Coding

[1]:
```
import numpy as np
```

[3]:
```
W = np.array([
    [0, 1, 1, 0, 0, 0],
    [0, 0, 1, 0, 0, 0],
    [1, 1, 0, 0, 1, 1],
    [0, 0, 0, 0, 1, 1],
    [0, 0, 0, 1, 0, 1],
    [0, 0, 1, 1, 0, 0],
])

D = np.array([
    [2, 0, 0, 0, 0, 0],
    [0, 1, 0, 0, 0, 0],
    [0, 0, 4, 0, 0, 0],
    [0, 0, 0, 2, 0, 0],
    [0, 0, 0, 0, 2, 0],
    [0, 0, 0, 0, 0, 2],
])

eta = 0.85
N = W.shape[0]
E = np.ones((N, N))

# Computing D inverse
D_inv = np.linalg.inv(D)

# Compute P
P = ((1 - eta) / N) * E + eta * np.dot(D_inv, W)
P
```

[3]:
```
array([[0.025 , 0.45  , 0.45  , 0.025 , 0.025 , 0.025 ],
       [0.025 , 0.025 , 0.875 , 0.025 , 0.025 , 0.025 ],
       [0.2375, 0.2375, 0.025 , 0.025 , 0.2375, 0.2375],
       [0.025 , 0.025 , 0.025 , 0.025 , 0.45  , 0.45  ],
       [0.025 , 0.025 , 0.025 , 0.45  , 0.025 , 0.45  ],
       [0.025 , 0.025 , 0.45  , 0.45  , 0.025 , 0.025 ]])
```

[22]:
```python
# Compute the eigenvectors(left) and eigenvalues
eigenvalues, eigenvectors = np.linalg.eig(P.T)

# Find the index
eigenvector_index = np.argmin(np.abs(eigenvalues - 1))

# Extract the PageRank vector
pi = eigenvectors[:, eigenvector_index].real

# Normalize
pi = pi / np.sum(pi)

pi
```

[22]:
```
array([0.07735886, 0.11023638, 0.24639464, 0.18635389, 0.15655927,
       0.22309696])
```

[23]:
```python
# Initialize uniformly
pi_t = np.ones(N) / N

threshold = 1e-6
delta = 1.0

# Iteratively compute pi^(t) until convergence
while delta > threshold:
    pi_t_new = np.dot(pi_t, P)
    delta = np.linalg.norm(pi_t_new - pi_t, 1)  # L1 norm
    pi_t = pi_t_new


pi_t
```

[23]:
```
array([0.07735895, 0.11023655, 0.24639486, 0.18635365, 0.15655919,
       0.22309679])
```