

Page Rank Algorithm (Big Data Analytic)

Project Report submitted in partial fulfillment of the requirement for the
degree of
Bachelor of Technology.

in

Computer Science & Engineering

under the Supervision of

Dr. Pardeep Kumar

By

Navneet Nara(131306)

Pranshu Sharma (131318)

to



Jaypee University of Information and Technology
Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled “Page Rank Algorithm”, submitted by ***Pranshu Sharma and Navneet Nara*** in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:04-05-2017

Dr. Pardeep Kumar
Assistant Professor
(Senior Grade)

Acknowledgement

We take this opportunity to express our profound gratitude and deep regards to our guide Dr. Pardeep Kumar for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by his time to time shall carry a long way in the journey of life on which we are about to embark.

The in-time facilities provided by the Computer Science department throughout the project development are also equally acknowledgeable.

At the end we would like to express my sincere thanks to all our friends and others who helped us directly or indirectly during this project work.

Date: 04-05-2017

Pranshu Sharma(131318)

Navneet Nara (131306)

Table of Contents

Serial Number	Topics	Page Numbers
1	Chapter-1. Introduction	1
2	1.1 Introduction	2
3	History	2
4	1.2 Problem Statement	3
5	1.3 Objective	3
6	1.4 Methodology	3
7	2. Literature Survey	4
8	Chapter-3 System Development	30
9	Chapter-4 Performance Analysis	33
10	Outputs	33
11	Graphs	36
12	Chapter-5 Conclusion	40
13	5.1 Conclusions	40
14	5.2 Future Scope	40
15	5.2.1 Improved Algorithm	40
16	5.3 Applications	41
17	Chapter-6 Appendix	42

List of Figures:

Figures	Page Number
Fig 2.1: 5V's of Big Data	4
Fig 2.2: Two pages pointing to each other	11
Fig 2.3: Simplified Algorithm	14
Fig 2.4: Matrix	14
Fig 2.5: Directed Graph	14
Fig 2.6: Types of Links	15
Fig 2.7,2.8: Instance 1	15,16
Fig 2.9,2.10: Instance 2	17
Fig 2.11: Instance 3	18
Fig 3.1: Instance 1	30

List Of Graphs:

Graphs	Page Numbers:
Graph 1: Number of edges v/s execution time	36
Graph 2: Number of links v/s execution time	37

List of Tables:

Tables	Page Numbers:
Table 2.1 : List of Pageranks	9
Table 3.1: Pageranks of different nodes	31
Table 3.2: Pagerank at different iterations	32

Chapter-1. Introduction

1.1 Introduction

PageRank is an algorithm used by Google Search to rank websites in their search engine results. PageRank was named after Larry Page, one of the founders of Google. PageRank is a way of measuring the importance of website pages. According to 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. The underlying assumption is that more important websites are likely to receive more links from other websites.

It is not the only algorithm used by Google to order search engine results, but it is the first algorithm that was used by the company, and it is the best-known.

PageRank is a link analysis algorithm and it assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of measuring its relative importance within the set. The algorithm may be applied to any collection of entities with reciprocal quotations and references. The numerical weight that it assigns to any given element E is referred to as the PageRank of E and denoted by other factors like Author Rank can contribute to the importance of an entity.

A PageRank results from a mathematical algorithm based on the webgraph, created by all World Wide Web pages as nodes and hyperlinks as edges, taking into consideration authority hubs such as cnn.com or usa.gov. The rank value indicates an importance of a particular page. A hyperlink to a page counts as a vote of support. The PageRank of a page is defined recursively and depends on the number and PageRank metric of all pages that link to it (incoming links). A page that is linked to by many pages with high PageRank receives a high rank itself.

HISTORY

Back in 1990s, the occurrence of the keyword is the only important rule to judge if a document is relevant or not. The document with the highest number of occurrences of keywords receives the highest score based on the traditional text retrieval model. This approach works fine on text retrieval, but it has its flaws. It only looks the content of a document, but ignore its influence. All documents in the collection are seen as equally important. In the large-scale web, this may undermine the retrieval quality. For instance, if you search harvard in your browser, you would expect that your search engine ranks the homepage of the Harvard University as the most relevant page. Suppose word harvard appears much more often in a Harvard student's homepage than in www.harvard.edu because that student listed all courses he has taken in Harvard, and all papers he has published, etc, which all contain harvard. Should we consider this student's homepage is more relevant than www.harvard.edu to our query. In worst scenario, if we create a web page that contains harvard a million times. Should we consider this page is relevant to the query harvard? The answer is of course not.

In 1998, Larry Page and Sergey Brin, two graduate students at Stanford University, has invented the PageRank algorithm that model the structure of pages on the web and quantize the importance of each page. PageRank is one of the most known and influential algorithms for computing the relevance of web pages, and is used by Google, the most successful search engine on the web. The basic idea of PageRank is that the importance of a web page depends on the pages that link to it.

1.2 Problem Statement

As discussed in introduction, there are a lot of web pages which are linked to each other so we need an algorithm in order to sort them according to their importance. Therefore, google page rank algorithm is used whose implementation is done using big data.

1.3 Objective

The objective of this project is to study google page rank algorithm and implement it using big data. Also in the long run find ways to improve the current algorithm.

1.4 Methodology

Within the PageRank concept, the rank of a document is given by the rank of those documents which link to it. Their rank again is given by the rank of documents which link to them. Hence, the PageRank of a document is always determined recursively by the PageRank of other documents. Since - even if marginal and via many links - the rank of any document influences the rank of any other, PageRank is, in the end, based on the linking structure of the whole web. Although this approach seems to be very broad and complex, Page and Brin were able to put it into practice by a relatively trivial algorithm.

2. Literature Survey

Big data

Big data is a term that describes the large volume of data – both structured and unstructured – that inundates a business on a day-to-day basis. But it's not the amount of data that's important. It's what organizations do with the data that matters. Big data can be analyzed for insights that lead to better decisions and strategic business moves.

While the term big data is relatively new, the act of gathering and storing large amounts of information for eventual analysis is ages old. The concept gained momentum in the early 2000s when industry analyst Doug Laney articulated the now-mainstream definition of big data as the three Vs:

Volume. Organizations collect data from a variety of sources, including business transactions, social media and information from sensor or machine-to-machine data. In the past, storing it would've been a problem – but new technologies (such as Hadoop) have eased the burden.

Velocity. Data streams in at an unprecedented speed and must be dealt with in a timely manner. RFID tags, sensors and smart metering are driving the need to deal with torrents of data in near-real time.

Variety. Data comes in all types of formats – from structured, numeric data in traditional databases to unstructured text documents, email, video, audio, stock ticker data and financial transactions.

At SAS, we consider two additional dimensions when it comes to big data:

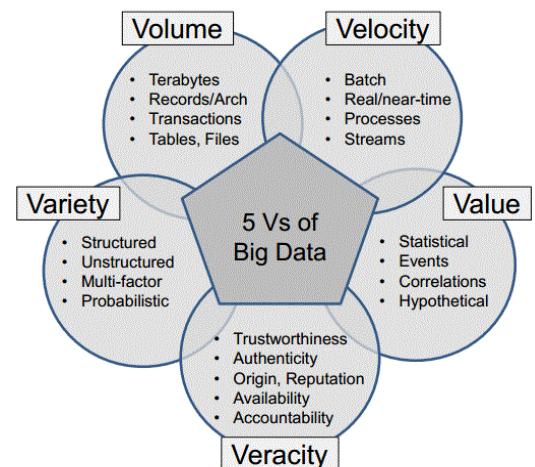


Fig 2. 1

Variability. In addition to the increasing velocities and varieties of data, data flows can be highly inconsistent with periodic peaks. Is something trending in social media? Daily, seasonal and event-triggered peak data loads can be challenging to manage. Even more so with unstructured data.

Complexity. Today's data comes from multiple sources, which makes it difficult to link, match, cleanse and transform data across systems. However, it's necessary to connect and correlate relationships, hierarchies and multiple data linkages or your data can quickly spiral out of control.

Why Is Big Data Important?

The importance of big data doesn't revolve around how much data you have, but what you do with it. You can take data from any source and analyze it to find answers that enable 1) cost reductions, 2) time reductions, 3) new product development and optimized offerings, and 4) smart decision making. When you combine big data with high-powered analytics, you can accomplish business-related tasks such as:

- Determining root causes of failures, issues and defects in near-real time.
- Generating coupons at the point of sale based on the customer's buying habits.
- Recalculating entire risk portfolios in minutes.

Who uses big data?

Big data affects organizations across practically every industry. See how each industry can benefit from this onslaught of information.

- **Banking:** With large amounts of information streaming in from countless sources, banks are faced with finding new and innovative ways to manage big data. While it's important to understand customers and boost their satisfaction, it's equally important to minimize risk and fraud while maintaining regulatory compliance. Big data brings big insights, but it also requires financial institutions to stay one step ahead of the game with advanced analytics.
- **Education:** Educators armed with data-driven insight can make a significant impact on school systems, students and curriculums. By analyzing big data, they can identify at-risk students, make sure students are making adequate progress, and can implement a better system for evaluation and support of teachers and principals.

- **Government:** When government agencies are able to harness and apply analytics to their big data, they gain significant ground when it comes to managing utilities, running agencies, dealing with traffic congestion or preventing crime. But while there are many advantages to big data, governments must also address issues of transparency and privacy.
- **Health Care:** Patient records. Treatment plans. Prescription information. When it comes to health care, everything needs to be done quickly, accurately – and, in some cases, with enough transparency to satisfy stringent industry regulations. When big data is managed effectively, health care providers can uncover hidden insights that improve patient care.
- **Manufacturing:** Armed with insight that big data can provide, manufacturers can boost quality and output while minimizing waste – processes that are key in today’s highly competitive market. More and more manufacturers are working in an analytics-based culture, which means they can solve problems faster and make more agile business decisions.
- **Retail:** Customer relationship building is critical to the retail industry – and the best way to manage that is to manage big data. Retailers need to know the best way to market to customers, the most effective way to handle transactions, and the most strategic way to bring back lapsed business. Big data remains at the heart of all those things.
- **How It Works**
 - Before discovering how big data can work for your business, you should first understand where it comes from. The sources for big data generally fall into one of three categories:
 - Streaming data

This category includes data that reaches your IT systems from a web of connected devices. You can analyze this data as it arrives and make decisions on what data to keep, what not to keep and what requires further analysis.
 - Social media data

The data on social interactions is an increasingly attractive set of information, particularly for marketing, sales and support functions. It's often in unstructured or semi structured forms, so it poses a unique challenge when it comes to consumption and analysis.
 - Publicly available sources

Massive amounts of data are available through open data sources like the US government’s data.gov, the CIA World Fact book or the European Union Open Data Portal.

- After identifying all the potential sources for data, consider the decisions you'll need to make once you begin harnessing information. These include:

- **How to store and manage it?**

Whereas storage would have been a problem several years ago, there are now low-cost options for storing data if that's the best strategy for your business.

- **How much of it to analyze?**

Some organizations don't exclude any data from their analyses, which is possible with today's high-performance technologies such as grid computing or in-memory analytics. Another approach is to determine upfront which data is relevant before analyzing it.

- **How to use any insights you uncover?**

The more knowledge you have, the more confident you'll be in making business decisions. It's smart to have a strategy in place once you have an abundance of information at hand.

The final step in making big data work for your business is to research the technologies that help you make the most of big data and big data analytics. Consider:

- Cheap, abundant storage.
 - Faster processors.
 - Affordable open source, distributed big data platforms, such as Hadoop.
 - Parallel processing, clustering, MPP, virtualization, large grid environments, high connectivity and high throughputs.
 - Cloud computing and other flexible resource allocation arrangements.
-

Page Rank is a topic much discussed by Search Engine Optimisation (SEO) experts. At the heart of PageRank is a mathematical formula that seems scary to look at but is actually fairly simple to understand.

Despite this many people seem to get it wrong! In particular Chris Ridings of www.searchenginesystems.net has written a paper entitled PageRank Explained: Everything you've always wanted to know about PageRank, pointed to by many people, that contains a fundamental mistake early on in the explanation! Unfortunately this means some of the recommendations in the paper are not quite accurate.

By showing code to correctly calculate real PageRank I hope to achieve several things in this response:

1. Clearly explain how PageRank is calculated.
2. Go through every instance in Chris paper, and add some more of my own, showing the correct PageRank for each diagram. By showing the code used to calculate each diagram I've opened myself up to peer review - mostly in an effort to make sure the instances are correct, but also because the code can help explain the PageRank calculations.
3. Describe some principles and observations on website design based on these correctly calculated instances.

Any good web designer should take the time to fully understand how PageRank really works - if you don't then your site's layout could be seriously hurting your Google listings!

[Note: I have nothing in particular against Chris. If I find any other papers on the subject I'll try to comment evenly]

How is PageRank Used?

PageRank is one of the methods Google uses to determine a page's relevance or importance. It is only one part of the story when it comes to the Google listing, but the other aspects are discussed elsewhere (and are ever changing) and PageRank is interesting enough to deserve a paper of its own.

PageRank is also displayed on the toolbar of your browser if you've installed the Google toolbar (<http://toolbar.google.com/>). But the Toolbar PageRank only goes from 0 – 10 and seems to be something like a logarithmic scale:

Toolbar PageRank
(log base 10)

Real PageRank

0	0 - 10
1	100 - 1,000
2	1,000 - 10,000
3	10,000 - 100,000
4	and so on...

Table 2.1

We can't know the exact details of the scale because, as we'll see later, the maximum PR of all pages on the web changes every month when Google does its re-indexing! If we presume the scale is logarithmic (although there is only anecdotal evidence for this at the time of writing) then Google could simply give the highest actual PR page a toolbar PR of 10 and scale the rest appropriately.

Also the toolbar sometimes guesses! The toolbar often shows me a Toolbar PR for pages I've only just uploaded and cannot possibly be in the index yet!

What seems to be happening is that the toolbar looks at the URL of the page the browser is displaying and strips off everything down the last (i.e. it goes to the parent page in URL terms). If Google has a Toolbar PR for that parent then it subtracts 1 and shows that as the Toolbar PR for this page. If there's no PR for the parent it goes to the parent's parent's page, but subtracting 2, and so on all the way up to the root of your site. If it can't find a Toolbar PR to display in this way, that is if it doesn't find a page with a real calculated PR, then the bar is greyed out.

Note that if the Toolbar is guessing in this way, the Actual PR of the page is 0 - though its PR will be calculated shortly after the Google spider first sees it.

PageRank says nothing about the content or size of a page, the language it's written in, or the text used in the anchor of a link!

Definitions

I've started to use some technical terms and shorthand in this paper. Now's as good a time as any to define all the terms I'll use:

- PR:** Shorthand for PageRank: the actual, real, page rank for each page as calculated by Google. As we'll see later this can range from 0.15 to billions.
- Toolbar PR:** The PageRank displayed in the Google toolbar in your browser. This ranges from 0 to 10.

Backlink: If page A links out to page B, then page B is said to have a backlink from page A.

That's enough of that, let's get back to the meat...

So what is PageRank?

In short PageRank is a vote, by all the other pages on the Web, about how important a page is. A link to a page counts as a vote of support. If there's no link there's no support (but it's an abstention from voting rather than a vote against the page).

Quoting from the original Google paper, PageRank is defined like this:

We assume page A has pages $T_1 \dots T_n$ which point to it (i.e., are citations). The parameter d is a damping factor which can be set between 0 and 1. We usually set d to 0.85. There are more details about d in the next section. Also $C(A)$ is defined as the number of links going out of page A. The PageRank of a page A is given as follows:

$$PR(A) = (1-d) + d (PR(T_1)/C(T_1) + \dots + PR(T_n)/C(T_n))$$

Note that the PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one.

PageRank or $PR(A)$ can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web.

but that's not too helpful so let's break it down into sections.

1. **$PR(T_n)$** - Each page has a notion of its own self-importance. That's $PR(T_1)$ for the first page in the web all the way up to $PR(T_n)$ for the last page
2. **$C(T_n)$** - Each page spreads its vote out evenly amongst all of its outgoing links. The count, or number, of outgoing links for page 1 is $C(T_1)$, $C(T_n)$ for page n, and so on for all pages.
3. **$PR(T_n)/C(T_n)$** - so if our page (page A) has a backlink from page n the share of the vote page A will get is $PR(T_n)/C(T_n)$
4. **$d(\dots)$** - All these fractions of votes are added together but, to stop the other pages having too much influence, this total vote is damped down by multiplying it by 0.85 (the factor d)
5. **$(1 - d)$** - The $(1 - d)$ bit at the beginning is a bit of probability math magic so the sum of all web pages' PageRanks will be one: it adds in the bit lost by the $d(\dots)$. It also means that if a page has no links to it (no backlinks) even then it will still get a small PR of 0.15 (i.e. $1 - 0.85$). (Aside: the Google paper says the sum of all pages but they mean the the normalised sum – otherwise known as the average to you and me.

How is PageRank Calculated?

This is where it gets tricky. The PR of each page depends on the PR of the pages pointing to it. But we won't know what PR those pages have until the pages pointing to **them** have their PR calculated and so on... And when you consider that page links can form circles it seems impossible to do this calculation!

But actually it's not that bad. Remember this bit of the Google paper:

PageRank or $PR(A)$ can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web.

What that means to us is that we can just go ahead and calculate a page's PR **without knowing the final value of the PR of the other pages**. That seems strange but, basically, each time we run the calculation we're getting a closer estimate of the final value. So all we need to do is remember the each value we calculate and repeat the calculations lots of times until the numbers stop changing much.

Lets take the simplest instance network: two pages, each pointing to the other:

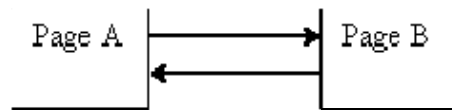


Fig 2. 2

Each page has one outgoing link (the outgoing count is 1, i.e. $C(A) = 1$ and $C(B) = 1$).

Guess 1

We don't know what their PR should be to begin with, so let's take a guess at 1.0 and do some calculations:

$$d = 0.85$$

$$PR(A) = (1 - d) + d(PR(B)/1)$$

$$PR(B) = (1 - d) + d(PR(A)/1)$$

i.e.

$$\begin{aligned} PR(A) &= 0.15 + 0.85 * 1 \\ &= 1 \end{aligned}$$

$$\begin{aligned} \text{PR}(B) &= 0.15 + 0.85 * 1 \\ &= 1 \end{aligned}$$

Hmm, the numbers aren't changing at all! So it looks like we started out with a lucky guess!!!

Guess 2

No, that's too easy, maybe I got it wrong (and it wouldn't be the first time). Ok, let's start the guess at 0 instead and re-calculate:

$$\begin{aligned} \text{PR}(A) &= 0.15 + 0.85 * 0 \\ &= 0.15 \end{aligned}$$

$$\begin{aligned} \text{PR}(B) &= 0.15 + 0.85 * 0.15 \\ &= 0.2775 \end{aligned}$$

NB. we've already calculated a next best guess at PR(A) so we use it here

And again:

$$\begin{aligned} \text{PR}(A) &= 0.15 + 0.85 * 0.2775 \\ &= 0.385875 \end{aligned}$$

$$\begin{aligned} \text{PR}(B) &= 0.15 + 0.85 * 0.385875 \\ &= 0.47799375 \end{aligned}$$

And again

$$\begin{aligned} \text{PR}(A) &= 0.15 + 0.85 * 0.47799375 \\ &= 0.5562946875 \end{aligned}$$

$$\begin{aligned} \text{PR}(B) &= 0.15 + 0.85 * 0.5562946875 \\ &= 0.622850484375 \end{aligned}$$

and so on. The numbers just keep going up. But will the numbers stop increasing when they get to 1.0? What if a calculation over-shoots and goes above 1.0?

Guess 3

Well let's see. Let's start the guess at 40 each and do a few cycles:

$$\text{PR}(A) = 40$$

$$\text{PR}(B) = 40$$

First calculation

$$\begin{aligned} \text{PR}(A) &= 0.15 + 0.85 * 40 \\ &= 34.25 \end{aligned}$$

$$\begin{aligned} \text{PR}(B) &= 0.15 + 0.85 * 0.385875 \\ &= 29.1775 \end{aligned}$$

And again

$$\begin{aligned} \text{PR}(A) &= 0.15 + 0.85 * 29.1775 \\ &= 24.950875 \end{aligned}$$

$$\begin{aligned} \text{PR}(B) &= 0.15 + 0.85 * 24.950875 \\ &= 21.35824375 \end{aligned}$$

Yup, those numbers are heading down alright! It sure looks the numbers will get to 1.0 and stop

Here's the code used to calculate this instance starting the guess at 0: [Show the code](#) | [Run the program](#)

- **Principle:** it doesn't matter where you start your guess, once the PageRank calculations have settled down, the normalized probability distribution (the average PageRank for all pages) will be 1.0

Getting the answer quicker

How many times do we need to repeat the calculation for big networks? That's a difficult question; for a network as large as the World Wide Web it can be many millions of iterations! The damping factor is quite subtle. If it's too high then it takes ages for the numbers to settle, if it's too low then you get repeated over-shoot, both above and below the average - the numbers just swing about the average like a pendulum and never settle down.

Also choosing the order of calculations can help. The answer will always come out the same no matter which order you choose, but some orders will get you there quicker than others.

I'm sure there's been several Master's Thesis on how to make this calculation as efficient as possible, but, in the instances below, I've used very simple code for clarity and roughly 20 to 40 iterations were needed!

Simplified algorithm:

Page ID	OutLinks
1	2,3,4,5,7
2	1
3	1,2
4	2,3,5
5	1,3,4,6
6	1,5
7	5

Fig 2. 3



Fig 2. 4

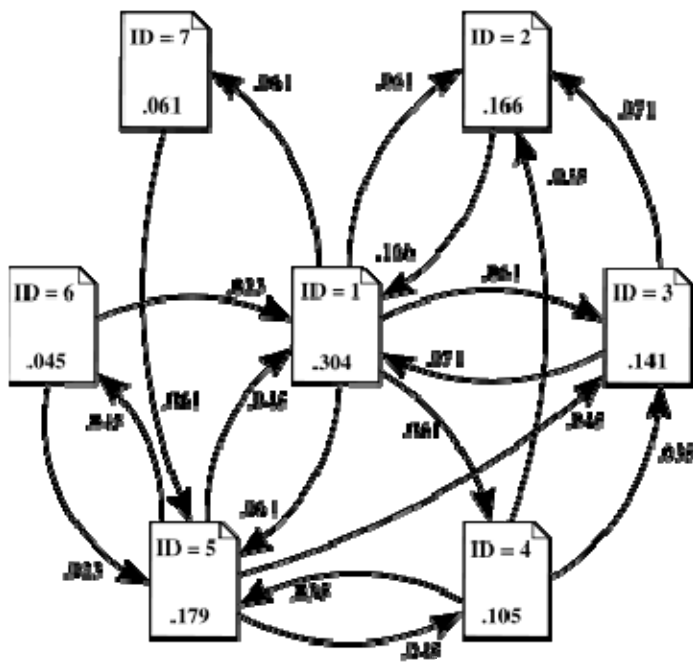


Fig 2. 5

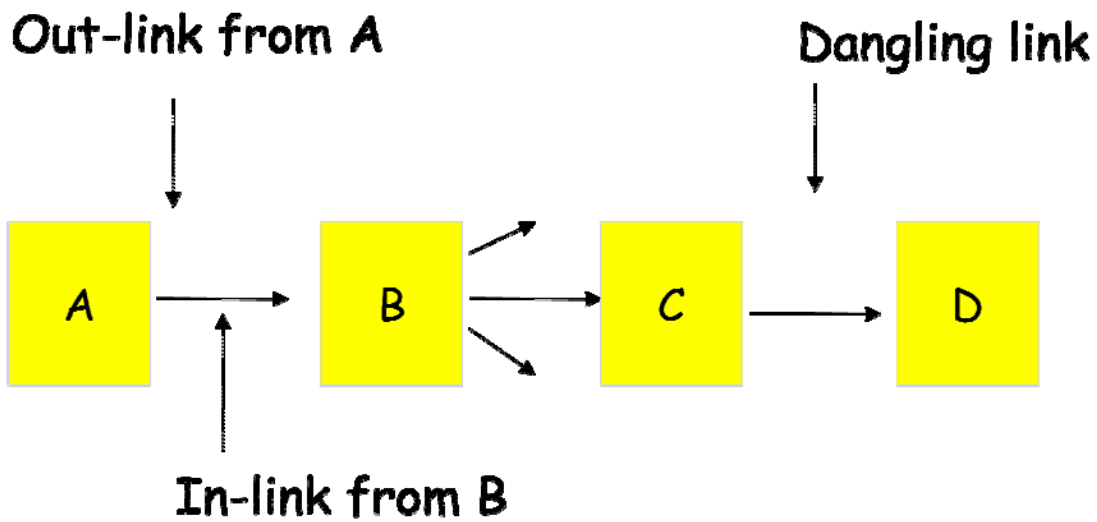


Fig 2. 6

Instance 1

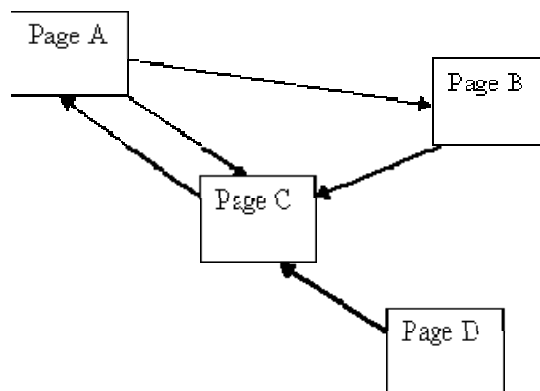
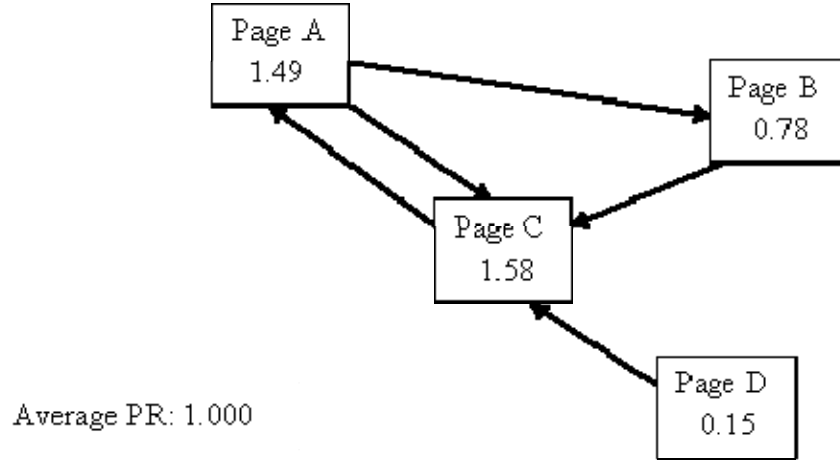


Fig 2. 7

I'm not going to repeat the calculations here, but you can see them by running the program (yes, if you click the link the program really is re-run to do the calculations for you)

[Show the code](#) | [Run the program](#)

So the correct PR for the instance is:



You can see it took about 20 iterations before the network began to settle on these values!

Look at Page D though - it has a PR of 0.15 even though no-one is voting for it (i.e. it has no incoming links)! Is this right?

The first part, or term to be technical, of the PR equation is doing this:

$$PR(A) = (1-d) + d (PR(T1)/C(T1) + \dots + PR(Tn)/C(Tn))$$

So, for Page D, no backlinks means the equation looks like this:

$$\begin{aligned} PR(A) &= (1-d) + d * (0) \\ &= 0.15 \end{aligned}$$

no matter what else is going on or how many times you do it.

Observation: every page has at least a PR of 0.15 to share out. But this may only be in theory - there are rumours that Google undergoes a post-spidering phase whereby any pages that have no incoming links at all are completely deleted from the index...

Instance 2

A simple hierarchy with some outgoing links

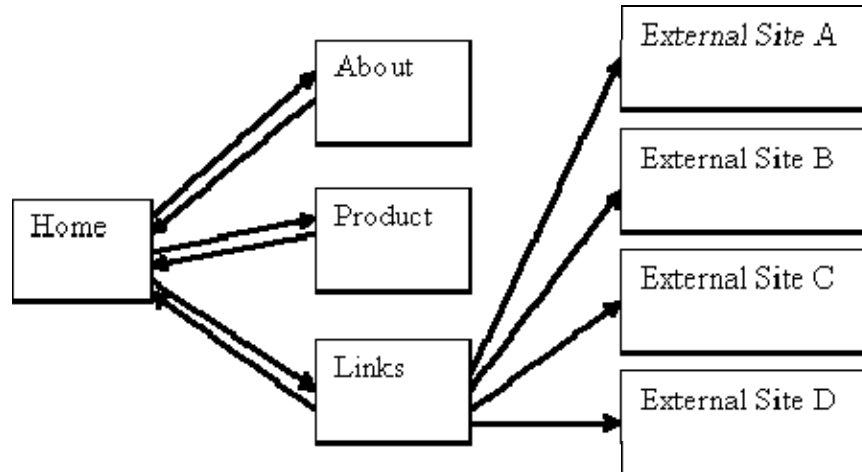


Fig 2. 9

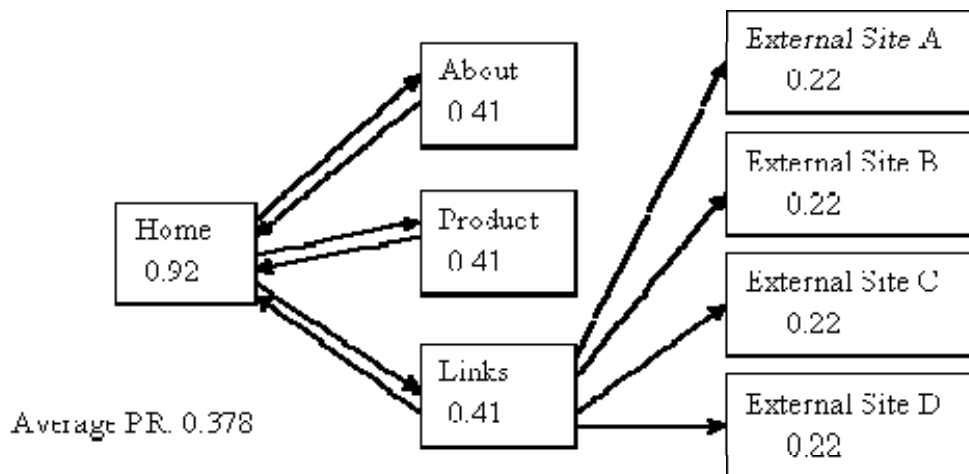


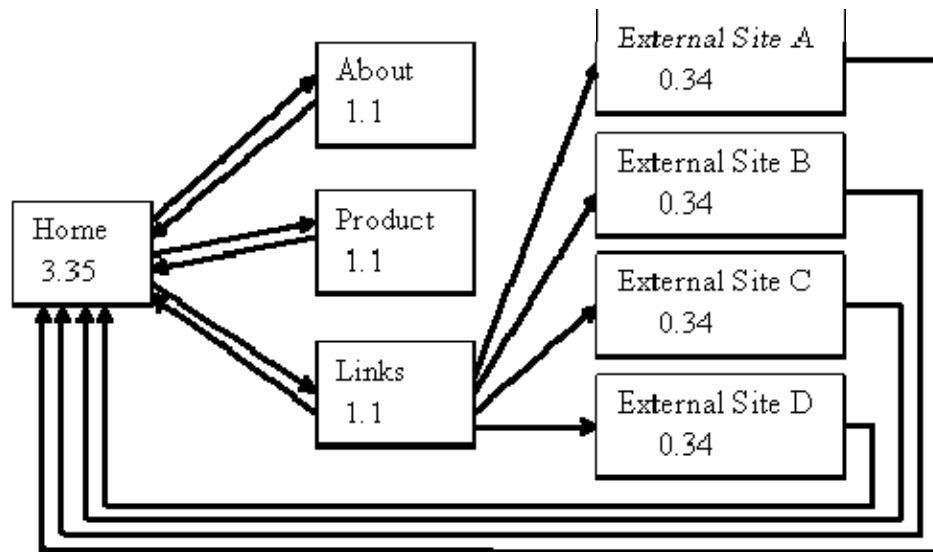
Fig 2. 10

As you'd expect, the home page has the most PR – after all, it has the most incoming links! But what's happened to the average? It's only 0.378!!! That doesn't tie up with what I said earlier so something is wrong somewhere!

Well no, everything is fine. But take a look at the external site pages – what's happening to their PageRank? They're not passing it on, they're not voting for anyone, they're wasting their PR like so much pregnant chad!!! (NB, a more accurate description of this issue can be found in this thread)

Instance 3

Let's link those external sites back into our home page just so we can see what happens to the average...



Average PR: 1.000

Fig 2. 11

That's better - it does work after all! And look at the PR of our home page! All those incoming links sure make a difference – we'll talk more about that later.

Instance 4

What happens to PR if we follow a suggestion about writing page reviews?

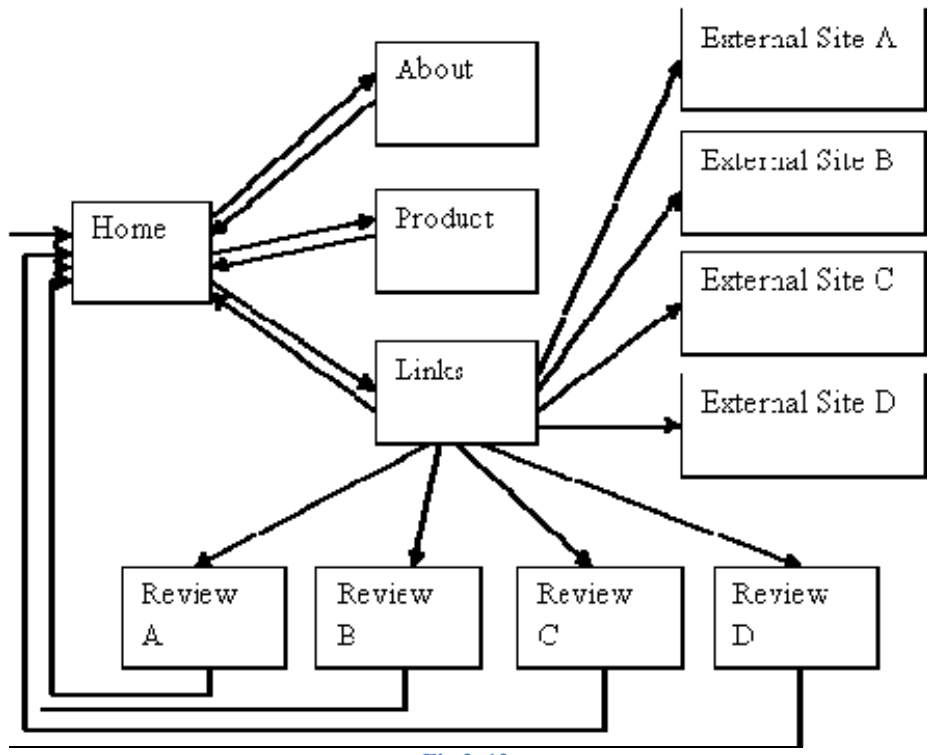


Fig 2. 12

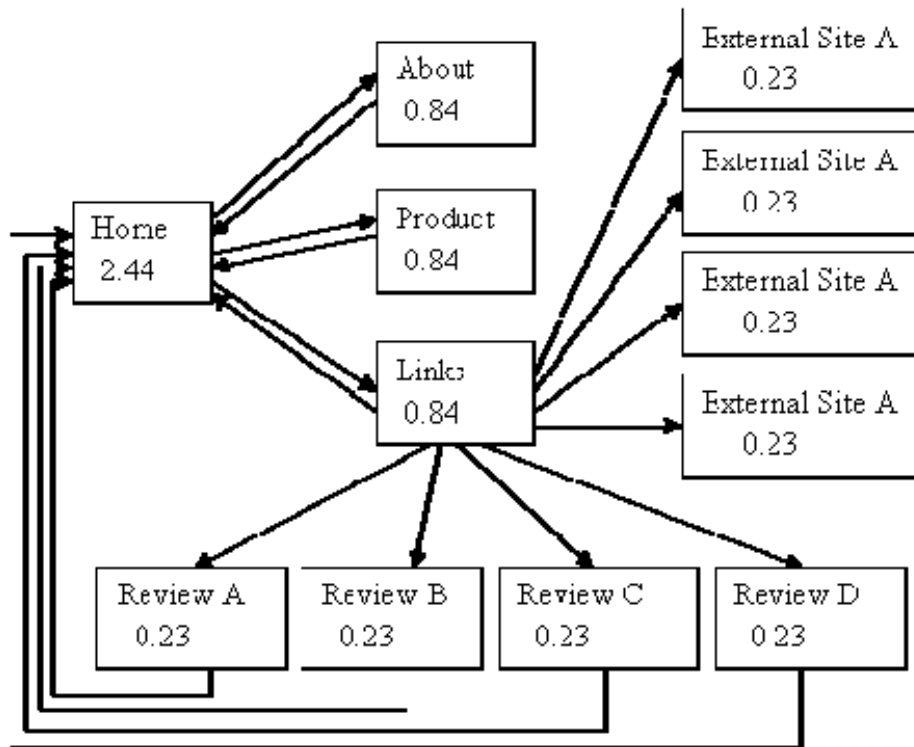


Fig 2. 13

Instance 5

A simple hierarchy

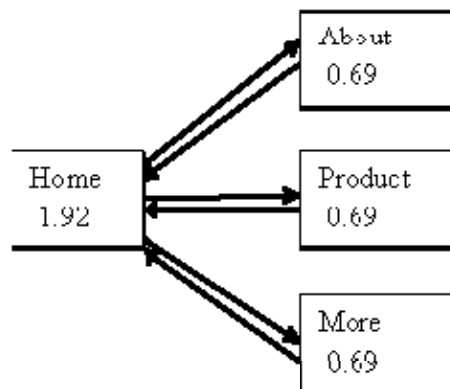


Fig 2. 14

Our home page has 2 and a half times as much PR as the child pages! Excellent!

- **Observation:** a hierarchy concentrates votes and PR into one page

Instance 6

Looping

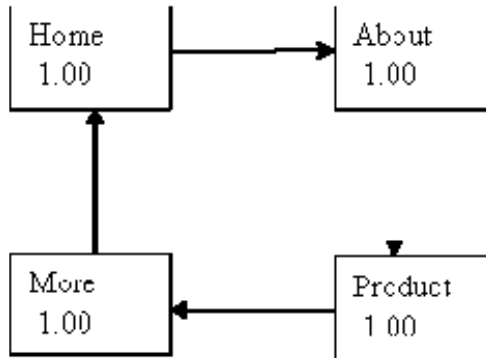


Fig 2. 15

This is what we'd expect. All the pages have the same number of incoming links, all pages are of equal importance to each other, all pages get the same PR of 1.0 (i.e. the average probability).

Instance 7

Extensive Interlinking – or Fully Meshed

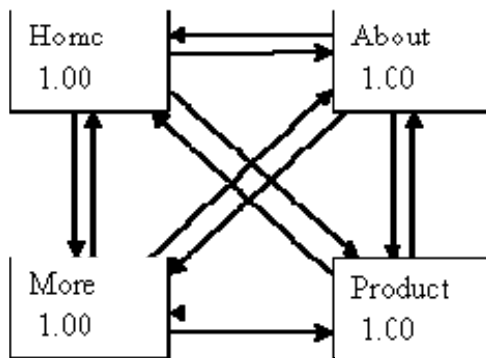


Fig 2. 16

Yes, the results are the same as the Looping instance above and for the same reasons.

Instance 8

Hierarchical – but with a link in and one out.

We'll assume there's an external site that has lots of pages and links with the result that one of the pages has the average PR of 1.0. We'll also assume the webmaster really likes us – there's just one link from that page and it's pointing at our home page.

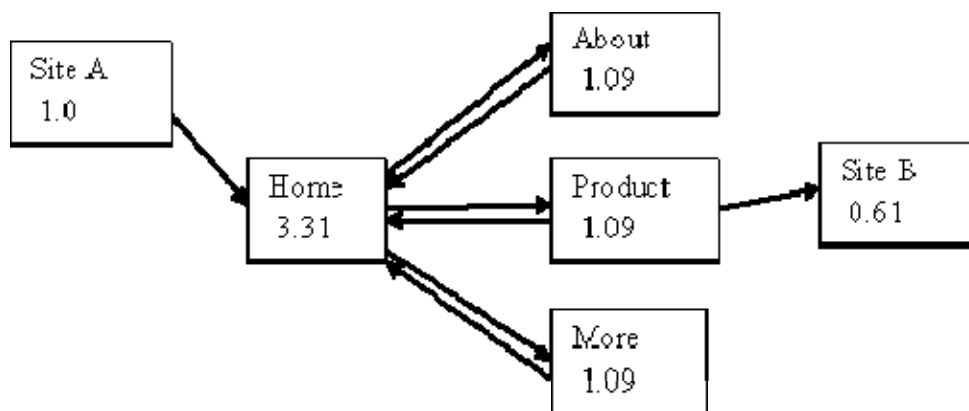


Fig 2. 17

In instance 5 the home page only had a PR of 1.92 but now it is 3.31! Excellent! Not only has site A contributed 0.85 PR to us, but the raised PR in the About, Product and More pages has had a lovely feedback effect, pushing up the home page's PR even further!

- Principle: a well structured site will amplify the effect of any contributed PR

Instance 9

Looping – but with a link in and a link out

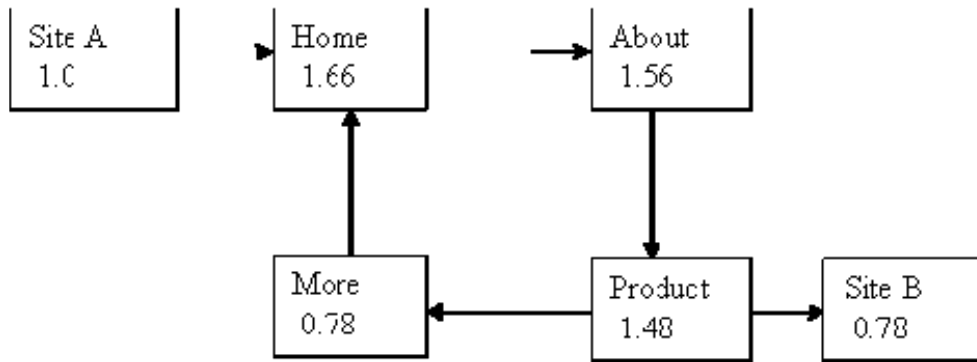


Fig 2.18

Well, the PR of our home page has gone up a little, but what's happened to the More page?

The vote of the Product page has been split evenly between it and the external site. We now value the external Site B equally with our More page. The More page is getting only half the vote it had before – this is good for Site B but very bad for us!

Instance 10

Fully meshed – but with one vote in and one vote out

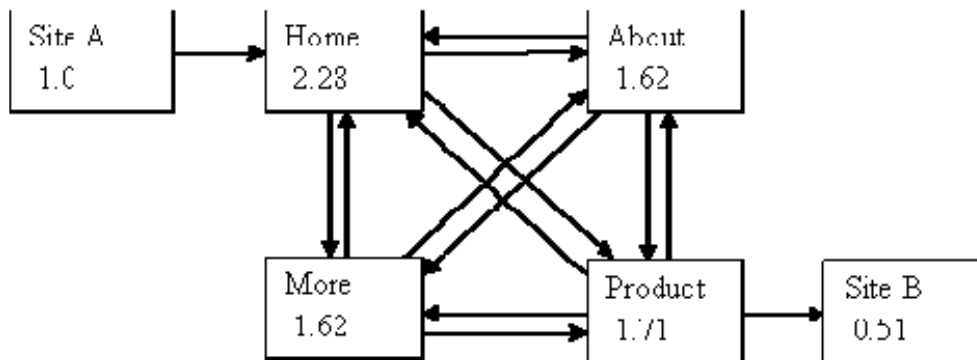


Fig 2.19

That's much better. The More page is still getting less share of the vote than in instance 7 of course, but now the Product page has kept three quarters of its vote within our site - unlike instance 10 where it was giving away fully half of its vote to the external site!

Keeping just this small extra fraction of the vote within our site has had a very nice effect on the Home Page too – PR of 2.28 compared with just 1.66 in instance 10.

- **Observation:** increasing the internal links in your site can minimise the damage to your PR when you give away votes by linking to external sites.
- **Principle:**
 1. If a particular page is highly important – use a hierarchical structure with the important page at the top.
 2. Where a group of pages may contain outward links – increase the number of internal links to retain as much PR as possible.
 3. Where a group of pages do not contain outward links – the number of internal links in the site has **no** effect on the site’s average PR. You might as well use a link structure that gives the user the best navigational experience.

Site Maps

Site maps are useful in at least two ways:

1. If a user types in a bad URL most websites return a really unhelpful 404 – page not found error page. This can be discouraging. Why not configure your server to return a page that shows an error has been made, but also gives the site map? This can help the user enormously
2. Linking to a site map on each page increases the number of internal links in the site, spreading the PR out and protecting you against your vote donations

Instance 11

Lets try to fix our site to artificially concentrate the PR into the home page.

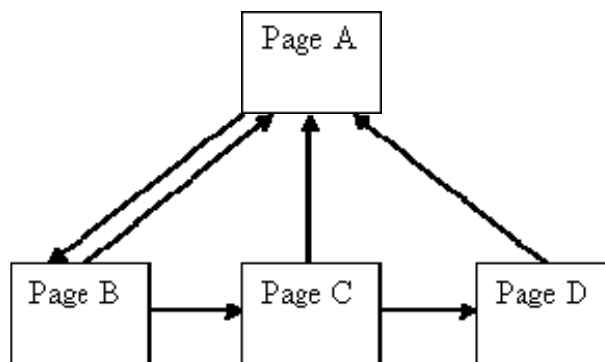


Fig 2. 20

That looks good, most of the links seem to be pointing up to page A so we should get a nice PR.

Try to guess what the PR of A will be before you scroll down or run the code.

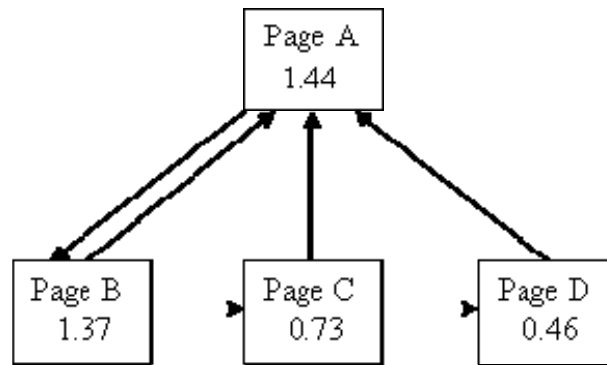


Fig 2. 21

Oh dear, that didn't work at all well – it's much worse than just an ordinary hierarchy! What's going on is that pages C and D have such weak incoming links that they're no help to page A at all!

- **Principle:** trying to abuse the PR calculation is harder than you think.

Instance 12

A common web layout for long documentation is to split the document into many pages with a Previous and Next link on each plus a link back to the home page. The home page then only needs to point to the first page of the document.

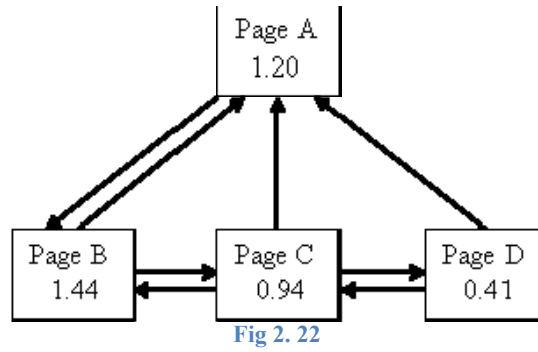


Fig 2. 22

In this simple instance, where there's only one document, the first page of the document has a higher PR than the Home Page! This is because page B is getting all the vote from page A, but page A is only getting fractions of pages B, C and D.

- **Principle:** in order to give users of your site a good experience, you may have to take a hit against your PR. There's nothing you can do about this - and neither should you try to or worry about it! If your site is a pleasure to use lots of other webmasters will link to it and you'll get back much more PR than you lost.

Can you also see the trend between this and the previous instance? As you add more internal links to a site it gets closer to the Fully Meshed instance where every page gets the average PR for the mesh.

- **Observation:** as you add more internal links in your site, the PR will be spread out more evenly between the pages.

Instance 13

Getting high PR the wrong way and the right way.

Just as an experiment, let's see if we can get 1,000 pages pointing to our home page, but only have one link leaving it...

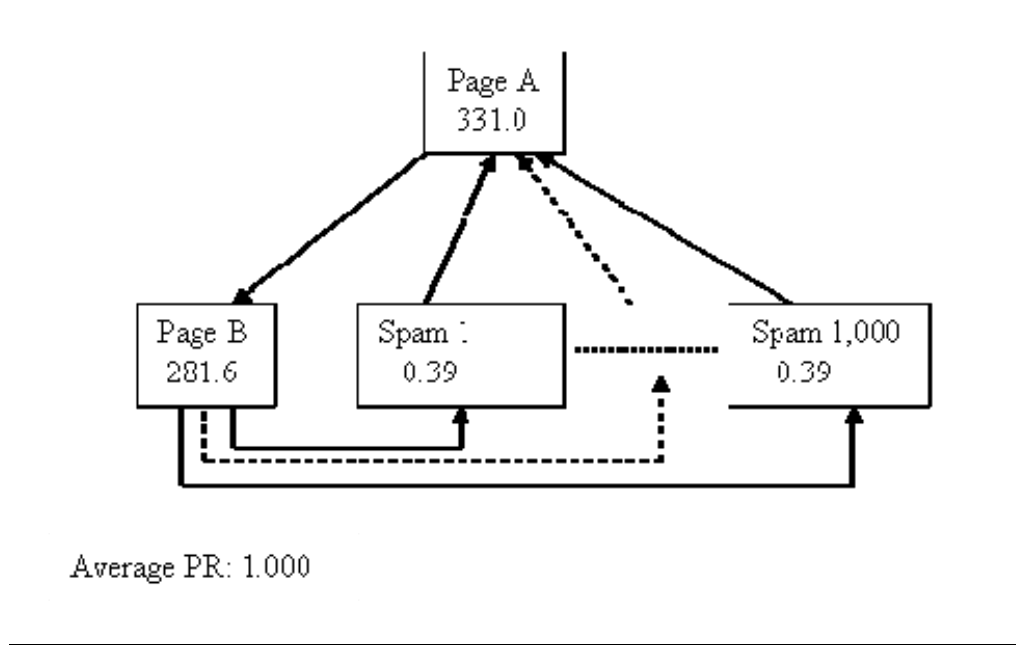


Fig 2. 23

Yup, those spam pages are pretty worthless but they sure add up!

- **Observation:** it doesn't matter how many pages you have in your site, your average PR will always be 1.0 at best. But a hierarchical layout can strongly concentrate votes, and therefore the PR, into the home page!

This is a technique used by some disreputable sites (mostly adult content sites). But I can't advise this - if Google's robots decide you're doing this there's a good chance you'll be banned from Google! Disaster!

On the other hand there are at least two right ways to do this:

1. Be a Mega-site

Mega-sites, like <http://news.bbc.co.uk> have tens or hundreds of editors writing new content – i.e. new pages - all day long! Each one of those pages has rich, worthwhile content of its own and a link back to its parent or the home page! That's why the Home page Toolbar PR of these sites is 9/10 and the rest of us just get pushed lower and lower by comparison...

- **Principle:** Content Is King! There really is no substitute for lots of good content...

2. Give away something useful

www.phpbb.com has a Toolbar PR of 8/10 (at the time of writing) and it has no big money or marketing behind it! How can this be?

What the group has done is write a very useful bulletin board system that is becoming very popular on many websites. And at the bottom of every page, in **every** installation, is this HTML code:

Powered by phpBB

The administrator of each installation can remove that link, but most don't because they want to return the favour...

Can you imagine all those millions of pages giving a fraction of a vote to www.phpbb.com? Wow!

- **Principle:** Make it worth other people's while to use your content or tools. If your give-away is good enough other site admins will gladly give you a link back.
- **Principle:** it's probably better to get lots (perhaps thousands) of links from sites with small PR than to spend any time or money desperately trying to get just the one link from a high PR page.

A Discussion on Averages

From the Brin and Page paper, the average Actual PR of all pages in the index is 1.0!

So if you add pages to a site you're building the total PR will go up by 1.0 for each page (but only if you link the pages together so the equation can work), but the average will remain the same.

If you want to concentrate the PR into one, or a few, pages then hierarchical linking will do that. If you want to average out the PR amongst the pages then fully meshing the site (lots of evenly distributed links) will do that - instances 5, 6, and 7 in my above. (NB. this is where Ridings' goes wrong, in his MiniRank model feedback loops will increase PR - indefinitely!)

Getting inbound links to your site is the only way to increase your site's average PR. How that PR is distributed amongst the pages on your site depends on the details of your internal linking and which of your pages are linked to.

If you give outbound links to other sites then your site's average PR will decrease (you're not keeping your vote in house as it were). Again the details of the decrease will depend on the details of the linking.

Given that the average of every page is 1.0 we can see that for every site that has an actual ranking in the millions (and there are some!) there must be lots and lots of sites who's Actual PR is below 1.0 (particularly because the absolute lowest Actual PR available is $(1 - d)$).

It may be that the Toolbar PR 1,2 correspond to Actual PR's lower than 1.0! E.g. the logbase for the Toolbar may be 10 but the Actual PR sequence could start quite low: 0.01, 0.1, 1, 10, 100, 1,000 etc...

Finally

PageRank is, in fact, very simple (apart from one scary looking formula). But when a simple calculation is applied hundreds (or billions) of times over the results can seem complicated.

PageRank is also only part of the story about what results get displayed high up in a Google listing. For instance there's some evidence to suggest that Google is paying a lot of attention these days to the text in a link's anchor when deciding the relevance of a target page – perhaps more so than the page's PR...

PageRank is still part of the listings story though, so it's worth your while as a good designer to make sure you understand it correctly.

Chapter-3 System Development

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 :

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

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 instance :

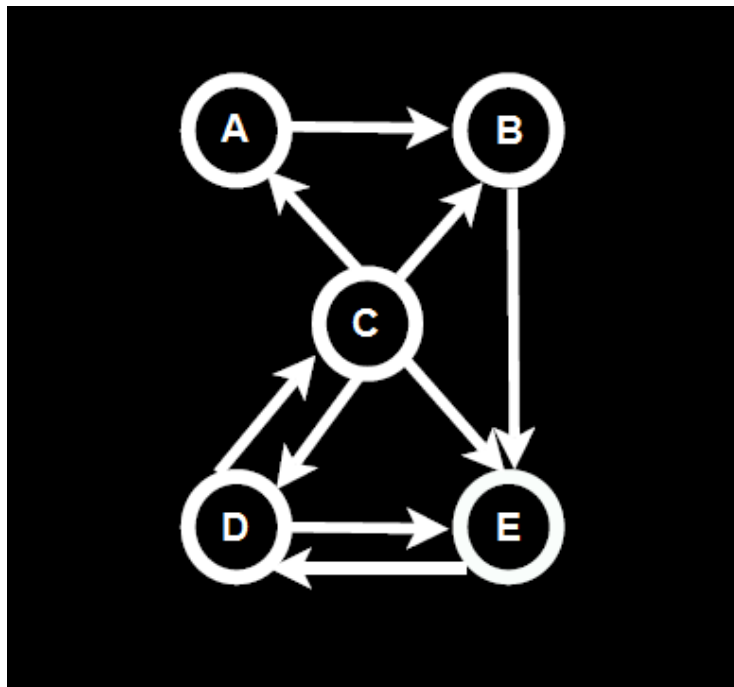


Fig 3. 1

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

Table 3.1

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 $1/4$ th 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

Table 3.2

Chapter -4 Performance Analysis

Output for n=5:

```
1001.
Enter the Number of Webpages

5
Enter the Adjacency Matrix 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
n value:5.0    init value :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 :    0.075000000000000001
Page Rank of 3 is :    0.125
Page Rank of 4 is :    0.375
Page Rank of 5 is :    0.1
```

After 2th Step

Page Rank of 1 is : 0.025
Page Rank of 2 is : 0.075000000000000001
Page Rank of 3 is : 0.125
Page Rank of 4 is : 0.375
Page Rank of 5 is : 0.4

After 3th Step

Page Rank of 1 is : 0.03125
Page Rank of 2 is : 0.05625
Page Rank of 3 is : 0.1875
Page Rank of 4 is : 0.43125
Page Rank of 5 is : 0.29375

BUILD SUCCESSFUL (total time: 70 minutes 37 seconds)

Output for n=3:

```
run:
Enter the Number of Webpages

3
Enter the Adjacency Matrix between two Webpages:

1 0 0
0 1 0
0 0 1
n value:3.0    init value :0.3333333333333333

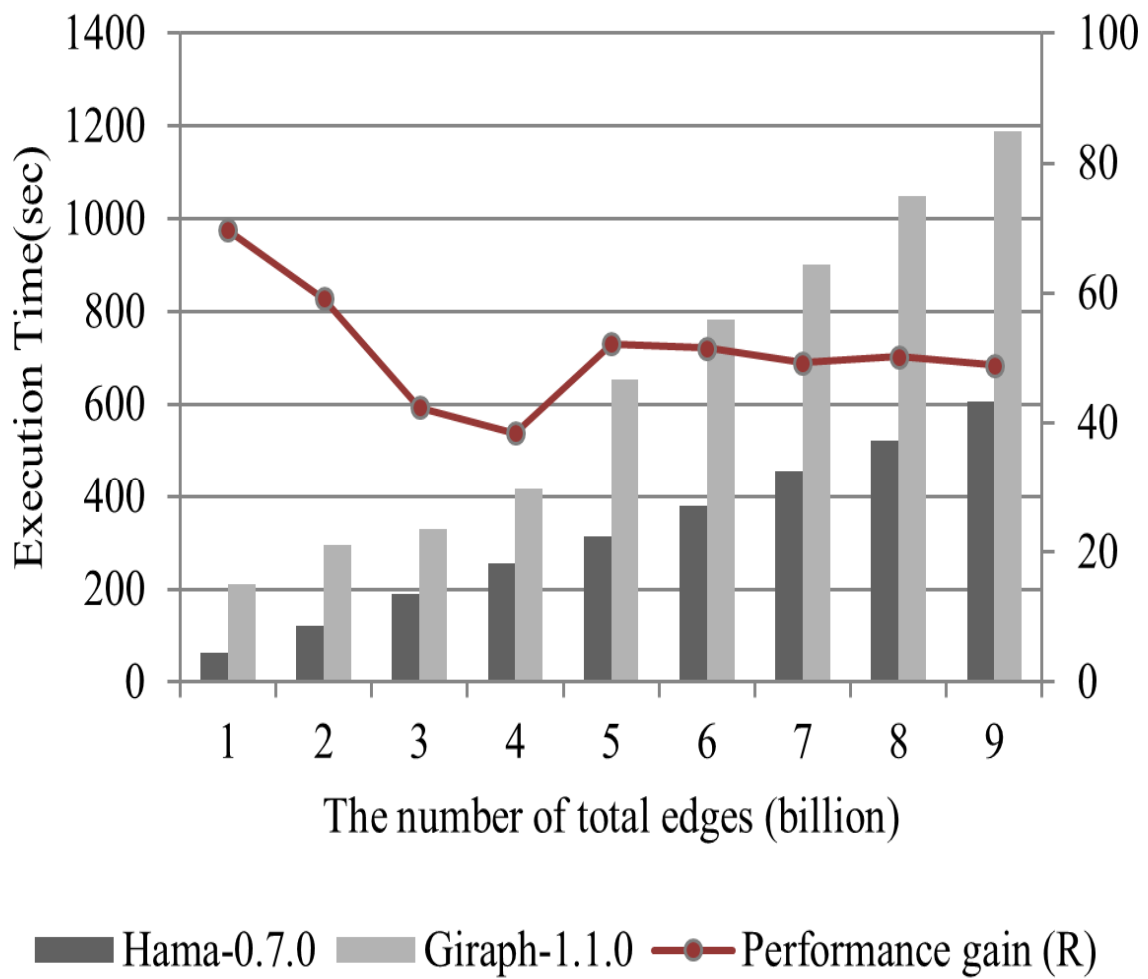
Initial Pagerank Values , 0th Step
Page Rank of 1 is :    0.3333333333333333
Page Rank of 2 is :    0.3333333333333333
Page Rank of 3 is :    0.3333333333333333

After 1th Step
Page Rank of 1 is :    0.0
Page Rank of 2 is :    0.0
Page Rank of 3 is :    0.0

After 2th Step
Page Rank of 1 is :    0.0
Page Rank of 2 is :    0.0
Page Rank of 3 is :    0.0

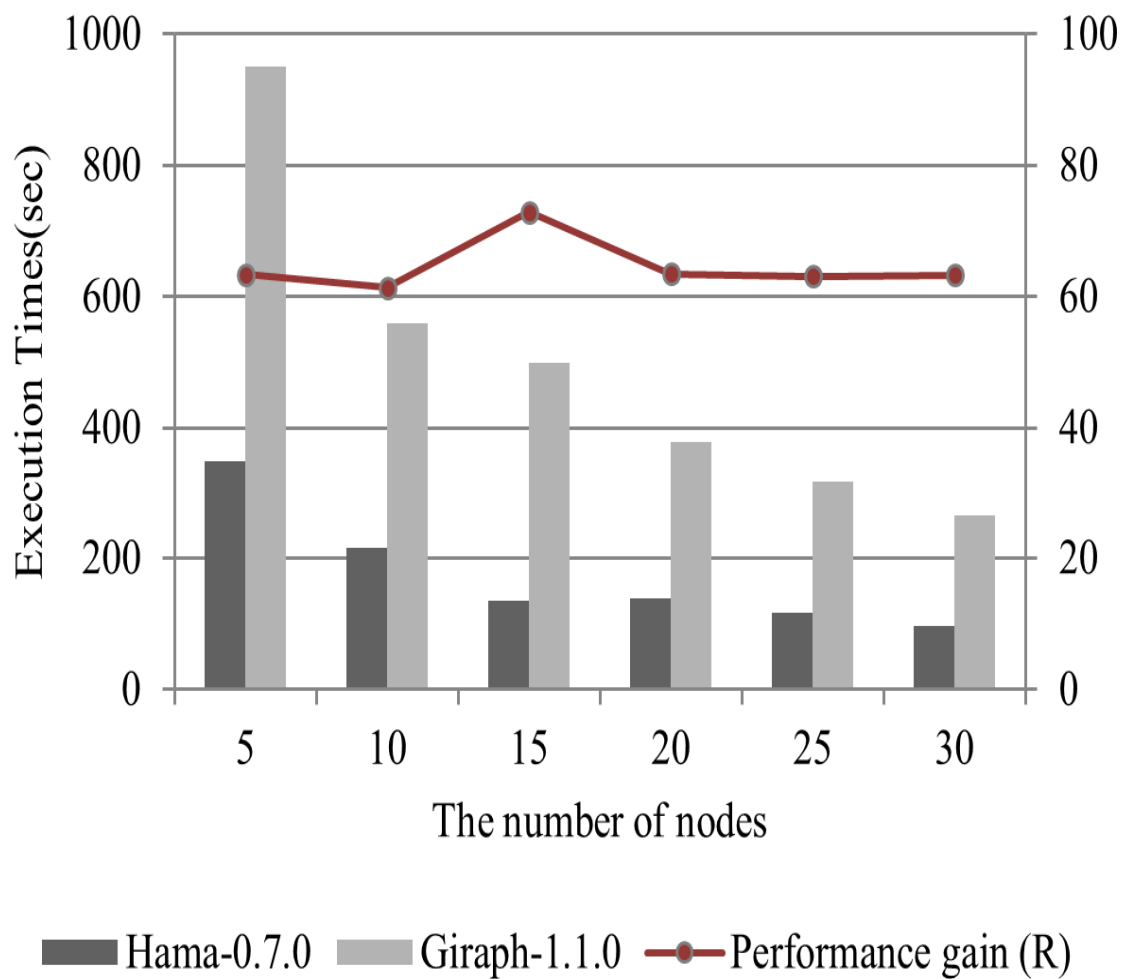
After 3th Step
Page Rank of 1 is :    0.0
Page Rank of 2 is :    0.0
Page Rank of 3 is :    0.0
BUILD SUCCESSFUL (total time: 19 seconds)
|
```

The graph of the number of edges to the execution time:



Graph 1

The graph of the number of links to the execution time :



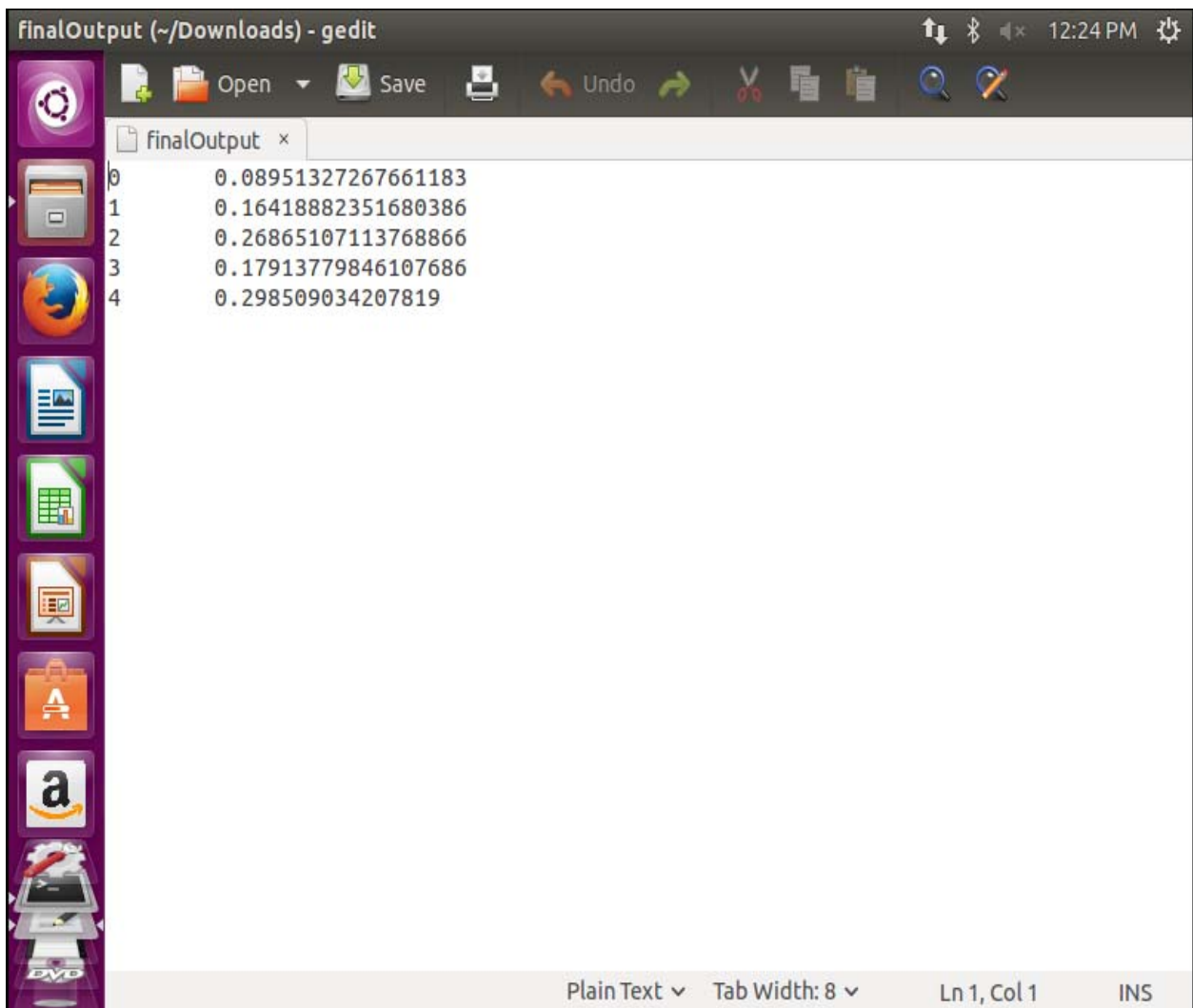
Graph 2

SVD Output:

```
[[0 0 0 0 0 0]
 [0 0 1 0 0 0]
 [0 0 0 0 0 1]
 [0 1 1 0 1 1]
 [0 0 0 1 0 1]
 [0 0 0 0 1 0]]
[[ 0.      0.      0.      0.      0.      0.      ]
 [ 0.      0.25900127  0.37950064 -0.14544289  0.37950064  0.05677919]
 [ 0.      0.11355838  0.05677919  0.40444417  0.05677919  0.75900127]
 [ 0.      0.81578046  1.07478173  0.05677919  1.07478173  0.98611803]
 [ 0.      0.05677919 -0.0886637  0.70222208 -0.0886637  1.16344544]
 [ 0.      0.25900127  0.37950064 -0.14544289  0.37950064  0.05677919]]
n value 5
initital value 0.2

initial page rank value, at 0th step -> [0, 0.2, 0.2, 0.2, 0.2, 0.2, 0, 0, 0, 0]
initial page rank value, at 1th step -> [0, 0.05, 0.25, 0.1, 0.25, 0.35, 0, 0, 0, 0]
initial page rank value, at 2th step -> [0, 0.025, 0.07500000000000001, 0.125, 0.375, 0.4, 0, 0, 0, 0]
initial page rank value, at 3th step -> [0, 0.03125, 0.05625, 0.1875, 0.43125, 0.29375, 0, 0, 0, 0]
[Finished in 0.268s]
```

Output in Hadoop:



The screenshot shows a gedit text editor window titled "finalOutput (~/Downloads) - gedit". The window contains a single tab named "finalOutput". The text in the editor is as follows:

```
0 0.08951327267661183
1 0.16418882351680386
2 0.26865107113768866
3 0.17913779846107686
4 0.298509034207819
```

The status bar at the bottom of the window displays "Plain Text", "Tab Width: 8", "Ln 1, Col 1", and "INS".

Chapter-5 Conclusion

5.1 Conclusions

The above developed model successfully generates pure page ranks for given 'n' number of webpages.

The model also shows the ability to use these page ranks as per requirements.

5.2 Future Scope

Future scope of development include the following:

5.2.1 Improved Algorithm

The present algorithm can be used to generate more pageranks by efficient utilization of big data.

Applications:

The use of PageRank is no way restricted to Search Engines. Here are a few other uses of PageRank :

1. Finding how well connected a person is on **Social Media** : One of the unexplored territory in social media analytics is the network information. Using this network information we can estimate how influential is the user. And therefore prioritize our efforts to please the most influential customers. Networks can be easily analyzed using Page Rank algorithm.
2. Fraud Detection in **Pharmaceutical industry** : Many countries including US struggle with the problem of high percentage medical frauds. Such frauds can be spotted using Page Rank algorithm.
3. Understand the importance of **packages** in any programming language : Page Rank algorithm can also be used to understand the layers of packages used in languages like R and Python.

Chapter -6. Appendix

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;
int ITERATION_STEP=1;

InitialPageRank = 1/totalNodes;
System.out.printf(" Total Number of Nodes :"+totalNodes+"\t Initial PageRank of All Nodes :"+InitialPageRank+"\n");

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)
{
```

```

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;
            while(k<=totalNodes)
            {
                if(this.path[ExternalNodeNumber][k] == 1 )
                {
                    OutgoingLinks=OutgoingLinks+1;
                }
                k=k+1;
            }

            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;
}

```

```

for(k=1;k<=totalNodes;k++)
{
    this.pagerank[k]=(1-DampingFactor)+ DampingFactor*this.pagerank[k];
}

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);

}

}

```

SVD Code in Python:

```
import numpy as np
LA = np.linalg

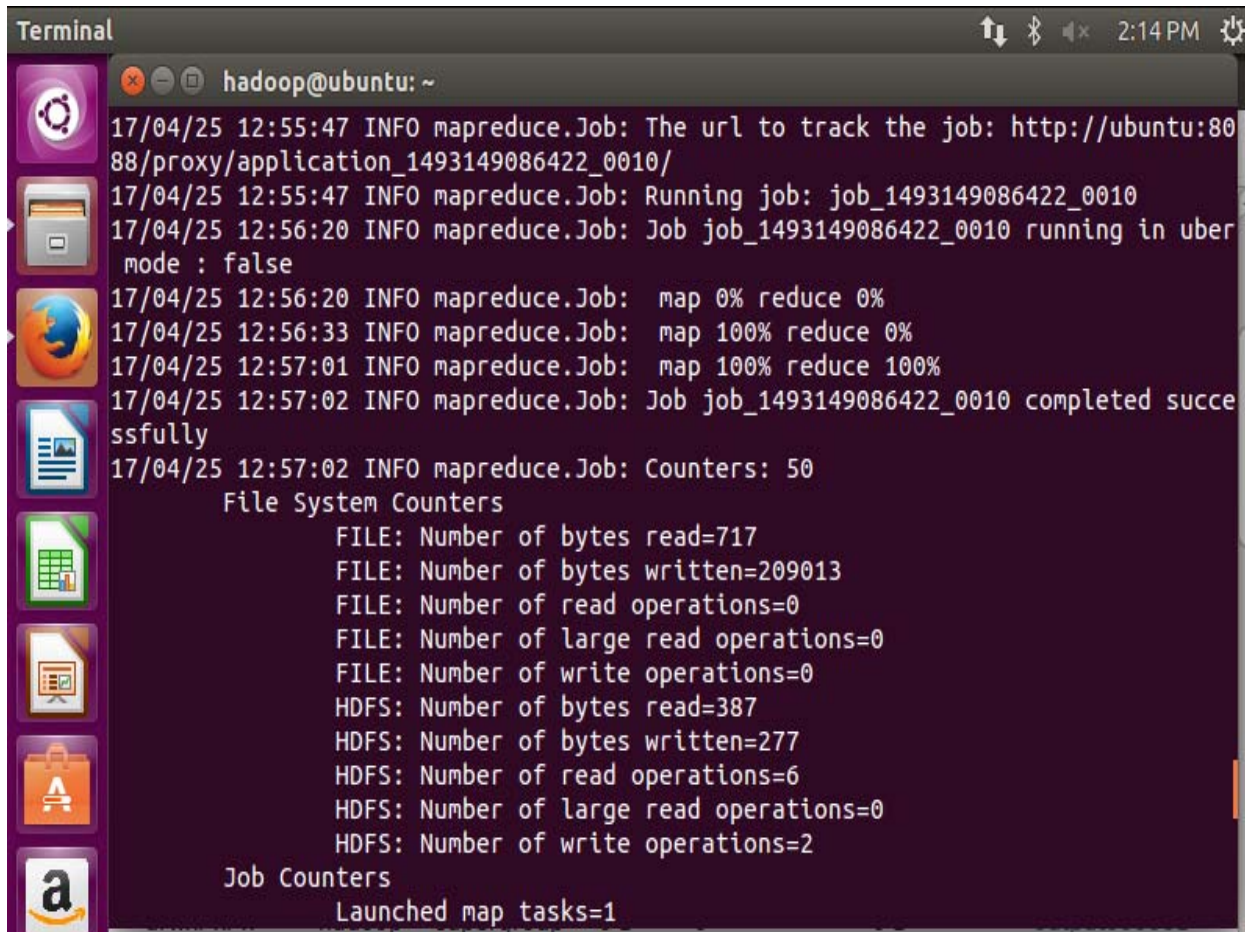
a = np.array([ [0, 0, 0, 0, 0, 0],
               [0, 0, 1, 0, 0, 0],
               [0, 0, 0, 0, 0, 1],
               [0, 1, 1, 0, 1, 1],
               [0, 0, 0, 1, 0, 1],
               [0, 0, 0, 0, 1, 0]])
print(a)

U, s, Vh = LA.svd(a, full_matrices=False)
assert np.allclose(a, np.dot(U, np.dot(np.diag(s), Vh)))

s[2:] = 0
path = np.dot(U, np.dot(np.diag(s), Vh))
print(path)

n=5
path = [ [0, 0, 0, 0, 0, 0],
         [0, 0, 1, 0, 0, 0],
         [0, 0, 0, 0, 0, 1],
         [0, 1, 1, 0, 1, 1],
         [0, 0, 0, 1, 0, 1],
         [0, 0, 0, 0, 1, 0]]
init = 1/float(n)
pagerank = [0,0,0,0,0, 0,0,0,0,0]
temp = [0,0,0,0,0, 0,0,0,0,0]
c=0
print ( n value + str(n) + \ninitital value + str(init) + \n )
for i in range (1,n+1):
    pagerank[i]=init
print ( initial page rank value, at 0th step -> + str(pagerank))
for u in range(1,3+1):
    for i in range (1, n+1):
        temp[i]=float(pagerank[i])
        pagerank[i]=float(0)
    for j in range (1,n+1):
        for i in range (1,n+1):
            if (path[i][j]==1):
                c=0
                for k in range (1,n+1):
                    if(path[i][k]==1):
                        c+=1
                pagerank[j]+=(temp[i]*(1/float(c)))
    print ( initial page rank value, at +str(u)+ th step -> + str(pagerank))
```

Simulation in Hadoop:



The image shows a terminal window titled "Terminal" with the prompt "hadoop@ubuntu: ~". The window displays the output of a Hadoop MapReduce job. The logs indicate that the job is running in uber mode, with map and reduce progress reaching 100% by 12:57:01. The job completed successfully at 12:57:02. The output includes counters for File System and Job, showing metrics such as bytes read/written, read/write operations, and launched map tasks.

```
17/04/25 12:55:47 INFO mapreduce.Job: The url to track the job: http://ubuntu:8088/proxy/application_1493149086422_0010/
17/04/25 12:55:47 INFO mapreduce.Job: Running job: job_1493149086422_0010
17/04/25 12:56:20 INFO mapreduce.Job: Job job_1493149086422_0010 running in uber mode : false
17/04/25 12:56:20 INFO mapreduce.Job:  map 0% reduce 0%
17/04/25 12:56:33 INFO mapreduce.Job:  map 100% reduce 0%
17/04/25 12:57:01 INFO mapreduce.Job:  map 100% reduce 100%
17/04/25 12:57:02 INFO mapreduce.Job: Job job_1493149086422_0010 completed successfully
17/04/25 12:57:02 INFO mapreduce.Job: Counters: 50
  File System Counters
    FILE: Number of bytes read=717
    FILE: Number of bytes written=209013
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=387
    HDFS: Number of bytes written=277
    HDFS: Number of read operations=6
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=2
  Job Counters
    Launched map tasks=1
```

```
Terminal
hadoop@ubuntu: ~
HDFS: Number of large read operations=0
HDFS: Number of write operations=2
Job Counters
  Launched map tasks=1
  Launched reduce tasks=1
  Data-local map tasks=1
  Total time spent by all maps in occupied slots (ms)=9314
  Total time spent by all reduces in occupied slots (ms)=24572
  Total time spent by all map tasks (ms)=9314
  Total time spent by all reduce tasks (ms)=24572
  Total vcore-seconds taken by all map tasks=9314
  Total vcore-seconds taken by all reduce tasks=24572
  Total megabyte-seconds taken by all map tasks=9537536
  Total megabyte-seconds taken by all reduce tasks=25161728
Map-Reduce Framework
  Map input records=5
  Map output records=30
  Map output bytes=651
  Map output materialized bytes=717
  Input split bytes=109
  Combine input records=0
  Combine output records=0
  Reduce input groups=5
  Reduce shuffle bytes=717
```

```
Terminal 2:16 PM
hadoop@ubuntu: ~
GC time elapsed (ms)=484
CPU time spent (ms)=6180
Physical memory (bytes) snapshot=329367552
Virtual memory (bytes) snapshot=1333907456
Total committed heap usage (bytes)=168038400
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=277
File Output Format Counters
  Bytes Written=278
17/04/25 12:58:32 INFO pagerank.PageRank: Counter Value : 0
17/04/25 12:58:32 INFO pagerank.PageRank: Final Page Rank Input File Exists. Delet
ing it
17/04/25 12:58:32 INFO pagerank.PageRank: COUNTER = 5
17/04/25 12:58:33 INFO pagerank.PageRank: Final Page Rank File Created
17/04/25 12:58:33 INFO pagerank.PageRank: Check the Final Output in the path /fi
nalOutput
hadoop@ubuntu:~$
```

References :

- A Survey of Google's PageRank

Reference URL: <http://pr.efactory.de/>

Date of reference: 15 November 2016

- PageRank Lecture Note by Keshi Dai

Date of issue: June 22, 2009

Reference URL: <http://www.ccs.northeastern.edu/home/daikeshi/notes/PageRank.pdf>

- Study of PageRank Algorithms by Tanmay

Date of reference: 20 December 2016

ReferenceURL: <http://www.cs.sjsu.edu/faculty/pollett/masters/Semesters/Fall11/tanmayee/Deliverable3.pdf>

- The Google Pagerank Algorithm and How It Works by Ian Rogers IPR Computing Ltd

Date of reference: 25 January 2017

ReferenceURL: www.cs.princeton.edu/~chazelle/courses/BIB/pagerank.htm