Name:	Swarup Satish Kakade
Roll No:	17
Class/Sem:	TE/V
Experiment No.:	10
Title:	Implementation of page rank algorithm
Date of Performance:	
Date of Submission:	
Marks:	
Sign of Faculty:	

Aim: To implement Page Rank Algorithm

Objective: Develop a program to implement a page rank algorithm.

Theory:

PageRank (PR) is an algorithm used by Google Search to rank web pages in their search engine results. PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. Page Rank Algorithm is designed to increase the effectiveness of search engines and improve their efficiency. It is a way of measuring the importance of website pages. Page rank is used to prioritize the pages returned from a traditional search engine using keyword searching. Page rank is calculated based on the number of pages that point to it. The value of the page rank is the probability will be between 0 and 1. A web page is a directed graph having two important components: nodes and connections. The pages are nodes and hyperlinks are the connections, the connection between two nodes. Page rank 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 website are likely to receive more links from other websites. The page rank value of individual node in a graph depends on the page rank value of all the nodes which connect to it and those nodes are cyclically connected to the nodes whose ranking we want; we use converging iterative method for assigning values to page rank. In short page rank is a vote, by all the other pages on the web, about how important a page is. A link to a page count as a vote of support. If there is no link, there is no support.

We assume that page A has pages B.....N which point to it. Page rank of a page A is given as follows:

$$PR(A)=(1-\beta) +\beta ((PR(B)/cout(B))+ (PR(C)/cout(C))+-----+(PR(N)/cout(N)))$$

Parameter β is a teleportation factor which can be set between 0 and 1. Cout(A) is defined as the number of links going out of page A.

CODE:

```
import java.util.*;
import java.io.*;
```

```
public class PageRank {
public int path[][] = new int[10][10];
public double pagerank[] = new double[10];
public void calc(double totalNodes) {
double InitialPageRank;
double OutgoingLinks = 0;
double DampingFactor = 0.85;
double TempPageRank[] = new double[10];
int ExternalNodeNumber;

int InternalNodeNumber;
int k = 1; // For Traversing
int ITERATION_STEP = 1;
InitialPageRank = 1 / totalNodes;
System.out.printf(" Total Number of Nodes :" + totalNodes + "\t Initial PageRank of All
Nodes :" + InitialPageRank + "\n");
// 0th ITERATION _ OR _ INITIALIZATION PHASE //
for (k = 1; k <= totalNodes; k++) {
this.pagerank[k] = InitialPageRank;
}
System.out.printf("\n Initial PageRank Values , 0th Step \n");
for (k = 1; k <= totalNodes; k++) {
System.out.printf(" Page Rank of " + k + " is :\t" + this.pagerank[k] + "\n");
}
while (ITERATION_STEP <= 2) // Iterations
{
// Store the PageRank for All Nodes in Temporary Array
for (k = 1; k <= totalNodes; k++) {
TempPageRank[k] = this.pagerank[k];
this.pagerank[k] = 0;
}
for (InternalNodeNumber = 1; InternalNodeNumber <= totalNodes;
InternalNodeNumber++) {
for (ExternalNodeNumber=1;
ExternalNodeNumber <= totalNodes;
ExternalNodeNumber++) {
if (this.path[ExternalNodeNumber][InternalNodeNumber] == 1) {
k = 1;
OutgoingLinks = 0; // Count the Number of Outgoing Links for each
ExternalNodeNumber
while (k <= totalNodes) {
if (this.path[ExternalNodeNumber][k] == 1) {
OutgoingLinks = OutgoingLinks + 1; // Counter for Outgoing Links
```

```
}
k = k + 1;
}
// Calculate PageRank
this.pagerank[InternalNodeNumber] += TempPageRank[ExternalNodeNumber] * (1 /
OutgoingLinks);
}
}
}
```

Output:

```
Enter the Number of WebPages
4
Enter the Adjacency Matrix with 1->PATH & 0->NO PATH Between two WebPages:
0 1 1 0
0 0 1 1
1 0 0 0
0 0 1 0

Total Number of Nodes: 4  Initial PageRank of All Nodes: 0.25

Initial PageRank Values, 0th Step
Page Rank of 1 is : 0.25
Page Rank of 2 is : 0.25
Page Rank of 3 is : 0.25
Page Rank of 4 is : 0.25

After 1th Step
Page Rank of 1 is : 0.125
Page Rank of 2 is : 0.2125
Page Rank of 3 is : 0.375
Page Rank of 4 is : 0.25

After 2th Step
Page Rank of 1 is : 0.15625
Page Rank of 2 is : 0.24625
Page Rank of 3 is : 0.35625
Page Rank of 4 is : 0.24125
```

Final Page Rank:
Page Rank of 1 is : 0.1833125
Page Rank of 2 is : 0.2593125
Page Rank of 3 is : 0.3528125
Page Rank of 4 is : 0.2385625

Conclusion:

What are the key parameters of the PageRank algorithm, and how do they affect the algorithm's performance?

The key parameters of the PageRank algorithm and their effects are:

1. **Damping Factor (*α*, typically 0.85):** Controls the probability of continuing to follow links versus jumping to a random page. Higher values give more weight to link structure; lower values make the rank more uniform.

2. **Number of Iterations:** Affects convergence. More iterations improve accuracy but increase computation time.

3. **Convergence Threshold:** Determines when to stop iterating. Tighter thresholds improve accuracy but require more computation.

4. **Initial PageRank Values:** Usually uniform, but non-uniform values can influence convergence speed.

5. **Graph Structure (Adjacency Matrix):** Affects the computation. Dense graphs increase complexity; sparse graphs are more efficient.

6. **Handling Dangling Nodes:** Redistributing their rank impacts final values, ensuring rank isn't trapped in nodes without outbound links.

Adjusting these parameters allows you to control the precision and performance of the PageRank algorithm, optimizing it for different applications and datasets.