Analysis of Large Graphs: Link Analysis, PageRank

CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
http://cs246.stanford.edu

New Topic: Graph Data!

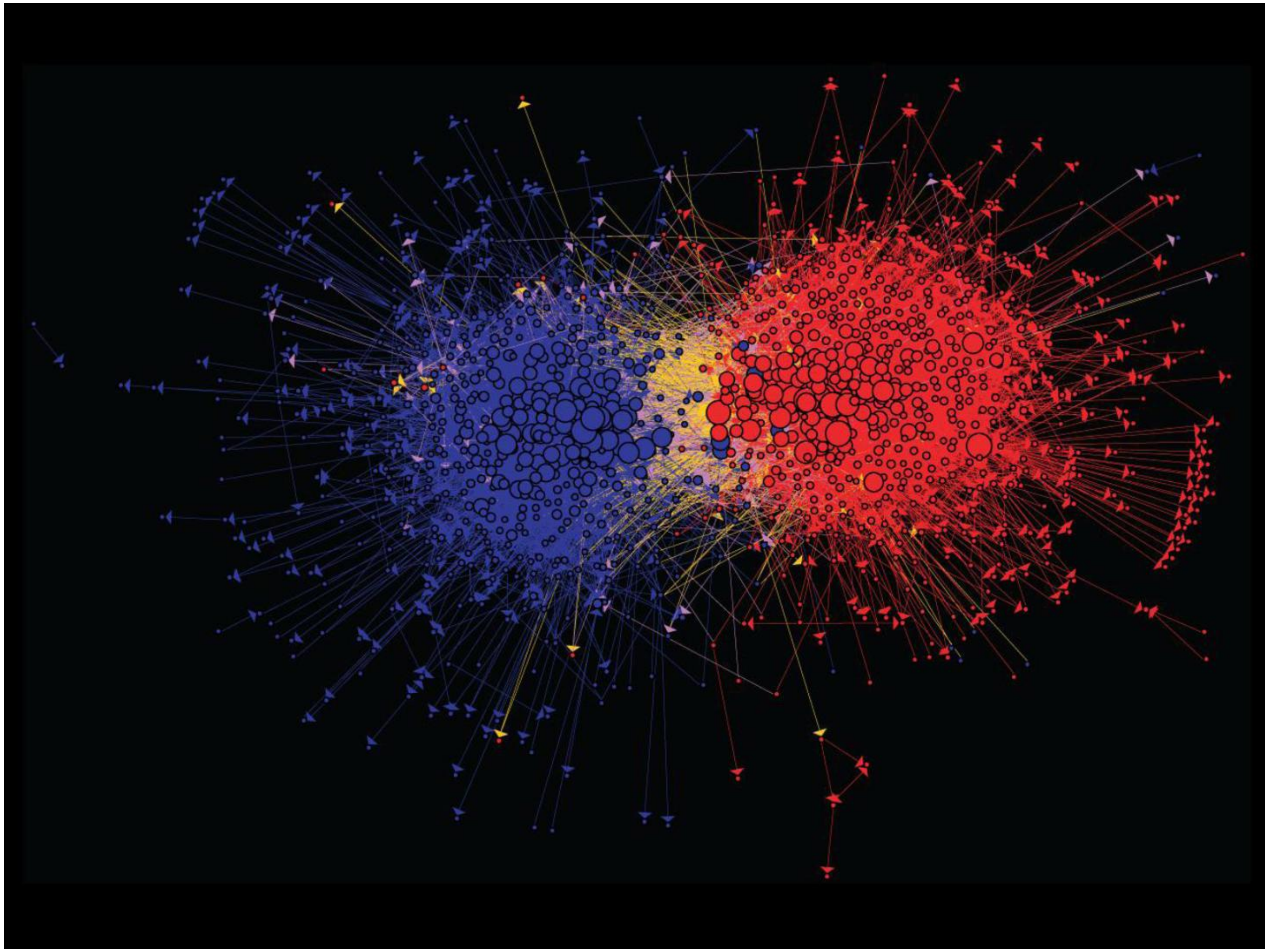
High dim. data	Graph data	Infinite data	Machine learning	Apps
Locality sensitive hashing	**PageRank,** SimRank	Filtering data streams	SVM	Recommender systems
Clustering	Community Detection	Web advertising	Decision Trees	Association Rules
Dimensionality reduction	Spam Detection	Queries on streams	Perceptron, kNN	Duplicate document detection

Graph Data: Social Networks

Facebook social graph

4-degrees of separation [Backstrom-Boldi-Rosa-Ugander-Vigna, 2011]

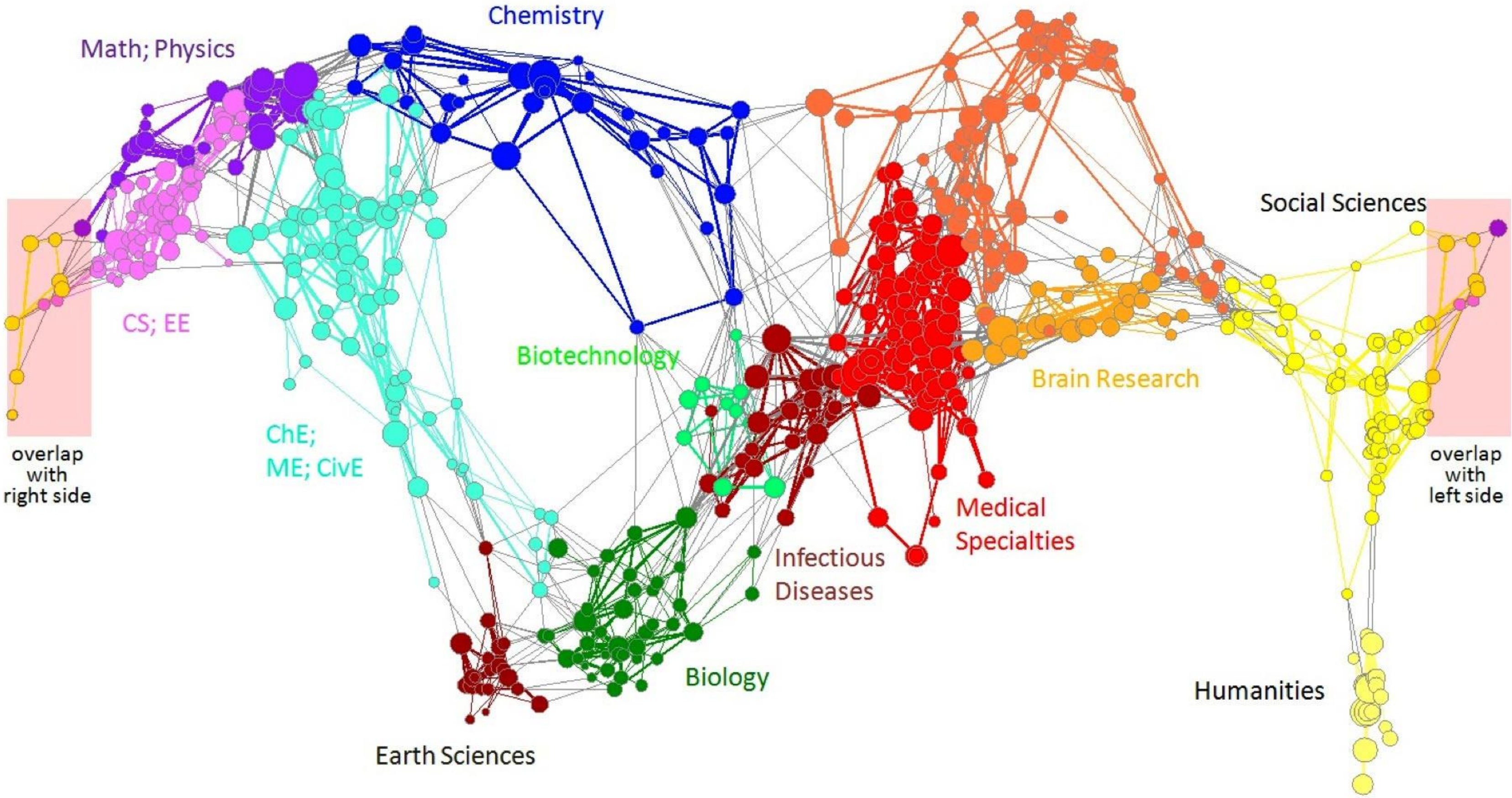
Graph Data: Media Networks



Connections between political blogs

Polarization of the network [Adamic-Glance, 2005]

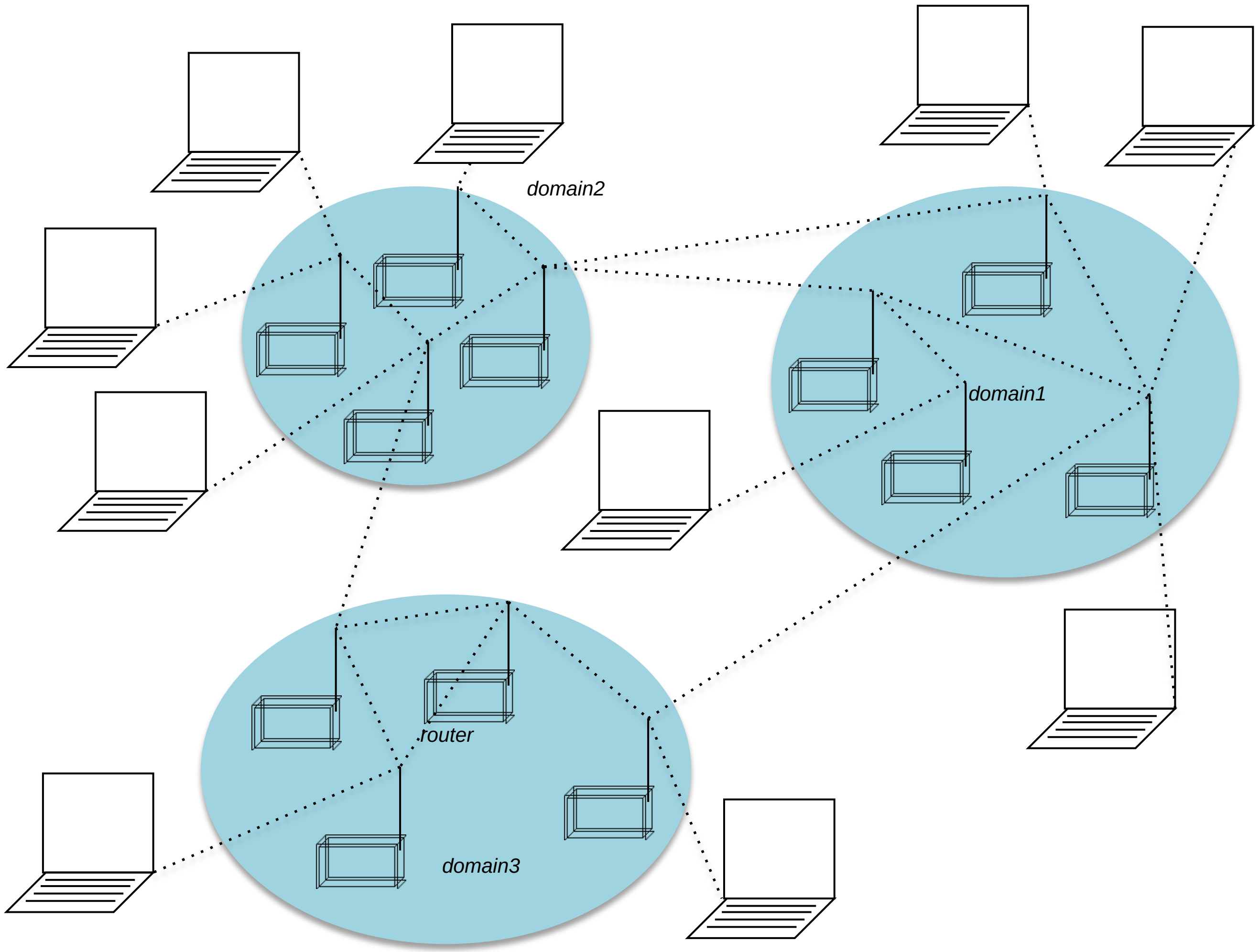
Graph Data: Information Nets



Citation networks and Maps of science

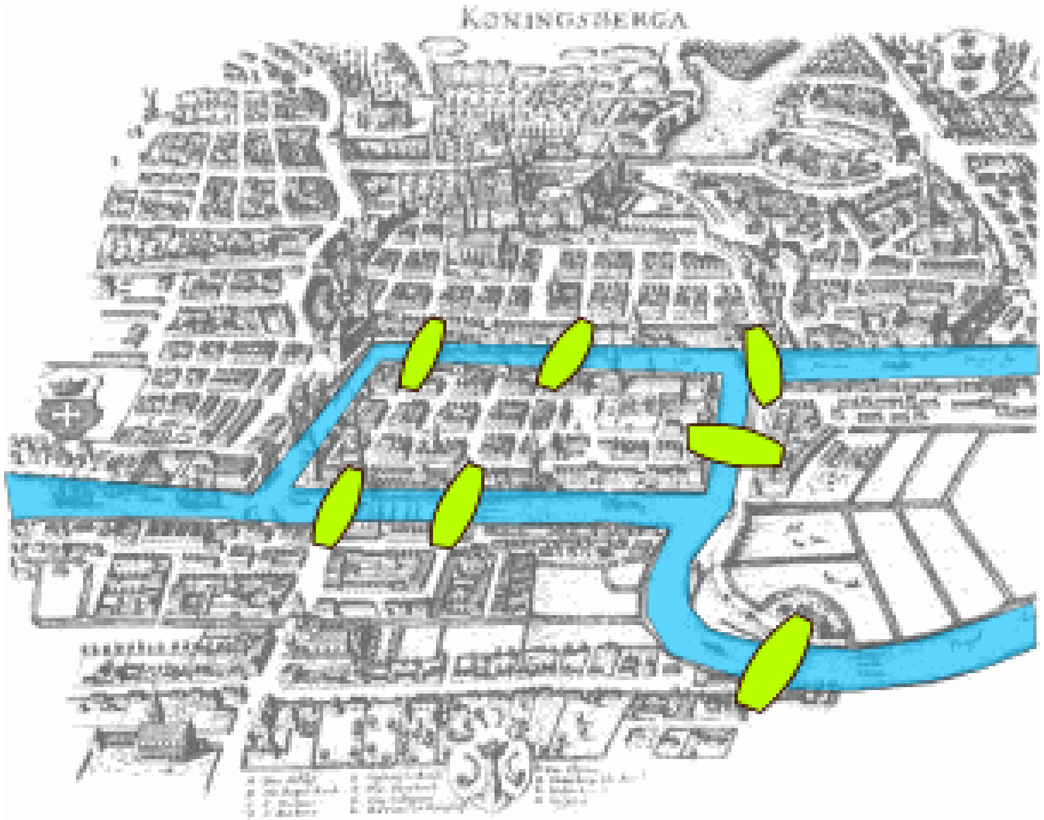
[Börner et al., 2012]

Graph Data: Communication Nets



Internet

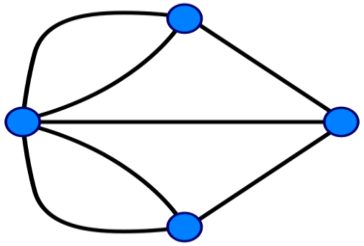
Graph Data: Technological Networks



Seven Bridges of Königsberg

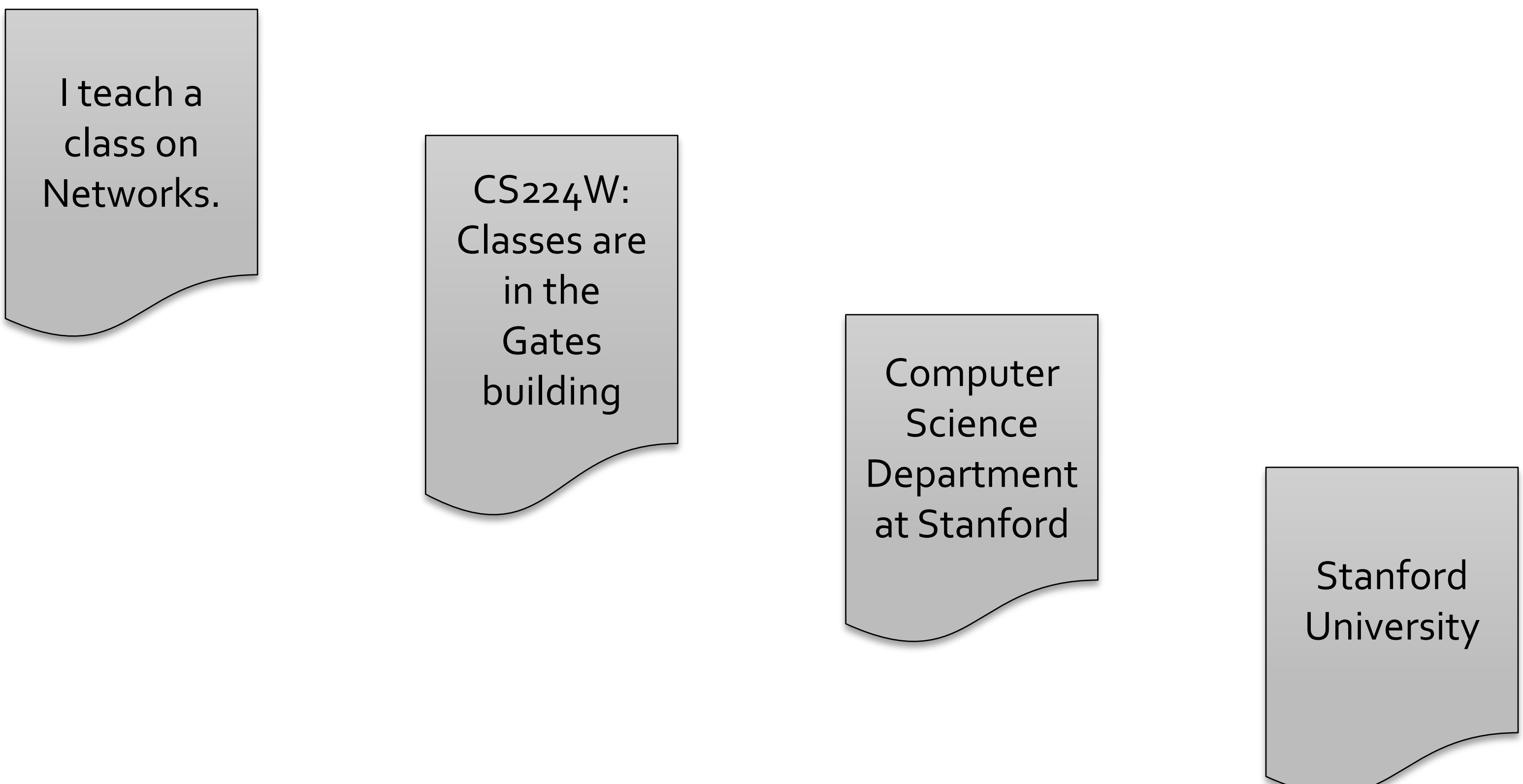
[Euler, 1735]

Return to the starting point by traveling each link of the graph once and only once.



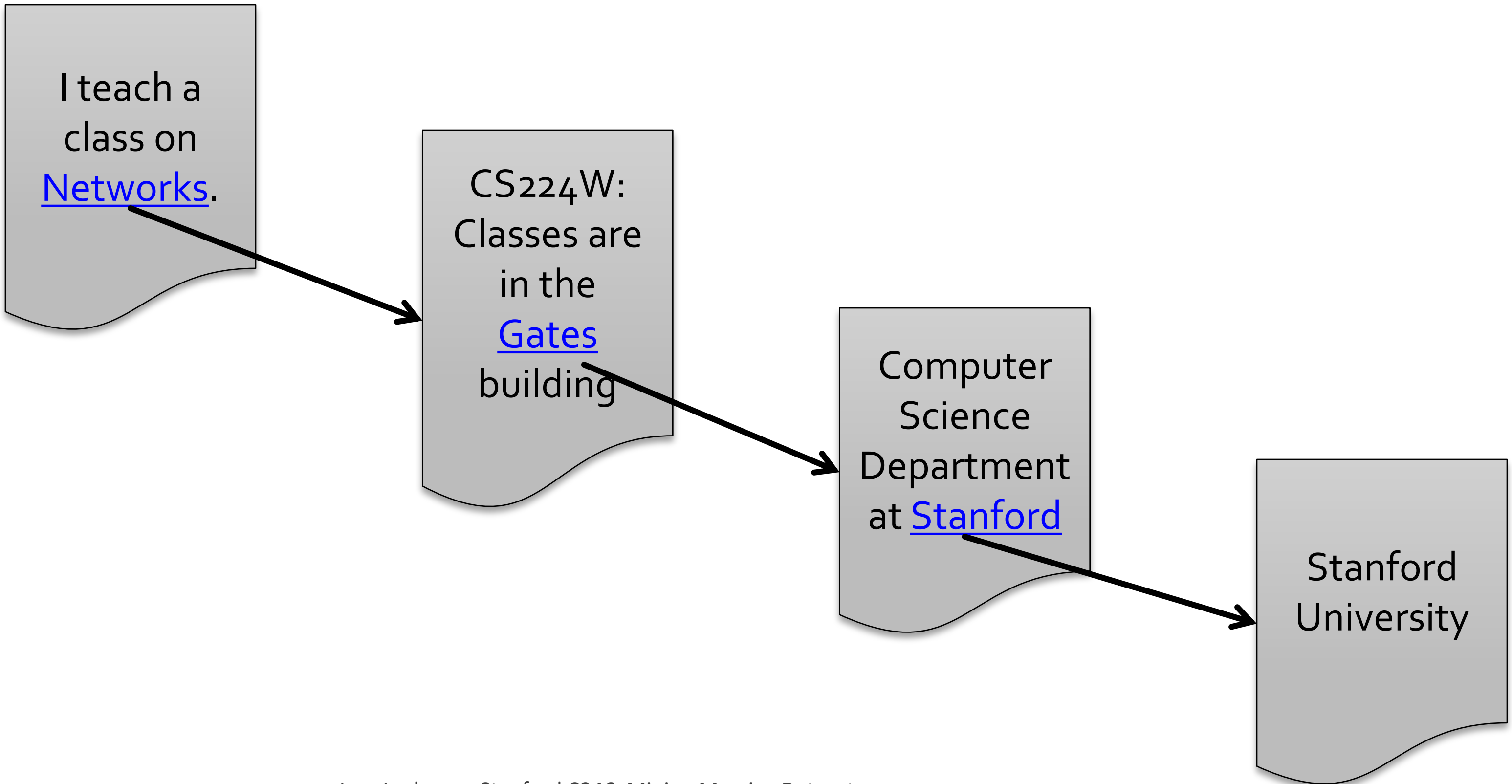
Web as a Graph

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

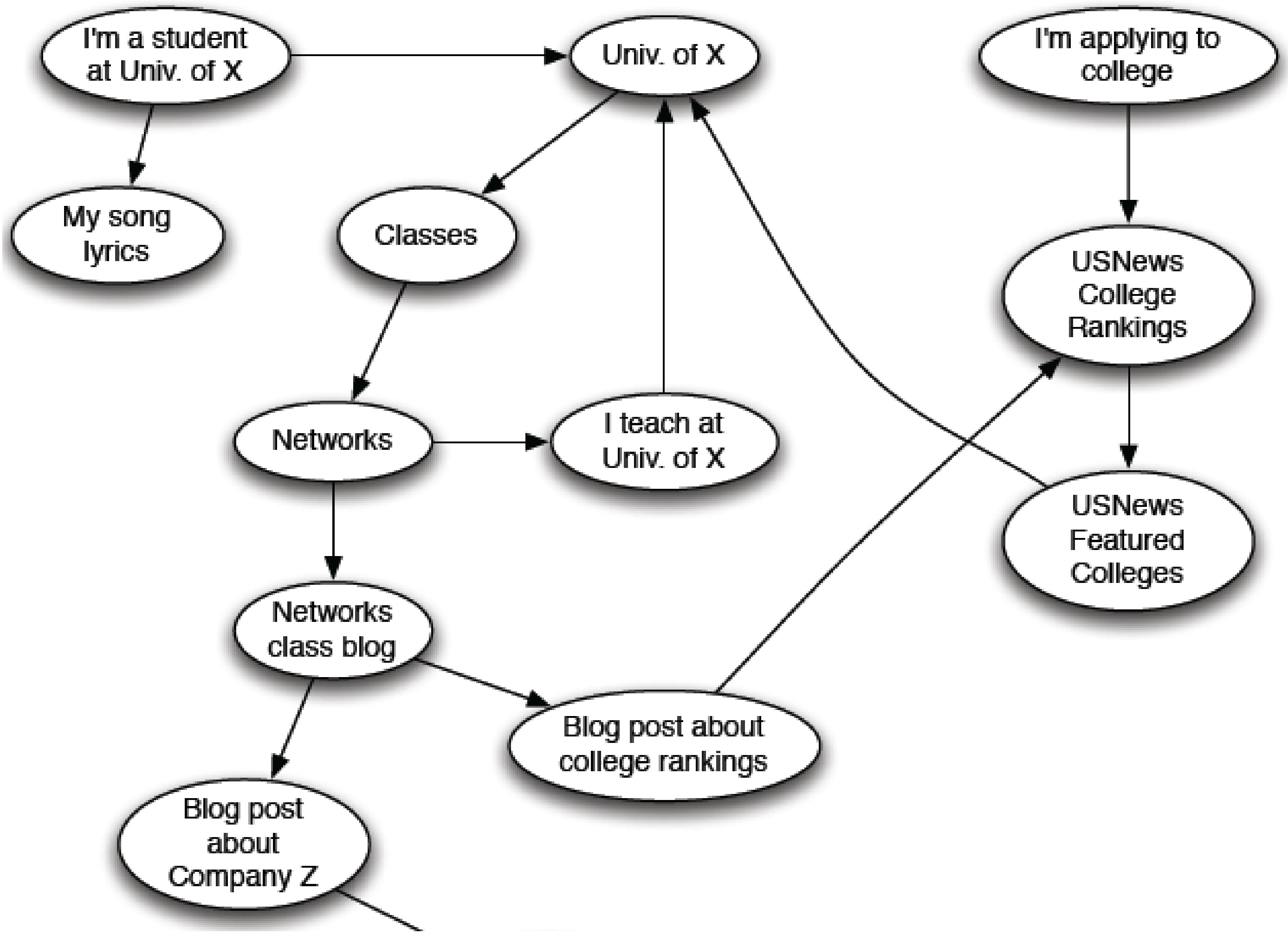


Web as a Graph

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



Web as a Directed Graph



Broad Question

- **How to organize the Web?**
- **First try:** Human curated **Web directories**
  - Yahoo, DMOZ, LookSmart
- **Second try: Web Search**
  - **Information Retrieval** investigates: Find relevant docs in a small and trusted set
    - Newspaper articles, Patents, etc.
  - **But:** Web is **huge**, full of untrusted documents, random things, web spam, etc.

Web Search: 2 Challenges

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

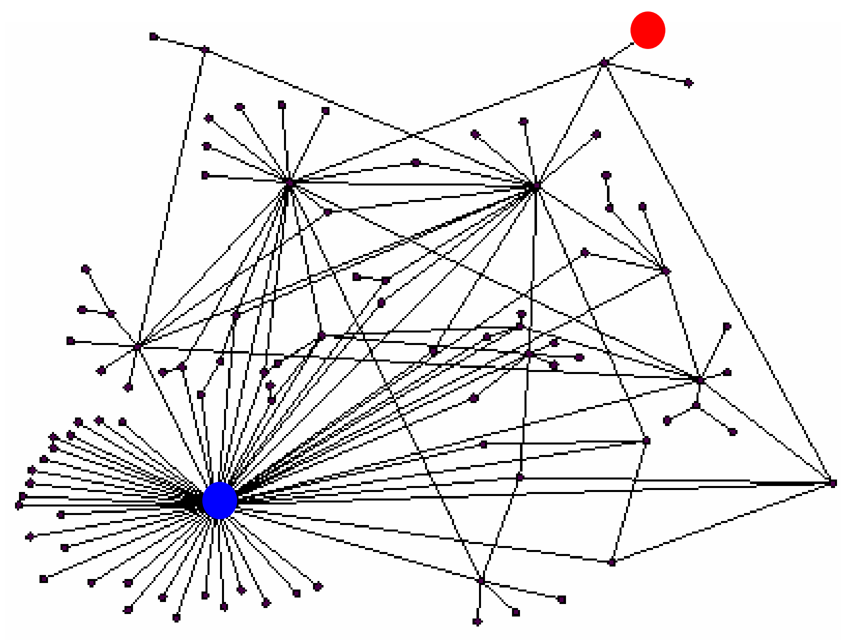
Ranking Nodes on the Graph

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

  www.joe-schmoe.com vs. www.stanford.edu

- There is large diversity in the web-graph node connectivity.

  Let's rank the pages by the link structure!



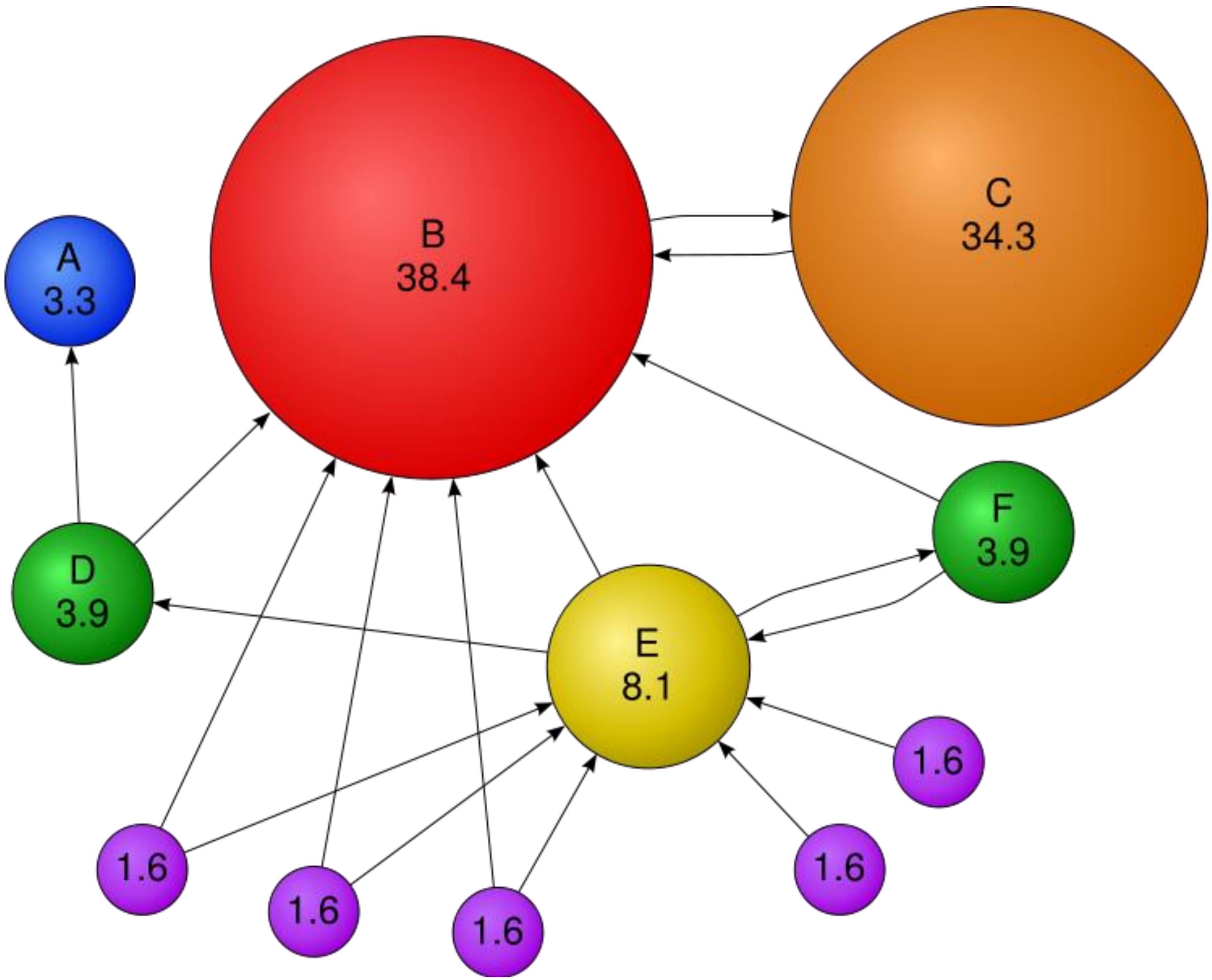
Link Analysis Algorithms

- 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

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!

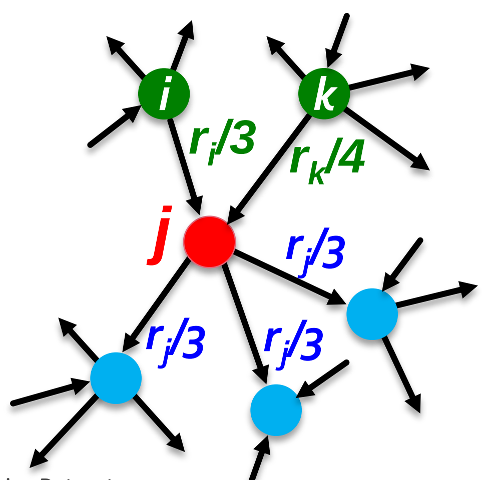
Example: PageRank Scores



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

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



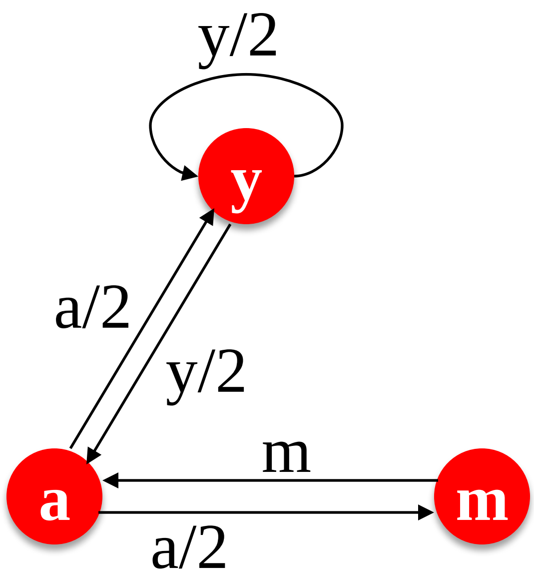
PageRank: The "Flow" Model

- **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

The web in 1839



"Flow" equations:

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

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:**
  - $\boldsymbol{r_y + r_a + r_m = 1}$
  - **Solution:** $\boldsymbol{r_y = \frac{2}{5},\ r_a = \frac{2}{5},\ r_m = \frac{1}{5}}$
- **Gaussian elimination method works for small examples, but we need a better method for large web-size graphs**
- **We need a new formulation!**

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**

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

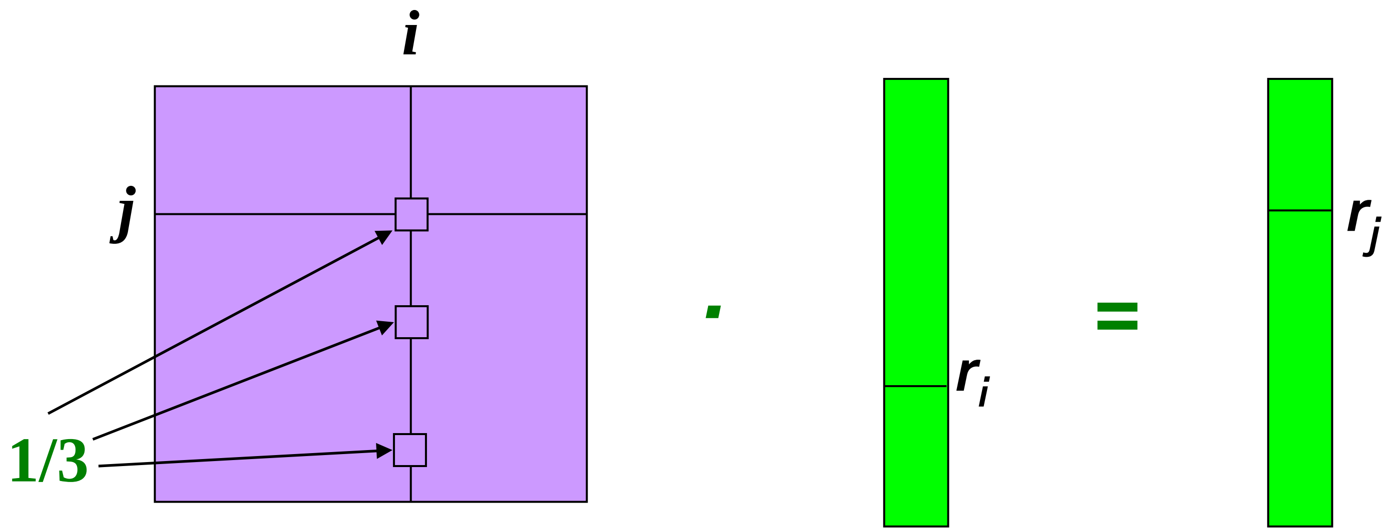
$$\boldsymbol{r} = \boldsymbol{M} \cdot \boldsymbol{r}$$

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***



$$M \cdot r = r$$

Eigenvector Formulation

- **The flow equations can be written**

$$r = M \cdot r$$

- 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
      - *We know **r** is unit length and each column of **M** sums to one, so* $Mr \le 1$

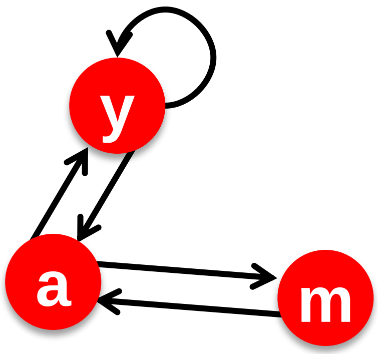
NOTE: x is an eigenvector with the corresponding eigenvalue **λ** if:

$$Ax = \lambda x$$

- **We can now efficiently solve for *r*!**

  The method is called Power iteration

Example: Flow Equations & M



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$$r = M \cdot r$$

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

$$\begin{bmatrix} y \\ a \\ m \end{bmatrix} = \begin{bmatrix} ½ & ½ & 0 \\ ½ & 0 & 1 \\ 0 & ½ & 0 \end{bmatrix} \begin{bmatrix} y \\ a \\ m \end{bmatrix}$$

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 $|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}|_1 < \varepsilon$
    - $|\mathbf{x}|_1 = \sum_{1\le i\le N}|x_i|$ is the L_1 norm

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

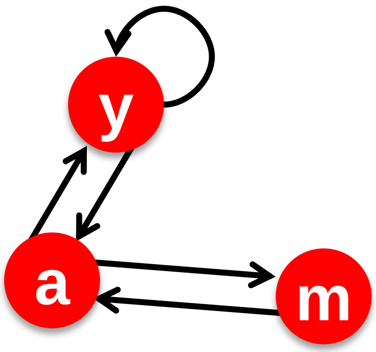
d_i …. out-degree of node i

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:**

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

Iteration 0, 1, 2, …

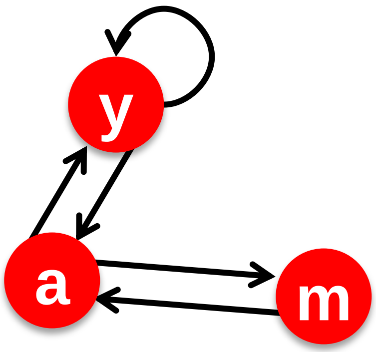


	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$\mathbf{r_y = r_y/2 + r_a/2}$
$\mathbf{r_a = r_y/2 + r_m}$
$\mathbf{r_m = r_a/2}$

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**



	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:**

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

Iteration 0, 1, 2, …

Why Power Iteration works? (1)

- **Power iteration:**
  A method for finding dominant eigenvector (the vector corresponding to the largest eigenvalue)
  - $\boldsymbol{r}^{(1)} = \boldsymbol{M} \cdot \boldsymbol{r}^{(0)}$
  - $\boldsymbol{r}^{(2)} = \boldsymbol{M} \cdot \boldsymbol{r}^{(1)} = \boldsymbol{M}\left(\boldsymbol{M}\boldsymbol{r}^{(1)}\right) = \boldsymbol{M}^{2} \cdot \boldsymbol{r}^{(0)}$
  - $\boldsymbol{r}^{(3)} = \boldsymbol{M} \cdot \boldsymbol{r}^{(2)} = \boldsymbol{M}\left(\boldsymbol{M}^{2}\boldsymbol{r}^{(0)}\right) = \boldsymbol{M}^{3} \cdot \boldsymbol{r}^{(0)}$
- **Claim:**
  Sequence $\boldsymbol{M} \cdot \boldsymbol{r}^{(0)}, \boldsymbol{M}^{2} \cdot \boldsymbol{r}^{(0)}, \ldots \boldsymbol{M}^{k} \cdot \boldsymbol{r}^{(0)}, \ldots$
  approaches the dominant eigenvector of $\boldsymbol{M}$

Why Power Iteration works? (2)

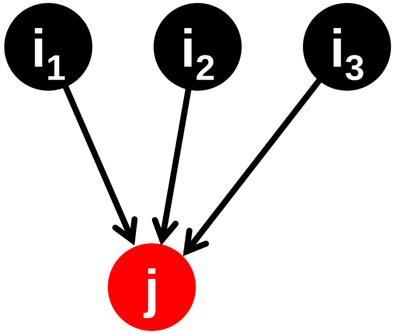
- **Claim:** Sequence $\boldsymbol{M \cdot r^{(0)}, M^2 \cdot r^{(0)}, \ldots M^k \cdot r^{(0)}, \ldots}$ approaches the dominant eigenvector of $\boldsymbol{M}$
- **Proof:**
  - Assume ***M*** has ***n*** linearly independent eigenvectors, $x_1, x_2, \ldots, x_n$ with corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, where $\lambda_1 > \lambda_2 > \cdots > \lambda_n$
  - Vectors $x_1, x_2, \ldots, x_n$ form a basis and thus we can write: $r^{(0)} = c_1\, x_1 + c_2\, x_2 + \cdots + c_n\, x_n$
  - $$\begin{aligned} \boldsymbol{Mr^{(0)}} &= \boldsymbol{M(c_1\, x_1 + c_2\, x_2 + \cdots + c_n\, x_n)} \\ &= c_1(Mx_1) + c_2(Mx_2) + \cdots + c_n(Mx_n) \\ &= c_1(\lambda_1 x_1) + c_2(\lambda_2 x_2) + \cdots + c_n(\lambda_n x_n) \end{aligned}$$
  - **Repeated multiplication on both sides produces** $$M^k r^{(0)} = c_1(\lambda_1^k x_1) + c_2(\lambda_2^k x_2) + \cdots + c_n(\lambda_n^k x_n)$$

Why Power Iteration works? (3)

- **Claim:** Sequence $\boldsymbol{M} \cdot \boldsymbol{r}^{(0)}, \boldsymbol{M}^{2} \cdot \boldsymbol{r}^{(0)}, \ldots \boldsymbol{M}^{k} \cdot \boldsymbol{r}^{(0)}, \ldots$ approaches the dominant eigenvector of $\boldsymbol{M}$
- **Proof (continued):**
  - Repeated multiplication on both sides produces $M^k r^{(0)} = c_1(\lambda_1^k x_1) + c_2(\lambda_2^k x_2) + \cdots + c_n(\lambda_n^k x_n)$
  - $M^k r^{(0)} = \lambda_1^k \left[c_1 x_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k x_2 + \cdots + c_n \left(\frac{\lambda_2}{\lambda_1}\right)^k x_n \right]$
  - Since $\lambda_1 > \lambda_2$ then fractions $\frac{\lambda_2}{\lambda_1}, \frac{\lambda_3}{\lambda_1} \ldots < 1$ and so $\left(\frac{\lambda_i}{\lambda_1}\right)^k = 0$ as $k \to \infty$ (for all $i = 2 \ldots n$).
  - **Thus:** $\boldsymbol{M^k r^{(0)} \approx c_1\left(\lambda_1^k x_1\right)}$
    - Note if $c_1 = 0$ then the method won't converge

Random Walk Interpretation

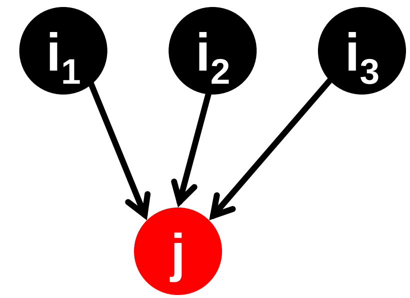
- **Imagine a random web surfer:**
  - At any time t, surfer is on some page i
  - At time $t+1$, the surfer follows an out-link from i uniformly at random
  - Ends up on some page j linked from i
  - Process repeats indefinitely
- **Let:**
  - $\boldsymbol{p}(\boldsymbol{t})$ … vector whose i^{th} coordinate is the prob. that the surfer is at page i at time t
  - So, $\boldsymbol{p}(\boldsymbol{t})$ is a probability distribution over pages



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

The Stationary Distribution

- **Where is the surfer at time $t+1$?**
  - Follows a link uniformly at random
    $$p(t+1) = M \cdot p(t)$$



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

- Suppose the random walk reaches a state
  $$p(t+1) = M \cdot p(t) = p(t)$$
  then $\boldsymbol{p}(t)$ is stationary distribution of a random walk
- Our original rank vector $\boldsymbol{r}$ satisfies $\boldsymbol{r} = \boldsymbol{M} \cdot \boldsymbol{r}$
  - **So, r is a stationary distribution for the random walk**

Announcement: We graded HW0 and HW1!

- Stanford students: Pick them up from the submission box in Gates
- SCPD students: SCPD will send you the HW

PageRank: 3 Questions

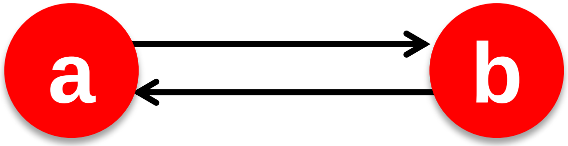
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}} \quad \text{or equivalently} \quad r = Mr$$

Does this converge?

Does it converge to what we want?

Are results reasonable?

Does this converge?



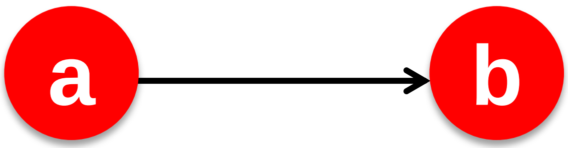
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

- **Example:**

$$\begin{matrix} r_a \\ r_b \end{matrix} = \begin{matrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{matrix}$$

Iteration 0, 1, 2, …

Does it converge to what we want?



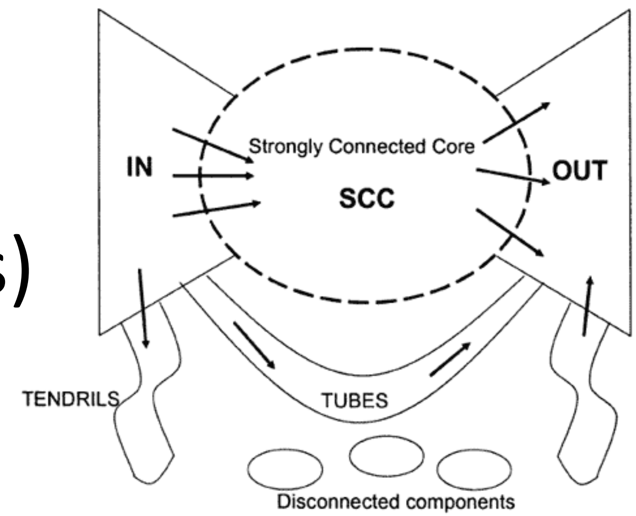
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d}_i}$$

- **Example:**

r_a	=	1	0	0	0
r_b		0	1	0	0

Iteration 0, 1, 2, …

RageRank: Problems



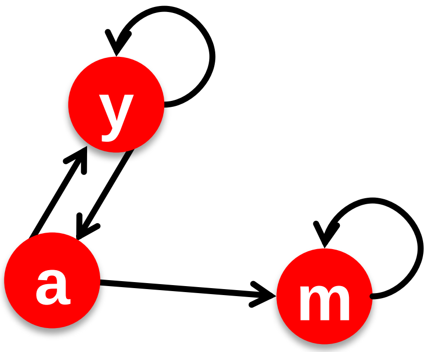
2 problems:

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

- **(2)** **Spider traps** (all out-links are within the group)
  - Eventually spider traps absorb all importance

Problem: Spider Traps

- **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	1

$\mathbf{r_y = r_y/2 + r_a/2}$

$\mathbf{r_a = r_y/2}$

$\mathbf{r_m = r_a/2 + r_m}$

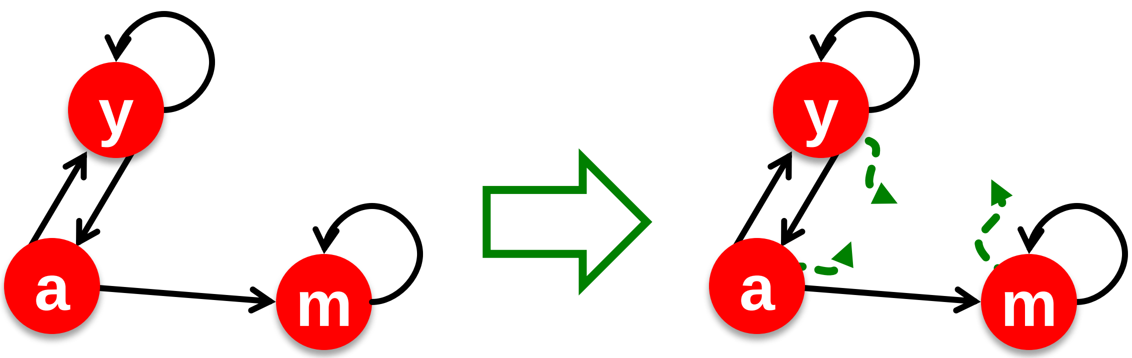
- **Example:**

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

Iteration 0, 1, 2, …

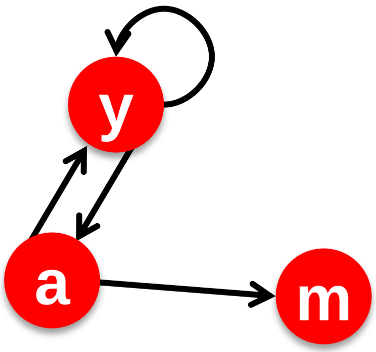
Solution: Random Teleports

- **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**



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

$\mathbf{r_y = r_y/2 + r_a/2}$

$\mathbf{r_a = r_y/2}$

$\mathbf{r_m = r_a/2}$

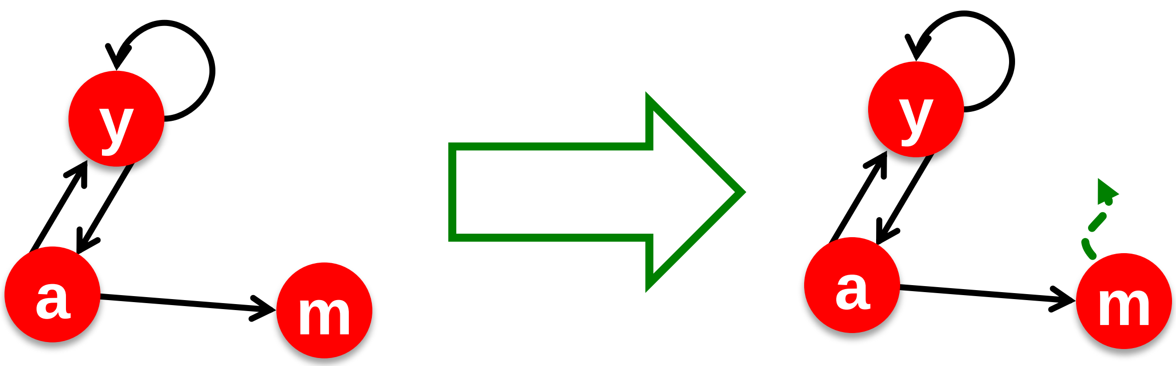
- **Example:**

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

Iteration 0, 1, 2, …

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

Why Teleports Solve the Problem?

$$r^{(t+1)} = Mr^{(t)}$$

Markov chains

- Set of states ***X***
- Transition matrix ***P*** where $\boldsymbol{P_{ij} = P(X_t = i \mid X_{t-1} = j)}$
- **π** specifying the stationary probability of being at each state $\boldsymbol{x \in X}$
- Goal is to find **π** such that **π = P π**

Why is This Analogy Useful?

- **Theory of Markov chains**

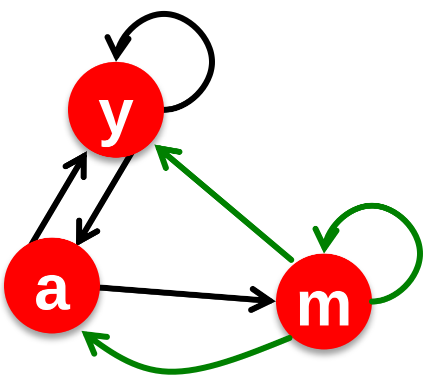
- **Fact:** For any start vector, the power method applied to a Markov transition matrix ***P*** will converge to a unique positive stationary vector as long as ***P*** is **stochastic**, **irreducible** and **aperiodic**.

Make M Stochastic

- **Stochastic:** Every column sums to 1
- **A possible solution:** Add green links

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

- a_i…=1 if node *i* has out deg 0, =0 else
- **e**…vector of all 1s

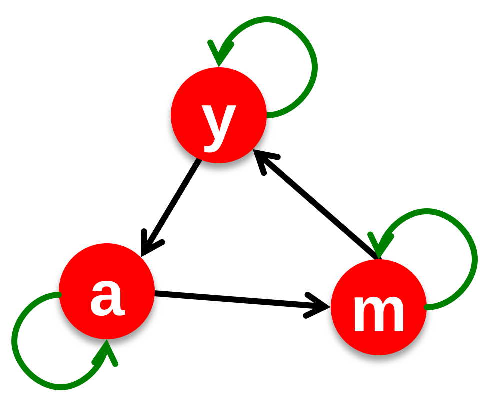


	y	a	m
y	½	½	1/3
a	½	0	1/3
m	0	½	1/3

$r_y = r_y/2 + r_a/2 + r_m/3$
$r_a = r_y/2 + r_m/3$
$r_m = r_a/2 + r_m/3$

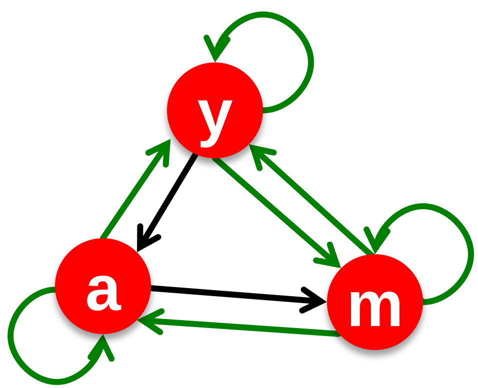
Make M Aperiodic

- 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.
- **A possible 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
- **A possible 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** [Brin-Page, 98]

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

d_i ... out-degree of node i

This formulation assumes that M has no dead ends. We can either preprocess matrix M to remove all dead ends or explicitly follow random teleport links with probability 1.0 from dead-ends.

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*:**

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

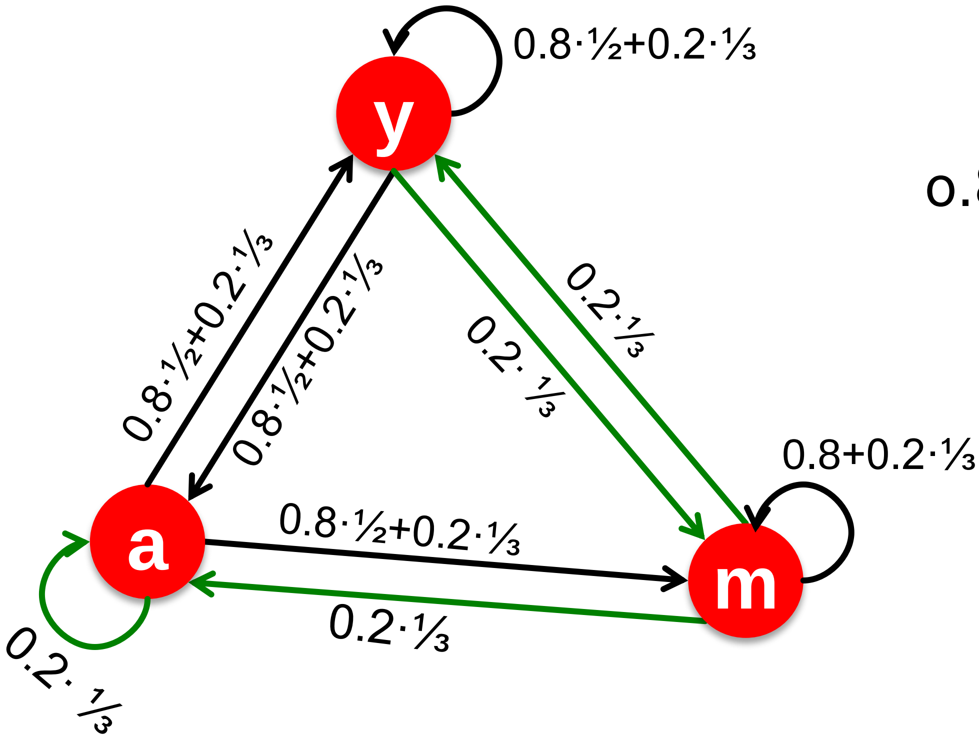
e…vector of all 1s

- ***A* is stochastic, aperiodic and irreducible, so**

$$\boldsymbol{r}^{(t+1)} = \boldsymbol{A}\cdot\boldsymbol{r}^{(t)}$$

- **What is β ?**
  - In practice β =0.8,0.9 (make 5 steps and jump)

Random Teleports ($\beta = 0.8$)



$$
0.8 \overset{\textbf{M}}{\boxed{\begin{matrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{matrix}}} + 0.2 \overset{\mathbf{1/n \cdot 1 \cdot 1^T}}{\boxed{\begin{matrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{matrix}}}
$$

$$
\begin{array}{c} y \\ a \\ m \end{array} \underset{\textbf{A}}{\boxed{\begin{matrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{matrix}}}
$$

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

How do we actually compute the PageRank?

Computing Page Rank

- **Key step is matrix-vector multiplication**
  - $\boldsymbol{r}^{new} = \boldsymbol{A} \cdot \boldsymbol{r}^{old}$
- Easy if we have enough main memory to hold **A**, $\mathbf{r}^{old}$, $\mathbf{r}^{new}$
- **Say N = 1 billion pages**
  - We need 4 bytes for each entry (say)
  - 2 billion entries for vectors, approx 8GB
  - Matrix **A** has N^2 entries
    - 10^{18} is a large number!

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

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

$$= \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}$$

Matrix Formulation

- Suppose there are N pages
- Consider page j, with $\mathbf{d}_j$ out-links
- We have $M_{ij} = 1/|d_j|$ when $j \longrightarrow i$
  and $M_{ij} = 0$ otherwise
- **The random teleport is equivalent to:**
  - Adding a **teleport link** from j to every other page and setting transition probability to $(1-\beta)/N$
  - Reducing the probability of following each out-link from $1/|d_j|$ to $\beta/|d_j|$
  - **Equivalent:** Tax each page a fraction $(1-\beta)$ of its score and redistribute evenly

Rearranging the Equation

- $\boldsymbol{r} = \boldsymbol{A} \cdot \boldsymbol{r},$ **where** $\boldsymbol{A_{ij} = \beta\, M_{ij} + \frac{1-\beta}{N}}$
- $r_i = \sum_{j=1}^{N} A_{ij} \cdot r_j$
- $r_i = \sum_{j=1}^{N} \left[\beta\, M_{ij} + \frac{1-\beta}{N}\right] \cdot r_j$

$$= \sum_{j=1}^{N} \beta\, M_{ij} \cdot r_j + \frac{1-\beta}{N} \sum_{j=1}^{N} r_j$$

$$= \sum_{j=1}^{N} \beta\, M_{ij} \cdot r_j + \frac{1-\beta}{N} \qquad \text{since } \sum r_j = 1$$

- **So we get:** $\boldsymbol{r = \beta\, M \cdot r + \left[\frac{1-\beta}{N}\right]_N}$

Note: Here we assumed **M** has no dead-ends.

[x]$_N$ … a vector of length *N* with all entries *x*

Sparse Matrix Formulation

- We just rearranged the **PageRank equation**

$$r = \beta M \cdot r + \left[\frac{1-\beta}{N}\right]_N$$

  - where $[(1-\beta)/N]_N$ is a vector with all N entries $(1-\beta)/N$
- ***M*** is a **sparse matrix!** (with no dead-ends)
  - 10 links per node, approx 10N entries
- So in each iteration, we need to:
  - Compute $r^{new} = \beta M \cdot r^{old}$
  - Add a constant value $(1-\beta)/N$ to each entry in r^{new}
    - **Note if M contains dead-ends then $\sum_i r_i^{new} < 1$ and we also have to renormalize r^{new} so that it sums to 1**

PageRank: The Complete Algorithm

- **Input: Graph G and parameter β**
  - Directed graph G with **spider traps** and **dead ends**
  - Parameter β
- **Output: PageRank vector r**
  - **Set:** $r_j^{(0)} = \frac{1}{N}, \quad t = 1$
  - **do:**
    - $\forall j: \boldsymbol{r'^{(t)}_j = \sum_{i \to j} \beta \frac{r_i^{(t-1)}}{d_i}}$

      $\boldsymbol{r'^{(t)}_j = 0}$ if in-deg. of $\boldsymbol{j}$ is **0**
    - **Now re-insert the leaked PageRank:**

      $\forall \boldsymbol{j}: \boldsymbol{r_j^{(t)} = r'^{(t)}_j + \frac{1-S}{N}}$ **where:** $S = \sum_j r'^{(t)}_j$
    - $\boldsymbol{t = t + 1}$
  - **while** $\sum_j \left| r_j^{(t)} - r_j^{(t-1)} \right| > \varepsilon$

Sparse Matrix Encoding

- Encode sparse matrix using only nonzero entries
  - Space proportional roughly to number of links
  - Say 10N, or 4*10*1 billion = 40GB
  - **Still won't fit in memory, but will fit on disk**

source node	degree	destination nodes
0	3	1, 5, 7
1	5	17, 64, 113, 117, 245
2	2	13, 23

Basic Algorithm: Update Step

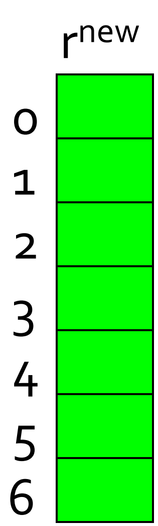
- Assume enough RAM to fit $\boldsymbol{r}^{new}$ into memory
  - Store $\boldsymbol{r}^{old}$ and matrix **M** on disk
- **Then 1 step of power-iteration is:**

Initialize all entries of r^{new} to $(1-\beta)/N$

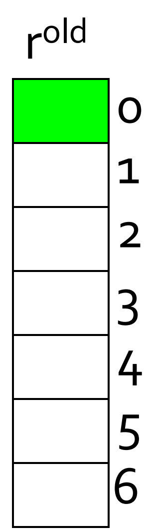
For each page p (of out-degree n):

Read into memory: p, n, $dest_1, \ldots, dest_n$, $r^{old}(p)$

for j = 1…n: $r^{new}(dest_j) \mathrel{+}= \beta\, r^{old}(p) / n$



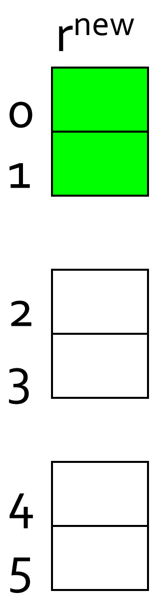
src	degree	destination
0	3	1, 5, 6
1	4	17, 64, 113, 117
2	2	13, 23



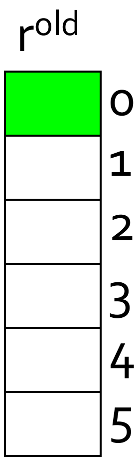
Analysis

- Assume enough RAM to fit $\boldsymbol{r}^{new}$ into memory
  - Store $\boldsymbol{r}^{old}$ and matrix **M** on disk
- In each iteration, we have to:
  - Read $\boldsymbol{r}^{old}$ and **M**
  - Write $\boldsymbol{r}^{new}$ back to disk
  - IO cost = $2|\boldsymbol{r}| + |\boldsymbol{M}|$
- **Question:**
  - What if we could not even fit $\mathbf{r}^{new}$ in memory?

Block-based Update Algorithm



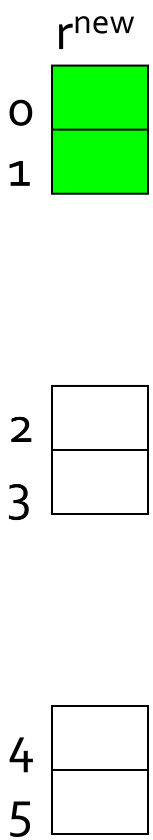
src	degree	destination
0	4	0, 1, 3, 5
1	2	0, 5
2	2	3, 4



Analysis of Block Update

- Similar to nested-loop join in databases
  - Break $\boldsymbol{r}^{\text{new}}$ into k blocks that fit in memory
  - Scan $\boldsymbol{M}$ and $\boldsymbol{r}^{\text{old}}$ once for each block
- k scans of $\boldsymbol{M}$ and $\boldsymbol{r}^{\text{old}}$
  - $k(|\boldsymbol{M}| + |\boldsymbol{r}|) + |\boldsymbol{r}| = k|\boldsymbol{M}| + (k+1)|\boldsymbol{r}|$
- **Can we do better?**
  - Hint: $\boldsymbol{M}$ is much bigger than $\boldsymbol{r}$ (approx 10-20x), so we must avoid reading it k times per iteration

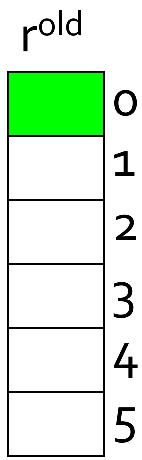
Block-Stripe Update Algorithm



src	degree	destination
0	4	0, 1
1	3	0
2	2	1

0	4	3
2	2	3

0	4	5
1	3	5
2	2	4



Block-Stripe Analysis

- Break $\boldsymbol{M}$ into stripes
  - Each stripe contains only destination nodes in the corresponding block of $\boldsymbol{r}^{\text{new}}$
- Some additional overhead per stripe
  - But it is usually worth it
- Cost per iteration
  - $|\boldsymbol{M}|(1+\varepsilon) + (k+1)|\boldsymbol{r}|$

Some Problems with Page Rank

- Measures generic popularity of a page
  - Biased against topic-specific authorities
  - **Solution:** Topic-Specific PageRank (next)
- Uses a single measure of importance
  - Other models e.g., hubs-and-authorities
  - **Solution:** Hubs-and-Authorities (next)
- Susceptible to Link spam
  - Artificial link topographies created in order to boost page rank
  - **Solution:** TrustRank (next)