**Project Report**

**Subject: Object Oriented Programming (CS 8392)**

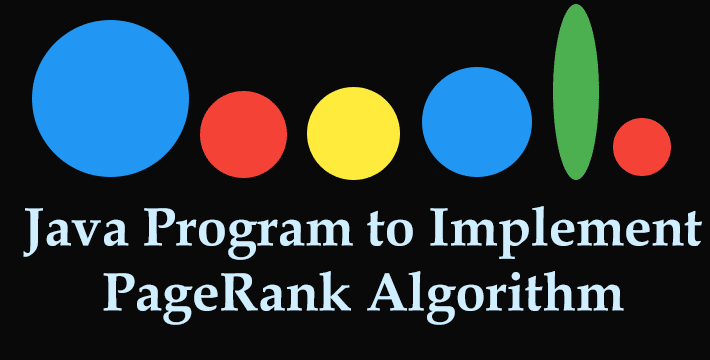
**Project name: A Search Result Ranking Algorithm for web pages**

**JASON PETER FALLON**

**ROLL NO: 201901048**

**CSE – SEC A – 11 Year**

**Introduction and background**



**What is a search engine algorithm?**

**Search engines** use **algorithms** to determine the quality of a website, the theme of a website, and what types of queries the website should show up for in **search** results. **Algorithms** are also used to determine where in organic **search** results a particular website should be displayed for a specific query.

**How does Page Rank algorithm work?**

**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.

**How is PageRank calculated?**

**PageRank** for a given page = Initial **PageRank** + (total ranking power ÷ number of outbound links) + … Google assigns every new web page an initial **PageRank** score.

**What is Google's algorithm?**

**Google's algorithms** are a complex system used to retrieve data from its search index and instantly deliver the best possible results for a query. The search engine uses a combination of **algorithms** and numerous ranking signals to deliver webpages ranked by relevance on its search engine results pages (SERPs).

**What is Page Rank in SEO?**

**PageRank** (PR) is an algorithm used by Google Search to **rank** web **pages** in their search engine results. **PageRank** was named after Larry **Page**, one of the founders of Google. ... **PageRank** works by counting the number and quality of links to a **page** to determine a rough estimate of how important the website is.

**Project Status Report – Summary**

**Content**:

**Work done for the project:**

After very focussed and valuable teaching by the faculty of this subject, I was allocated the above topic for OOP and sufficient guidance was also given by the faculty to us to develop the entire project and submit the report.

The phase and sequence of developing the project involved finalising the topic “A Search Result Ranking Algorithm for web pages” as my Project. A number of relevant web sites were visited in order to obtain sufficient knowledge about the topic. This lead to a sufficient information gathering and proper Literature survey for the project. A presentation was made, which gave a brief of the existing algorithms used and a description of the proposed system. The presentation also included

* Scope of the System/Introduction and background
* Literature Survey of the System
* Basics Involved
* Analysis and Design
* Workflow Charts
* Applications of the system

An OOP report was drafted which contained the concept giving a brief idea of the project followed by the explanation and the conclusion inferred from our literature survey. Finally a report was compiled based on the information / research done on the topic and the same was presented to the management/OOP Faculty.

**Summary Progress Report**

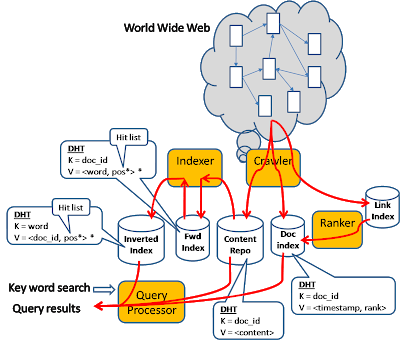
|  |  |  |  |
| --- | --- | --- | --- |
| **Project Name:** | **A Search Result Ranking Algorithm for Web pages** | **Report Month:** | **July-August** |
| **Project Guide:** |  | **Phone, E-mail:** |  |
| **Project Description:** | Locating useful information effectively from the World Wide Web (WWW) is of wide interest. The huge volume of the return results makes the user only focus on the top results. So the ranking problem becomes the important task for the search Systems. On studying the various ranking algorithms, and analyzing their merits and demerits, using several evaluating methods to assess and contrast with Google can be presented. | | |
| **Upcoming tasks for this period:** | Plan to design an effective and running Ranking Algorithm | | |

**How ranking in Google Search works**

**Search engine and the processing phases**

A very basic search engine includes a number of processing phases.

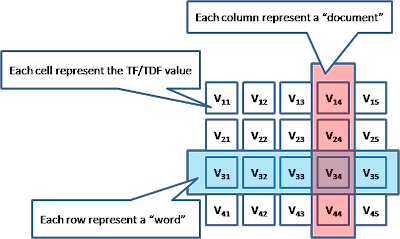
* **Crawling:** to discover the web pages on the internet
* **Indexing:** to build an index to facilitate query processing
* **Query Processing:** Extract the most relevant page based on user's query terms
* **Ranking:** Order the result based on relevancy

[](http://2.bp.blogspot.com/_j6mB7TMmJJY/S4140VTxW0I/AAAAAAAAAbk/jWcEZ8zC-rY/s1600-h/p1.png)

Notice that each element in the above diagram reflects a logical function unit but not its physical boundary. For example, the processing unit in each orange box is in fact executed across many machines in parallel. Similarly, each of the data store element is spread physically across many machines based on the key partitioning.

### Vector Space Model

Here we use the "Vector Space Model" where each document is modeled as a multi-dimensional vector (each word represents a dimension). If we put all documents together, we form a matrix where the rows are documents and columns are words, and each cell contains the [TF/IDF value](http://horicky.blogspot.com/2009/01/solving-tf-idf-using-map-reduce.html) of the word within the document.

[](http://4.bp.blogspot.com/_j6mB7TMmJJY/S46fNdfOGBI/AAAAAAAAAcU/ujpleeWteho/s1600-h/p1.png)

To determine the similarity between 2 documents, we can apply the dot product between 2 documents and the result will represents the degree of similarity.

### Crawler

Crawler's job is to collect web pages on the internet, it is typically done by a farm of crawlers, who do the following

Start from a set of seed URLs, repeat following ...

1. Pick the URL that has the highest traversal priority.
2. Download the page content from the URLs to the content repository (which can be a distributed file system, or DHT), as well as update the entry in the doc index
3. Discover new URL links from the download pages. Add the link relationship into the link index and add these links to the traversal candidates
4. Prioritize the traversal candidates

The content repository can be any distributed file system, here lets say it is a DHT.

There are a number of considerations.

* How to make sure different Crawlers are working on different set of contents (rather than crawling the same page twice) ? When the crawler detects overlapping is happening (url is already exist in the page repository with pretty recent time), the crawler will skip the processing on this URL and pick up the next best URL to crawl.
* How does the crawler determines which is the next candidate to crawl ? We can use a heuristic algorithm based on some utility function (e.g. we can pick the URL candidate which has the highest page rank score)
* How frequent do we re-crawl? We can track the rate of changes of the page to determine the frequency of crawling.

### Indexer

The Indexer's job is to build the inverted index for the query processor to serve the online search requests.

First the indexer will build the "forward index"

1. The indexer will parse the documents from the content repository into a token stream.
2. Build up a "hit list" which describe each occurrence of the token within the document (e.g. position in the doc, font size, is it a title, archor text ... etc).
3. Apply various "filters" to the token stream (like stop word filters to remove words like "a", "the", or a stemming filter to normalize words "happy", "happily", "happier" into "happy")
4. Compute the term frequency within the document.

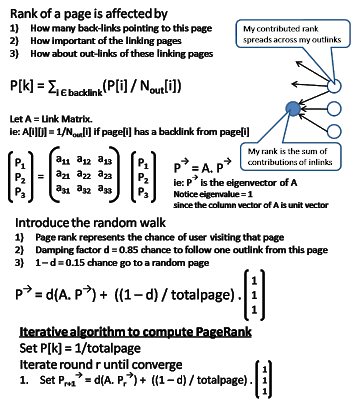
From the forward index, the indexer will proceed to build a reverse index (typically through a Map/Reduce mechanism). The result will be keyed by word and stored in a DHT.

### Ranker

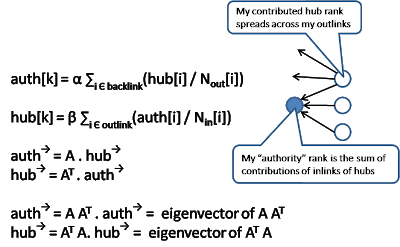
Ranker's job is to compute the rank of a document, based on how many in-links pointing to the document as well as the rank of the referrers (hence a recursive definition). Two popular ranking algorithms including the "Page Rank" and "HITs".

* **Page Rank Algorithm**

Page rank is a global rank mechanism. It is precomputed upfront and is independent of the query

* [](http://4.bp.blogspot.com/_j6mB7TMmJJY/S46R5QlyJ_I/AAAAAAAAAcE/VPN-rl0fPuM/s1600-h/P2.png)**HITS Algorithm**

In HITS, every page is playing a dual role: "hub" role and "authority" role. It has two corresponding ranks on these two roles. Hub rank measures the quality of the outlinks. A good hub is one that points to many good authorities. Authority ranks measures the quality of my content. A good authority is one that has many good hubs pointing to.

[](http://4.bp.blogspot.com/_j6mB7TMmJJY/S46bN88QRlI/AAAAAAAAAcM/YwCOJtSezd8/s1600-h/P3.png)

Notice that HITS doesn't pre-compute the hub and authority score. Instead it invoke a regular search engine (which only do TF/IDF matches but not ranking) to get a set of initial results (typically with a predefined fix size) and then expand this result set by tracing the outlinks into the expand result set. It also incorporate a fix size of inlinks (by sampling the inlinks into the initial result set) into the expanded result set. After this expansion, it runs an iterative algorithm to compute the authority ranks and hub ranks. And use the combination of these 2 ranks to calculate the ultimate rank of each page, usually pages with high hub rank will weight more than high authority rank.

Notice that the HITS algorithm is perform at query time and not pre-computed upfront. The advantage of HITS is that it is sensitive to the query (as compare to PageRank which is not). The disadvantage is that it perform ranking per query and hence expensive.

### Query Processor

When user input a search query (containing multiple words), the query will be treated as a "query document". Relevancy is computed and combined with the rank of the document and return an ordered list of result.

There are many ways to compute the relevancy. We can consider only the documents that contains all the terms specified in the query. In this model, we search for each term (with the query) a list of document id and then do an intersection with them. If we order the document list by the document id, the intersection can be computed pretty efficiently.

Alternatively, we can return the union (instead of intersection) of all document and order them by a combination of the page rank TF/IDF score. Document that have more terms intersecting with the query will have a higher TF/IDF score.

In some cases, an automatic query result feedback loop can be used to improve the relevancy.

1. In first round, the search engine will perform a search (as described above) based on user query
2. Construct a second round query by expanding the original query with additional terms found in the return documents which has high rank in the first round result
3. Perform a second round of query and return the result.

### Outstanding Issues

Fighting the spammer is a continuous battle in search engine. Because of the financial value of being shown up in the first page of search result. Many spammers try to manipulate their page. Earlier attempt is to modify a page to repeat the terms many many times (trying to increase the TF/IDF score). The evolution of Page rank has mitigate this to some degree because page rank in based on "out-of-page" information that the site owner is much harder to manipulate.

But people use Link-farms to game the page rank algorithms. The ideas is to trade links between different domains. There is active research in this area about how to catch these patterns and discount their ranks

**Google page rank**



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.

It is an important off-page optimization factor. It decides how easily and quickly your users can find your webpages on the world wide web. If your PageRank is good the user will find you easily as the search engine will keep your site higher on the search engine listings.

**How is a PageRank determined?**

It is calculated by a proprietary mathematical formula that considers each link to a website as a vote. A website is compared with every other website with similar content and keywords in a popularity contest. The website with most links including the most valuable links gets a higher rank in popularity.

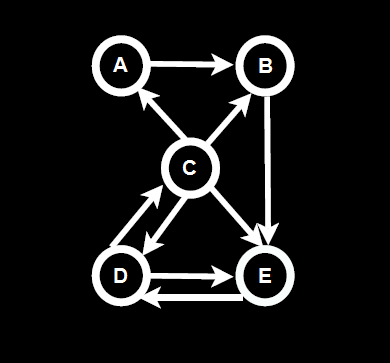
**Some important link building strategies to improve your Google PageRank**

* List your site on renowned and popular directories and stay away from link farms that are not useful.
* Be a part of online forums and share valuable comments with backlinks to your site.
* Publish relevant articles on trusted and popular article submission sites like Ezine Articles and Associated Content.
* Find out popular sites and encourage them for link exchange
* Be active on social media; use different techniques like Social Bookmarking, Twitter profile links and Google+ shares for building links

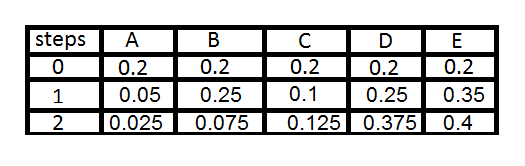
## [Java Program to Implement PageRank Algorithm](https://codispatch.blogspot.com/2015/12/java-program-implement-google-page-rank-algorithm.html)

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 :

[](https://2.bp.blogspot.com/-ax1tVrOD3GA/WEmkCUFTnXI/AAAAAAAAAYI/0l4-yoyzwMMV838bdRIqTHmJ5EEZhiP_wCLcB/s1600/Java+Program+to+Implement+Simple+PageRank+Algorithm.png)

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.

[](https://3.bp.blogspot.com/--xQU4Qbl6BA/WEmgSLbO9GI/AAAAAAAAAX8/w9jfLQbHXpwPP_7HTrVITDrONvweAZp7gCLcB/s1600/PageRank+Values.png)

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.

he 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 .

|  |
| --- |
| [Java Program to Implement Simple PageRank Algorithm](https://1.bp.blogspot.com/-mG1vM1WSvGM/WEmfcOVzNrI/AAAAAAAAAX0/PQ4fgS7qkvMrNBZ_FDEhm6ATOHQQOTUvwCPcB/s1600/PageRankTable.png) |
| Edith Law - lecture12 Java Code for Page Rank Algorithm : 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);  }  }  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

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).