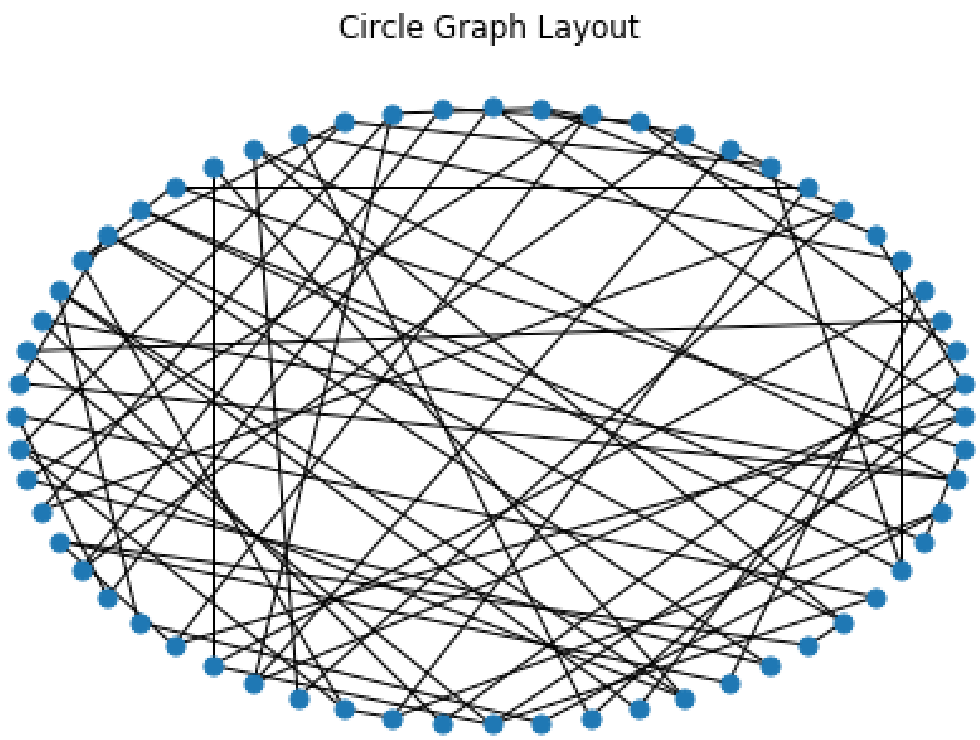
[54]:
```
def solveProblem2():
    A, x_circ, y_circ, n, m = getProblem2Inputs()
    def computeJ(x, y):
        ret = 0
        # Don't double count edges.
        for j in range(n):
            for i in range(j):
                if A[i,j] == 1:
                    ret += ((x[i] - x[j])**2 + (y[i] - y[j])**2)
        return ret
    # Find the degree of node i
    deg = np.sum(A, axis=0)
    B = np.diag(deg)

    # Our symmetric matrix.
    C = B - A

    # Need to find the smallest two eigvalues and vectors.
    # Values is in ascending order.
    values, vectors = np.linalg.eigh(C)

    # Optimal value is sum of smallest two.
    print("J optimal is %s." % (values[1] + values[2]))

    # Compute xopt and yopt
    v1 = vectors[:,1]
    v2 = vectors[:,2]
    optX = v1 - np.mean(v1)
    optY = v2 - np.mean(v2)

    # Verify constraints are met.
    assert np.allclose(np.sum(optX), 0)
    assert np.allclose(np.sum(optY), 0)
    assert np.allclose(np.sum(optX**2), 1)
    assert np.allclose(np.sum(optY**2), 1)
```

```
    assert np.allclose(np.sum(optX * optY), 0)
    print("J optimal is %s." % computeJ(optX, optY))
    print("J for circle is %s" % computeJ(x_circ.flatten(), y_circ.flatten()))

    # Plot the graph and whatnot.
    graph = nx.from_numpy_matrix(A)

    # Plot optimal.
    nx.draw(graph, list(zip(optX, optY)), node_size=50)
    plt.title('Optimal Graph Layout')
    plt.savefig('../hw7/data/optimal_graph')
    plt.show()

    nx.draw(graph, list(zip(x_circ.flatten(), y_circ.flatten())), node_size=50)
    plt.title('Circle Graph Layout')
    plt.savefig('../hw7/data/circle_graph')
```

[55]:
```
solveProblem2()
```

```
J optimal is 0.1072932869526993.
J optimal is 0.10729328695269982.
J for circle is 5.32711444
```





1.2 Problem 4: Simultaneously estimating student ability and exercise difficulty

[56]:
```
def getProblem4Inputs():
    m = 7;
    n = 78;
    G = np.array([
    [20, 20, 20, 12, 20, 15, 20],
    [20, 20, 0, 10, 16, 20, 17],
    [20, 20, 20, 10, 14, 18, 2],
    [20, 0, 0, 20, 20, 8, 2],
    [7, 20, 10, 10, 12, 0, 0],
    [6, 5, 20, 20, 20, 4, 1],
    [8, 20, 15, 10, 20, 7, 0],
    [16, 0, 20, 10, 20, 20, 16],
    [12, 5, 20, 20, 17, 19, 16],
    [17, 20, 20, 20, 20, 13, 20],
    [10, 18, 20, 10, 20, 20, 18],
    [19, 20, 10, 10, 20, 7, 17],
    [10, 0, 10, 10, 5, 13, 2],
    [20, 0, 20, 10, 20, 19, 4],
    [17, 20, 20, 10, 20, 15, 20],
```

```
    [18, 20, 10, 10, 20, 20, 18],
    [20, 20, 20, 15, 18, 0, 0],
    [20, 20, 20, 20, 15, 19, 20],
    [10, 0, 20, 10, 20, 0, 13],
    [0, 0, 20, 10, 20, 0, 1],
    [20, 0, 20, 20, 20, 5, 20],
    [19, 20, 20, 20, 20, 15, 8],
    [20, 20, 20, 20, 20, 4, 6],
    [20, 20, 20, 10, 19, 12, 20],
    [10, 20, 15, 20, 20, 20, 3],
    [17, 20, 0, 20, 18, 15, 10],
    [15, 0, 5, 8, 20, 6, 15],
    [7, 20, 20, 20, 19, 13, 10],
    [9, 0, 20, 20, 20, 9, 12],
    [18, 0, 0, 20, 20, 15, 18],
    [8, 20, 20, 20, 20, 15, 16],
    [16, 20, 20, 20, 20, 20, 15],
    [20, 20, 20, 20, 20, 16, 17],
    [18, 20, 20, 20, 20, 20, 15],
    [13, 0, 15, 20, 20, 20, 16],
    [20, 20, 20, 20, 20, 20, 0],
    [15, 10, 20, 20, 20, 20, 20],
    [6, 0, 20, 20, 20, 20, 5],
    [16, 15, 20, 20, 20, 20, 10],
    [20, 0, 20, 20, 20, 20, 18],
    [20, 20, 20, 10, 20, 20, 19],
    [10, 0, 5, 18, 12, 7, 5],
    [10, 0, 5, 20, 17, 15, 2],
    [16, 20, 0, 10, 17, 20, 17],
    [9, 0, 20, 10, 20, 20, 20],
    [20, 20, 20, 10, 20, 20, 20],
    [20, 20, 20, 10, 20, 7, 20],
    [17, 0, 20, 10, 20, 16, 20],
    [20, 20, 20, 10, 17, 20, 0],
    [10, 15, 20, 20, 18, 8, 0],
    [20, 20, 20, 19, 19, 13, 18],
    [18, 20, 20, 20, 20, 15, 20],
    [17, 0, 20, 20, 20, 20, 0],
    [8, 0, 10, 17, 14, 0, 0],
    [20, 0, 20, 20, 20, 12, 14],
    [15, 0, 5, 20, 14, 0, 0],
    [14, 0, 20, 20, 18, 15, 8],
    [10, 0, 5, 10, 20, 15, 5],
    [19, 20, 5, 20, 20, 15, 20],
    [8, 0, 20, 10, 18, 20, 10],
    [20, 0, 10, 10, 17, 10, 20],
    [12, 0, 10, 10, 20, 10, 6],
```

```
    [16, 20, 20, 20, 20, 20, 20],
    [20, 0, 20, 5, 16, 0, 4],
    [12, 0, 10, 10, 20, 5, 20],
    [20, 20, 20, 20, 20, 15, 19],
    [20, 20, 20, 18, 19, 12, 20],
    [11, 0, 0, 10, 20, 14, 16],
    [10, 0, 20, 18, 20, 15, 8],
    [11, 15, 15, 20, 20, 15, 20],
    [18, 0, 5, 18, 20, 0, 0],
    [12, 0, 20, 20, 20, 0, 0],
    [13, 0, 15, 20, 20, 20, 16],
    [9, 0, 10, 18, 15, 0, 5],
    [20, 20, 15, 20, 20, 20, 16],
    [20, 20, 20, 20, 19, 20, 20],
    [10, 0, 20, 10, 20, 20, 19],
    [15, 0, 10, 10, 18, 0, 5]
    ]).T
    return G, n, m
```

[114]:
```
def solveProblem4():
    G,n,m = getProblem4Inputs()
    # Skinny SVD.
    U, S, VT = np.linalg.svd(G, full_matrices=False)
    sigma = S[0]
    u, v = U[:, 0], VT[0,:]

    # Compute difficulties.
    d = m / (np.sum(1 / u) * u)
    with np.printoptions(formatter={'float': '{: 0.3f}'.format}):
        print("The difficulties are:")
        print(d)
    a = m * sigma / np.sum(1 / u) * v

    # Compute optimal value J
    d.shape = (d.shape[0], 1)
    a.shape = (a.shape[0], 1)
    Jopt = 1 / np.sqrt(m*n) * np.linalg.norm(G - np.dot(1 / d, a.T))
    print("The optimal value achieved is $J_{\\text{opt}} = %s$." % (Jopt))

    rmse = 1 / np.sqrt(m*n) * np.linalg.norm(G)
    print("The ratio of $J_{\\text{opt}}$ and the RMSE of $G$ is %s." % (Jopt /␣
 ↪rmse))
```

[115]:
```
solveProblem4()
```

```
The difficulties are:
[ 0.943  1.278  0.902  0.920  0.773  1.042  1.143]
```

The optimal value achieved is $J_{\text{opt}} = 5.675923069899214$.
The ratio of J_{opt} and the RMSE of G is 0.35742617468116206.

1.3 Problem 8: Sensor selection and observer design

[116]:
```
A = np.array([
    [1, 0, 0, 0],
    [1, 1, 0, 0],
    [0, 1, 1, 0],
    [1, 0, 0, 0]
])
C = np.array([
    [1, 1, 0, 0],
    [0, 1, 1, 0],
    [0, 0, 0, 1]
])
```

[116]: 3

[117]:
```
np.linalg.matrix_rank(A), np.linalg.matrix_rank(C)
```

[117]: (3, 3)

[120]:
```
# Since C by itself is only rank 3, we can't reconstruct.
O = np.vstack((C, np.dot(C,A)))
```

[122]:
```
# Using just the first derivative is enough. We now have rank 4.
np.linalg.matrix_rank(O)
```

[122]: 4

[125]:
```
# We need to find the left inverse of the matrix. The matrix is skinny and
# full rank, so we have.
F = np.dot(np.linalg.inv(np.dot(O.T, O)), O.T)
```

[128]:
```
# Verify left inverse.
assert np.allclose(np.dot(F, O), np.identity(4))
```

[130]:
```
# But not right inverse.
assert not np.allclose(np.dot(O, F), np.identity(6))
```

[137]:
```
# Split into Fk
F0 = F[:, :3]
F1 = F[:, 3:]
```

[138]:
```
F0
```

[138]:
```
array([[-0.25 ,  0.125,  0.   ],
       [ 0.75 , -0.375,  0.   ],
       [-1.   ,  1.   ,  0.   ],
       [ 0.   ,  0.   ,  1.   ]])
```

[139]:
```
F1
```

[139]:
```
array([[ 3.75000000e-01, -1.25000000e-01,  6.25000000e-01],
       [-1.25000000e-01,  3.75000000e-01, -8.75000000e-01],
       [-6.66133815e-16,  4.44089210e-16,  1.00000000e+00],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00]])
```

[ ]: