In [5]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import Normalizer
```

In [9]:

```
# Circle topology

# Unweighted adjacency matrix
Atilde = np.zeros((8,8), dtype=int)
Atilde = np.array([
    [0, 1, 0, 0, 0, 0, 0, 1],
    [1, 0, 1, 0, 0, 0, 0, 0],
    [1, 1, 0, 1, 1, 0, 0, 0],
    [0, 0, 1, 0, 1, 0, 0, 0],
    [0, 0, 0, 1, 0, 1, 0, 0],
    [0, 0, 0, 0, 1, 0, 1, 0],
    [0, 0, 0, 0, 0, 1, 0, 1],
    [1, 0, 0, 0, 0, 0, 1, 0],
])
print('Unweighted adjacency matrix')
print(Atilde)

# find weighted adjacency matrix
print('Weighted adjacency matrix')
A = Atilde / np.sum(Atilde, 0)
print(A)
```

```
Unweighted adjacency matrix
[[0 1 0 0 0 0 0 1]
 [1 0 1 0 0 0 0 0]
 [1 1 0 1 1 0 0 0]
 [0 0 1 0 1 0 0 0]
 [0 0 0 1 0 1 0 0]
 [0 0 0 0 1 0 1 0]
 [0 0 0 0 0 1 0 1]
 [1 0 0 0 0 0 1 0]]
Weighted adjacency matrix
[[0.         0.5        0.         0.         0.         0.
  0.         0.5       ]
 [0.33333333 0.         0.5        0.         0.         0.
  0.         0.        ]
 [0.33333333 0.5        0.         0.5        0.33333333 0.
  0.         0.        ]
 [0.         0.         0.5        0.         0.33333333 0.
  0.         0.        ]
 [0.         0.         0.         0.5        0.         0.5
  0.         0.        ]
 [0.         0.         0.         0.         0.33333333 0.
  0.5        0.        ]
 [0.         0.         0.         0.         0.         0.5
  0.         0.5       ]
 [0.33333333 0.         0.         0.         0.         0.
  0.5        0.        ]]
```

In [15]:

```
# Power method

b0 = 0.125*np.ones((8,1))
print('b0 = ', b0)

b1 = A @ b0
print('b1 = ', b1)

b = b0.copy()
for k in range(1000):
    b = A @ b

print('1000 iterations')
print('b = ',b)
```

```
b0 =  [[0.125]
 [0.125]
 [0.125]
 [0.125]
 [0.125]
 [0.125]
 [0.125]
 [0.125]]
b1 =  [[0.125     ]
 [0.10416667]
 [0.20833333]
 [0.10416667]
 [0.125     ]
 [0.10416667]
 [0.125     ]
 [0.10416667]]
1000 iterations
b =  [[0.11538462]
 [0.15384615]
 [0.23076923]
 [0.15384615]
 [0.11538462]
 [0.07692308]
 [0.07692308]
 [0.07692308]]
```

Node 3 is most important, because after 1000 iterations, node 3 has the highest probibility in the distribution.

2

In [24]:

```
# Hub topology

Atildehub = np.zeros((9,9), dtype=int)
Atildehub = np.array([
   # 1,2,3,4,5,6,7,8,9
    [0,0,0,0,0,0,0,0,1],
    [1,0,0,0,0,0,0,0,1],
    [0,0,0,0,0,0,0,0,1],
    [0,0,0,0,0,0,0,0,1],
    [0,0,0,0,0,0,0,0,1],
    [0,0,0,0,0,0,0,0,1],
    [0,0,0,0,0,0,0,0,1],
    [0,0,0,0,0,0,0,0,1],
    [1,1,1,1,1,1,1,1,0],
])

print('Unweighted adjacency matrix')
print(Atildehub)

# find weighted adjacency matrix
Ahub = Atildehub / np.sum(Atildehub, 0)
print('Weighted adjacency matrix')
print(Ahub)
```

```
Unweighted adjacency matrix
[[0 0 0 0 0 0 0 0 1]
 [1 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 1]
 [1 1 1 1 1 1 1 1 0]]
Weighted adjacency matrix
[[0.    0.    0.    0.    0.    0.    0.    0.    0.125]
 [0.5   0.    0.    0.    0.    0.    0.    0.    0.125]
 [0.    0.    0.    0.    0.    0.    0.    0.    0.125]
 [0.    0.    0.    0.    0.    0.    0.    0.    0.125]
 [0.    0.    0.    0.    0.    0.    0.    0.    0.125]
 [0.    0.    0.    0.    0.    0.    0.    0.    0.125]
 [0.    0.    0.    0.    0.    0.    0.    0.    0.125]
 [0.    0.    0.    0.    0.    0.    0.    0.    0.125]
 [0.5   1.    1.    1.    1.    1.    1.    1.    0.   ]]
```

In [43]:

```
b0 = (1/9)*np.ones((9,1))
print('b0 = ', b0)

bhub1 = Ahub @ b0
print('bhub1 = ', bhub1)
print('bhub1 = ', bhub1)

bhub = b0.copy()
for k in range(1000):
    bhub = Ahub @ bhub

print('1000 iterations')
print('bhub = ', bhub)

bhubr = b0.copy()
for k in range(93):
    bhubr = Ahub @ bhubr

print('93 iterations')
print('bhubr = ',bhubr)
```

```
b0 =  [[0.11111111]
 [0.11111111]
 [0.11111111]
 [0.11111111]
 [0.11111111]
 [0.11111111]
 [0.11111111]
 [0.11111111]
 [0.11111111]]
bhub1 =  [[0.01388889]
 [0.06944444]
 [0.01388889]
 [0.01388889]
 [0.01388889]
 [0.01388889]
 [0.01388889]
 [0.01388889]
 [0.83333333]]
1000 iterations
bhub =  [[0.06060606]
 [0.09090909]
 [0.06060606]
 [0.06060606]
 [0.06060606]
 [0.06060606]
 [0.06060606]
 [0.06060606]
 [0.48484848]]
93 iterations
bhubr =  [[0.06052684]
 [0.09087232]
 [0.06052684]
 [0.06052684]
 [0.06052684]
 [0.06052684]
 [0.06052684]
 [0.06052684]
 [0.4854398 ]]
```

In [ ]: