Academic Year: 2024-25 **Semester: V**
Class/Branch: T.E. DS **Subject: DWMLab**

EXPERIMENT NO. 10

1. **Aim:** Implementation of Page Rank algorithm in python.

2. **Objectives:** From this experiment, the student will be able to

- Learn about the Page Rank algorithm.
- Learn about web structure mining.

3. Theory:

PageRank Algorithm:

PageRank is an algorithm developed by Google founders Larry Page and Sergey Brin that measures the relevance or importance of web pages on the Internet. PageRank is a page ranking algorithm commonly used in web structure mining.

The PageRank algorithm treats the web as a vast network of interconnected pages. Each page is represented on the web as a node with links between pages at the edges. The basic principle of PageRank is that a page is considered more important if other vital pages link it. The algorithm determines the initial PageRank value for each web page. This initial value can be uniform or based on certain factors, such as the number of incoming links to the page. The algorithm then repeatedly calculates the PageRank value of each page, taking into account the PageRank value of the pages that are related to the pages. During each iteration, the PageRank value of the page is updated based on the sum of the PageRank values of the incoming links. Pages with more inbound links have a more significant impact on the landing page's PageRank.
Additionally, a page's PageRank is divided by its number of outbound links, which divides its influence on the pages it links to. The iterative process continues until the PageRank values converge, indicating that the algorithm has reached a stable rank.

The resulting PageRank scores describe the relative importance of each web page. Pages with a higher PageRank score are considered more influential and likely to appear higher in search engines.

The PageRank algorithm uses a recursive formula to calculate the PageRank of each page, based on the PageRank of the pages that link to it. The formula is: **PR(A) = (1 - d) + d * (PR(B) / L(B) + PR(C) / L(C) + ... + PR(N) / L(N))** where PR(A) is the PageRank of page A, d is a damping factor (usually set to 0.85), L(B) is the number of outbound links from page B, and PR(B) / L(B) is the contribution of page B to the PageRank of page A. The formula is applied iteratively until the PageRank values converge to a stable state.

The implementation of PageRank algorithm is explained in the steps below:

Code:

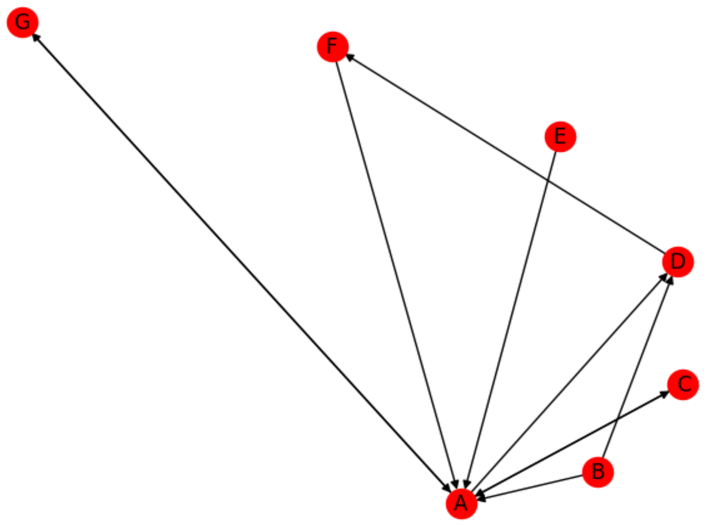
1. Importing required libraries.

```
import matplotlib.pyplot as plt
import networkx as nx
import pandas as pd
import scipy as scipy
```

2. Create graph with nodes and edges.

```
G = nx.DiGraph()

[G.add_node(k) for k in ["A", "B", "C", "D", "E", "F", "G"]]
G.add_edges_from([('G','A'), ('A','G'),('B','A'),
                  ('C','A'),('A','C'),('A','D'),
                  ('E','A'),('F','A'),('B','D'),
                  ('D','F')])
pos = nx.spiral_layout(G)
nx.draw(G, pos, with_labels = True, node_color="red")
```

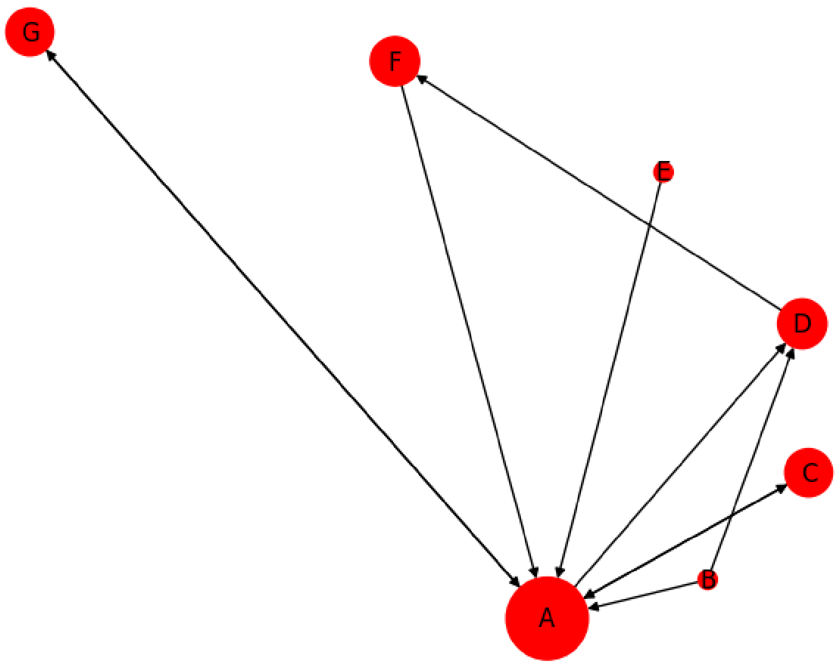


3. **Get the Pageranks for each Page using pagerank( ) function:**

- **pos: A dictionary with nodes as keys and positions as values**
- **nodelist: list of nodes to draw**
- **node_size: An optional scalar or array that specifies the size of the nodes**
- **with_labels: An optional boolean that specifies whether to draw labels on the nodes**

```
pr1 = nx.pagerank(G)
print(pr1)
nx.draw(G, pos, nodelist=list(pr1.keys()), node_size=[round(v * 4000) for
v in pr1.values()],
        with_labels = True, node_color="red")
```

```
{'A': 0.40001520046189115, 'B': 0.021428571428571432, 'C':
0.1347663991011727, 'D': 0.14387354195831553, 'E': 0.021428571428571432, 'F':
0.1437213165203047, 'G': 0.1347663991011727}
```



4. Conclusion: