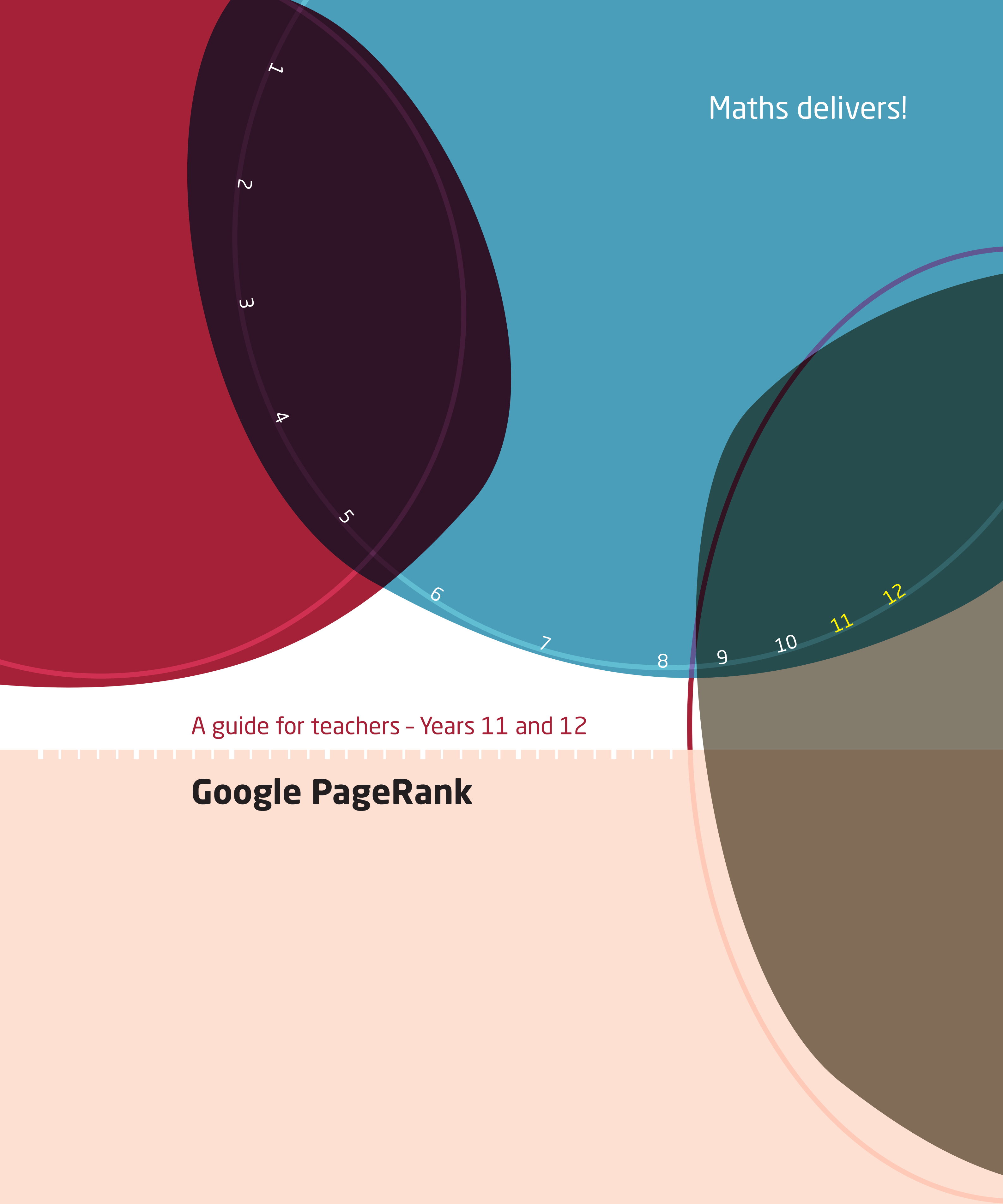
Maths delivers!



A guide for teachers – Years 11 and 12

Google PageRank

AMSI
AUSTRALIAN MATHEMATICAL SCIENCES INSTITUTE

Maths delivers!
Google PageRank

Professor Brian A. Davey, La Trobe University

Editor: Dr Jane Pitkethly, La Trobe University

Illustrations and web design: Catherine Tan, Michael Shaw

For teachers of Secondary Mathematics

Australian Mathematical Sciences Institute
Building 161
The University of Melbourne
VIC 3010
Email: enquiries@amsi.org.au
Website: www.amsi.org.au

maths
delivers!

Google PageRank

Introduction

What happens when you submit a search query to Google?

1. A list of relevant web pages is produced, based on the words in your query.
2. This list of web pages is ordered, so that (hopefully) the pages most useful to you will be at the top of the list.

Google's success at performing this second step has played a major role in making it the world's favourite search engine.

The program that produces the list of relevant web pages, based on your query, is called the **query module**. The list is passed to the **ranking module**. This program gives each web page in the list an **overall score**, which is obtained by combining together:

- a **content score**, which is a measure of how relevant the page is to your query
- a **popularity score**, which is completely independent of your query.

The list of web pages is displayed in the order determined by their overall scores.

A web page's popularity score is also called its **Google PageRank**. In these notes, which accompany the *maths delivers!* video of the same name, we explain how the PageRank of a web page is calculated, and we discuss some of the mathematics which guarantees that the calculation actually works.

The PageRank formula was presented to the world in Brisbane at the Seventh World Wide Web Conference (WWW98) by Sergey Brin and Larry Page, the founders of Google, in 1998. That year, Larry Page filed a patent for their process to calculate the PageRank of web pages, and the patent was granted in 2001. The rest is history!

Figure 1: Larry Page and Sergey Brin.

The basic idea

We would like to attach a number to each web page that represents its *importance*.

Google's founders Brin and Page suggested the idea of an imaginary web surfer, whom we shall call **Webster**, who surfs the web randomly. Webster starts at a random web page. Whenever he visits a web page, he randomly chooses a hyperlink on that page and follows it. Webster continues this indefinitely. In the limit, the proportion of the time, $r(P_i)$, that Webster spends at a particular page P_i is a measure of the importance of the page P_i.

Converting this idea into a formula that can be calculated for each of the more than 14 billion web pages is the achievement that led Google to become the top web search engine in existence.

The hyperlink graph of a set of web pages

A system of web pages with hyperlinks between them is viewed as a directed graph W, called the **hyperlink graph** of the system. The nodes (or vertices) of the graph W are the pages, and there is an edge from page P_i to page P_j if there is a hyperlink that points from page P_i to page P_j; this is an **outlink** from page P_i and an **inlink** to page P_j. (Multiple links from P_i to P_j are treated as a single edge of the graph, and self links from a page to itself are ignored. Thus W is a simple, loopless directed graph.)

The graph W in figure 2 is the hyperlink graph of a system of six web pages. We can see, for example, that there are outlinks from page P_3 to pages P_1, P_2 and P_4, and that P_2 has inlinks from pages P_1 and P_3 but has no outlinks at all.

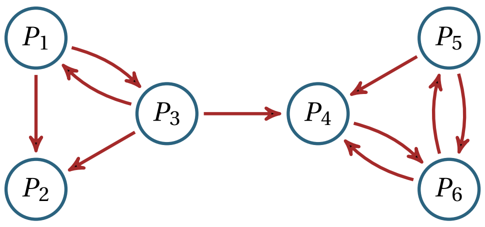


Figure 2: The hyperlink graph W.

We will pretend that this graph W represents a miniature 'World Wide Web', and see how to calculate the PageRank for each of the six web pages.

The hyperlink matrix

The graph W shown in figure 2 is encoded in the 6×6 table on the left in figure 3: the cell in row P_i, column P_j contains a 1 if there is a link from page P_i to page P_j, and contains a 0 otherwise.

	P_1	P_2	P_3	P_4	P_5	P_6
P_1	0	1	1	0	0	0
P_2	0	0	0	0	0	0
P_3	1	1	0	1	0	0
P_4	0	0	0	0	0	1
P_5	0	0	0	1	0	1
P_6	0	0	0	1	1	0

$$\rightsquigarrow \qquad A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Figure 3: The hyperlink matrix A of W.

Removing the row and column labels of the table leaves us with the **hyperlink matrix** A (also known as the **adjacency matrix**) of the graph W. The matrix A is shown on the right in figure 3.

The PageRank equations

How do we measure the importance of a page? A link from your page to my page is an endorsement of my page. The more inlinks my page has, the more important it is. However, some links are more valuable than others.

- A link to my page from a very important page is more valuable. The more important a page is, the more it should contribute to the importance of the pages to which it links.
- A link to my page from a page with a large number of outlinks is not so valuable. The more outlinks a page has, the less valuable is the recommendation provided by each of its individual outlinks.

In order to formalise these two ideas, we first introduce some notation:

- Let the PageRank of page P_i be denoted by $r(P_i)$. This will be a number that measures the importance of page P_i.
- Let the number of outlinks from page P_i be denoted by $|P_i|$. Note that $|P_i|$ is equal to the sum of the entries in the ith row of the hyperlink matrix A.

Each page P_j that links to page P_i contributes some of its importance $r(P_j)$ to the importance of page P_i. The more outlinks such a page has, the less it should contribute to the importance of page P_i. Consequently, if page P_i has an inlink from page P_j, we will say that P_j contributes

$$\frac{r(P_j)}{|P_j|}$$

to the PageRank $r(P_i)$ of page P_i. Adding up all of these contributions from pages with a link to P_i yields the PageRank of P_i.

For our six-page 'mini web', this leads to the following **PageRank equations**:

$$r(P_1) = \frac{r(P_3)}{3},$$

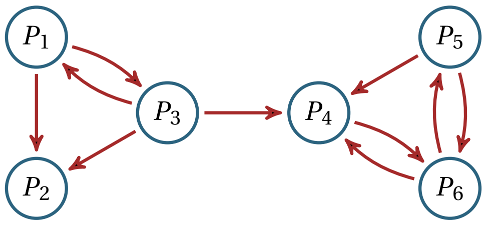
$$r(P_2) = \frac{r(P_1)}{2} + \frac{r(P_3)}{3},$$

$$r(P_3) = \frac{r(P_1)}{2},$$

$$r(P_4) = \frac{r(P_3)}{3} + \frac{r(P_5)}{2} + \frac{r(P_6)}{2},$$

$$r(P_5) = \frac{r(P_6)}{2},$$

$$r(P_6) = \frac{r(P_4)}{1} + \frac{r(P_5)}{2}.$$



We are searching for a **PageRank row vector**

$$\boldsymbol{v} = \big(r(P_1),\ r(P_2),\ r(P_3),\ r(P_4),\ r(P_5),\ r(P_6)\big)$$

that satisfies these six PageRank equations. Moreover, since $r(P_i)$ should give the proportion of time that our random web surfer Webster spends at page P_i, the vector $\boldsymbol{v}$ should be a **probability vector**. That is, we should have $r(P_i) \geq 0$, for each page P_i, and

$$r(P_1) + r(P_2) + r(P_3) + r(P_4) + r(P_5) + r(P_6) = 1.$$

If we denote the set of pages with a hyperlink to P_i by L_{P_i}, then each PageRank equation can be written in summation notation as

$$r(P_i) = \sum_{P_j \in L_{P_i}} \frac{r(P_j)}{|P_j|}.$$

The PageRank equations via matrices

The six PageRank equations given in the previous section can be conveniently summarised as a single matrix equation.

To do this, we first normalise each row of the hyperlink matrix A, that is, we divide each entry in the matrix A by the sum of its row. Now each non-zero row will sum to 1. The resulting matrix H is called the **row-normalised hyperlink matrix** of the graph W, and is shown on the right in figure 4.

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \quad \rightsquigarrow \quad H = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

Figure 4: The row-normalised hyperlink matrix H of W.

Our six PageRank equations are now equivalent to the following matrix equation:

$$\begin{aligned} &\begin{pmatrix} r(P_1) & r(P_2) & r(P_3) & r(P_4) & r(P_5) & r(P_6) \end{pmatrix} \\ &= \begin{pmatrix} r(P_1) & r(P_2) & r(P_3) & r(P_4) & r(P_5) & r(P_6) \end{pmatrix} \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}. \end{aligned}$$

This equation can be written somewhat more briefly as

$$\boldsymbol{v} = \boldsymbol{v} H.$$

It would be easy to solve our six PageRank equations by hand. There is also a systematic method for solving such a system of simultaneous equations (the **Gaussian algorithm**) which can be carried out by a computer.

However, finding PageRanks for the entire World Wide Web involves a system of more than 14 billion PageRank equations, and it is not feasible to try to solve these PageRank equations directly, even by computer.

Solving the PageRank equations iteratively

Brin and Page suggested that the matrix equation $\boldsymbol{v} = \boldsymbol{v}\,H$ given in the previous section should be solved iteratively, that is, by calculating a sequence of approximations to the solution vector $\boldsymbol{v}$.

For the initial approximation $\boldsymbol{v}^{(0)}$, they take all pages to have the same PageRank. For our 'mini web' example, we have

$$\boldsymbol{v}^{(0)} = \left(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right).$$

This is the 0th approximation to the PageRank row vector $\boldsymbol{v}$. Successive approximations are obtained by multiplying on the right by the matrix H:

$$\begin{aligned}\boldsymbol{v}^{(1)} &= \boldsymbol{v}^{(0)}\,H\\ \boldsymbol{v}^{(2)} &= \boldsymbol{v}^{(1)}\,H\\ \boldsymbol{v}^{(3)} &= \boldsymbol{v}^{(2)}\,H\\ &\ \ \vdots\end{aligned}$$

In general, we have $\boldsymbol{v}^{(k+1)} = \boldsymbol{v}^{(k)}H$. We hope that the sequence $\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \boldsymbol{v}^{(3)}, \dots$ will converge to a probability vector $\boldsymbol{v}$ with $\boldsymbol{v} = \boldsymbol{v}\,H$.

There are two important questions:

A Is there a theorem that guarantees the sequence $\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \boldsymbol{v}^{(3)}, \dots$ will converge to a vector $\boldsymbol{v}$?

B If the sequence $\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \boldsymbol{v}^{(3)}, \dots$ converges to a vector $\boldsymbol{v}$, will $\boldsymbol{v}$ necessarily be a probability vector?

Solving the dangling-node problem

We shall address question B first. In the case of our six-page 'mini web' W, we have

$$\boldsymbol{v}^{(0)} = \left(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right).$$

Using an Excel spreadsheet, we can easily apply the iterative formula $\boldsymbol{v}^{(k+1)} = \boldsymbol{v}^{(k)}H$ to calculate $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \boldsymbol{v}^{(3)}, \dots$. The results are given in figure 5. See the *Appendix* section of these notes for instructions on how to set up such a spreadsheet.

	P_1	P_2	P_3	P_4	P_5	P_6
v^(0)	0.16666667	0.16666667	0.16666667	0.16666667	0.16666667	0.16666667
v^(1)	0.05555556	0.13888889	0.08333333	0.22222222	0.08333333	0.25000000
v^(2)	0.02777778	0.05555556	0.02777778	0.19444444	0.12500000	0.26388889
v^(3)	0.00925926	0.02314815	0.01388889	0.20370370	0.13194444	0.25694444
v^(4)	0.00462963	0.00925926	0.00462963	0.19907407	0.12847222	0.26967593
v^(5)	0.00154321	0.00385802	0.00231481	0.20061728	0.13483796	0.26331019
v^(6)	0.00077160	0.00154321	0.00077160	0.19984568	0.13165509	0.26803627
v^(7)	0.00025720	0.00064300	0.00038580	0.20010288	0.13401813	0.26567323
v^(8)	0.00012860	0.00025720	0.00012860	0.19997428	0.13283661	0.26711195
v^(9)	0.00004287	0.00010717	0.00006430	0.20001715	0.13355597	0.26639259
v^(10)	0.00002143	0.00004287	0.00002143	0.19999571	0.13319629	0.26679513
v^(11)	0.00000714	0.00001786	0.00001072	0.20000286	0.13339757	0.26659386
v^(12)	0.00000357	0.00000714	0.00000357	0.19999929	0.13329693	0.26670164
v^(13)	0.00000119	0.00000298	0.00000179	0.20000048	0.13335082	0.26664775
v^(14)	0.00000060	0.00000119	0.00000060	0.19999988	0.13332388	0.26667589
v^(15)	0.00000020	0.00000050	0.00000030	0.20000008	0.13333794	0.26666182
v^(16)	0.00000010	0.00000020	0.00000010	0.19999998	0.13333091	0.26666905
v^(17)	0.00000003	0.00000008	0.00000005	0.20000001	0.13333453	0.26666543
v^(18)	0.00000002	0.00000003	0.00000002	0.20000000	0.13333272	0.26666728
v^(19)	0.00000001	0.00000001	0.00000001	0.20000000	0.13333364	0.26666636
v^(20)	0.00000000	0.00000001	0.00000000	0.20000000	0.13333318	0.26666682
v^(21)	0.00000000	0.00000000	0.00000000	0.20000000	0.13333341	0.26666659
v^(22)	0.00000000	0.00000000	0.00000000	0.20000000	0.13333329	0.26666671
v^(23)	0.00000000	0.00000000	0.00000000	0.20000000	0.13333335	0.26666665
v^(24)	0.00000000	0.00000000	0.00000000	0.20000000	0.13333332	0.26666668
v^(25)	0.00000000	0.00000000	0.00000000	0.20000000	0.13333334	0.26666666
v	0	0	0	1/5	2/15	4/15

Figure 5: The Excel spreadsheet using H.

While this is not a proof, we see that the limit vector $\boldsymbol{v}$ seems to be

$$\boldsymbol{v} = \left(0,\ 0,\ 0,\ \frac{1}{5},\ \frac{2}{15},\ \frac{4}{15}\right).$$

Indeed, it is easy to check that this choice of $\boldsymbol{v}$ satisfies the matrix equation $\boldsymbol{v} = \boldsymbol{v}H$. Unfortunately, since the sum of the coordinates is $\frac{3}{5}$ rather than 1, this is not a probability vector.

The problem is caused by the row of zeros in the matrix H. This row of zeros corresponds to the fact that P_2 is a **dangling node**, that is, it has no outlinks. Dangling nodes are very common in the World Wide Web (for example: image files, PDF documents, etc.), and they cause a problem for our random web surfer. When Webster enters a dangling node, he has nowhere to go and is stuck.

To overcome this problem, Brin and Page declare that, when Webster enters a dangling page, he may then jump to any page at random. This corresponds to replacing each row of 0's in the matrix H by a row of $\frac{1}{n}$'s, where n is the total number of nodes in our graph. This new matrix S is called the **stochastic matrix** of the graph W, as each row sums to 1.

For our ‘mini web’ example, this transformation is shown in figure 6.

$$H = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix} \quad \rightsquigarrow \quad S = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

Figure 6: The stochastic matrix S of W.

We can now repeat the iteration using the stochastic matrix S instead of H. That is, we now use the formula

$$\boldsymbol{v}^{(k+1)} = \boldsymbol{v}^{(k)} S$$

to calculate $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \boldsymbol{v}^{(3)}, \ldots$. Using Excel, we obtain the results shown in figure 7.

	P_1	**P_2**	**P_3**	**P_4**	**P_5**	**P_6**
v^(0)	0.16666667	0.16666667	0.16666667	0.16666667	0.16666667	0.16666667
v^(1)	0.08333333	0.16666667	0.11111111	0.25000000	0.11111111	0.27777778
v^(2)	0.06481481	0.10648148	0.06944444	0.25925926	0.16666667	0.33333333
v^(3)	0.04089506	0.07330247	0.05015432	0.29089506	0.18441358	0.36033951
v^(4)	0.02893519	0.04938272	0.03266461	0.30131173	0.19238683	0.39531893
v^(5)	0.01911866	0.03358625	0.02269805	0.31297154	0.20588992	0.40573560
v^(6)	0.01316372	0.02272305	0.01515704	0.31897648	0.20846551	0.42151420
v^(7)	0.00883952	0.01542138	0.01036904	0.32382938	0.21454428	0.42699641
v^(8)	0.00602658	0.01044634	0.00698999	0.32679692	0.21606843	0.43367174
v^(9)	0.00407105	0.00708434	0.00475434	0.32894114	0.21857693	0.43657219
v^(10)	0.00276550	0.00480103	0.00321625	0.33034006	0.21946682	0.43941033
v^(11)	0.00187226	0.00325501	0.00218292	0.33131083	0.22050534	0.44087365
v^(12)	0.00127014	0.00220627	0.00147863	0.33195963	0.22097932	0.44210600
v^(13)	0.00086059	0.00149566	0.00100278	0.33240325	0.22142071	0.44281701
v^(14)	0.00058354	0.00101383	0.00067957	0.33270240	0.22165778	0.44336288
v^(15)	0.00039550	0.00068726	0.00046074	0.33290583	0.22185041	0.44370026
v^(16)	0.00026812	0.00046587	0.00031229	0.33304346	0.22196467	0.44394558
v^(17)	0.00018174	0.00031580	0.00021171	0.33313687	0.22205043	0.44410344
v^(18)	0.00012320	0.00021407	0.00014351	0.33320014	0.22210436	0.44421472
v^(19)	0.00008351	0.00014512	0.00009728	0.33324305	0.22214304	0.44428800
v^(20)	0.00005661	0.00009837	0.00006594	0.33327213	0.22216819	0.44433876
v^(21)	0.00003838	0.00006668	0.00004470	0.33329185	0.22218577	0.44437262
v^(22)	0.00002601	0.00004520	0.00003030	0.33330521	0.22219742	0.44439585
v^(23)	0.00001763	0.00003064	0.00002054	0.33331427	0.22220546	0.44441146
v^(24)	0.00001195	0.00002077	0.00001392	0.33332041	0.22221083	0.44442211
v^(25)	0.00000810	0.00001408	0.00000944	0.33332457	0.22221451	0.44442929
v	0	0	0	1/3	2/9	4/9

Figure 7: The Excel spreadsheet using S.

The limit of the sequence $\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \boldsymbol{v}^{(3)}, \ldots$ of row vectors is now

$$\boldsymbol{v} = \left(0,\ 0,\ 0,\ \frac{1}{3},\ \frac{2}{9},\ \frac{4}{9}\right),$$

which is a probability vector since

$$\frac{1}{3} + \frac{2}{9} + \frac{4}{9} = 1.$$

So replacing the rows of zeros has solved the problem caused by dangling nodes: If we use the stochastic matrix S instead of the row-normalised hyperlink matrix H, then the limiting vector $\boldsymbol{v}$, when it exists, will be a probability vector.

Even when the limiting vector $\boldsymbol{v}$ exists, however, our example illustrates another problem. We wish to use the PageRanks of our six web pages to produce an ordering of the pages. Unfortunately, the PageRanks calculated above do not allow us to distinguish between the pages P_1, P_2 and P_3, as they all have PageRank equal to 0. The best we could do is to rank the pages $(4,4,4,2,3,1)$.

In order to eliminate the problem of multiple pages with a PageRank of 0, we add a third question to our list:

C If the sequence $\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \boldsymbol{v}^{(3)}, \ldots$ converges to a vector $\boldsymbol{v}$, how can we guarantee that $\boldsymbol{v}$ is a **positive vector**, that is, each entry in $\boldsymbol{v}$ is positive?

Guaranteeing that the limiting vector exists

There is a simple modification to the matrix S that will simultaneously answer questions A and C: We will be guaranteed that the sequence of vectors $\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \boldsymbol{v}^{(3)}, \ldots$ converges to a positive vector $\boldsymbol{v}$.

This modification can be justified in terms of the behaviour of our random web surfer. Brin and Page suggest that, from time to time, the web surfer Webster will become bored with following hyperlinks, and he will request a completely random web page. Once there, he will continue following hyperlinks until he becomes bored again.

If the proportion of the time that Webster continues to follow hyperlinks is d, then the proportion of the time that he changes to another randomly chosen page is $1 - d$. The number d, called the **damping factor**, is chosen so that $0 < d < 1$.

To represent this new surfing behaviour for our six-page example W, we must replace each row $\boldsymbol{r}$ of the matrix S by

$$d\boldsymbol{r} + (1-d)\Big(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\Big).$$

This yields a new matrix G, called the **Google matrix** of the graph W:

$$G = d\begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix} + (1-d)\begin{pmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{pmatrix}.$$

In their original 1998 paper, Brin and Page state that they usually set $d = 0.85$. With this choice, the Google matrix of our graph W is

$$G = \begin{pmatrix} \frac{1}{40} & \frac{9}{20} & \frac{9}{20} & \frac{1}{40} & \frac{1}{40} & \frac{1}{40} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{37}{120} & \frac{37}{120} & \frac{1}{40} & \frac{37}{120} & \frac{1}{40} & \frac{1}{40} \\ \frac{1}{40} & \frac{1}{40} & \frac{1}{40} & \frac{1}{40} & \frac{1}{40} & \frac{7}{8} \\ \frac{1}{40} & \frac{1}{40} & \frac{1}{40} & \frac{9}{20} & \frac{1}{40} & \frac{9}{20} \\ \frac{1}{40} & \frac{1}{40} & \frac{1}{40} & \frac{9}{20} & \frac{9}{20} & \frac{1}{40} \end{pmatrix}.$$

Once again we can repeat the iteration, this time using the Google matrix G (with damping factor $d = 0.85$). That is, we now use the formula

$$\boldsymbol{v}^{(k+1)} = \boldsymbol{v}^{(k)}G$$

to calculate $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \boldsymbol{v}^{(3)}, \ldots$. Using Excel, we obtain the results shown in figure 8.

	P_1	P_2	P_3	P_4	P_5	P_6
v^(0)	0.16666667	0.16666667	0.16666667	0.16666667	0.16666667	0.16666667
v^(1)	0.09583333	0.16666667	0.11944444	0.23750000	0.11944444	0.26111111
v^(2)	0.08245370	0.12318287	0.08934028	0.24418981	0.15958333	0.30125000
v^(3)	0.06776399	0.10280681	0.07749373	0.26361815	0.17048216	0.31783517
v^(4)	0.06152086	0.09032055	0.06836399	0.26905572	0.17464424	0.33609464
v^(5)	0.05716521	0.08331157	0.06394177	0.27422924	0.18063563	0.34071657
v^(6)	0.05491931	0.07921452	0.06109769	0.27649400	0.18160702	0.34666747
v^(7)	0.05353307	0.07687377	0.05956276	0.27804972	0.18355573	0.34842494
v^(8)	0.05276657	0.07551812	0.05864201	0.27885835	0.18397105	0.35024390
v^(9)	0.05231364	0.07473943	0.05812419	0.27935499	0.18455206	0.35091570
v^(10)	0.05205661	0.07428990	0.05782138	0.27963040	0.18472726	0.35147445
v^(11)	0.05190713	0.07403118	0.05764846	0.27979285	0.18490105	0.35171933
v^(12)	0.05182148	0.07388201	0.05754828	0.27988514	0.18496847	0.35189462
v^(13)	0.05177196	0.07379609	0.05749075	0.27993878	0.18502183	0.35198059
v^(14)	0.05174349	0.07374658	0.05745753	0.27996952	0.18504620	0.35203668
v^(15)	0.05172707	0.07371805	0.05743842	0.27998729	0.18506302	0.35206616
v^(16)	0.05171761	0.07370161	0.05742739	0.27999751	0.18507151	0.35208437
v^(17)	0.05171216	0.07369214	0.05742105	0.28000340	0.18507692	0.35209433
v^(18)	0.05170902	0.07368668	0.05741739	0.28000680	0.18507981	0.35210030
v^(19)	0.05170721	0.07368354	0.05741528	0.28000876	0.18508158	0.35210365
v^(20)	0.05170616	0.07368173	0.05741406	0.28000988	0.18508255	0.35210561
v^(21)	0.05170556	0.07368068	0.05741336	0.28001053	0.18508313	0.35210673
v^(22)	0.05170522	0.07368008	0.05741296	0.28001091	0.18508346	0.35210738
v^(23)	0.05170502	0.07367973	0.05741273	0.28001112	0.18508365	0.35210775
v^(24)	0.05170490	0.07367953	0.05741259	0.28001125	0.18508376	0.35210797
v^(25)	0.05170484	0.07367942	0.05741252	0.28001132	0.18508382	0.35210809
v	0.0517	0.0737	0.0574	0.2800	0.1851	0.3521

Figure 8: The Excel spreadsheet using G.

To four-decimal places, the PageRank row vector for our example is

$$\boldsymbol{v} = (0.0517,\ 0.0737,\ 0.0574,\ 0.2800,\ 0.1851,\ 0.3521).$$

This allows us to rank our six web pages in order from most important to least important as P_6, P_4, P_5, P_2, P_3, P_1.

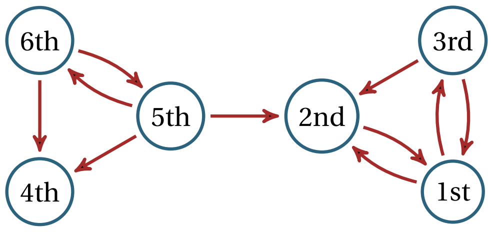


Figure 9: Ranking of web pages for our example.

The calculation of PageRanks for our example involves a 6×6 matrix. However, the calculation by Google of PageRanks for the entire World Wide Web involves a matrix that is more than 14 billion by 14 billion! The reason that this calculation is even possible is that the Google matrix G is based on the original hyperlink matrix A, which is **sparse** (meaning that most of its entries are zeros). If there are n web pages, then the matrix A has n^2 entries. On average, a web page has only 10 outlinks, and so there are approximately $10n$ non-zero entries in the matrix A. As the number of web pages n is extremely large, the number $10n$ is very much smaller than n^2.

The calculation of PageRanks from the Google matrix relies on several deep theorems of mathematics:

1 Since G is a positive stochastic matrix (that is, every entry is positive and each row sums to 1), an important theorem of *linear algebra*, proved by Oskar Perron in 1907, guarantees that there is a unique positive probability vector $\boldsymbol{v}$ that satisfies $\boldsymbol{v} = \boldsymbol{v}G$.

2 The theory of *Markov chains* guarantees:

   a the sequence of iterates $\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \boldsymbol{v}^{(3)}, \dots$ converges to $\boldsymbol{v}$, regardless of the choice for the starting vector $\boldsymbol{v}^{(0)}$

   b to produce PageRank scores with approximately m digits of accuracy, the number of iterations should be around $\dfrac{m}{\log_{10}(\frac{1}{d})}$.

Using the formula in 2b above, we obtain the following table.

Calculation of PageRank scores

Digits of accuracy	Iterations required
1	15
2	29
3	43
4	57
5	71
6	86
7	100

Apparently, Google uses somewhere between 50 and 100 iterations, meaning that they are guaranteed an accuracy of between 3 and 7 digits.

To illustrate the independence of the limiting vector $\boldsymbol{v}$ from the starting vector, we can simply change the value of $\boldsymbol{v}^{(0)}$ in our spreadsheet. If we choose

$$\boldsymbol{v}^{(0)} = \big(1,\ 0,\ 0,\ 0,\ 0,\ 0\big),$$

then the sequence of iterates is as given in figure 10. As the theory predicts, this produces the same final PageRank vector $\boldsymbol{v}$ as before.

	P_1	**P_2**	**P_3**	**P_4**	**P_5**	**P_6**
v^(0)	1.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
v^(1)	0.02500000	0.45000000	0.45000000	0.02500000	0.02500000	0.02500000
v^(2)	0.21625000	0.22687500	0.09937500	0.23750000	0.09937500	0.12062500
v^(3)	0.08529688	0.17720313	0.14904688	0.17879688	0.10840625	0.30125000
v^(4)	0.09233372	0.12858490	0.08635495	0.26643763	0.17813503	0.24815378
v^(5)	0.06768343	0.10692526	0.08245803	0.24885617	0.14868155	0.34539557
v^(6)	0.06351085	0.09227631	0.06891320	0.27349363	0.18694086	0.31486515
v^(7)	0.05759788	0.08459000	0.06506459	0.27086544	0.17189017	0.34999193
v^(8)	0.05541855	0.07989765	0.06146268	0.27721844	0.18573015	0.34027253
v^(9)	0.05373326	0.07728614	0.05987172	0.27728440	0.18093466	0.35088982
v^(10)	0.05291252	0.07574916	0.05878551	0.27893793	0.18507704	0.34853784
v^(11)	0.05238702	0.07487485	0.05821895	0.27917335	0.18385971	0.35148611
v^(12)	0.05210264	0.07436713	0.05787176	0.27962462	0.18498887	0.35104500
v^(13)	0.05193234	0.07407596	0.05767896	0.27974673	0.18472947	0.35183654
v^(14)	0.05183647	0.07390771	0.05756534	0.27987702	0.18502462	0.35178884
v^(15)	0.05178044	0.07381094	0.05750076	0.27992616	0.18498052	0.35200119
v^(16)	0.05174843	0.07375512	0.05746324	0.27996566	0.18505705	0.35201050
v^(17)	0.05172989	0.07372297	0.05744172	0.27998360	0.18505311	0.35206870
v^(18)	0.05171924	0.07370445	0.05742929	0.27999601	0.18507328	0.35207772
v^(19)	0.05171310	0.07369377	0.05742214	0.28000227	0.18507450	0.35209422
v^(20)	0.05170956	0.07368762	0.05741802	0.28000626	0.18507999	0.35209854
v^(21)	0.05170752	0.07368408	0.05741564	0.28000840	0.18508096	0.35210340
v^(22)	0.05170634	0.07368204	0.05741427	0.28000970	0.18508252	0.35210512
v^(23)	0.05170567	0.07368086	0.05741348	0.28001042	0.18508297	0.35210660
v^(24)	0.05170528	0.07368018	0.05741303	0.28001084	0.18508343	0.35210724
v^(25)	0.05170505	0.07367979	0.05741277	0.28001108	0.18508360	0.35210770
v	0.0517	0.0737	0.0574	0.2800	0.1851	0.3521

Figure 10: The Excel spreadsheet using G: new $\boldsymbol{v}^{(0)}$.

Postscript

Given the importance of mathematics to Google, it is not surprising that they actively seek out talented mathematicians to join their team. In 2004, Google tried an ingenious way to do this. They wrote

$$\{\text{ first 10-digit prime found in consecutive digits of } e\ \}\text{.com}$$

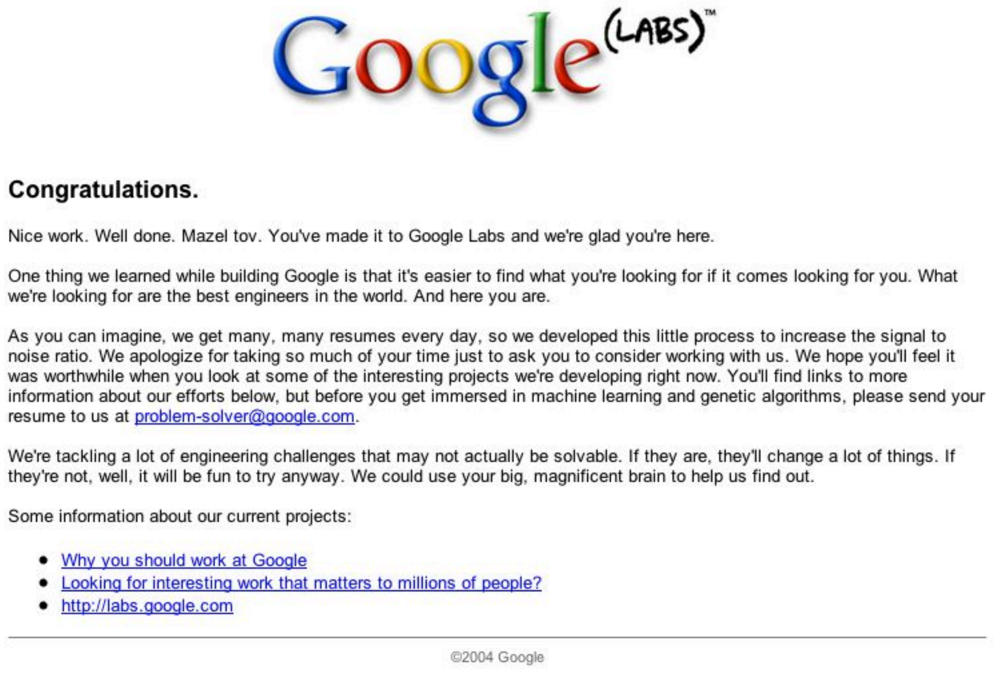
on a banner at the Harvard Square subway stop in Cambridge, Massachusetts, and on a billboard on Highway 101 in California's Silicon Valley — see figure 11.

Figure 11: The Google advertisement.

Those who worked out that this was code for `7427466391.com` and surfed to the URL were greeted with a message and an even more challenging puzzle:

> Congratulations. You've made it to level 2. Go to `www.Linux.org` and enter *Bobsyouruncle* as the login and the answer to this equation as the password.
>
> $$f(1) = 7182818284, \quad f(2) = 8182845904, \quad f(3) = 8747135266,$$
>
> $$f(4) = 7427466391, \quad f(5) = \underline{\hspace{3cm}}.$$

If you found the password and logged in, you landed on the Google recruiting page shown in figure 12, and you had an invitation to apply for a job with Google.



Congratulations.

Nice work. Well done. Mazel tov. You've made it to Google Labs and we're glad you're here.

One thing we learned while building Google is that it's easier to find what you're looking for if it comes looking for you. What we're looking for are the best engineers in the world. And here you are.

As you can imagine, we get many, many resumes every day, so we developed this little process to increase the signal to noise ratio. We apologize for taking so much of your time just to ask you to consider working with us. We hope you'll feel it was worthwhile when you look at some of the interesting projects we're developing right now. You'll find links to more information about our efforts below, but before you get immersed in machine learning and genetic algorithms, please send your resume to us at problem-solver@google.com.

We're tackling a lot of engineering challenges that may not actually be solvable. If they are, they'll change a lot of things. If they're not, well, it will be fun to try anyway. We could use your big, magnificent brain to help us find out.

Some information about our current projects:

- Why you should work at Google
- Looking for interesting work that matters to millions of people?
- http://labs.google.com

©2004 Google

Figure 12: The Google recruiting page.

What was the password? If you guess that this also has something to do with the digits of e, you could go to the website `apod.nasa.gov/htmltest/gifcity/e.2mil`, which gives the first two million digits of e, and search for each of the numbers $f(1),\ldots,f(4)$ as consecutive digits. Indeed, you will find them. But what should $f(5)$ be? Surely it must be ten consecutive digits of e, but which ten? I leave you to solve this final puzzle, with a big hint: add up the digits of each of the numbers $f(1),\ldots,f(4)$.

You'll probably need to write a program to do the search for $f(5)$. I guess that was part of Google's intention.

Appendix: PageRank calculations in Excel

In this section, we give instructions for setting up an Excel spreadsheet to calculate PageRanks for the following system of three web pages.

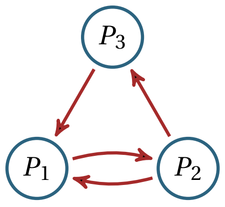
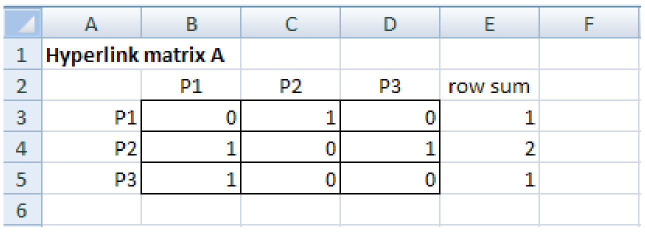


Figure 13: Baby web example.

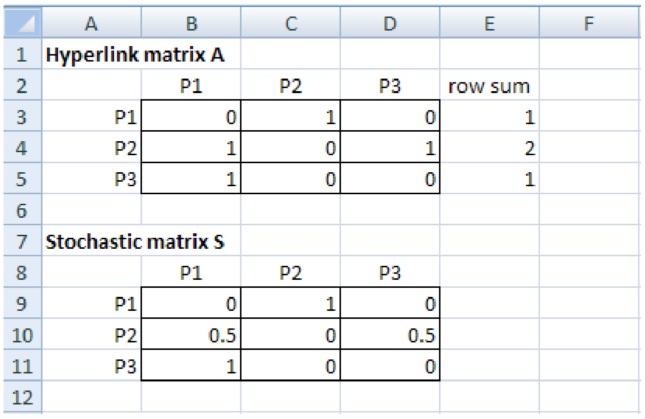
Step 1: The hyperlink matrix. In an Excel spreadsheet, enter the hyperlink matrix A that corresponds to the graph, as shown in the following screenshot. The row sums have been calculated by entering the formula =SUM(B3:D3) in cell E3 and then dragging down to cell E5.



	A	B	C	D	E	F
1	**Hyperlink matrix A**					
2		P1	P2	P3	row sum	
3	P1	0	1	0	1	
4	P2	1	0	1	2	
5	P3	1	0	0	1	
6						

Figure 14: Setting up the hyperlink matrix.

Step 2: The stochastic matrix. Now set up the spreadsheet to automatically calculate the stochastic matrix S from the matrix A: Enter the formula =IF($E3=0, 1/3, B3/$E3) in cell B9, and then drag across and down to create the 3×3 matrix. (Note the important use of dollar signs in this formula, which prevents the column name 'E' from changing as the formula is dragged.)



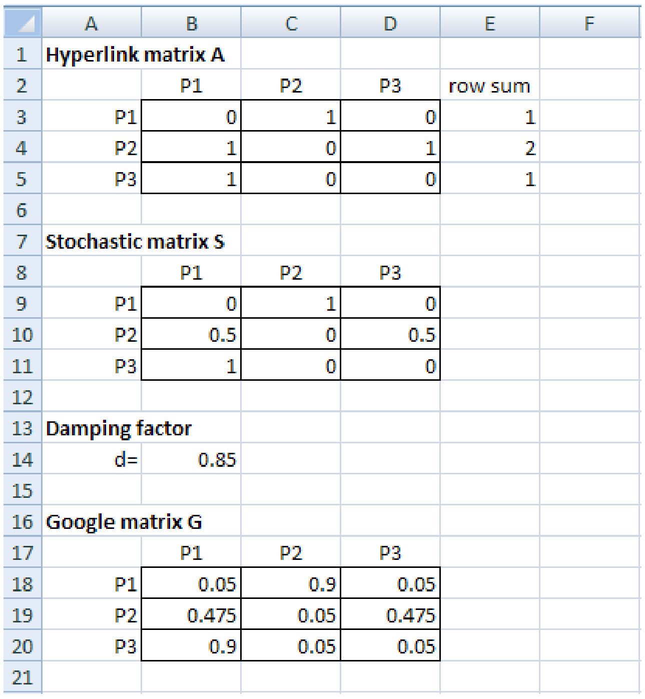
	A	B	C	D	E	F
1	**Hyperlink matrix A**					
2		P1	P2	P3	row sum	
3	P1	0	1	0	1	
4	P2	1	0	1	2	
5	P3	1	0	0	1	
6						
7	**Stochastic matrix S**					
8		P1	P2	P3		
9	P1	0	1	0		
10	P2	0.5	0	0.5		
11	P3	1	0	0		
12						

Figure 15: Setting up the stochastic matrix.

Step 3: The Google matrix. Enter the value 0.85 into cell B14; this will be the damping factor. Now create the Google matrix: Enter the formula

=\$B\$14*B9 + (1-\$B\$14)/3

in cell B18, and then drag across and down.



	A	B	C	D	E	F
1	**Hyperlink matrix A**					
2		P1	P2	P3	row sum	
3	P1	0	1	0	1	
4	P2	1	0	1	2	
5	P3	1	0	0	1	
6						
7	**Stochastic matrix S**					
8		P1	P2	P3		
9	P1	0	1	0		
10	P2	0.5	0	0.5		
11	P3	1	0	0		
12						
13	**Damping factor**					
14	d=	0.85				
15						
16	**Google matrix G**					
17		P1	P2	P3		
18	P1	0.05	0.9	0.05		
19	P2	0.475	0.05	0.475		
20	P3	0.9	0.05	0.05		
21						

Figure 16: Setting up the Google matrix.

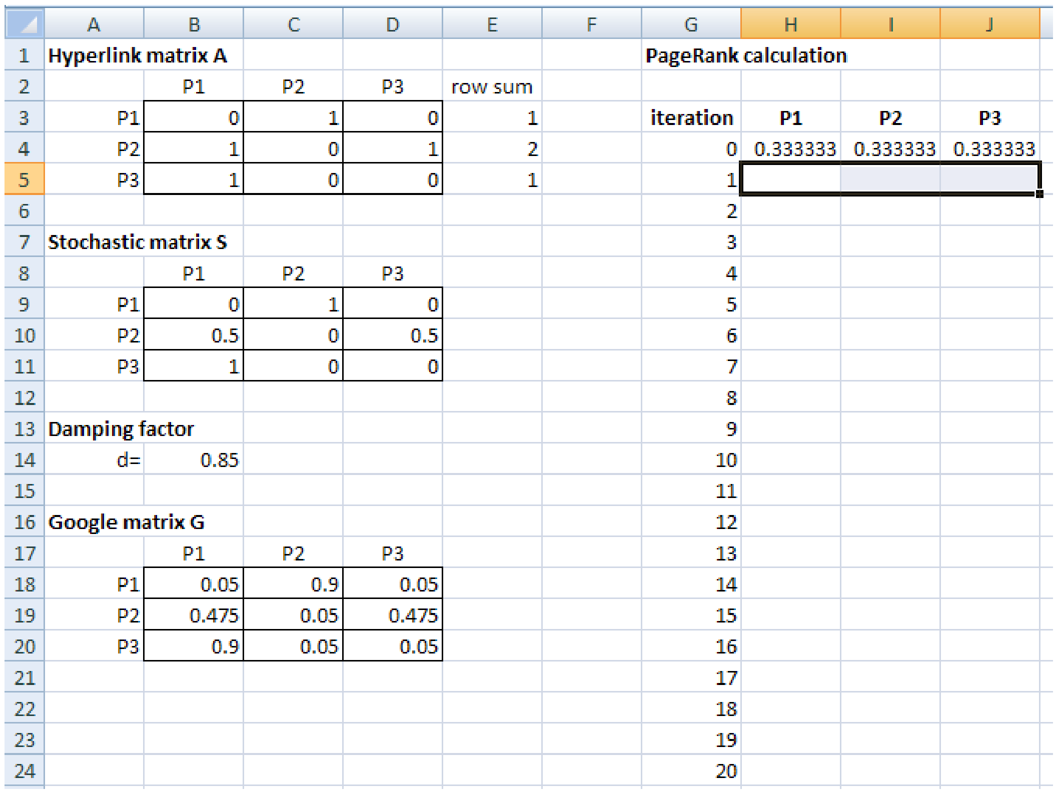
Step 4: First iteration. We're nearly ready to perform the iteration to solve the PageRank equation $\boldsymbol{v} = \boldsymbol{v}G$. The following screenshot shows the preparation for the iteration. The initial vector $\boldsymbol{v}^{(0)} = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ has been added by entering =1/3 in cells H4, I4, J4.

In order to perform the matrix multiplications needed to calculate $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \boldsymbol{v}^{(3)}, \ldots$, we will use an Excel *array formula*.

We first want to calculate the vector $\boldsymbol{v}^{(1)} = \boldsymbol{v}^{(0)}G$. As shown in the following screenshot, highlight the range H5:J5, which is where we want the result of the matrix multiplication. While these cells are still highlighted, type the formula

```
=MMULT(H4:J4, $B$18:$D$20)
```

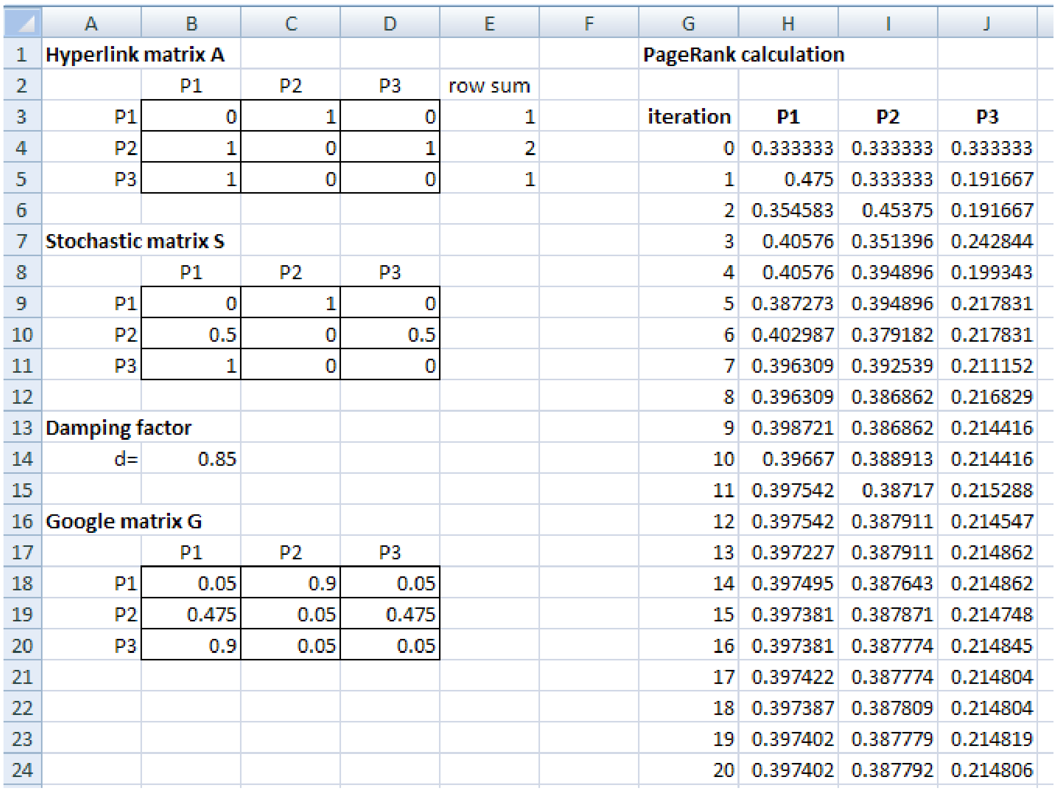
and press Control-Shift-Enter. You should obtain $\boldsymbol{v}^{(1)} = (0.475, 0.333333, 0.191667)$.



	A	B	C	D	E	F	G	H	I	J
1	**Hyperlink matrix A**						**PageRank calculation**			
2		P1	P2	P3	row sum					
3	P1	0	1	0	1		**iteration**	**P1**	**P2**	**P3**
4	P2	1	0	1	2		0	0.333333	0.333333	0.333333
5	P3	1	0	0	1		1			
6							2			
7	**Stochastic matrix S**						3			
8		P1	P2	P3			4			
9	P1	0	1	0			5			
10	P2	0.5	0	0.5			6			
11	P3	1	0	0			7			
12							8			
13	**Damping factor**						9			
14	d=	0.85					10			
15							11			
16	**Google matrix G**						12			
17		P1	P2	P3			13			
18	P1	0.05	0.9	0.05			14			
19	P2	0.475	0.05	0.475			15			
20	P3	0.9	0.05	0.05			16			
21							17			
22							18			
23							19			
24							20			

Figure 17: Preparation for the iteration.

Step 5: More iterations. Highlight the range H5:J5 again, and drag down to obtain an appropriate number of iterations. Your spreadsheet should now match the one in the following screenshot. We have found that the PageRanks of P_1, P_2 and P_3 are 0.397, 0.388 and 0.215, respectively.



	A	B	C	D	E	F	G	H	I	J
1	Hyperlink matrix A						PageRank calculation			
2		P1	P2	P3	row sum					
3	P1	0	1	0	1		iteration	P1	P2	P3
4	P2	1	0	1	2		0	0.333333	0.333333	0.333333
5	P3	1	0	0	1		1	0.475	0.333333	0.191667
6							2	0.354583	0.45375	0.191667
7	Stochastic matrix S						3	0.40576	0.351396	0.242844
8		P1	P2	P3			4	0.40576	0.394896	0.199343
9	P1	0	1	0			5	0.387273	0.394896	0.217831
10	P2	0.5	0	0.5			6	0.402987	0.379182	0.217831
11	P3	1	0	0			7	0.396309	0.392539	0.211152
12							8	0.396309	0.386862	0.216829
13	Damping factor						9	0.398721	0.386862	0.214416
14	d=	0.85					10	0.39667	0.388913	0.214416
15							11	0.397542	0.38717	0.215288
16	Google matrix G						12	0.397542	0.387911	0.214547
17		P1	P2	P3			13	0.397227	0.387911	0.214862
18	P1	0.05	0.9	0.05			14	0.397495	0.387643	0.214862
19	P2	0.475	0.05	0.475			15	0.397381	0.387871	0.214748
20	P3	0.9	0.05	0.05			16	0.397381	0.387774	0.214845
21							17	0.397422	0.387774	0.214804
22							18	0.397387	0.387809	0.214804
23							19	0.397402	0.387779	0.214819
24							20	0.397402	0.387792	0.214806

Figure 18: Final spreadsheet.

Now that the spreadsheet is set up, you can experiment by

- changing the initial probability vector $\boldsymbol{v}^0$
- changing the value of the damping factor d
- changing the hyperlink matrix A to correspond to a different three-node graph W.

Using a similar process, you can also set up a spreadsheet to check the calculations for the six-page 'mini web' example used throughout these notes.

References

General reference

- Amy N. Langville and Carl D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*, Princeton University Press, 2006.

Technical references

- S. Brin and L. Page, 'The anatomy of a large-scale hypertextual Web search engine', *Computer Networks and ISDN Systems* 33 (1998), 107–117.
- O. Perron, 'Zur Theorie der Matrices', *Mathematische Annalen* 64 (1907), 248–263.