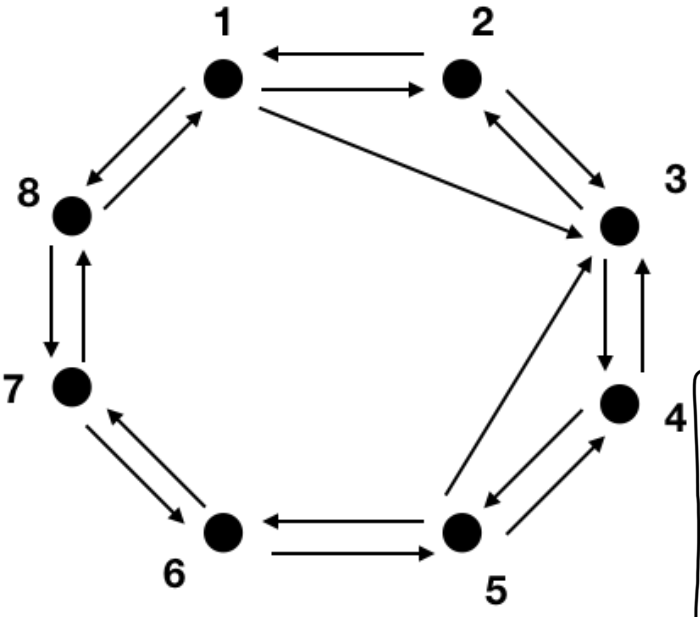
CS/ECE/ME532 Activity 14

Estimated time: 20 *minutes for Q1,* 20 *minutes for Q2 and* 10 *minutes for Q3.*

1. A ring-like network of links between web pages is shown below.



a). $\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

b). $\begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{3} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix}$

a) Find the unweighted adjacency matrix for this network.

b) Find the weighted adjacency matrix for this network. Note that the entries in each column of the weighted adjacency matrix are nonnegative and sum to one. Such a matrix is called a column stochastic matrix.

c) Suppose the entries of a vector $\boldsymbol{b}$ sum to one. It is easy to show that the entries of $\boldsymbol{Ab}$ also sum to one since each column of the weighted adjacency matrix $\boldsymbol{A}$ sums to one. The PageRank algorithm thus uses the power method without normalizing the length of the vector at each iteration. Each iteration gives a new vector with positive entries that sum to one. Find the estimate of the PageRank vector after one iteration using an initial vector $\boldsymbol{b}_0 = \frac{1}{8}\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$. The initial vector $\boldsymbol{b}_0$ gives equal importance to all pages.

$$\begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{3} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{8} \\ \frac{1}{8} \\ \frac{1}{8} \\ \frac{1}{8} \\ \frac{1}{8} \\ \frac{1}{8} \\ \frac{1}{8} \\ \frac{1}{8} \end{bmatrix} = \begin{bmatrix} \frac{1}{16} + \frac{1}{16} \\ \frac{1}{24} + \frac{1}{16} \\ \frac{1}{24} + \frac{1}{16} + \frac{1}{16} + \frac{1}{24} \\ \frac{1}{16} + \frac{1}{24} \\ \frac{1}{16} + \frac{1}{16} \\ \frac{1}{24} + \frac{1}{16} \\ \frac{1}{16} + \frac{1}{16} \\ \frac{1}{24} + \frac{1}{16} \end{bmatrix} = \begin{bmatrix} \frac{1}{8} \\ \frac{5}{48} \\ \frac{5}{24} \\ \frac{5}{48} \\ \frac{1}{8} \\ \frac{5}{48} \\ \frac{1}{8} \\ \frac{5}{48} \end{bmatrix}$$

d). $\begin{bmatrix} 0.1154 \\ 0.1538 \\ 0.2308 \\ 0.1538 \\ 0.1154 \\ 0.0769 \\ 0.0769 \\ 0.0769 \end{bmatrix}$

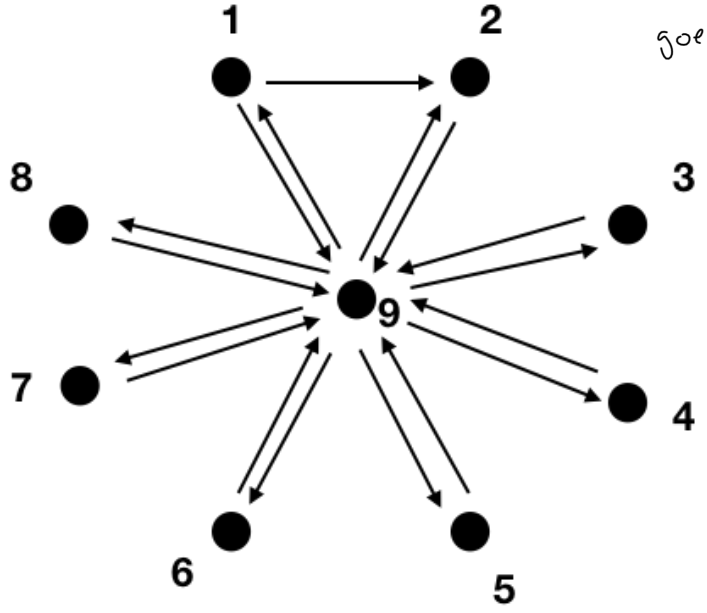
d) Find the estimate of the PageRank vector after 1000 iterations of the power method using an initial vector $\boldsymbol{b}_0 = \frac{1}{8}\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$. A skeleton script is available. You will need to enter the adjacency matrix into the code.

e) Do any nodes seem to be more important than other nodes? Explain.

Node 3 has the highest probability of reaching it, so it appears to be the most important.

The probability ranking goes: 3
2 4
1 5
6 7 8

2. A hub-like network of links between web pages is shown below.



a). $\begin{bmatrix} 0&0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&1 \\ 1&1&1&1&1&1&1&1&0 \end{bmatrix}$

b). $\begin{bmatrix} 0&0&0&0&0&0&0&0&\frac{1}{8} \\ \frac{1}{2}&0&0&0&0&0&0&0&\frac{1}{8} \\ 0&0&0&0&0&0&0&0&\frac{1}{8} \\ 0&0&0&0&0&0&0&0&\frac{1}{8} \\ 0&0&0&0&0&0&0&0&\frac{1}{8} \\ 0&0&0&0&0&0&0&0&\frac{1}{8} \\ 0&0&0&0&0&0&0&0&\frac{1}{8} \\ 0&0&0&0&0&0&0&0&\frac{1}{8} \\ \frac{1}{2}&1&1&1&1&1&1&1&0 \end{bmatrix}$

a) Find the unweighted adjacency matrix for this network.

b) Find the weighted adjacency matrix for this network.

c). $\begin{bmatrix} \frac{1}{72} \\ \frac{5}{72} \\ \frac{1}{72} \\ \frac{1}{72} \\ \frac{1}{72} \\ \frac{1}{72} \\ \frac{1}{72} \\ \frac{1}{72} \\ \frac{5}{6} \end{bmatrix}$

c) Find the estimate of the PageRank vector after one iteration using an initial vector $\boldsymbol{b}_0 = \frac{1}{9}\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$.

d) Find the estimate of the PageRank vector after 1000 iterations of the power

method using an initial vector $\boldsymbol{b}_0 = \frac{1}{9}\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$. $\begin{bmatrix} 0.0606 \\ 0.0909 \\ 0.0606 \\ 0.0606 \\ 0.0606 \\ 0.0606 \\ 0.0606 \\ 0.0606 \\ 0.0606 \\ 0.4848 \end{bmatrix}$

e) Are any nodes more important than other nodes? Explain.
There is the highest probability to reach page 9. Then node 2, and finally all of the rest.

f) Experiment with the number of iterations of the power method that are needed to find an answer that is correct to three decimal places.
90 iterations will give an answer correct to 3 decimal places without rounding, and 110 iterations gives an answer correct with rounding.

3. Consider expressing the SVD of a rank-r matrix $\boldsymbol{X}$ as

$$\boldsymbol{X} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$$

where σ_i is the ith singular value with left singular vector $\boldsymbol{u}_i$ and right singular vector $\boldsymbol{v}_i$. Is the sign of the singular vectors unique? Why or why not? *Hint:* Consider replacing $\boldsymbol{u}_1$ with $\tilde{\boldsymbol{u}}_1 = -\boldsymbol{u}_1$.

If $\tilde{u}_1 = -u_1$, then the sign of v_1^T would also have to be changed so that the resulting matrix isn't negative.
Since you can get the same matrix as long as you change both signs of a pair of u_i and v_i^T vectors, the sign is not unique.

```
import numpy as np
import matplotlib.pyplot as plt
```

1a)

```
# Circle topology
# Unweighted adjacency matrix

# Option 1: Manually enter the entries
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]])

# Option 2: or you can exploit the patterns
# Atilde = np.zeros((8,8))
# for i in range(8): #
#     Atilde[i,(i+1)%8] = 1
#     Atilde[i,(i-1)%8] = 1
# Atilde[2,0] = 1
# Atilde[2,4] = 1

print('Unweighted adjacency matrix')
print(Atilde)
print(' ')
```

```
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]]
```

1b)

```
# Find weighted adjacency matrix
# option 1: normalize columns with a for loop
A = np.zeros((8,8), dtype=float)
for k in range(8):
    norm = np.sum(Atilde[:,k])
```

```
    A[:,k] = Atilde[:,k]/norm

# option 2: normalize using numpy.sum() and broadcasting, in a single line
A = Atilde / Atilde.sum(axis=0)

print('Weighted adjacency matrix')
print(A)
```

```
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.        ]]
```

1c) and 1d)

```
# Power method
print(A)
print(' ')

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

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

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

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

```
[[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.        ]]

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]]
```

1e) Explanation goes here.

Node 3 has the highest probability of reaching it after the 1000 iterations of the power method, so it appears to be the most important. The probability ranking goes: Node 3, Node 2 and 4, Node 1 and 5, Node 6, 7, 8.

2a)

```
# Hub topology

Atildehub = np.array(
         [[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)
print(' ')
```

```
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]]
```

2b)

```
# find weighted adjacency matrix

Ahub = Atildehub / Atildehub.sum(axis=0)

print('Weighted adjacency matrix')
print(Ahub)
```

```
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.   ]]
```

2c) and 2d)

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

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

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

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

bhubr = b0.copy()
for k in range(90):
    bhubr = Ahub@bhubr

print('90 iterations')
print('bhubr = ',bhubr)

bhubr2 = b0.copy()
for k in range(110):
    bhubr2 = Ahub@bhubr2

print('90 iterations')
print('bhubr = ',bhubr2)
```

```
b0 =  [[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]
 [0.83333333]]

1000 iterations
bhub =  [[0.06060606]
 [0.09090909]
 [0.06060606]
 [0.06060606]
 [0.06060606]
 [0.06060606]
 [0.06060606]
 [0.48484848]]
```

```
90 iterations
bhubr =  [[0.0607036 ]
 [0.09095436]
 [0.0607036 ]
 [0.0607036 ]
 [0.0607036 ]
 [0.0607036 ]
 [0.0607036 ]
 [0.0607036 ]
 [0.48412044]]
90 iterations
bhubr =  [[0.06063043]
 [0.0909204 ]
 [0.06063043]
 [0.06063043]
 [0.06063043]
 [0.06063043]
 [0.06063043]
 [0.06063043]
 [0.48466655]]
```

Complete 2e and 2f below.

2e) There is the highest probability to reach page 9, so that is the most important page. Then the next highest probability is for page 2, and all other pages are equal probability after it.

2f) 90 iterations will give an answer correct to 3 decimal places without rounding, while 110 iterations will give an aswer correct to 3 decimal places with rounding.