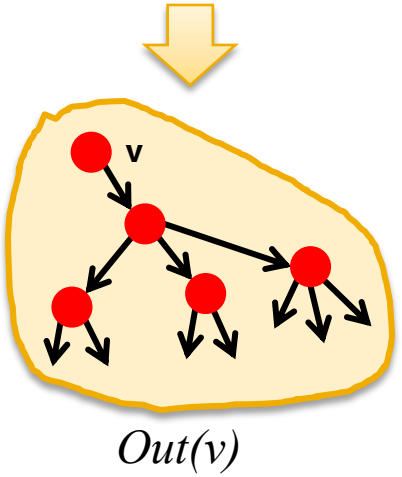
Link Analysis: PageRank

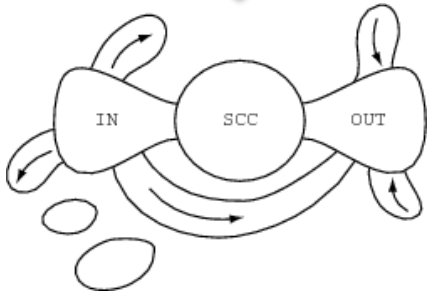
CS224W: Analysis of Networks
Jure Leskovec, Stanford University
http://cs224w.stanford.edu

Web as a Graph

Structure of the Web

- **Today we will talk about how does the Web graph look like:**
  - **1)** We will take a real system: **the Web**
  - **2)** We will represent it as a **directed graph**
  - **3)** We will use the language of graph theory
    - **Strongly Connected Components**
  - **4)** We will design a **computational experiment:**
    - Find In- and Out-components of a given node ***v***
  - **5) We will learn something about the structure of the Web: BOWTIE!**

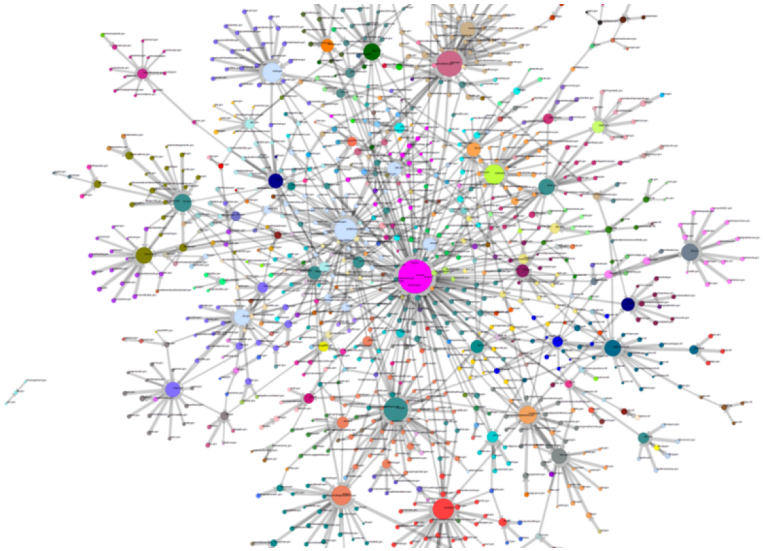




The Web as a Graph

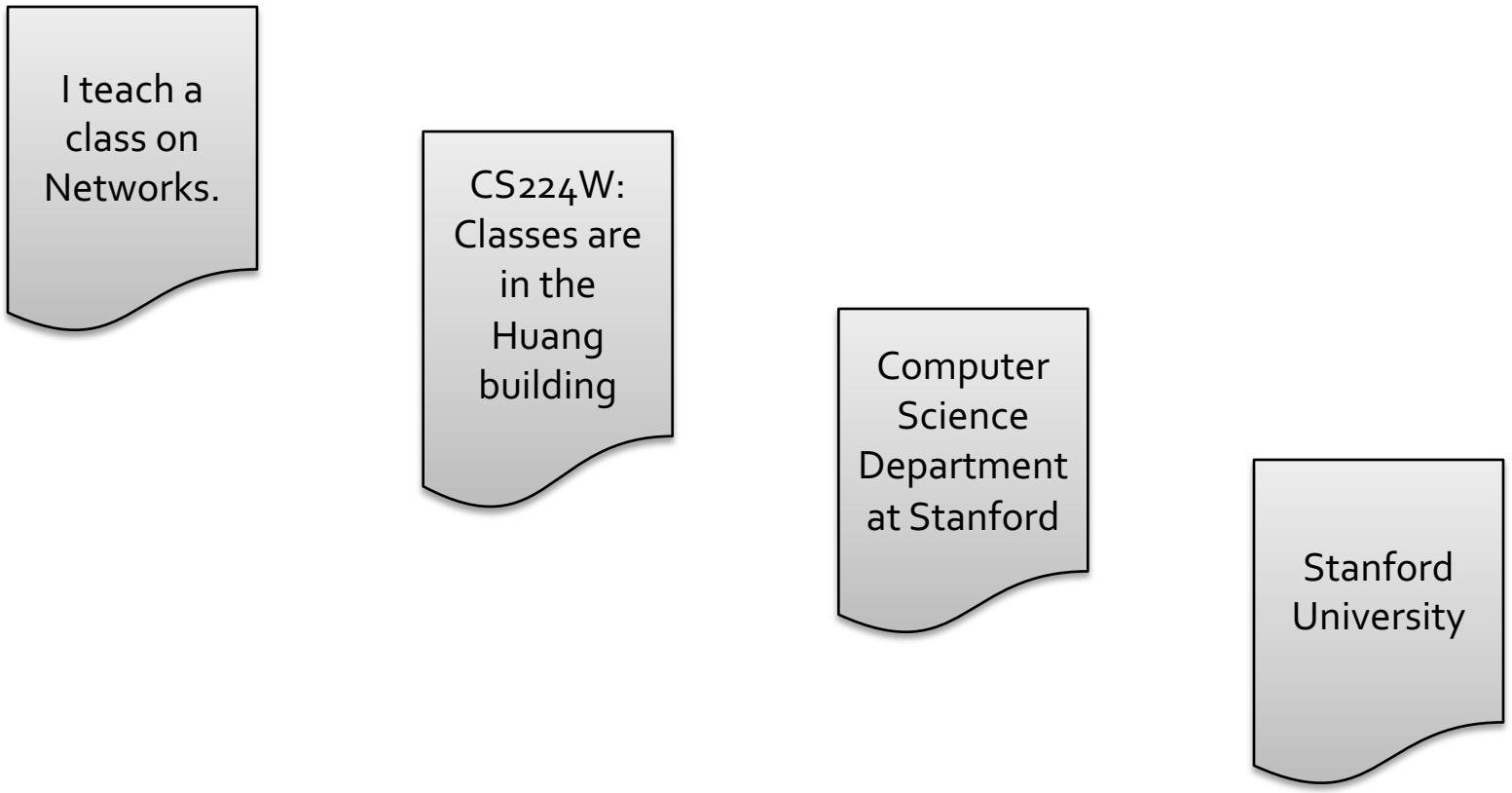
Q: What does the Web "look like" at a global level?

- **Web as a graph:**
  - Nodes = web pages
  - Edges = hyperlinks

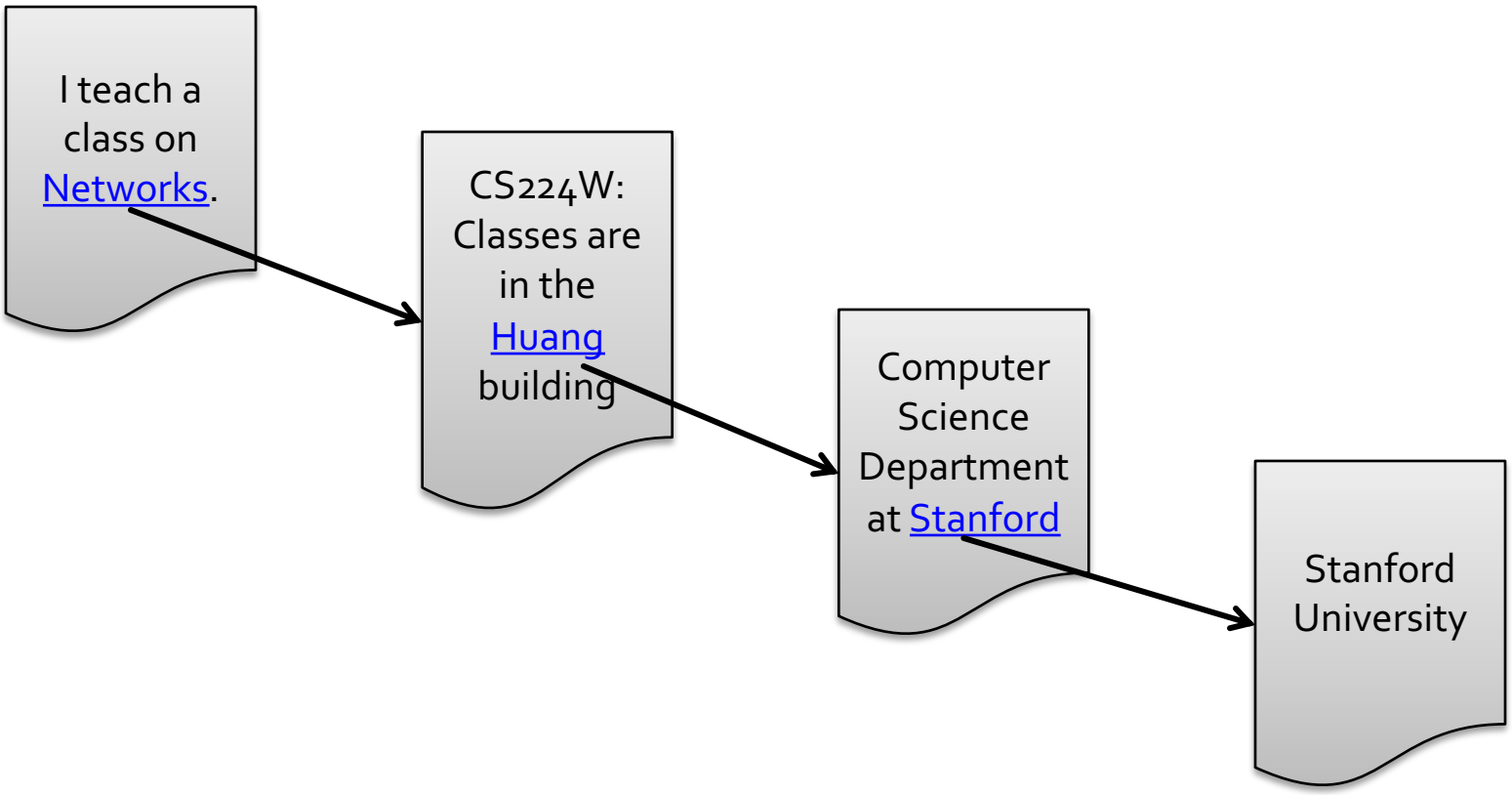


  - **Side issue:** What is a node?
    - Dynamic pages created on the fly
    - "dark matter" – inaccessible database generated pages

The Web as a Graph

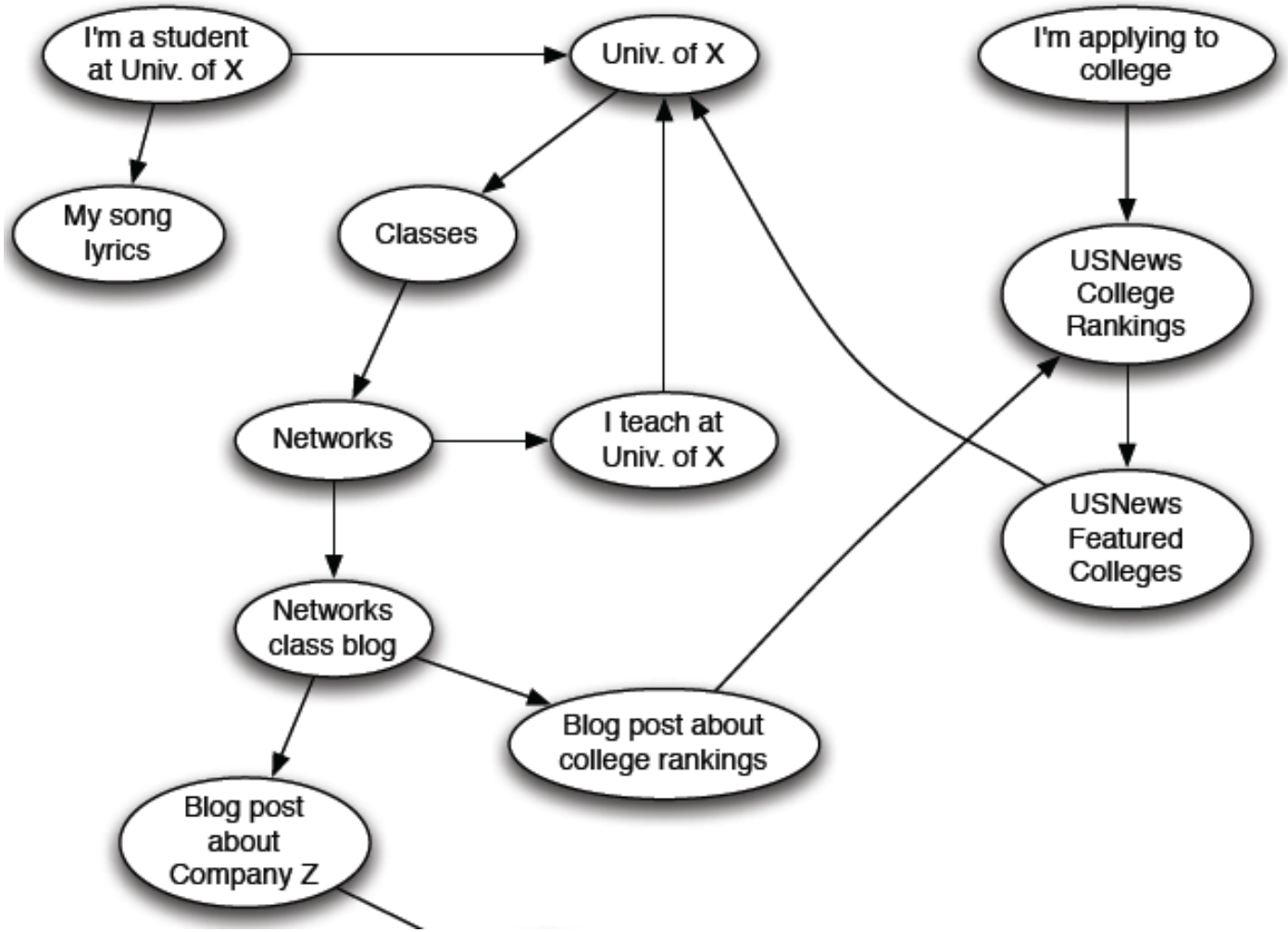


The Web as a Graph

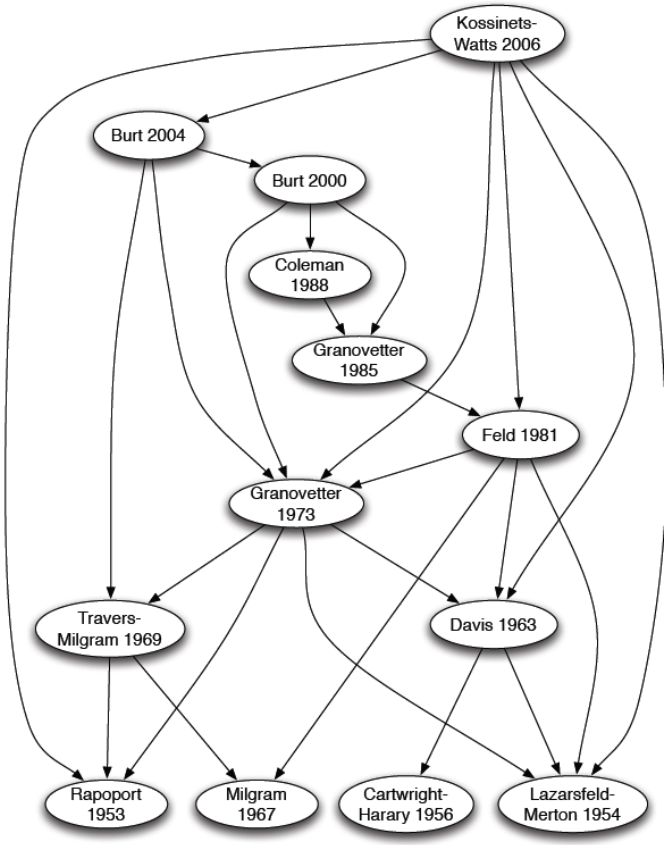


- In early days of the Web links were **navigational**
- Today many links are **transactional** (used **not** to navigate from page to page, but to post, comment, like, buy, …)

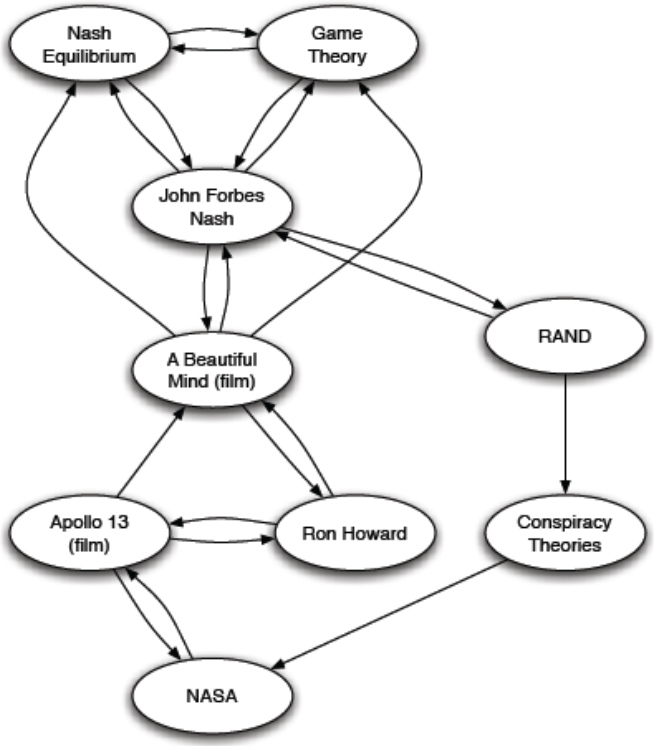
The Web as a Directed Graph



Other Information Networks



Citations



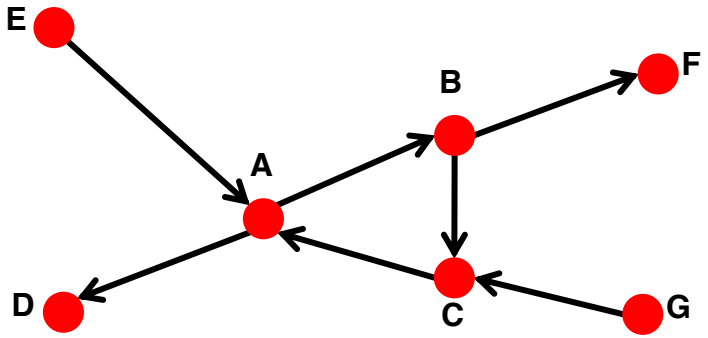
References in an Encyclopedia

What Does the Web Look Like?

- **How is the Web linked?**
- **What is the "map" of the Web?**

Web as a directed graph [Broder et al. 2000]:

- Given node v, what can v reach?
- What other nodes can reach v?



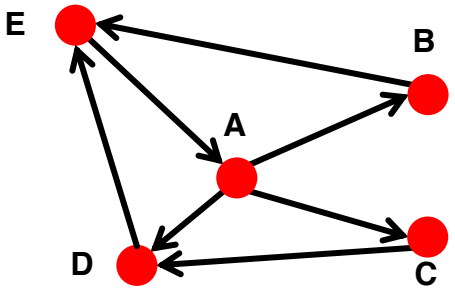
$$In(v) = \{w \mid w \text{ can reach } v\}$$
$$Out(v) = \{w \mid v \text{ can reach } w\}$$

For example:
In(A) = {A,B,C,E,G}
Out(A)={A,B,C,D,F}

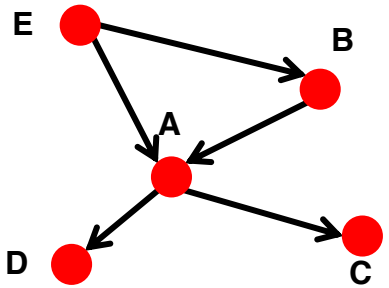
Reasoning about Directed Graphs

- **Two types of directed graphs:**
  - **Strongly connected:**
    - Any node can reach any node via a directed path

      $In(A)=Out(A)=\{A,B,C,D,E\}$



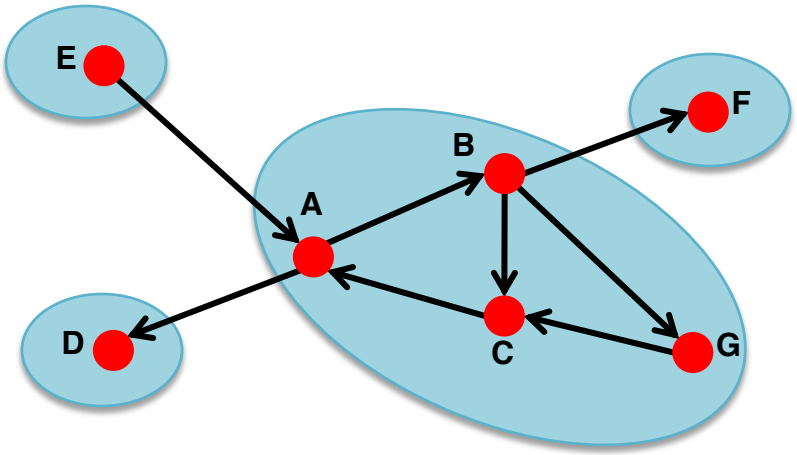
  - **Directed Acyclic Graph (DAG):**
    - Has no cycles: if u can reach v, then v cannot reach u



- **Any directed graph (the Web) can be expressed in terms of these two types!**
  - Is the Web a big strongly connected graph or a DAG?

Strongly Connected Component

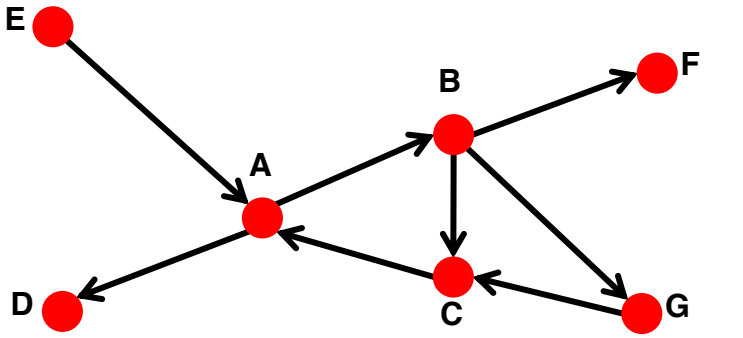
- **A Strongly Connected Component (SCC)** is a set of nodes S so that:
  - Every pair of nodes in S can reach each other
  - There is no larger set containing S with this property



Strongly connected components of the graph: {A,B,C,G}, {D}, {E}, {F}

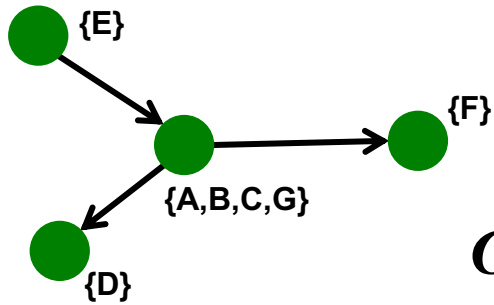
Strongly Connected Component

- **Fact:** **Every directed graph is a DAG on its SCCs**
  - **(1)** SCCs partition the nodes of G
    - That is, each node is in exactly one SCC
  - **(2)** If we build a graph G' whose nodes are SCCs, and with an edge between nodes of G' if there is an edge between corresponding SCCs in G, then G' is a DAG



G

(1) Strongly connected components of graph G: {A,B,C,G}, {D}, {E}, {F}
(2) G' is a DAG:



G'

Structure of the Web

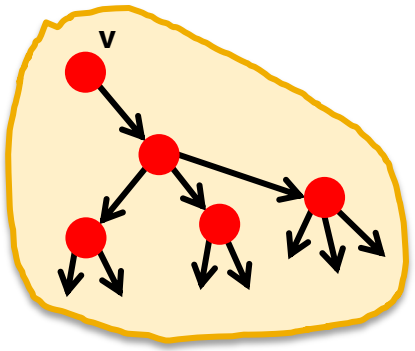
- **Broder et al.: Altavista web crawl (Oct '99)**
  - Web crawl is based on a large set of starting points accumulated over time from various sources, including voluntary submissions.
  - 203 million URLS and 1.5 billion links

Goal: Take a large snapshot of the Web and try to understand how its SCCs "fit together" as a DAG

Tomkins, Broder, and Kumar

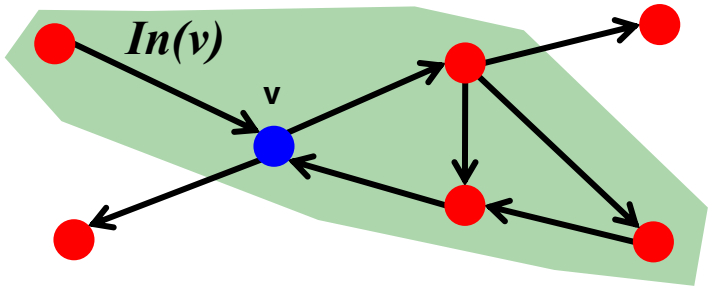
Graph Structure of the Web

- **Computational issue:**
  - Want to find a SCC containing node v?



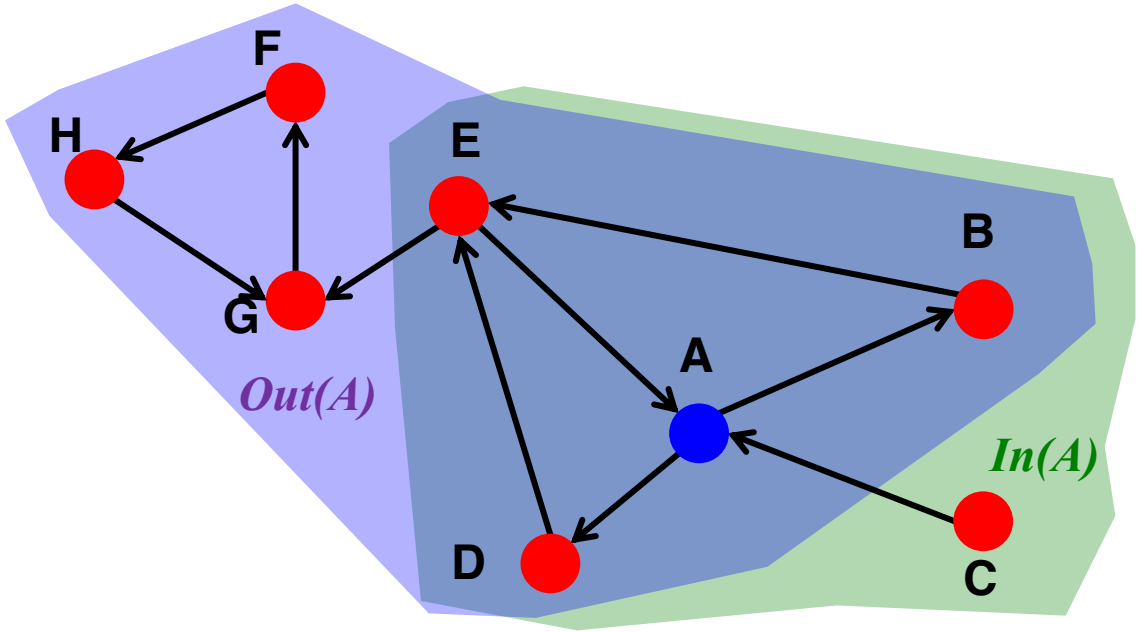
Out(v)

- **Observation:**
  - ***Out(v)*** … nodes that can be reached from v (w/ BFS)
  - **SCC containing v is:** $Out(v) \cap In(v)$
    $= Out(v,G) \cap Out(v,G')$, where G' is G with all edge directions flipped



Out(A) ∩ In(A) = SCC

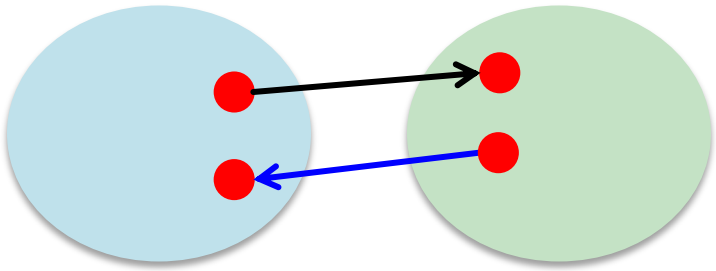
- **Example:**



  - Out(A) = {A, B, D, E, F, G, H}
  - In(A) = {A, B, C, D, E}
  - So, SCC(A) = Out(A) ∩ In(A) = {A, B, D, E}

Graph Structure of the Web

- **There is a single giant SCC**
  - That is, there won't be two SCCs
- **Why only 1 big SCC? Heuristic argument:**
  - Assume two equally big SCCs.
  - It just takes 1 page from one SCC to link to the other SCC.
  - If the two SCCs have millions of pages the likelihood of this not happening is very very small.

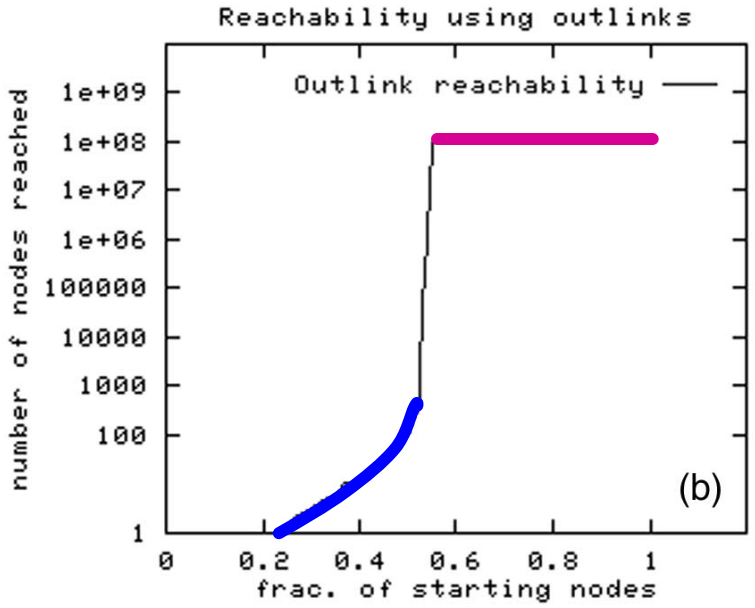


Structure of the Web

- **Directed version of the Web graph:**
  - Altavista crawl from October 1999
    - 203 million URLs, 1.5 billion links

Computation:

- Compute $IN(v)$ and $\mathrm{OUT}(v)$ by starting at random nodes.
- **Observation:** The BFS either visits **many nodes** or **very few**

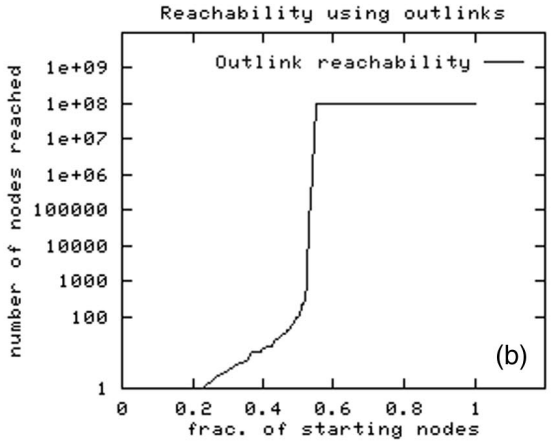


x-axis: rank
y-axis: number of reached nodes

Structure of the Web

Result: Based on IN and OUT of a random node *v*:

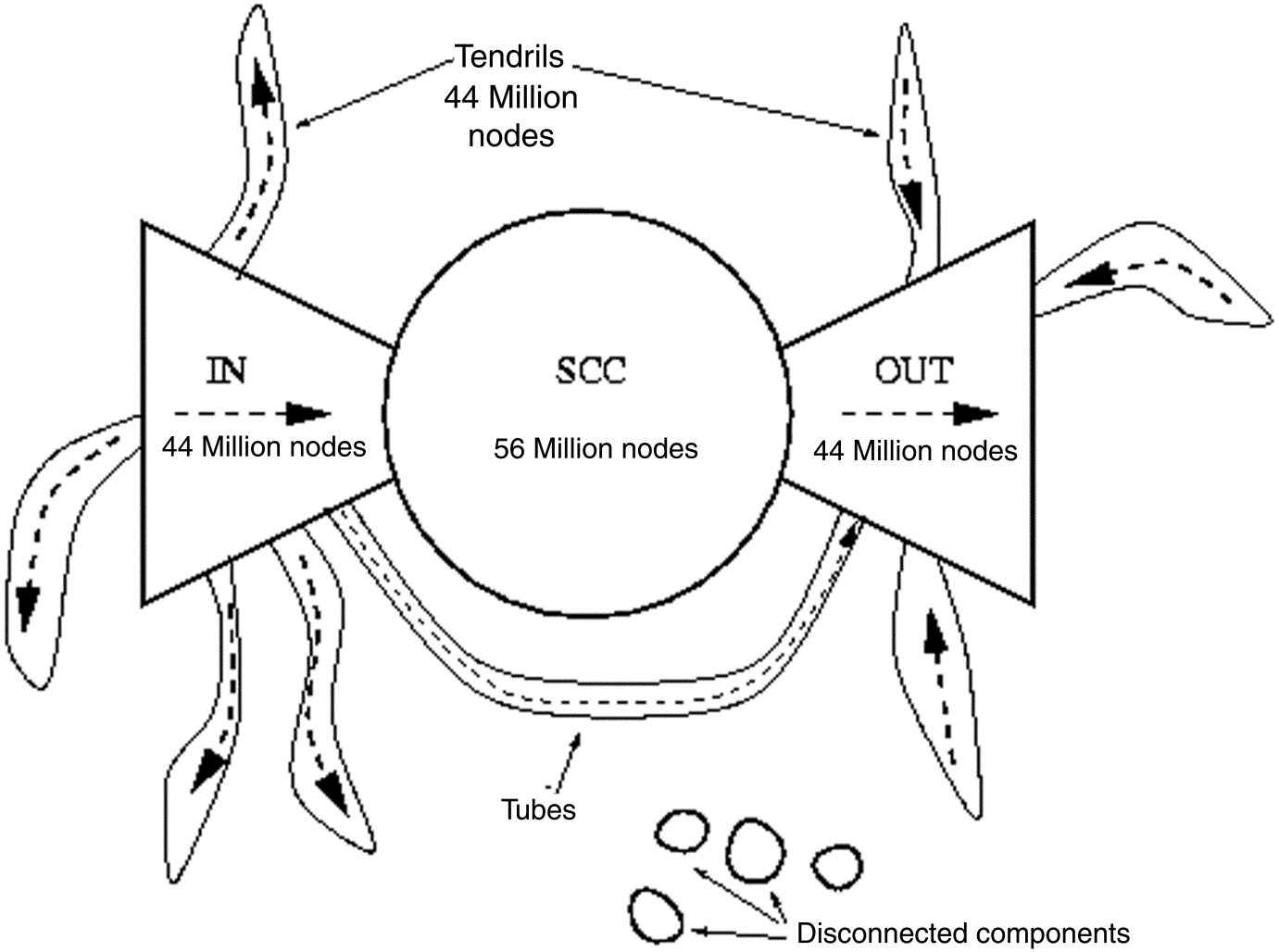
- **Out(*v*)** ≈ 100 million (**50%** nodes)
- **In(*v*)** ≈ 100 million (**50%** nodes)
- **Largest SCC:** 56 million (**28%** nodes)



x-axis: rank
y-axis: number of reached nodes

- **What does this tell us about the conceptual picture of the Web graph?**

Bowtie Structure of the Web



203 million pages, 1.5 billion links [Broder et al. 2000]

How to Organize the Web

PageRank

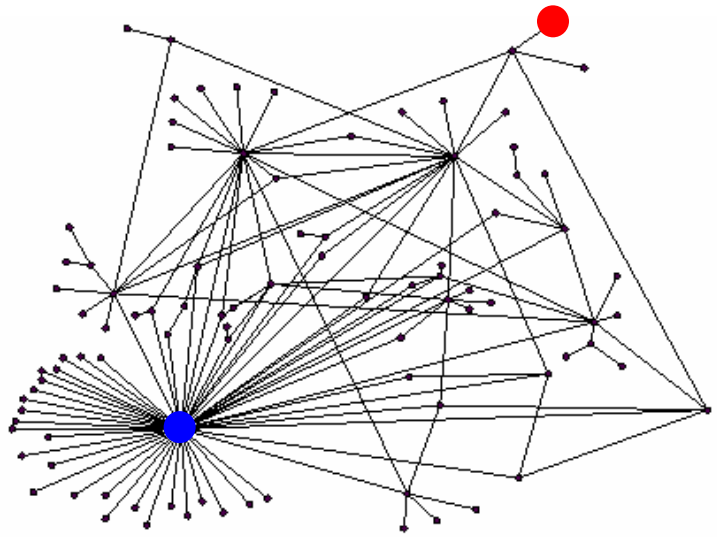
(aka the Google Algorithm)

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.
- **So, let's rank the pages using the web graph link structure!**



Link Analysis Algorithms

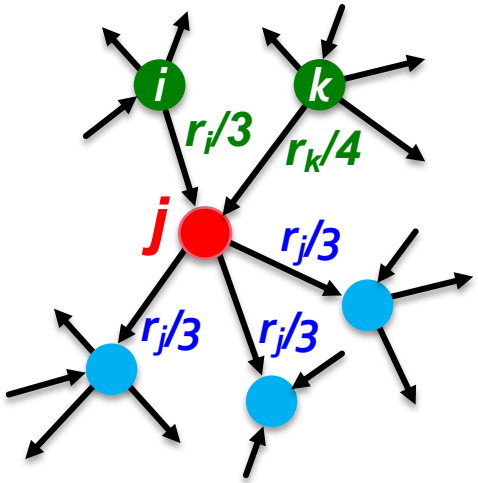
- **We will cover the following Link Analysis approaches** to computing **importance** of nodes in a graph:
  - Page Rank
  - Random Walk with Restarts
  - SimRank

Links as Votes

- **Idea: Links as votes**
  - **A 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 equal?**
  - Links from important pages count more
  - Recursive question!

PageRank: The "Flow" Model

- **A "vote" from an important page is worth more:**
  - Each link's vote is proportional to the **importance** of its source page
  - If page ***i*** with importance r_i has d_i out-links, each link gets r_i / d_i votes
  - Page ***j***'s own importance r_j is the sum of the votes on its in-links



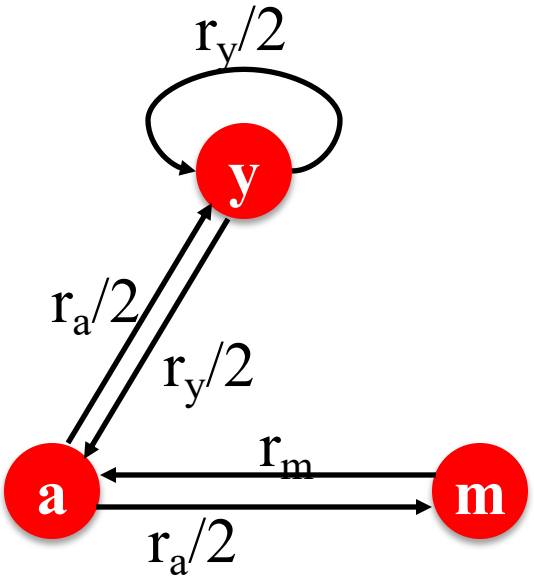
$$r_j = r_i/3 + r_k/4$$

PageRank: The "Flow" Model

- **A page is important if it is pointed to by other important pages**
- **Define a "rank" r_j for node j**

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

d_i … out-degree of node i



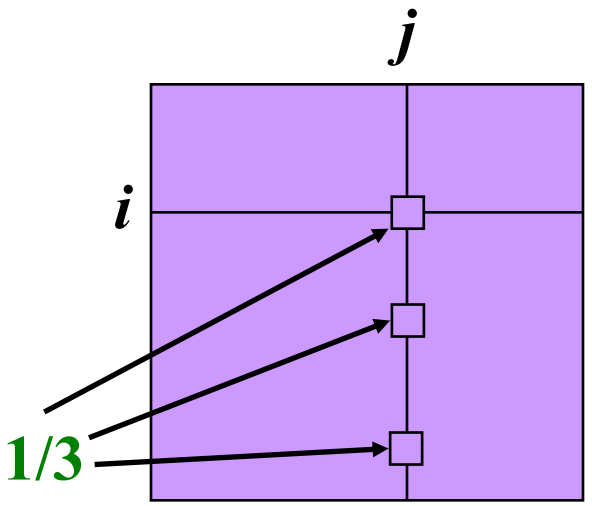
"Flow" equations:

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

You might wonder: Let's just use Gaussian elimination to solve this system of linear equations. Bad idea (G is **too** large!)

PageRank: Matrix Formulation

- **Stochastic adjacency matrix M**
  - Let page j have d_j out-links
  - If $j \rightarrow i$, then $M_{ij} = \frac{1}{d_j}$
    - M is a **column stochastic matrix**
      - **Columns** sum to **1**



- **Rank vector r:** An entry per page
  - r_i is the importance score of page i
  - $\sum_i r_i = 1$
- **The flow equations can be written**

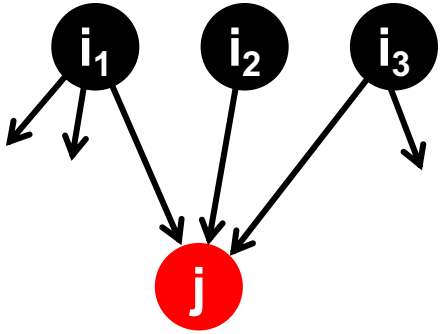
$$r = M \cdot r$$

M

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

Random Walk Interpretation

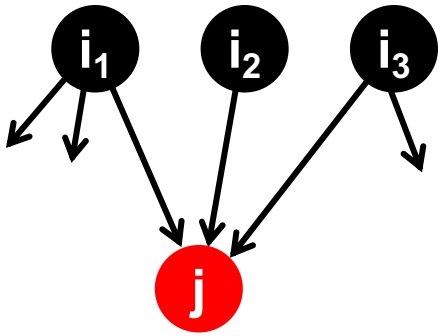
- **Imagine a random web surfer:**
  - At any time t, surfer is on some page i
  - At time $t+1$, the surfer follows an out-link from i uniformly at random
  - Ends up on some page j linked from i
  - Process repeats indefinitely



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

- **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

The Stationary Distribution



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

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

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

- 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, $\boldsymbol{r}$ is a stationary distribution for the random walk**

PageRank: How to solve?

PageRank: How to solve?

Given a web graph with *n* nodes, where the nodes are pages and edges are hyperlinks

- Assign each node an initial page rank
- Repeat until convergence ($\Sigma_i |r_i^{(t+1)} - r_i^{(t)}| < \varepsilon$)
  - Calculate the page rank of each node

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

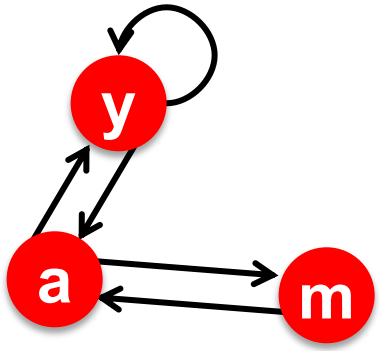
d_i …. out-degree of node i

PageRank: How to solve?

- **Power Iteration:**
  - Set $r_j \leftarrow 1/N$
  - **1:** $r'_j \leftarrow \sum_{i \to j} \frac{r_i}{d_i}$
  - **2:** $r \leftarrow r'$
  - If $|r - r'| > \varepsilon$: 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

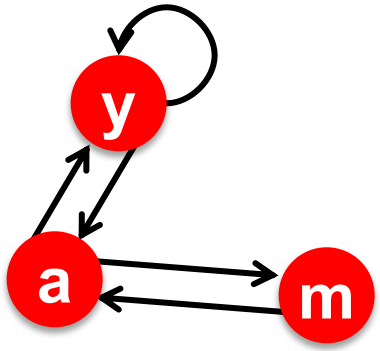
$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$

PageRank: How to solve?

- **Power Iteration:**
  - Set $r_j \leftarrow 1/N$
  - **1:** $r'_j \leftarrow \sum_{i \to j} \frac{r_i}{d_i}$
  - **2:** $r \leftarrow r'$
  - If $|r - r'| > \varepsilon$: goto **1**
- **Example:**



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

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

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

Iteration 0, 1, 2, …

PageRank: Three Questions

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

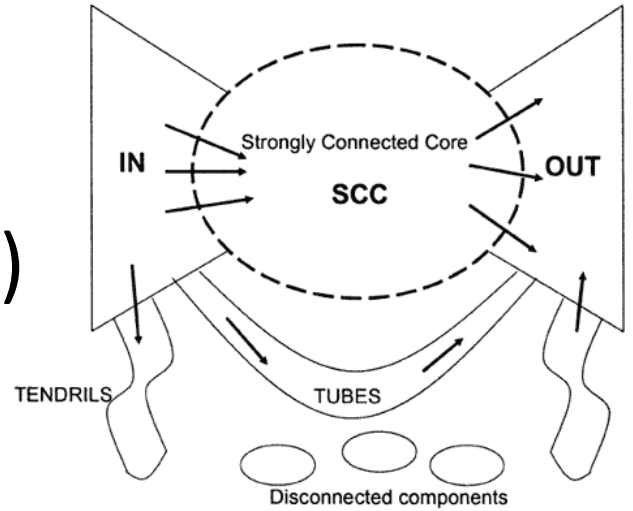
or equivalently $r = Mr$

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

RageRank: Problems

Two problems:

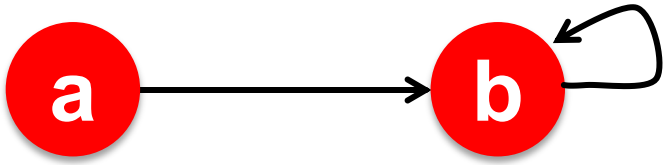
- **(1)** Some pages are **dead ends** (have no out-links)
  - Such pages cause importance to "leak out"



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

Does this converge to what we want?

- **The "Spider trap" problem:**



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

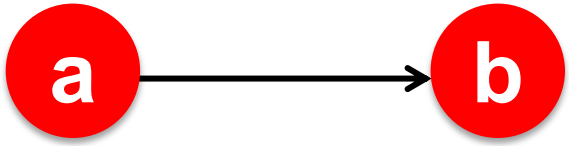
- **Example:**

Iteration: 0, 1, 2, 3…

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

Does it converge to what we want?

- **The "Dead end" problem:**



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

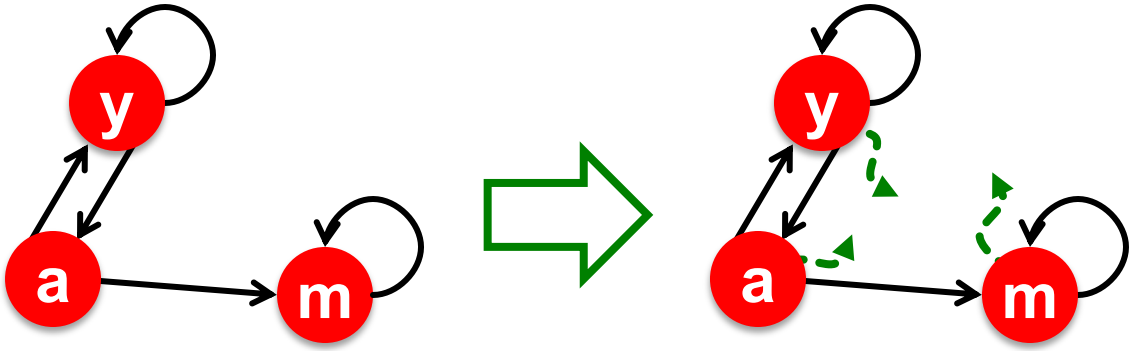
- **Example:**

Iteration: 0, 1, 2, 3…

$$\begin{matrix} r_a \\ r_b \end{matrix} = \left[\begin{array}{c|c|c|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}\right]$$

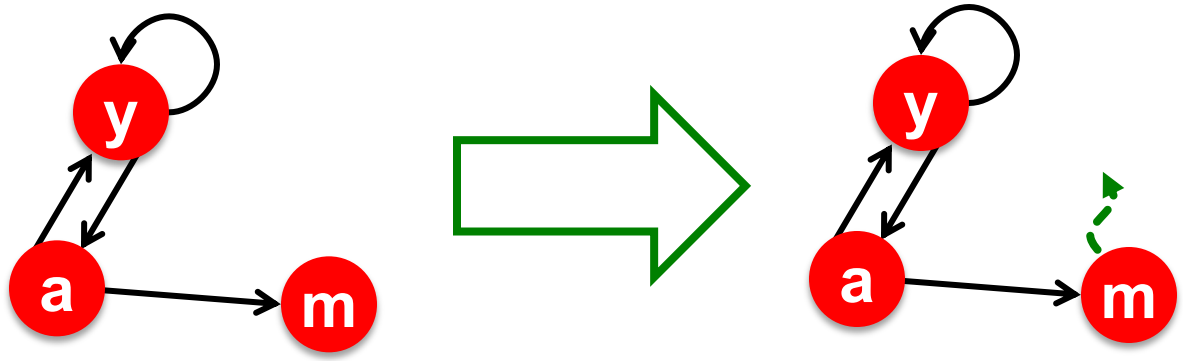
Solution to Spider Traps

- **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 a 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**



Solution to Dead Ends

- **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

Final PageRank Equation

- **Google's solution: 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]

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

d_i ... out-degree of node i

The above formulation assumes that $\boldsymbol{M}$ has no dead ends. We can either preprocess matrix $\boldsymbol{M}$ **(bad!)** or explicitly follow random teleport links with probability 1.0 from dead-ends. See P. Berkhin, *A Survey on PageRank Computing*, Internet Mathematics, 2005.

The PageRank Algorithm

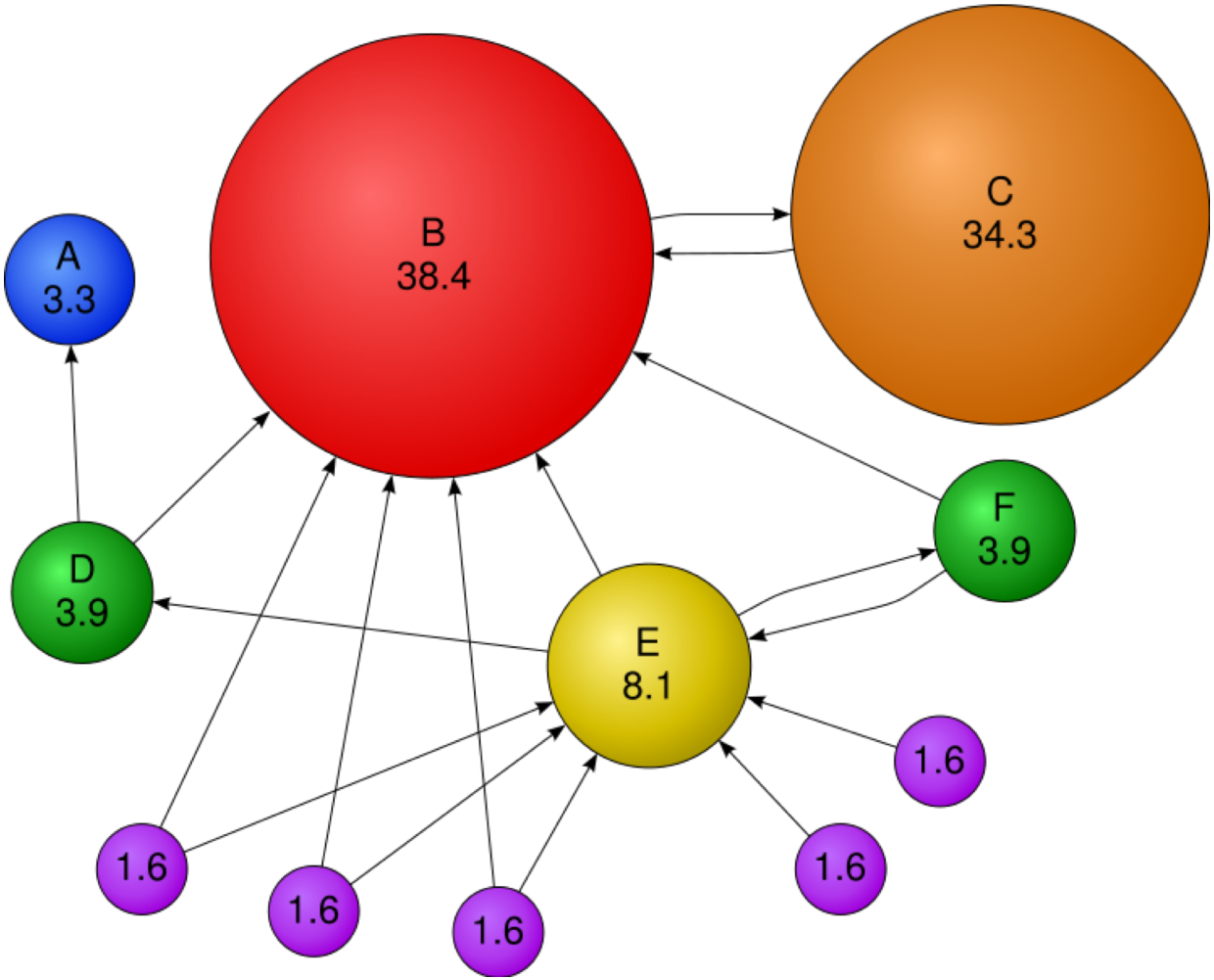
- **Input: Graph G and parameter β**
  - Directed graph G with **spider traps** and **dead ends**
  - Parameter β
- **Output: PageRank vector r**
  - **Set:** $r_j^{(0)} = \frac{1}{N}, \quad t = 1$
  - **do:**
    - $\forall j$: $r'^{(t)}_j = \sum_{i \to j} \beta \frac{r_i^{(t-1)}}{d_i}$

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

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

Example

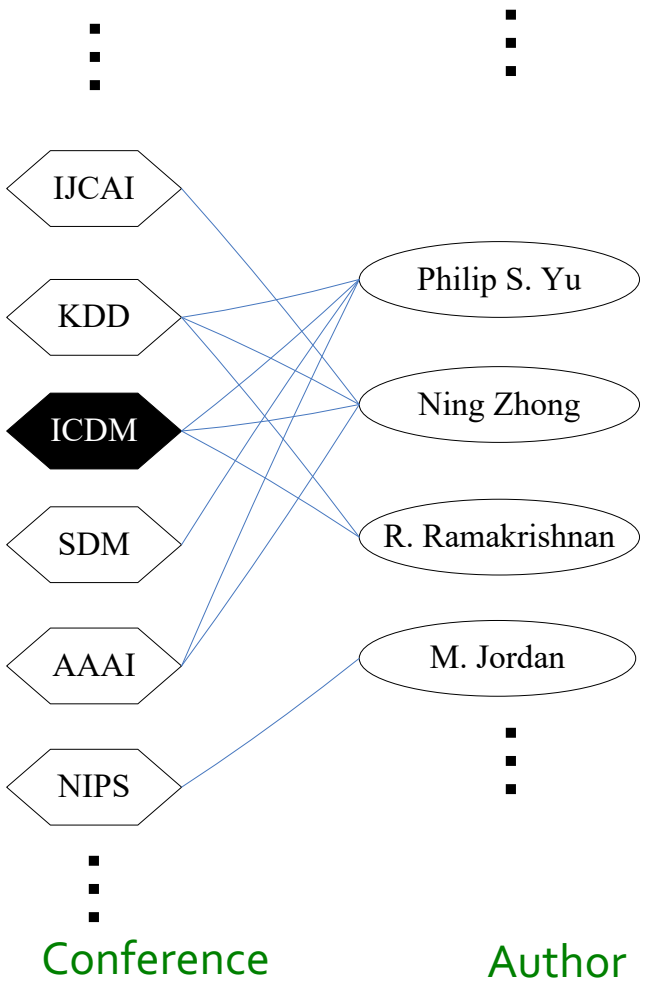
Node size proportional to the PageRank score



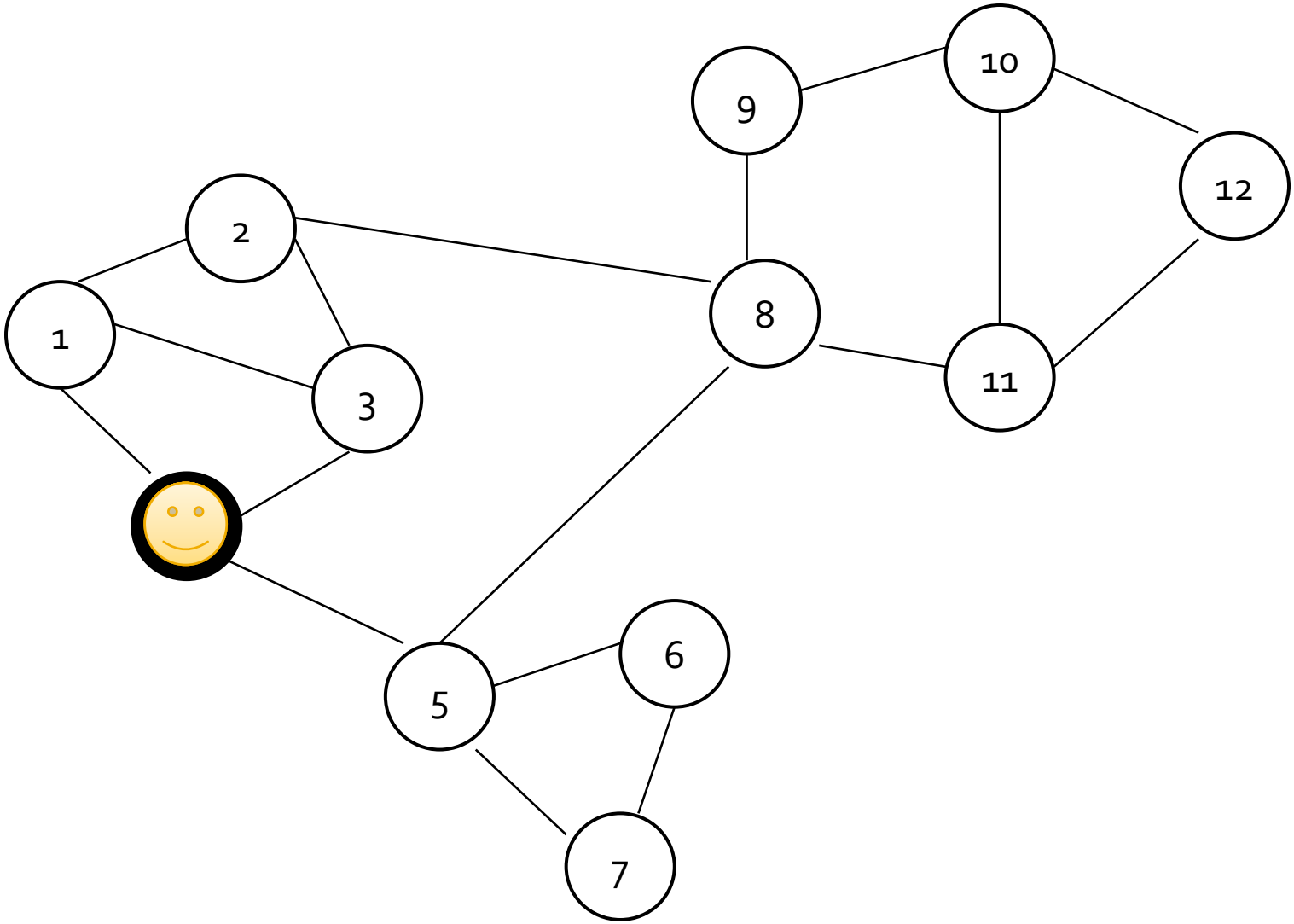
Random Walk with Restarts and Personalized PageRank

Example Application: Graph Search

- **Given:**
  Conferences-to-authors graph
- **Goal:**
  Proximity on graphs
  - Q: What is most related conference to ICDM?



Random Walk with Restarts



Personalized PageRank

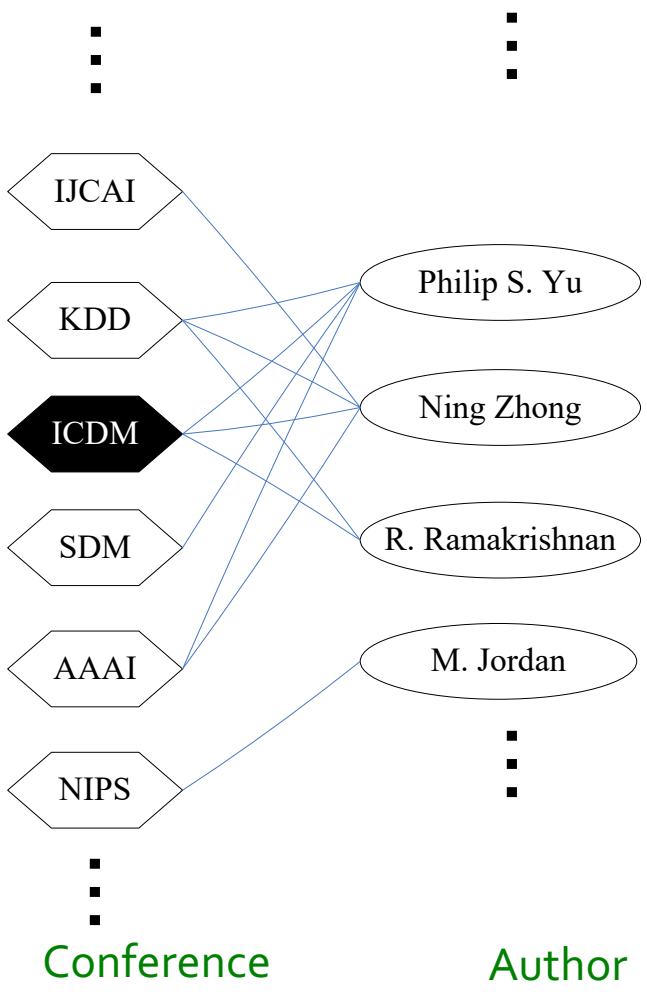
- **Goal:** Evaluate pages not just by popularity but by how close they are to the topic
- **Teleporting can go to:**
  - **Any page with equal probability**
    - PageRank (we used this so far)
  - **A topic-specific set of "relevant" pages**
    - Topic-specific (personalized) PageRank (S ...teleport set)

$$M'_{ij} = \beta M_{ij} + (1-\beta)/|S| \quad \text{if } i \in S$$
$$= \beta M_{ij} \quad \text{otherwise}$$

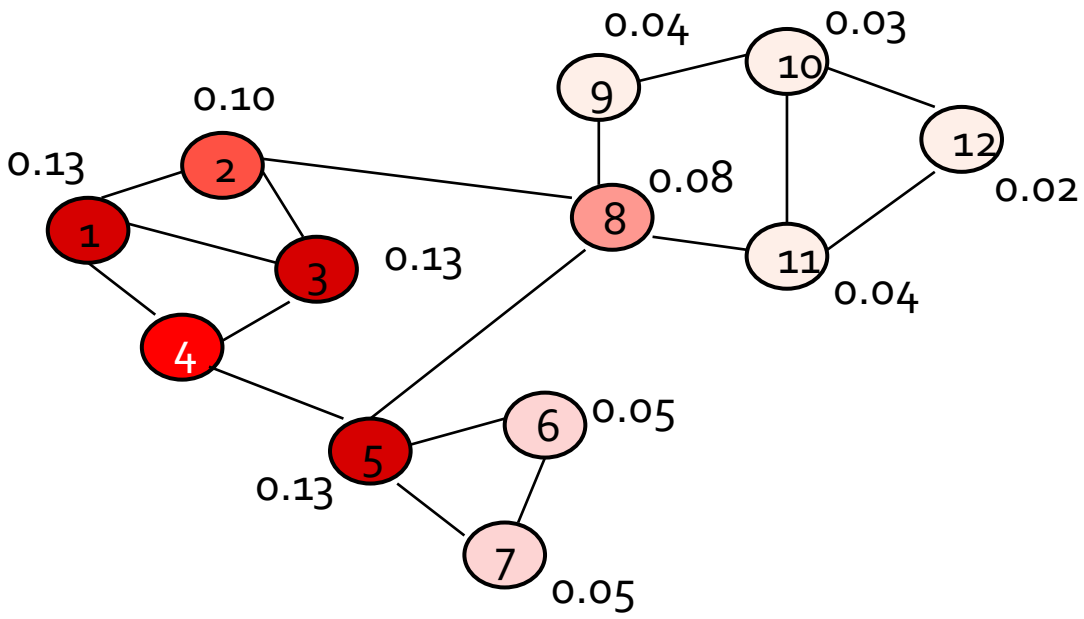
  - **A single page/node ($|S| = 1$),**
    - Random Walk with Restarts

PageRank: Applications

- **Graphs and web search:**
  - Ranks nodes by "importance"
- **Personalized PageRank:**
  - Ranks proximity of nodes to the teleport set S
- **Proximity on graphs:**
  - **Q:** What is most related conference to **ICDM**?
  - **Random Walks with Restarts**
    - Teleport back to the starting node: $S = \{$ **single node** $\}$



Random Walk with Restarts



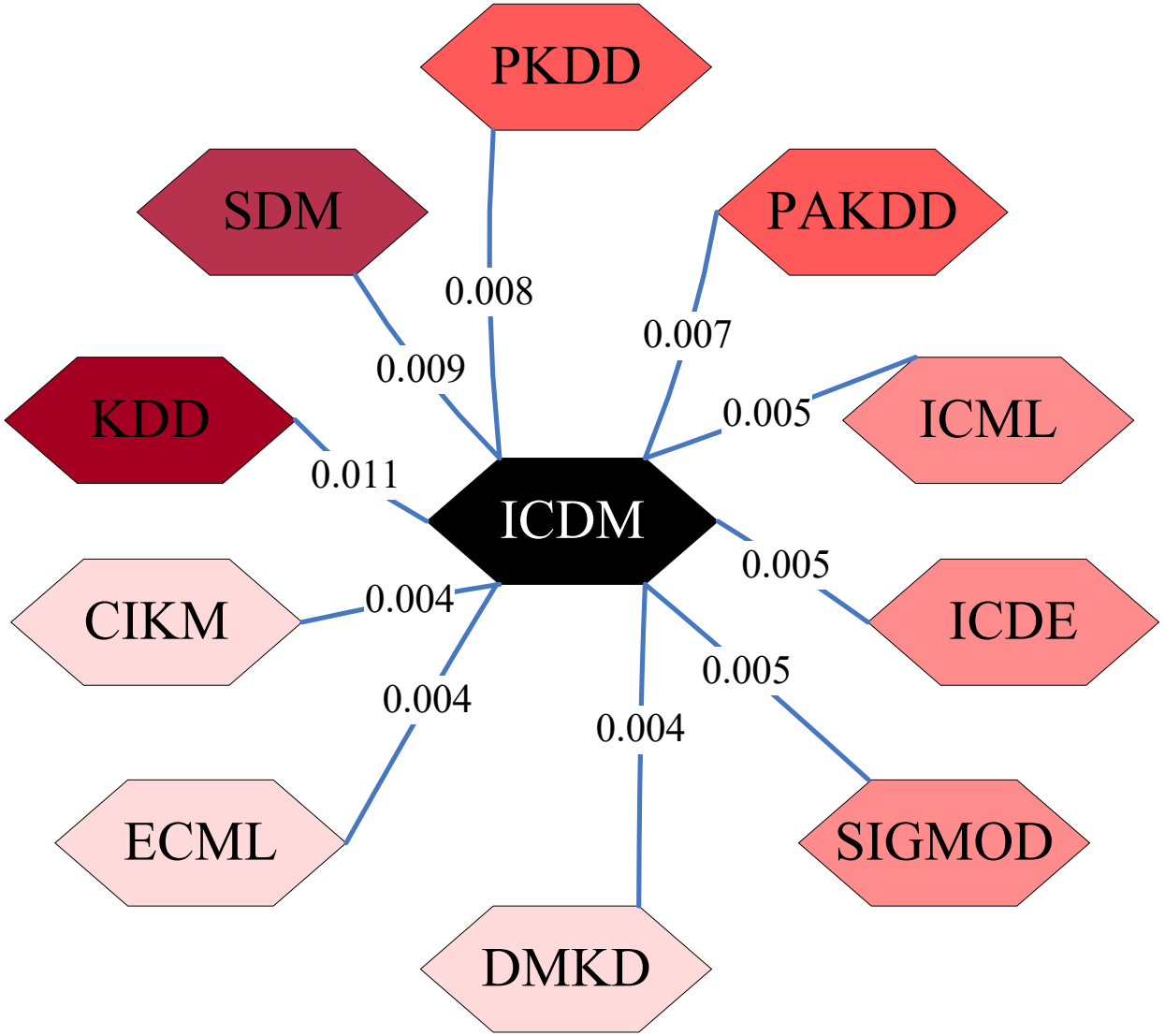
	Node 4
Node 1	0.13
Node 2	0.10
Node 3	0.13
Node 4	/
Node 5	0.13
Node 6	0.05
Node 7	0.05
Node 8	0.08
Node 9	0.04
Node 10	0.03
Node 11	0.04
Node 12	0.02

Ranking vector

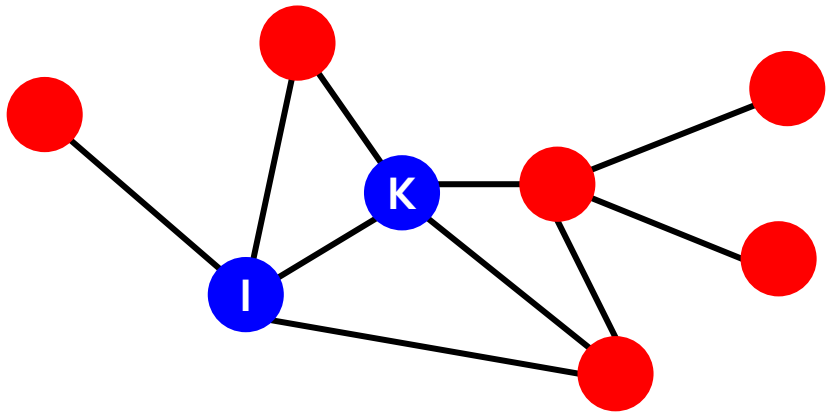
S={4}

Notice: Nearby nodes have higher scores (are more red)

Most related conferences to ICDM



Personalized PageRank



Graph of CS conferences

Q: Which conferences are closest to KDD & ICDM?

A: Personalized PageRank with teleport set S={KDD, ICDM}

SimRank

Problem: Measuring Similarity

- The problem of measuring "similarity" of objects (nodes in a graph) arises in many applications
- The approach should be applicable in any domain with object-to-object relationships
- **SimRank intuition: Two objects are similar if they are related to similar objects.**

SimRank

- We model objects and relationships as a directed graph $G = (V, E)$
- For a node *v* in a graph, we denote by $I(v)$ and $O(v)$ the set of in-neighbors and out-neighbors
- Basic SimRank Equation
  - If $a = b$ then $s(a, b)$ is defined to be 1. Otherwise,

$$s(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)||I(b)|} \sum_{j=1} s(I_i(a), I_j(b))$$

    where *C* is a constant between 0 and 1
  - Set $s(a, b) = 0$ when $I(a) = \emptyset$ or $I(b) = \emptyset$

SimRank: How to Compute

- Computing SimRank — Naive Method
  - $R_0(a,b)$ is initialized with the lower bound on the $s(a,b)$

$$R_0(a,b) = \begin{cases} 0 & (\text{if } a \neq b) \\ 1 & (\text{if } a = b) \end{cases}$$

  - **To compute $R_{k+1}(a,b)$ from $R_k(\cdot,\cdot)$:**

$$R_{k+1}(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} R_k(I_i(a), I_j(b)) \qquad \text{when } a \neq b$$

$$R_{k+1}(a,b) = 1 \qquad a = b$$

Example: Shopping Graph

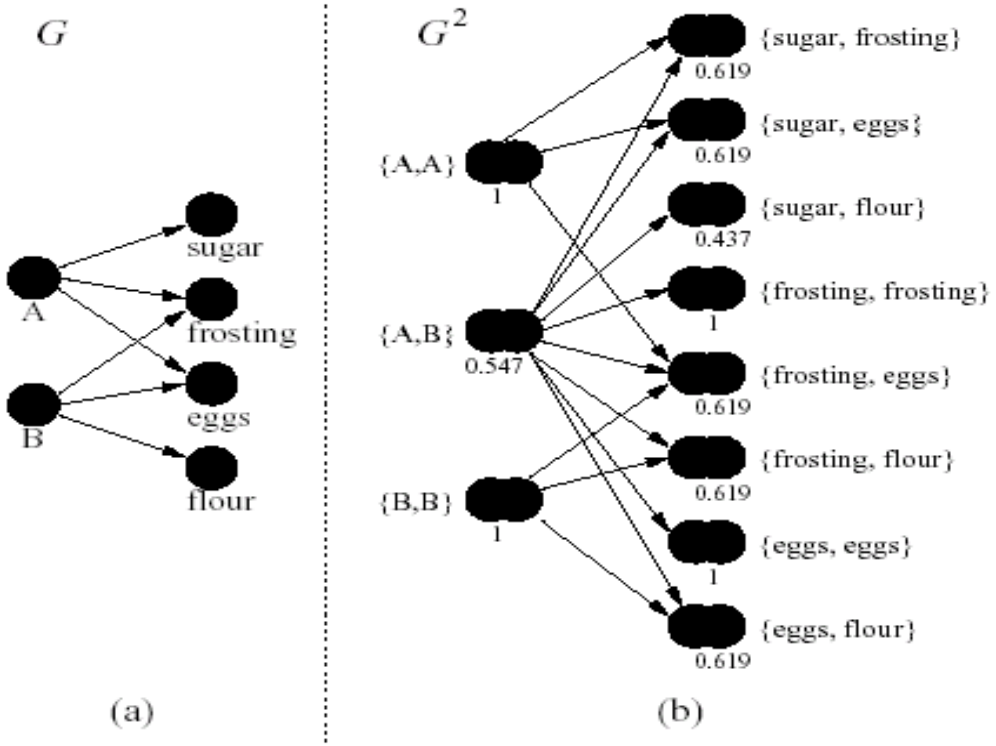


Figure 2: Shopping graph G and a simplified version of the derived node-pairs graph G^2. Bipartite SimRank scores are shown for G^2 using $C_1 = C_2 = 0.8$.