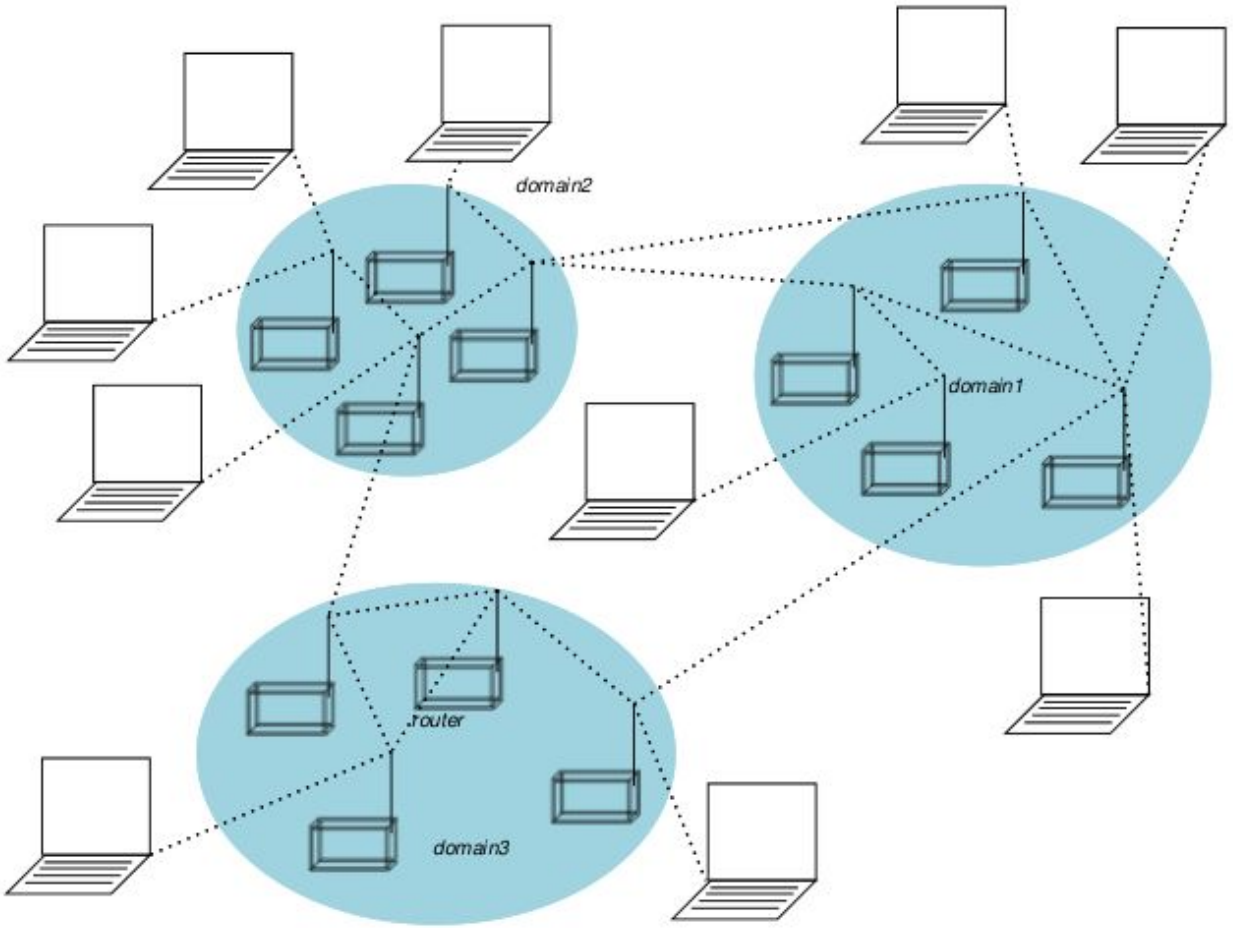
Link Analysis

Graph Data: Social Networks

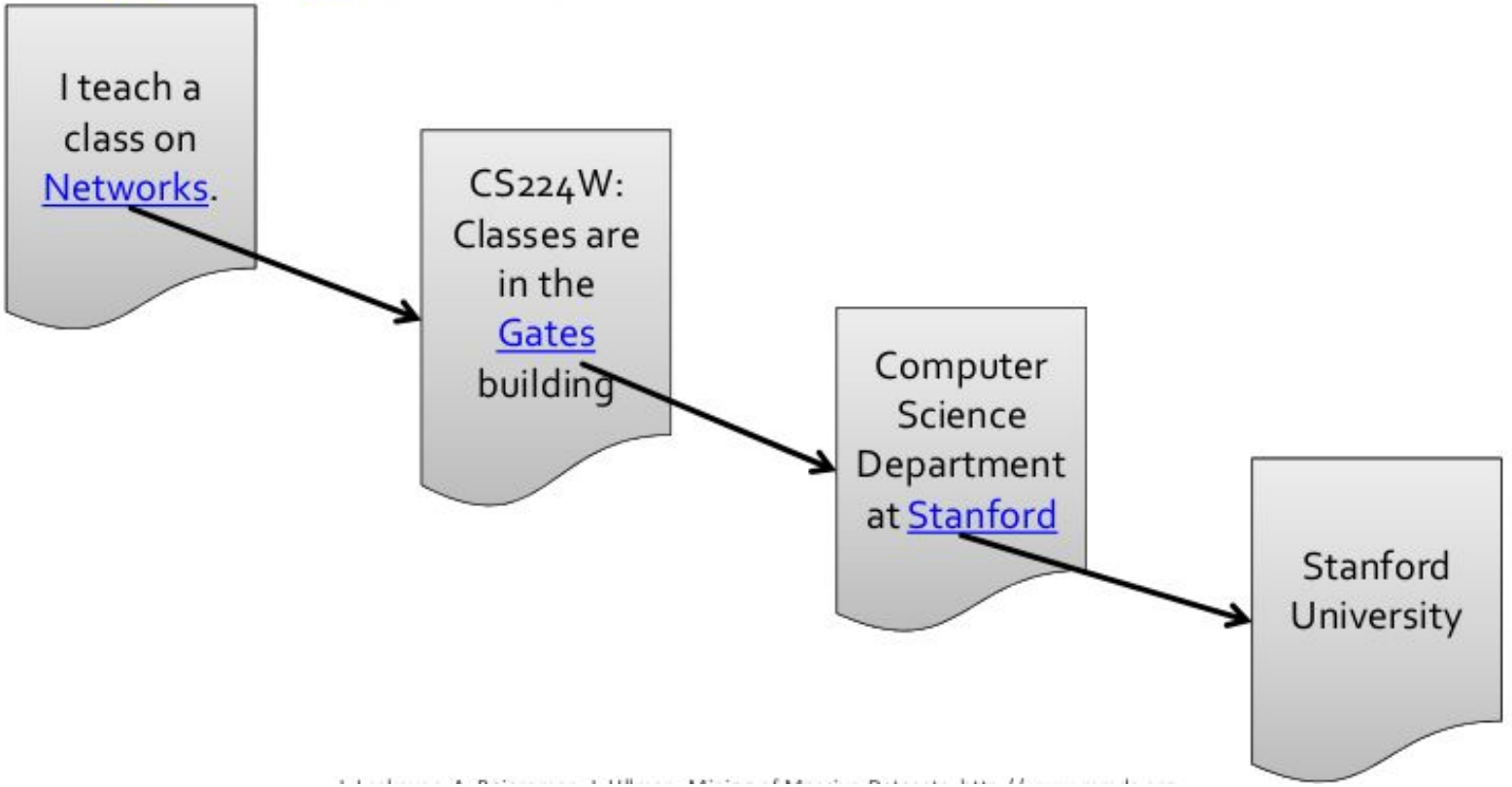
Graph Data: Communication Nets



Internet

Web as a Graph

- **Web as a directed graph:**
  - **Nodes: Webpages**
  - **Edges: Hyperlinks**



2 challenges of web search:

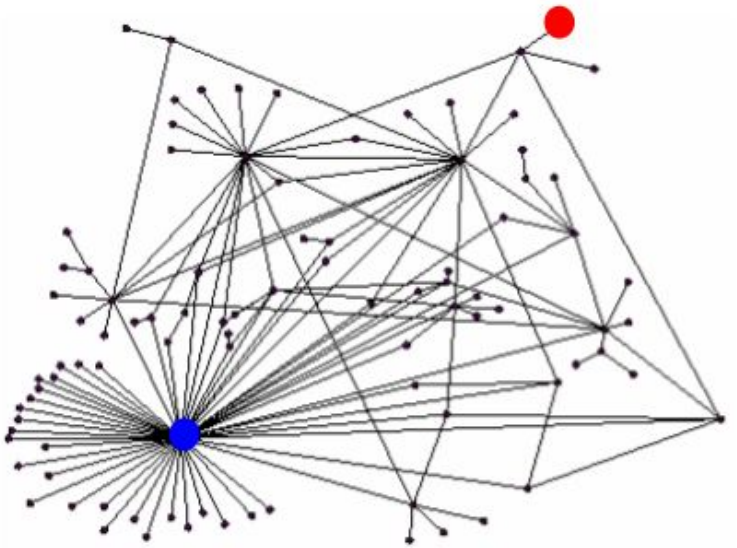
- **(1) Web contains many sources of information Who to "trust"?**
  - **Trick:** Trustworthy pages may point to each other!
- **(2) What is the "best" answer to query "newspaper"?**
  - No single right answer
  - **Trick:** Pages that actually know about newspapers might all be pointing to many newspapers

Ranking Nodes on the Graph

- **All web pages are not equally "important"**

  Www.trinitySchool.com vs. www.stanford.edu

- There is large diversity in the web-graph node connectivity. **Let's rank the pages by the link structure!**



Link Analysis Algorithms

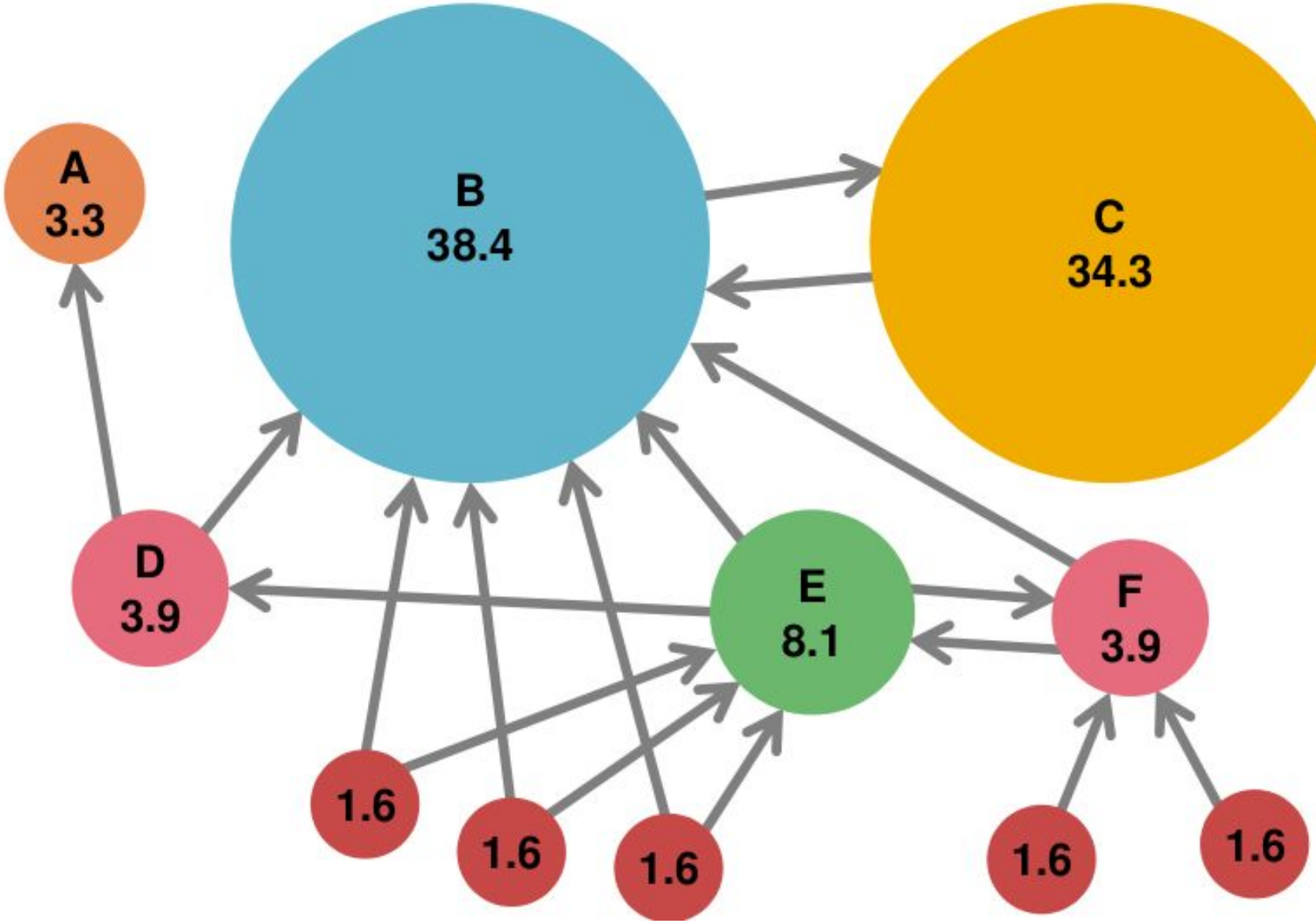
- We will cover the following **Link Analysis approaches** for computing **importances** of nodes in a graph:
  - Page Rank
  - Hubs and Authorities (HITS)
  - Topic-Specific (Personalized) Page Rank
  - Web Spam Detection Algorithms

Formulation of a page Rank – Page rank depends on various factors one of them is link.

Links as Votes

- **Idea: Links as votes**
  - **Page is more important if it has more links**
    - In-coming links? Out-going links?
- **Think of in-links as votes:**
  - www.stanford.edu has 23,400 in-links
  - www.joe-schmoe.com has 1 in-link
- **Are all in-links are equal?**
  - **Links from important pages count more**
  - Recursive question!

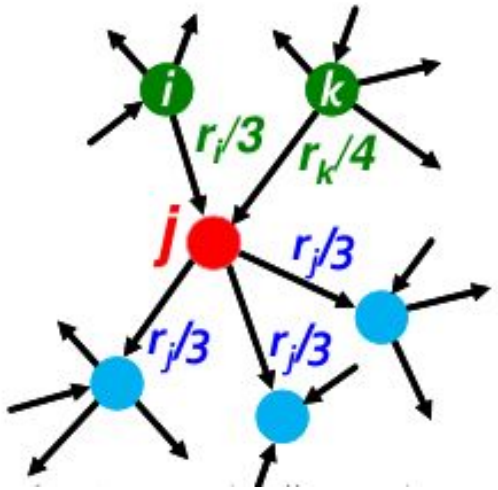
Example: PageRank Scores



Simple Recursive Formulation

- Each link's vote is proportional to the **importance** of its source page
- If page ***j*** with importance r_j has ***n*** out-links, each link gets r_j / n votes
- Page ***j***'s own importance is the sum of the votes on its in-links

$$r_j = r_i/3 + r_k/4$$

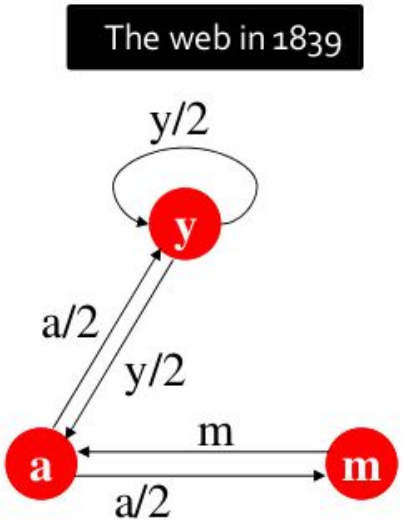


PageRank: The "Flow" Model

- A "vote" from an important page is worth more
- A page is important if it is pointed to by other important pages
- Define a "rank" r_j for page j

$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d_i}}$$

d_i ... out-degree of node i



"Flow" equations:

$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$

- The importance of node m is importance of node a divided by 2

Solving the Flow Equations

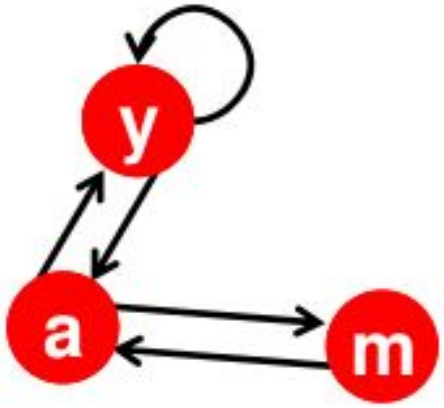
Flow equations:

$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2 + r_m$$
$$r_m = r_a/2$$

- **3 equations, 3 unknowns, no constants**
  - No unique solution
  - All solutions equivalent modulo the scale factor
- **Additional constraint forces uniqueness:**
  - $r_y + r_a + r_m = 1$
  - Solution: $r_y = \frac{2}{5}$, $r_a = \frac{2}{5}$, $r_m = \frac{1}{5}$
- **Gaussian elimination method works for small examples, but we need a better method for large web-size graphs**
- **We need a new formulation!**

Example: Flow Equations & M

Rows represent inbound links and columns represent outbound links.



$$M = \begin{array}{c|ccc} & y & a & m \\ y & 1/2 & 1/2 & 0 \\ a & 1/2 & 0 & 1 \\ m & 0 & 1/2 & 0 \end{array}$$

$$r = M \cdot r$$

$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2 + r_m$$
$$r_m = r_a/2$$

These flow equ. can be written in a matrix form as

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix}$$

- We can find an infinite set of solutions to these set of equations. There is no unique solution. So we add a constraint to the system that our paging scores have to sum to 1. So we could use any kind of linear system to solve these equations. For e.g Gaussian Elimination. This is how we can compute the importances of nodes in the graph. But this approach works well only for small graphs and not for web graphs. We have billions of web pages meaning we have billions of equations to solve. So we need different formulation – Flow formulation

PageRank: Matrix Formulation

- **Stochastic adjacency matrix $\boldsymbol{M}$**
  - Let page i has d_i out-links
  - If $i \rightarrow j$, then $M_{ji} = \frac{1}{d_i}$ else $M_{ji} = 0$
    - $\boldsymbol{M}$ is a **column stochastic matrix**
      - Columns sum to 1
- **Rank vector $\boldsymbol{r}$:** vector with an entry per page
  - r_i is the importance score of page i
  - $\sum_i r_i = 1$
- **The flow equations can be written**

$$r_j = \sum_{i \rightarrow j} \frac{r_i}{\mathrm{d_i}}$$

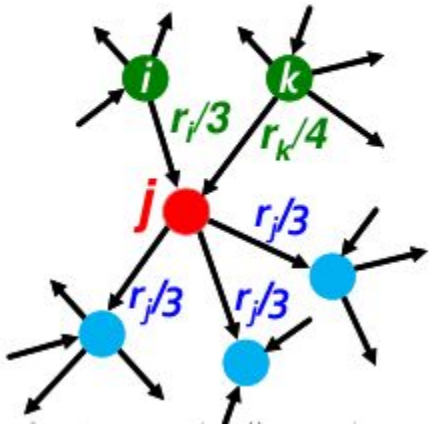
$$\boldsymbol{r} = \boldsymbol{M} \cdot \boldsymbol{r}$$

Summary

- We are representing whole graph as a column stochastic matrix. Now we take all the page rank scores of nodes and represent that as a columnar vector r. In order to create such a vector, we think that we have one entry in our vector per page. Our pages are numbered as 1,2,3...N So our vector has N rows and 1 column. Every entry in the vector is equal to page rank score of that web page. Now we will write our flow equation in terms of Matrix M and rank vector r. So we can write it as rank vector r equals the matrix M times the rank vector r again. M represents outdegree of a graph which is fixed as web graph is not changing and we figure out new rank vector r.

Page *j*'s own importance is the sum of the votes on its in-links

$r_j = r_i/3 + r_k/4$



Why $r = M.r = r_i/3 + r_k/4$

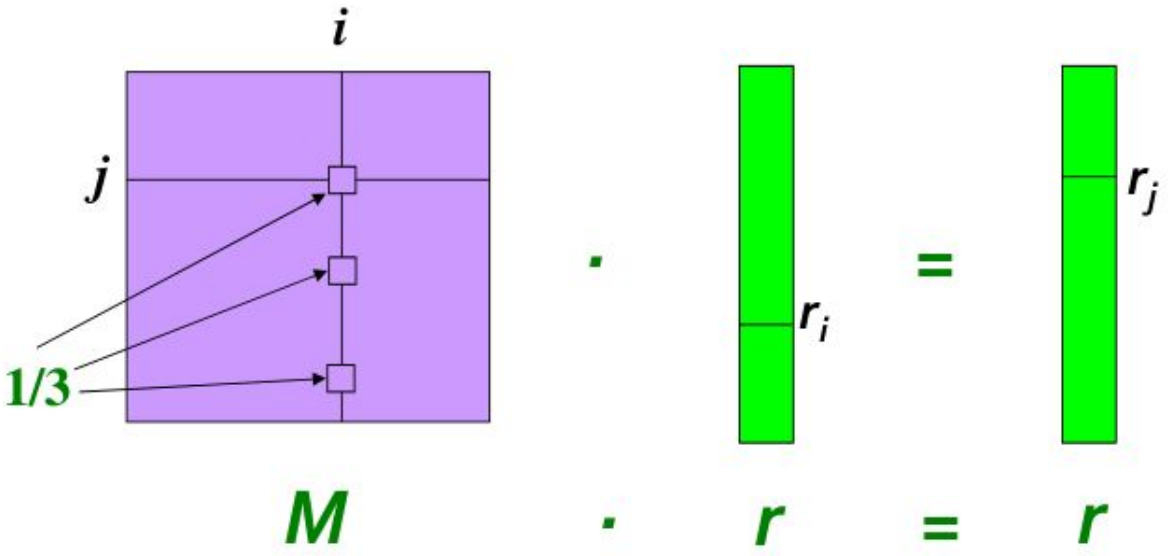
Imagine I take jth row and multiply it with the vector r. When i am scanning jth row from left to right and vector r from top to bottom, i am basically computing page rank score of jth node. page rank score of jth node is the sum of the importances that are stored in r times the out degree of that node that points to j.

Example

- **Remember the flow equation:** $r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d_i}}$
- **Flow equation in the matrix form**

$$M \cdot r = r$$

- **Suppose page *i* links to 3 pages, including *j***



Flow equation can be expressed as a vector matrix product.

Eigen Vector

- An eigenvector v of a linear transformation T is a non-zero vector that, when T is applied to it, does not change direction. Applying T to the eigenvector only scales the eigenvector by the scalar value λ, called an eigenvalue. This condition can be written as the equation $T(v) = v\lambda$

Eigenvector Formulation

- **The flow equations can be written**

$$\boldsymbol{r} = \boldsymbol{M} \cdot \boldsymbol{r}$$

- So the **rank vector** ***r*** is an **eigenvector** of the stochastic web matrix ***M***
  - In fact, its first or principal eigenvector, with corresponding eigenvalue ***1***
    - Largest eigenvalue of ***M*** is **1** since ***M*** is column stochastic (with non-negative entries)
      - *We know* ***r*** *is unit length and each column of* ***M*** *sums to one, so* $\boldsymbol{Mr} \leq \mathbf{1}$

NOTE: ***x*** is an eigenvector with the corresponding eigenvalue **λ** if:

$$\boldsymbol{Ax} = \boldsymbol{\lambda x}$$

- **We can now efficiently solve for *r*!**
  The method is called Power iteration

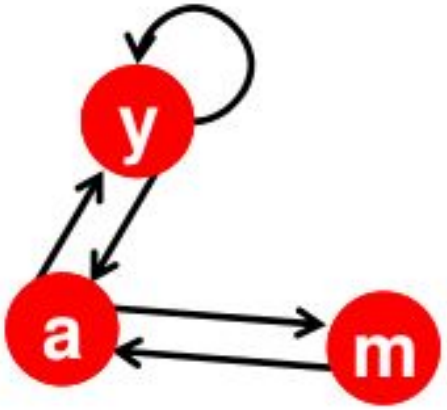
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

or equivalently

$$r = Mr$$

r = M . r ---- this is a recursive flow equation. This equation very much looks like a Eigen value problem.
Eigen value equation is given by $Ax = \lambda x$. A is a given matrix , and we want to compute two things, vector x and λ which is scalar – real or complex number. Here vector x is called as eigen vector. Now compare our flow equation r = M . r with $Ax = \lambda x$. M times r is same as A times x and $\lambda = 1$.
Vector r has a unit length meaning that its coordinate sum are non-negative and they sum to 1. And each column of M also sums to 1. So M times r will be atmost 1. This means the largest eigen value is 1. We represented web as a graph We reformulated our flow equations into this matrix formulation. Now we establish the connection between the matrix formulation and eigen vectors of matrix M. So instead of solving a system of equations we can think of our problem as finding eigen vecor of Matrix M. There is an efficient method of finding eigen vector of matrix. Method is called power iteration. So we now actually know how to compute page rank. To compute page rank we have to find the eigen vector of Matrix M that corresponds to eigen value = 1.

Example: Flow Equations & M



$$M = \begin{array}{c|ccc} & y & a & m \\ \hline y & 1/2 & 1/2 & 0 \\ a & 1/2 & 0 & 1 \\ m & 0 & 1/2 & 0 \end{array}$$

$$r = M \cdot r$$

$$r_y = r_y/2 + r_a/2$$
$$r_a = r_y/2 + r_m$$
$$r_m = r_a/2$$

These flow equ. can be written in a matrix form as

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix}$$

Power Iteration Method

- **Given a web graph with *n* nodes, where the nodes are pages and edges are hyperlinks**
- **Power iteration:** a simple iterative scheme
  - Suppose there are *N* web pages
  - Initialize: $\mathbf{r}^{(0)} = [1/N,\ldots.,1/N]^T$
  - Iterate: $\mathbf{r}^{(t+1)} = \mathbf{M} \cdot \mathbf{r}^{(t)}$
  - Stop when $|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}|_1 < \varepsilon$

$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i}$$

d_i …. out-degree of node i

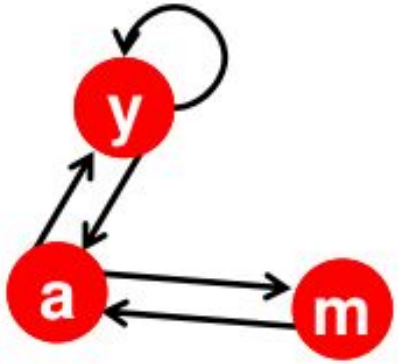
$|\mathbf{x}|_1 = \sum_{1 \le i \le N} |x_i|$ is the $\mathbf{L_1}$ norm
Can use any other vector norm, e.g., Euclidean

PageRank: How to solve?

- **Power Iteration:**
  - Set $r_j = 1/N$
  - **1:** $r'_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - **2:** $r = r'$
  - Goto **1**
- **Example:**

$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{matrix} 1/3 \\ 1/3 \\ 1/3 \end{matrix}$$

Iteration 0, 1, 2, …



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

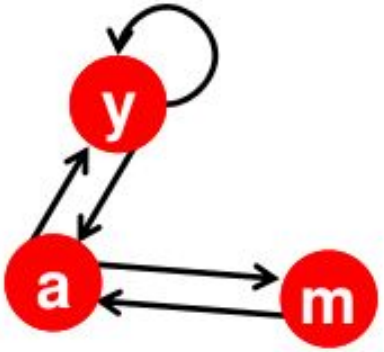
$r_m = r_a/2$

PageRank: How to solve?

- Algorithm for

  Power Iteration:

  - Set $r_j = 1/N$
  - **1:** $r'_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - **2:** $r = r'$
  - Goto **1**



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$

- **Example:**

$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{matrix} 1/3 & 1/3 & 5/12 & 9/24 & \\ 1/3 & 3/6 & 1/3 & 11/24 & \dots \\ 1/3 & 1/6 & 3/12 & 1/6 & \end{matrix} \begin{matrix} 6/15 \\ 6/15 \\ 3/15 \end{matrix}$$

Iteration 0, 1, 2, …

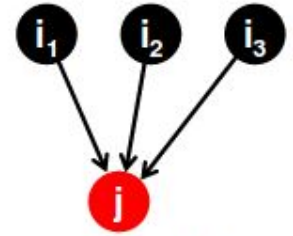
6/15 = 2/5
This is same ans that we got with system of equ.

How does power iteration method works?

We start with some intial guess of how our rank vector r is then we multiply it with M usually around 50 to 100 times, we monitor r from one iteration to next iteration and When r stops changing we stop. And what we get is the page rank score. Thus instead of solving flow equations we can find the matrix M, to find the page rank scores.

Random Walk Interpretation

- **Imagine a random web surfer:**
  - At any time t, surfer is on some page i
  - At time $t+1$, the surfer follows an out-link from i uniformly at random
  - Ends up on some page j linked from i
  - Process repeats indefinitely



$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d}_{\mathrm{out}}(\mathrm{i})}$$

- **Let:**
  - $p(t)$ ... vector whose i^{th} coordinate is the prob. that the surfer is at page i at time t
  - So, $p(t)$ is a probability distribution over pages

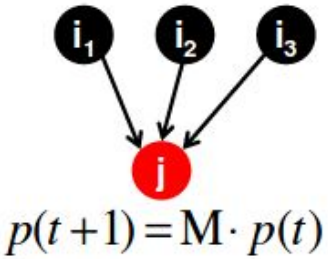
Random Walk Interpretation

- Page rank scores are equivalent to a probability distribution of our random walker in a graph. Page rank scores corresponds to the probability that this random surfer at some given time t resides at the given node. This random walk is also called Markov process

The Stationary Distribution

- **Where is the surfer at time $t+1$?**
  - Follows a link uniformly at random

$$\boldsymbol{p(t+1) = M \cdot p(t)}$$



- Suppose the random walk reaches a state

$$\boldsymbol{p(t+1) = M \cdot p(t) = p(t)}$$

  then $\boldsymbol{p(t)}$ is **stationary distribution** of a random walk

- **Our original rank vector $\boldsymbol{r}$** satisfies $\boldsymbol{r = M \cdot r}$
  - **So, r is a stationary distribution for the random walk**

Existence and Uniqueness

- **A central result from the theory of random walks (a.k.a. Markov processes):**

For graphs that satisfy **certain conditions**, the **stationary distribution is unique** and eventually will be reached no matter what the initial probability distribution at time **t = 0**

- Web graph or Matrix M must satisfy some conditions in order for page rank to exist. We know that importance of page j = sum of importance of pages i that point to j.

PageRank: Three Questions

$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

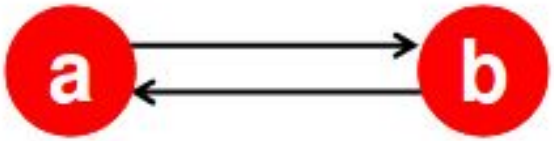
or equivalently

$$r = Mr$$

- Does this converge?
- Does it converge to what we want?
- Are results reasonable?

As we have said in our previous slides that the page rank vector is unique and stationary distribution will always be raeched regardless of how do we initialize out initial vector r. But in the following graph, our page rank computation will never converge. This problem is called spider trap problem.

Does this converge?



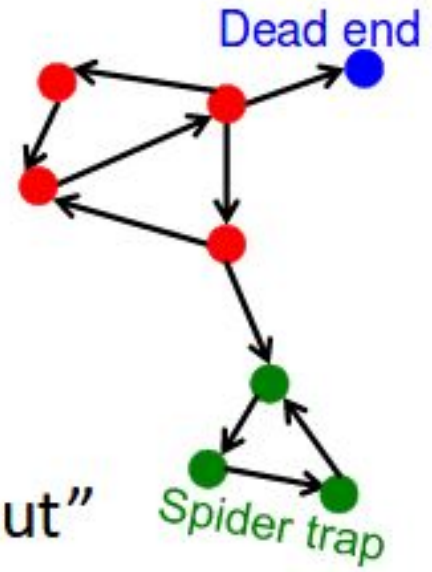
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

The "Dead end" problem:

Does it converge to what we want?

$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

PageRank: Problems



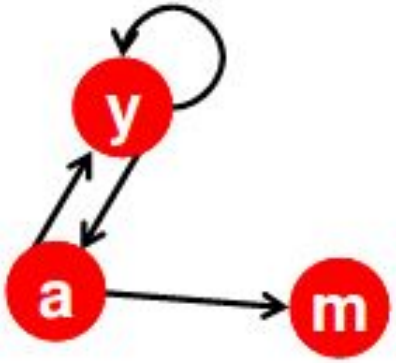
2 problems:

- **(1)** Some pages are **dead ends** (have no out-links)
  - Random walk has "nowhere" to go to
  - Such pages cause importance to "leak out"

- **(2) Spider traps:** (all out-links are within the group)
  - Random walked gets "stuck" in a trap
  - And eventually spider traps absorb all importance

Problem: Dead Ends

- **Power Iteration:**
  - Set $r_j = 1$
  - $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
    - And iterate



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

$r_y = r_y/2 + r_a/2$
$r_a = r_y/2$
$r_m = r_a/2$

- **Example:**

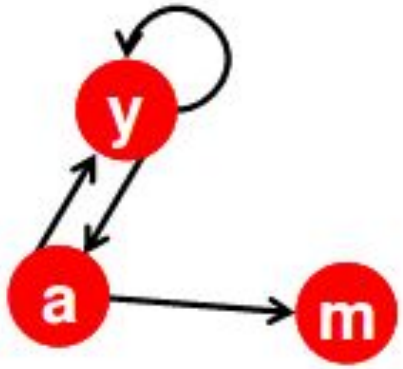
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\ 1/3 & 1/6 & 1/12 & 2/24 & & 0 \end{matrix}$$

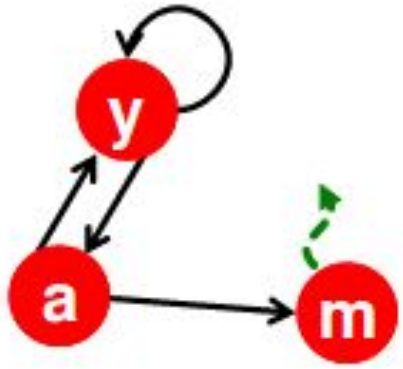
Iteration 0, 1, 2, …

Here the PageRank "leaks" out since the matrix is not stochastic.

Solution: Always Teleport!

- **Teleports:** Follow random teleport links with probability 1.0 from dead-ends
  - Adjust matrix accordingly





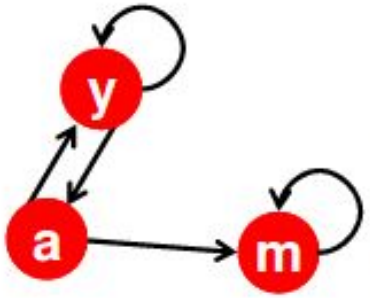
	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

	y	a	m
y	½	½	⅓
a	½	0	⅓
m	0	½	⅓

If we think of a random walker browsing this graph and if we ask after lot of time where do you think the random walker will be? The ramdom walker will be at node m and will never be able to come out. This means that basically all the page rank scores will be concentrated at node m.

Problem: Spider Traps

- **Power Iteration:**
  - Set $r_j = 1$
  - $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
    - And iterate



m is a spider trap

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

$r_y = r_y/2 + r_a/2$
$r_a = r_y/2$
$r_m = r_a/2 + r_m$

- **Example:**

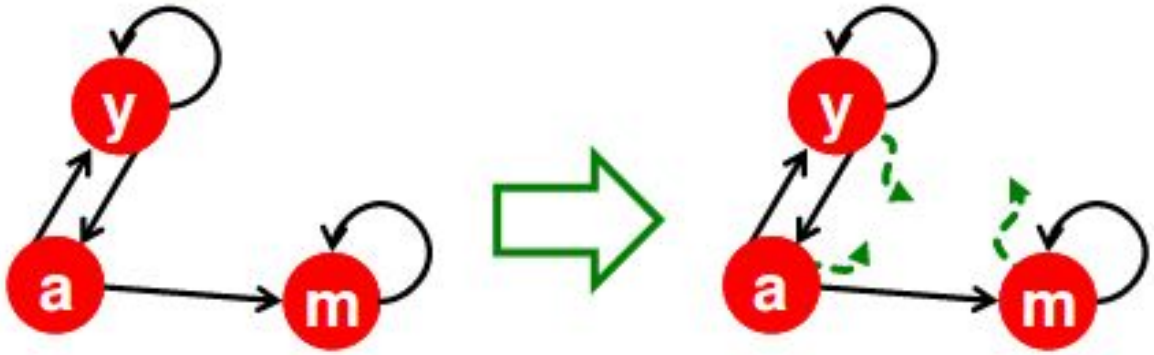
$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \dots & 0 \\ 1/3 & 3/6 & 7/12 & 16/24 & & 1 \end{matrix}$$

Iteration 0, 1, 2, …

All the PageRank score gets "trapped" in node m.

Solution: Teleports!

- **The Google solution for spider traps: At each time step, the random surfer has two options**
  - With prob. β, follow a link at random
  - With prob. **1-β**, jump to some random page
  - Common values for β are in the range 0.8 to 0.9
- **Surfer will teleport out of spider trap within a few time steps**

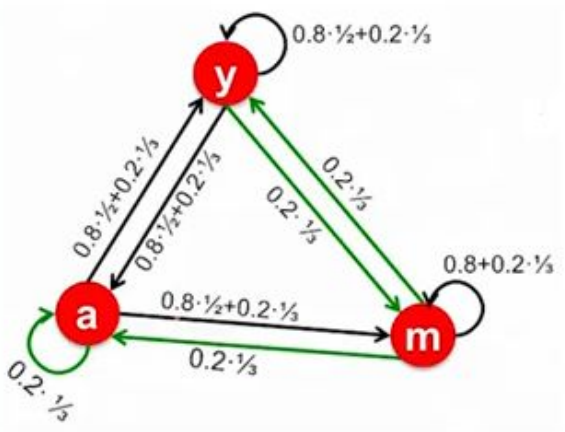


$$\mathbf{A} = \beta \cdot \mathbf{M} + (1-\beta)\,[1/N]_{N \times N}$$

$$A = 0.8 \begin{bmatrix} ½ & ½ & 0 \\ ½ & 0 & 0 \\ 0 & ½ & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

Green edges are due to random jumps. This is how teleports solve the problem. Score of node y is 7 over 33, Score of node a is 5 over 33, Score of node m is 21 over 33

Random Teleports ($\beta = 0.8$)



$$A = 0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix} = \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}$$

M $[1/N]_{NxN}$ A

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

The good value of beta is set between 0.8 to 0.9 and usually it is set at 0.85 If beta is set to 0.85 then every after 5 steps random walker would do a random jump. So a random walker on an average would do 5 steps and then do a random jump another 5 steps and jump so on.

Why Teleports Solve the Problem?

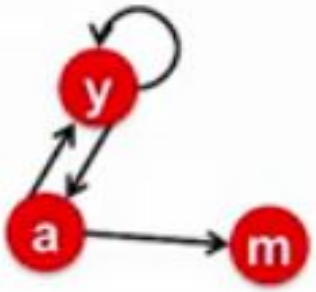
Why are dead-ends and spider traps a problem and why do teleports solve the problem?

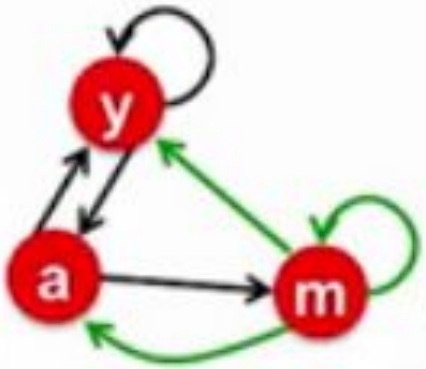
- **Spider-traps** are not a problem, but with traps PageRank scores are **not** what we want
  - **Solution:** Never get stuck in a spider trap by teleporting out of it in a finite number of steps
- **Dead-ends** are a problem
  - The matrix is not column stochastic so our initial assumptions are not met
  - **Solution:** Make matrix column stochastic by always teleporting when there is nowhere else to go

Fact: For **any vector**, the power method applied to a matrix ***P*** will **converge** to a **unique** positive stationary vector as long as ***P*** is **stochastic**, **irreducible** and **aperiodic**.

Make M Stochastic

- **Stochastic:** Every column sums to **1**
- **Solution:** Add **green** links

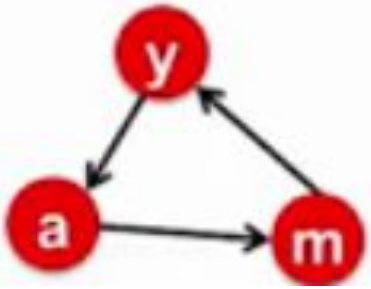


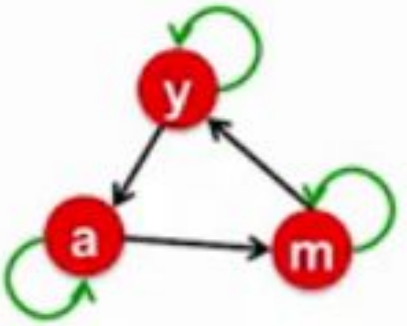


	y	a	m
y	½	½	1/3
a	½	0	1/3
m	0	½	1/3

Make M Aperiodic

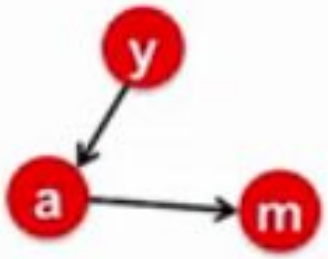
- A chain is **periodic** if there exists $k > 1$ such that the interval between two visits to some state s is always a multiple of k
- **Solution:** Add **green** links

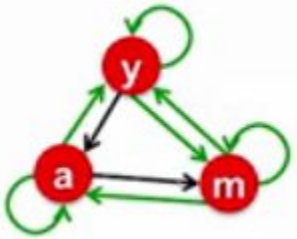




Make *M* Irreducible

- From any state, there is a non-zero probability of going from any one state to any another
- **Solution:** Add **green** links





Solution: Random Jumps

- **Google's solution that does it all:**
  - Makes ***M*** **stochastic, aperiodic, irreducible**
- **At each step, random surfer has two options:**
  - With probability β, follow a link at random
  - With probability $1\text{-}\beta$, jump to some random page
- **PageRank equation**

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

d_i ... out-degree of node i

The above formulation assumes that ***M*** has no dead ends. We can either preprocess matrix ***M***

The Google Matrix

- **PageRank equation** [Brin-Page, '98]

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

- **The Google Matrix *A*:**

$[1/N]_{N \times N}$…N by N matrix where all entries are 1/N

$$A = \beta M + (1 - \beta) \left[\frac{1}{N}\right]_{N \times N}$$

- **We have a recursive problem: $r = A \cdot r$**

  And the Power method still works!

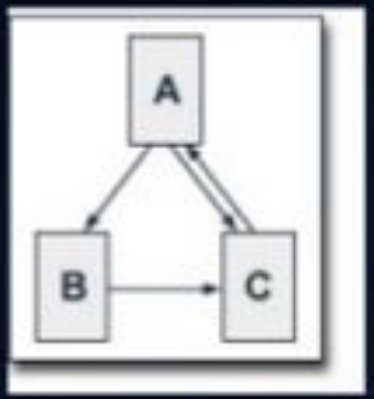
- **What is β?**
  - In practice $\beta = 0.8, 0.9$ (make 5 steps on avg., jump)

PageRank flow equations using Map Reduce

Implementing PageRank Using MapReduce

- Multiple stages of mappers and reducers are needed
- Output of reducers are feed into the next stage mappers
- The initial input data for the example will be organized as

A	B	C
B	C	
C	A	



- In each row
  - The first column contains our nodes
  - Other columns are the nodes that the main node has an outbound link to

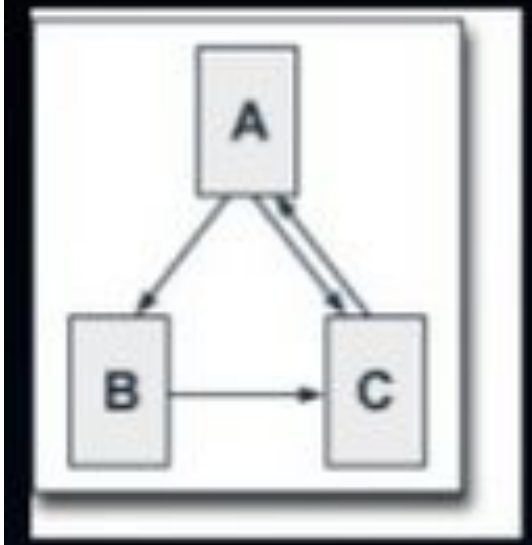
Implementing PageRank Using MapReduce

- Mappers receive values as follows
  - (y, PR(y) x1 x2 … xn)
- And emit the following values for each row
  - (y, PR(y) x1 x2 … xn)
  - for i = 1 … n
    (xi, $\frac{PR(y)}{C(y)}$)

Implementing PageRank Using MapReduce

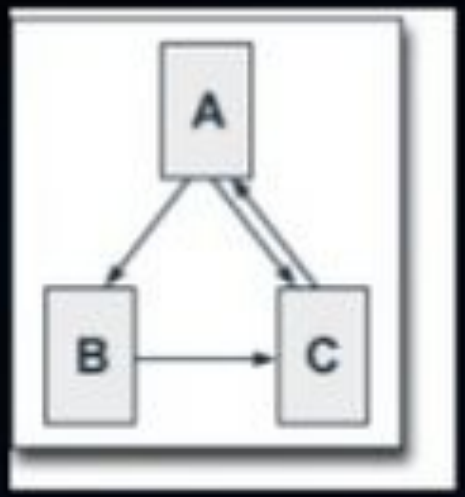
- Reducers receive values from mappers and use the PageRank formula to aggregate values and calculate new PageRank values
- New Input file for the next phase is created
- The differences between New PageRanks and old PagesRanks are compared to the convergence factor

Implementing PageRank Using MapReduce



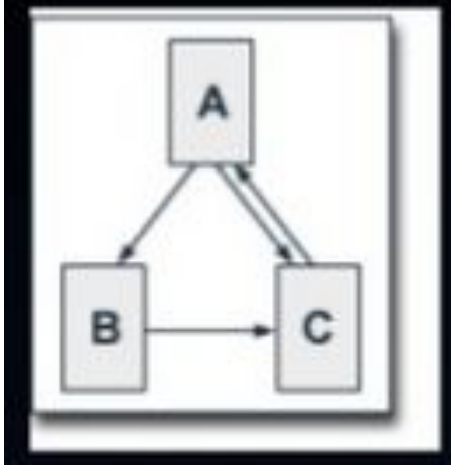
- Mappers in our example
  - A 1/3 B C =>
    (B, 1/6)
    (C, 1/6)
  - B 1/3 C =>
    (C, 1/3)
  - C 1/3 A =>
    (A, 1/3)

Implementing PageRank Using MapReduce



- Reducers in our example
  - (A, 1/3 B C)
    (A, 1/3)
    => (A, 1/3 B C)
  - (B, 1/3 C)
    (B, 1/6)
    => (B, 1/6 C)
  - (C, 1/3 A)
    (C, 1/6)
    (C, 1/3)
    => (C, 1/6+1/3 A)

Implementing PageRank Using MapReduce



- The new input file for mappers in the next phase will be
  - A 0.3333 B C
    B 0.1917 C
    C 0.4750 A