Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

A.Y. 2022-2023

DATA MINING AND WAREHOUSE

AYUSH JAIN

COMPUTER ENGINEERING | TE – B2 | 60004200132

EXPERIMENT – 8

Aim: To implement 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.

Algorithm:
The PageRank algorithm outputs a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. PageRank can be calculated for collections of documents of any size. It is assumed in several research papers that the distribution is evenly divided among all documents in the collection at the beginning of the computational process. The PageRank computations require several passes, called “iterations”, through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.

Simplified algorithm:
Assume a small universe of four web pages: A, B, C, and D. Links from a page to itself, or multiple outbound links from one single page to another single page, are ignored. PageRank is initialized to the same value for all pages. In the original form of PageRank, the sum of PageRank over all pages was the total number of pages on the web at that time, so each page in this example would have an initial value of 1. However, later versions of PageRank, and the remainder of this section, assume a

probability distribution between 0 and 1. Hence the initial value for each page in this example is 0.25.
The PageRank transferred from a given page to the targets of its outbound links upon the next iteration is divided equally among all outbound links.
If the only links in the system were from pages B, C, and D to A, each link would transfer 0.25 PageRank to A upon the next iteration, for a total of 0.75.

$$PR(A) = PR(B) + PR(C) + PR(D)$$

Suppose instead that page B had a link to pages C and A, page C had a link to page A, and page D had links to all three pages. Thus, upon the first iteration, page B would transfer half of its existing value, or 0.125, to page A and the other half, or 0.125, to page C. Page C would transfer all of its existing value, 0.25, to the only page it links to, A. Since D had three outbound links, it would transfer one-third of its existing value, or approximately 0.083, to A. At the completion of this iteration, page A will have a PageRank of approximately 0.458.

$$PR(A) = \frac{PR(B)}{2} \quad \frac{PR(C)}{1} \quad \frac{PR(D)}{3} + \; +$$

In other words, the PageRank conferred by an outbound link is equal to the document’s own PageRank score divided by the number of outbound links L( ).

$$PR(A) = \frac{PR(B)}{L(B)} \quad \frac{PR(C)}{L(C)} \quad \frac{PR(D)}{L(D)} + \; +$$

In the general case, the PageRank value for any page u can be expressed as:

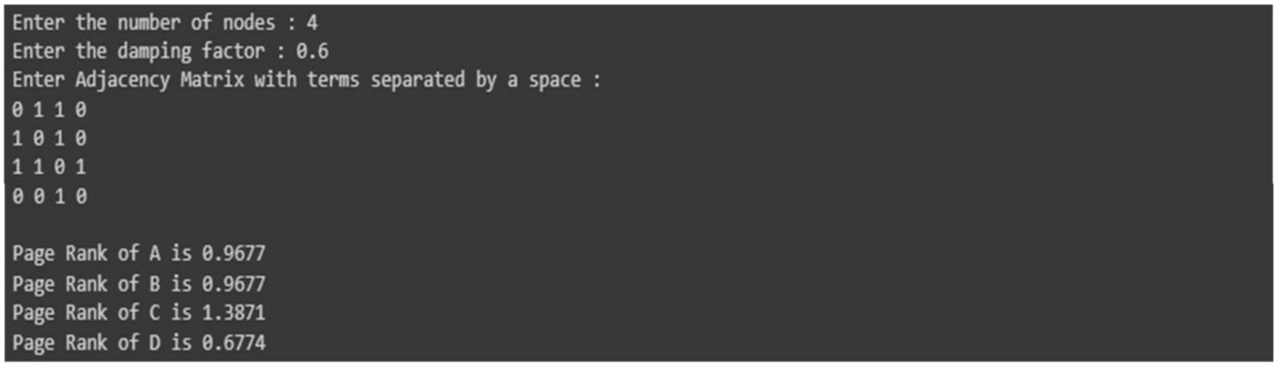
$$PR(u) = \Sigma \qquad \frac{PR(v)}{_{v \in B_u} L(v)}$$

i.e., the PageRank value for a page u is dependent on the PageRank values for each page v contained in the set Bu (the set containing all pages linking to page u), divided by the number L(v) of links from page v. The algorithm involves a damping factor for the calculation of the PageRank. It is like the income tax which the govt extracts from one despite paying him itself.

Code:

```
import numpy as np
def page_rank_algorithm(graph,damping_factor):
    outgoing = dict() incoming_nodes =
    dict()  coefficients  =  dict()  for  i  in
    range(len(graph)):
      outgoing[i]=0

    for i,node in enumerate(graph):
      for edge in node:
         if edge:
            outgoing[i] += 1
 for  i  in  range(len(graph)):  temp=[]  for
    node in graph:
         if node[i]:
            temp.append(node)
      incoming_nodes[i] = temp

    for i,node in enumerate(graph):
      temp = [] for j,other_node in enumerate(graph):
         if other_node in incoming_nodes[i]:
            temp.append(damping_factor*(1.0/outgoing[j])) elif i == j:
            temp.append(-1) else:
            temp.append(0) coefficients[i] = temp

    coefficients_list = [] for key,value in
    coefficients.items():
    coefficients_list.append(value)

    constant_matrix    =    []    for    i    in
    range(len(graph)):
      constant_matrix.append(damping_factor-1)

    pageranks = np.linalg.solve(np.array(coefficients_list),np.array(constant_matrix)) print() for i,rank in
    enumerate(pageranks):
      print('Page Rank of {} is {:.4f}'.format(chr(65+i), rank))

def main():
    n = int(input('Enter the number of nodes : ')) d = float(input('Enter the damping
    factor : ')) graph = [] print('Enter Adjacency Matrix with terms separated by a
    space : ') for i in range(n):
        temp_list       =       input().split('       ')
        graph.append(list(map(int,temp_list)))
    page_rank_algorithm(graph,d)

main()
```

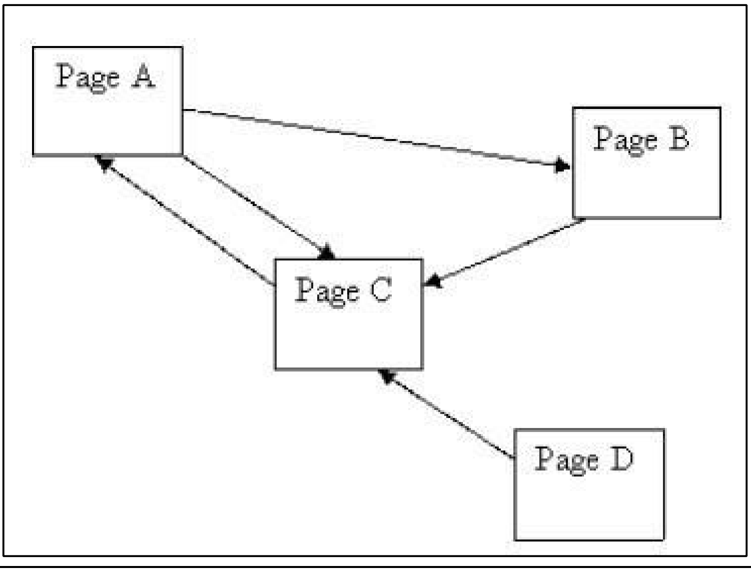
Output:



```
Enter the number of nodes : 4
Enter the damping factor : 0.6
Enter Adjacency Matrix with terms separated by a space :
0 1 1 0
1 0 1 0
1 1 0 1
0 0 1 0

Page Rank of A is 0.9677
Page Rank of B is 0.9677
Page Rank of C is 1.3871
Page Rank of D is 0.6774
```

Conclusion:

Learnt about page rank algorithm in web structure mining and implemented it in python for the graph:


Page A
Page B
Page C
Page D