Exploring Applications of Linear Algebra

Linear thinking

about

Google™

Tim Chartier

DAVIDSON

Department of Mathematics

tichartier@davidson.edu

This webinar was made possible, in part, with funding from the Associated Colleges of the South.

5 clicks to Jesus

A form of Wikiracing that mimics golf

<u>**Challenge**</u>:

- Surf from a Random Article to the Jesus entry of Wikipedia in as few clicks as possible.
- Reaching the article in 5 clicks is considered 'par', with clicks over or under five being referred to as 'bogeys' and 'birdies' respectively.
- Lowest score wins!

WIKIPEDIA

Random start

EXPLORE & RACE THROUGH WIKIPEDIA ARTICLES

THE WIKI GAME

You can also compete against others in this game by visiting:

http://thewikigame.com/5-clicks-to-jesus

Random page on wikipedia → **HOW MANY CLICKS?**

Terminology

As you surfed through Wikipedia, you:

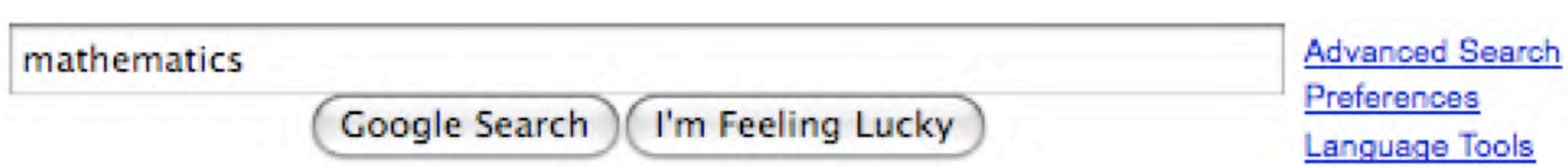
- clicked a link (*outlink* or *hyperlink*) on a web page to go to another page.
- used the hyperlink structure of Wikipedia to surf. That is, you got from one place to another only by clicking links.

A web address is also called a URL.

WIKIPEDIA

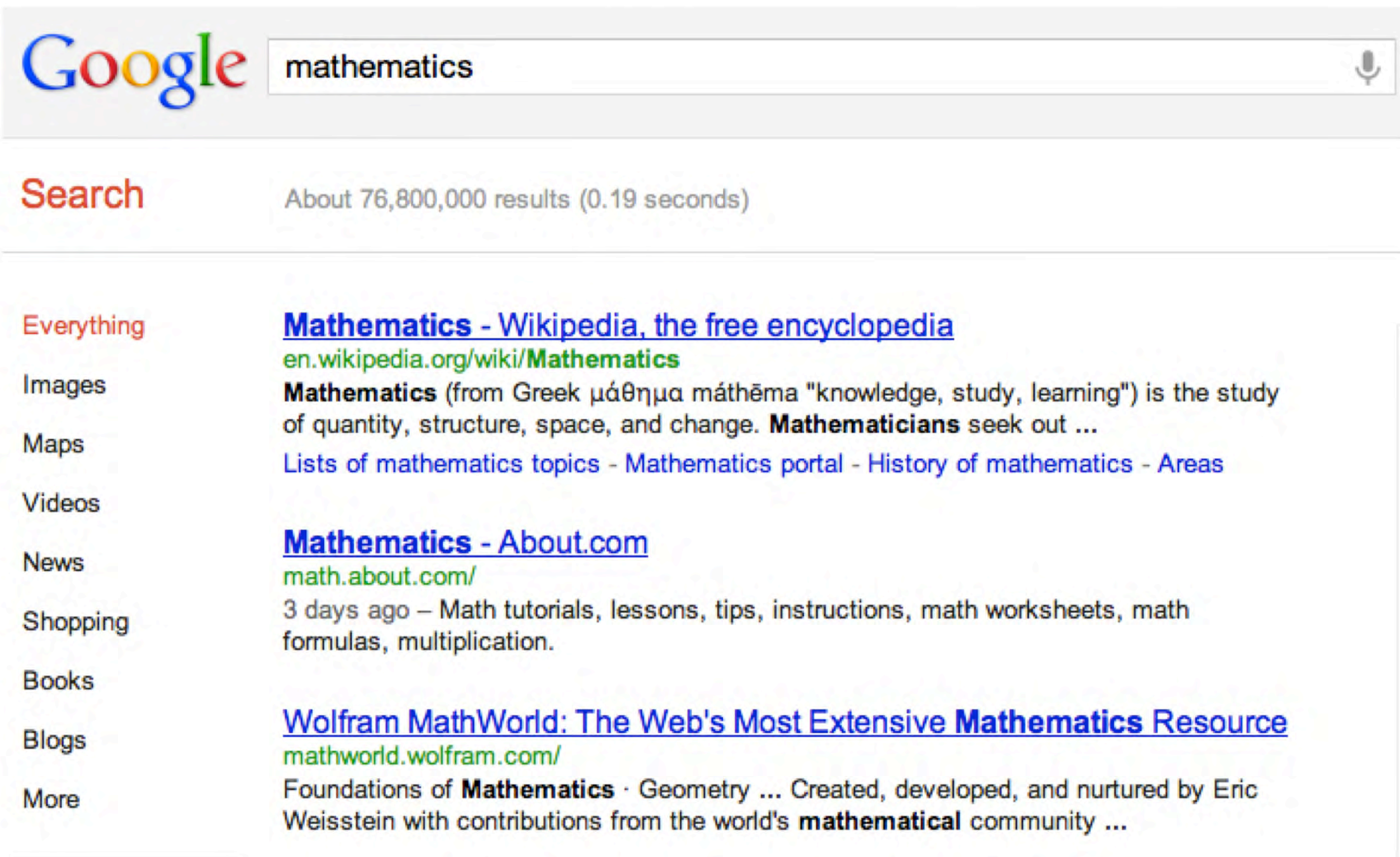
Query

- Your Wikipedia surfing will help us understand the linear algebra used by Google.
- Suppose you submit the word “mathematics” to Google.



Ranked results

A ranked list of web pages is returned.



PageRank

- Assuming 2 web pages are deemed equally relevant to a query, why is one page ranked over the other?
- Google measures the quality of pages.
- Quality pages are linked by quality pages!

Random Surfer

- PageRank measures quality by the hyperlink structure of the web.
- It models internet activity as as the actions of a random surfer who randomly follows links on a web page.

http://3.bp.blogspot.com/_wC_5u4IkdIQ/RwyB9fKTQwI/AAAAAAAAAAw/jIuvtvqR-dU/s400/surfer-dude-cartoon-sketch.jpg

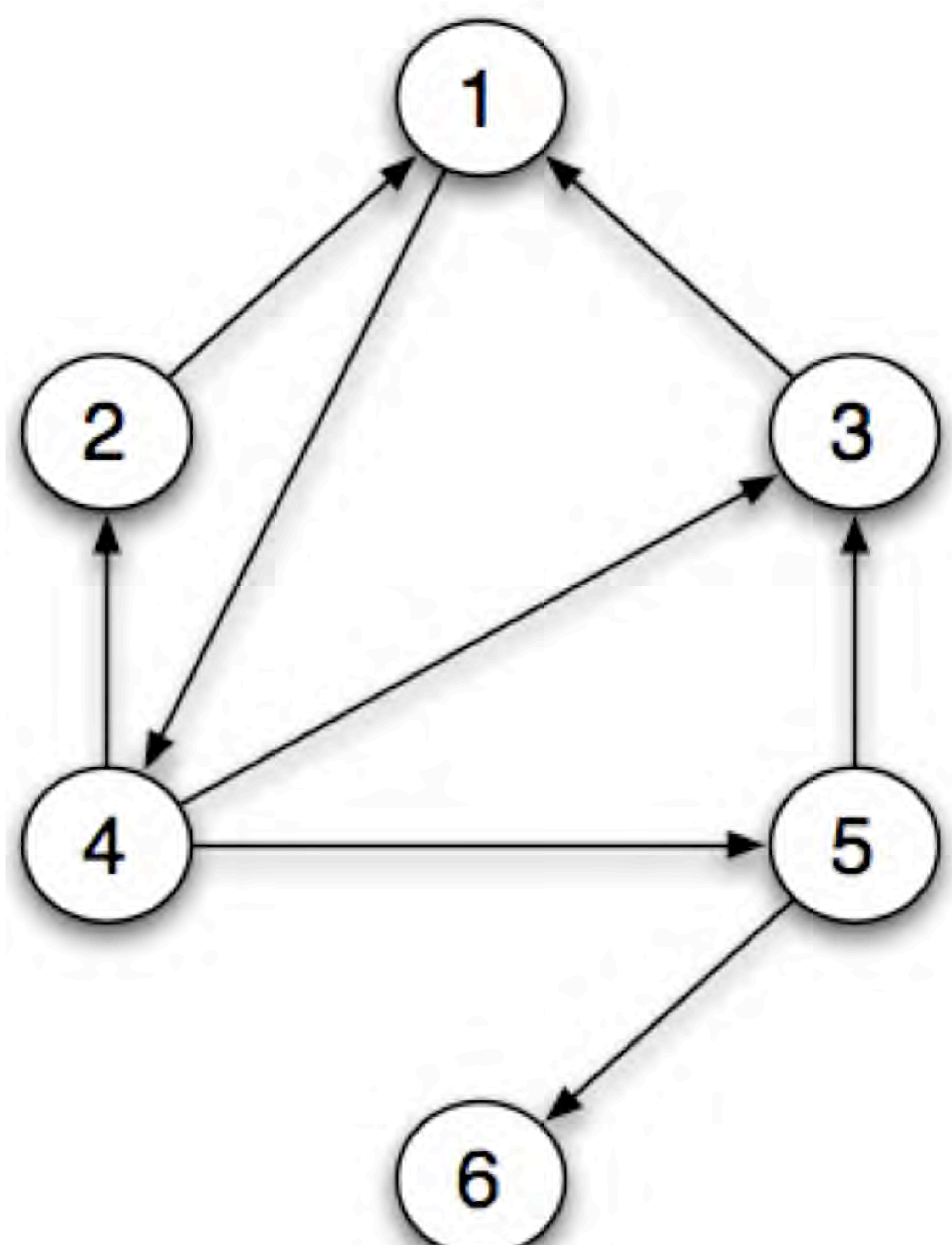
Chance visit

- Suppose a random surfer surfed the web indefinitely.
- The probability he visits a web page is that pages PageRank.
- Higher PageRank correlates to higher quality.

Picture credit:http://www.comicsbeat.com/wp-content/uploads/2010/04/monopoly_junior_chance_card_small.jpg

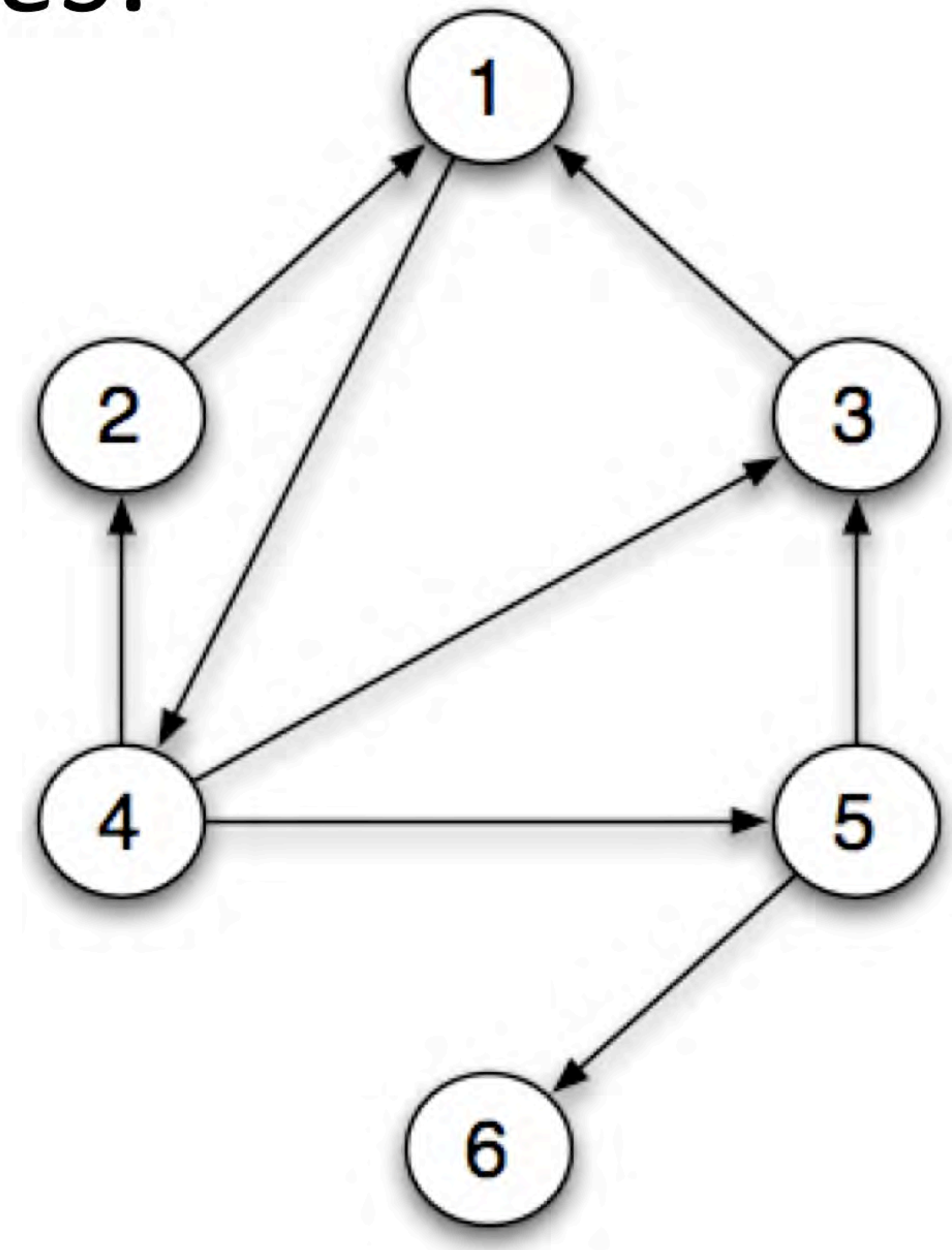
Linked in

- Earlier, we surfed only be following links.
- This isn't a realistic model of surfing.
- Why?



Caught dangling

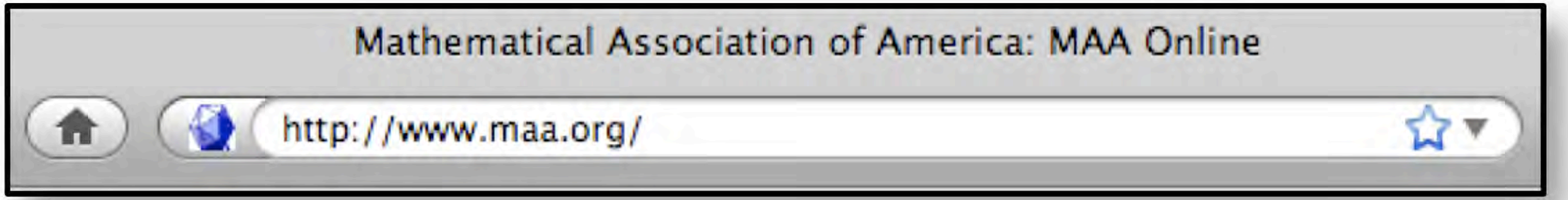
- If you can't jump, you can get stuck!
- Web pages with no outlinks are called *dangling nodes*.



Teleporting

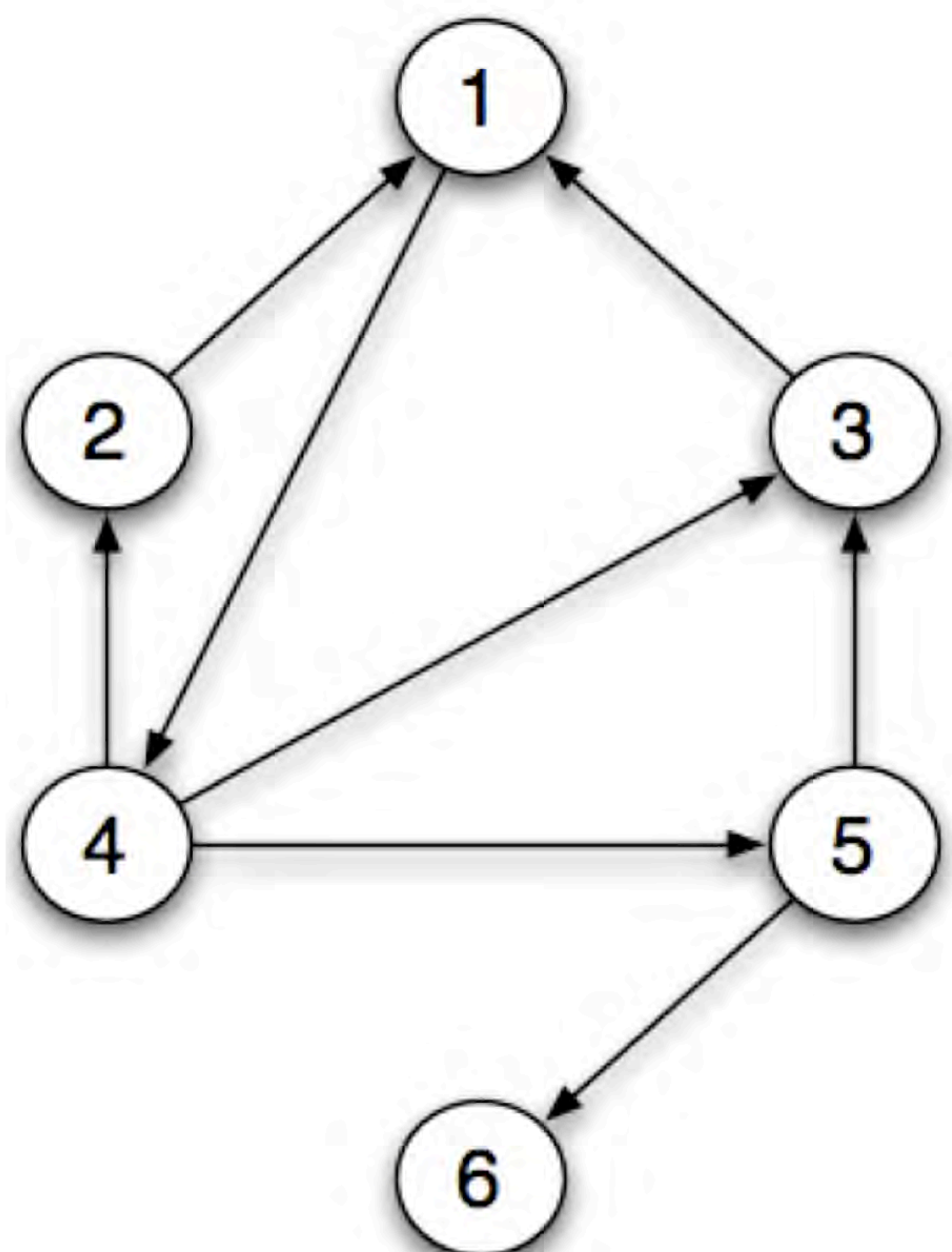
The PageRank model assumes

- 85% chance of following a hyperlink on a page
- 15% chance of jumping to any web page in the network (with uniform probability).



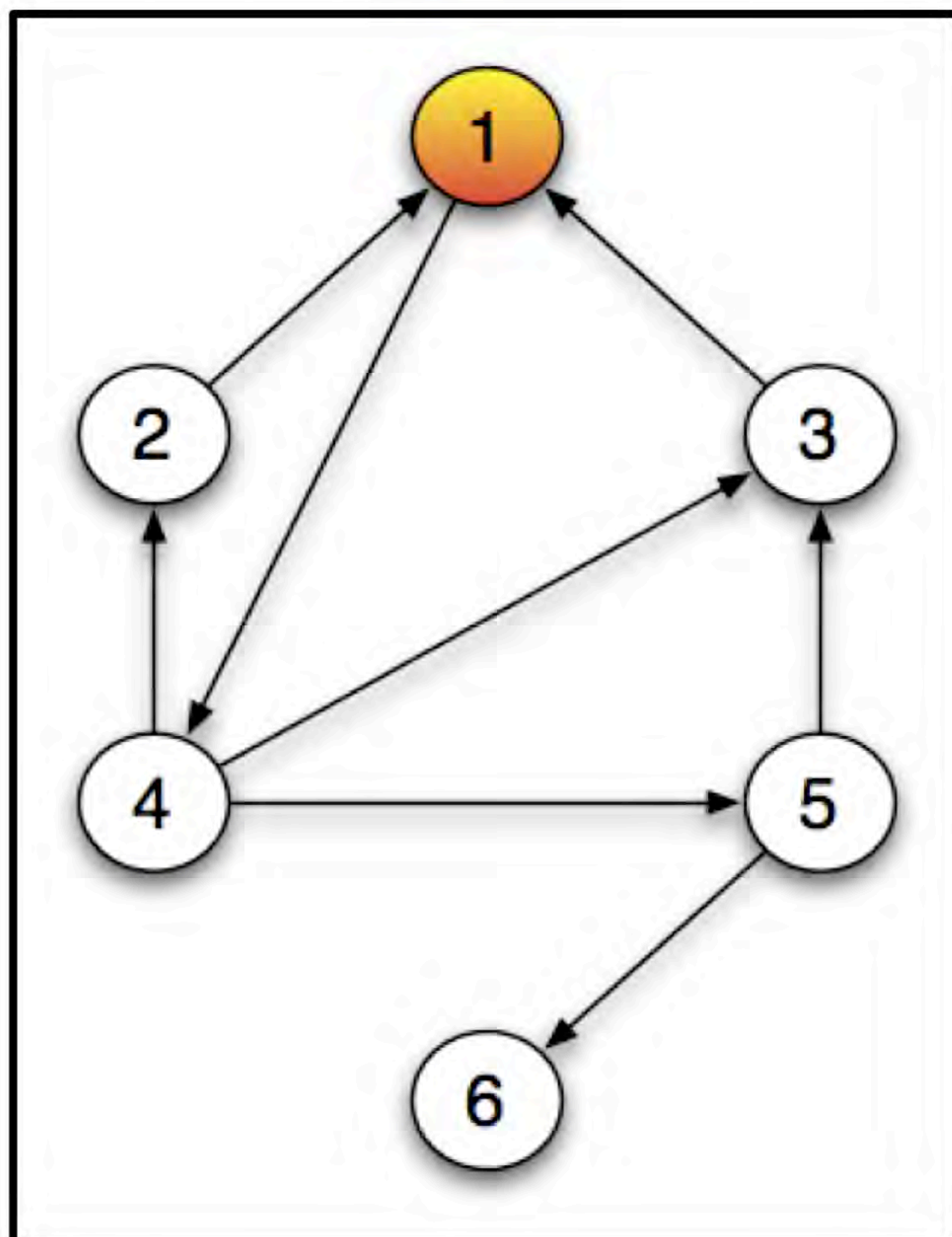
Google in Monte Carlo

We can use Monte Carlo simulation to determine the quality of pages.

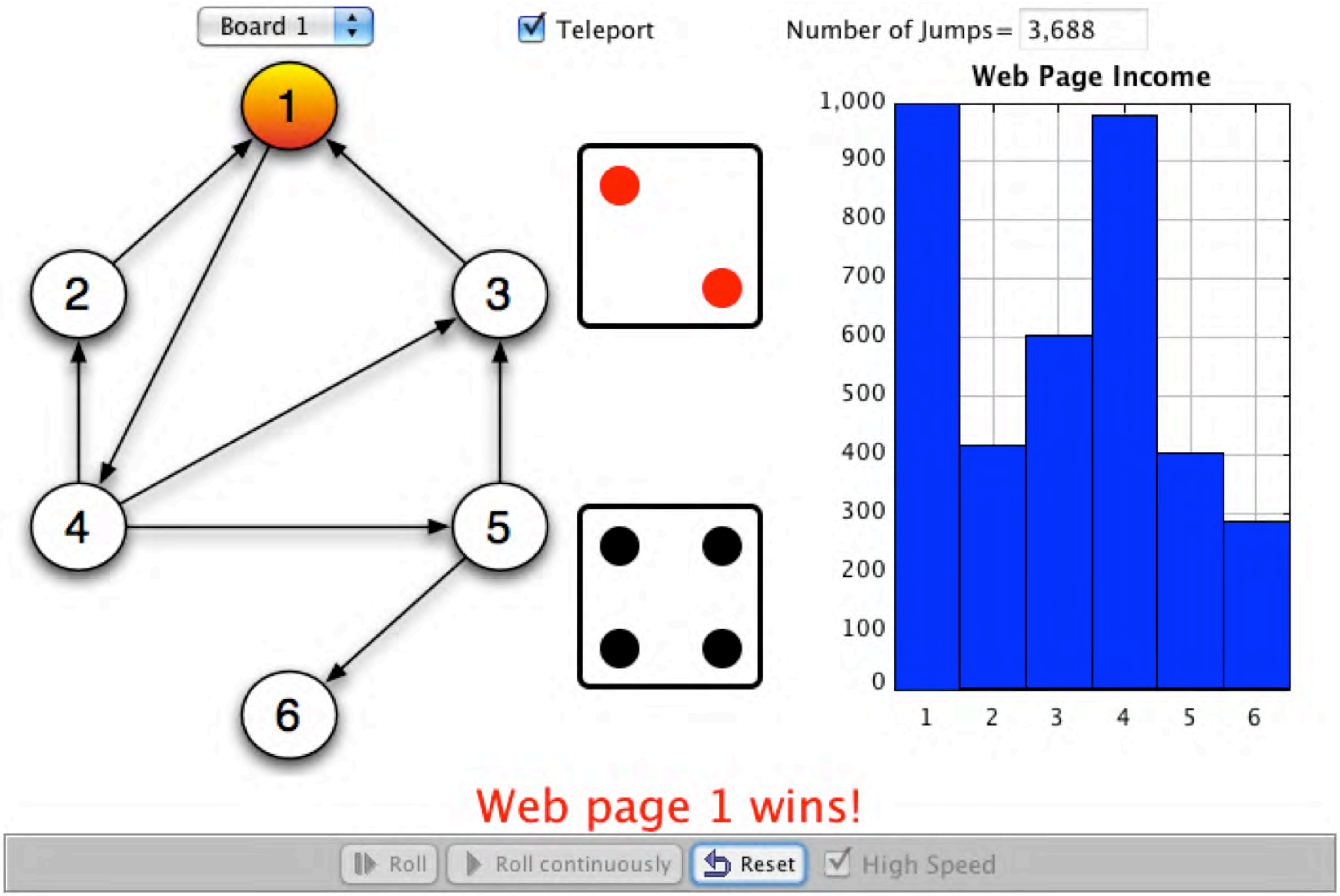


Simulation

- Let's compute with simulation.
- We'll use a die as a random number generator.

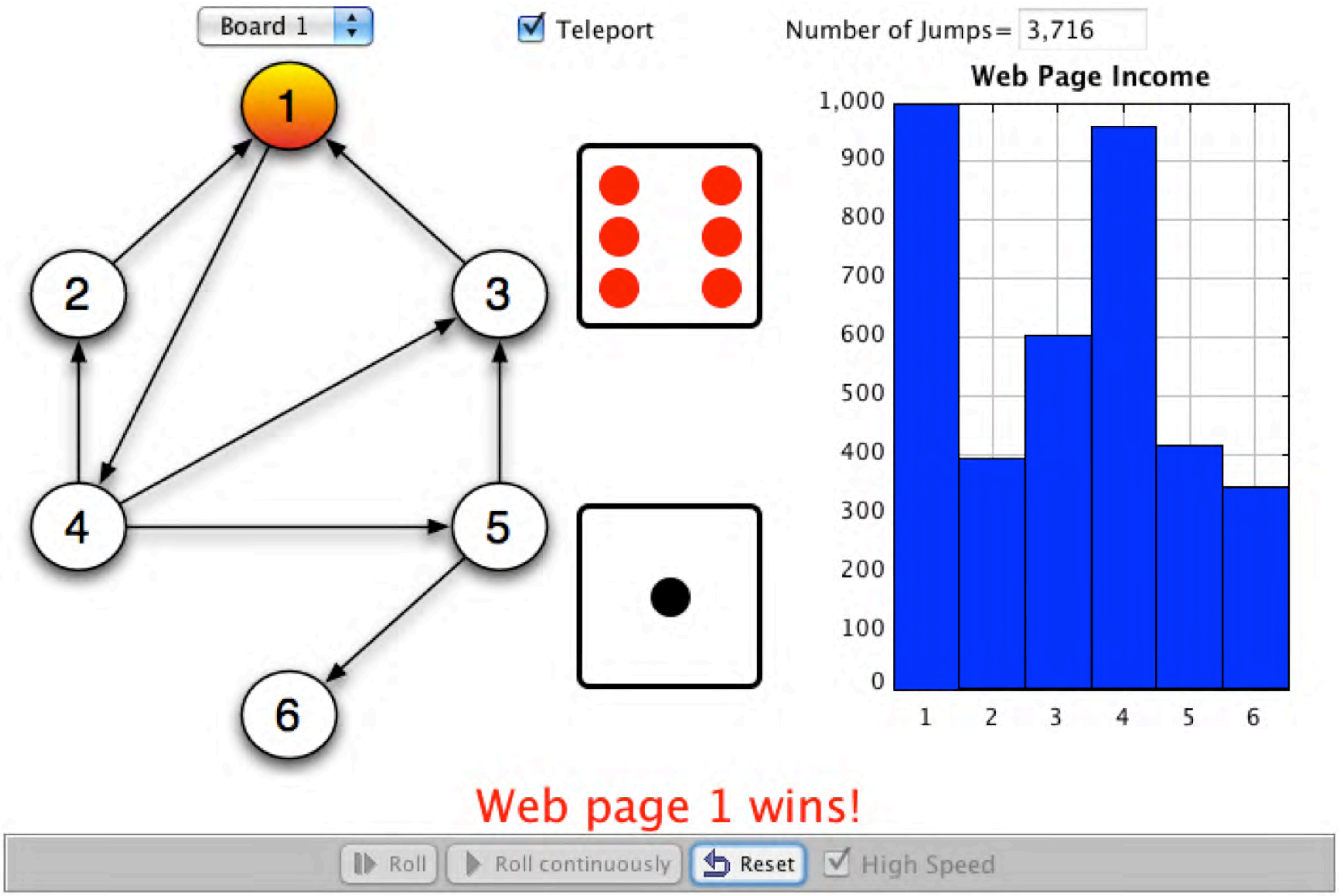


Run 1



Board 1
Teleport
Number of Jumps= 3,688
Web Page Income
1,000
900
800
700
600
500
400
300
200
100
0
1
2
3
4
5
6
1
2
3
4
5
6
Web page 1 wins!
Roll
Roll continuously
Reset
High Speed

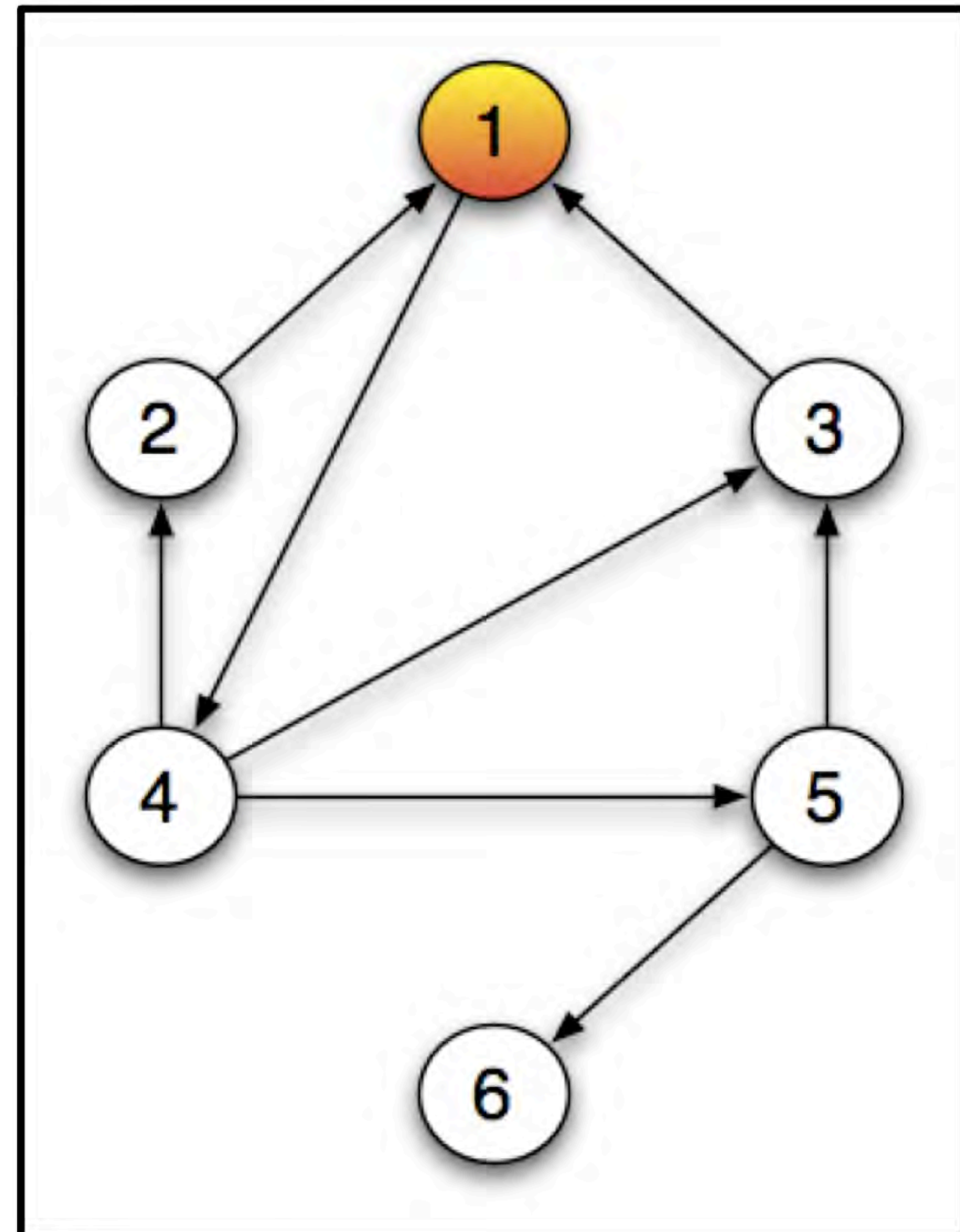
Run 2



Web page 1 wins!

Google™-OPOLY

- Let's compute with simulation using a die as a random number generator.
- For more details see the *Loci* article "Google-oply" by C., Kreutzer, Langville and Pedings available at:



http://mathdl.maa.org/mathDL/23/?sa=viewDocument&pa=content&nodeId=3355

linear algebra?

Wait a minute? Where is the linear algebra?

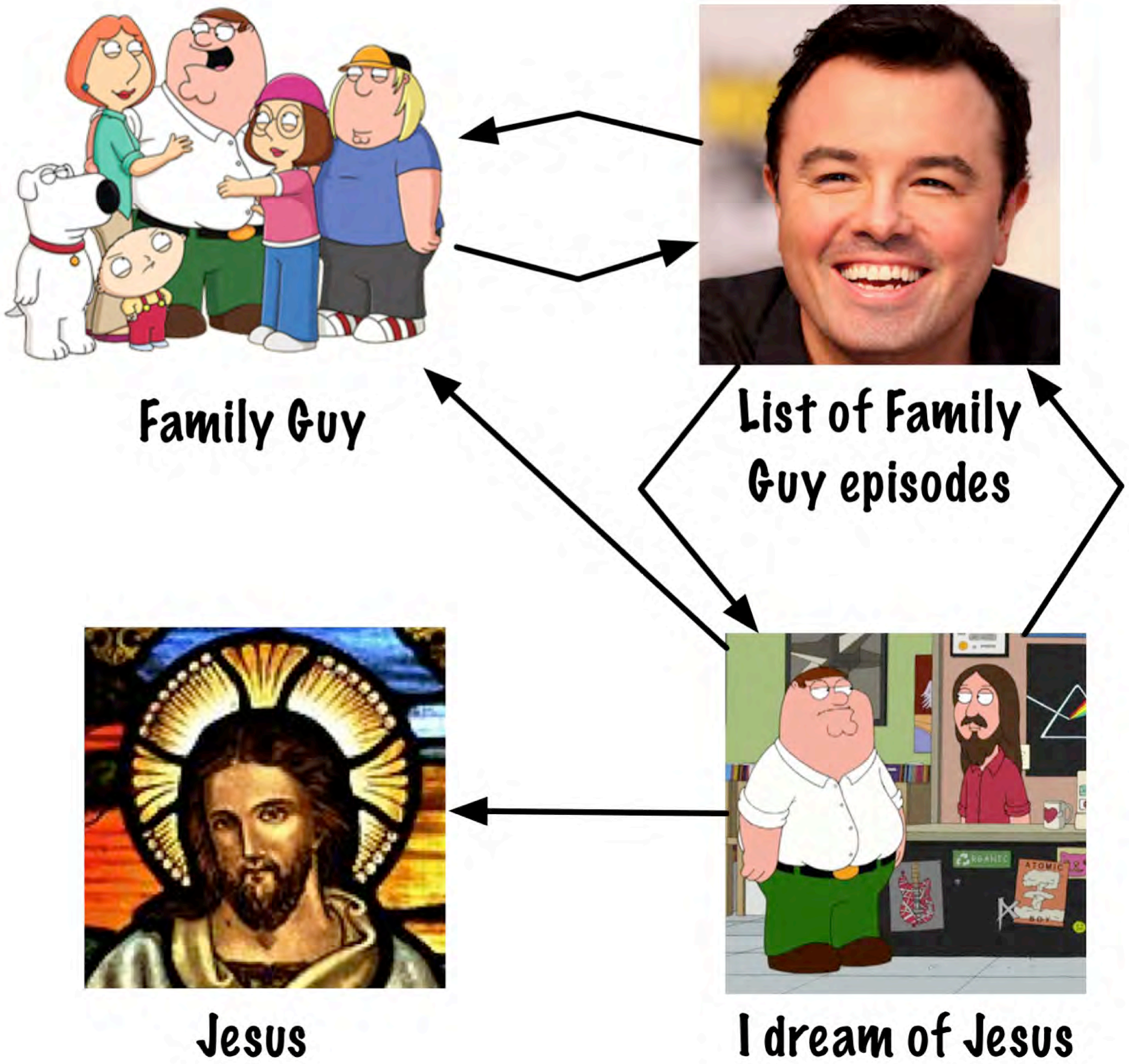
Billion-sided die?

- Rather than Google rolling some billion-sided die, it uses linear algebra.
- In fact, we use math ideas developed 100 years ago.

Picture credit:http://online.wsj.com/media/info-GiftGuide0711-Gdice.jpg

Wiki-Jesus

- Let's return to the 5-clicks to Jesus exercise.
- Here is a path from the Family Guy to Jesus pages on Wikipedia.
- Let's consider Google's model constrained only to this network.

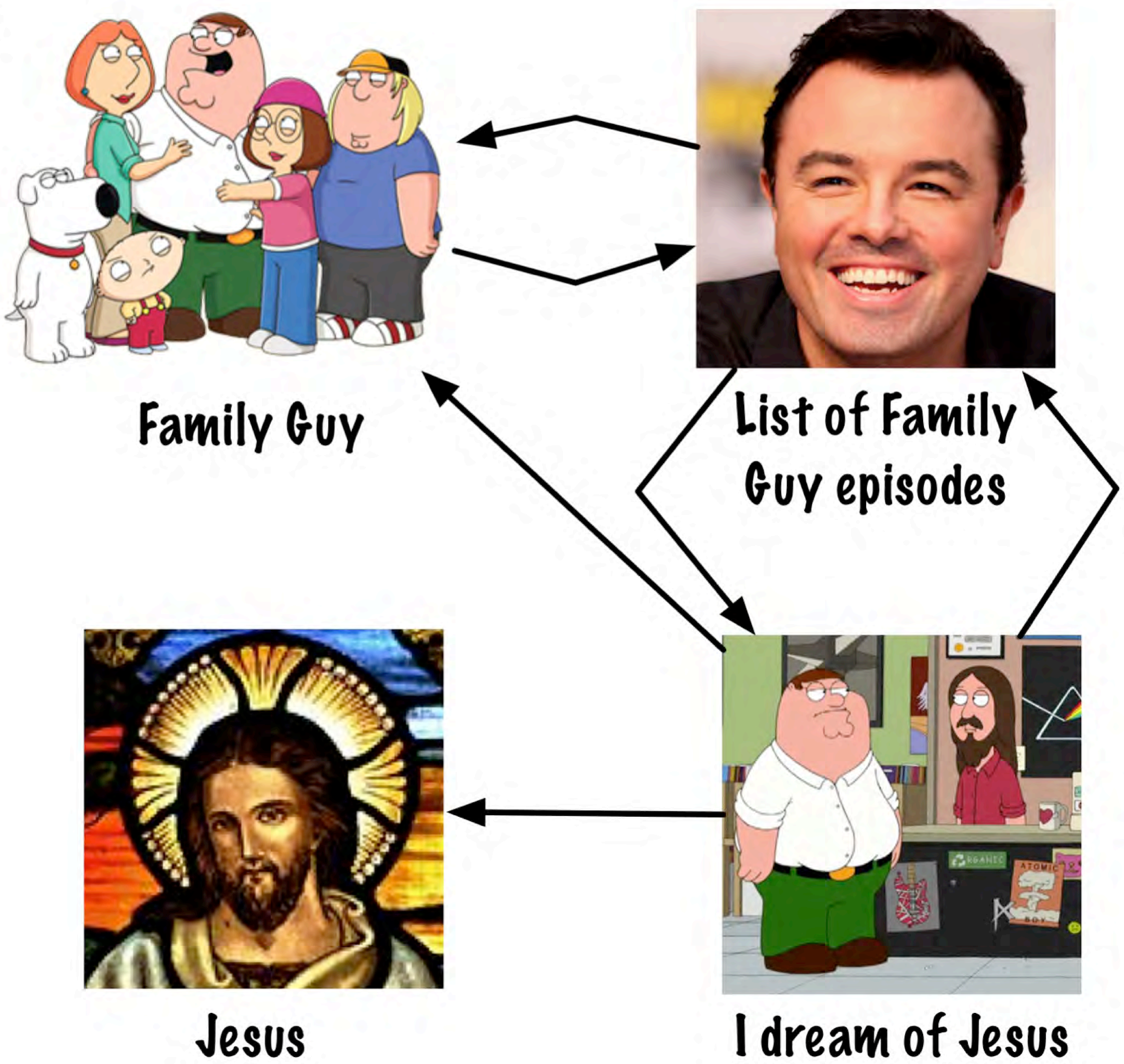


Probable surfing

Under Google's model, if you are at the *Family Guy* web page, what is the probability of:

- visiting the page listing episodes?

- visiting Jesus?



Probable surfing

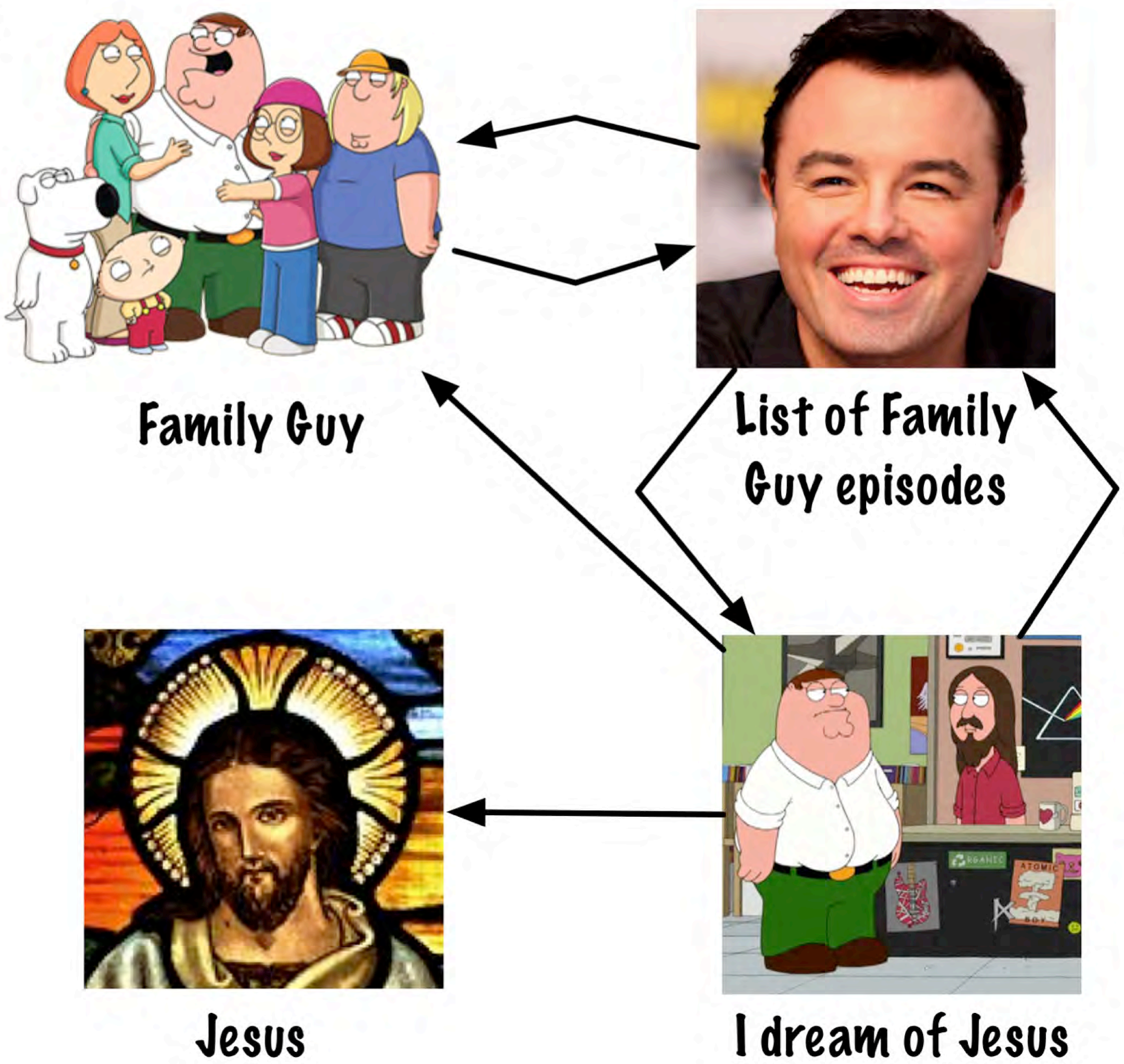
Under Google's model, if you are at the *Family Guy* web page, what is the probability of:

- visiting the page listing episodes?

  .85 + .15/4 = .8875

- visiting Jesus?

  .0375



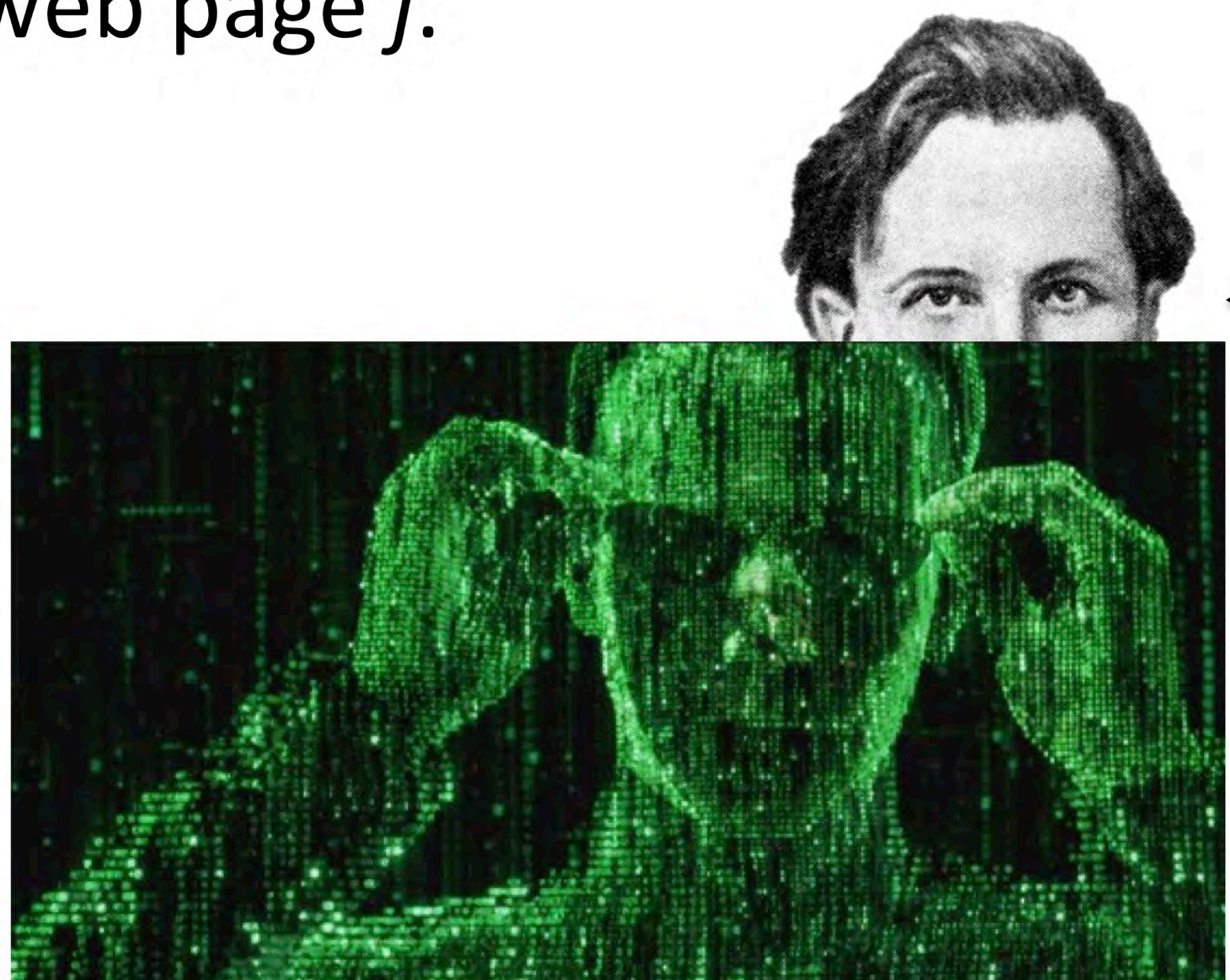
Leaning on Markov

- Finding the probability of visiting web page *j* from web page *i* allows us to use Markov Chains (processes).
- First used for linguistic purposes to model the letter sequences in works of Russian literature.

Andrei Andreevich Markov
(1856 - 1922)

Enter the matrix

We create a transition matrix G where g_{ij} equals the probability of moving from web page i to web page j.



Markov

Time to Order

First, order the columns (and rows)

(

column 1

column 2

column 3

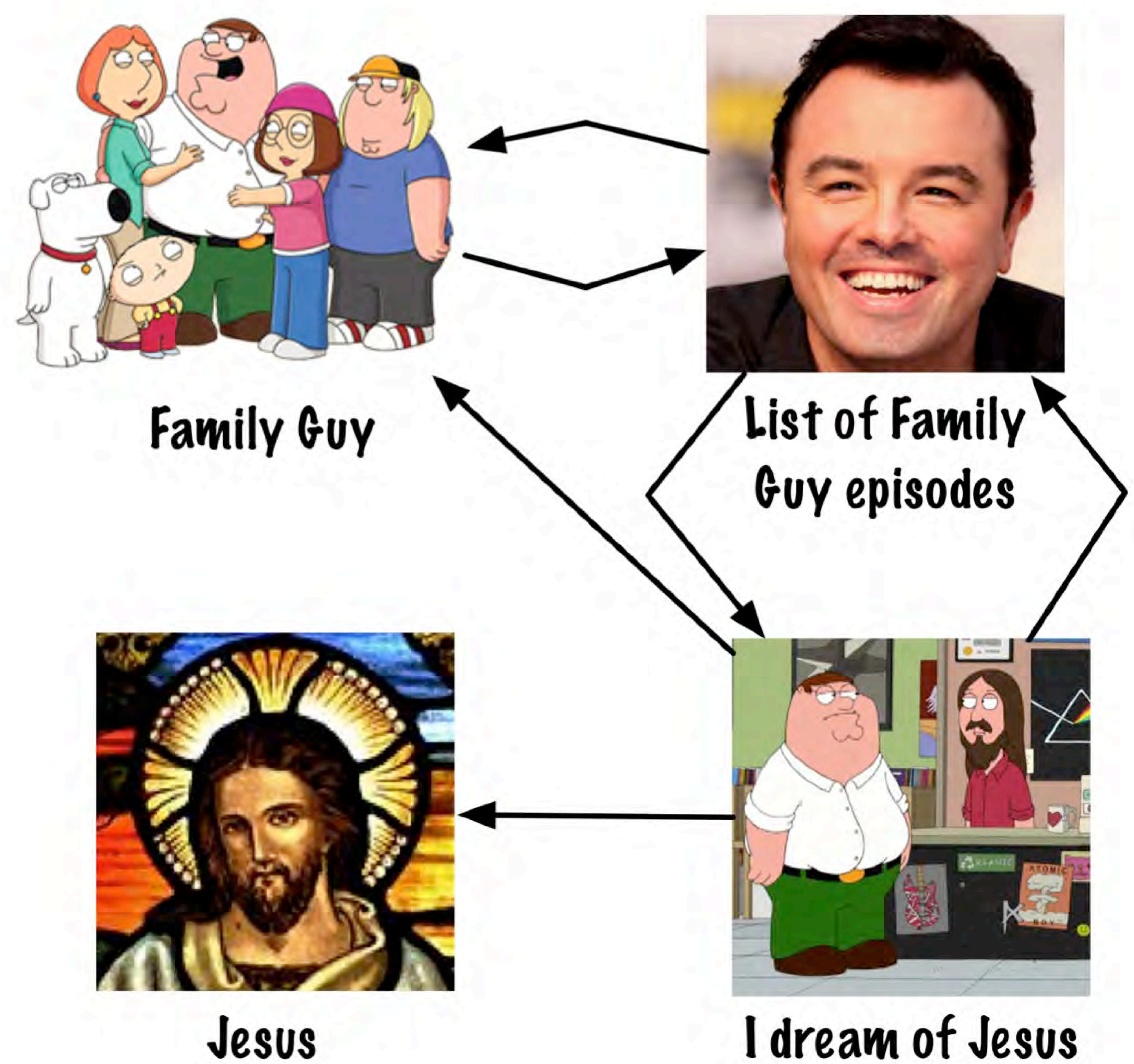
column 4

)

Row

The first row contains the probabilities of jumping from web page 1 to other web pages.

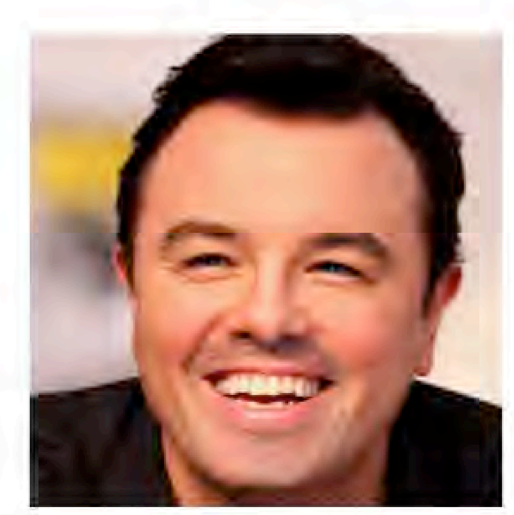
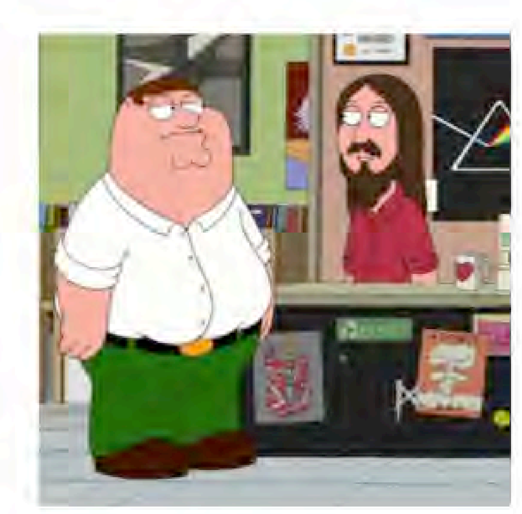
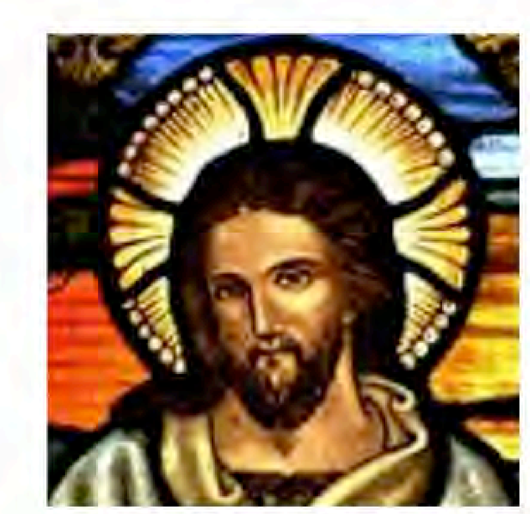
$$\begin{pmatrix} 0.0375 & 0.8875 & 0.0375 & 0.0375 \end{pmatrix}$$



row, row, row

So, the entire transition matrix becomes:

$$G = \begin{pmatrix} 0.0375 & 0.8875 & 0.0375 & 0.0375 \\ 0.4625 & 0.0375 & 0.4625 & 0.0375 \\ 0.3208 & 0.3208 & 0.0375 & 0.3208 \\ 0.2500 & 0.2500 & 0.2500 & 0.2500 \end{pmatrix}$$

<u>Note</u>: The entries of each row sum to 1.

baby steps

- We can then walk through a series of steps.
- Assume we start at *Family Guy*, then

$$\begin{aligned}
\mathbf{v}_0 G &= \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0.0375 & 0.8875 & 0.0375 & 0.0375 \\ 0.4625 & 0.0375 & 0.4625 & 0.0375 \\ 0.3208 & 0.3208 & 0.0375 & 0.3208 \\ 0.2500 & 0.2500 & 0.2500 & 0.2500 \end{pmatrix} \\
&= \begin{pmatrix} 0.0375 & 0.8875 & 0.0375 & 0.0375 \end{pmatrix} \\
&= \mathbf{v}_1
\end{aligned}$$

The one step

- Since

$$\begin{aligned}
\mathbf{v}_0 G &= \begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix} \begin{pmatrix} 0.0375 & 0.8875 & 0.0375 & 0.0375 \\ 0.4625 & 0.0375 & 0.4625 & 0.0375 \\ 0.3208 & 0.3208 & 0.0375 & 0.3208 \\ 0.2500 & 0.2500 & 0.2500 & 0.2500 \end{pmatrix} \\
&= \begin{pmatrix}0.0375 & 0.8875 & 0.0375 & 0.0375\end{pmatrix} \\
&= \mathbf{v}_1
\end{aligned}$$

- We know the probability of being at each web page after one step assuming we start at web page 1.

Step by step

- Where will you be after two steps?

$$
\begin{aligned}
\mathbf{v}_1 G &= \begin{pmatrix} 0.0375 \\ 0.8875 \\ 0.0375 \\ 0.0375 \end{pmatrix}^T \begin{pmatrix} 0.0375 & 0.8875 & 0.0375 & 0.0375 \\ 0.4625 & 0.0375 & 0.4625 & 0.0375 \\ 0.3208 & 0.3208 & 0.0375 & 0.3208 \\ 0.2500 & 0.2500 & 0.2500 & 0.2500 \end{pmatrix} \\
&= \begin{pmatrix} 0.4333 & 0.0880 & 0.4227 & 0.0561 \end{pmatrix} \\
&= \mathbf{v}_2
\end{aligned}
$$

- But, how do we find the probability of being at each web page after infinitely many steps?

Iterating

Note that:

$$v_2 = v_1 G = v_0 G^2,$$

$$v_3 = v_2 G = v_0 G^3,$$

$$\vdots$$

$$v_n = v_{n-1} G = v_0 G^n,$$

Lotsa steps

So, let's take many more steps:

$$\mathbf{v}_0 G^{100} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}^T \begin{pmatrix} 0.0375 & 0.8875 & 0.0375 & 0.0375 \\ 0.4625 & 0.0375 & 0.4625 & 0.0375 \\ 0.3208 & 0.3208 & 0.0375 & 0.3208 \\ 0.2500 & 0.2500 & 0.2500 & 0.2500 \end{pmatrix}^{100}$$

$$= \begin{pmatrix} 0.2836 & 0.3682 & 0.2210 & 0.1271 \end{pmatrix}$$

$$= \mathbf{v}_{100} = \mathbf{v}_{200} \text{ (to 4 decimal places)}$$

We have converged to the *steady-state vector.*

Steady

- In fact, for this vector, we reach steady state (to 4 decimal places) at the 18th step, which will be very important to Google.
- This gives us the PageRank of these pages:

$$\mathbf{v} = \begin{pmatrix} 0.2836 & 0.3682 & 0.2210 & 0.1271 \end{pmatrix}$$




Keep in mind that Google indexes billions of web pages!

Image thanks to David Gleich

Questions!

- Will this process converge for any network of web pages?
- Is there more than one steady-state vector?
- Will this scale up to billions of pages?

Stepping in place

- The steady-state vector has property:

$$\mathbf{v}A = \mathbf{v}$$

- This relationship means that the vector **v** is an eigenvector of *A* with an associated eigenvalue of 1.
- Recall if **v** is an eigenvector of *A* then *c***v** is an eigenvector of *A* for any nonzero scalar *c*.
- We want *c***v** such that the elements of **v** sum to 1.

Unique solution

- Why does this Markov process converge to an e-vector associated with the e-value 1?
- Further, is this even a unique eigenvector?
- Both are guaranteed for PageRank.

<u>Theorem</u> (Perron) Every real square matrix *P* who entries are all positive has a unique eigenvector with all positive entries, its corresponding eigenvalue has multiplicity one, and it is the dominant eigenvalue, in that every other eigenvalue has strictly smaller magnitude.

Time to dominate

- Let M be a Markov transition matrix.
- The rows of M sum to 1. So, $M\mathbf{1} = \mathbf{1}$, where $\mathbf{1}$ is the column vector of all ones.
- So, $\mathbf{1}$ is a right eigenvector of M associated with the eigenvalue 1.
- Perron's Theorem ensures that $\mathbf{1}$ is the unique right eigenvector with all positive entries, and hence its eigenvalue must be the dominant one.

Right from the left

- The right and left *eigenvalues* of a matrix are the same, therefore 1 is the dominant left eigenvalue as well.
- So, there exists a unique steady-state vector **v** that satisfies $\mathbf{v}M = \mathbf{v}$.
- Normalizing this eigenvector so the sum of its entries are 1 gives *the* steady-state vector.
- Perron's Theorem also guarantees this vector has positive entries.

Converging

- To find PageRank, one simply iterate with:

$$\mathbf{v}_{n+1} = \mathbf{v}_n G$$

  until we have convergence.

- Why does this Markov process converge to the dominant e-vector? We will use:

$$|\lambda^n| \to 0 \text{ as } n \to \infty \text{ if } |\lambda| < 1,$$

$$|\lambda^n| = 1 \text{ for all } n \text{ if } |\lambda| = 1,$$

$$|\lambda^n| \to \infty \text{ as } n \to \infty \text{ if } |\lambda| > 1,$$

Full combo

- Assume M has n linear independent eigenvectors.
- Let's take an arbitrary initial guess $\mathbf{x}$
- We can express it as a linear combination of the eigenvectors

$$\mathbf{x}^{(0)} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \ldots + c_n\mathbf{v}_n$$

Full combo

- After one iteration of the Markov chain:

$$\begin{aligned}\mathbf{x}^{(1)} &= \mathbf{x}^{(0)}M \\ &= c_1\mathbf{v}_1M + c_2\mathbf{v}_2M + \cdots + c_n\mathbf{v}_nM \\ &= c_1\lambda_1\mathbf{v}_1 + c_2\lambda_2\mathbf{v}_2 + \cdots + c_n\lambda_n\mathbf{v}_n\end{aligned}$$

- Multiplying again by *M* yields

$$\begin{aligned}\mathbf{x}^{(2)} &= \mathbf{x}^{(1)}M \\ &= c_1\lambda_1\mathbf{v}_1M + c_2\lambda_2\mathbf{v}_2M + \cdots + c_n\lambda_n\mathbf{v}_nM \\ &= c_1\lambda_1^2\mathbf{v}_1 + c_2\lambda_2^2\mathbf{v}_2 + \cdots + c_n\lambda_n^2\mathbf{v}_n\end{aligned}$$

Establishing a pattern

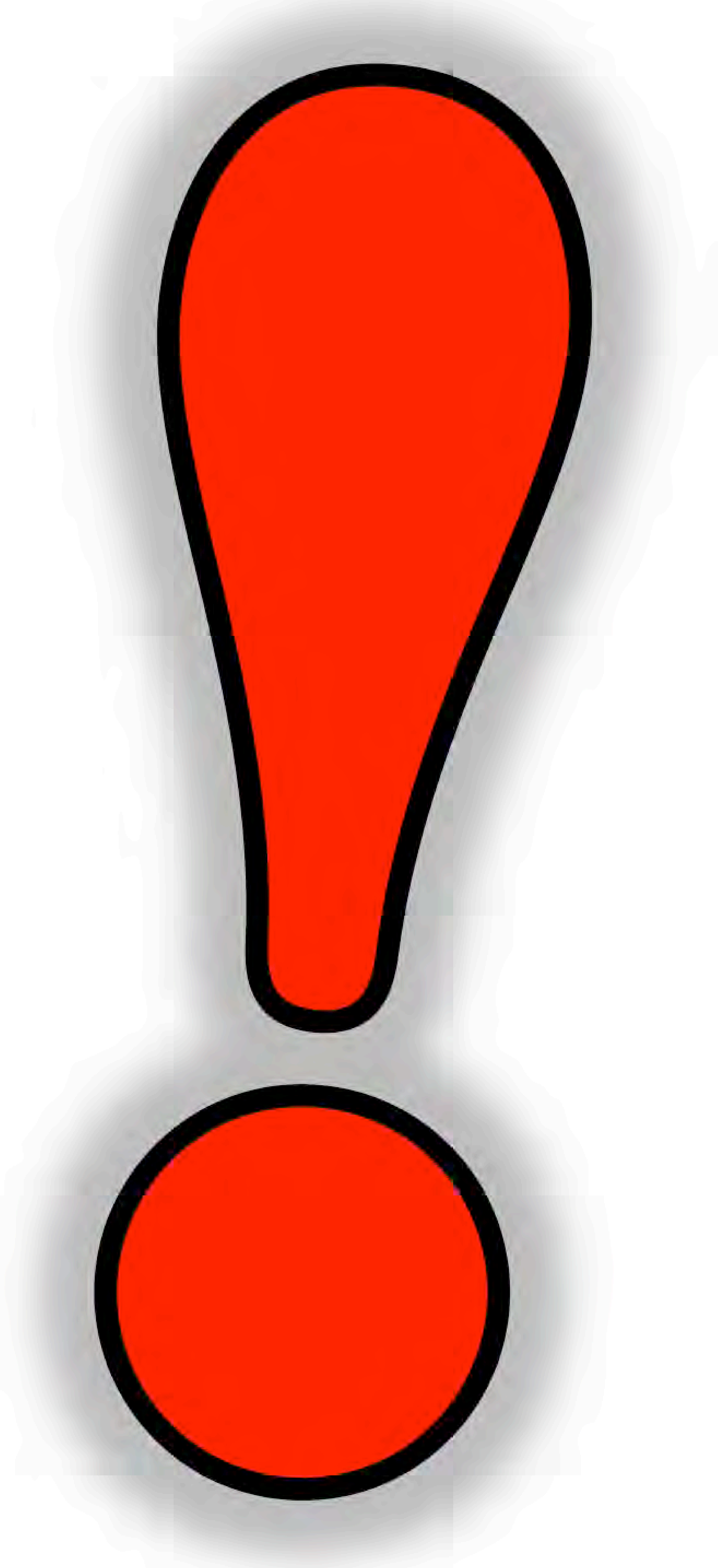
- In general:

$$\begin{aligned} \mathbf{x}^{(k)} &= \mathbf{x}^{(k-1)} M \\ &= c_1 \lambda_1^k \mathbf{v}_1 + c_2 \lambda_2^k \mathbf{v}_2 + \cdots + c_n \lambda_n^k \mathbf{v}_n \end{aligned}$$

- Recall, we know from Perron's theorem that $\lambda_1 = 1$ and $\lambda_i < 1$ for $i > 1$.
- So, our Markov process will converge to $c_1\mathbf{v}$.
- But, c_1 will equal 1 since the sum of the entries of $\mathbf{x}_0$ is 1.

Answers!

- Will this process converge for any network of web pages?
- Is there more than one steady-state vector?
- Will this scale up to billions of pages?



Googling Twitter

- Let's try this entire process on a group of web pages.
- In particular, we'll take pages from Twitter for the celebrities below.


BillGates


JimmyFallon


KimKardashian


PaulaAbdul


RyanSeacrest

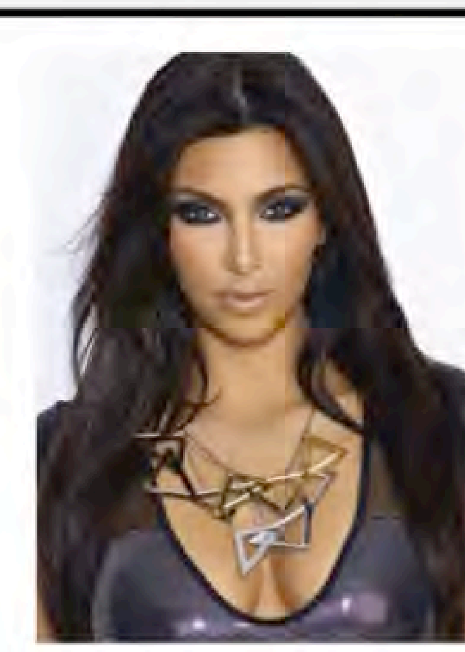

TheEllenShow

Twitter on the Web

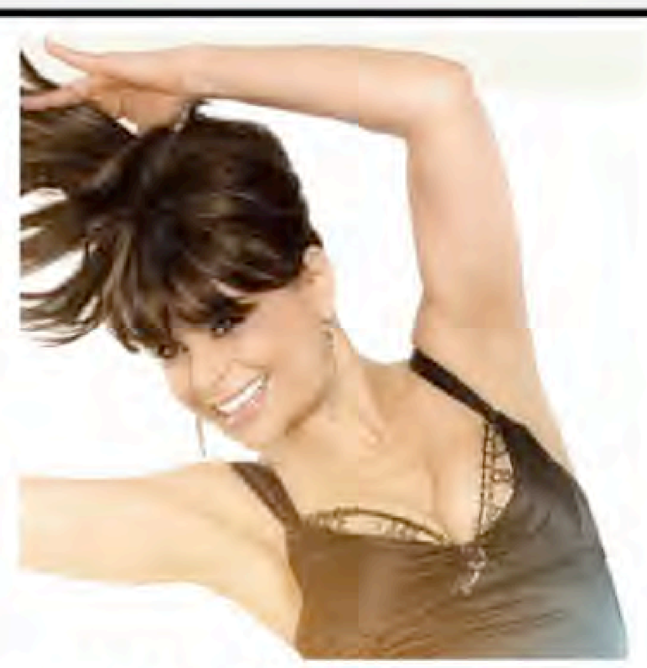
- The names are listed in terms of the celebrity's screen name on Twitter.
- If you want to view Bill Gates' Twitter web page at http://www.twitter.com/billgates.
- You don't need a Twitter account to view this webpage.

BillGates

JimmyFallon



KimKardashian



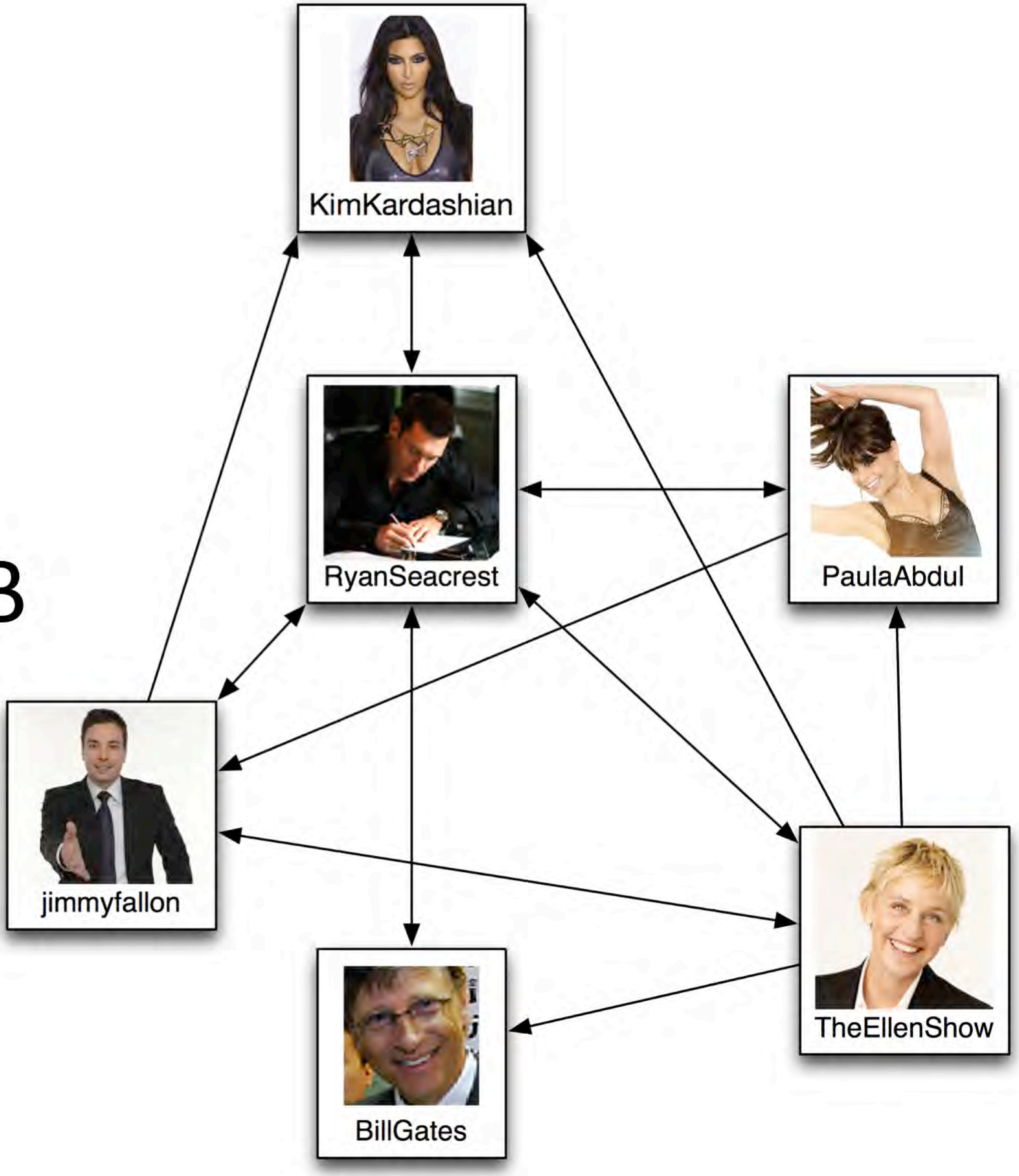
PaulaAbdul

RyanSeacrest

TheEllenShow

Graphic Celebrities

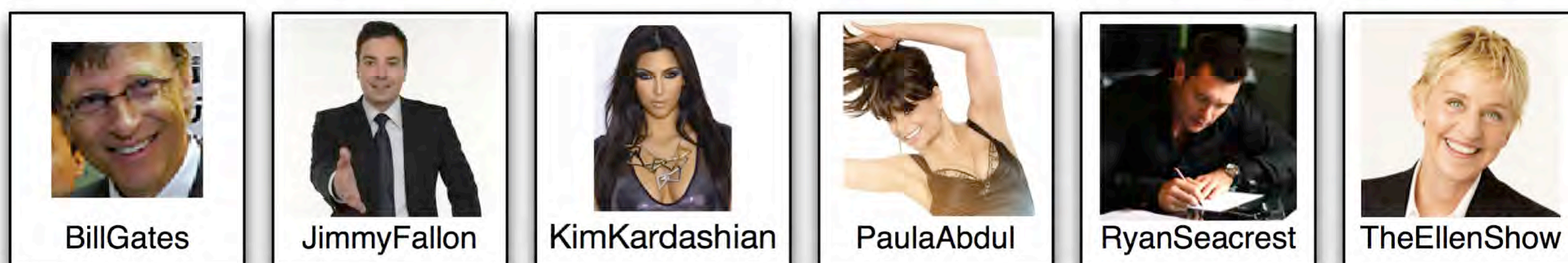
- Here is the graph of connectivity of the celebrities on Twitter.
- There is an edge from celebrity A to celebrity B if celebrity A follows celebrity B on Twitter.



Google matrix

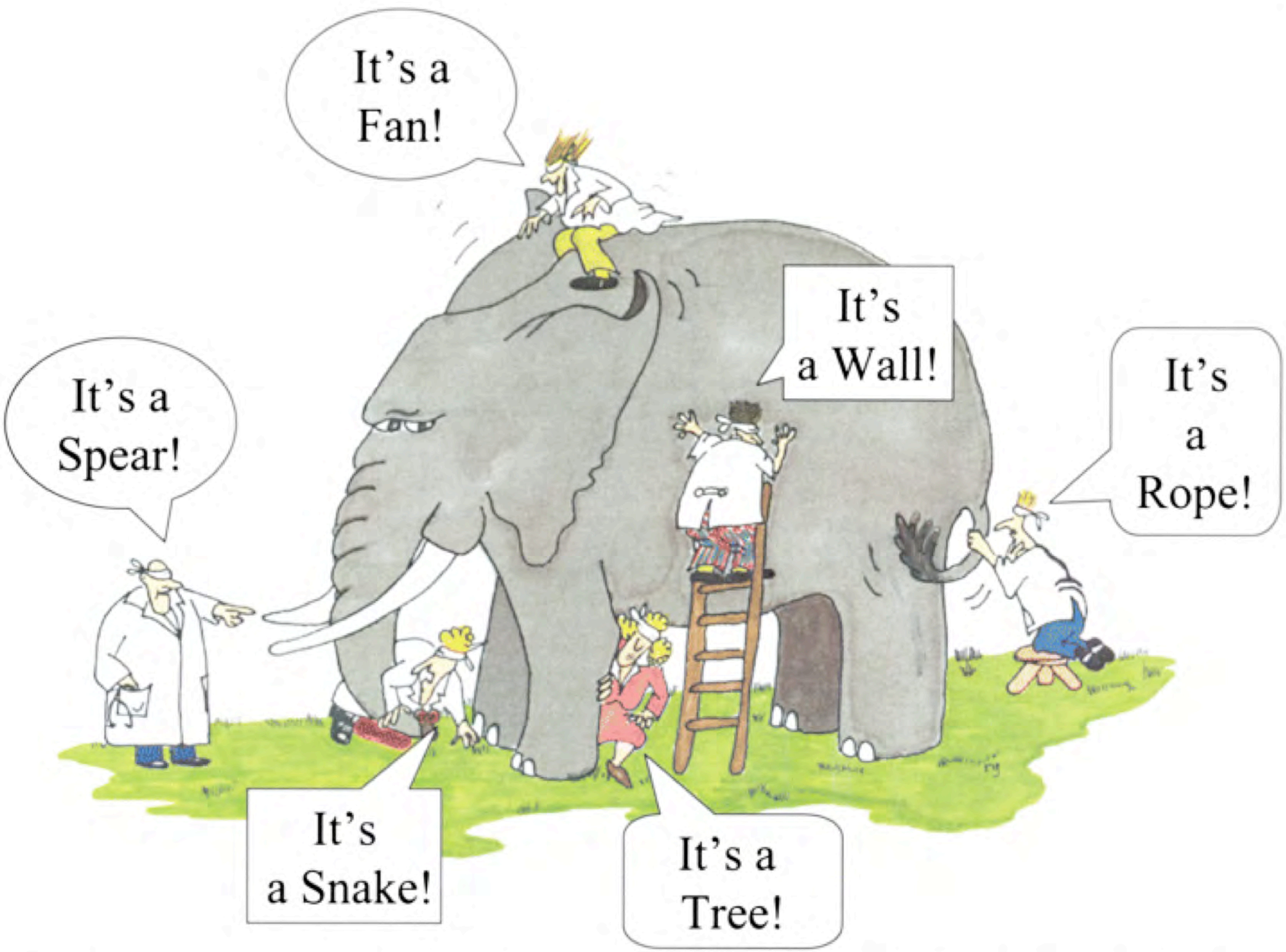
First, we form the Google matrix:

$$G = \begin{pmatrix} 0.025 & 0.025 & 0.0250 & 0.025 & 0.8750 & 0.0250 \\ 0.025 & 0.025 & 0.3083 & 0.025 & 0.3083 & 0.3083 \\ 0.025 & 0.025 & 0.0250 & 0.025 & 0.8750 & 0.0250 \\ 0.025 & 0.450 & 0.0250 & 0.025 & 0.4500 & 0.0250 \\ 0.195 & 0.195 & 0.1950 & 0.195 & 0.0250 & 0.1950 \\ 0.195 & 0.195 & 0.1950 & 0.195 & 0.1950 & 0.0250 \end{pmatrix}$$



Different perspectives

Let’s compute PageRank in 3 different ways.



Picture credit: http://www.halogensoftware.com/blog/wp-content/uploads/2011/08/different-perspectives.gif

Method 1

The first technique to finding the PageRank for these web pages is to compute:

$$\mathbf{v}^{100} = [1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]\, M^{100}$$

$$\mathbf{v} = \begin{pmatrix} 0.1071 & 0.1526 & 0.1503 & 0.1071 & 0.3544 & 0.1285 \end{pmatrix}$$

Note: Taking a matrix to a high power is impractical computationally for a large number of web pages.

Method 2

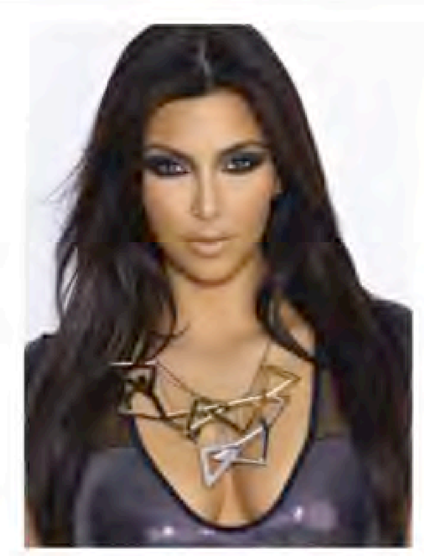
Iterate:

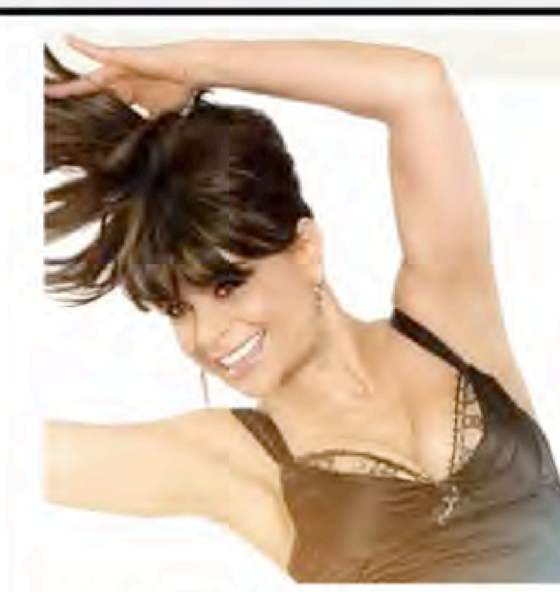
$$\mathbf{v}_{k+1} = \mathbf{v}_k M,$$

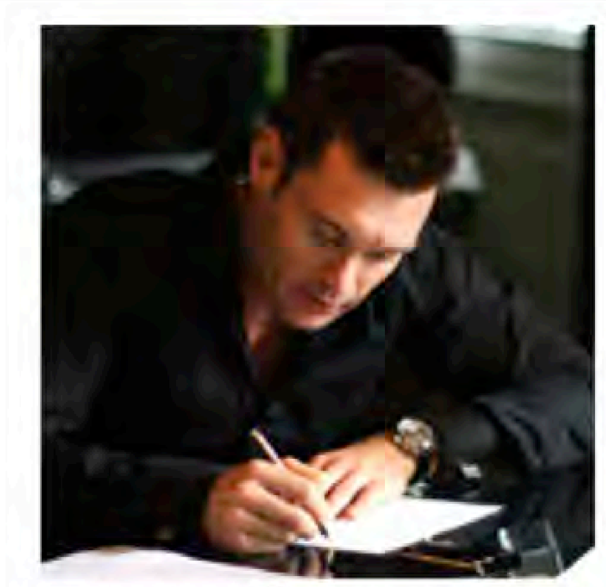
until the elements of $\mathbf{v}_k$ have suitably converged.


BillGates


JimmyFallon


KimKardashian


PaulaAbdul


RyanSeacrest


TheEllenShow

$$\mathbf{v} = \begin{pmatrix} 0.1071 & 0.1526 & 0.1503 & 0.1071 & 0.3544 & 0.1285 \end{pmatrix}$$

<u>Note:</u> This is the Power Method and is the algorithm of choice for computing PageRank.

Method 3

- Compute the (left) eigenvectors of M

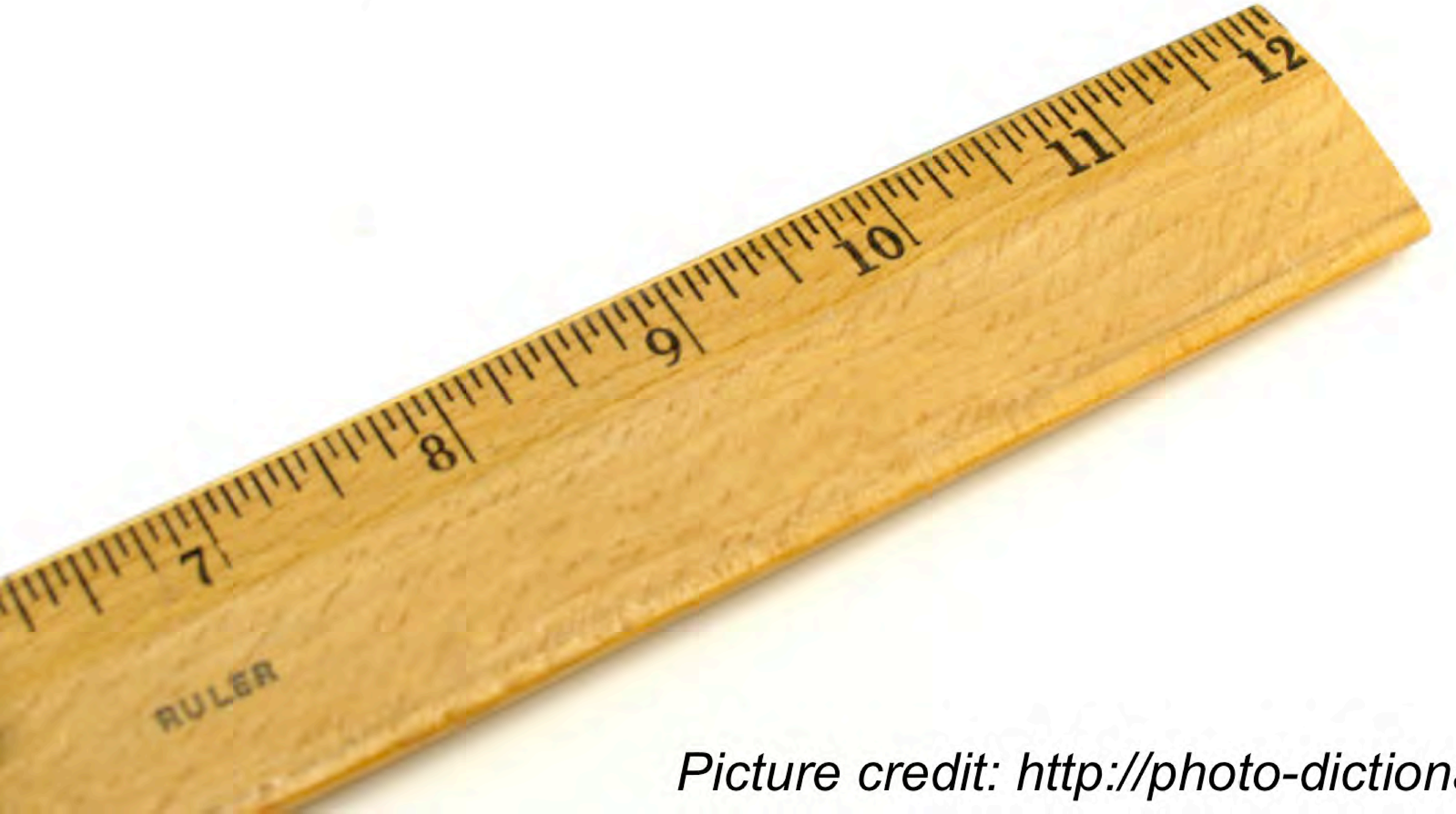
$$\mathbf{v} = \lambda \mathbf{v} M.$$

  Note, you are finding a LOT more information than needed.

- From linear algebra classes, we know how to find right eigenvectors. As such, we simply find eigenvectors of M^T.

Changing rulers

- Remember, if **v** is an eigenvector of M then so is *c***v**.
- As such, a software program has infinitely many choices to return as the dominant eigenvector.



Picture credit: http://photo-dictionary.com/photofiles/list/7918/10687wooden_ruler.jpg

Being square

- For our Twitter network, the dominant eigenvector with length 1 under the 2-norm is:

$$\begin{pmatrix} 0.2332 & 0.3323 & 0.3273 & 0.2332 & 0.7717 & 0.2798 \end{pmatrix}$$

since

$$1 = \sqrt{(0.2332)^2 + (0.3323)^2 + (0.3273)^2 + (0.2332)^2 + (0.7717)^2 + (0.2798)^2}$$

- We want a vector where the sum of the entries is 1.
- Can you think how to do this?

To be 1

- For any vector v, the following will be a parallel vector with entries that sum to 1.

$$\left(\frac{1}{(\sum_{i=1}^{n} v_i)}\right)\mathbf{v}$$

- Now,

$$\begin{aligned} 2.1775 &= 0.2332 + 0.3323 + 0.3273 + \\ &\quad 0.2332 + 0.7717 + 0.2798 \end{aligned}$$

PageRank

Therefore, the vector we want for the Twitter network is:

$$\left(\frac{1}{2.1775}\right)\begin{pmatrix}0.2332 & 0.3323 & 0.3273 & 0.2332 & 0.7717 & 0.2798\end{pmatrix}$$

$$\mathbf{v} = \begin{pmatrix}0.1071 & 0.1526 & 0.1503 & 0.1071 & 0.3544 & 0.1285\end{pmatrix}$$

Further exploration

- Want to dive further into this topic?
- Here are a few ideas…

Picture credit: http://kevinklau.com/wp-content/uploads/2011/10/explore3.jpg

Teleportation

- Earlier, we took the teleportation parameter to equal 0.85.
- Change this value so it is closer to 1. Then,
- change it so it is closer to 0.
- What impact does this have on convergence? What impact does it have on the ranking?

Picture credit: http://www.blavish.com/wp-content/uploads/2006/10/star-trek-teleport-10-5-2006.jpg

HITS

- HITS is an alternative algorithm for ranking.
- Implement this algorithm, that also uses linear algebra.
- How do the results vary from PageRank?

Picture credit: http://www.photoshopessentials.com/images/photo-effects/punch-through/photoshop-punch-through-image.jpg

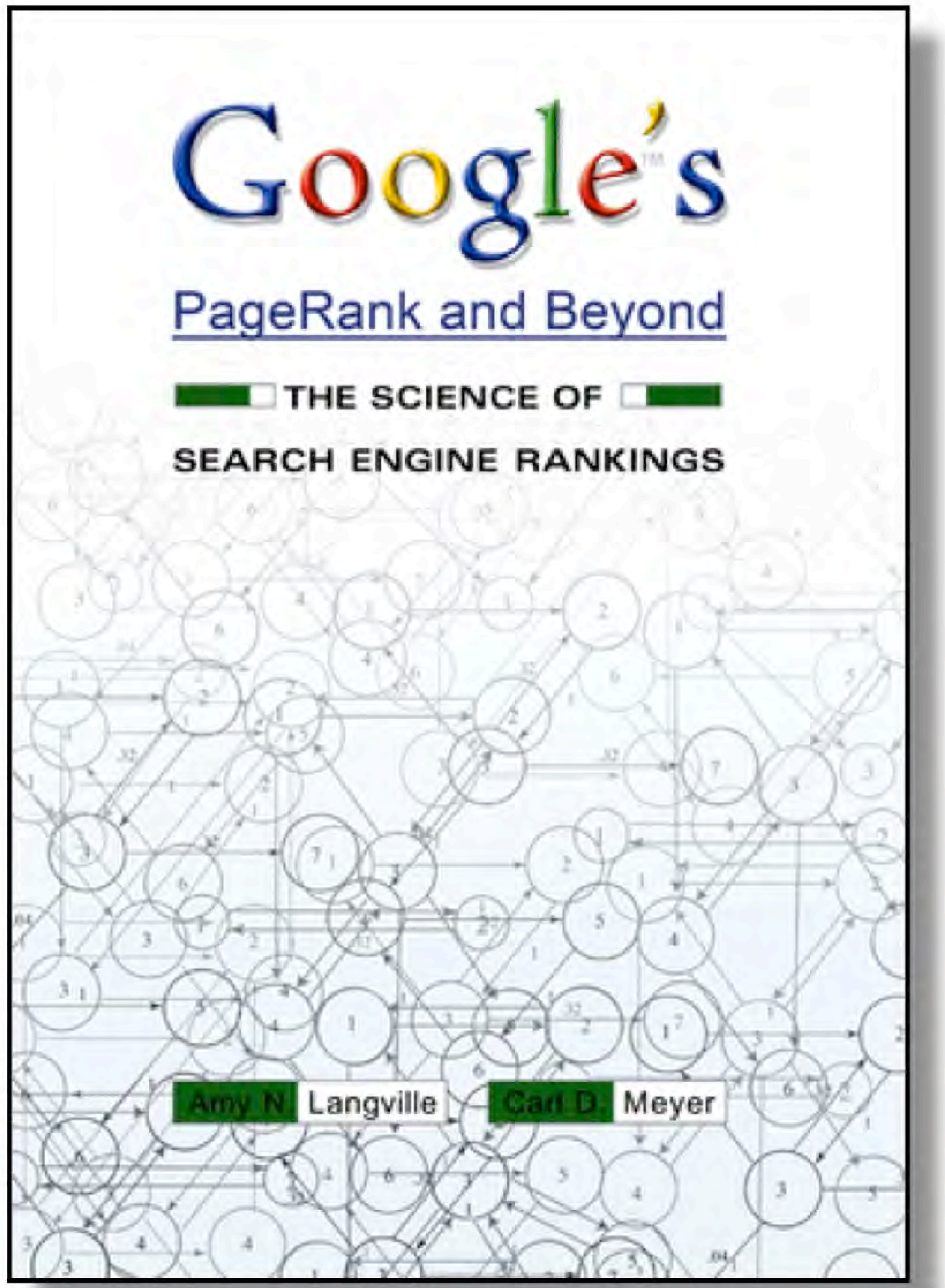
Application

- Applying PageRank to other networks from fields such as biology or archeology.
- Do you have a directed graph? Could PageRank be applied?
- You could even try sports ranking!

Picture credit: http://www.inspirecusd.org/misc_images/000/000/000/170/ApplyNowButton.jpg

Scalability

- How does PageRank scale up to billions of pages?
- A key is expressing the Google matrix in a different form so you only store the (sparse) adjacency matrix and a vector.
- Else, you store an *n* x *n* dense matrix.



Just for fun...

To motivate the random surfer outlook on PageRank, see the video at:

http://vimeo.com/11548769

A mysterious package

In the video, Emmie receives a mysterious package with Google goggles.

Virtual world

- She enters Google-topia and meets Randy the random surfer.
- They surf that world's web and discover the ideas of Brin and Page, founders of Google.