LAB EXPERIMENT NO. 08

NAME:PARTH PAREKH SAP ID:60004200006

BATCH:A1 BRANCH:COMPUTER ENGG.

Aim:

To Implement Page Rank algorithm.

Theory:

Page Rank Algorithm

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

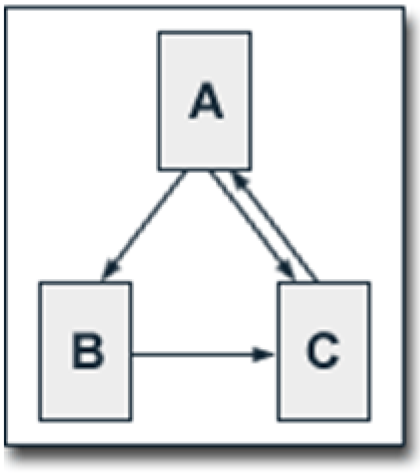
The PageRank algorithm outputs a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. PageRank can be calculated for collections of documents of any size. It is assumed in several research papers that the distribution is 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.

Page Rank is calculated using the following formula

$$PR(p) = (1 - d) + d \sum \frac{PR(q)}{N_q}$$

Where q = dampening factor

The page rank of the following graph is calculated as shown below



PR(A) = 0.5 + 0.5 PR(C)

PR(B) = 0.5 + 0.5 (PR(A) / 2)

PR(C) = 0.5 + 0.5 (PR(A) / 2 + PR(B))

These equations can easily be solved.

We get the following PageRank values for the single pages:

PR(A) = 14/13 = 1.07692308

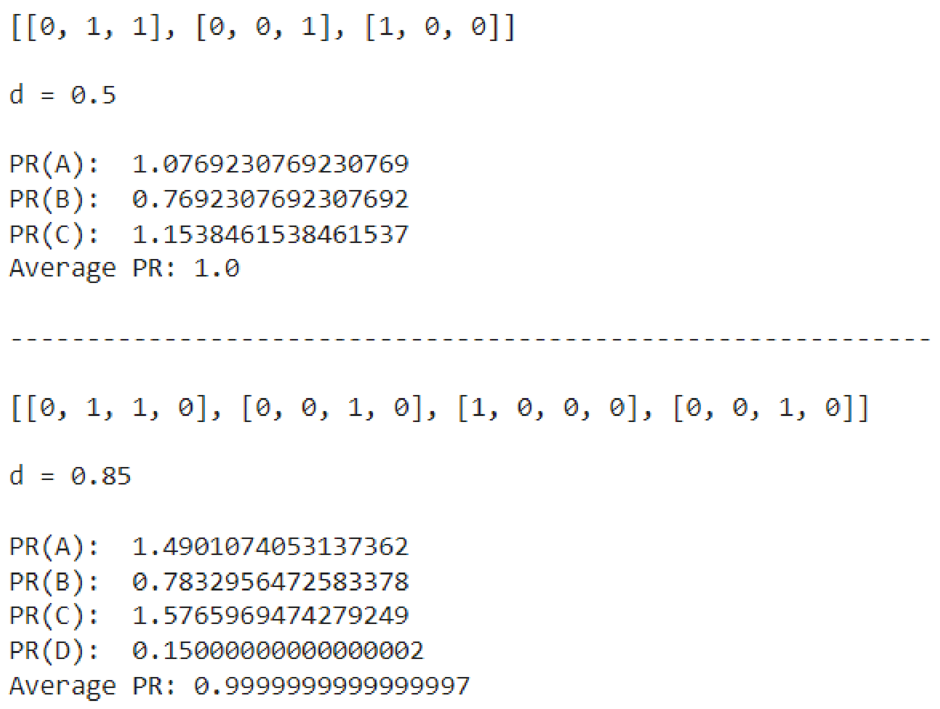
PR(B) = 10/13 = 0.76923077

PR(C) = 15/13 = 1.15384615

Code:

```
import numpy as np

def input():
    outgoing_matrix = [[0, 1, 1], [0, 0, 1], [1, 0, 0]]
    return outgoing_matrix

def second():
    outgoing_matrix = [[0, 1, 1, 0], [0, 0, 1, 0], [1, 0, 0, 0], [0, 0, 1, 
 0]]
    return outgoing_matrix

def result(d, matrix, row_index):
    summation = []
    for elem in matrix:
        summation.append(0)
    summation[row_index] = 1

    for i in range(len(matrix)):
        divisor = 0
        if matrix[i][row_index] == 1:
            for elem in matrix[i]:
                if elem == 1:
                    divisor += 1
            summation[i] = (-1 * d/divisor)
    return summation

def constant(d, matrix):
    b = []
    for i in matrix:
        b.append(1 - d)
    return b

def calculate(d, matrix):
    print(matrix, '\n\nd = {}\n'.format(d))
    a = []
    b = constant(d, matrix)
    for row_index in range(len(matrix)):
        a.append(result(d, matrix, row_index))

    a = np.array(a)
    b = np.array(b)
    x = np.linalg.solve(a, b)
    for i in range(len(x)):
        print("PR({}): ".format(chr(i+65)), x[i])
    print("Average PR: {}".format(np.average(x, weights=None)))

def main():
    matrix = input()
    calculate(0.5, matrix)

    print("\n------------------------------------------------------------
\n")
    matrix = second()
    calculate(0.85, matrix)

if __name__ == "__main__":
    main()
```

Output



```
[[0, 1, 1], [0, 0, 1], [1, 0, 0]]

d = 0.5

PR(A):  1.0769230769230769
PR(B):  0.7692307692307692
PR(C):  1.1538461538461537
Average PR: 1.0

------------------------------------------------------------

[[0, 1, 1, 0], [0, 0, 1, 0], [1, 0, 0, 0], [0, 0, 1, 0]]

d = 0.85

PR(A):  1.4901074053137362
PR(B):  0.7832956472583378
PR(C):  1.5765969474279249
PR(D):  0.15000000000000002
Average PR: 0.9999999999999997
```

Conclusion:

Thus, we have successfully implemented Page Rank algorithm. It outputs a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page.