Java Program to Implement Simple PageRank Algorithm

Shravan Chitpady 12/14/2015 03:57:00 pm Algorithms, Java 9 comments

Please Disable the Adblocker!

When you go and type some keywords in Google Search Engine a list of Web Pages will be displayed ,but how does the search engine know which page to be shown first to the user ? To solve this problem a algorithm called PageRank was developed at Stanford university by Larry Page and Sergey Brin in 1996.The PageRank Algorithm uses probabilistic distribution to calculate rank of a Web page and using this rank display the search results to the user. The Pagerank is recalculated every time the search engine crawls the web.

The original Page Rank algorithm which was described by Larry Page and Sergey Brin is :

$$PR(A) = (1-d) + d\ (PR(W1)/C(W1) + ... + PR(Wn)/C(Wn))$$

Where :
PR(A) – Page Rank of page A
PR(Wi) – Page Rank of pages Wi which link to page A
C(Wi) - number of outbound links on page Wi
d - damping factor which can be set between 0 and 1

To calculate PageRank for the n Webpages ,First we initialise all Webpages with equal page rank of 1/n each.Then Step by Step we calculate Page Rank for each Webpage one after the other.

Let us take one example :

Codispatch Pages

About
Projects
Sitemap
Contact Us

Follow Codispatch on Google +

Subscribe to Our Newsletter

Email address... Submit

What's Popular this Week ?

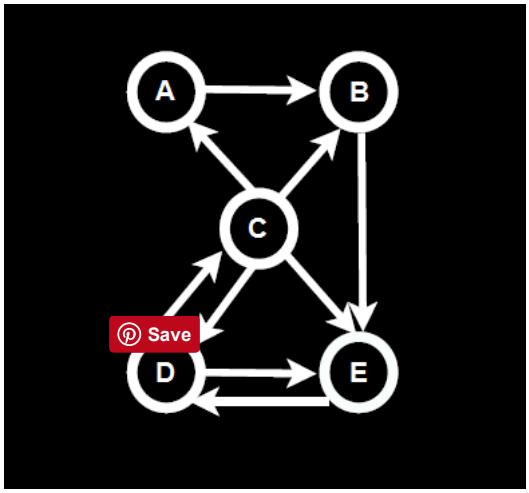

Java Program to Implement Simple PageRank Algorithm

Java Program to Implement Dijkstra's Shortest Path Algorithm

Java Program to Implement Prim's Minimum Spanning Tree


Java Program to Search Pattern using Horspool Algorithm


Login Password Protected Page using Sessions | PHP



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

steps	A	B	C	D	E
0	0.2	0.2	0.2	0.2	0.2
1	0.05	0.25	0.1	0.25	0.35
2	0.025	0.075	0.125	0.375	0.4

At 0th Step we have all Webpages PageRank values 0.2 that is 1/5 (1/n) . To get PageRank of Webpage A ,consider all the incoming links to A .So we have 1/4th the Page Rank of C is pointed to A. So it will be (1/5)*(1/4) which is (1/20) or 0.05 the Page Rank of A.

Similarly the Page Rank of B will be (1/5)*(1/4)+(1/5)*(1/1) which is (5/20) or 0.25 because A's PageRank value is 1/5 or 0.2 from Step 0 . Even though we got 0.05 of A's PageRank in Step 1 we are considering 0.05 when we are Calculating Page Rank of B in Step 2.

The general rule is --> we consider (N-1)th step values when we are calculating the Page Rank values for Nth Step . Not Clear ? Please Comment it below .

In 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 .

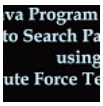
	Iteration 0	Iteration 1	Iteration 2	Page Rank
P_1	1/5	1/20	1/40	5
P_2	1/5	5/20	3/40	4
P_3	1/5	1/10	5/40	3
P_4	1/5	5/20	15/40	2
P_5	1/5	7/20	16/40	1

Edith Law - lecture12

Java Code for Page Rank Algorithm :


Java Program to Generate Student Rank for Competitive Exams


Java Program to Implement Breadth First Search (BFS)


Java Program to Search Pattern using Brute Force Technique

Codispatch Posts By Label

Java (7) Algorithms (6) PHP (3) Go (Golang) (1) Learn-Fast (1)

```
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;InternalNodeNumb
er++)
 {

for(ExternalNodeNumber=1;ExternalNodeNumber<=totalNodes;ExternalNodeNumb
er++)
   {
    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);


   }

}
```

Please Click and Drag from the beginning to End of the above Source code for Selection and Copying.

To Compile and Run for above Example:

javac PageRank.java
java PageRank

Enter the Number of WebPages : 5
Enter the Adjacency Matrix with 1->PATH & 0->NO PATH Between two WebPages:

0 1 0 0 0
0 0 0 0 1
1 1 0 1 1
0 0 1 0 1
0 0 0 1 0

Total Number of Nodes :5.0 Initial PageRank of All Nodes :0.2

Initial PageRank Values , 0th Step
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
Page Rank of 5 is : 0.2

After 1th Step
Page Rank of 1 is : 0.05
Page Rank of 2 is : 0.25
Page Rank of 3 is : 0.1
Page Rank of 4 is : 0.25
Page Rank of 5 is : 0.35

After 2th Step
Page Rank of 1 is : 0.025
Page Rank of 2 is : 0.07500000000000001
Page Rank of 3 is : 0.125
Page Rank of 4 is : 0.375
Page Rank of 5 is : 0.4

Final Page Rank :
Page Rank of 1 is : 0.17125
Page Rank of 2 is : 0.21375000000000002
Page Rank of 3 is : 0.25625000000000003
Page Rank of 4 is : 0.46875
Page Rank of 5 is : 0.49000000000000005

Note:

- Final Page Rank Includes Damping Factor of 0.85 which is usually set between 0 and 1.
- InternalNodeNumber represents the Node which you are currently calculating its PageRank.
- ExternalNodeNumber represents the Nodes Other than InternalNodeNumber.

For every InternalNodeNumber check if there is any Incoming Links from ExternalNodeNumber if No - Ignore and move to next ExternalNodeNumber,If Yes - Count all the OutgoingLinks for that ExternalNodeNumber.

Finally Calculate Pagerank :
PR(InternalNodeNumber) += PR(ExternalNodeNumber)/All OutgoingLinks for ExternalNodeNumber

So from the above values , We have Webpage A(1) is the most important Page , Webpage B(2) and C(3) have almost equal importance with B(2) slightly more importance ,Webpage D(4) has some importance and Webpage E(5) has least importance.This helps to Rank Webpages in the Search results.

Please Note: Actual google Page rank Algorithm for large network of webpages grows logarithmic and slightly different from the one above. This Page Rank algorithm is fully owned by google inc and I just illustrated with a help of a Java Program to implement this Algorithm .I hope you enjoyed this .Thanks Have Nice Day.

Update1: New Example has been Added and Images are Updated.
Update2: I have Considered Damping Factor in my Implementation which is set to 0.85.
Update3:while(u<=2) Changed to while(ITERATION_STEP<=2).

Related Posts:

Java Program to Implement Simple PageRank Algorithm
When you go and type some keywords in Google Search Engine a list of Web Pages will be displayed ,but how does the search engine know which page to … Learn More

Java Program to Search Pattern using Brute Force Technique
Brute-Force technique of searching the pattern in the given string has a Worst case time complexity of O(m.n) where m is the length of the string an… Learn More

Java Program to Implement Dijkstra's Shortest Path Algorithm
Simple Java Program code to find the shortest path from single source using Dijkstra's Single Source Shortest Path Algorithm .It is similar to Prim'… Learn More

Java Program to Search Pattern using Horspool Algorithm
Horspool Algorithm is used to search the pattern in the given string using a shift table. Its another variation of the Boyer-Moore Algorithm where i… Learn More

Java Program to Implement Prim's Minimum Spanning Tree
A Spanning tree of a connected graph G is a acyclic subgraph of graph G that includes all vertices of G. A minimum spanning tree of connected graph… Learn More

Newer Post Home Older Post

9 comments:

Sunayana Salunkhe 15 April 2016 at 21:56

why while condition is u<=2? Please explain

Reply Delete

Replies

Shravan Chitpady 15 April 2016 at 23:03

Thanks Sunayana Salunkhe for your comment .

Some of the pages on the web have no links to other pages. Some of the links in the database have been picked up from fetched web pages but the web pages those links are pointing to are not yet fetched. If a page has not been fetched it is considered to be a page that has no links to other pages. These pages, called dangling links, without any links are a problem because their PageRank cannot be redistributed to other pages. Their PageRank “leaks out” of the graph and therefore gets lost.

To prevent this, all pages without links and links to these pages are removed before calculation of PageRank values. Removing pages from the graph can cause other pages to lose all their links and become dangling links themselves. This is why removing dangling links is an iterative process.

On every iteration all dangling links are identified before they are removed, and the number of iterations needed is counted. It is not necessary to remove all dangling links. A couple of iterations will remove most of them.

Ref: Google: data structures and algorithms

Delete

Reply

Spiti Sawant 20 April 2016 at 22:10

what is the purpose of scanner class in the code: Scanner in = new Scanner(System.in); ?

Reply Delete

Replies

Shravan Chitpady 21 April 2016 at 15:28

There are various ways to read input from the keyboard, the java.util.Scanner class is one of them.The Java Scanner class breaks the input into tokens using a delimiter that is whitespace bydefault. It provides many methods to read and parse various primitive values Java Scanner class is widely used to parse text for string and primitive types using regular expression.Java Scanner class extends Object class and implements Iterator and Closeable interfaces.

Example : Scanner in = new Scanner(System.in);
int a = in.nextInt(); // Variable "a" has Integer input from Console

Delete

Reply

Ryuuji 30 October 2016 at 03:30

How do I use the damping factor?

Reply Delete

Ryuuji 30 October 2016 at 03:31

How do I use the damping factor?

Reply Delete

Replies

Shravan Chitpady 10 December 2016 at 09:45

I have Updated my Post and Added DampingFactor of 0.85

Delete

Reply

Duminda Madhusanka 6 December 2016 at 00:45

can you explain,in step 2, how D become 0.1 without any incoming edges and how E become 0.2 ..and also how you have got Adjacency Matrix as you mentioned .(the way you creater the matrix)thank you

Reply Delete

Replies

Shravan Chitpady 9 December 2016 at 01:04

I have Updated my Post. Adjacency Matrix is got from the Directed Graph. Check if there is a Direct Path between two web pages then take path has 1, if there is no direct path or path goes through other nodes then take path has 0. :)

Delete

Reply

Add comment

Enter your comment...

Comment as: Samkit Jain (G ▾ Sign out

Publish Preview Notify me