**Experiment 10**

**Aim: Implementation of Page Rank Algorithm**

**Theory:**

PageRank (PR) is an algorithm used by Google Search to rank websites in their search engine results. PageRank was named after Larry Page, one of the founders of Google. PageRank is a way of measuring the importance of website pages. According to Google:

PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

It is not the only algorithm used by Google to order search engine results, but it is the first algorithm that was used by the company, and it is the best-known.

The above centrality measure is not implemented for multi-graphs.

Working:

Step 1: Web Graph Representation

Represent the web as a directed graph, where web pages are nodes, and hyperlinks are directed edges.

Step 2: Initialize PageRank

Assign an initial PageRank value to each page, typically a uniform value or equally distributing PageRank across all pages.

Step 3: Damping Factor

Introduce a damping factor (usually around 0.85) to account for random user behavior and ensure convergence.

Step 4: Iterative Calculation

Iteratively update PageRank values for each page. During each iteration:

Step 5: PageRank Calculation

Calculate the new PageRank value for each page using the formula:

PR(A) = (1 - d) / N + d \* (PR(B)/L(B) + PR(C)/L(C) + ... + PR(N)/L(N))

PR(A) is the new PageRank value for page A.

d is the damping factor.

N is the total number of pages.

PR(B), PR(C), ..., PR(N) are the current PageRank values of pages linking to page A.

L(B), L(C), ..., L(N) are the number of outbound links on pages B, C, ..., N.

Step 6: Iterate Until Convergence

Repeat the PageRank calculation iteratively until PageRank values stabilize (converge). This typically requires several iterations.

Step 7: Ranking

Rank pages based on their final PageRank scores. Higher PageRank scores indicate greater importance.

Step 8: Handling Sink Nodes

Address "sink nodes" (pages with no outbound links) to redistribute PageRank and prevent loss.

Step 9: Personalized PageRank

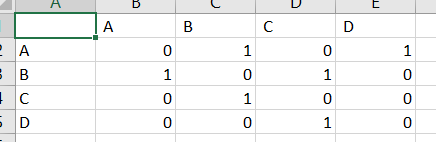
Calculate personalized PageRank to provide customized rankings based on user preferences or interests.

Step 10: Handling Dead-Ends

Consider how to handle "dead-end" pages (pages with no inbound links) to ensure they receive some PageRank value.

These steps summarize the key concepts of the PageRank algorithm, which plays a crucial role in web search and information retrieval systems.

**Dataset:**



**Program:**

import numpy as np

# Load the adjacency matrix from pagerank.csv

adjacency\_matrix = np.genfromtxt('pagerank.csv', delimiter=',', skip\_header=1, usecols=range(1, 5))

damping\_factor = 0.85

num\_iterations = 100

num\_nodes = len(adjacency\_matrix)

pagerank = np.ones(num\_nodes) / num\_nodes

for \_ in range(num\_iterations):

new\_pagerank = np.zeros(num\_nodes)

for i in range(num\_nodes):

for j in range(num\_nodes):

if adjacency\_matrix[j, i] == 1:

new\_pagerank[i] += pagerank[j] / np.sum(adjacency\_matrix[j])

pagerank = (1 - damping\_factor) / num\_nodes + damping\_factor \* new\_pagerank

# Create a list of nodes and their corresponding PageRank scores

nodes = ['A', 'B', 'C', 'D']

pagerank\_scores = list(pagerank)

# Sort the nodes based on their PageRank scores in descending order

sorted\_nodes = [node for \_, node in sorted(zip(pagerank\_scores, nodes), reverse=True)]

# Print the ranked nodes

for i, node in enumerate(sorted\_nodes):

print(f'{i+1}. Node {node}: {pagerank\_scores[nodes.index(node)]}')

**Output:**

