NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

CE/CZ4052 Cloud Computing

Pagerank Algorithm

Dr. Tan, Chee Wei

Email: cheewei.tan@ntu.edu.sg

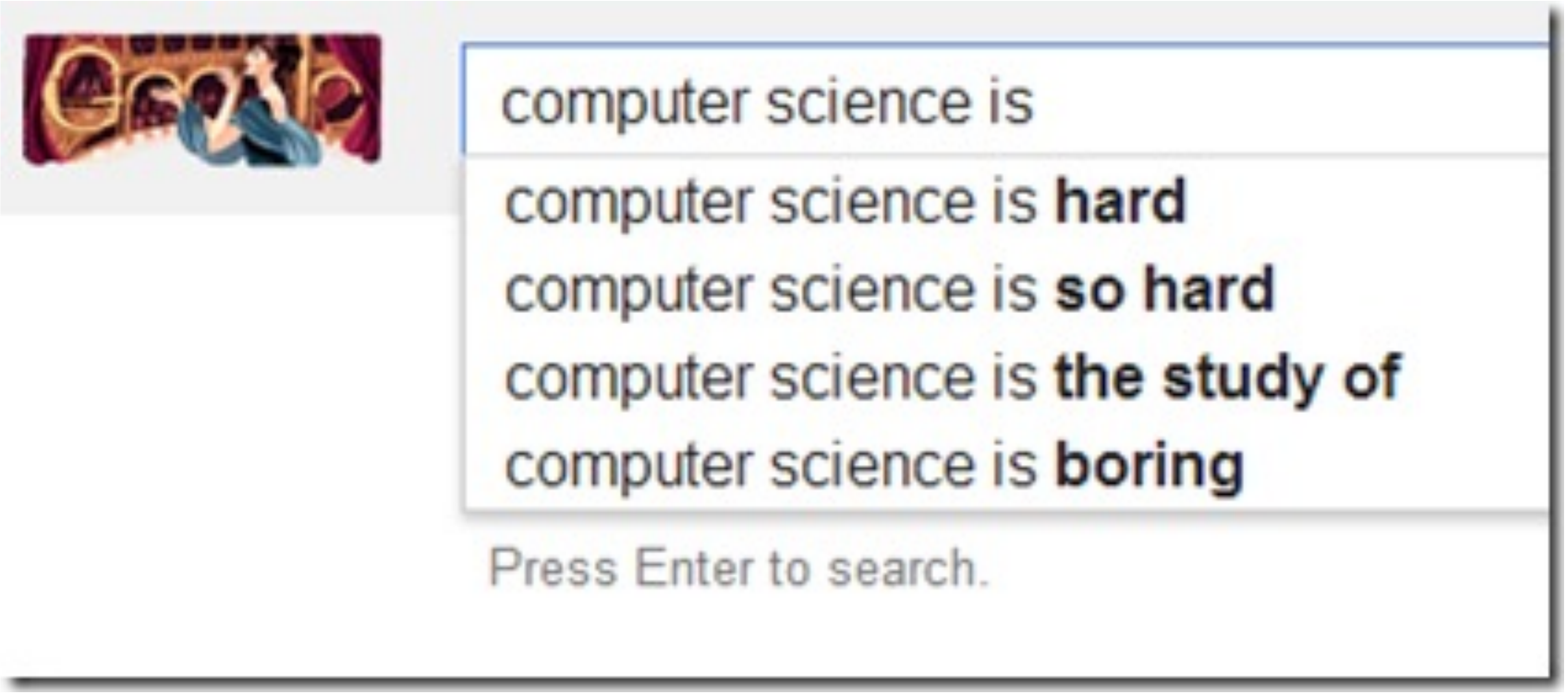
Office: N4-02c-104

How does Google rank the Web?

Acknowledgement:
https://www.cazencott.info/dotclear/public/lectures/lsml19/2019-03-28_lsml19_systems.pdf

Search Engine Technologies

- **Computer Science is …**



Search Engine Technologies

Definition of **google** in English

google

Pronunciation: /ˈɡuːɡl/

Translate **google** | into French | into Italian | into Spanish

verb

[*with object*]

search for information about (someone or something) on the Internet using the search engine Google:

on Sunday she googled an ex-boyfriend

[*no object*]:

*I **googled for** a cheap hotel/flight deal*

Derivatives

googleable

(also **googlable**) *adjective*

The History of PageRank

- PageRank was developed by Larry Page (hence the name *Page*-Rank) and Sergey Brin at Stanford in 1999.
- Shortly after, Page and Brin founded Google.
- Challenges: Web contains many sources of information – including spam.
  - What is the best answer to a web query “google” in 1998?
  - A good web search algorithm enables trust
- Use links as votes to rank pages
- Are all links equally important?
  - Links from important pages count more.
  - This question is recursive.

The PageRank in Search Engines (1997)

Location: http://walrus.stanford.edu/diglib/pub/people.html

What's New? | What's Cool? | Destinations | Net Search | People | Software

Stanford University Participants

The Database group

Prof. Hector Garcia-Molina , Misturu Akizawa (Visiting Scholar from Hitachi) , Edward Chang , Chen-Chuan K. Chang , Arturo Crespo , Luis Gravano , Matt Jacobsen , Steven Ketchpel , Yusuke Mishina (Visiting Scholar from Hitachi) , Narayanan Shivakumar

The Project on People Computers and Design

Prof. Terry Winograd , Michelle Q Wang Baldonado , Steve Cousins , Mauria Finley , Frankie James , Larry Page , Christian P. Rohrer , Martin Röscheisen , Alan Steremberg , Trace Wax

The Nobots group

Prof. Daphne Koller , Prof. Yoav Shoham , Marko Balabanovic , Avi Pfeffer , Mehran Sahami , Katsumi Tanaka (Visiting Scholar)

The Testbed group

Scott Hassan , Andy Kacsman , Andreas Paepcke , Tom Schirmer

Stanford Libraries and Academic Information Resources

Rebecca Lasher , Vicky Reich

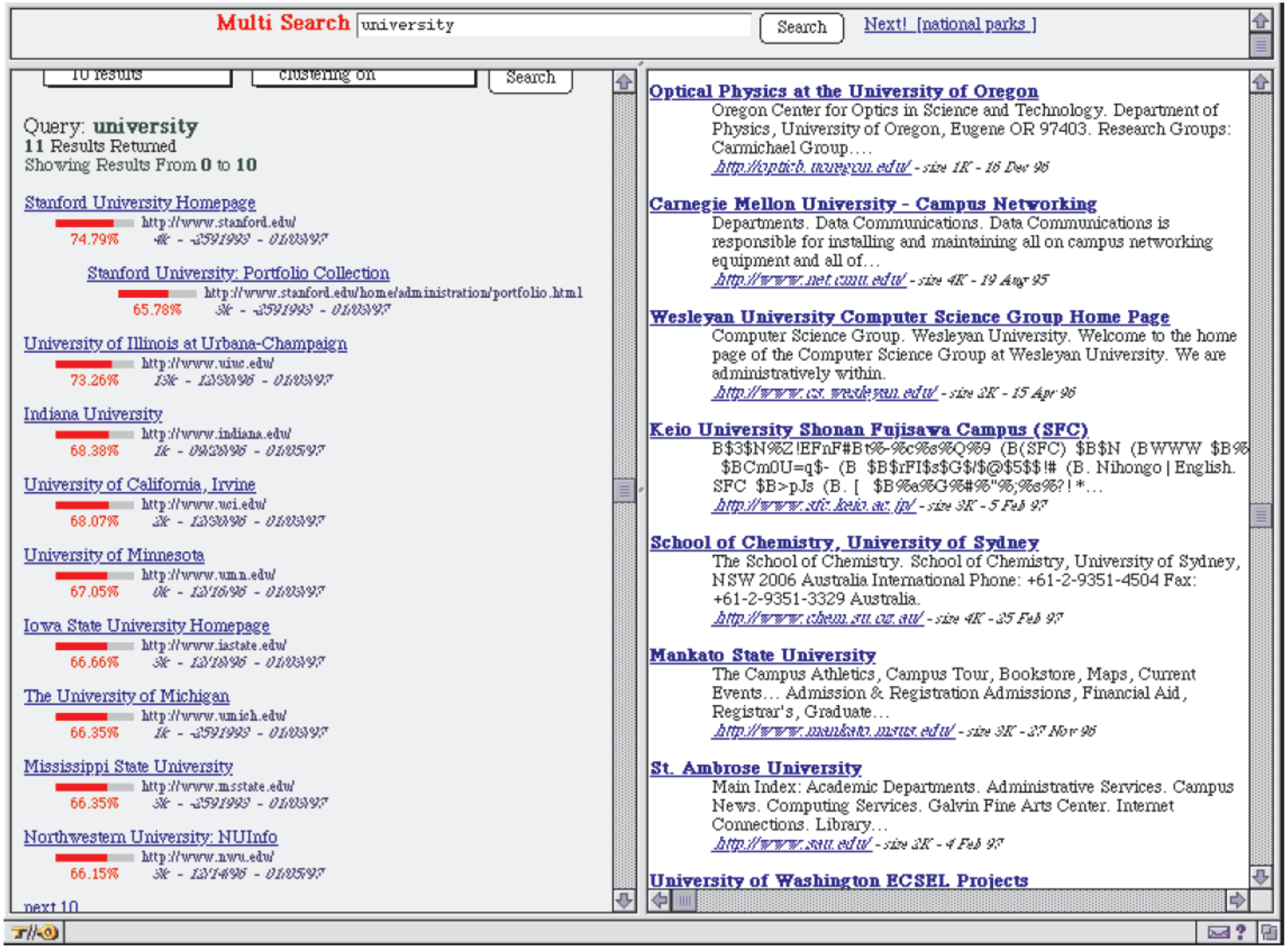
Engineering-Economic Systems

Tim Stanley

Alumni from the Stanford Digital Libraries Project

Perry Arnold , Kenichi Kamiya , James Kittock , Christian Mogensen , Tak Yan

Searching with PageRank (1997)

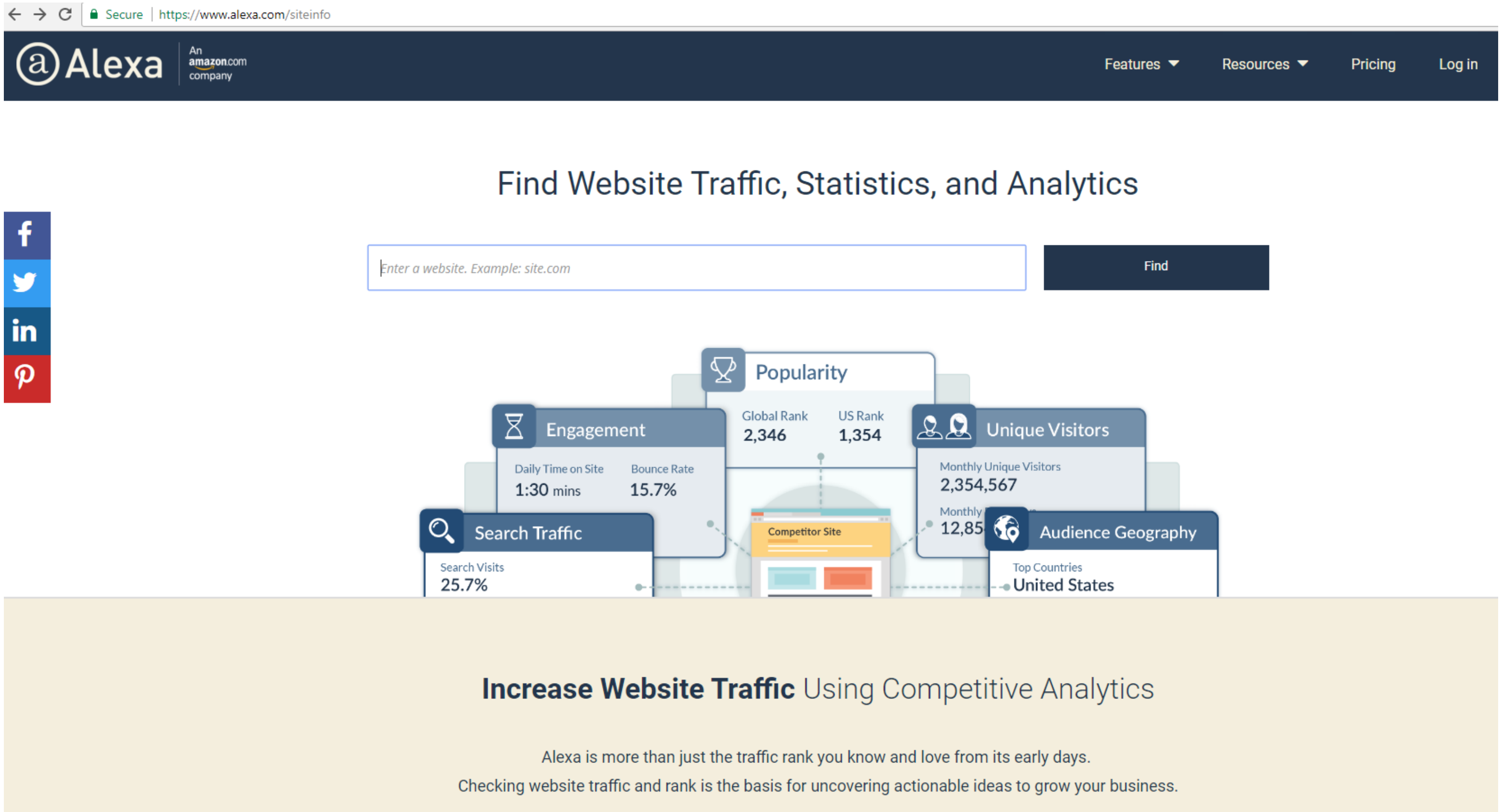


Searching with PageRank (1997)

Web Page	PageRank (average is 1.0)
Download Netscape Software	11589.00
http://www.w3.org/	10717.70
Welcome to Netscape	8673.51
Point: It's What You're Searching For	7930.92
Web-Counter Home Page	7254.97
The Blue Ribbon Campaign for Online Free Speech	7010.39
CERN Welcome	6562.49
Yahoo!	6561.80
Welcome to Netscape	6203.47
Wusage 4.1: A Usage Statistics System For Web Servers	5963.27
The World Wide Web Consortium (W3C)	5672.21
Lycos, Inc. Home Page	4683.31
Starting Point	4501.98
Welcome to Magellan!	3866.82
Oracle Corporation	3587.63

Top 15 Page Ranks: July 1996

The PageRank in Search Engines (2017)

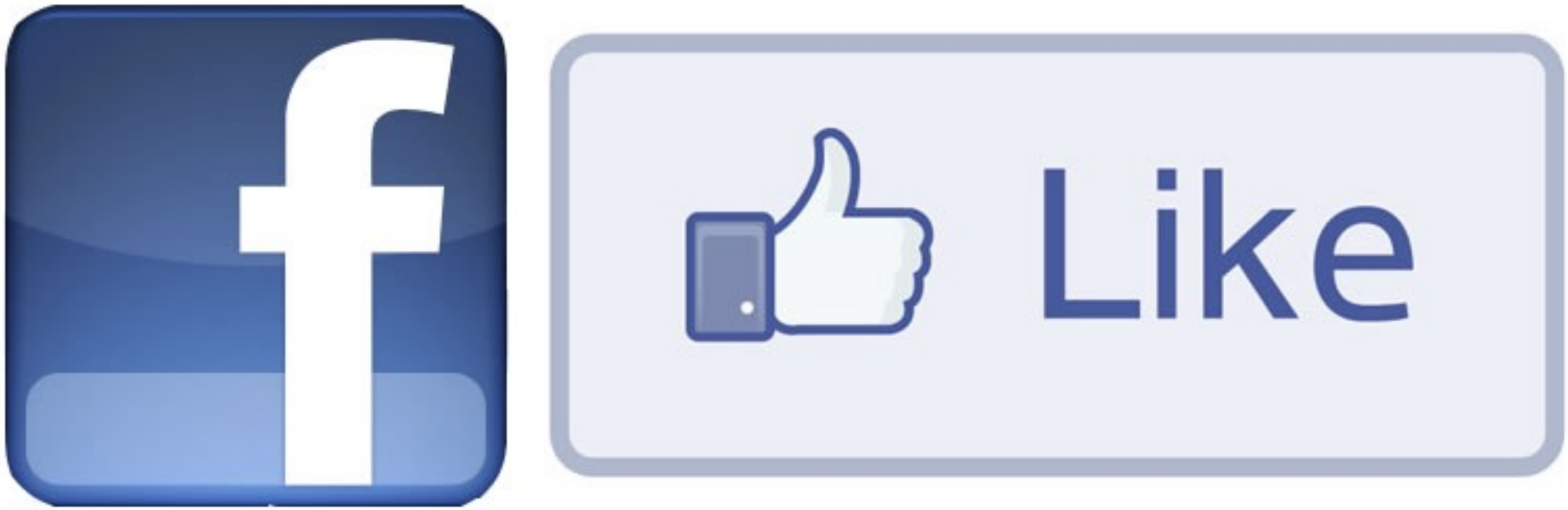


https://www.alexa.com/siteinfo https://moonsy.com/alexa_rank/

Guess, who has top ranking, i.e., number 1?

Link Analysis

- Consider links as votes of confidence in a page
- A hyperlink is the open Web’s version of …

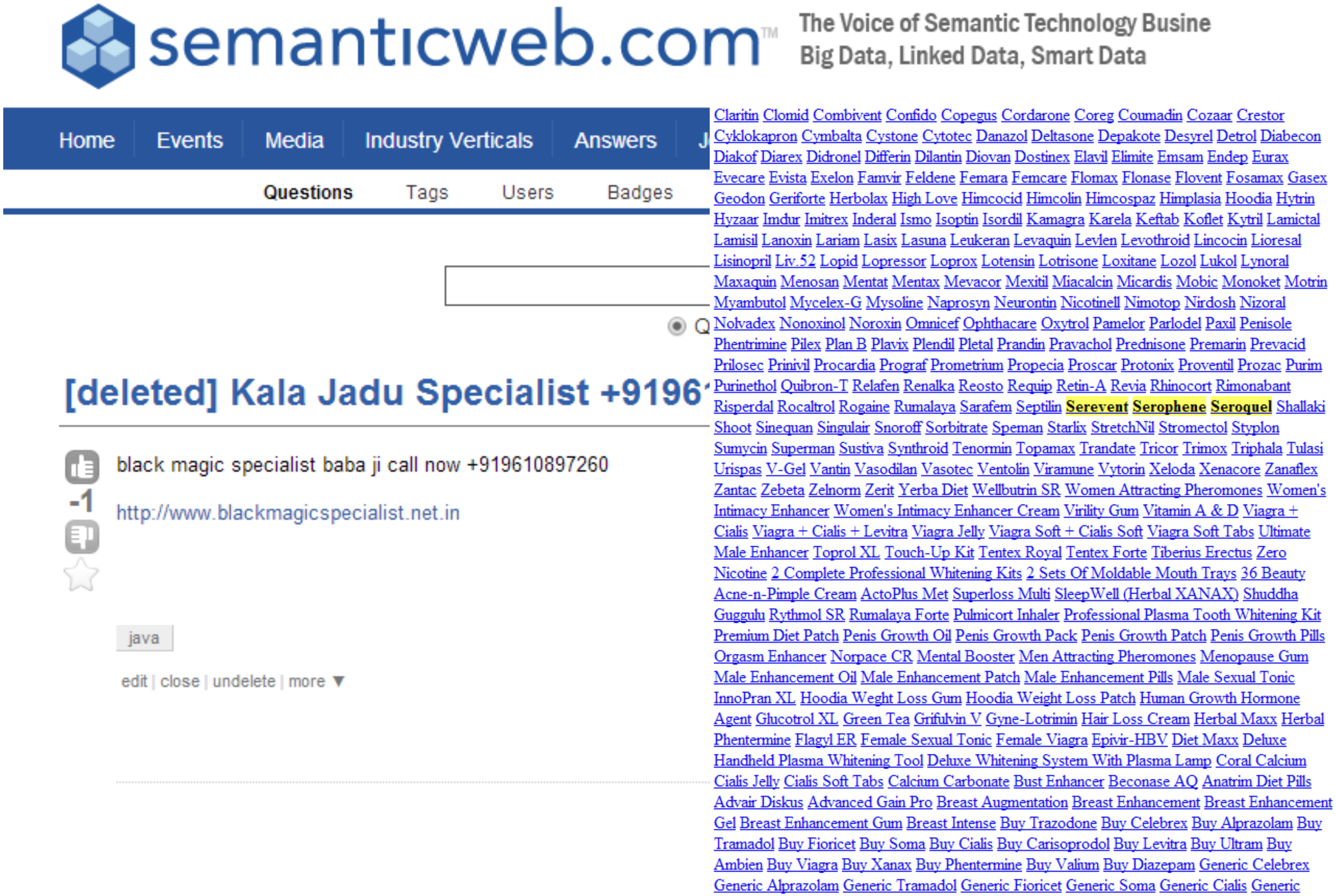


(… even if the page is linked in a negative way.)

Link Analysis

So if we just count the number of inlinks a web-page receives we know its importance, right?

Link Spamming



PageRank

PageRank

- Not just a count of inlinks
  - A link from a more important page is more important
  - A link from a page with fewer links is more important

  ∴ A page with lots of inlinks from important pages (which have few outlinks) is more important

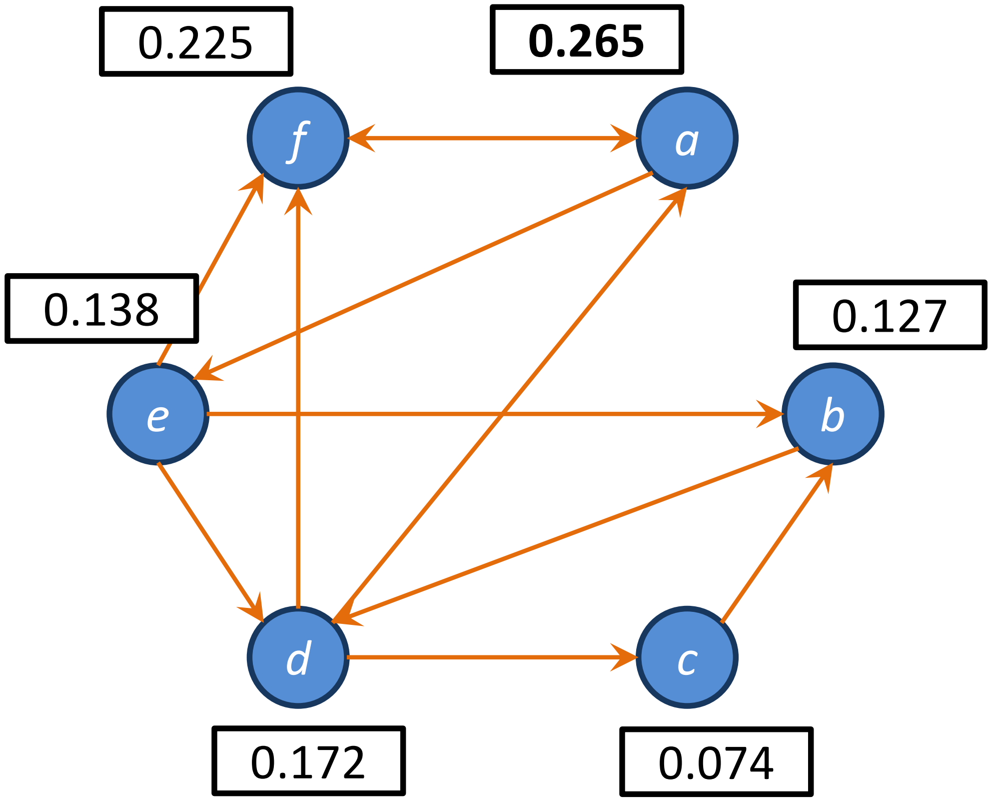
PageRank Model

- The Web: a directed graph

$$G = (\boxed{V}, \boxed{E})$$

Vertices (*pages*)

Edges (*links*)



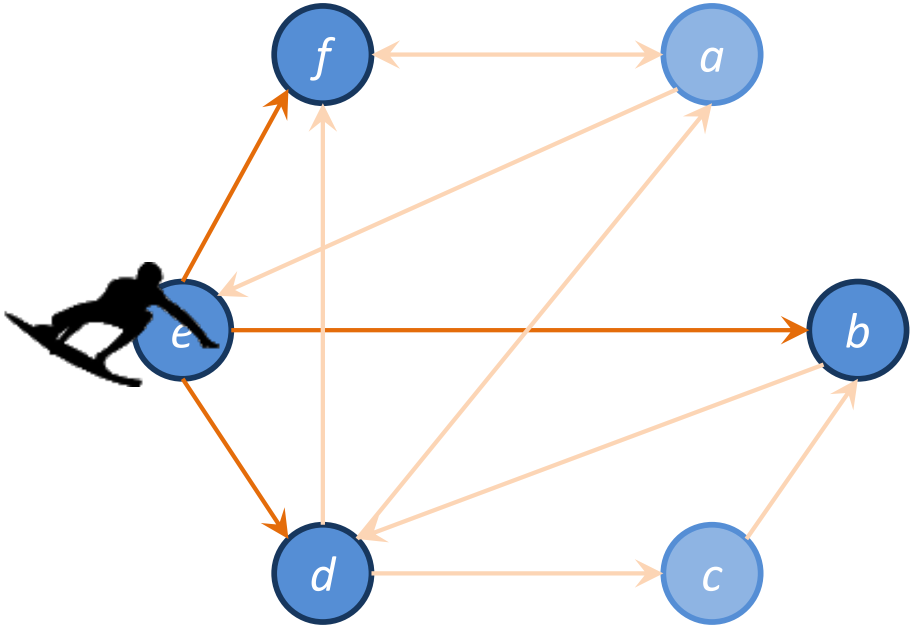
Which is the most "important" vertex?

$$V = \{a, b, c, d, e, f\}$$

$$E = \{(a, e), (a, f), (b, d), (c, b), (d, a), (d, c), (d, f), (e, b), (e, d), (e, f), (f, a)\}$$

PageRank: Random Surfer Model

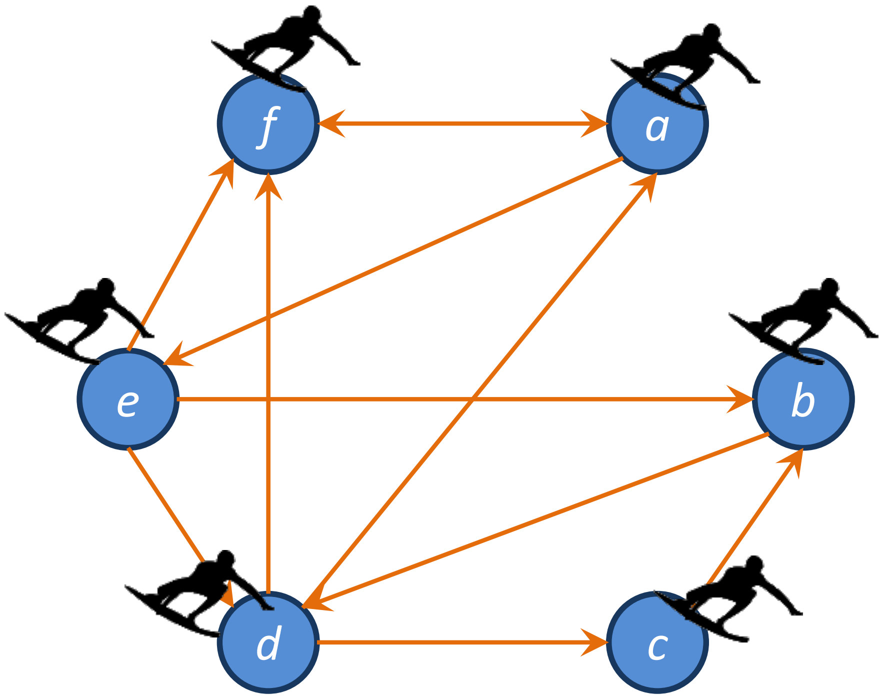
= someone surfing the web, clicking links randomly



- What is the probability of being at page *x* after *n* hops?

PageRank: Random Surfer Model

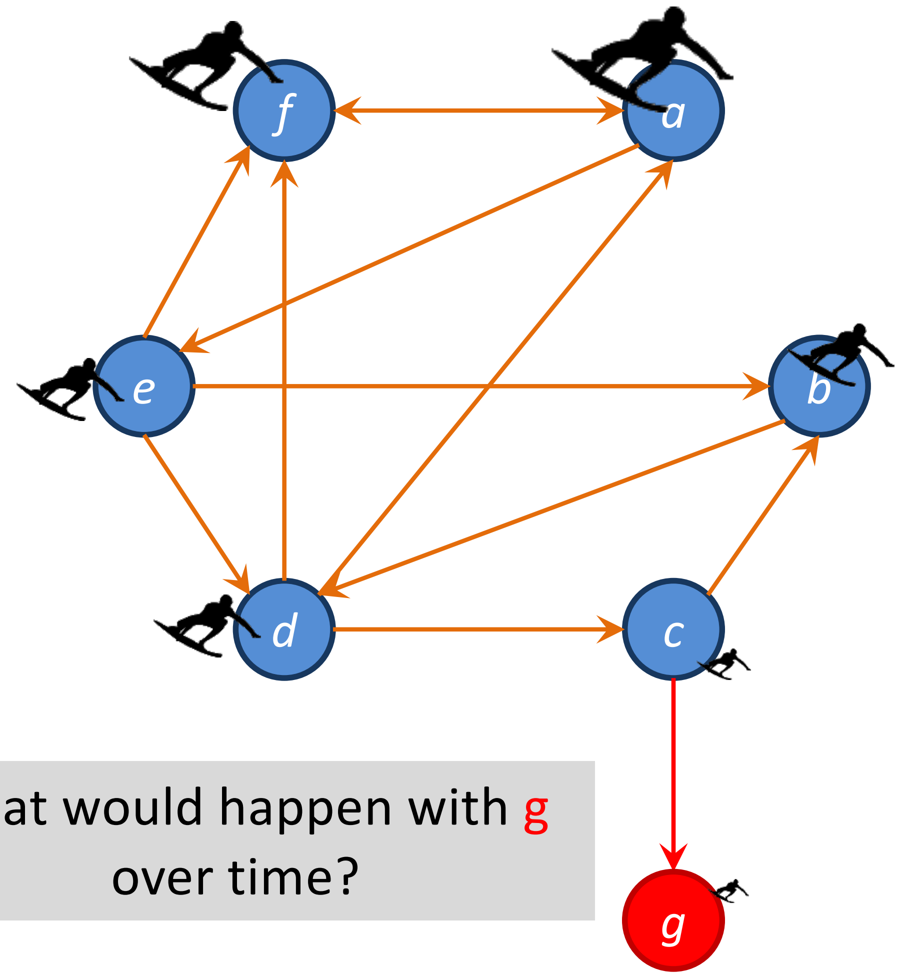
= someone surfing the web, clicking links randomly



- What is the probability of being at page *x* after *n* hops?
- *Initial state:* surfer equally likely to start at any node

PageRank: Random Surfer Model

= someone surfing the web, clicking links randomly

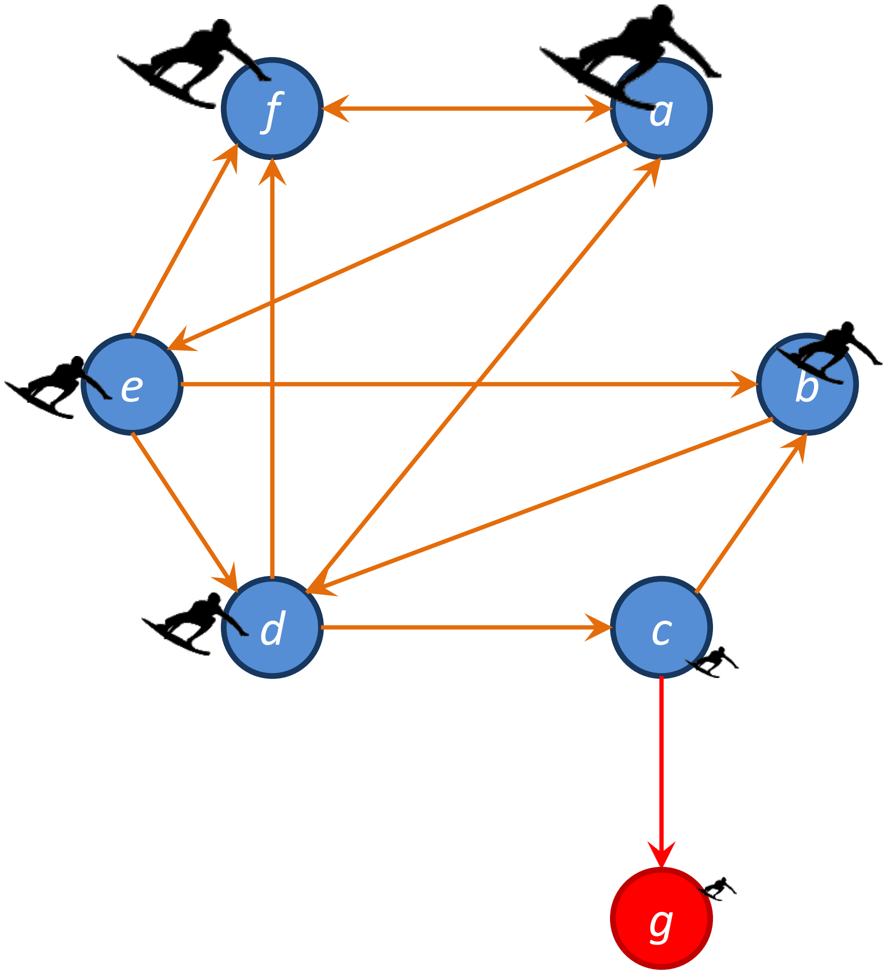


- What is the probability of being at page *x* after *n* hops?
- *Initial state:* surfer equally likely to start at any node
- PageRank applied iteratively for each hop: score indicates probability of being at that page after that many hops

What would happen with g over time?

PageRank: Random Surfer Model

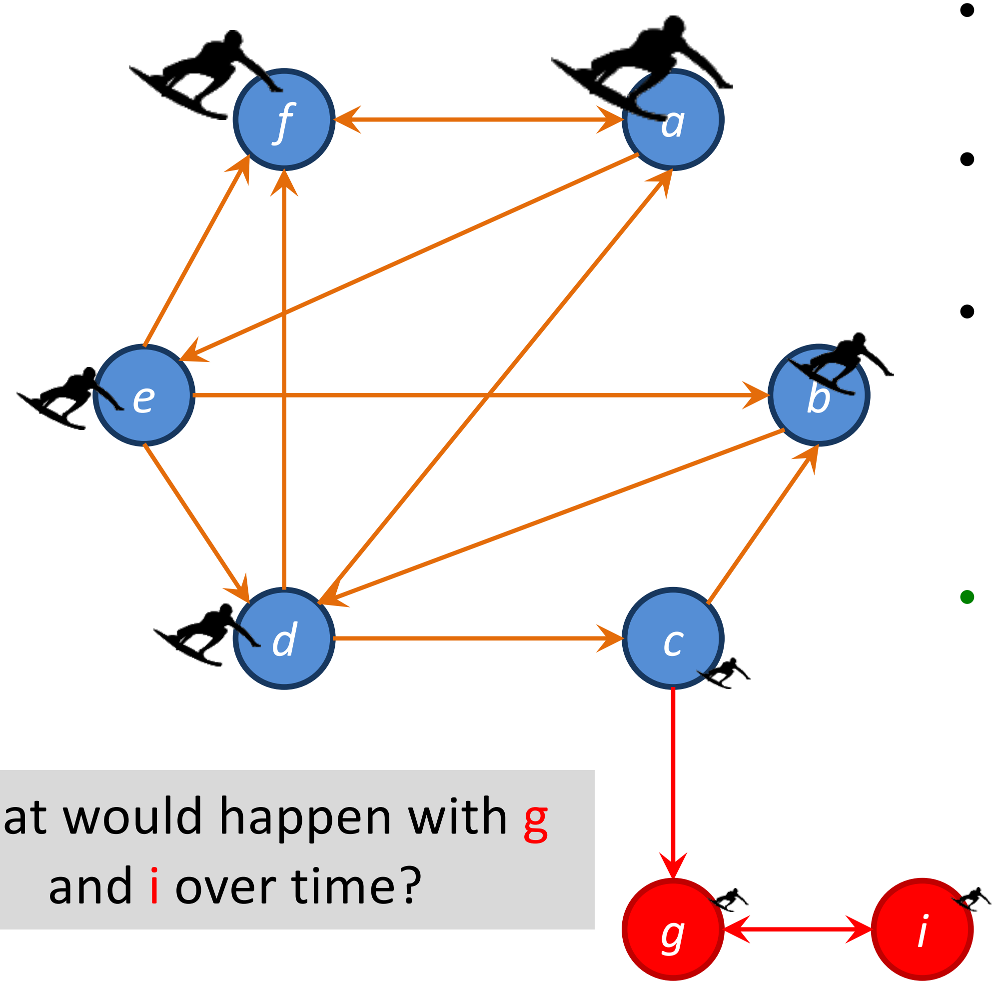
= someone surfing the web, clicking links randomly



- What is the probability of being at page *x* after *n* hops?
- *Initial state:* surfer equally likely to start at any node
- PageRank applied iteratively for each hop: score indicates probability of being at that page after than many hops
- If the surfer reaches a page without links, the surfer randomly jumps to another page

PageRank: Random Surfer Model

= someone surfing the web, clicking links randomly

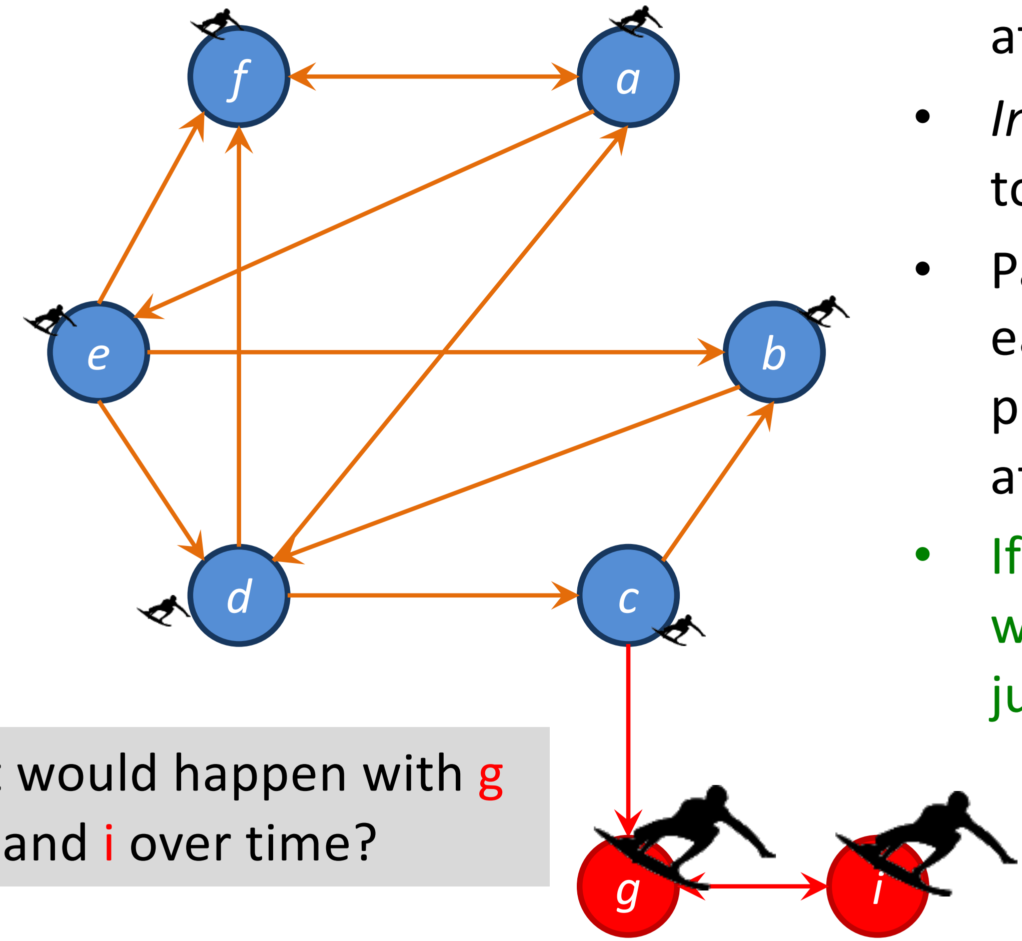


What would happen with g and i over time?

- What is the probability of being at page *x* after *n* hops?
- *Initial state:* surfer equally likely to start at any node
- PageRank applied iteratively for each hop: score indicates probability of being at that page after than many hops
- If the surfer reaches a page without links, the surfer randomly jumps to another page

PageRank: Random Surfer Model

 = someone surfing the web, clicking links randomly

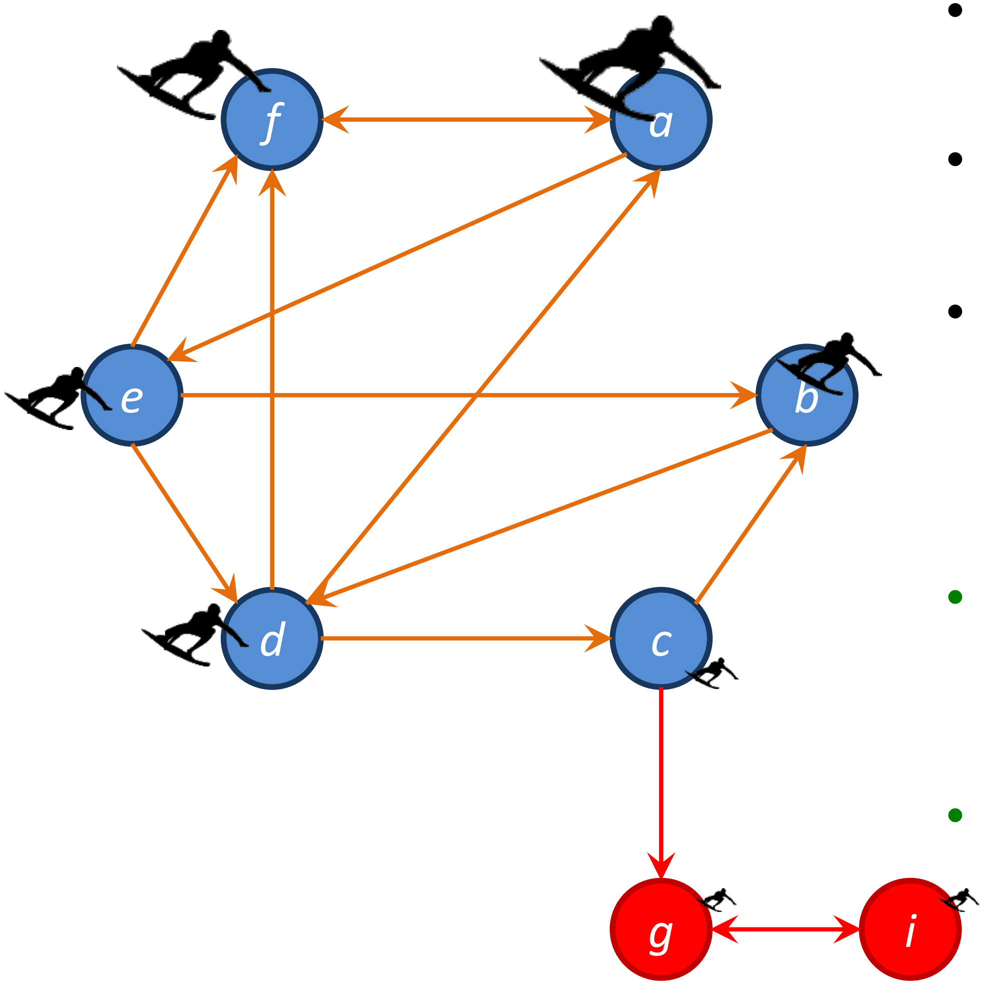


What would happen with g and i over time?

- What is the probability of being at page *x* after *n* hops?
- *Initial state:* surfer equally likely to start at any node
- PageRank applied iteratively for each hop: score indicates probability of being at that page after than many hops
- If the surfer reaches a page without links, the surfer randomly jumps to another page

PageRank: Random Surfer Model

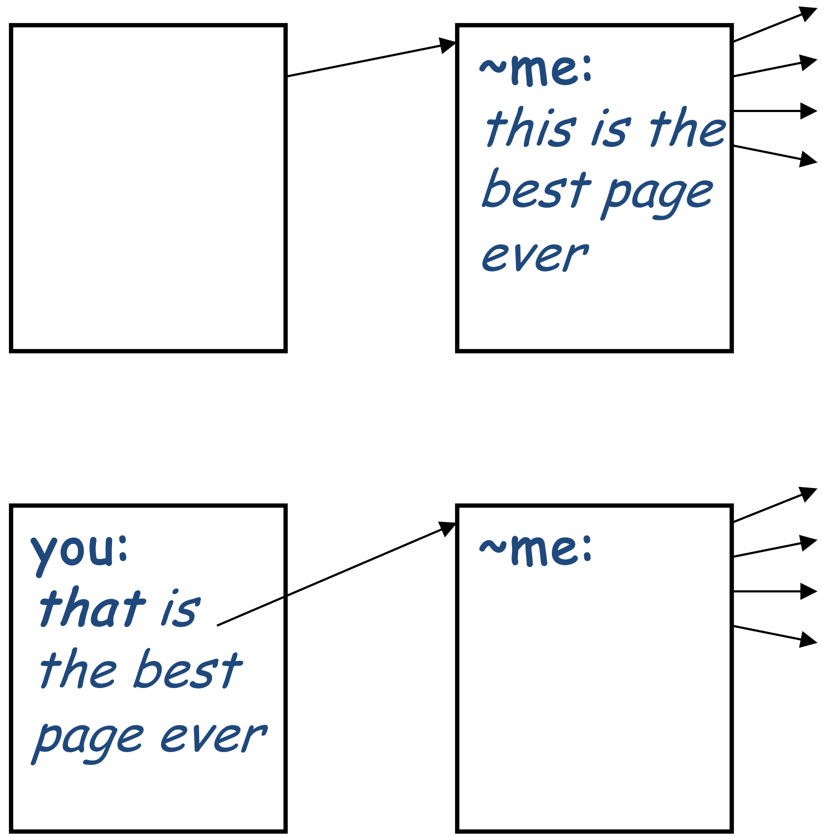
= someone surfing the web, clicking links randomly



- What is the probability of being at page *x* after *n* hops?
- *Initial state:* surfer equally likely to start at any node
- PageRank applied iteratively for each hop: score indicates probability of being at that page after than many hops
- If the surfer reaches a page without links, the surfer randomly jumps to another page
- The surfer will jump to a random page at any time with a probability 1 – *d* … *this avoids traps and ensures convergence!*

Google search: anchor text

- Pagerank
- Anchor text



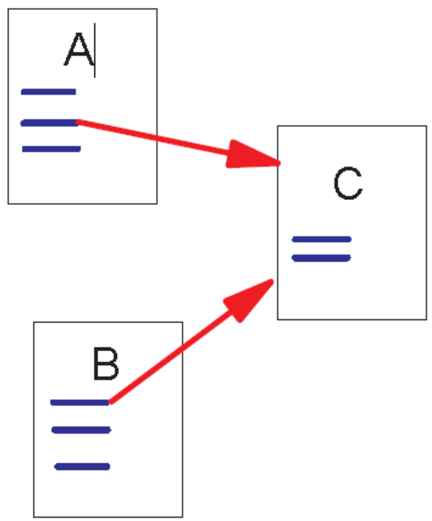
Google uses:

- In anchor text?
- In URL?
- Title
- Meta tags
- <h> level
- Rel font size
- Capitalization
- Word pos in doc
- Secret ingredients

… and weighs them according to a secret recipe

Link Structure of the Web

- 150 million web pages → 1.7 billion links



inlinks outlinks

Backlinks and Forward links:

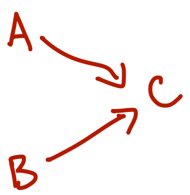
➢A and B are C's backlinks

➢C is A and B's forward link

Intuitively, a webpage is important if it has a lot of backlinks.

A Simple Version of PageRank

eg.



$$R(u) = \hat{c} \sum_{v \in B_u} \frac{R(v)}{N_v}$$

c

A, B

The influence given by a web page

eg. if $R(A) = 10$ and has 4 forward links, $\frac{R(A)}{N} = 2.5$

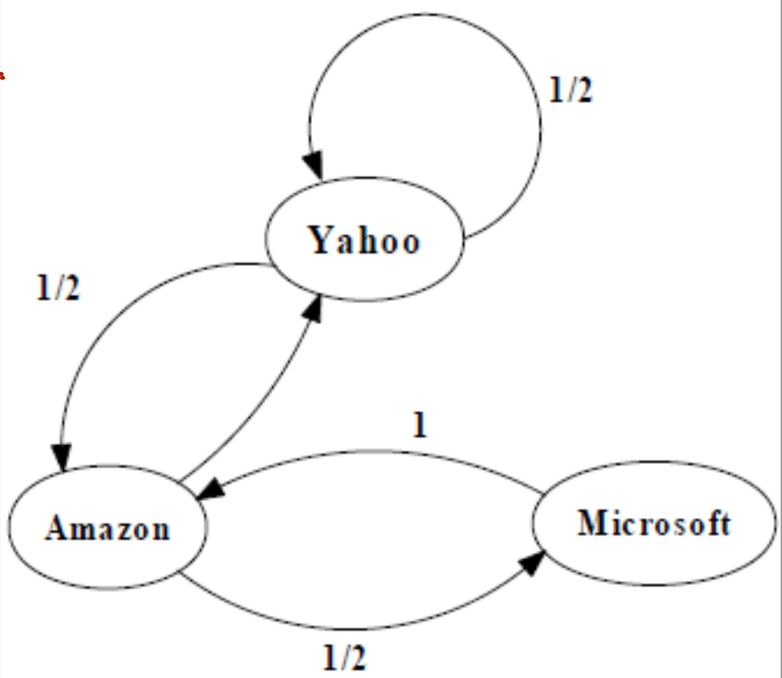
Then sum it all up

- u: a web page
- B_u: the set of u's backlinks
- N_v: the number of forward links of page v
- $\hat{c}$: the normalization factor to make R(1) + … + R(T) = 1 where there are T pages in total

Total value of all ranks = 1.

An example of Simplified PageRank

$\lim_{k\to\infty} R^{(k)} = R^*$



run this until converge.

$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

params for R(A)

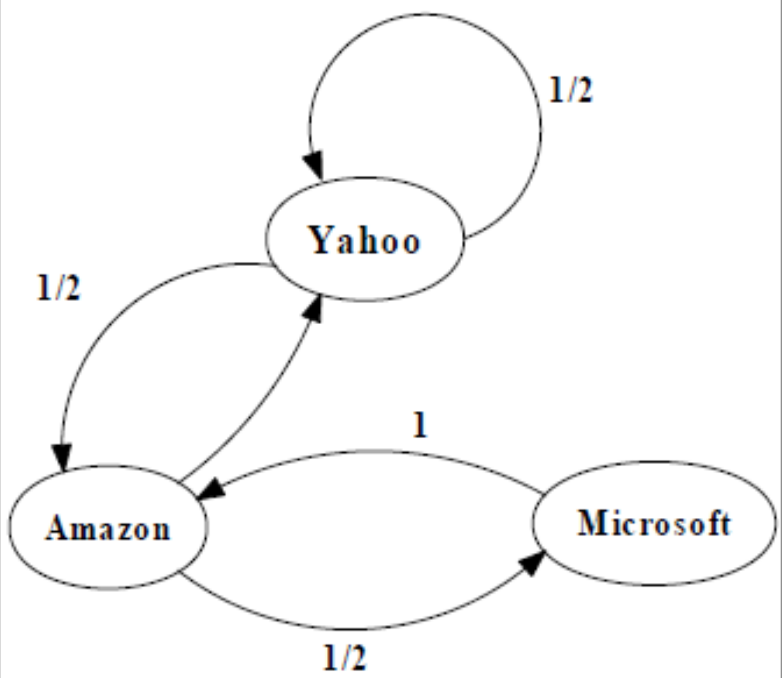
$R^{(k+1)} = M R^{(k)}$

$\underline{1}^T R = 1$

$$R = \begin{array}{c} R(A) \\ R(B) \\ R(C) \end{array} \begin{bmatrix} 1/3 \\ 1/2 \\ 1/6 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

PageRank Calculation: first iteration

An example of Simplified PageRank



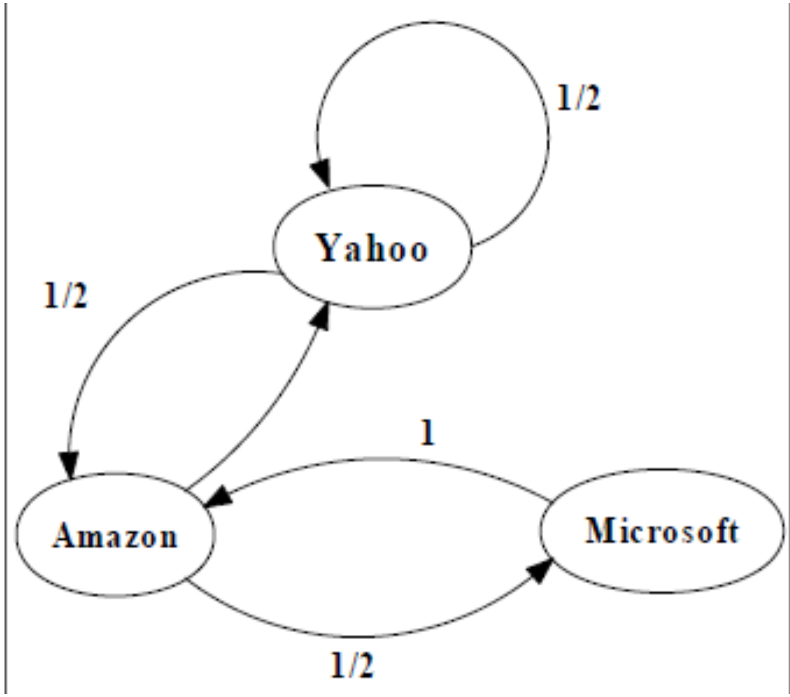
$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 5/12 \\ 1/3 \\ 1/4 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/2 \\ 1/6 \end{bmatrix}$$

PageRank Calculation: second iteration

An example of Simplified PageRank



$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix}$$

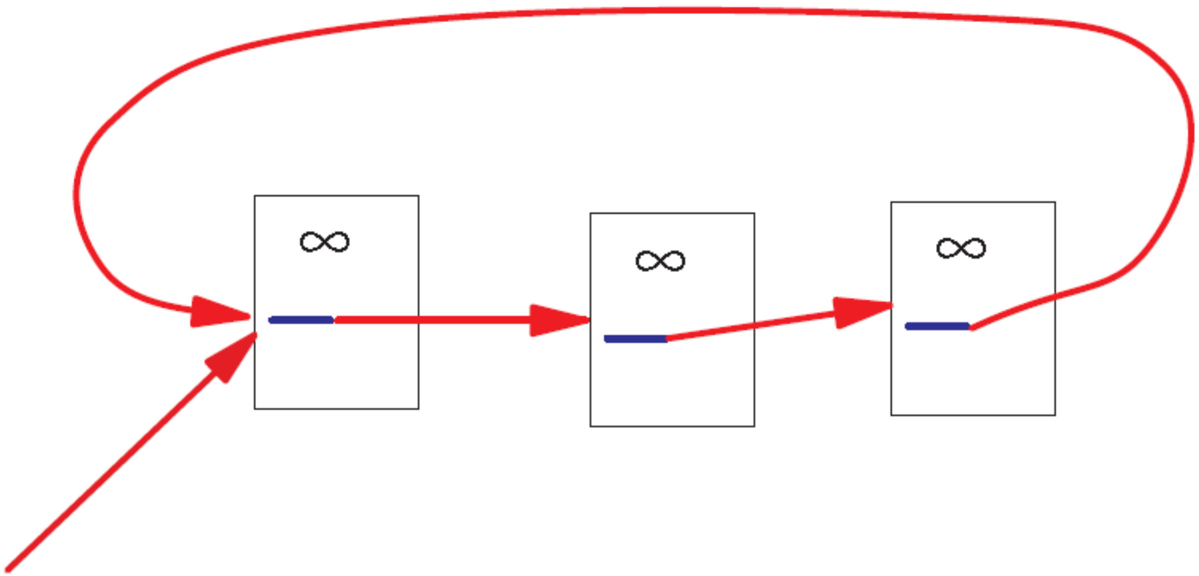
$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 3/8 \\ 11/24 \\ 1/6 \end{bmatrix} \begin{bmatrix} 5/12 \\ 17/48 \\ 11/48 \end{bmatrix} \dots \begin{bmatrix} 2/5 \\ 2/5 \\ 1/5 \end{bmatrix}$$

Convergence after some iterations

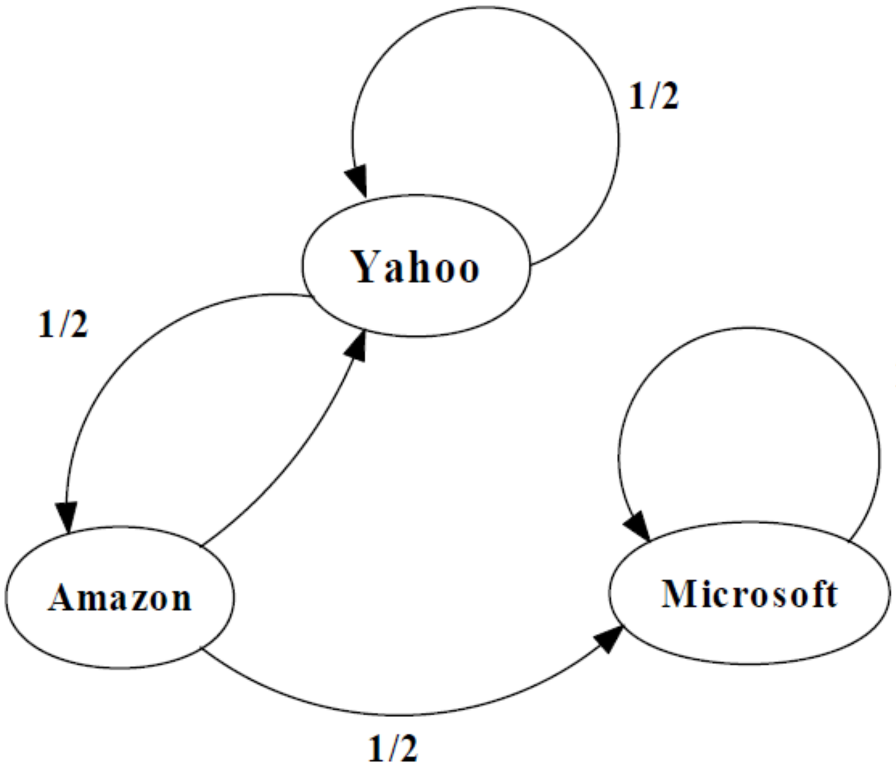
A Problem with Simplified PageRank

A loop:



During each iteration, the loop accumulates rank but never distributes rank to other pages!

An example of the Problem

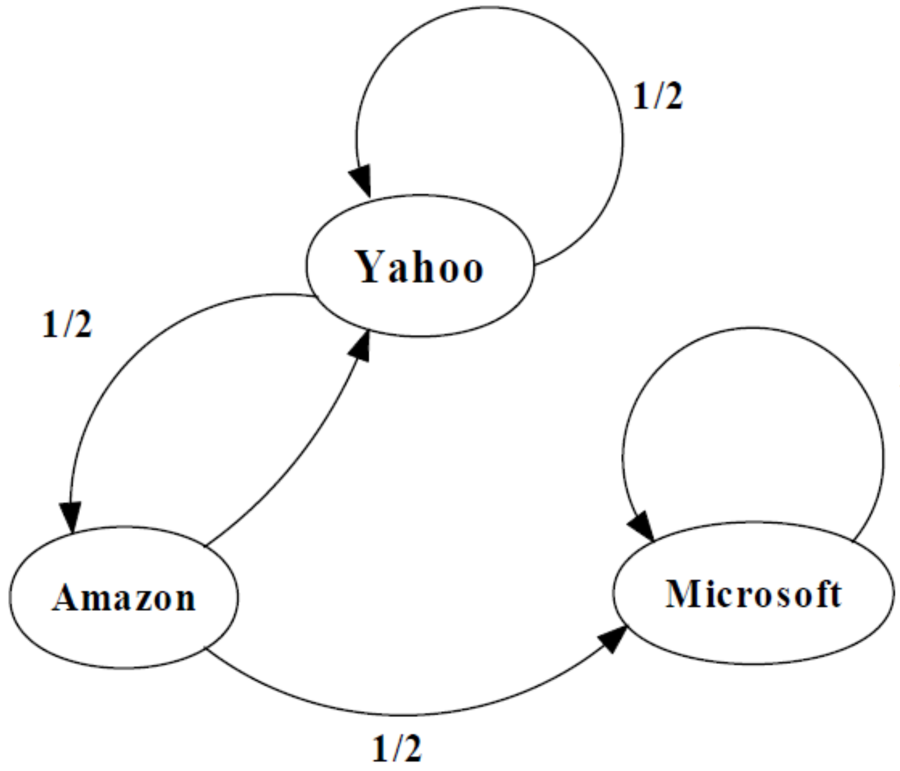


$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 1/3 \\ 1/6 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

An example of the Problem

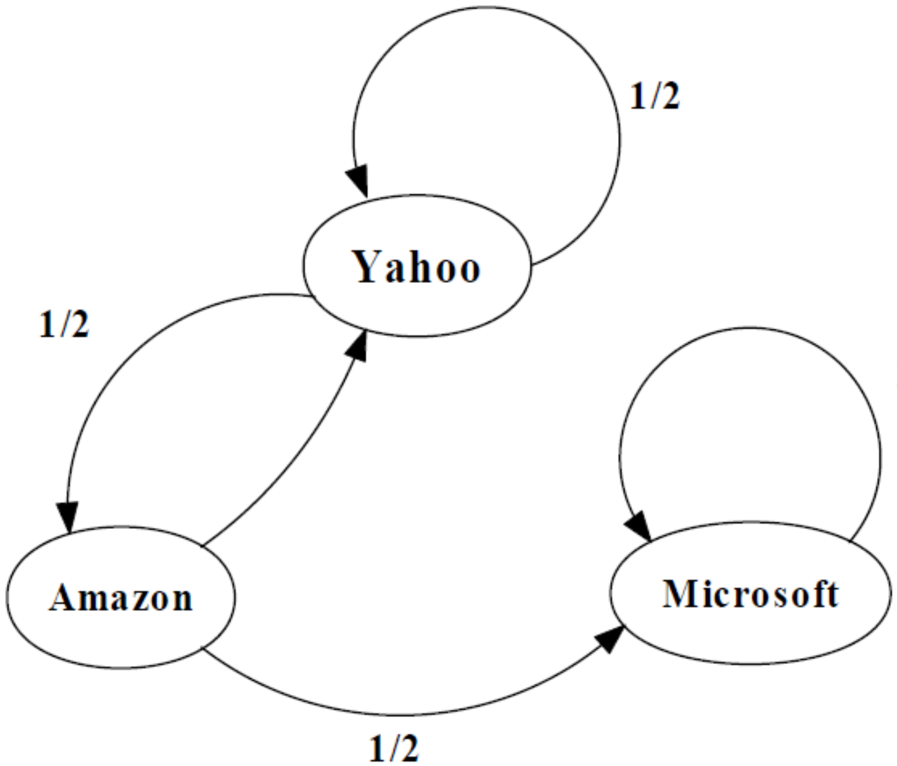


$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 1/4 \\ 1/6 \\ 7/12 \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/6 \\ 1/2 \end{bmatrix}$$

An example of the Problem



$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\begin{bmatrix} 5/24 \\ 1/8 \\ 2/3 \end{bmatrix} \begin{bmatrix} 1/6 \\ 5/48 \\ 35/48 \end{bmatrix} \cdots\cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Random Walks in Graphs

- The Random Surfer Model
  - The simplified model: the standing probability distribution of a random walk on the graph of the web. simply keeps clicking successive links at random
- The Modified Model
  - The modified model: the "random surfer" simply keeps clicking successive links at random, but periodically "gets bored" and jumps to a random page based on the distribution of E

Modified Version of PageRank

$$R' = c_1 M R' + \underbrace{(1-c_1)E}_{b} \;,\; R'^{T}\mathbb{1} = 1$$

$$R'(u) = \underbrace{c_1 \sum_{v \in B_u} \frac{R'(v)}{N_v}}_{\text{surf}} + \underbrace{\overset{1-c_1}{\cancel{c_2}}\, E(u)}_{\text{jump}}$$

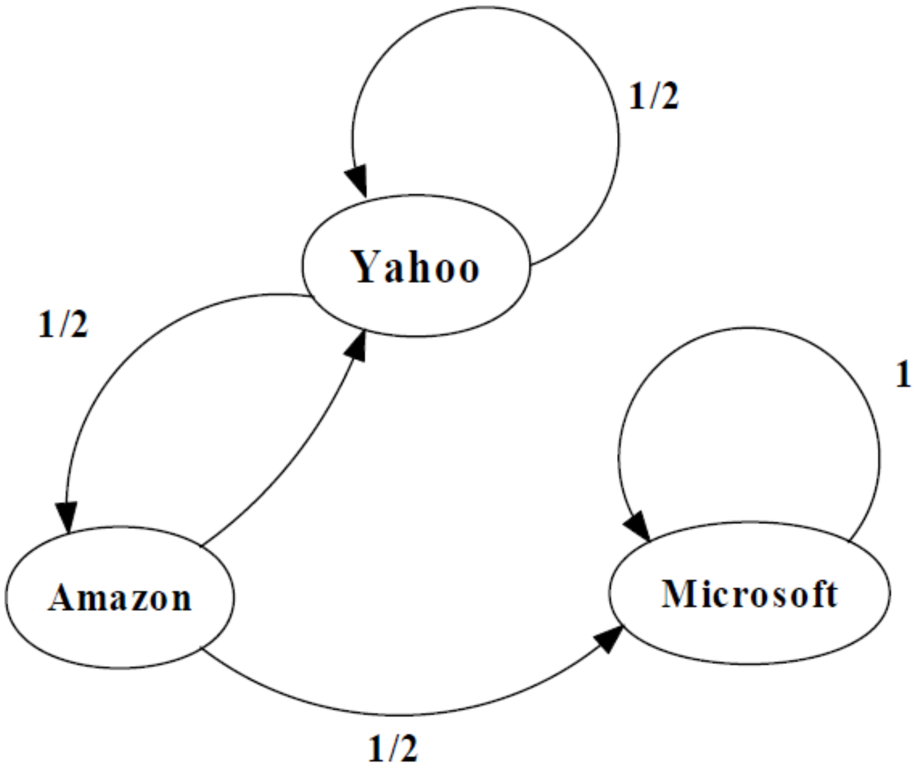
$$(I - c_1 M) R' = b$$

E(u): a distribution of ranks of web pages that "users" jump to when they "gets bored" after successive links at random.

$$R' = (I - c_1 M)^{-1} b$$

An example of Modified PageRank

$E = \begin{bmatrix} \frac{1}{r} \\ \frac{1}{r} \\ \frac{1}{r} \end{bmatrix}$



$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \text{yahoo} \\ \text{Amazon} \\ \text{Microsoft} \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$C_1 = 0.8 \qquad C_2 = 0.2$

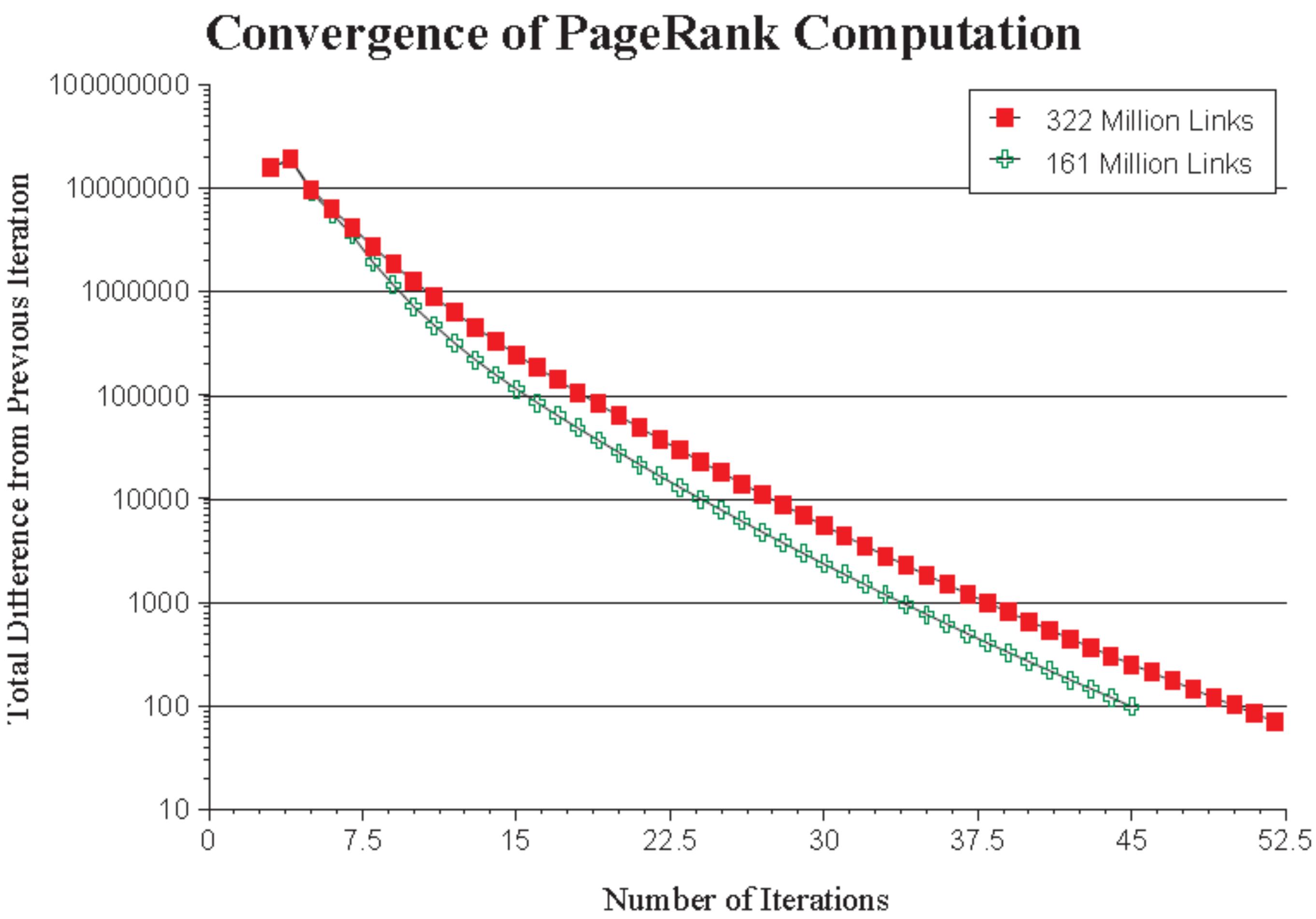
$$\begin{bmatrix} 0.333 \\ 0.333 \\ 0.333 \end{bmatrix} \begin{bmatrix} 0.333 \\ 0.200 \\ 0.467 \end{bmatrix} \begin{bmatrix} 0.280 \\ 0.200 \\ 0.520 \end{bmatrix} \begin{bmatrix} 0.259 \\ 0.179 \\ 0.563 \end{bmatrix} \ldots \begin{bmatrix} 7/33 \\ 5/33 \\ 21/33 \end{bmatrix}$$

Dangling Links

- Links that point to any page with no outgoing links
- Most are pages that have not been downloaded yet
- Affect the model since it is not clear where their weight should be distributed
- Do not affect the ranking of any other page directly
- Can be simply removed before pagerank calculation and added back afterwards

Convergence Property

- PR (322 Million Links): 52 iterations
- PR (161 Million Links): 45 iterations
- Scaling factor is roughly linear in *logn*



Convergence Property

won't come up in exams.

- The Web is an expander-like graph
  - Theory of random walk: a random walk on a graph is said to be rapidly-mixing if it quickly converges to a limiting distribution on the set of nodes in the graph. A random walk is rapidly-mixing on a graph if and only if the graph is an expander graph.
  - Expander graph: every subset of nodes S has a neighborhood (set of vertices accessible via outedges emanating from nodes in S) that is larger than some factor α times of |S|. A graph has a good expansion factor if and only if the largest eigenvalue is sufficiently larger than the second-largest eigenvalue.

PageRank vs. Web Traffic

- Important component of PageRank calculation is *E*
  - A vector over the web pages (used as source of rank)
  - Powerful parameter to adjust the page ranks
- The vector E corresponds to the distribution of web pages that a random surfer periodically jumps to from the search engine
- Some highly accessed web pages have low page rank possibly because
  - People do not want to link to these pages from their own web pages (the example in 1998 PageRank paper is pornographic sites…)
  - Some important backlinks are omitted
  - Use web usage data as a start vector for PageRank.

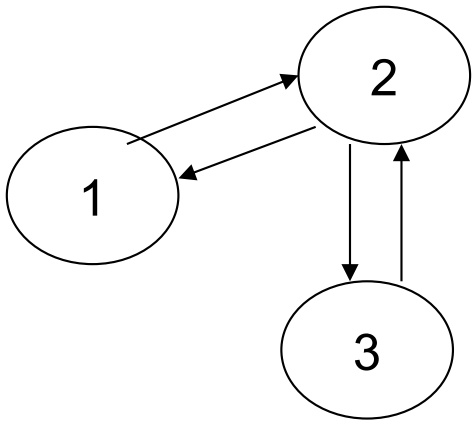
Web Spamming by Gaming Pagerank

- Since 2000, Google Search has become the default gateway to the web
- Very high premium to appear on the first few pages of search results
  - E-commerce sites
  - Advertising-driven sites

- Spamming: Manipulating the text of web pages in order to appear relevant to queries
- Approximately 10-15% of web pages are spam
- Spammers' goal: Maximize the page rank of a target page *t*
- Spammers' technique: Manipulating the text of web pages so as to appear relevant to queries and get as many links from accessible pages as possible to target page *t*

Exercise on PageRank

- Consider a Web graph with three nodes 1, 2, and 3. The links are as follows: 1->2, 3->2, 2->1, 2->3. Write down the transition probability matrices P for the surfer's walk with teleporting, with the value of teleport probability α=0.5.



A=

0	1	0
1	0	1
0	1	0

Each 1 divided by the number of ones in this row

(1- α)*

0	1	0
½	0	½
0	1	0

+

α*

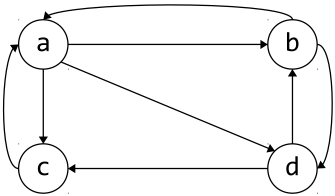
1/3	1/3	1/3
1/3	1/3	1/3
1/3	1/3	1/3

=

1/6	2/3	1/6
5/12	1/6	5/12
1/6	2/3	1/6

PageRank example

$$r_i = \sum_{j:j\to i\in\mathcal{E}} \frac{r_j}{d_j}$$



Equations:

- $r_a = \frac{r_b}{2} + r_c.$
- $r_b = \frac{r_a}{3} + \frac{r_d}{2}.$
- $r_c = \frac{r_a}{3} + \frac{r_d}{2}.$
- $r_d = \frac{r_a}{3} + \frac{r_b}{2}.$

- 4 equations, 4 unknowns, no constants.

  No **unique solution:** all solutions are equivalent modulo a scale factor.

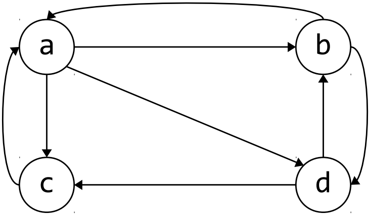
- Additional **constraint** for uniqueness:

$$\sum_i r_i = 1.$$

- **Solution** by **Gaussian elimination:**
  - $r_a = \frac{1}{3}.$
  - $r_b = r_c = r_d = \frac{2}{9}.$

Random walkers

- For **large graphs,** solving linear systems of equations is intractable.
- **Random surfers:** Where do you end if you follow links at random?

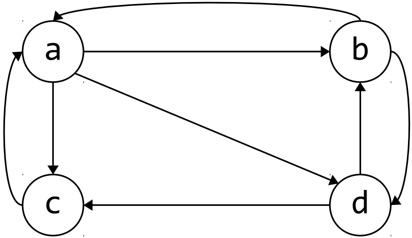


Start at node a: after one step, end up in b, c, or d with probability $\frac{1}{3}$.

- **Transition matrix:** $M_{ij} = \frac{1}{d_j}$ if $j \to i \in \mathcal{E}$ and 0 otherwise.

  The transition matrix is **column-stochastic:** columns sum to 1.

Random walkers: Transition matrix example



– **Transition matrix:**

$$\begin{bmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

PageRank with random walkers

- Start random surfers **at all pages** with **equal probability** $\frac{1}{n}$

$$\vec{v}_0 = [1/n, 1/n, \ldots, 1/n] .$$

- **After one step,** the distribution will be

$$\vec{v}_1 = M\vec{v}_0 .$$

- **After** k **steps:**

$$\vec{v}_k = M^k\vec{v}_0 .$$

- **Markov process:** The distribution approaches a limiting distribution $\vec{v}$ such that $\vec{v} = M\vec{v}$ if
  - The graph is **strongly connected:** can get from a node to any other node.
  - No **dead ends:** nodes that have no out-links.

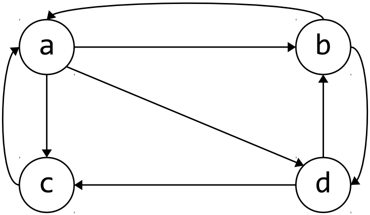
PageRank with random walkers

$$\vec{v} = M\vec{v}.$$

- Surfers are **stationary.**
- The more important a page, and the more likely it is to have a surfer.
- $\vec{v}$ is … **the principal eigenvector** of M. (M stochastic has largest eigenval 1.)
- **Power iteration:** compute $\vec{v}$ by iterative **matrix-vector multiplications.**
  - Stop when $||\vec{v}_t - \vec{v}_{t-1}|| \leq \epsilon$.
  - How eigenvectors are computed in large dimensions (eg. Lanczos method.)
  - Amenable to **MapReduce** parallelization.
- Equivalent to previous PageRank formulation:

$$v_i = \sum_{j:j \to i \in \mathcal{E}} \frac{v_j}{d_j}$$

Example



Transition matrix:

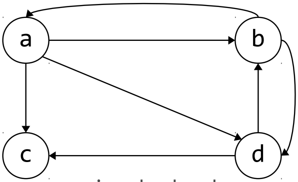
$$\begin{bmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

- **Initialization:** $\vec{v}_0 = [1/4,\ 1/4,\ 1/4,\ 1/4]\ .$
- **After one step:** $\vec{v}_1 = [9/24,\ 5/24,\ 5/24,\ 5/24]\ .$
- **After two steps:** $\vec{v}_2 = [15/48,\ 11/48,\ 11/48,\ 11/48]\ .$

  ...
- **Converges to:** $\vec{v} = [1/3,\ 2/9,\ 2/9,\ 2/9]\ .$

Dead ends

- **Dead ends:** nodes that have no out-links.



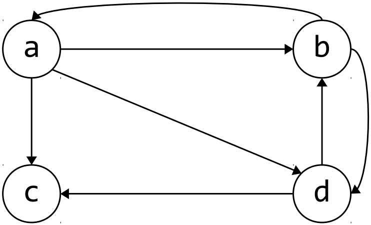
c is a dead end

Transition matrix:

$$\begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

- The **transition matrix** does not have full rank.
- It cannot be **inverted**, i.e. our linear system of equations has **no solution.**
- The **power method** converges to $\vec{v} = \vec{0}$.
- **Solutions:**
  - Recursively **remove** dead ends and their incoming links.
  - When at a dead end, **teleport** (with equal probability) to another node.

Example



Transition matrix:

$$\begin{bmatrix} 0 & 1/2 & \mathbf{0} & 0 \\ 1/3 & 0 & \mathbf{0} & 1/2 \\ 1/3 & 0 & \mathbf{0} & 1/2 \\ 1/3 & 1/2 & \mathbf{0} & 0 \end{bmatrix}$$

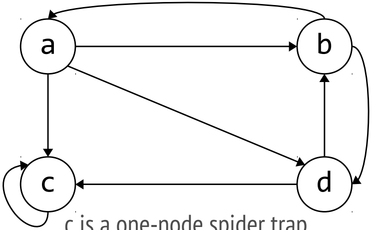
- New **transition matrix:**

$$\begin{bmatrix} 0 & 1/2 & \mathbf{1/4} & 0 \\ 1/3 & 0 & \mathbf{1/4} & 1/2 \\ 1/3 & 0 & \mathbf{1/4} & 1/2 \\ 1/3 & 1/2 & \mathbf{1/4} & 0 \end{bmatrix}$$

- Eventually, $\vec{v} = [1/5,\ 4/15,\ 4/15,\ 4/15]$.

Spider traps

- **Spider trap:** set of nodes with no dead ends but no links out.
- **Problem:**
  - All random surfers end up in the spider trap.



c is a one-node spider trap

- **Transition matrix:**

$$\begin{bmatrix} 0 & 1/2 & \mathbf{0} & 0 \\ 1/3 & 0 & \mathbf{0} & 1/2 \\ 1/3 & 0 & \mathbf{1} & 1/2 \\ 1/3 & 1/2 & \mathbf{0} & 0 \end{bmatrix}$$

- $\vec{v}$ **converges to** $\vec{v} = [0,\ 0,\ 1,\ 0]$.

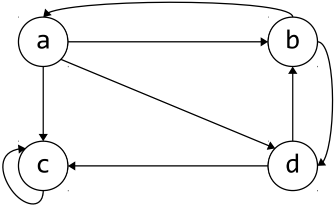
Taxation

- How to get out of **spider traps?**
  - A random surfer can **leave the graph** at any moment.
  - **New surfers** can be started at any page at any moment.
- **Taxation:** Allow each random surfer a probability $1 - \beta$ of **teleporting** to a random page

$$\vec{v} = \beta M \vec{v} + \frac{(1-\beta)}{n}\vec{1}.$$

Typically, $\beta \in [0.8 - 0.9]$.

Example



Transition matrix:

$$\begin{bmatrix} 0 & 1/2 & \mathbf{0} & 0 \\ 1/3 & 0 & \mathbf{0} & 1/2 \\ 1/3 & 0 & \mathbf{1} & 1/2 \\ 1/3 & 1/2 & \mathbf{0} & 0 \end{bmatrix}$$

$$\vec{v} = \beta M \vec{v} + \frac{(1-\beta)}{n}\vec{1}$$

- $\beta = 0.8 = 4/5$

$$\vec{v} = \begin{bmatrix} 0 & 2/5 & 0 & 0 \\ 4/15 & 0 & 0 & 2/5 \\ 4/15 & 0 & 4/5 & 2/5 \\ 4/15 & 2/5 & 0 & 0 \end{bmatrix} \vec{v} + \begin{bmatrix} 1/20 \\ 1/20 \\ 1/20 \\ 1/20 \end{bmatrix}, \quad \vec{v}_0 = \left[\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right].$$

- **Solution:** $\vec{v} = \left[\frac{15}{148}, \frac{19}{148}, \frac{95}{148}, \frac{19}{148}\right].$

Summary

- **Large-scale data** poses new technical problems for:
  - **storage** $\Rightarrow$ distributed file systems.
  - **computations** $\Rightarrow$ MapReduce programming model.
    - Split the data in chunks.
    - Map workers all execute the same operation on a chunk and return a key-val pair.
    - Reduce workers process all key-val pairs with the same key at once.
- **Algorithmic costs** of MapReduce:
  - **Communication costs** vs. **computation costs.**
  - **Reducer size** and **replication rate.**
- Extensions of MapReduce: **Spark** and **TensorFlow.**
- MapReduce for **machine learning.**
- **Link analysis** with **PageRank.**

PageRank Summary

- Robust and scalable algorithm with proven convergence guarantees
- Distributed algorithm in Google’s data center-drive breakthroughs in compute (Google MapReduce) and storage (Google File System)
- Amenable to distributed computation via parallel computation (MapReduce in next Lecture)
- MapReduce Code walkthrough:
- http://web.archive.org/web/20221216071408/https://michaelnielsen.org/blog/using-mapreduce-to-compute-pagerank