Name: **Jigar Siddhpura** **SAPID:** 60004210155

DIV: C/C2 **Branch:** Computer Engineering

DMW - Experiment 8

Jigar Siddhpura
60004210155

DMW Experiment 8

Aim: To implement page rank algorithm.

Theory: 1. It is a algorithm used by google search engine & it was named after Larry page.

2. It works by counting no. of linked pages & quality pages to determine rough estimate of how important websites are likely to receive more links from other websites.

Algorithm: 1. This algorithm outputs a probability distribution used to represent the likelihood that a person randomly clicking on the links will arrive at any particular page.

2. Page rank can be calculated for collection of doc of any size.
3. Here, computations require several pass called 'iterations' through collection to adjust approx page rank.
4. Assume a universe of 4 pages - A, B, C, D where self links are ignored.
5. Initially, page is same for all. In original form, sum of page rank over all pages at that time, so each page in this example would have initial value 1.
6. we assume a probability distribution btw 0 & 1. Thus, here rank = 0.25 initially.
7. Page rank transferred from a given page to targets of its outbound links upon next iteration is divided equally among all outbound links.

8. If only links in the system were from pages B, C, D to A, each link would transfer 0.25 page rank to A upon next iteration.

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

9. Suppose that page B had a links to pg C & A, pg C has a link to pg A & D has links to all others.

10. Thus, on 2^{ND} iteration page B would transfer 0.125 to C. C would transfer all to A. Since D has 3 outbound links, it would tranfer 1/3 of existing value to A.

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

So, general formula is:

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

where L: No. of outbound links.

$$\therefore PR(u) = \sum \frac{PR(v)}{L(v)}$$

This algorithm involves a damping factor for the calculation of page rank. It is like income tax which govt. extract from one despite paying him itself.

Conclusion: Page rank offers a valuable insight into intricate world of web page ranking. On implementation, we can observe how page rank relies on link structure of a collection of documents to assign importance ~~so~~ scores.

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

   coefficients_list = []

    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.append(temp)

    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))
```

```
if __name__=="__main__":
    n = int(input("Enter the number of nodes : "))
    d = float(input("Enter the damping factor : "))
   # graph repr connected points

    graph = []
    # graph = [[0,1,1,0],[1,0,1,0],[1,1,0,1],[0,0,1,0]]
    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)
```

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
PS D:\SEM 5\DMW\EXPERIMENTS>
```

Conclusion:

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