Name: Siddharth Unny Batch: A4 Sap Id: 60004200080

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.

Working:
Each link from one page (A) to another (B) casts a so-called vote, the weight of which depends on the collective weight of all the pages that link to page A. And we can't know their weight till we calculate it, so the process goes in cycles.

The mathematical formula of the original PageRank is the following:

$$PR(A) = \frac{1-d}{N} + d\left(\frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \cdots\right)$$

Where A, B, C, and D are some pages, L is the number of links going out from each of them, and N is the total number of pages in the collection (i.e. on the Internet).

As for d, d is the so-called damping factor. Considering that PageRank is calculated simulating the behavior of a user who randomly gets to a page and clicks links, we apply this damping d factor as the probability of the user getting bored and leaving a page.

As you can see from the formula, if there are no pages pointing to the page, its PR will be not zero

$$PR(A) = \frac{1-d}{N}$$

As there's a probability that the user could get to this page not from some other pages but, say, from bookmarks.

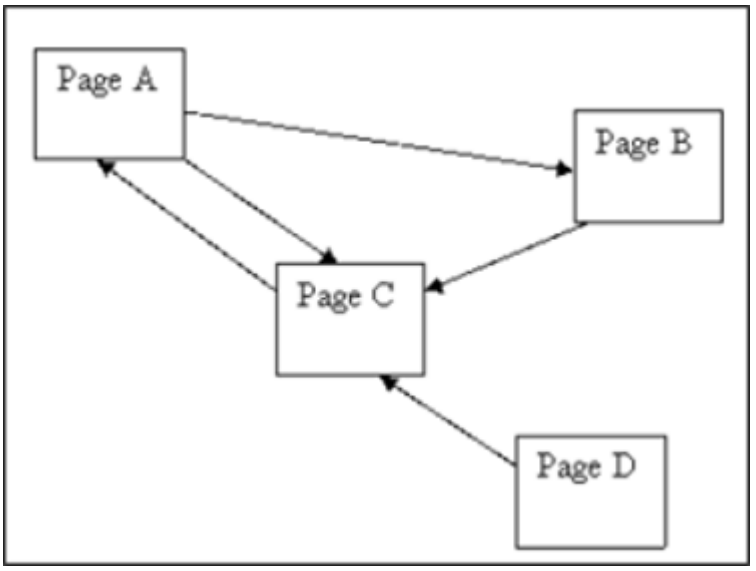
Code:

```
import numpy as np

def page_rank_algorithm(graph,damping_factor):
        outgoing = dict()
        incoming_nodes = dict()
        coefficients = dict()
        # Outgoing Nodes
        for i in range(len(graph)):
                outgoing[i]=0
for i,node in enumerate(graph):
                for edge in node:
                        if edge:
                                outgoing[i] += 1

        # Incoming Nodes for i in
        range(len(graph)): temp=[]
        for node in graph:
                 if node[i]:
                    temp.append(node)
                incoming_nodes[i] = temp

        # Coefficient Matrix 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_matr
    ix))
    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()
```

Graph:

Name: Siddharth Unny Batch: A4 Sap Id: 60004200080



Output:

```
Enter the number of nodes : 4
Enter the damping factor : 0.85
Enter Adjacency Matrix with terms separated by a space : 0 1 1 0
0  0 1 0
1  0 0 0
0 0 1 0

Page Rank of A is 1.4901
Page Rank of B is 0.7833
Page Rank of C is 1.5766 Page Rank of D is 0.1500
```

Conclusion:

Page Rank algorithm is one of the first algorithms in the history of Google search engine and is used to rank web pagesIt is a Web Structure Mining algorithm. Page Rank calculated is based on the incoming links (Backlinks). A dampening factor is used to avoid the rank sink problem. It's only drawback is that it favours old pages rather than the new ones but is still a widely used algorithm because of its efficiency.