## PageRank

[1]
```
# Before we begin, let's load the libraries.
%pylab notebook
import numpy as np
import numpy.linalg as la
np.set_printoptions(suppress=True)
```

```
Populating the interactive namespace from numpy and matplotlib
```

[2]
```
L = np.array([[0,   1/2, 1/3, 0, 0,   0 ],
              [1/3, 0,   0,   0, 1/2, 0 ],
              [1/3, 1/2, 0,   1, 0,   1/2 ],
              [1/3, 0,   1/3, 0, 1/2, 1/2 ],
              [0,   0,   0,   0, 0,   0 ],
              [0,   0,   1/3, 0, 0,   0 ]])
```

[3]
```
eVals, eVecs = la.eig(L) # Gets the eigenvalues and vectors
order = np.absolute(eVals).argsort()[::-1] # Orders them by their eigenvalues
eVals = eVals[order]
eVecs = eVecs[:,order]

r = eVecs[:, 0] # Sets r to be the principal eigenvector
100 * np.real(r / np.sum(r)) # Make this eigenvector sum to one, then multiply by 100 Procrastinating Pats
```

```
array([16.        ,  5.33333333, 40.        , 25.33333333,  0.        ,
       13.33333333])
```

[4]
```
r = 100 * np.ones(6) / 6 # Sets up this vector (6 entries of 1/6 × 100 each)
r # Shows it's value
```

```
[5] for i in np.arange(100) : # Repeat 100 times
        r = L @ r
    r
```

```
array([16.        ,  5.33333333, 40.        , 25.33333333,  0.        ,
       13.33333333])
```

```
[6] r = 100 * np.ones(6) / 6 # Sets up this vector (6 entries of 1/6 × 100 each)
    lastR = r
    r = L @ r
    i = 0
    while la.norm(lastR - r) > 0.01 :
        lastR = r
        r = L @ r
        i += 1
    print(str(i) + " iterations to convergence.")
    r
```

```
18 iterations to convergence.
array([16.00149917,  5.33252025, 39.99916911, 25.3324738 ,  0.        ,
       13.33433767])
```

```
[9] d = 0.5 # Feel free to play with this parameter after running the code once.
    M = d * L2 + (1-d)/7 * np.ones([7, 7]) # np.ones() is the J matrix, with ones for each entry.
```

```
[13]
    # Use the following function to generate internets of different sizes.
    generate_internet(5)
```

```
array([[1. , 0.2, 0.2, 0.2, 0.2],
       [0. , 0.2, 0.2, 0.2, 0.2],
       [0. , 0.2, 0.2, 0.2, 0.2],
       [0. , 0.2, 0.2, 0.2, 0.2],
       [0. , 0.2, 0.2, 0.2, 0.2]])
```

```
[14] pageRank(L, 1)
```

```
array([16.00149917,  5.33252025, 39.99916911, 25.3324738 ,  0.        ,
       13.33433767])
```

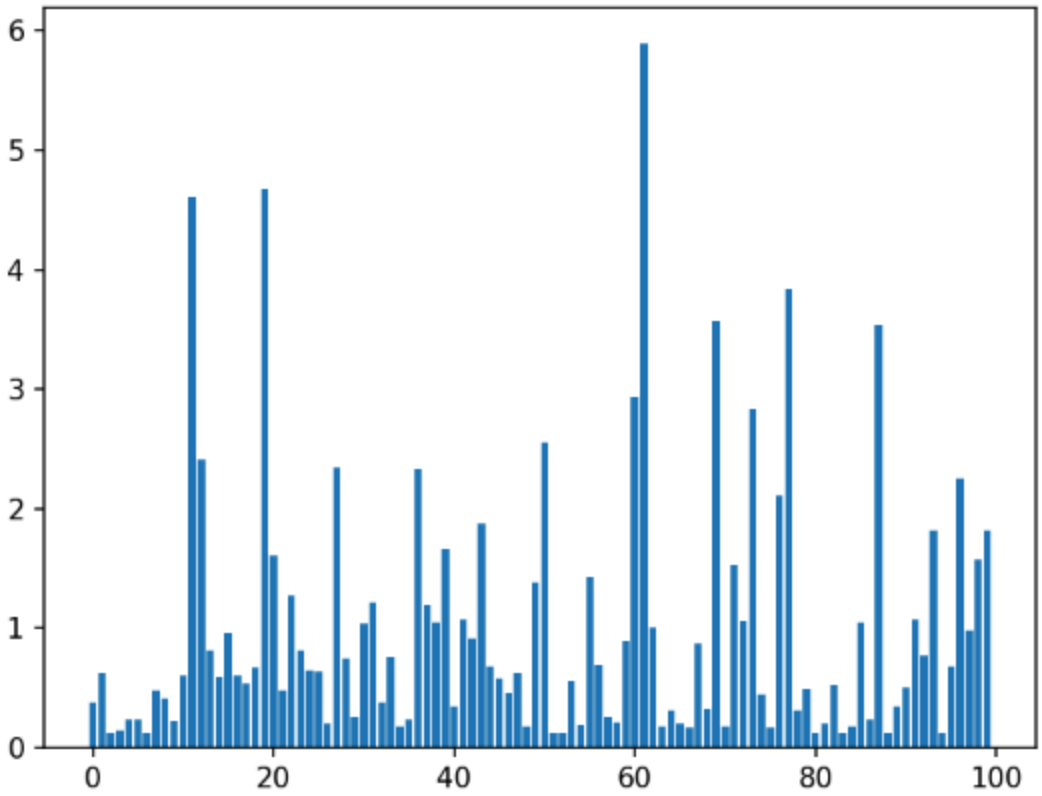
[15]
```
# Do note, this is calculating the eigenvalues of the link matrix, L,
# without any damping. It may give different results that your pageRank function.
# If you wish, you could modify this cell to include damping.
eVals, eVecs = la.eig(L) # Gets the eigenvalues and vectors
order = np.absolute(eVals).argsort()[::-1] # Orders them by their eigenvalues
eVals = eVals[order]
eVecs = eVecs[:,order]

r = eVecs[:, 0]
100 * np.real(r / np.sum(r))
```

```
array([16.        ,  5.33333333, 40.        , 25.33333333,  0.        ,
       13.33333333])
```

[17]
```
# You may wish to view the PageRank graphically.
# This code will draw a bar chart, for each (numbered) website on the generated internet,
# The height of each bar will be the score in the PageRank.
# Run this code to see the PageRank for each internet you generate.
# Hopefully you should see what you might expect
# - there are a few clusters of important websites, but most on the internet are rubbish!
%pylab notebook
r = pageRank(generate_internet(100), 0.9)
plt.bar(arange(r.shape[0]), r);
```

```
Populating the interactive namespace from numpy and matplotlib
```


0
1
2
3
4
5
6
0
20
40
60
80
100