PageRank

Import libs

In [1]:
```
import numpy as np
import pandas as pd
```

Using transation matrix

In [2]:
```
def PageRank1( M, dmp = 0.85, eps = 1.0e-5, max_iter = 100):
    """
        M       : transation matrix
        dmp     : Damping factor ; usually set to 0.85
        eps     : Pre-specified threshold (desired precision); //used in Stopping condition
        max_iter : Maximum number of iterations
    """
    nb_site = len (M)
    R_old = [1/nb_site for _ in range(nb_site)]
    cpp = 0
    while True:
        R = M.dot(R_old)
        R = [ (1-dmp)/nb_site + dmp*r for r in R]

        flag = True
        for r1, r2 in zip(R, R_old):
            if abs(r1 - r2) > eps:
                flag = False
                break

        if flag or cpp >= max_iter:
            break

        R_old = R
        cpp+=1

    return cpp, R
```

In [3]:
```
A = np.array([[   0,  0 , 1, 1/2],
              [ 1/3,   0, 0,   0],
              [ 1/3, 1/2, 0, 1/2],
              [ 1/3, 1/2, 0,   0]])
eps = 0.00001
nb_etr = 100
Steps = PageRank1(A, eps = eps, max_iter = nb_etr)[0]
PRvector = PageRank1(A, eps = eps, max_iter = nb_etr)[1]
print('Steps :', Steps)
print('PageRank vector :', PRvector)
```

```
Steps : 12
PageRank vector : [0.3681575511754208, 0.14180903446209795, 0.28795884393472515, 0.2020745702688743]
```

In [4]:
```
B = np.array([
    [1/11, 1/11, 1/11, 1/11, 1/11, 1/11, 1/11, 1/11, 1/11, 1/11, 1/11],
    [0.15/11, 0.15/11, (0.15/11)+0.85, 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11],
    [0.15/11, (0.15/11)+0.85, 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11],
    [(0.15/11)+(0.85/2), (0.15/11)+(0.85/2), 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11,
    [0.15/11, (0.15/11)+(0.85/3), 0.15/11, (0.15/11)+(0.85/3), 0.15/11, (0.15/11)+(0.85/3), 0.15/11, 0.15/11, 0.15/11,
    [0.15/11, (0.15/11)+(0.85/2), 0.15/11, 0.15/11, (0.15/11)+(0.85/2), 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11,
    [0.15/11, (0.15/11)+(0.85/2), 0.15/11, 0.15/11, (0.15/11)+(0.85/2), 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11,
    [0.15/11, (0.15/11)+(0.85/2), 0.15/11, 0.15/11, (0.15/11)+(0.85/2), 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11,
    [0.15/11, 0.15/11, 0.15/11, 0.15/11, (0.15/11)+(0.85/1), 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11,
    [0.15/11, 0.15/11, 0.15/11, 0.15/11, (0.15/11)+(0.85/1), 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11, 0.15/11,
])
print(B)
print(PageRank1(B))
```

```
[[0.09090909 0.09090909 0.09090909 0.09090909 0.09090909 0.09090909
  0.09090909 0.09090909 0.09090909 0.09090909 0.09090909]
 [0.01363636 0.01363636 0.86363636 0.01363636 0.01363636 0.01363636
  0.01363636 0.01363636 0.01363636 0.01363636 0.01363636]
 [0.01363636 0.86363636 0.01363636 0.01363636 0.01363636 0.01363636
  0.01363636 0.01363636 0.01363636 0.01363636 0.01363636]
 [0.43863636 0.43863636 0.01363636 0.01363636 0.01363636 0.01363636
  0.01363636 0.01363636 0.01363636 0.01363636 0.01363636]
 [0.01363636 0.2969697  0.01363636 0.2969697  0.01363636 0.2969697
  0.01363636 0.01363636 0.01363636 0.01363636 0.01363636]
 [0.01363636 0.43863636 0.01363636 0.01363636 0.43863636 0.01363636
  0.01363636 0.01363636 0.01363636 0.01363636 0.01363636]
 [0.01363636 0.43863636 0.01363636 0.01363636 0.43863636 0.01363636
  0.01363636 0.01363636 0.01363636 0.01363636 0.01363636]
 [0.01363636 0.43863636 0.01363636 0.01363636 0.43863636 0.01363636
  0.01363636 0.01363636 0.01363636 0.01363636 0.01363636]
 [0.01363636 0.01363636 0.01363636 0.01363636 0.86363636 0.01363636
  0.01363636 0.01363636 0.01363636 0.01363636 0.01363636]]
(0, [0.09090909090909091, 0.09090909090909094, 0.09090909090909093, 0.09090909090909088, 0.09090909090909091, 0.09090909090909088, 0.09090909090909088, 0.09090909090909088, 0.09090909090909088, 0.09090909090909091, 0.09090909090909091])
```

In [5]:
```
T = np.array([
        [ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ],
        [ 0 , 0 , 0.85/1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ],
        [ 0 , 0.85/1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ],
        [ 0.85/2 , 0.85/2 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ],
        [ 0 , 0.85/3 , 0 , 0.85/3 , 0 , 0.85/3 , 0 , 0 , 0 , 0 , 0 ],
        [ 0 , 0.85/2 , 0 , 0 , 0.85/2 , 0 , 0 , 0 , 0 , 0 , 0 ],
        [ 0 , 0.85/2 , 0 , 0 , 0.85/2 , 0 , 0 , 0 , 0 , 0 , 0 ],
        [ 0 , 0.85/2 , 0 , 0 , 0.85/2 , 0 , 0 , 0 , 0 , 0 , 0 ],
        [ 0 , 0.85/2 , 0 , 0 , 0.85/2 , 0 , 0 , 0 , 0 , 0 , 0 ],
        [ 0 , 0 , 0 , 0 , 0.85/1 , 0 , 0 , 0 , 0 , 0 , 0 ],
        [ 0 , 0 , 0 , 0 , 0.85/1 , 0 , 0 , 0 , 0 , 0 , 0 ]
    ])
```

In [6]:
```
print('Steps :',PageRank1(T)[0])
print('PageRank vector :',PageRank1(T)[1])
```

```
Steps : 22
PageRank vector : [0.013636363636363644, 0.04916371515284579, 0.04916371515284579, 0.036326175758241076, 0.04577578244757678, 0.047939167732744, 0.047939167732744, 0.047939167732744, 0.047939167732744, 0.04671462030970301, 0.04671462030970301]
```

In [7]:
```
pages = ['A' , 'B' , 'C' , 'D' , 'E' , 'F' , 'G' , 'H' , 'I' , 'L' , 'M']
frame = pd.DataFrame(list(zip(pages , PageRank1(T)[1] )),columns=['pages','Vector'])
frame
```

Out[7]:

	pages	Vector
0	A	0.013636
1	B	0.049164
2	C	0.049164
3	D	0.036326
4	E	0.045776
5	F	0.047939
6	G	0.047939
7	H	0.047939
8	I	0.047939
9	L	0.046715
10	M	0.046715

In [ ]:
```

```

Using xml graph

In [8]:
```
import xml.etree.ElementTree as ET
```

In [9]:
```
def pagerank2( G, dmp = 0.85, eps = 1.0e-5, max_iter = 100):
    """
        PageRank computes a ranking of the nodes in the graph G based on
        the structure of the incoming links. It was originally designed as
        an algorithm to rank web pages.

        Parameters
        ----------
        G           : xml path of graph
        M           : transation matrix; The adjacency matrix of the web graph
        dmp         : damping factor ; usually set to 0.85
        eps         : Pre-specified threshold (desired precision); used in Stopping condition
        max_iter    : Maximum number of iterations              ; used in Stopping condition

        Returns
        -------
        pagerank : dictionary; Dictionary of nodes with PageRank as value
    """
    graph = ET.parse(G).getroot()
    nb_site = len(graph)

    # Transform graph from xml into dictionary
    link_dict ={}
    for i,node in enumerate(graph):
        link_dict[node.get("link")] =[ i, [link.get('value') for link in node] ]
    print(link_dict)
    print('#####################################')

    # dictionary 2 translation matrix
    M = [[0 for _ in range(nb_site)] for _ in range(nb_site)] # transation matrix
    for i,node in enumerate(link_dict):
        for link in link_dict[node][1]:
            M[link_dict[link][0]][link_dict[node][0]] = 1/len(link_dict[node][1])
    M = np.array(M)
    print('M:')
    print(pd.DataFrame(M))
    print('#####################################')

    R_old = [1/nb_site for _ in range(nb_site)]
    cpp = 0
    while True:
        R = M.dot(R_old)
        R = [ (1-dmp)/nb_site + dmp*r for r in R]

        flag = True
        for r1, r2 in zip(R, R_old):
            if abs(r1 - r2) > eps:
                flag = False
                break

        if flag or cpp >= max_iter: # Stopping condition
            break

        R_old = R   # update
        cpp+=1

    print('Steps :',cpp)
    print('#####################################')
    return {'id':list(range(nb_site)), 'link': list(link_dict), 'rank': R}
```

In [10]:
```
dict1 = pagerank2('graph1.xml')
df = pd.DataFrame(dict1)
df.sort_values(by=['rank'], ascending=False, inplace=True)
df.reset_index(drop=True,inplace=True)
```

```
{'page1': [0, ['page2', 'page3', 'page4']], 'page2': [1, ['page3', 'page4']], 'page3': [2, ['page1']], 'page4': [3, ['page3', 'page1']]}
#####################################
M:
          0    1    2    3
0  0.000000  0.0  1.0  0.5
1  0.333333  0.0  0.0  0.0
2  0.333333  0.5  0.0  0.5
3  0.333333  0.5  0.0  0.0
#####################################
Steps : 12
#####################################
```

In [11]:
```
df
```

Out[11]:

	id	link	rank
0	0	page1	0.368158
1	2	page3	0.287959
2	3	page4	0.202075
3	1	page2	0.141809

In [ ]:
```

```