Mahavir Education Trust's

SHAH & ANCHOR KUTCHHI ENGINEERING COLLEGE

Chembur, Mumbai - 400 088

UG Program in Cyber Security

<table>
<tr><th colspan="6">Experiment Number: 10</th></tr>
<tr><td colspan="2">Date of Performance:</td><td colspan="4"></td></tr>
<tr><td colspan="2">Date of Submission:</td><td colspan="4"></td></tr>
<tr><td>Program Execution/ formation/ correction/ ethical practices (07)</td><td>Documentation (02)</td><td>Timely Submission (03)</td><td>Viva Answer to sample questions (03)</td><td>Experiment Total (15)</td><td>Sign</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

Mahavir Education Trust's

SHAH & ANCHOR KUTCHHI ENGINEERING COLLEGE

Chembur, Mumbai - 400 088

UG Program in Cyber Security

Experiment 10

Aim: Implementation of Page rank algorithm.

Lab outcome: CSL 503.4: Implement Association rule mining and web mining algorithms.

Problem Statement: To implement Page rank algorithm.

Theory:

PageRank is an algorithm used by Google for link analysis. It was invented by Larry Page and Sergey Brin; the owners of Google. It was developed by Google to rank pages on their merit and not on meta tags as people had started misusing meta tags to improve their rankings. It evaluates the quality and quantity of links to a webpage and accordingly assigns a score on a 0 to 10 scale based on the page's importance and authority.

Page Rank Algorithm

Links from a page to itself, or multiple outbound links from one single page to another single page, are ignored. The PageRank theory holds the idea that an imaginary surfer who is randomly clicking on any links will eventually stop clicking at some point in time.

The probability, at any step, that the person will continue with a damping factor d. Various studies have tested different damping factors. It is generally assumed that the damping factor will be set around 0.85.

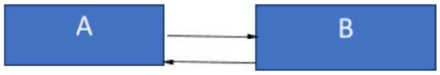
The formula for calculating the Page Rank of any page, A is –

Next iteration = Previous Iteration/Number of outgoing links.

$PR(node)=(1-d)+d*(PR(Ti)/C(Ti)\ldots+PR(Tn)/C(Tn))$

One has to refer to firstly incoming links and then outgoing links of the specific node. Ti is from where incoming links are shown onto node and C is the count of how many outgoing links are seen from Ti.

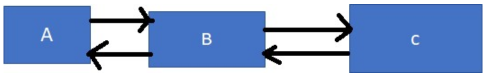
Assuming there is a web graph given as below –



d is a damping factor; its value is 0.85.

$PR(A) = (1-d)+d*(PR(B)/1)$

$PR(B) = (1-d)+d*(PR(A)/1)$



$PR(A) = (1-d)+d*(PR(B)/2)$

B has an incoming link onto A and B’s 2 outgoing links are onto C and onto A respectively.

$PR(B) = (1-d)+d*(PR(A)/1)+(PR(C)/1))$

B and C have incoming links and even they have each one outgoing link

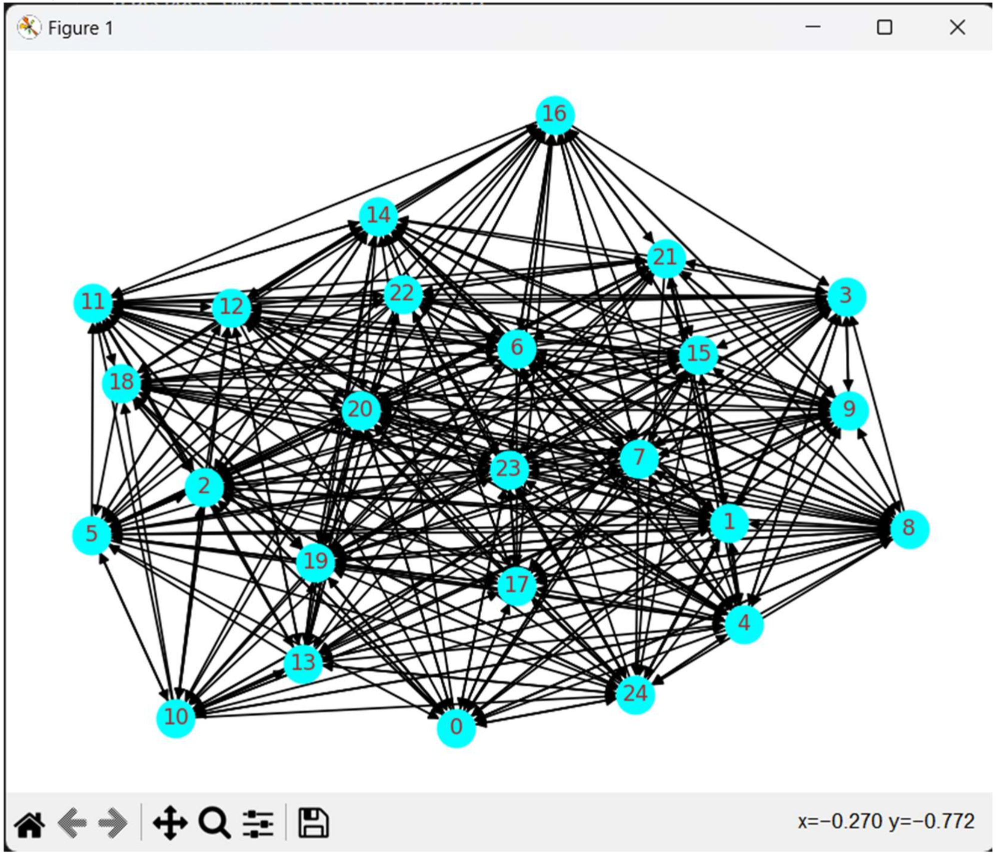
$PR(C) = (1-d)+d*(PR(B)/1)$

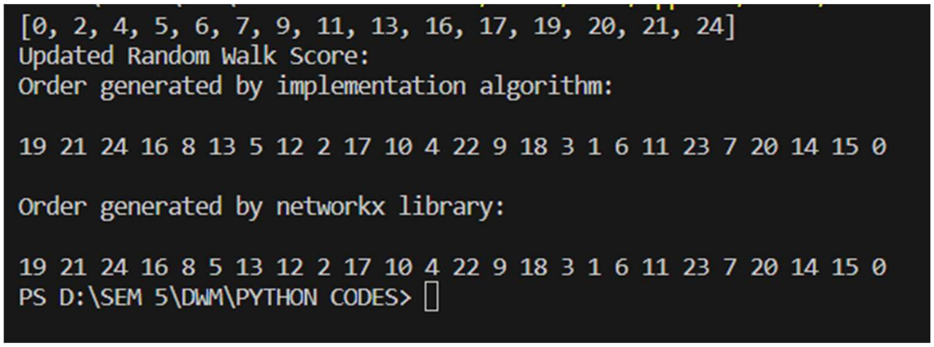
Finally, while solving the PageRank calculations, the values at each iteration have not been rounded at all. One can go through the number of iterations until one gets constant value.

Code:

```
import networkx as nx
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import operator
import random as rd

graph = nx.gnp_random_graph(25, 0.6, directed=True)
nx.draw(graph, with_labels=True, font_color='BROWN', font_size=10,
node_color='CYAN')
plt.show()

count = graph.number_of_nodes()
print(list(graph.neighbors(1)))

rank_dict = {}
x = rd.randint(0, 25)
for j in range(0, 25):
    rank_dict[j] = 0

rank_dict[x] = rank_dict[x] + 1

for i in range(600000):
    list_n = list(graph.neighbors(x))
    if len(list_n) == 0:
        x = rd.randint(0, 25)
        rank_dict[x] = rank_dict[x] + 1
    else:
        x = rd.choice(list_n)
        rank_dict[x] = rank_dict[x] + 1

print("Updated Random Walk Score:")

for j in range(0, 25):
    rank_dict[j] = rank_dict[j] / 600000

pagerank = nx.pagerank(graph)
pagerank_sorted = sorted(pagerank.items(), key=lambda v: (v[1], v[0]), reverse=True)
rank_dict_sorted = sorted(rank_dict.items(), key=lambda v: (v[1], v[0]),
reverse=True)

print("Order generated by implementation algorithm:\n")
for i in rank_dict_sorted:
    print(i[0], end=" ")

print("\n\nOrder generated by networkx library:\n")
for i in pagerank_sorted:
    print(i[0], end=" ")
```

Output:-





```
[0, 2, 4, 5, 6, 7, 9, 11, 13, 16, 17, 19, 20, 21, 24]
Updated Random Walk Score:
Order generated by implementation algorithm:

19 21 24 16 8 13 5 12 2 17 10 4 22 9 18 3 1 6 11 23 7 20 14 15 0

Order generated by networkx library:

19 21 24 16 8 5 13 12 2 17 10 4 22 9 18 3 1 6 11 23 7 20 14 15 0
PS D:\SEM 5\DWM\PYTHON CODES> 
```

Conclusion: Here we implemented Page rank algorithm.