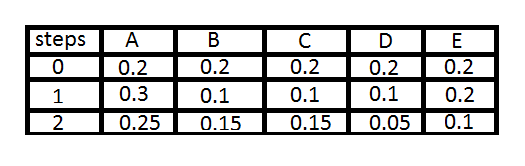
# Page Rank Algorithm

Suppose the number of webpages is n. First, we initialize the page rank of all webpages with 1/n. Then step by step we calculate Page Rank for each Webpage one after the other.  
  
**Example**



There are 5 Web pages represented by Nodes A, B, C, D and E. The hyperlink from each webpage to the other is represented by the arrow head. 

[](http://4.bp.blogspot.com/-WmFLq19ed7c/Vm6Q0NR3i-I/AAAAAAAAAKk/q9jiaMr2Y-M/s1600/PageRank+Values.png)

***Initialization***  
At 0th Step we have all Webpages PageRank values 0.2 that is 1/5 (1/n).

***Iteration step 1.***

To get PageRank of Webpage A, consider all the incoming links to A. So we have half the Page Rank of D (since D has two out links, A get 1/2 of D’s page rank) and full Page Rank of E (since E has one out link, A get 1/1 of E’s page rank). So it will be (1/5)\*(1/2) + (1/5)\*(1/1) which is (3/10) or 0.3 the Page Rank of A.

Similarly the Page Rank of B will be (1/5)\*(1/2) which is (1/10) or 0.1 because A's PageRank value is **1/5** or **0.2** from Step 0 . Even though we got 0.3 of A's PageRank in Step 1 we are considering 0.3 when we are Calculating Page Rank of B in Step 2.  
  
**We consider (N-1)th step values when we are calculating the Page Rank values for Nth Step.**  
  
In a similar way we calculate all the Page Rank Values and Sort them to get the most important webpage to be displayed in the Search Results.

Java Code for Page Rank Algorithm :

**package** edu.mum.prem.pagerank;

**import java.io.\*;**

**import java.util.\*;**

**public class PageRank {**

**private int n;**

**private int numberOfiteration;**

**private boolean normalize;**

**private double d;**

**private double edge[][];**

**private double prevPagerank[];**

**private double nextPagerank[];**

**public PageRank(int n, int numberOfiteration, int normalize, double d ) {**

**super();**

**this.n = n;**

**this.numberOfiteration = numberOfiteration;**

**this.normalize = (normalize == 1)? true : false;**

**this.d = d;**

**this.edge = new double[n][n];**

**this.prevPagerank = new double[n];**

**this.nextPagerank = new double[n];**

**}**

**public void setData(Scanner in){**

**double temp = 0;**

**double sum = 0;**

**System.out**

**.println("Enter the Adjacency Matrix with 1->Link & 0->NO Link Between two WebPages:");**

**for (int i = 0; i < n; i++){**

**sum = 0;**

**for (int j = 0; j < n; j++) {**

**temp = (double) in.nextInt();**

**edge[i][j] = temp;**

**if (temp != 0) sum++;**

**}**

**if (sum != 0){**

**for (int j = 0; j < n; j++) {**

**edge[i][j] = edge[i][j] / sum;**

**}**

**}**

**}**

**System.out.println("The Modified Adjacency Matrix: \n");**

**for (int i = 0; i < n; i++){**

**for (int j = 0; j < n; j++) {**

**System.out.print(edge[i][j] + "\t");**

**}**

**System.out.println();**

**}**

**double init = 1.0 / n;**

**System.out.printf(" n value:" + n + "\t init value :" + init + "\n");**

**for (int i = 0; i < n; i++)**

**this.prevPagerank[i] = init;**

**System.out.printf("\n Initial PageRank Values , 0th Step \n");**

**for (int i = 0; i < n; i++)**

**System.out.printf(" Page Rank of " + i + " is :\t"**

**+ this.prevPagerank[i] + "\n");**

**}**

**public void computeNextPageRank(int iteration) {**

**double temp = 0;**

**for (int i = 0; i < n; i++){**

**temp = 0;**

**for (int j = 0; j < n; j++){**

**if (edge[j][i] != 0)**

**temp = temp + prevPagerank[j] \* edge[j][i];**

**}**

**nextPagerank[i] = temp;**

**}**

**for (int i = 0; i < n; i++){**

**prevPagerank[i] = nextPagerank[i];**

**nextPagerank[i] = 0;**

**}**

**if (normalize){**

**temp = 0;**

**for (int i = 0; i < n; i++){**

**temp += prevPagerank[i];**

**}**

**for (int i = 0; i < n; i++){**

**prevPagerank[i] = prevPagerank[i] / temp;**

**}**

**}**

**for (int i = 0; i < n; i++)**

**prevPagerank[i] = (1 - d) + d \* prevPagerank[i];**

**System.out.printf("\nPageRank Values, " + iteration + "th Step \n");**

**for (int i = 0; i < n; i++)**

**System.out.printf(" Page Rank of " + i + " is :\t"**

**+ this.prevPagerank[i] + "\n");**

**System.out.println();**

**}**

**public void computePageRank(){**

**for (int i = 0; i < numberOfiteration; i++ )**

**computeNextPageRank(i);**

**}**

**public static void main(String args[]) {**

**Scanner in = new Scanner(System.in);**

**System.out.println("Enter: Number of WebPages, number of iterations, normalize, damping factor");**

**int nodes = in.nextInt();**

**int iterations = in.nextInt();**

**int normalize = in.nextInt();**

**double d = in.nextDouble();**

**PageRank p = new PageRank(nodes, iterations, normalize, d);**

**p.setData(in);**

**in.close();**

**p.computePageRank();**

**}**

**}**

Sample Run (above graph)

Enter: Number of WebPages, number of iterations, normalize, damping factor

5 50 1 .85

Enter the Adjacency Matrix with 1->Link & 0->NO Link Between two WebPages:

0 1 1 0 0

0 0 0 1 1

0 0 0 0 0

1 0 0 0 1

1 0 0 0 0

The Modified Adjacency Matrix:

0.0 0.5 0.5 0.0 0.0

0.0 0.0 0.0 0.5 0.5

0.0 0.0 0.0 0.0 0.0

0.5 0.0 0.0 0.0 0.5

1.0 0.0 0.0 0.0 0.0

n value:5 init value :0.2

Initial PageRank Values , 0th Step

Page Rank of 0 is : 0.2

Page Rank of 1 is : 0.2

Page Rank of 2 is : 0.2

Page Rank of 3 is : 0.2

Page Rank of 4 is : 0.2

PageRank Values, 0th Step

Page Rank of 0 is : 0.46875000000000006

Page Rank of 1 is : 0.25625000000000003

Page Rank of 2 is : 0.25625000000000003

Page Rank of 3 is : 0.25625000000000003

Page Rank of 4 is : 0.36250000000000004

PageRank Values, 1th Step

Page Rank of 0 is : 0.46034883720930236

Page Rank of 1 is : 0.29825581395348844

Page Rank of 2 is : 0.29825581395348844

Page Rank of 3 is : 0.231046511627907

Page Rank of 4 is : 0.31209302325581395

PageRank Values, 2th Step

Page Rank of 0 is : 0.42922063421170165

Page Rank of 1 is : 0.3002970075926753

Page Rank of 2 is : 0.3002970075926753

Page Rank of 3 is : 0.24737606074140245

Page Rank of 4 is : 0.32280928986154533

PageRank Values, 3th Step

Page Rank of 0 is : 0.44200726967201426

Page Rank of 1 is : 0.29035419677082924

Page Rank of 2 is : 0.29035419677082924

Page Rank of 3 is : 0.24819645640000898

Page Rank of 4 is : 0.3290878803863184

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

PageRank Values, 48th Step

Page Rank of 0 is : 0.441644011872917

Page Rank of 1 is : 0.29368571757266254

Page Rank of 2 is : 0.29368571757266254

Page Rank of 3 is : 0.24554854574234086

Page Rank of 4 is : 0.3254360072394172

PageRank Values, 49th Step

Page Rank of 0 is : 0.441644011872917

Page Rank of 1 is : 0.29368571757266254

Page Rank of 2 is : 0.29368571757266254

Page Rank of 3 is : 0.24554854574234086

Page Rank of 4 is : 0.3254360072394172

So from the above values, We have Webpage A is most important Page followed by Webpage E. Webpage B and C have equal importance and Webpage D is the least important. This helps to Rank Webpages in the Search results.

Note 1: If you choose to run normalize = 0, sum of all page ranks will not be 1 due to nodes such as c with no out links.

Note 2: if you choose to run d = 1, no damping takes place.