FUNDEMENTALS OF GOOGLE SEARCH ENGINE

DONE BY:

B.E. PRANAV KUMAAR

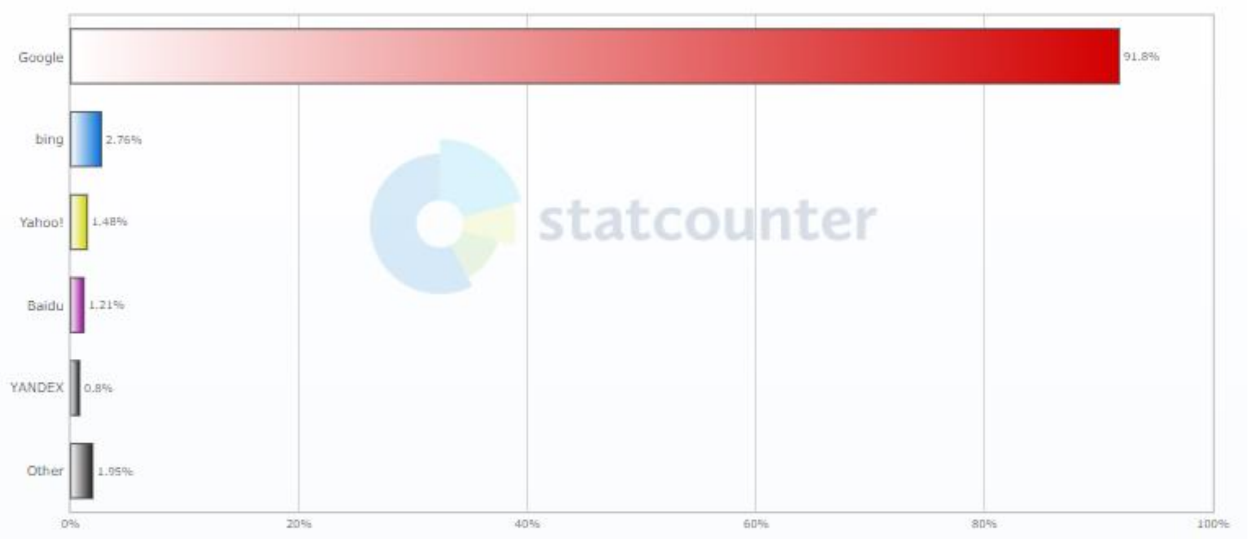
INTRODUCTION:

What is a search engine?

A search engine is a software system that is designed to carry out web searches (Internet searches), which means to search the World Wide Web(www) in a systematic way for particular information specified in a textual web search query. The search results are generally presented in a line of results, often referred to as search engine results pages (SERPs) The information may be a mix of links to web pages, images, videos, infographics, articles, research papers, and other types of files. Search engines also order the SERP's accounting for various factors that are user specific.

In short, a search engine is a tool that allows users to navigate the web to obtain the information they were looking for in an efficient manner.

A quick search shows the following information on the search engine market share over the last 3 months (Nov 2020 - Jan 2021) globally,



From the above chart we can see that google is the indisputable champion having more than 90% of the entire share. It is also individually the most used search engine in every country around the world except only to baidu in China. Needless to say, Google has had a significant lead for long time now.

Now let's see what factors distinguishes google from other search engines that allow it to remain so dominant. For understanding this we can start by looking at the creation of google.

History of Google's Search Engine:

The Web search engine company Google, Inc, was founded by Larry Page and Sergey Brin in 1998. But not long before Page had been working on a web project during his grad years which exploited the link structure of the web which was called BackRub and soon after he met Brin in 1995 who found his work exciting and realized the potential of creating a search engine capable of adapting to the increasing size of the web he hoped on board. Initially they tried to convincing existing companies to adopt their technology as they were certain that it was far superior to the older ones but found no success. Then with small financial assistance from investors they started their own company that currently rules the web navigation.

Google's **higher quality of their SERP's** impressed the academic Web search community and pleased the general public (who had access to internet at the time) that those of other search engines at the time. At the time most search engines based their results only on the content of a web page which can be easily manipulated by web developers if they understood what factors play a role, spamming it and keeping it hidden (suppose the site matched the search query word to word and it was ranked based on frequency of search term then the web developers can add any number of web tags to increase their score but this said nothing about the validity, authenticity and relevance). To account for this, they observed the web's properties (link structure) and took advantage of it with some clever mathematics (matrix theory, numerical analysis, information retrieval, and graph theory) and applied by computer science to enable them to come up with more efficient ranking algorithms this was named PageRank after Larry Page. During the processing of a query, Google's search algorithm combined precomputed PageRank scores with text matching scores to obtain an overall ranking score for each webpage.

Now the goal of a web developer to create an ideal webpage is to attain maximum PageRank score as it serves as the heart of googles rankings. as a result, the exact workings of the Google algorithm are not disclosed in detail but we can understand the process followed to gain a better insight.

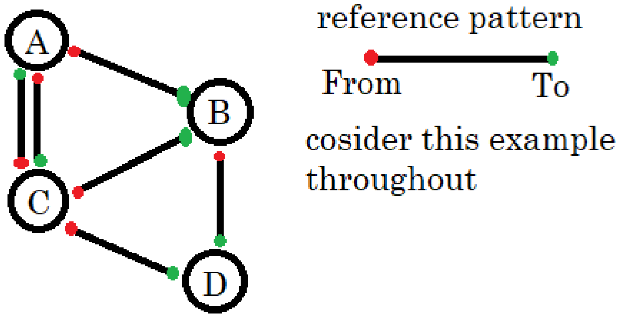
The Working of PageRank:

Logical analysis:

From the POV of any user we see that he/she will find the SERP's more useful if it gives him/her the most useful information (the website he/she is most likely will find what he/she wants and spend most time given he/she spends equal time viewing each webpage (this is later achieved by a probability) on it logically speaking if he/she likes what he/she is given). This verifies that the algorithm can indeed rank on some probabilistic basis. To find how a user will behave, the algorithm models the behavior of an idealized random Web surfer sometimes called Web crawlers (further referred to as object). This object is subjected to surfing the web endlessly (because an idealized person will never grow tired and this rules out "starting luck" if enough repetitions and is the goal of the web - to make the user surf as much as possible (if he likes what he/she is given he will naturally do so)) either by using the links in the current webpage to move to the next one or by entering the link of the webpage he likes to visit next (ex: one that his friend told him/her) with no memory of the previous websites visited. Doing this also means that he will not be influenced by pat websites which is vital because we are ranking websites as a whole before subjecting it to be user specific.

Thus, the PageRank score of a webpage represents the probability that a random Web surfer chooses to view the webpage. We will now see that all of this is possible due to the webs structure.

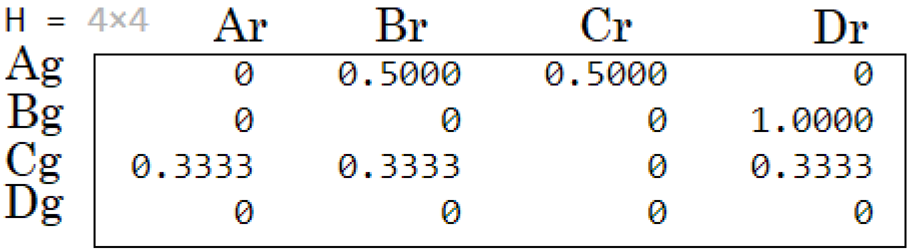
Structure of the Web:



This figure shows and example structure (one such possible structure) of the web reduced down to 4webpages – A, B, C, D this can be further used to demonstrate some concepts.

This view or manipulation of the web is called as Directed Web Graph (DWG). This visualization lets us represent the webpages as nodes (circles A, B, C, D) and the links used to traverse the web are edges (all connecting patterns). This type of visualization can be applied to any scale.

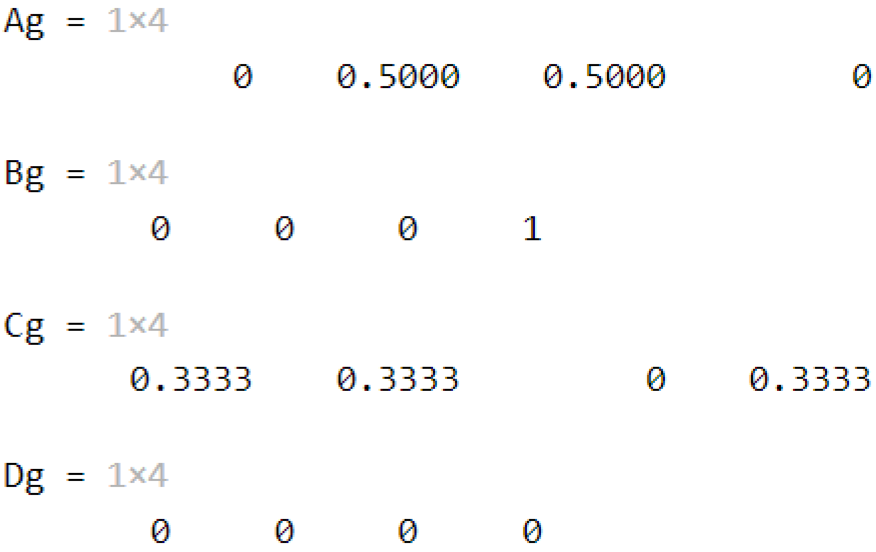
Mathematical analysis:



H = 4×4

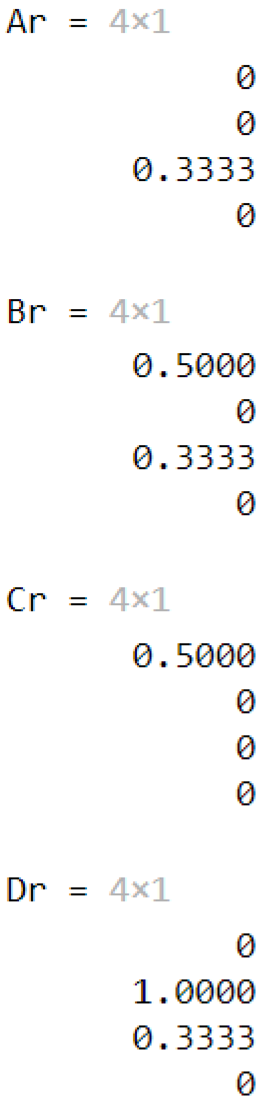
	Ar	Br	Cr	Dr
Ag	0	0.5000	0.5000	0
Bg	0	0	0	1.0000
Cg	0.3333	0.3333	0	0.3333
Dg	0	0	0	0

This represents the H matrix for above mentioned model



```
Ag = 1×4

         0    0.5000    0.5000         0

Bg = 1×4

     0     0     0     1

Cg = 1×4

    0.3333    0.3333         0    0.3333

Dg = 1×4

     0     0     0     0
```

This image shows the ratio of count of red dots



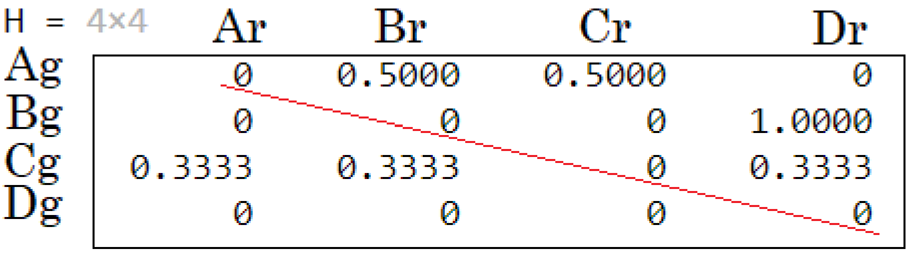
```
Ar = 4×1

         0
         0
    0.3333
         0

Br = 4×1

    0.5000
         0
    0.3333
         0

Cr = 4×1

    0.5000
         0
         0
         0

Dr = 4×1

         0
    1.0000
    0.3333
         0
```

This image shows the ratio of count of green dots

We can use these individual vectors to verify the H matrix generated

The method used to determine the PageRank begins in representing the Web in a more mathematically workable manner (so it is easier to perform operations and understand the interaction between various websites). We do this by expressing the DWG as a *n x n* “hyperlink matrix”, H, where *n* represents the number of webpages in the DWG. If we consider any webpage *i* and that it has at least one link to another webpage *j*, then the element in the row *i* and column *j* of H_{ij} = 1/ (no of links given away by page *i* or no of red dots) and if it does not have any link then H_{ij} = 0 (no of red dots = 0). By filling the matrix in the above order, the column represents the link of the webpage found in other webpages (the endorsements that they received) and the row represents the links of other webpages that the given webpage referenced in their webpage (the endorsements that they gave).

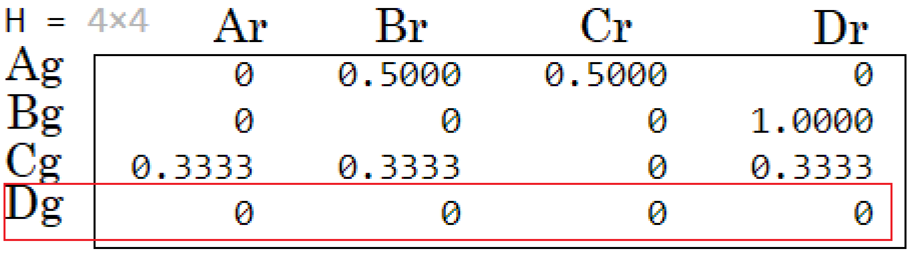
The H*ij* also account for the fact that links are more meaningful if less of them are given as the sum of columns of each row adds up to 1 (this makes it so that a website can split between 1 giving score).



$$H = 4\times4 \quad \begin{array}{c|cccc} & Ar & Br & Cr & Dr \\ \hline Ag & 0 & 0.5000 & 0.5000 & 0 \\ Bg & 0 & 0 & 0 & 1.0000 \\ Cg & 0.3333 & 0.3333 & 0 & 0.3333 \\ Dg & 0 & 0 & 0 & 0 \end{array}$$

We observe from this matrix that all diagonal elements are 0 as it represents the links of the website given and received by itself which is not possible as websites never link to themselves (adds no meaning)

Next let us consider a scenario of a dangling node.



$$H = 4\times4 \quad \begin{array}{c|cccc} & Ar & Br & Cr & Dr \\ \hline Ag & 0 & 0.5000 & 0.5000 & 0 \\ Bg & 0 & 0 & 0 & 1.0000 \\ Cg & 0.3333 & 0.3333 & 0 & 0.3333 \\ Dg & 0 & 0 & 0 & 0 \end{array}$$

A dangling node represents a webpage that references no other webpage or that there are no links to continue to other webpages from it (in this case D is a dangling node). This In the matrix will be a row of zeros. Since majority of websites are dangling nodes, the object has two options

1. Stop browsing (not possible – since he is an ideal browser)
2. Enter a custom URL of any other random webpage in the address line of a Web browser.

H alone cannot model the possibility of moving from the dangling node webpages to other webpages, so we add another term to fix this issue.

0-vector fix:

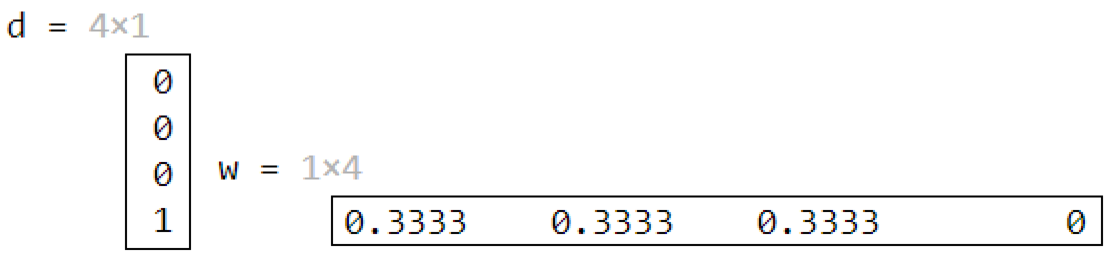
This is also called dangling node fix by some people. As mentioned, earlier we do not know which method is employed by google to fix this problem, but we can take a shot at the dark by saying whatever method it does use it solves the purpose. Let us look at some possible ways we can resolve this issue.

Method 1:

Each 0-vector row of H is replaced by an equal probability distribution vector, w, a vector with non-negative elements that sum to 1(all elements being 1/n except the one that represents itself which is 0 – cannot link to itself) (this removes any unwanted favorability)

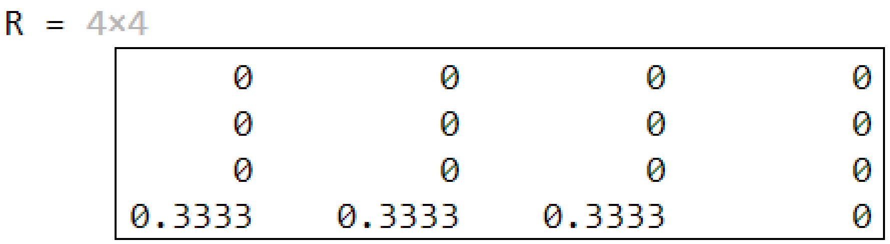
The resulting matrix Is of the form

S = H + R



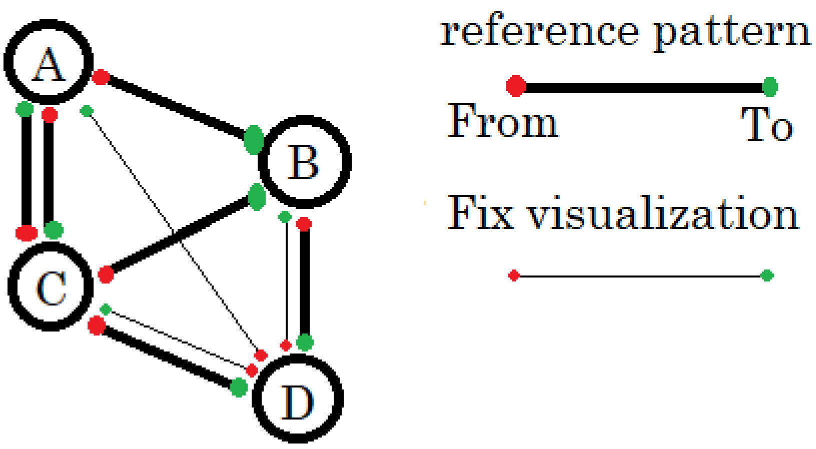
```
d = 4×1
        0
        0
        0        w = 1×4
        1
                     0.3333    0.3333    0.3333         0
```

R = d*w



```
R = 4×4
            0         0         0         0
            0         0         0         0
            0         0         0         0
       0.3333    0.3333    0.3333         0
```

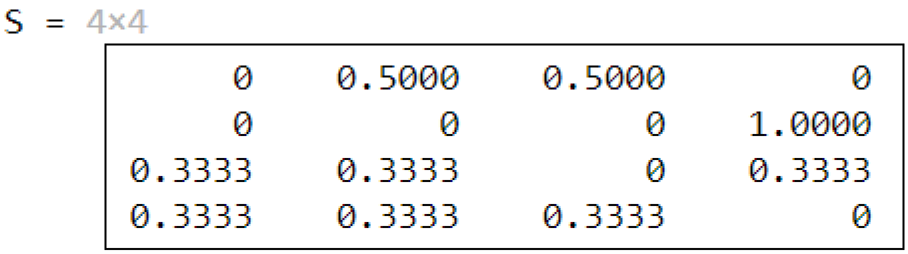
Intuitively we can see that this matrix R solves dangling node by providing valid giving vector. (gives D a bunch of red points)



*we can note here that this is a for of adapting term

Here d is a column vector the identifies dangling nodes, d is filled by the condition that if all elements of a row is 0 then the value of the corresponding element is 1 otherwise it is 0.

What this allows us to do is by matrix multiplication it generates the missing terms without changing the old ones



```
S = 4×4

        0    0.5000    0.5000         0
        0         0         0    1.0000
   0.3333    0.3333         0    0.3333
   0.3333    0.3333    0.3333         0
```

- This is valid and can be verified by the diagonal elements giving us a reference

General procedure of this method:

A set of closely related topics and their ranks are given more preference that is determined by a more wholesome calculation. And the term is more or less like the previous one which is then added to get S.

S = H + adapting term

Even after we deal with the 0-vector we need to address another possible way a real user would navigate through the web, he/she can also just hop from one site to another without using the links by using URLs and now we see how we can handle that situation.

Hopping fix:

The general form of the matrix that accounts for hopping behavior of users is represented by G. we can also account for another feature of the search engine by making it user specific (we need to make adjustments for users based on their personal preferences on using the search engine) and this is done by using a personalization vector, v. This uses data from previous searches of its users to keep adjusting its values giving more frequently visited websites better probabilities than those less frequent.

V-vector is used to mimic user preferences. (the data required for these changes can be made to a ratio and appended to are stored in the cloud segment of each user)

The hopping is a common behavior for most users and it also requires and experimental term with a ratio(value). This term is referred to as damping factor, a (scalar; 1>a>=0), in the matrix indicates that random Web surfer move to a different webpage by some means other than selecting a link with probability 1-a.

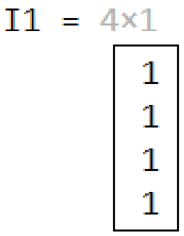
Again, by intuition we can say that ‘a’ must be greater than 0.5 but what values must we use to make the algorithm most efficient?

The only possible way to know the prefect value is through trial and error and seeing what works best. Brin and Page during development of PageRank algorithm used a = 0.85 and v to be another w just to test it but most people agree between the range of values from 0.85 to 0.99 although the exact value used by google is unknown.

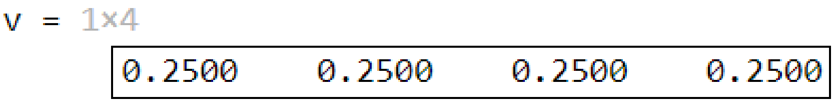
With the above things in mind, we can formulate the general form of a final matrix that most Page Ranking algorithms work on

Final Matrix:

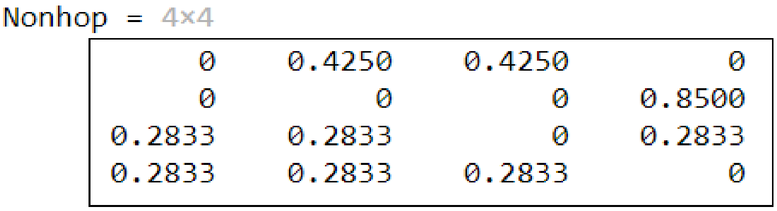
G = a*S + (1-a) *İ*v



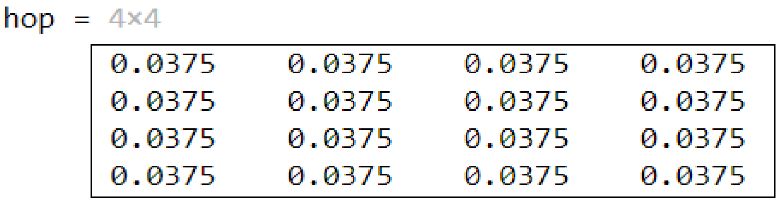
```
I1 = 4×1
      1
      1
      1
      1
```



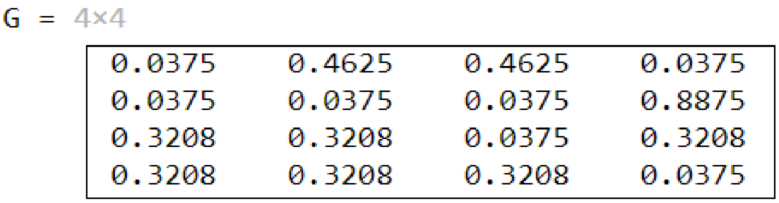
```
v = 1×4
      0.2500    0.2500    0.2500    0.2500
```



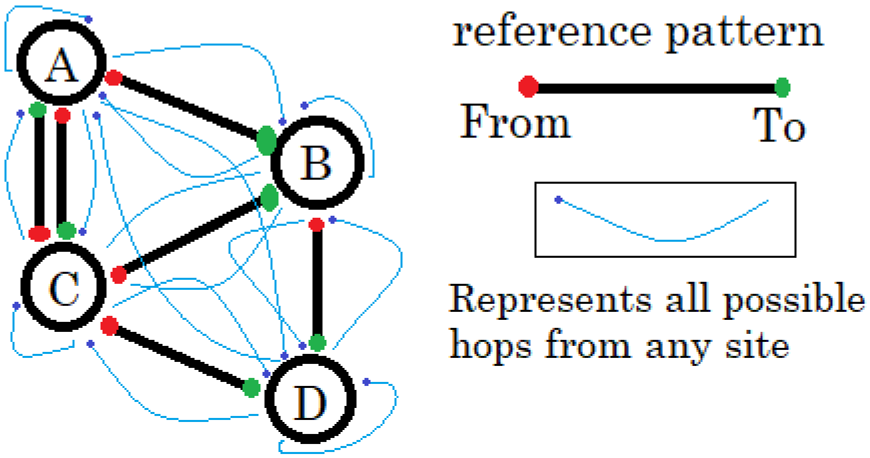
```
Nonhop = 4×4
             0    0.4250    0.4250         0
             0         0         0    0.8500
        0.2833    0.2833         0    0.2833
        0.2833    0.2833    0.2833         0
```



```
hop = 4×4
        0.0375    0.0375    0.0375    0.0375
        0.0375    0.0375    0.0375    0.0375
        0.0375    0.0375    0.0375    0.0375
        0.0375    0.0375    0.0375    0.0375
```



```
G = 4×4
        0.0375    0.4625    0.4625    0.0375
        0.0375    0.0375    0.0375    0.8875
        0.3208    0.3208    0.0375    0.3208
        0.3208    0.3208    0.3208    0.0375
```



*in the above matrix G is correct theoretically but we see that matrix diagonal terms are not zero – this represents the probability of entering the same link. (Hopping to the same site)

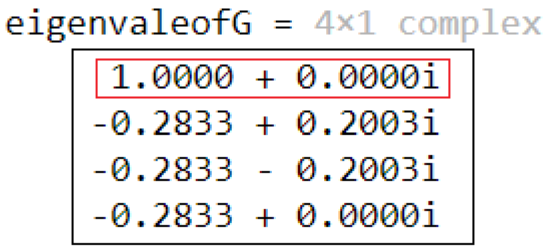
İ is a column vector used to identify all rows of the matrix which can be used to add the personalized hop from any website in combination with the probability to do so.

Further we refer to G as Google matrix for simplicity.

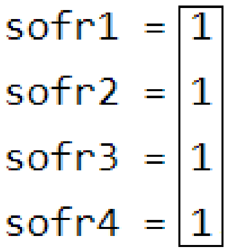
Spam protection:

If we closely look to inspect the final matrix, we see that on using v = w we make the matrix exposed to link spamming. Link spamming is the practice by some search engine optimization experts of adding more links to their clients' webpages for the sole purpose of increasing the PageRank score of those webpages. This attempt to manipulate PageRank scores is one reason Google does not reveal the current damping factor or personalization vector for the Google matrix (we can be pretty sure that v is not equal to w). In 2004, however, Gy¨ongyi, Garcia-Molina, and Pederson developed the TrustRank algorithm to create a personalization vector that decreases the harmful effect of link spamming, and Google registered the trademark for TrustRank on March 16, 2005.

Interesting observations of G:



```
eigenvaleofG = 4×1 complex
   1.0000 + 0.0000i
  -0.2833 + 0.2003i
  -0.2833 - 0.2003i
  -0.2833 + 0.0000i
```



```
sofr1 = 1
sofr2 = 1
sofr3 = 1
sofr4 = 1
```

- All elements of G lie between 0 and 1 and the sum of each element in each row of G is 1 by definition this makes it a stochastic matrix.
- Though not required, the personalization vector, v, and dangling node vector, w, often are defined to have all positive entries that sum to 1 instead of all nonnegative entries that sum to 1. Defined this way, the PageRank vector also has all positive entries that sum to 1.
- $\lambda = 1$ is not a repeated eigenvalue of G and is greater in magnitude than any other eigenvalue of G. Hence, the eigensystem, $\pi G = \pi$, has a unique solution, where π is a row probability distribution vector.
- For smaller a, v plays more role in G and vice-versa. Complimentary to the relation between a and S.

We say that $\lambda = 1$ is the dominant eigenvalue of G, and π is the corresponding dominant left eigenvector of G. The i th entry of π is the PageRank score for webpage i, and π is called the PageRank vector.

PageRank Scores:

This is the final score based on which the order of webpages for any query is determined but when I comes to searching the entire web, we start to run into some large-scale calculations as I contains 6 billion indexed web pages as of 2020. We need to use some approximation methods which allow us to control the error limit to a great extent that enable us to minimalize calculations. The oldest and easiest technique for approximating a dominant eigenvector of a matrix is the power method.

The power method converges when the dominant eigenvalue is not a repeated eigenvalue for most starting vectors. Since $\lambda = 1$ is the dominant eigenvalue of G and π is the dominant left eigenvector, the power method applied to G converges to the PageRank vector. This method was the original choice for computing the PageRank vector.

Given a starting vector π (0), e.g., π (0) = v, the power method calculates successive iterates

$\pi (k) = \pi (k-1) G$, where k = 1, 2, ...,

until some convergence criterion is satisfied. Notice that $\pi (k) = \pi (k-1) G$ can also be stated $\pi (k) = \pi (0) Gk$. As the number of nonzero elements of the personalization vector increases, the number of nonzero elements of G increases. Thus, the multiplication of π (k−1) with G is expensive; however, since $S = H + dw$ and $G = \alpha S + (1 - \alpha)11v$, we can express the multiplication as follows:

$\pi (k) = \pi (k-1) G$

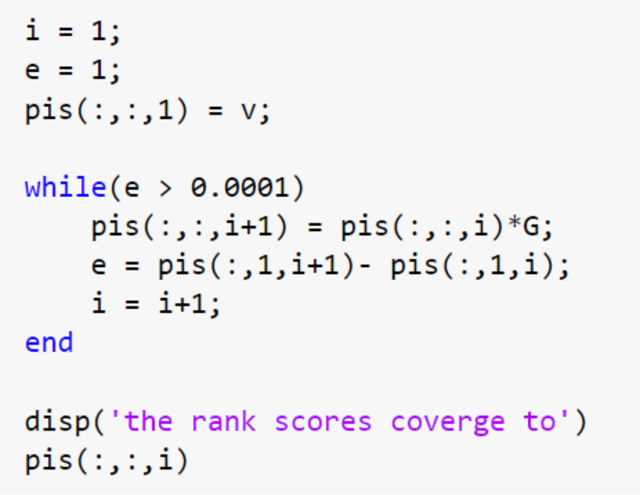
$= \pi\ (k-1)\ [\alpha\ (H + dw) + (1 - \alpha)\ \dot{I}\ v]$

$= \alpha\pi(k-1)\ H + \alpha\ (\pi\ (k-1)\ d)\ w + (1 - \alpha)\ (\pi\ (k-1)\ \dot{I})\ v$

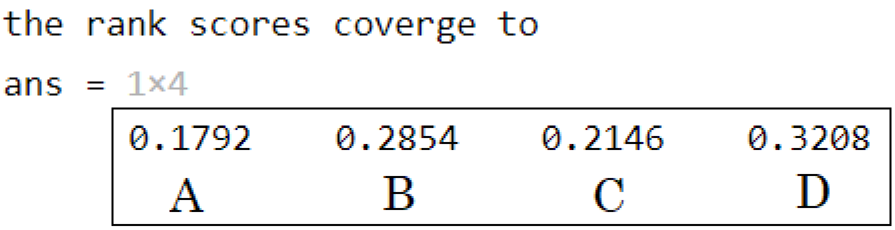
$= \alpha\pi(k-1)\ H + \alpha\ (\pi\ (k-1)\ d)\ w + (1 - \alpha)\ v$, since $\pi\ (k-1)\ \dot{I} = 1$.

The criterion used in the following code is e which stands for error (it is a term that replaces (k−1) in the above formula with error-controlled accuracy term e)

Matlab code:



```
i = 1;
e = 1;
pis(:,:,1) = v;

while(e > 0.0001)
    pis(:,:,i+1) = pis(:,:,i)*G;
    e = pis(:,1,i+1)- pis(:,1,i);
    i = i+1;
end

disp('the rank scores coverge to')
pis(:,:,i)
```



```
the rank scores coverge to
ans = 1×4
    0.1792    0.2854    0.2146    0.3208
      A         B         C         D
```

This shows the priority order as D>B>C>A

On observing the used DWG, we can clearly see the pattern.

We have used Matlab to demonstrate the inner workings of PageRank. But some additional features like priority ranking(sorting) etc., can be more easily implemented using more user compatible programing languages

This is a sum of three vectors: a multiple of $\pi\ (k-1)\ H$, a multiple of w, and a multiple of v. (Notice that $\pi\ (k-1)\ d$ is a scalar.) The only matrix-vector multiplication require is with the hyperlink matrix H. A 2004 investigation of Web documents estimates that the average number of out links for a webpage is 52. This means that for a typical row of the hyperlink matrix only 52 of the 25 billion elements are nonzero, so most elements in H are 0 (H is very sparse).

Since all computations involve the sparse matrix H and vectors w and v, an iteration of the power method is cheap (the operation count is proportional to the matrix dimension n).

The ratio of the two eigenvalues largest in magnitude for a given matrix determines how quickly the power method converges. Haveliwala and Kamvar were the first to prove that the second largest eigenvalue in magnitude of G is less than or equal to the damping factor α. This means that the ratio is less than or equal to α for the Google matrix. Thus, the power method converges quickly when α is less than 1. This might explain why Brin and Page originally used $\alpha = 0.85$. No more than 29 iterations are required for the maximal element of the difference in successive iterates, $\pi (k+1) - \pi (k)$, to be less than $10-2$ for $\alpha = 0.85$. The number of iterations increases to 44 for $\alpha = 0.90$.

An Alternative Way to Compute:

PageRank Although Brin and Page originally defined PageRank as a solution to the eigensystem $\pi G = \pi$, the problem can be restated as a linear system. Recall, $G = \alpha S + (1 - \alpha)\ 11v$. Transforming $\pi G = \pi$ to $0 = \pi - \pi G$ gives:

$0 = \pi - \pi G$

$= \pi I - \pi\ (\alpha S + (1 - \alpha)\ 11v)$

$= \pi\ (I - \alpha S) - (1 - \alpha)\ (\pi 11)\ v$

$= \pi\ (I - \alpha S) - (1 - \alpha)\ v$

The last equality follows from the fact that π is a probability distribution vector, so the elements of π are nonnegative and sum to 1. In other words, $\pi 11 = 1$. Thus,

$\pi\ (I - \alpha S) = (1 - \alpha)\ v$,

which means π solves a linear system with coefficient matrix $I - \alpha S$ and right hand 10 side $(1 - \alpha)\ v$. Since the matrix, $I - \alpha S$, is nonsingular, the linear system has a unique solution.

Other factors to deliver results:

We haven't looked at some factors which play a big role in helping to decide the PageRank scores because they are either additions to the G matrix (like the terms we used to account for hopping and 0-vectors) or some ratio value manipulation.

- Keywords used in the query – any portion of the text query if surrounded by double quotes “example query” is taken as a search query's keyword if not specified, it has a linguistic saturation algorithm to select keywords in a sentence and then bases its SERP on it.
- The location of query – the search results vary based on the location of the request these are often managed by localized control servers and each of them have specific tweaks based off of their location. For example – on an Indian server the search for “Holi colors”

lists places to buy colors but the same search on an American server might give results on description of the festival.

- The time relevance of query –the freshness of the webpages also affects the SERP. For example – latest news about a topic is more likely to appear before older ones.

Salient features of Google’s search engine:

A part of googles fame is attributed to its own web browser and

1. It is user friendly and user appealing – Its simple design invites new users to fully experience it and mistakes are less frustrating as its algorithm copes for user mistakes like spelling. It also provides search suggestions and can give the right results if only a description about the item is given (true clarity of object is not known).
2. Framework of compatible apps – It also has multiple external apps like Google Maps, drive, YouTube and these are also very good apps that are complimentary to google providing everything a user might need enforcing the use of google.
3. The program is highly tested and regularly updated, making its algorithms faster and more efficient. This comes in handy with user satisfaction.
4. Despite achieving the top ranks in its field, they strive to delivery further by focusing their research on new technologies.
5. Adds – we can pay google to rank our webpage higher than others to serve as advertisements (based on specific search results) this shows how much control they have over their engine.

Search engine optimization and result review:

To make sure new changes are making an effect googles result for the better they have their own method of making set of professionals review the new SERPs side by side with their guidelines and they take into consideration the user inputs from revies time to time.

From all that we have learnt we can conclusively say that in order to optimize any website there are no shortcuts but to be committed towards delivering quality content. These set of rules can serve as a checklist to have the best chance.

1. Getting started.
2. Help Google find your content.
3. Tell Google which pages shouldn't be crawled.
4. Help Google (and users) understand your content.
5. Manage your appearance in Google Search results.
6. Organize your site hierarchy.
7. Optimize your content.
8. Optimize your images.

References:

1. Google's PageRank: The Math Behind the Search Engine by Rebecca S. Wills
2. Using Linear Algebra for Intelligent Information Retrieval by M.W Berry, S.T.Dumais & G.W.O'Brien