PageRank

Ben Burns, Dan Magazu, Lucas Chagas,
Thomas Webster, Trung Do

Fall 2021

Table of Contents

Motivation

- Problem: the internet has a *lot* of web pages
- A lot of the information out there either isn't relevant to us, or is inaccurate
- Motivation: we want a program that, when provided a phrase, returns webpages with information relevant to the input
- Intuitive solution: return back websites that either contain that phrase, or contain similar phrases
- This helps us find more *relevant* pages, but we can't know if we're getting the best information (much less *accurate* information) without manually going through each result
- We desire a stronger solution

PageRank

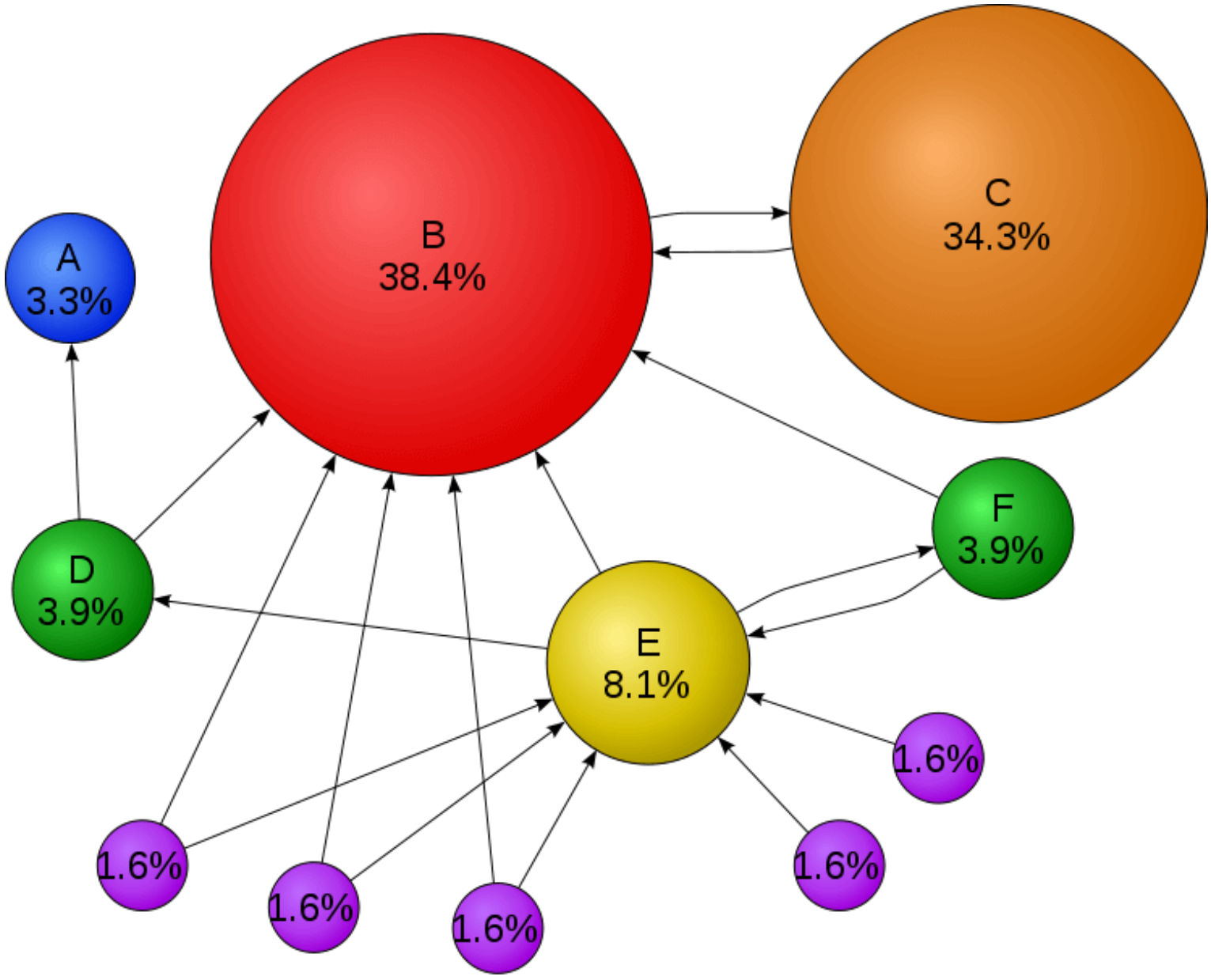
- Invented by Sergey Brin and Larry Page (1998)[1]
  - Publication marks them becoming co-founders of Google
- Idea: we want some way to numerically discern how important each webpage is
- PageRank: the procedure for scoring each website
- Google: the database that indexes the PageRank of each website for search
- Everytime Google crawls the web to reindex, it recalculates the PageRank scores for all webpages

[1] Many use the year of the original manuscript, 1996

The Big Idea behind PageRank

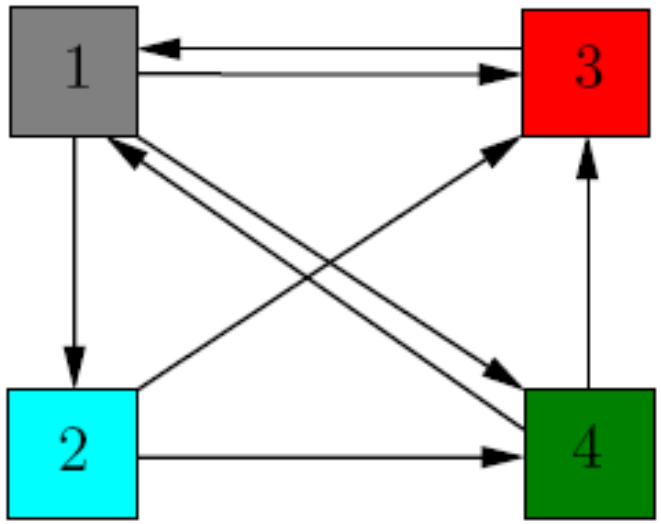
- PageRank is an algorithm to determine the importance of a website relatively to all other websites. The algorithm ranks the importance of website w, i.e., PR(w), based on the number of links points to website w and the *quality* of each pointing link from the other source website.
- **The underlying assumption**: More important website are likely to receive more links from other website. Since the algorithm measures the relative popularity ("ranking") between all websites, websites with higher ranking score are ranked higher. The sum of all ranking score equals 1 (will see why later).

Visulization of PageRank score



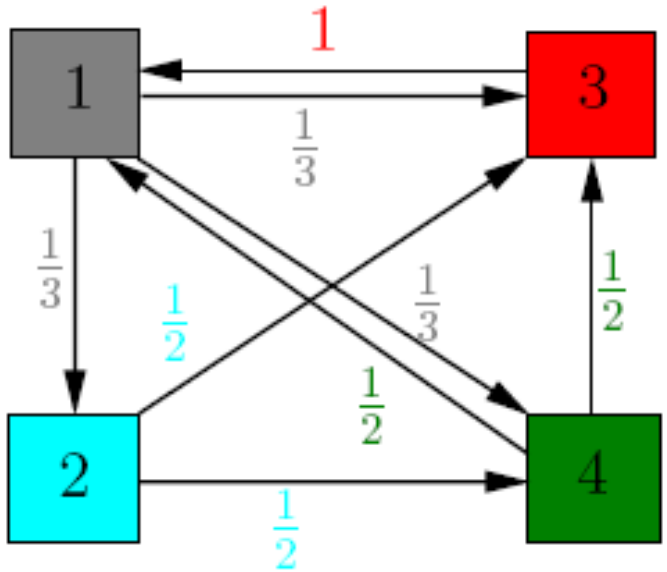
Formalizing the PageRank problem

- As a graph: Each website is represented by a node assigned with a PageRank value, denoted by PR(w).
- If a website w has a link to another website v (meaning there are an outbound link from w and an inbound link to v), then there is a directed edge from node w to node v.
- Multiple links from w to v is treated as a single edge from node w to v, and all self-links from a website to itself are ignored. **Thus, this is a node-weighted, simple, no self-loop directed graph**.

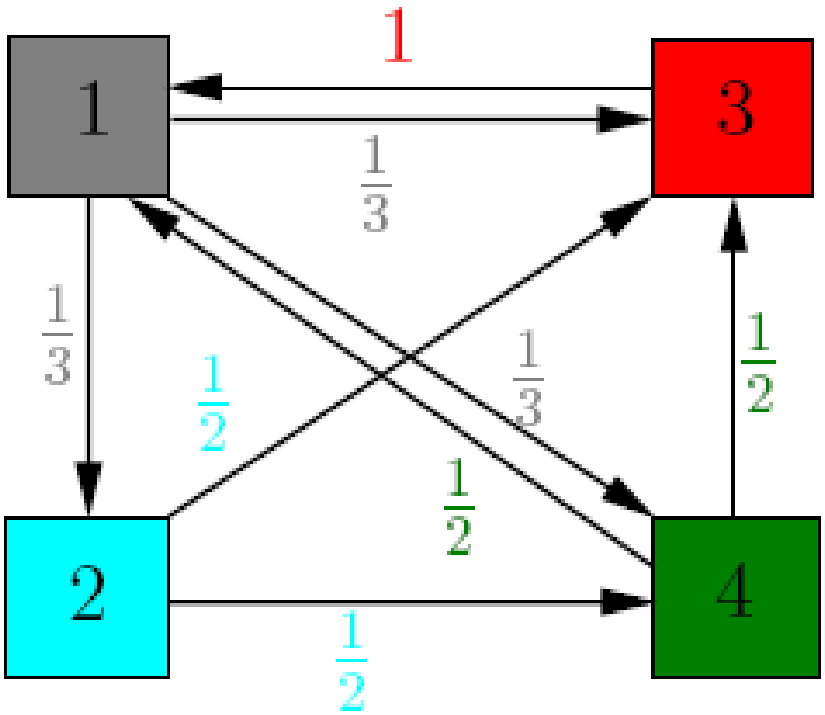


Add Weight to Edges

- An edge from u to v will "transfer" the score from node u to node v an amount of $\mathrm{PR}(u)/\mathrm{O}(u)$, where $\mathrm{PR}(u)$ is the current score of u, and $\mathrm{O}(u)$ is the total number of outlinks of node u. Thus, $\mathrm{PR}(v)$ += $\mathrm{PR}(w)/\mathrm{L}(w)$.
- In another words, for a given node in the graph:
- **An outbound link** will "give" away the PR value of the source node to the recipient node.
- **An inbound link** will add the PR value from the source node to the recipient node.



Formulate the transition matrix A



$$A = \begin{bmatrix} 0 & 0 & 1 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{bmatrix}$$

- Notice some special properties of matrix A:
  - All entries of A are non-negative.
  - If there is an outlink from node w, then the column corresponding to node w has the sum of its entries equals 1. Otherwise, the sum equals 0.
- Matrix A is called a **column stochastic matrix** (all matrix entries are non-negative and the sum of the entries in every column is 1).

Translate PageRank Problem into an Eigenvector Problem

- Let us denotes vector v with each entry of the vector as the PageRank value of the corresponding nodes:

$$v = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

- Analyzing the relationship between 4 nodes in the example we get the system

$$\begin{cases} x_1 = 1 \cdot x_3 + \frac{1}{2} \cdot x_4 \\ x_2 = \frac{1}{3} \cdot x_1 \\ x_3 = \frac{1}{3} \cdot x_1 + \frac{1}{2} \cdot x_2 + \frac{1}{2} \cdot x_4 \\ x_4 = \frac{1}{3} \cdot x_1 + \frac{1}{2} \cdot x_2 \end{cases}$$

Eigenvector Problem (cont.)

- Thus we can translate the problem into the eigenvector problem:

$$A \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Continue Solving

- The conditions to rigorously use the eigenvector method will be defined below. For now, we can solve for the eigenvector corresponding to the eigenvalue 1:

$$c \cdot \begin{bmatrix} 12 \\ 4 \\ 9 \\ 6 \end{bmatrix}$$

- Every vector in the above format is the eigenvector corresponding to eigenvalue 1.

Continue Solving (cont.)

- We choose c such that the sum of all entries in this vector equals 1 (we later refer to it as "the probabilistic eigenvector corresponding to the eigenvalue 1")

$$\frac{1}{31} \cdot \begin{bmatrix} 12 \\ 4 \\ 9 \\ 6 \end{bmatrix} \sim \begin{bmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{bmatrix}$$ is our PageRank vector

Power Iteration Method

- Want to determine the eigenvector of the square matrix A
- Start out with

$$\vec{v_0} = \begin{bmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{bmatrix}$$

  as a candidate eigenvector corresponding to eigenvalue of 1
- Calculate $\vec{v_1} = A\vec{v_0}$ to get a new vector
- Continue to iterate until v_k converges (if it does as at all)

A few questions to ask...

- Will v_k always converge?
- Will v_k give us any useful information about the ranking of websites?

Random Walk Interpretation

- Consider someone browsing the web who is randomly clicking on links. We refer to this person as the "random walker"
- Interpret our rank vector r as a probability distribution $p(t)$ where t is time
- $p(t) = \left[\frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N}\right]^T$ is a vector where at some time t the i^{th} index represents the probability that the random walker is at page i.
- To predict where the random walker is at time $t+1$ we calculate $Mp(t) = p(t+1)$

Random Walk Interpretation

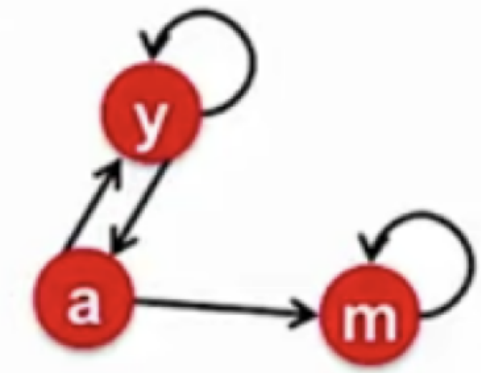
- Suppose that for some t we find that $Mp(t) = p(t)$
- In other words, we have found a steady state distribution!
- We recognize this as a Markov process where the probability that the random walker is in a given page at some time t is solely determined by what our probability distribution looked like at time $t - 1$
- From Markov theory, we know that the transition matrix M needs to follow a few characteristics for $p(t)$ to converge to a unique vector
- What constraints do we need to apply to M so that the power iteration algorithm will always converge?

When Power Iteration Fails

- Before discussing constaints on M we need to observe the cases where power iteration either fails to converge or produces a nonsense result
- So far, we have gone along with the formulation of PageRank where the importance of a webpage depends on its incoming links
- We haven't taken the time to consider if this approach even makes sense
- Consider two problems that arise:
  - Spider Traps
  - Dead Ends

Spider Traps

- Our random walker gets stuck in a portion of the graph leading to most of the "importance" to flow into subgraph where the walker is stuck
- Example: what happens when we run power iteration on this web graph? (notice that the random walker will get stuck on page m)



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

Spider Trap Continued

- Power iteration successfuly converges! So we're done right?
- Wrong. Page m receives a rank of $r_m = 1$ while the rest of the pages are of no importance
- From the random walker's perspective this makes sense. But this is not what we want.

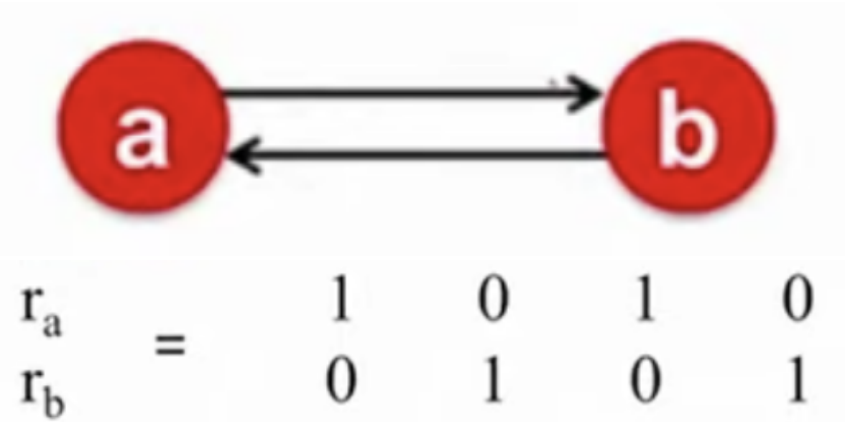
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \dots & 0 \\ 1/3 & 3/6 & 7/12 & 16/24 & & 1 \end{matrix}$$

Iteration 0, 1, 2, ...

Another Spider Trap

Another example:

Page a starts with an importance of 1 and passes it page b in the first iteration. Then b passes 1 to $a, \cdots$ And this process continues indefinitely.



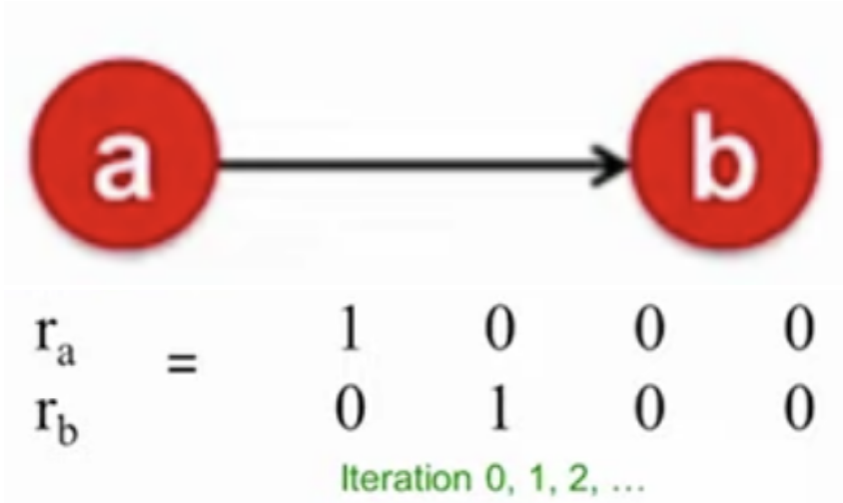
We fail to converge!

The Solution to the Spider Traps

- At each iteration, we give the random walker the option to either continue following outbound links or to randomly "teleport" to another page on the web
- We denote the probability that the user clicks a link by β, and the probability of teleporting by 1 -β
- Now our random walkers can't get stuck in a subgraph of our web, and we avoid having a subset of pages hog all of the "importance" value

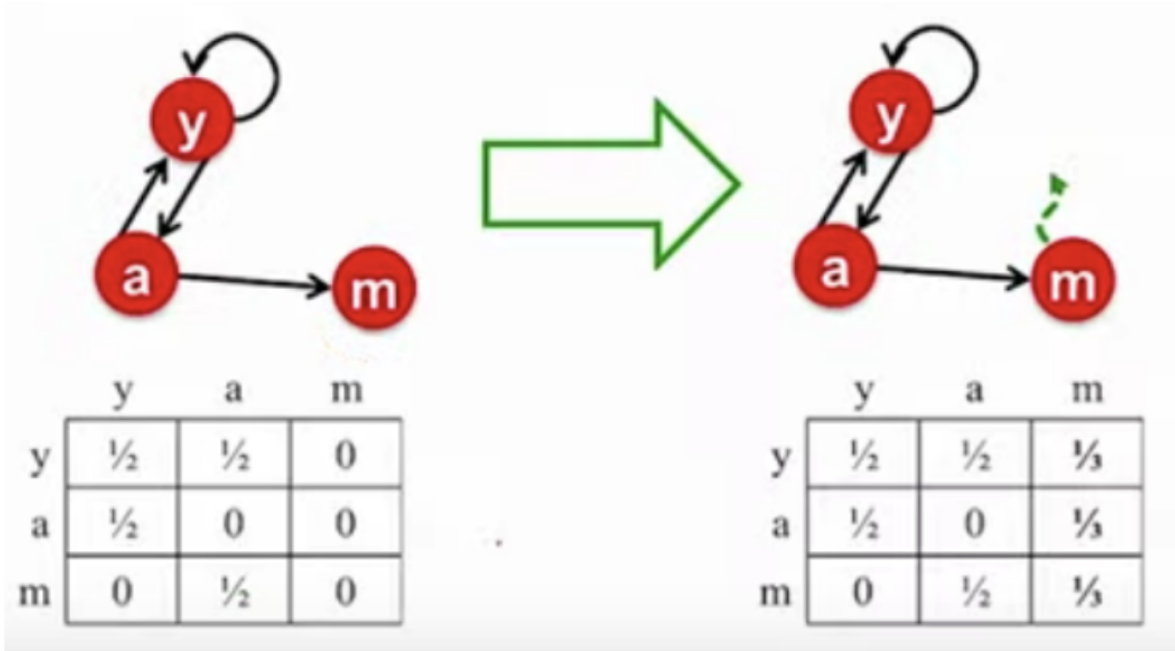
Dead Ends

- The random walker encounters a web page without any outbound links leaving them nowhere to go.
- Example: We converge but power iteration tells us that both page a and b are unimportant!



The Solution to Dead Ends

- If our random walker encounters a page with no outbound links, we immediately teleport them to a random web page
- Example:



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

	y	a	m
y	½	½	⅓
a	½	0	⅓
m	0	½	⅓

Do Teleports Solve Convergence?

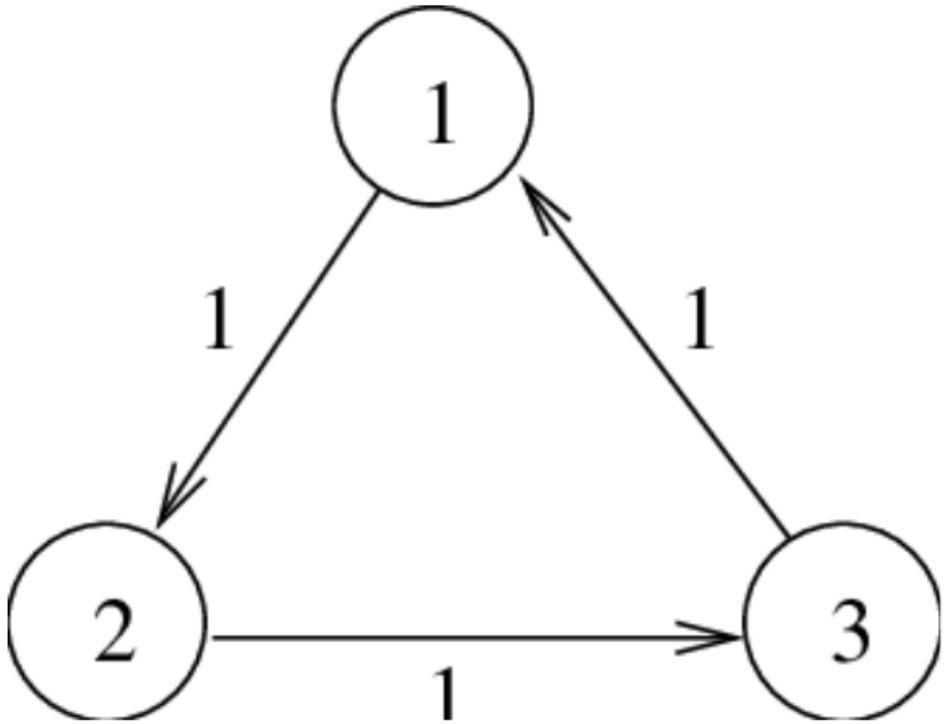
- Yes. In fact, after dealing with spider traps and dead ends, the power iteration algorithm will always converge
- Recall that power iteration is simply a Markov process where if matrix M has special characteristics it will always converge
- Markov theory tells us that if M is stochastic, aperiodic, and irreducible, then it will always converge to a unique, positive, stationary vector

Stochastic

- Every column in the transition matrix must sum to 1
- When initially constructing our adjacency matrix A, if a page has no outbound links (dead end) then the entire column would sum up to 0.
- Our solution to deadends guarantees that every column adds up to 1.

Aperiodic

- A chain is periodic if there exist $k > 1$ such that the interval between two visits to some state s is always a multiple of k
- Our solution to spider traps also introduces self-loops into the graph
- This turns the graph aperiodic



Irreducible

- Irreducible: From any state, there is a non-zero probability of going from any one state to another
- Our solution to spider traps gives the random walker the opportunity to randomly jump to any page in the web graph
- Therefore, we will always be able to navigate

The PageRank equation

- Google's solution makes the Markov Transition matrix stochastic, aperiodic, and irreducible.
- PageRank equation:

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1 - \beta)\frac{1}{n}$$

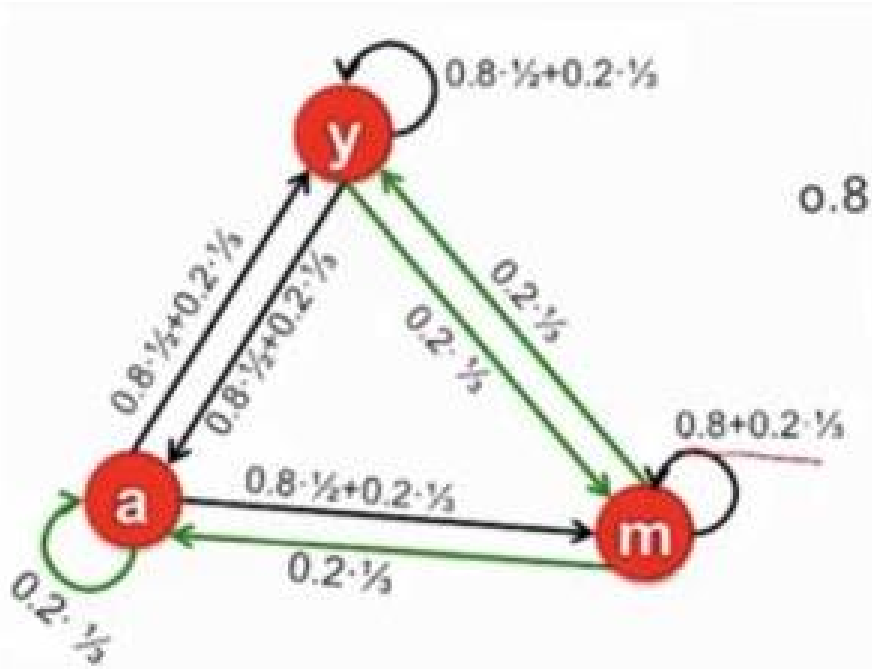
  - The summation is the sum of all of the importance of node I that point to it where r_j and r_i is the probability that the random surf is on this node.
  - This is divided by d_i, which is the probability that the random surf traverses the link when iterating towards j. This only happens with probability β, when the surfer decides to follow the link.
  - The $(1 - \beta)/n$ represents when the random walkers decide to jump somewhere else, using the probability $(1 - \beta) \cdot 1/n$, where n is the number of nodes in the entire network.

The Google Matrix

$$A = \beta M + (1 - \beta)\frac{1}{n} e \cdot e^T$$

- Matrix A, also known as the google matrix, is the transition matrix multiplied with ß(for random jumps) plus the probabilities due to random jumps, known as (1-ß).
- This is then multiplied by e, the outer product of a vector that is of all 1s.

Using the Google Matrix Example



$$0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

M $\qquad$ $1/n \cdot 1 \cdot 1^T$

$$\begin{array}{c} \\ y \\ a \\ m \end{array} \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}$$

A

$$\begin{array}{c} y \\ a \\ m \end{array} = \begin{array}{c} \\ 1/3 \\ 1/3 \\ 1/3 \end{array} \begin{array}{cccc} & 0.33 & 0.24 & 0.26 \\ & 0.20 & 0.20 & 0.18 \\ & 0.46 & 0.52 & 0.56 \end{array} \ldots \begin{array}{c} \\ 7/33 \\ 5/33 \\ 21/33 \end{array}$$

Sources

https://www.classes.cs.uchicago.edu/archive/2017/fall/12100-1/lecture-examples/PageRank/slides.pdf
https://web.stanford.edu/class/cs246/slides/10-spam.pdf
https://www.cs.rpi.edu/ slotag/classes/FA16/slides/lec04-web1.pdf
https://www.stat.cmu.edu/ ryantibs/datamining/lectures/03-pr.pdf
https://www.ccs.neu.edu/home/daikeshi/notes/PageRank.pdf
http://www.cs.sjsu.edu/faculty/pollett/masters/Semesters/Fall11/tanmayee/Deliverable3.pdf
https://www.stat.uchicago.edu/ lekheng/meetings/mathofranking/ref/gleich.pdf
http://web.eecs.utk.edu/ roffutt/files/sp20ppts/PageRank.pdf
https://www.amsi.org.au/teacher_modules/pdfs/Maths_delivers/Pagerank5.pdf
http://www.ams.org/publicoutreach/feature-column/fcarc-pagerank