Analysis of Large Graphs: Link Analysis, PageRank

Mining of Massive Datasets
Jure Leskovec, Anand Rajaraman, Jeff Ullman
Stanford University
http://www.mmds.org

New Topic: Graph Data!

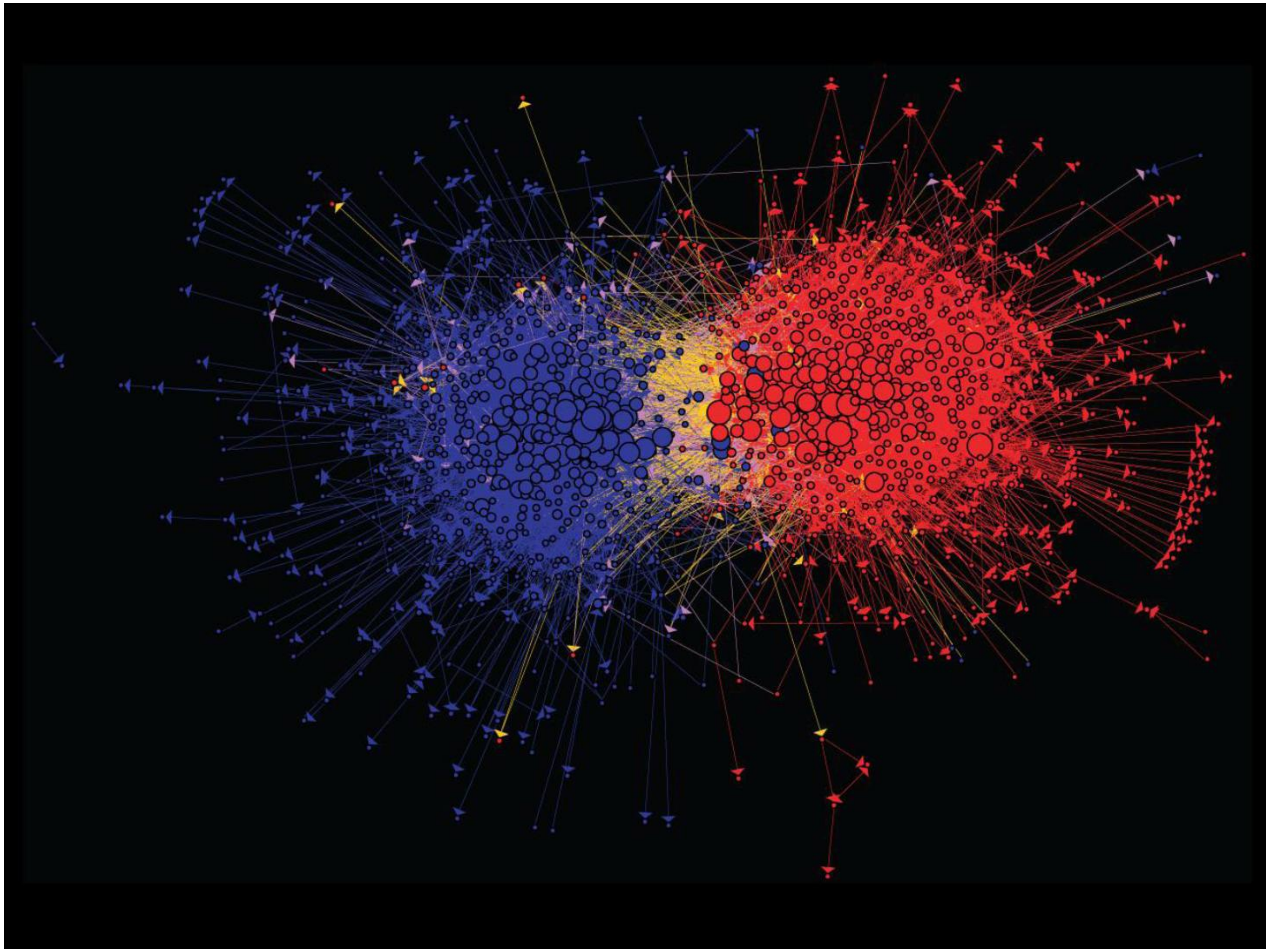
High dim. data	Graph data	Infinite data	Machine learning	Apps
Locality sensitive hashing	**PageRank,** SimRank	Filtering data streams	SVM	Recommen der systems
Clustering	Community Detection	Web advertising	Decision Trees	Association Rules
Dimensional ity reduction	Spam Detection	Queries on streams	Perceptron,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

Graph Data: Social Networks

Facebook social graph

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

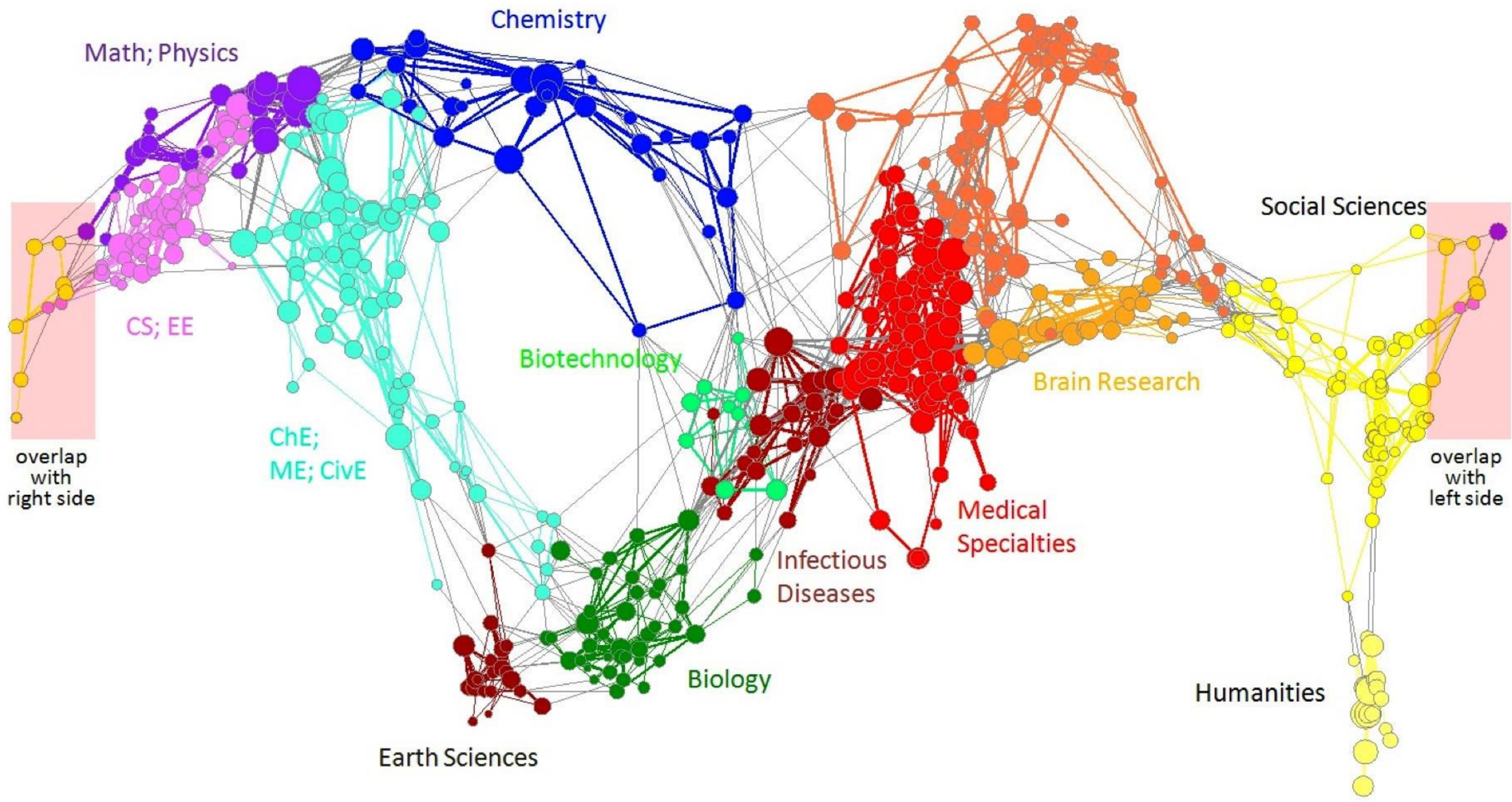
Graph Data: Media Networks



Connections between political blogs

Polarization of the network [Adamic-Glance, 2005]

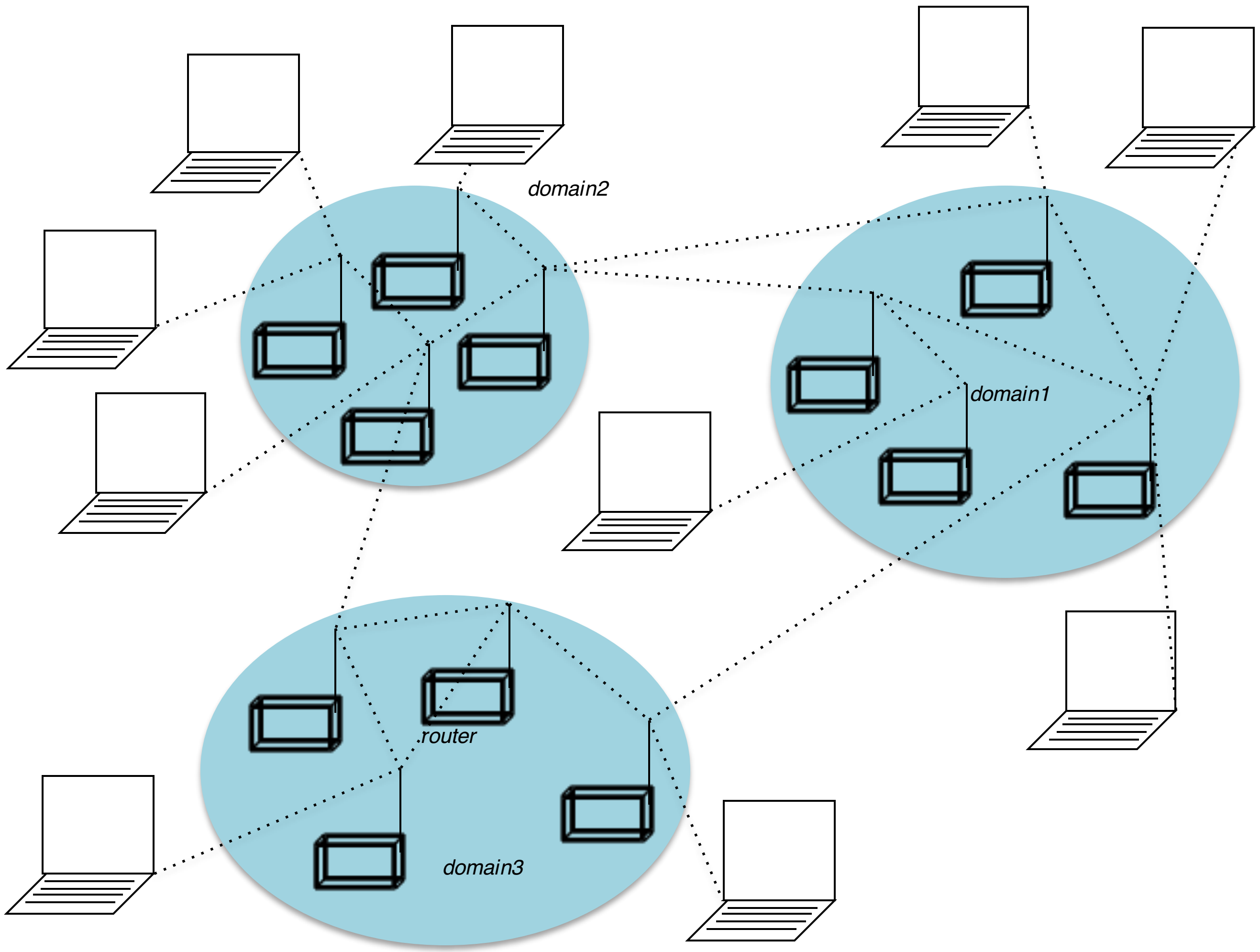
Graph Data: Information Nets



Citation networks and Maps of science

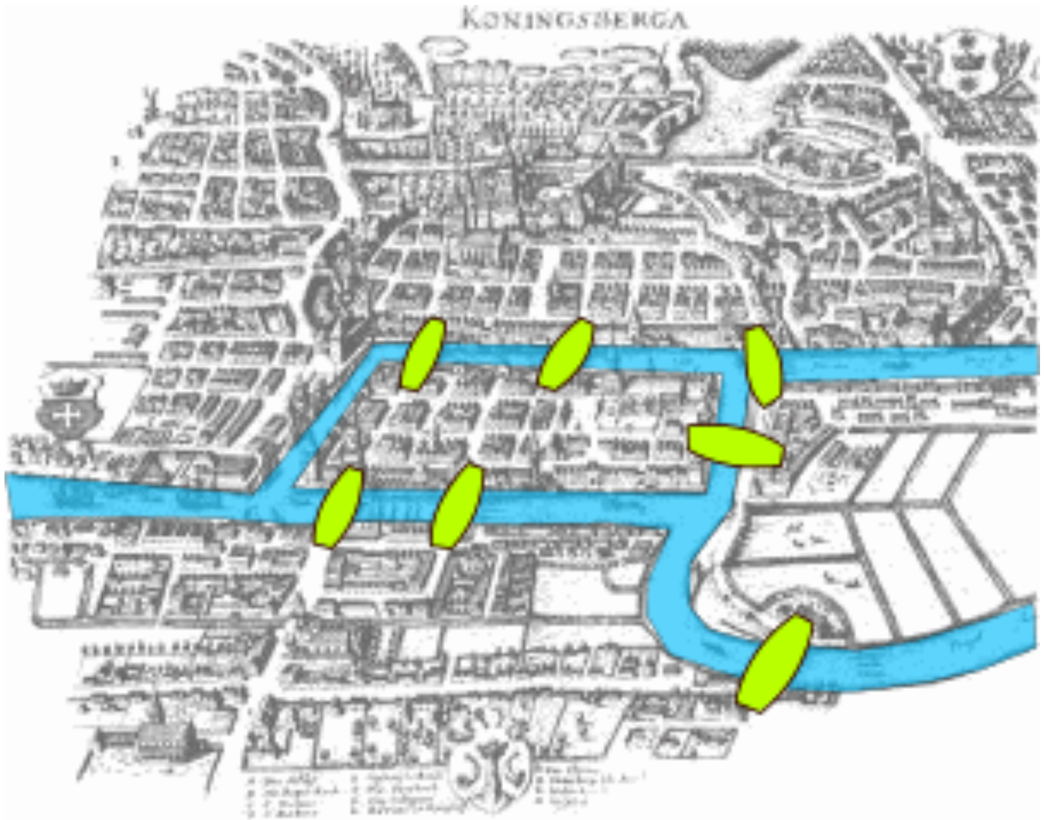
[Börner et al., 2012]

Graph Data: Communication Nets



Internet

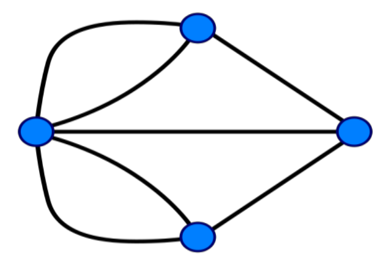
Graph Data: Technological Networks



Seven Bridges of Königsberg

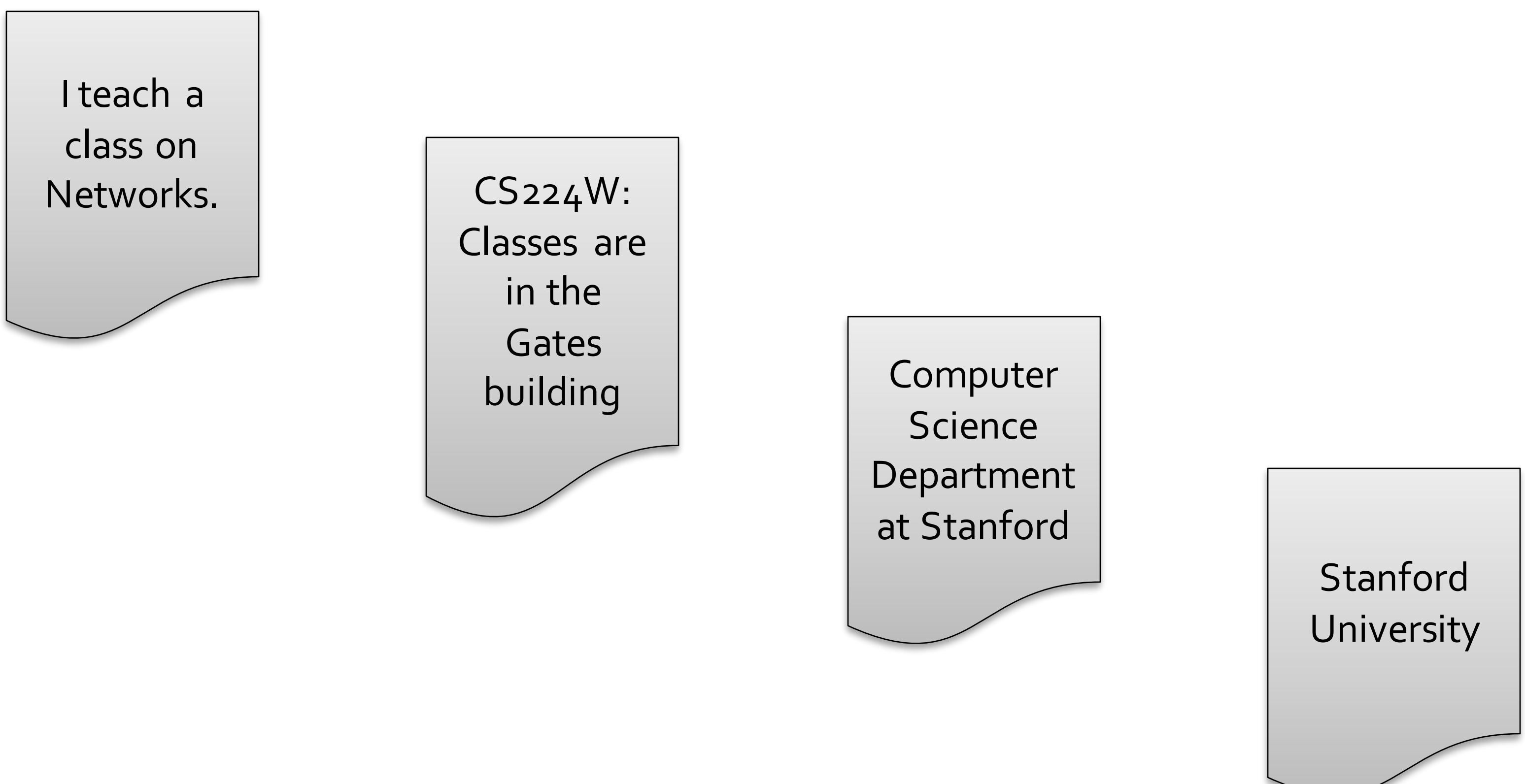
[Euler, 1735]

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



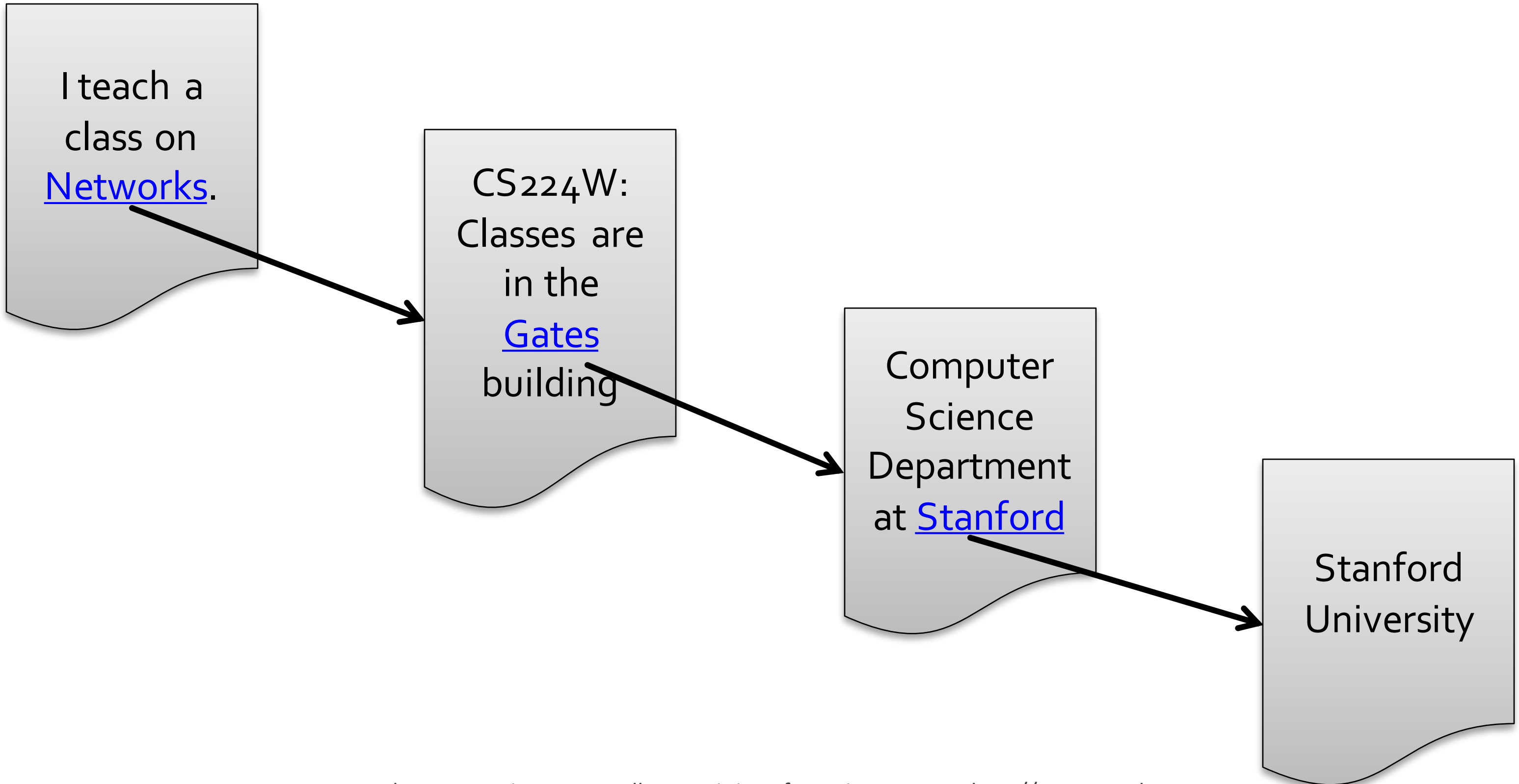
Web as a Graph

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

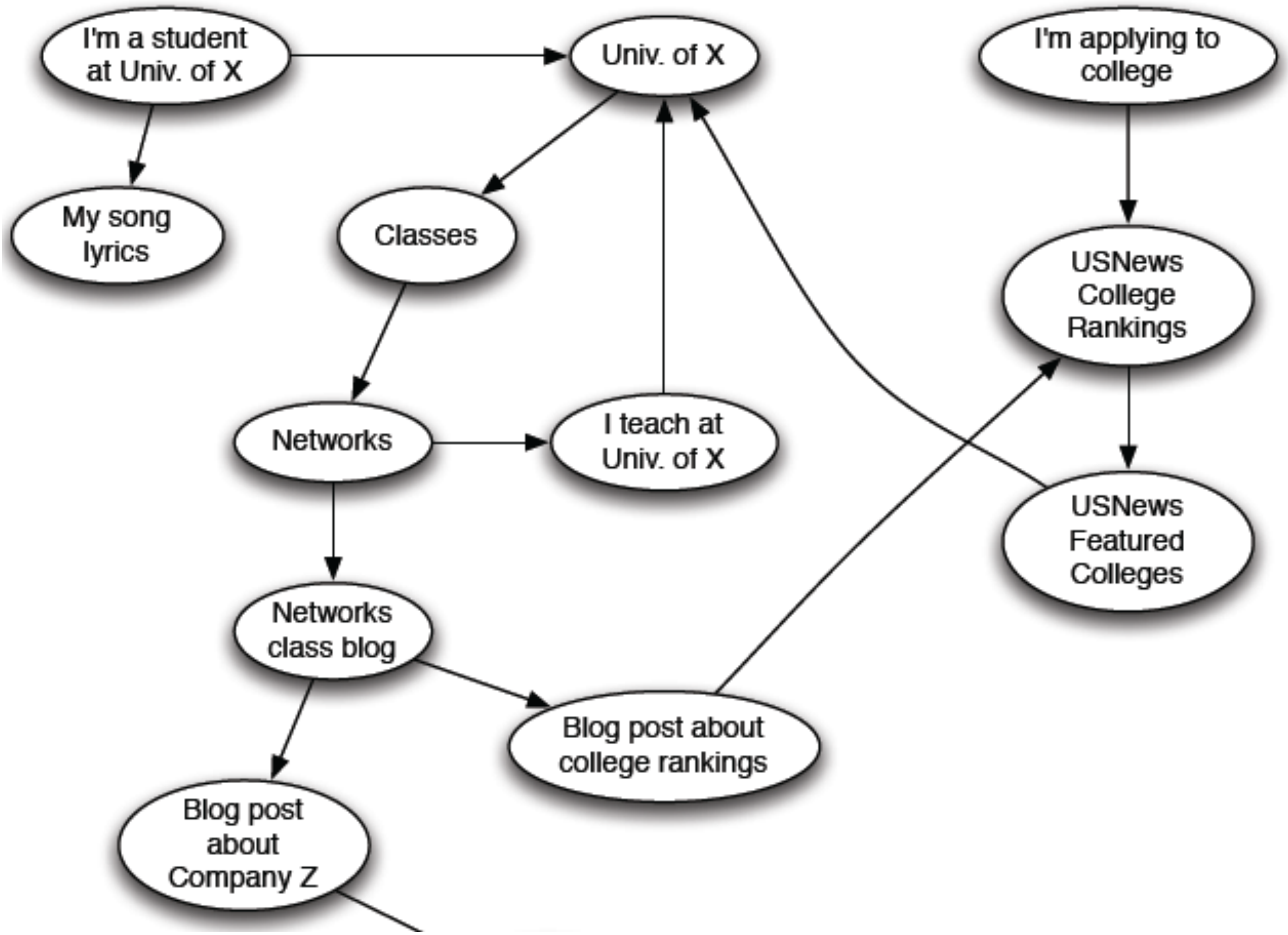


Web as a Graph

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



Web as a Directed Graph



SHORT HISTORY of INFORMATION RETRIEVAL

- **Classic Document Collections:** Hierarchical Indexing, Libraries
- **1° Digital Revolution:** Centralized Database Systems and Search Systems (Programming Languages for Queries, SQL)
- **2° Digital Revolution (1989): The World Wide Web:**
  - HTML Pages, Hyperlinks. Decentralized Information System.
  - IR on the WEB: **WEB Search Engines** and the **Link Analysis**

Information Retrieval on WWW

Main Differences and New Challenges in WWW IR:

- Huge Size
- Evolving
- Self-Organized & Distributed (no standard rules)
- Hyperlinked

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**, dynamics, full of untrusted documents, random things, web spam, etc.

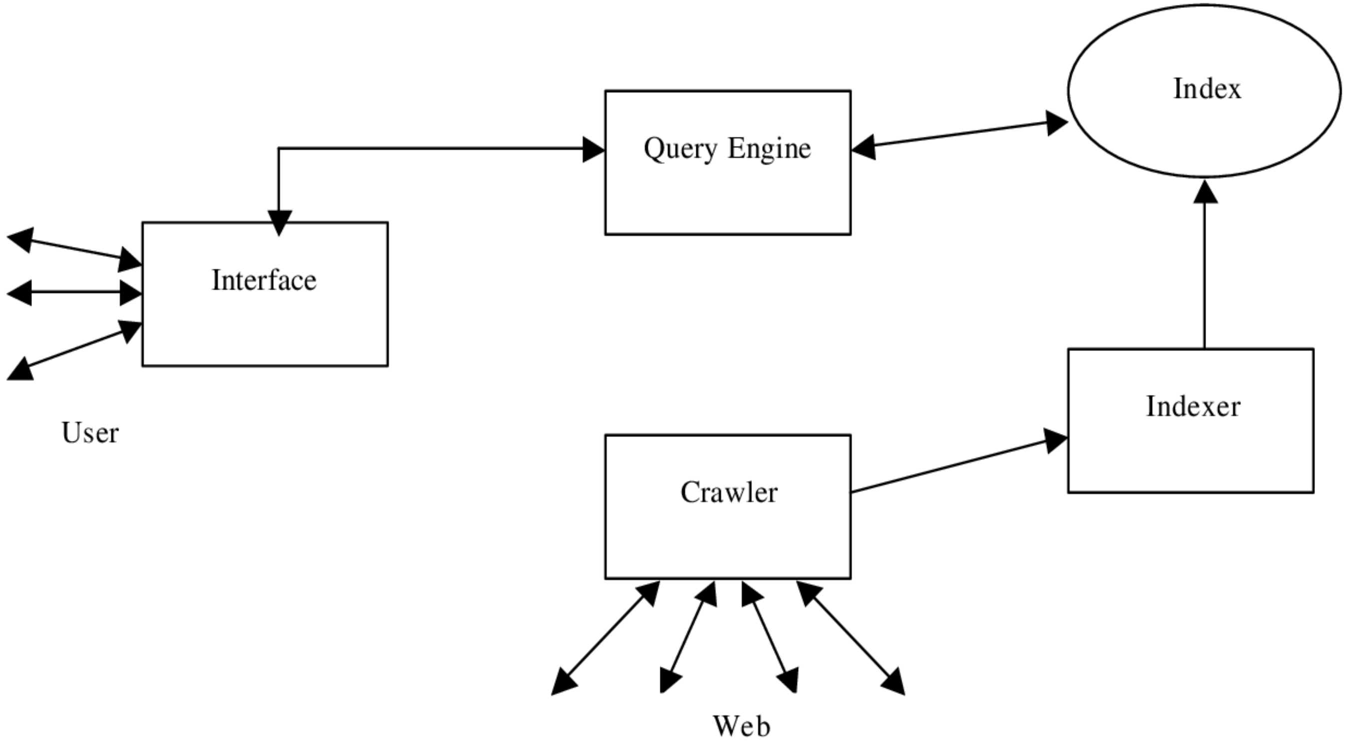
WWW Search Engines

- **MAIN TASK:** Given a set *S* of words, find the ***most relevant*** Web Pages for *S*

- The term «***most relevant***» *hides the most challenging aspect of **WWW IR**: How to select the first **few** pages which are more relevant for S among millions of them?*

WWW Search Engines

- **WWW Search Engines** are complex systems formed by several SW modules (see for instance [Langville_Meyer_06])



Web Search: 2 Challenges

2 challenges of web search:

- **(1) Web contains many sources of information Who to "trust"? How to recover them?**
  - **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

WWW Search Engines: The Query Module

- **Query Module (QM):** it converts a user's natural language Query into a language the *WWW-Search Engine System* can understand (usually numbers), and consults the **Index Module**

- **QM** consults the **content index** and its **inverted file** to select a set ***P*** of pages that contain the Query terms T, i.e.,

  P **:= { Relevant Pages for T }**

- Then, **QM** passes ***P*** to the ***Ranking Module***

WWW Search Engines: The Ranking Module

- **Ranking Module:**
  - **(i)** Compute and Assign an **Overall Score** to every **page p** in **P,**
  - **(ii)** Set **P** is then returned to the *User* in order of the **Overall Score**.
- **Overall Score** is based on **two scores** (computed by the Module):
- **Content Score:** it depends on several parameters: # query-words's occurrences in **p**; query-word's presence in the *title* or in *bold* in **p**.
- **Popularity Score** (**focus** of our course): it is determined from a *Link Analysis* of the **Web's hyperlink structure** (i.e. a Directed Graph).
- ***Note:*** *The Ranking Module is perhaps the* ***most important component*** *of the Search Process because the output of the query module often results in too many (thousands of) Relevant Pages that the User must sort through.*

Popularity Score & Link Analysis: The WWW IR Revolution

- «*Nobody wants to be picked last for teams in gym class. Likewise, nobody wants their webpage to appear last in the list of relevant pages for a search query. As a result, many grown-ups transfer their high school wishes to be the "Most Popular" to their webpages*»

- By 1998, the traditional **Content Score** was buckling under the **Web's massive size** and the **death grip of spammers**. In 1998, the **Popularity Score** came to the rescue of the **Content Score**. The **Popularity Score** became a crucial complement to the **Content Score** and provided *Search Engines* with **impressively accurate results** for **all types of queries**. The **Popularity Score**, also known as the *importance score*, harnesses the information in the immense Graph created by the **Web's hyperlink structure**. Thus, models exploiting the **Web's hyperlink structure** are called **Link-Analysis** models.

- **Note:** The impact that these link-analysis models have had is truly awesome. Since 1998, the use of *Web Search Engines* has increased dramatically. In fact, an April 2004 survey by *Websense, Inc*., reported that half of the respondents would rather forfeit their habitual morning cup of coffee than their connectivity. That's because today's *Search Tools* allow Users to answer in seconds queries that were impossible just a decade ago (from fun searches for pictures, quotes, and snooping amateur detective work to more serious searches for academic research papers and patented inventions).

The WWW IR Revolution: The Page-Rank Algorithm

Before 1998, the **Web Graph** was largely an *untapped* source of information. *While **Computer Science Researchers** like Kleinberg and Brin and Page* recognized this **Graph's Potential**, most people wondered just what the **Web Graph** had to do with *Search_Engine* results ☺☺.

Key Idea of the PR Algorithm: The connection is understood by viewing a hyperlink as a recommendation. A hyperlink from my homepage to your page is my **endorsement** of your page. Thus, a page with more recommendations (which are realized through in-links) must be more important than a page with a few in-links.

The Page-Rank Algorithm

Key Issue:

- Similar to other Recommendation Systems (biblio citations, letters of references), the **status** of the recommender is also important: one personal endorsement from B. Obhama probably does more to strengthen a job app than 20 endorsements from 20 unknown teachers and colleagues. On the other hand, if the job interviewer learns that B. Obhama is very generous with his praises of employees, and he (or his secretary) has written over 40,000 recommendations in his life, then his recommendation suddenly drops in weight. **Thus, weights signifying the status of a recommender must be lowered for recommenders with little discrimination. In fact, the weight of each endorsement should be tempered by the total number of recommendations made by the recommender**.

Actually, this is exactly how Google's PageRank popularity score works.

The Page-Rank Algorithm

Main Idea of PageRank Algorithm:

«a Web-Page is ***important*** if it is pointed to by other ***important*** Web-Pages»

Sounds *circular*, doesn't it? ☺☺

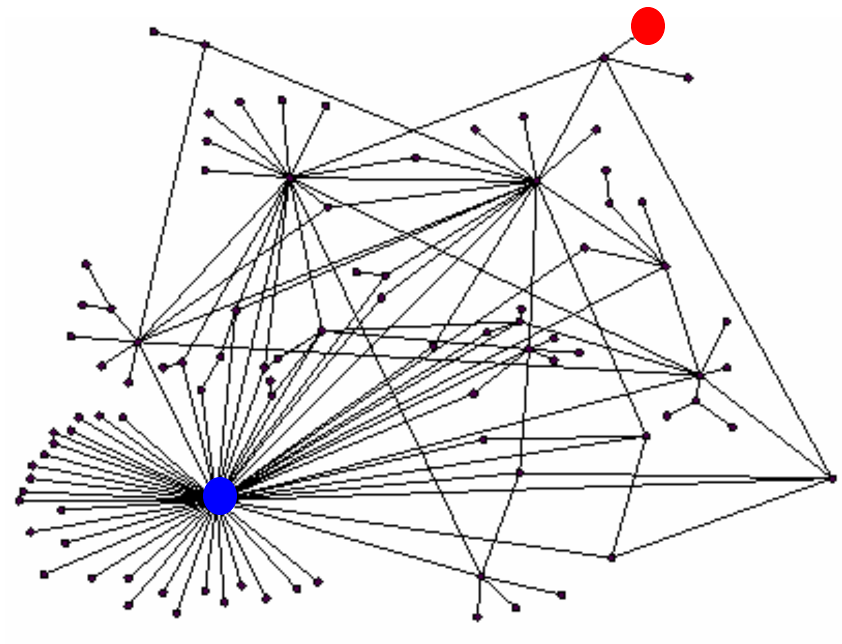
We will see this can be efficiently implemented thanks to a beautifully simple *mathematical algorithm*.

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

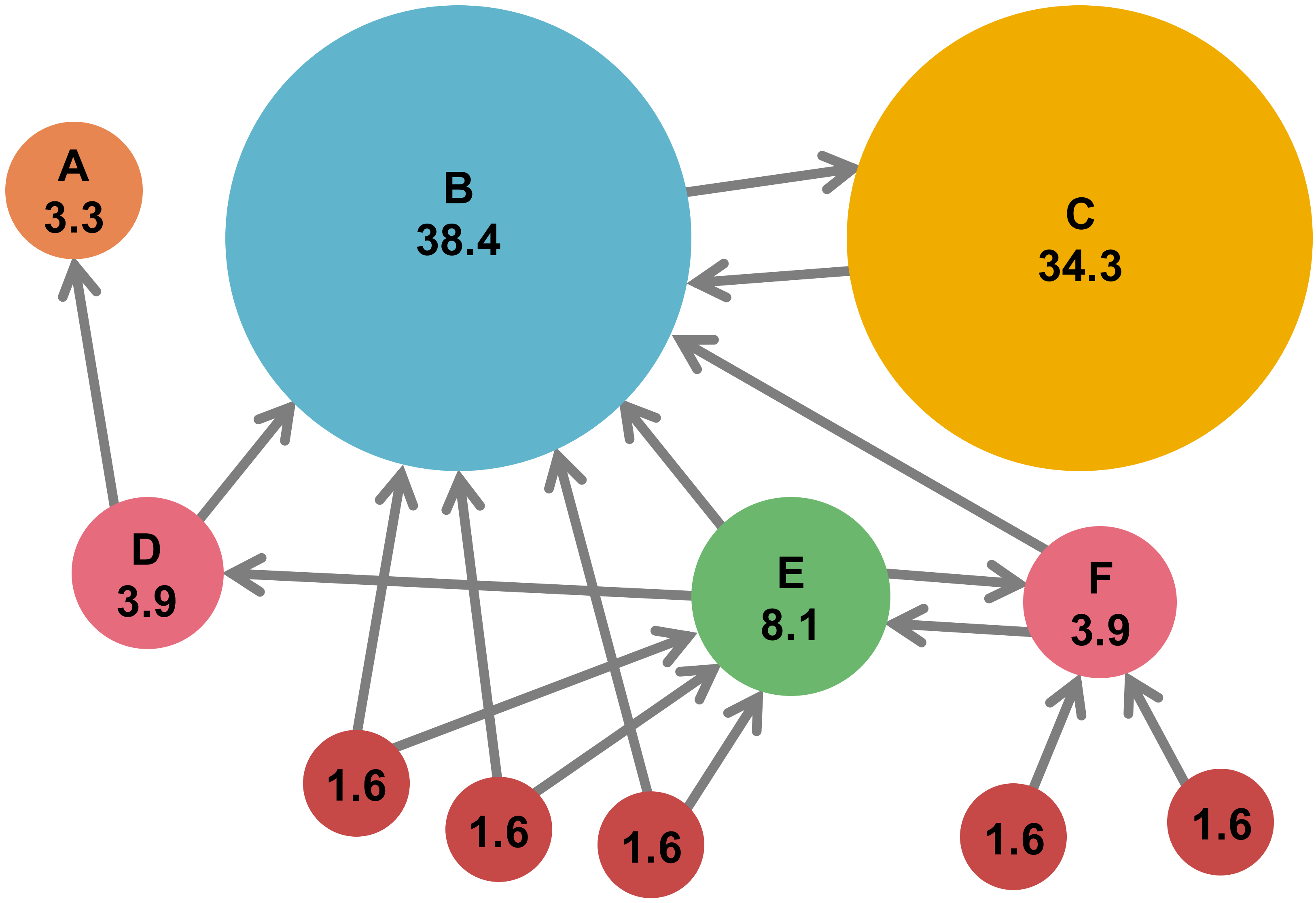
- We will cover the following **Link Analysis approaches** for computing **importances of nodes in a graph**:
  - Page Rank
  - Topic-Specific (Personalized) Page Rank
  - Web Spam Detection Algorithms

PageRank:
The “Flow” Formulation

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! How to solve it?

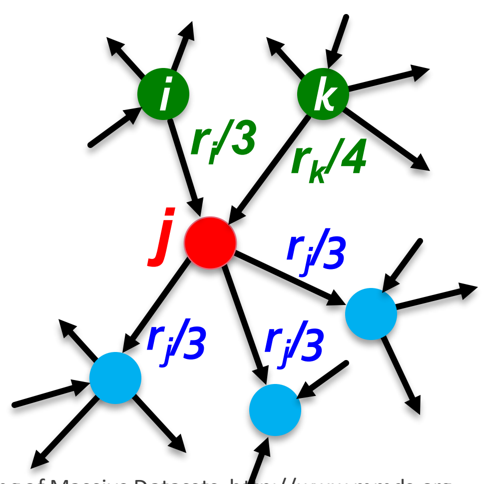
Example: PageRank Scores



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

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



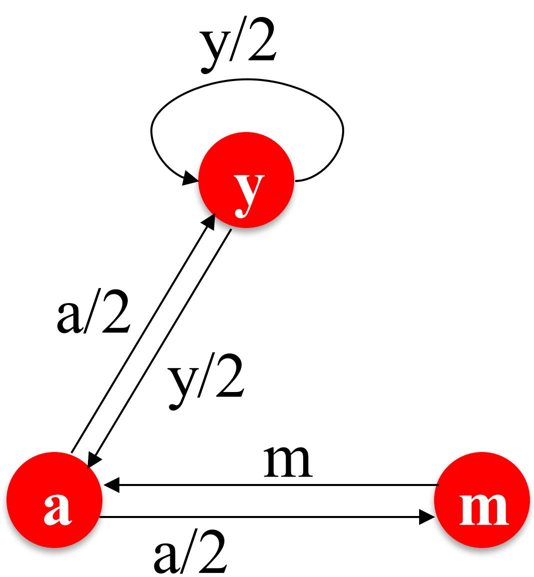
PageRank: The "Flow" Model

- 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:

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

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!**

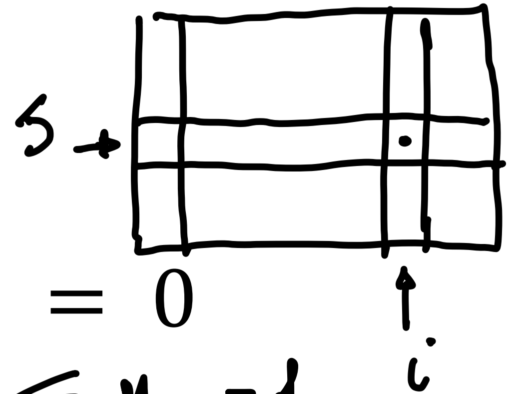
NOTA

- LA PARTE DI ALGEBRA LINEARE E DI MARKOV CHAIN E' FATTA BENE NELLE NOTE:

- 1B-Google_slides.pdf

- disponibili su Teams (folder: *materiale mining massive data*)

PageRank: Matrix Formulation



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

$$r = M \cdot r$$

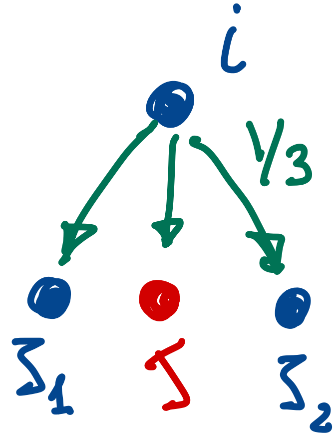
$$r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}$$

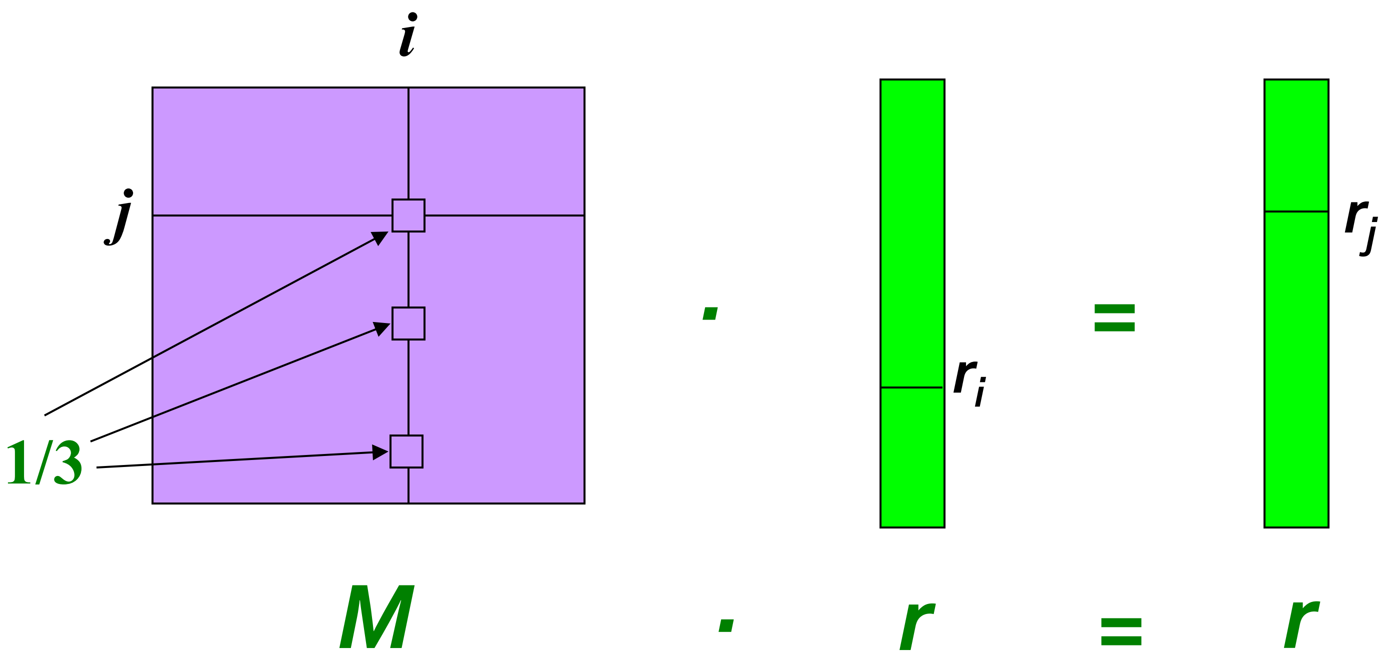
Example

- 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*





Eigenvector Formulation

- The flow equations can be written

$$\boldsymbol{r} = \boldsymbol{M} \cdot \boldsymbol{r} \quad (\lambda = 1)$$

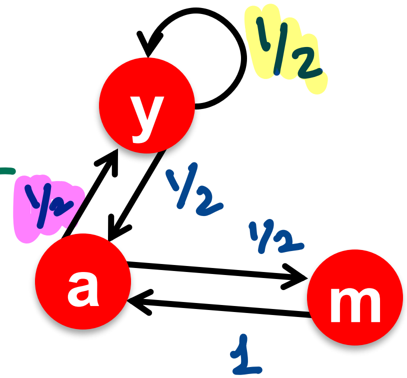
- So the **rank vector** ***r*** is an **eigenvector** of the stochastic web matrix ***M***
  - In fact, its first or principal eigenvector, with corresponding eigenvalue ***1*** = MAX
    - Largest eigenvalue of ***M*** is **1** since ***M*** is column stochastic (with non-negative entries)
      - *We know **r** is unit length and each column of **M** sums to one, so $\boldsymbol{Mr} \le \mathbf{1}$* in L_1 norm

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

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

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

Example: Flow Equations & M



TRANS MATRIX

	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

MATRIX REPR.

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

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

New score for y

old score for y

$$\begin{bmatrix} y \\ a \\ m \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 1 \\ 0 & \frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} y \\ a \\ m \end{bmatrix}$$

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: $\mathbf{r}^{(0)} = [1/N,\ldots,1/N]^T$ → UNIF. START
  - Iterate: $\mathbf{r}^{(t+1)} = \mathbf{M} \cdot \mathbf{r}^{(t)}$
  - Stop when $|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}|_1 < \varepsilon$ → "ALMOST" a FIXED POINT
    - $|\mathbf{x}|_1 = \sum_{1 \le i \le N} |x_i|$ is the L_1 norm
    - Can use any other vector norm, e.g., Euclidean

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

d_i .... out-degree of node i

FIXED POINT ≡ EIGENVECTOR of M with $\lambda = 1$

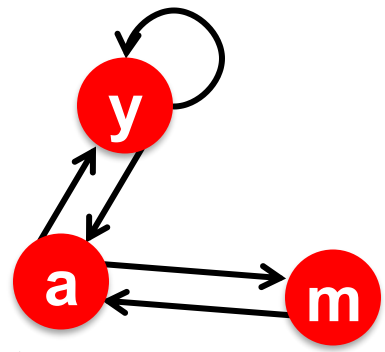
PageRank: How to solve?

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

t=0

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

Iteration 0, 1, 2, …



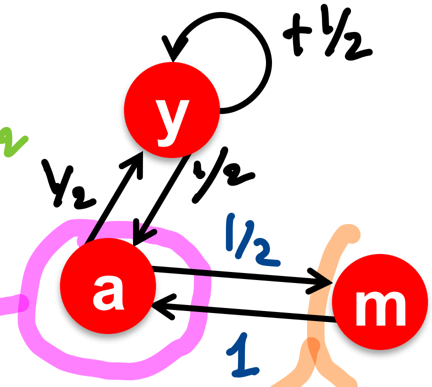
	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

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

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**

balance is +1/2



+1/2 } LOWER
−1 } FLOW

	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:**

	t=0	t=1	t=2	t=3	....	
r_y =	1/3	1/3	5/12	9/24		6/15
r_a =	1/3	3/6	1/3	11/24	…	6/15
r_m =	1/3	1/6	3/12	1/6		3/15

fixed point in Eigen Vector!

Iteration 0, 1, 2, …

Why Power Iteration works? (1)

Details!

- **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 $\boldsymbol{M}$

Proof → next slides

Why Power Iteration works? (2) Details!

- **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 $\boldsymbol{M}$ (WHEN?)
- **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 $\lambda_1 > \lambda_2 > \cdots > \lambda_n$] $\det(M) \neq 0$

$\lambda_1 = 1$

  - Vectors $x_1, x_2, \ldots, x_n$ form a basis and thus we can write:

$\forall$ any $r^{(0)}$: $r^{(0)} = c_1\, x_1 + c_2\, x_2 + \cdots + c_n\, x_n$, for some $\langle c_1, \ldots, c_n \rangle$ $\to$ MUST BE $\neq 0$!

  - $\boldsymbol{M}\boldsymbol{r}^{(0)} = \boldsymbol{M}(\boldsymbol{c_1\, x_1 + c_2\, x_2 + \cdots + c_n\, x_n})$

$$= c_1(Mx_1) + c_2(Mx_2) + \cdots + c_n(Mx_n)$$

$$= c_1(\lambda_1 x_1) + c_2(\lambda_2 x_2) + \cdots + c_n(\lambda_n x_n)$$

  - **Repeated multiplication on both sides produces**

$$M^k r^{(0)} = c_1(\underbrace{\lambda_1^k}_{\text{largest}=1} x_1) + c_2(\underbrace{\lambda_2^k}_{<1} x_2) + \cdots + c_n(\underbrace{\lambda_n^k}_{<1} x_n)$$

Why Power Iteration works? (3)

Details!

- **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
    $$M^k r^{(0)} = c_1(\lambda_1^k x_1) + c_2(\lambda_2^k x_2) + \cdots + c_n(\lambda_n^k x_n)$$
  - $$M^k r^{(0)} = \underset{=1}{\lambda_1^k}\left[c_1 x_1 + \boxed{c_2\left(\frac{\lambda_2}{\lambda_1}\right)^k x_2 + \cdots + c_n\left(\frac{\lambda_2}{\lambda_1}\right)^k x_n}\right] \to 0$$
  - Since $\lambda_1 > \lambda_2$ then fractions $\frac{\lambda_2}{\lambda_1}, \frac{\lambda_3}{\lambda_1} \ldots < 1$

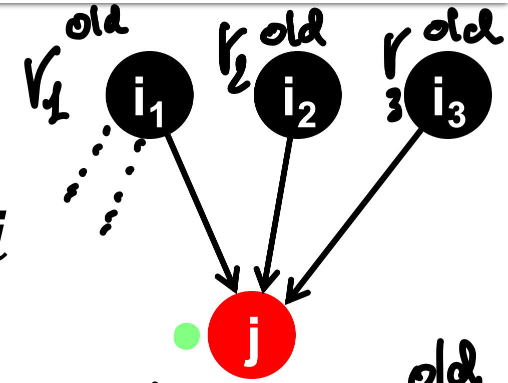
    and so $\left(\frac{\lambda_i}{\lambda_1}\right)^k \to 0$ as $k \to \infty$ (for all $i = 2 \ldots n$).
  - **Thus:** $\boldsymbol{M^k r^{(0)} \approx c_1(\lambda_1^k x_1)}$ → *
    - Note if $c_1 = 0$ then the method won't converge

Converge * in NORM L_1

→ the initial vector must have a >0 comp. on x_1

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



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

- **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

VERY IMPORT

$\sum_i p_i(t) = 1$

MARKOV CHAIN

Markov Chain

n = time slots t

- A Markov chain is a sequence $X_1, X_2, X_3, \ldots$ of random variables ($\Sigma_{v \text{ all possible values of } X}$ P(X=v) = 1) with the property:
- **Markov property**: the conditional probability distribution of the next future state X_{n+1} given the present and past states is a function of the present state X_n alone

t = w

$\forall x, x_0, \ldots, x_n$:

$$\Pr(X_{n+1} = x | \cancel{X_0 = x_0, X_1 = x_1, \ldots,} X_n = x_n) = \Pr(X_{n+1} = x | X_n = x_n).$$

- If the state space is finite then the transition probabilities can be described with a matrix $P_{ij}=P(X_{n+1}= j \mid X_n = i)$, i,j =1, …m = N

Pr[1 → 2]

j i → riga

$$* \begin{pmatrix} .85 & .15 \\ .38 & .62 \end{pmatrix} = \begin{pmatrix} P(X_{n+1} = 1 \mid X_n = 1) & P(X_{n+1} = 2 \mid X_n = 1) \\ P(X_{n+1} = 1 \mid X_n = 2) & P(X_{n+1} = 2 \mid X_n = 2) \end{pmatrix}$$

Note: ||| $P = M^T$

column j = 1

i = 2

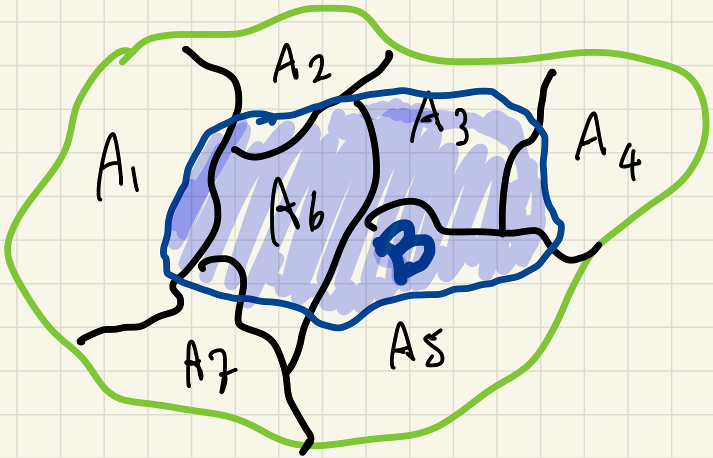
↳ link (2,1)

*[rows sum to 1]

PROBABILITY SPACE $\langle \Omega, \Pr \rangle$ $|\Omega| = N$

Sets $\{A_1, \dots, A_k\}$ a partition of Ω then, $\forall B \subseteq \Omega$, it holds:

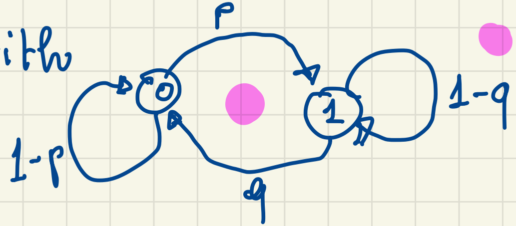
$$\Pr[B] = \sum_{j=1}^{k} \Pr[B \mid A_j] \cdot \Pr[A_j]$$



We use this LAW for computing MARKOV CHAIN's DISTRIBUTIONS!

$$\bar{\pi}(t+1) = F(\bar{\pi}(t))$$

Ex/ M.C. (**) $\left[\Omega = \{0,1\}\ ;\ \text{with} \quad \{X_0, X_1, \dots, X_t, \dots\} \right.$



$$\begin{cases} \Pr[X_t = 1 \mid X_{t-1} = 0] = p \\ \Pr[X_t = 0 \mid X_{t-1} = 1] = q \end{cases}$$

$$\begin{array}{c|cc} & j=0 & j=1 \\ \hline i=0 & 1-p & p \\ i=1 & q & 1-q \end{array} \quad \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix} \begin{matrix} \to \Sigma = 1 \\ \to \Sigma = 1 \end{matrix} \equiv \text{TRANSITION MATRIX}$$

$$P = \langle P[i,j] \rangle \equiv \Pr[X_t \overset{\text{COLUMN}}{=} j \mid X_{t-1} \overset{\text{raw}}{=} i] \quad \forall t\ !$$

Consider $\bar{\pi}(t) = \begin{pmatrix} \pi_0(t) \\ \pi_1(t) \end{pmatrix} \equiv$ distr. of X_t (it is a col vector!)

How to compute $\bar{\pi}(t+1)$? We need to know $\bar{\pi}(t)$:

$\forall j = 0,1$, we have (1) $\Pr[X_t = j] \equiv \sum_{i=0,1} \Pr[X_t = j \mid X_{t-1} = i] \cdot \Pr[X_{t-1} = i]$ *

In MATRIX ALGEBRA, (1) is equivalent to: → vett. colonna

$$\bar{\pi}(t+1) = P \cdot \bar{\pi}(t) \Rightarrow \bar{\pi}(t+1) = P \cdot (P \cdot \bar{\pi}(t-1)) = \dots =$$

$$= P^t \cdot \bar{\pi}(0) \to \text{STARTING DISTRIB.}$$

(**) To define a M.C., we need:

$X \equiv \langle \Omega, P \rangle$, i.e., SET of STATES $\wedge$ TRANS. MATRIX

MAIN QUESTIONS:

I) starting from $\Pi(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ (arbitrary distrib.): $\underbrace{\mathbb{P}_R[\text{Agent is INFECTED at round } t=3]}_{(a)} = ?$

Answer POWER METHOD

$$P^3 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix}^3 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} (a) \\ 1-(a) \end{pmatrix}$$

[State 0 ≡ SUSCEPTIBLE
state 1 ≡ INFECTED]

II) starting from any $\Pi(0)$, after a large $\tilde{t} \to +\infty$, can we found a stationary distribution Π s.t. $\forall t \geq \tilde{t}$, we have

$$\Pi(t) = P \cdot \Pi(t-1) = \Pi \quad ?$$

ANSWER We need to find a vector $\tilde{\Pi}$ s.t. $P \cdot \tilde{\Pi} = \tilde{\Pi}$, i.e., an eigenvector $\tilde{\Pi}$ with eigenvalue $\lambda = 1$

MAIN QUESTIONS ON MARKOV CHAIN: $X \equiv \langle \Omega, P \rangle$

(1) when $P^t \pi(0)$ converge?

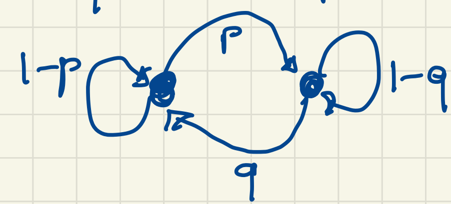
(2) where? which is the meaning?

(3) CONVERGENCE TIME?

Consider birth-death M.C. ( ): $\Omega = \{0,1\}$; $p \equiv$ birth prob

$q \equiv$ death prob.

$\forall \bar{\pi} = \langle \Pi_0, \Pi_1 \rangle$, $\Pi(0) = \bar{\Pi}$:



$$\begin{pmatrix} \Pi_0(1) = \bar{\Pi}_0 \cdot (1-p) + \Pi_1 \cdot q \\ \Pi_1(1) = \Pi_0 \cdot p + \Pi_1 \cdot (1-q) \end{pmatrix} \equiv P \cdot \Pi(0),$$

In general: $\Pi(t) = P^t \cdot \pi(0)$, when stops? CNES to stop is

$\exists \tau \in \mathbb{N}$ s.t. (2) $\bar{\Pi}(\tau) = P \cdot \bar{\Pi}(\tau - 1)$ $\equiv$ a fixed point of the process $\{\Pi(0), \dots, \bar{\Pi}(t), \dots\}$

Indeed, if (2) holds then:

$$\forall t \geq \tau \Rightarrow \bar{\Pi}(t) = P \cdot \tilde{\Pi}(t-1) = \dots = \overbrace{P \cdot P \cdots P \cdot \Pi(\tau-1)}^{\Pi(\tau)} = \bar{\Pi}(\tau-1)$$

So the M.C. reaches a STATIONARY DISTRIBUTION $\overline{\pi}$ iff

$\overline{\pi} = P\,\overline{\pi} \iff \overline{\pi}$ is an eigenvector of P with (MAX) eigenvalue $\lambda = 1$ $\quad \overline{\pi} = P \cdot (\lambda \cdot \overline{\pi})$

Ex $(\cdot)$ $\begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix} \Rightarrow$ consider $\boxed{\tilde{\pi} = \begin{pmatrix} q/p+q \\ p/p+q \end{pmatrix}}$:

$$\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 1-p & p \\ 1 & q & 1-q \end{array} \quad \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix} \underbrace{\begin{pmatrix} q/p+q \\ p/p+q \end{pmatrix}}_{\tilde{\pi}} = \begin{pmatrix} (1-p) \cdot \frac{q}{p+q} + p\,\frac{q}{p+q} = \frac{q - \cancel{pq} + \cancel{pq}}{p+q} \\ \text{SIMILAR CALCULUS} \end{pmatrix} = \underbrace{\begin{pmatrix} q/p+q \\ p/p+q \end{pmatrix}}_{\tilde{\pi}}$$

so, $\tilde{\pi} = (\;)$ is an eigenvector of P and is

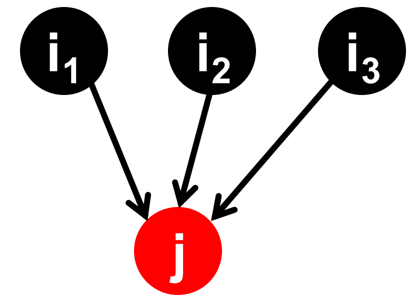
A STATIONARY DISTRIB. for M.C. $P \equiv (\cdot)$

Q. Is it UNIQUE? $\quad$ Q. $\forall$ starting π, $P^t \pi \to \tilde{\pi}$??

The Stationary Distribution

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

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



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

- Suppose the random walk reaches a state

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

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

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

column vector

$M = (M(i,j) = P_R[i \to j] = \frac{r_i}{d_i}$

SURFER

Existence and Uniqueness

stationary pag-rank r exists? is it UNIQUE?

- **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**

Ergodic Markov chains

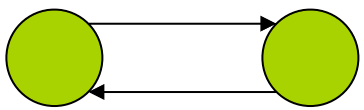
→ GRAPH Theory!

→ STRONG CONNECT. + ∀ length!

- A Markov chain is **ergodic** if:
  - Informally: *there is a path from any state to any other; and the states are not partitioned into sets such that all state transitions occur cyclically from one set to another.*
  - Formally: for any start state, after a finite transient time T_0, the probability of being in any state **at any fixed time $T>T_0$** is nonzero.

→ $\forall i \in [N]$, vector $P^{(t)}(i) > 0, \forall t \geq T_0$

BIPARTITE



Not ergodic (even/odd).

Not ergodic: the probability to be in a state, at a fixed time, e.g., after 500 transitions, is always either 0 or 1 according to the initial state.

Ergodic Markov chains

- For any ergodic Markov chain, there is a **unique long-term visit rate** for each state
  - *Steady-state probability distribution*
- Over a long time-period, we visit each state in proportion to this rate
- It doesn't matter where we start.
- *Note: non ergodic Markov chains may still have a steady state.*

PageRank:
The Google Formulation

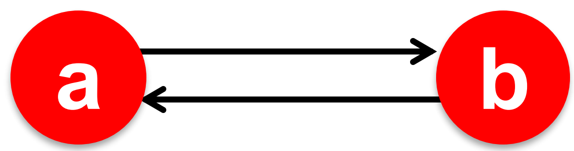
PageRank: Three Questions

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

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

Does this converge?

MAIN ISSUE : WEB GRAPH IS NOT ERGODIC ☹



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

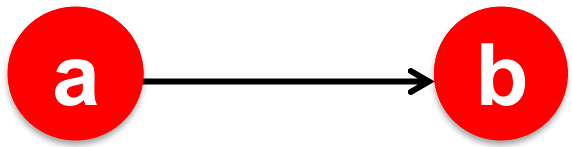
- **Example:**

$$\begin{array}{c} \mathrm{r_a} \\ \mathrm{r_b} \end{array} = \begin{array}{cccc} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array}$$

Iteration 0, 1, 2, …

Does it converge to what we want?

it converges! But....



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

- **Example:**

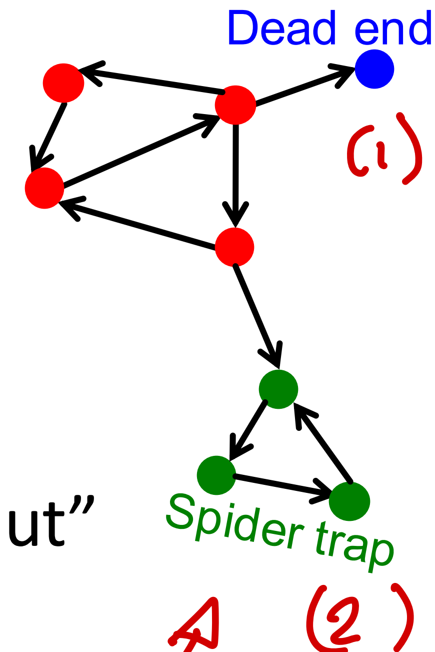
$$\begin{array}{ccccccc} r_a & = & 1 & 0 & 0 & 0 \\ r_b & & 0 & 1 & 0 & 0 \end{array}$$

Iteration 0, 1, 2, …

= ??

Fixed Point no meaning!

PageRank: Problems



2 problems:

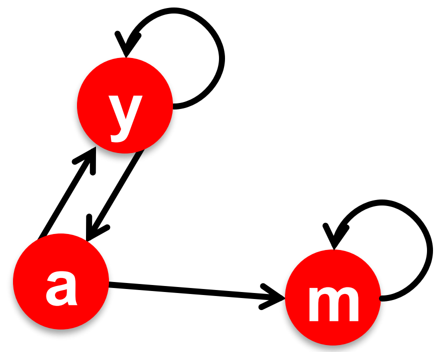
- **(1)** Some pages are **dead ends** (have no out-links)
  - Random walk has "nowhere" to go to
  - Such pages cause importance to "leak out"

there is a "GANG"

- **(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: 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

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

- **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, …

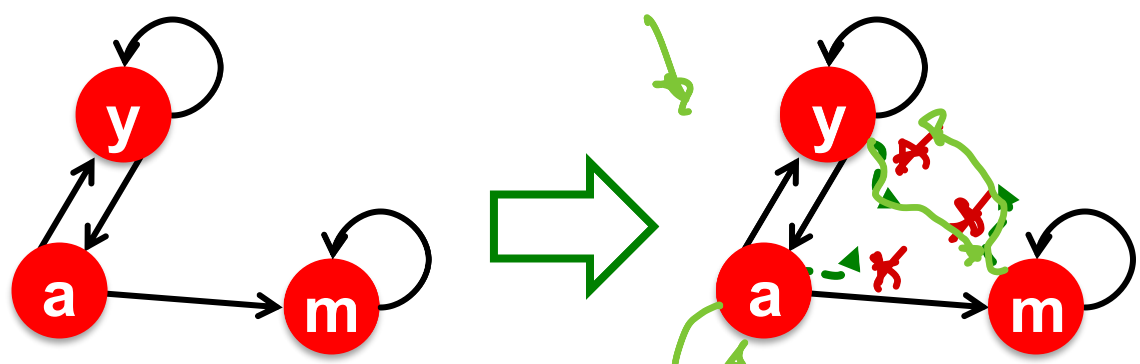
All the PageRank score gets “trapped” in node m.

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 (PREVIOUS SIMPLE RND WALK)
  - 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**

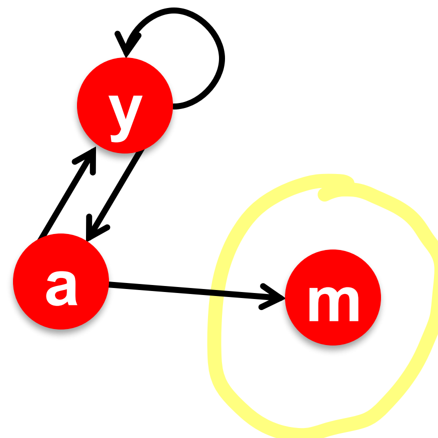
▷ TELEPORTS BRIDGES

* the walker does not follow the link! it jumps to any page of the WEB u.a.r.



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

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

- **Example:**

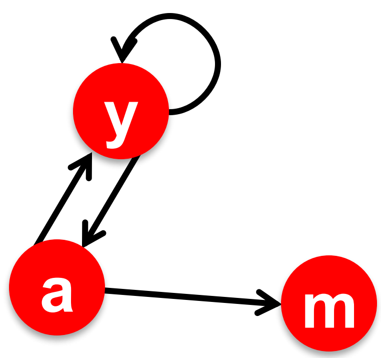
$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{array}{cccccc} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\ 1/3 & 1/6 & 1/12 & 2/24 & & 0 \end{array}$$

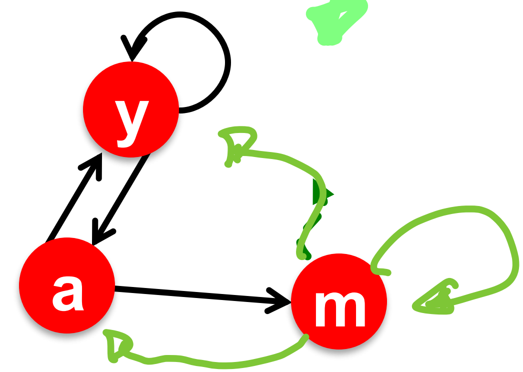
Iteration 0, 1, 2, …

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

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

$\beta = 0!$

  - **Solution:** Make matrix column stochastic by always teleporting when there is nowhere else to go

Solution: Random Teleports

- <u>**Google's solution that does it all:**</u>
  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]

OLD FLUID EQ.s

TELEPORT TERM

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

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:**

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

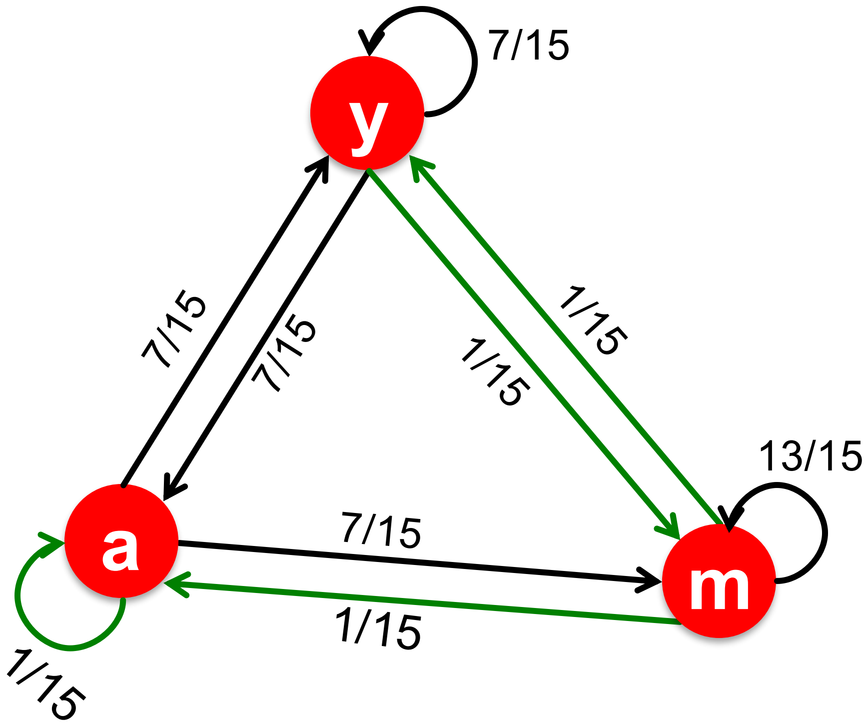
$$A = \beta\, M + (1-\beta)\left[\frac{1}{N}\right]_{N\times N}$$

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

And the Power method still works! : $r^{(t)} = A^{(t)} \cdot r^{(0)}$

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

Random Teleports ($\beta = 0.8$)



$$0.8 \overset{\mathbf{M}}{\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix}} + 0.2 \overset{[1/N]_{N\times N}}{\begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}}$$

+ Teleport

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

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

MAX Eigen Vector of A

popular SCORES

How do we actually compute the PageRank?

Computing Page Rank

- **Key step is matrix-vector multiplication**
  - $\boldsymbol{r}^{\text{new}} = \boldsymbol{A} \cdot \boldsymbol{r}^{\text{old}}$
- Easy if we have enough main memory to hold **A**, $\mathbf{r}^{\text{old}}$, $\mathbf{r}^{\text{new}}$
- **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!

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

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

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

Matrix Formulation

- Suppose there are N pages
- Consider page i, with d_i out-links
- We have $M_{ji} = 1/|d_i|$ when $i \to j$ and $M_{ji} = 0$ otherwise
- **The random teleport is equivalent to:**
  - Adding a **teleport link** from i to every other page and setting transition probability to $(1-\beta)/N$
  - Reducing the probability of following each out-link from $1/|d_i|$ to $\beta/|d_i|$
  - **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** $\boldsymbol{A_{ji} = \beta\, M_{ji} + \frac{1-\beta}{N}}$
- $r_j = \sum_{\mathrm{i}=1}^{N} A_{ji} \cdot r_i$
- $r_j = \sum_{i=1}^{N} \left[\beta\, M_{ji} + \frac{1-\beta}{N}\right] \cdot r_i$

$$= \sum_{\mathrm{i}=1}^{N} \beta\, M_{ji} \cdot r_i + \frac{1-\beta}{N} \sum_{\mathrm{i}=1}^{N} r_i$$

$$= \sum_{\mathrm{i}=1}^{N} \beta\, M_{ji} \cdot r_i + \frac{1-\beta}{N}$$

since $\sum r_i = 1$

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

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-β)/N]$_N$** is a vector with all ***N*** entries **(1-β)/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 $\boldsymbol{r}^{\text{new}} = \beta \boldsymbol{M} \cdot \boldsymbol{r}^{\text{old}}$
  - Add a constant value **(1-β)/N** to each entry in $\boldsymbol{r}^{\text{new}}$
    - **Note if M contains dead-ends then $\sum_j r_j^{new} < 1$ 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 (can have **spider traps** and **dead ends**)
  - Parameter β
- **Output: PageRank vector r^{new}**
  - **Set:** $r_j^{old} = \frac{1}{N}$
  - **repeat until convergence:** $\sum_j \left| r_j^{new} - r_j^{old} \right| > \varepsilon$
    - $\forall j$: $\boldsymbol{r'^{new}_j} = \sum_{i \to j} \boldsymbol{\beta} \frac{\boldsymbol{r_i^{old}}}{\boldsymbol{d_i}}$

      $\boldsymbol{r'^{new}_j} = \mathbf{0}$ if in-degree of $\boldsymbol{j}$ is **0**
    - **Now re-insert the leaked PageRank:**

      $\forall \boldsymbol{j}$: $\boldsymbol{r_j^{new}} = \boldsymbol{r'^{new}_j} + \frac{\mathbf{1-S}}{\boldsymbol{N}}$ **where:** $S = \sum_j r'^{new}_j$
    - $\boldsymbol{r^{old}} = \boldsymbol{r^{new}}$

If the graph has no dead-ends then the amount of leaked PageRank is **1-β**. But since we have dead-ends the amount of leaked PageRank may be larger. We have to explicitly account for it by computing **S**.

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 r^{new} into memory**
  - Store r^{old} and matrix **M** on disk
- **1 step of power-iteration is:**

> **Initialize** all entries of $\mathbf{r}^{new} = (1-\beta) / N$
> For each page i (of out-degree d_i):
>   Read into memory: i, d_i, $dest_1, \ldots, dest_{d_i}$, $r^{old}(i)$
>   For $\mathbf{j} = 1 \ldots \mathbf{d_i}$
> $\mathbf{r^{new}(dest_j) \mathrel{+}= \beta\, r^{old}(i) / d_i}$

r^{new}: 0, 1, 2, 3, 4, 5, 6

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

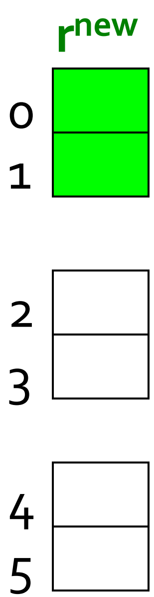
r^{old}: 0, 1, 2, 3, 4, 5, 6

Analysis

- **Assume enough RAM to fit r^{new} into memory**
  - Store $\boldsymbol{r}^{old}$ and matrix $\boldsymbol{M}$ on disk
- **In each iteration, we have to:**
  - Read $\boldsymbol{r}^{old}$ and $\boldsymbol{M}$
  - Write $\boldsymbol{r}^{new}$ back to disk
  - **Cost per iteration of Power method:**
    $= 2|\boldsymbol{r}| + |\boldsymbol{M}|$

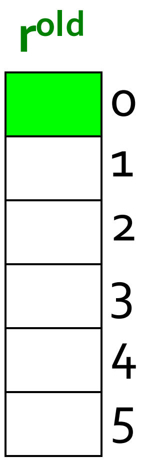
- **Question:**
  - What if we could not even fit $\boldsymbol{r}^{new}$ in memory?

Block-based Update Algorithm



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

M



- Break r^{new} into k blocks that fit in memory
- Scan M and r^{old} once for each block

Analysis of Block Update

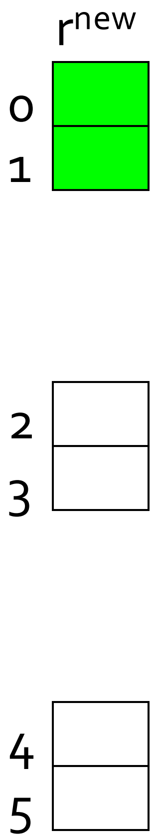
- **Similar to nested-loop join in databases**
  - Break $\boldsymbol{r}^{\text{new}}$ into $\boldsymbol{k}$ blocks that fit in memory
  - Scan $\boldsymbol{M}$ and $\boldsymbol{r}^{\text{old}}$ once for each block
- **Total cost:**
  - $\boldsymbol{k}$ scans of $\boldsymbol{M}$ and $\boldsymbol{r}^{\text{old}}$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

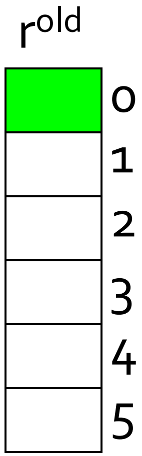
Block-Stripe Update Algorithm

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





Break *M* into stripes! Each stripe contains only destination nodes in the corresponding block of r^{new}

Block-Stripe Analysis

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

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 of importance
  - **Solution:** Hubs-and-Authorities
- **Susceptible to Link spam**
  - Artificial link topographies created in order to boost page rank
  - **Solution:** TrustRank