**Aim:** To implement Page Rank Algorithm

**Objective:** Develop a program to implement 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-β) +β ( (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);

}

}

}

System.out.printf("\n After " + ITERATION\_STEP + "th Step \n");

for (k = 1; k <= totalNodes; k++)

System.out.printf(" Page Rank of " + k + " is :\t" + this.pagerank[k] + "\n");

ITERATION\_STEP = ITERATION\_STEP + 1;

}

// Add the Damping Factor to PageRank

for (k = 1; k <= totalNodes; k++) {

this.pagerank[k] = (1 - DampingFactor) + DampingFactor \* this.pagerank[k];

}

// Display PageRank

System.out.printf("\n Final Page Rank : \n");

for (k = 1; k <= totalNodes; k++) {

System.out.printf(" Page Rank of " + k + " is :\t" + this.pagerank[k] + "\n");

}

}

public static void main(String args[]) {

int nodes, i, j, cost;

Scanner in = new Scanner(System.in);

System.out.println("Enter the Number of WebPages \n");

nodes = in .nextInt();

PageRank p = new PageRank();

System.out.println("Enter the Adjacency Matrix with 1->PATH & 0->NO PATH Between two WebPages: \n");

for (i = 1; i <= nodes; i++)

for (j = 1; j <= nodes; j++) {

p.path[i][j] = in .nextInt();

if (j == i)

p.path[i][j] = 0;

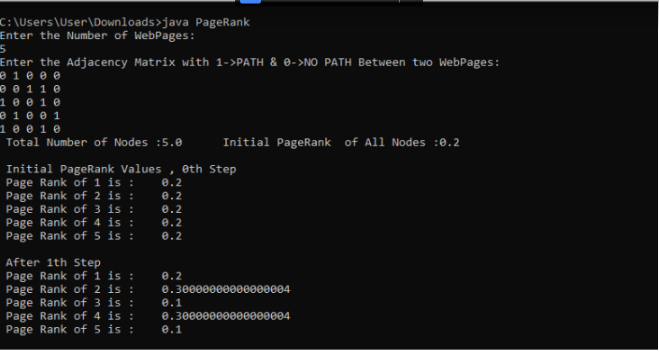
}

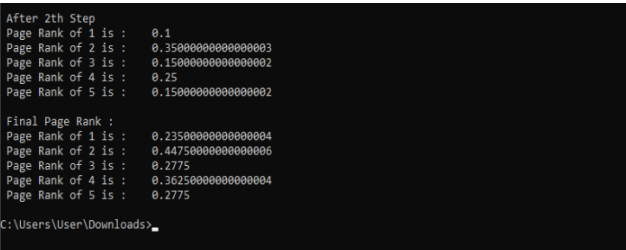
p.calc(nodes);

}

}

**OUTPUT:**





**Conclusion:**

Thus, we have learned to implement Page Rank Algorithm using python. 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.