AIM3 – Scalable Data Analysis and Data Mining

Graph Mining

Christoph Boden, Sebastian Schelter, Juan Soto,
Volker Markl

Based on Material from Jure Leskovec and Jeff Ullman (Stanford)

Fachgebiet Datenbanksysteme und Informationsmanagement
Technische Universität Berlin

http://www.dima.tu-berlin.de/

Networks & Complex Systems

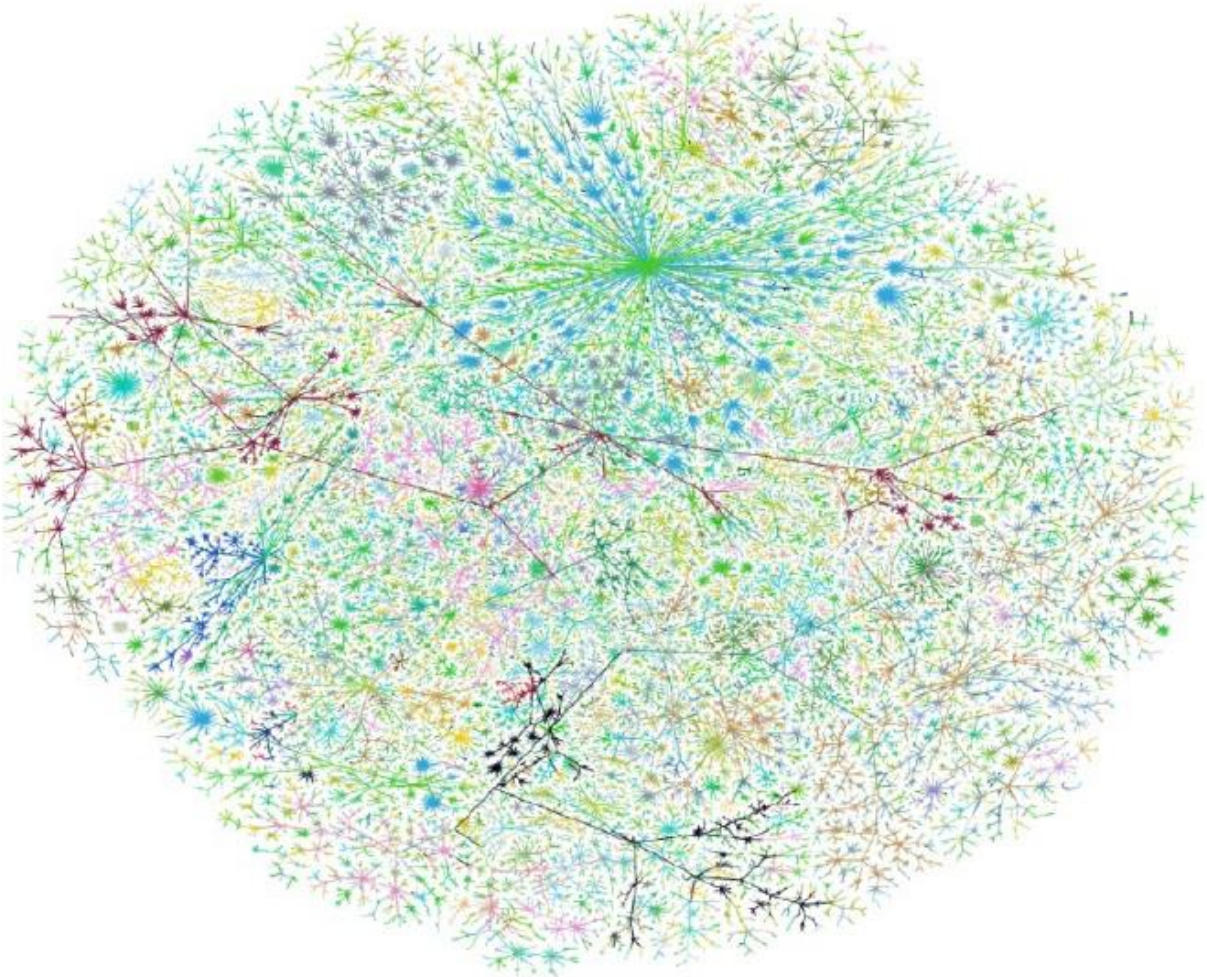
- We are surrounded by hopelessly complex systems:
  - Society is a collection of six billion individuals
  - Communication systems link electronic devices
  - Information and knowledge is organized and linked
  - Thousands of genes in our cells work together in a seamless fashion
  - Our thoughts are hidden in the connections between billions of neurons in our brain

- These systems, random looking at first, display signatures of order and self-organization

Networks

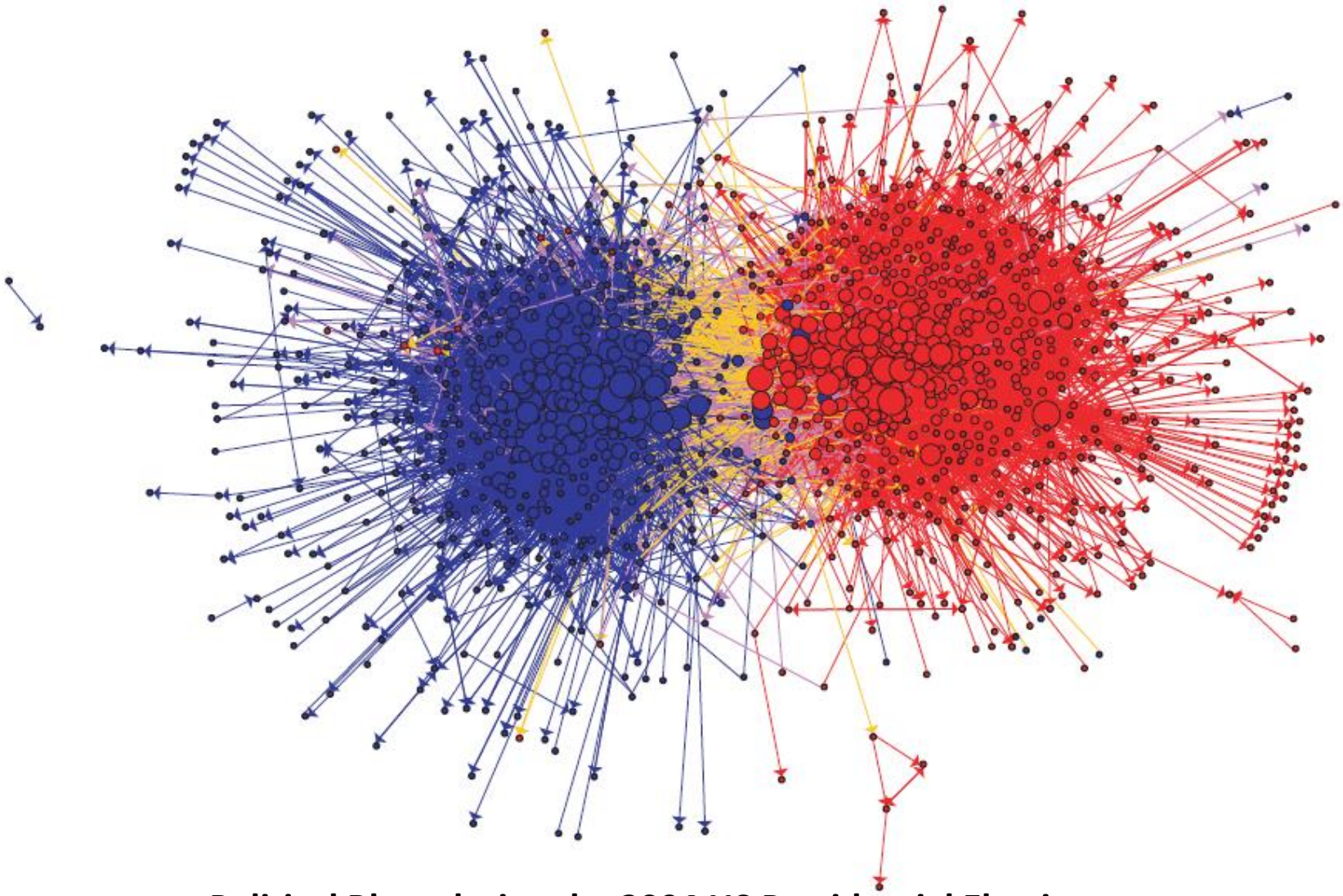
- Each such system can be represented as a **network**, that defines the **interactions** between the components

Networks: Communication



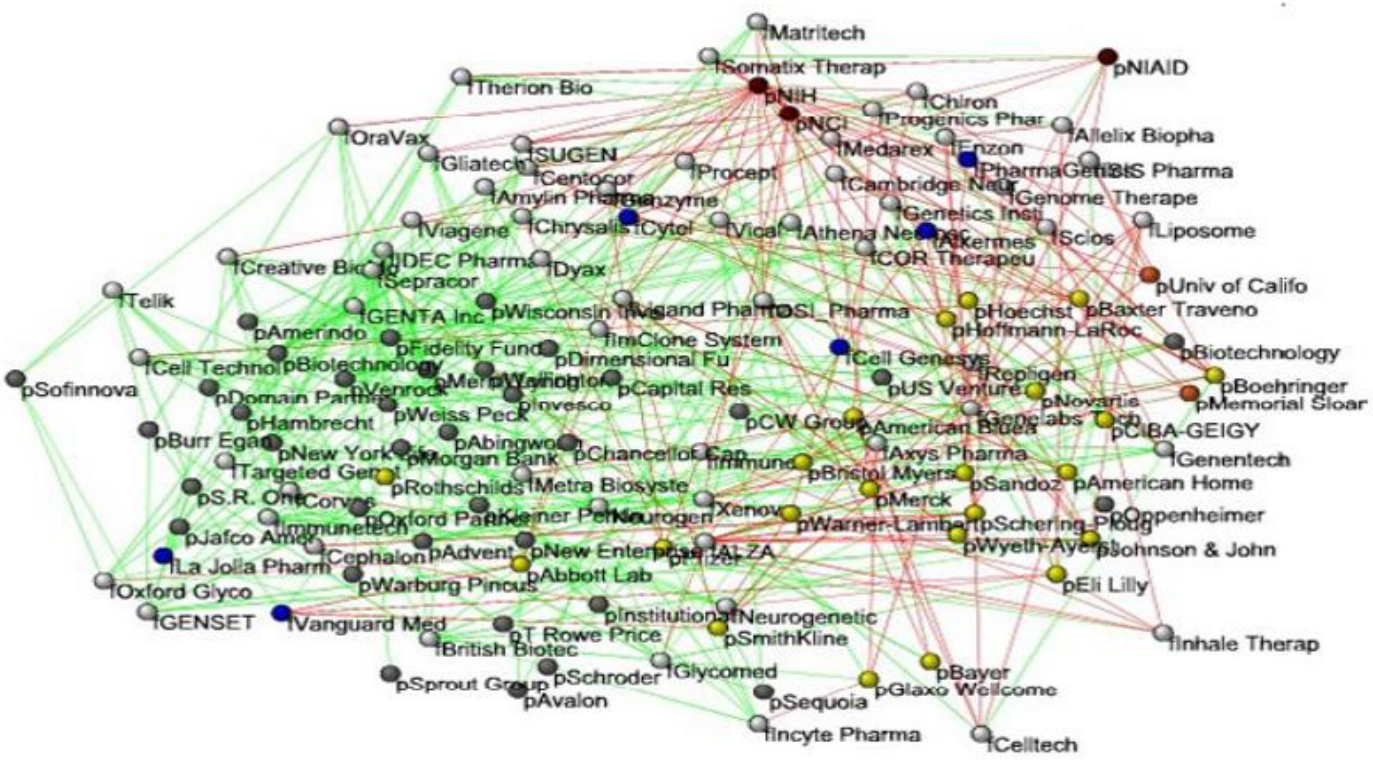
Graph of the Internet
(Autonomous Systems)

Networks: Information



Political Blogs during the 2004 US Presidential Election

Networks: Economics



Nodes:

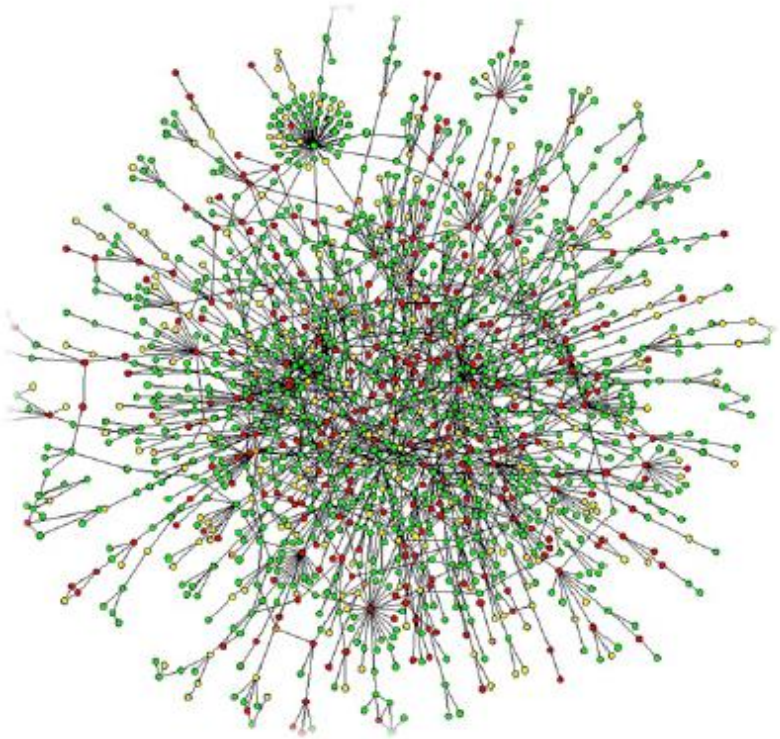
- Companies
- Investment
- Pharma
- Research Labs
- Public
- Biotechnology

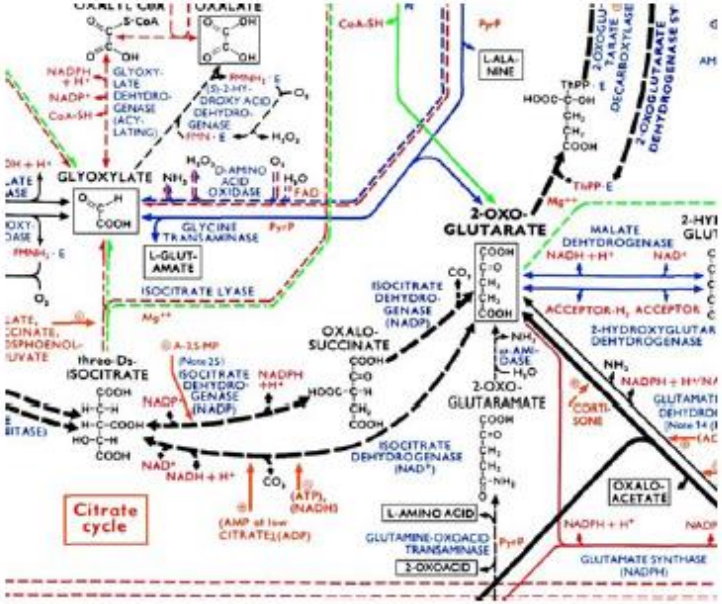
Links:

- Collaborations
- Financial
- R&D

Bio-tech companies, 1991

Network: Cells





Protein-Protein Interaction Networks:

Nodes: Proteins

Edges: 'physical' interactoins

Metabolic networks:

Nodes: Metabolites and enzymes

Edges: Chemical reactions

Why Networks?

- Behind each such system there is an intricate wiring diagram, **a network**, that defines the **interactions** between the components

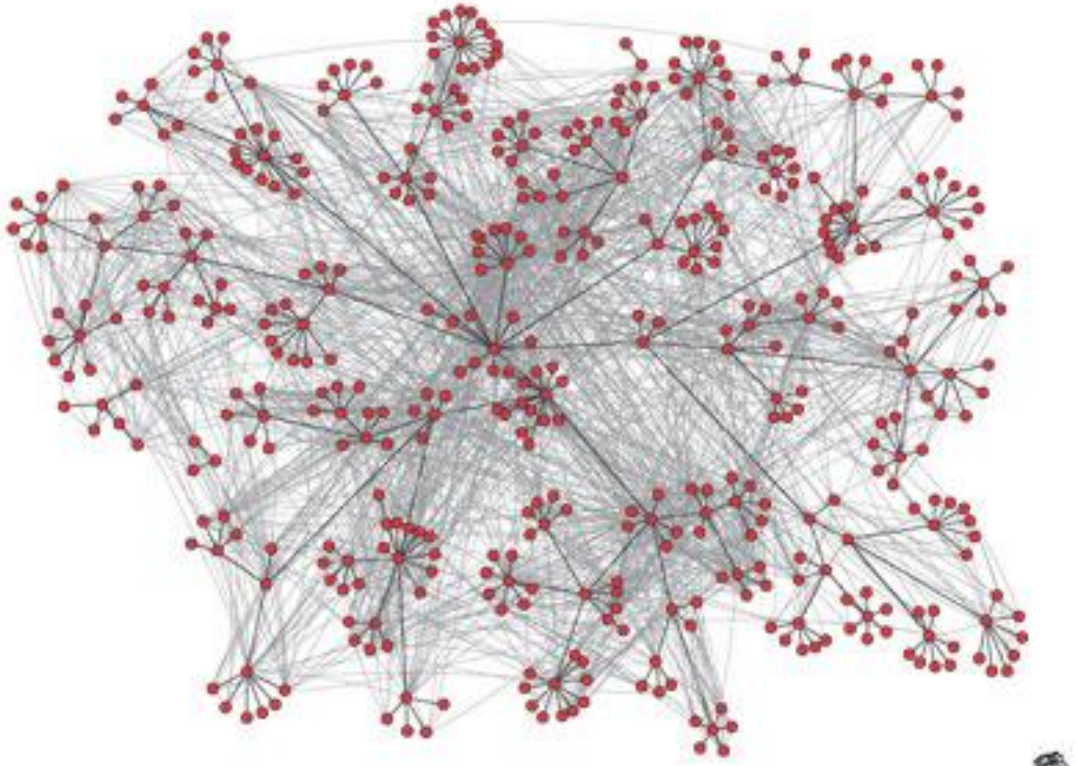
- We will never understand a complex system unless we understand the networks behind it

Why Networks? Why Now?

- **Why is the role of networks expanding?**
  - **Data availability**
    - Rise of the Web 2.0 and Social media
  - **Universality**
    - Networks from various domains of science, nature, and technology are more similar than one would expect
  - **Shared vocabulary between fields**
    - Computer Science, Social science, Physics, Economics, Statistics, Biology

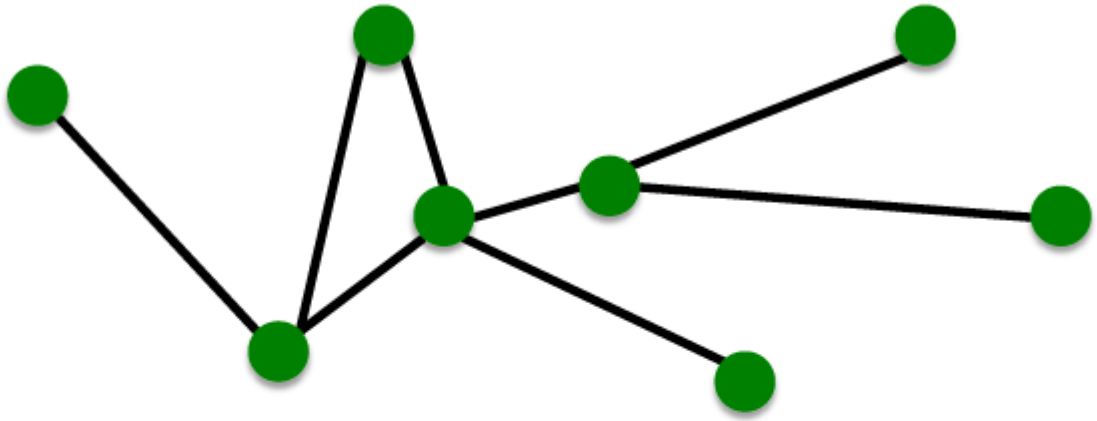
Structure of Networks

- Network is a collection of objects where some pairs of objects are connected by links



- **What is the structure of the network?**

Components of a Network



- Objects: nodes, vertices — N
- Interactions: links, edges — E
- System: network, graph — $G(N,E)$

Networks or Graphs?

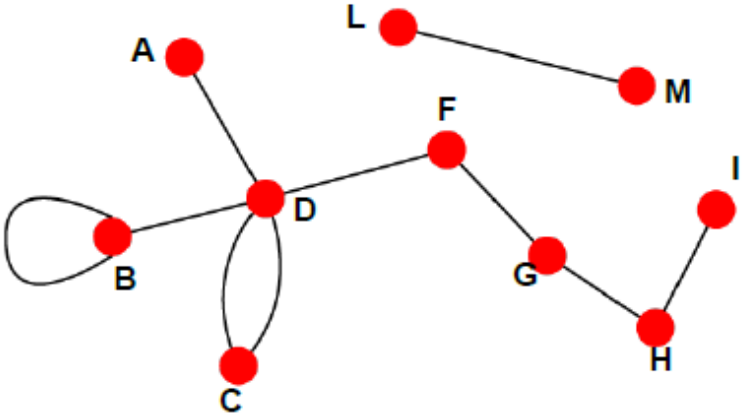
- **Network** often refers to real systems
  Web, Social network, Metabolic network
  - Language: Network, node, link

- **Graph:** mathematical representation of a network
  Web graph, Social graph (a Facebook term)
  - Language: Graph, vertex, edge

Directed vs. Undirected Data

Undirected

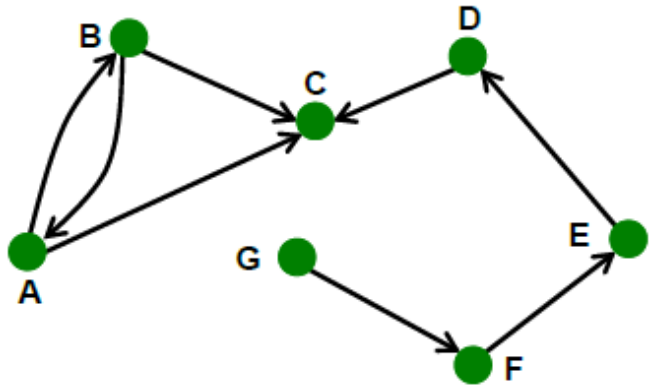
- Links: undirected (symmetrical)



- Undirected links:
  - Collaborations
  - Friendship on Facebook

Directed

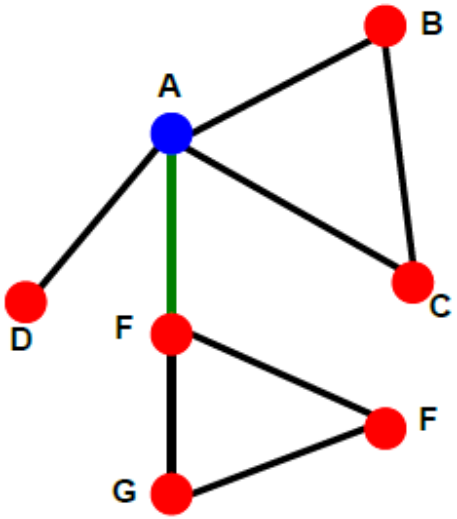
- Links: directed (arcs)

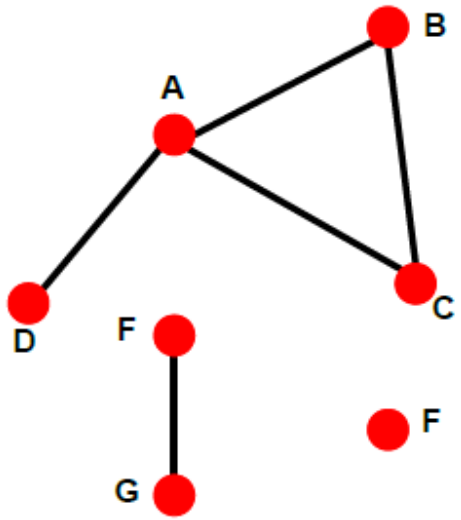


- Directed links:
  - Phone calls
  - Following on Twitter

Connectivity of Graphs

- Connected (undirected) graph:
  - Any two vertices can be joined by a path.
- A disconnected graph is made up by two or more connected components





Largest Component:
Giant Component

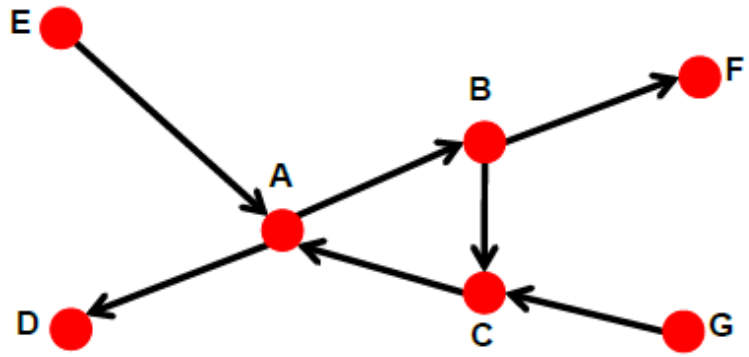
Isolated node (F)

Bridge edge: If we erase it, the graph becomes disconnected.

Articulation point: If we erase it, the graph becomes disconnected.

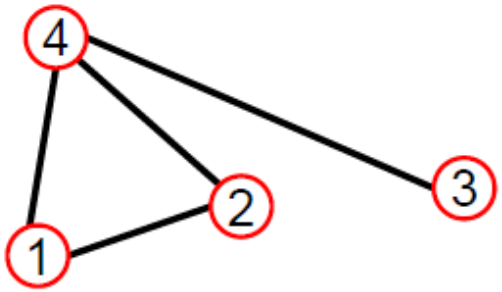
Connectivity of directed Graphs

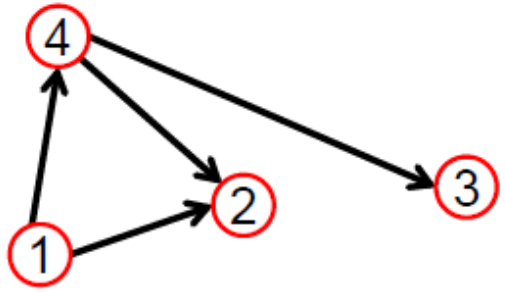
- Strongly connected directed graph
  - has a path from each node to every other node and vice versa (e.g., A-B path and B-A path)
- Weakly connected directed graph
  - is connected if we disregard the edge directions



Graph on the left is not strongly connected.

Adjecency Matrix





A_{ij}=1 if there is a link between node *i* and *j*

A_{ij}=0 if nodes *i* and *j* are not connected to each other

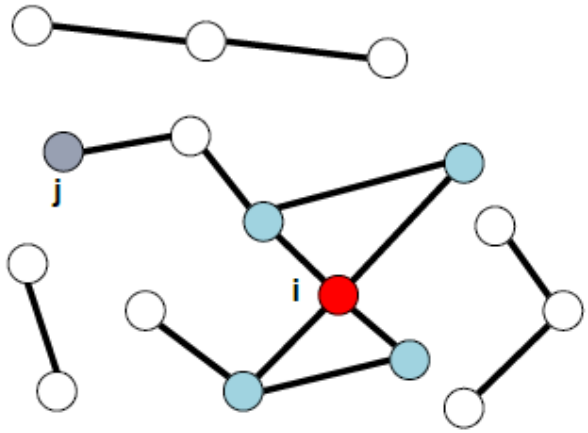
$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Note that for a directed graph (right) the matrix is not symmetric.

Node Degree

Undirected

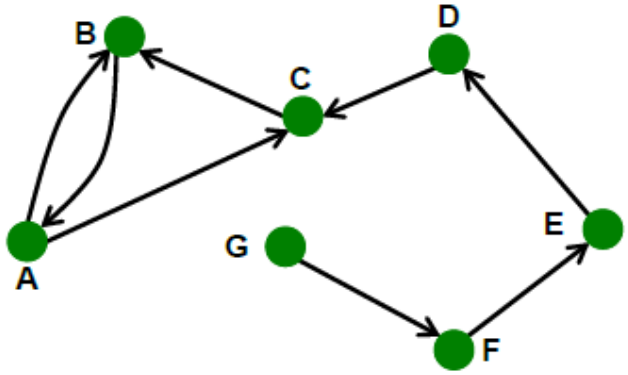


Node degree: the number of links connected to the node

$$k_i = 4$$

Avg. degree: $\overline{k} \equiv \frac{1}{N}\sum_{i=1}^{N} k_i = \frac{2E}{N}$

Directed



In directed networks we define an in-degree and out-degree.

The (total) degree of a node is the sum of in- and out-degree.

$$k_C^{in} = 2 \qquad k_C^{out} = 1 \qquad k_C = 3$$

Source: A node with $k^{in} = 0$

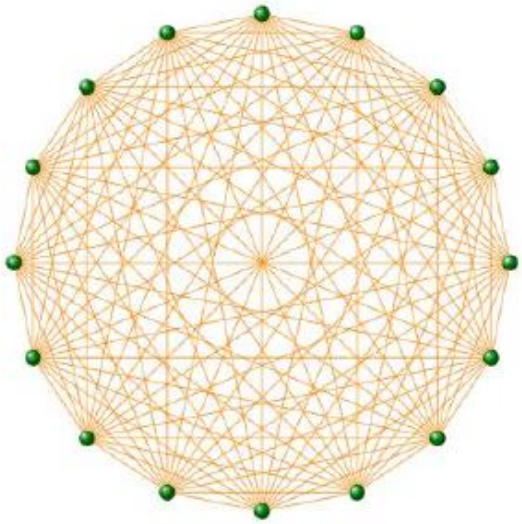
Sink: A node with $k^{out} = 0$

$$\overline{k} = \frac{E}{N} \qquad \overline{k^{in}} = \overline{k^{out}}$$

Complete Graph

The maximum number of edges in an undirected graph on N nodes is

$$E_{\max} = \binom{N}{2} = \frac{N(N-1)}{2}$$



A graph with the number of edges $E=E_{max}$ is a **complete graph**, and its average degree is $N\text{-}1$

Networks are sparse graphs

Most real-world networks are sparse

$$E << E_{max} \quad (\text{or } \bar{k} << N-1)$$

WWW (Stanford-Berkeley):	N=319,717	⟨k⟩=9.65
Social networks (LinkedIn):	N=6,946,668	⟨k⟩=8.87
Communication (MSN IM):	N=242,720,596	⟨k⟩=11.1
Coauthorships (DBLP):	N=317,080	⟨k⟩=6.62
Internet (AS-Skitter):	N=1,719,037	⟨k⟩=14.91
Roads (California):	N=1,957,027	⟨k⟩=2.82
Protein (S. Cerevisiae):	N=1,870	⟨k⟩=2.39

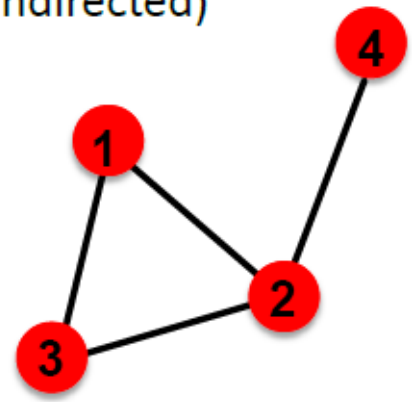
(Source: *Leskovec et al., Internet Mathematics, 2009*)

Consequence: Adjacency matrix is filled with zeros!

(Density (E/N^2): WWW=1.51×10^{-5}, MSN IM = 2.27×10^{-8})

Weight

- **Unweighted**
  (undirected)



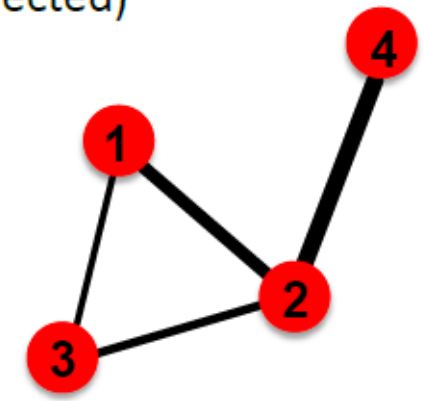
$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

$$E = \frac{1}{2}\sum_{i,j=1}^{N} A_{ij} \qquad \bar{k} = \frac{2E}{N}$$

Friendships, WWW

- **Weighted**
  (undirected)



$$A_{ij} = \begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

$$E = \frac{1}{2}\sum_{i,j=1}^{N} nonzero(A_{ij}) \qquad \bar{k} = \frac{2E}{N}$$

Call graph, Email graph

Network Representations

WWW >> directed multigraph with self-interactions

Facebook friendships >> undirected, unweighted

Citation networks >> unweighted directed acyclic

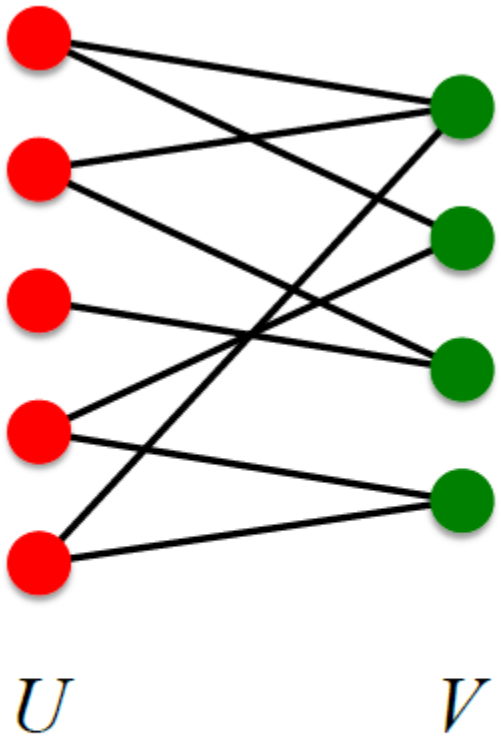
Collaboration networks >> undirected multigraph or weighted

Mobile phone calls >> directed, (weighted?) multigraph

Protein Interactions >> undirected, unweighted with self-interactions

BiPartite Graphs

- **Bipartite graph** is a graph whose nodes can be divided into two disjoint sets U and V such that every link connects a node in U to one in V; that is, U and V are independent sets.



- **Examples:**
  - Authors-to-papers
  - Movies-to-Actors
  - Users-to-Movies
- **"Folded" networks**
  - Author collaboration networks
  - Actor collaboration networks

Properties of Networks

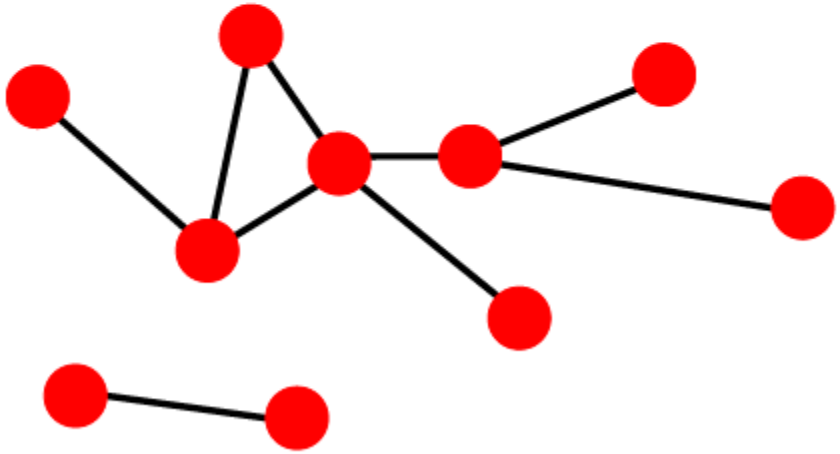
How to Characterize a Network

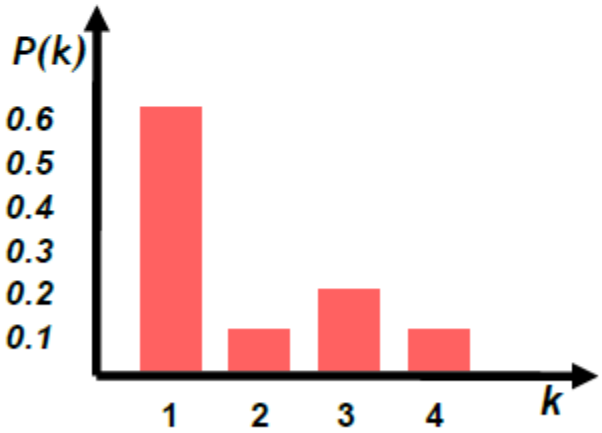
Degree Distribution

- **Degree distribution $P(k)$:** Probability that a randomly chosen node has degree k

N_k = # nodes with degree k

$P(k) = N_k / N$ → **plot**



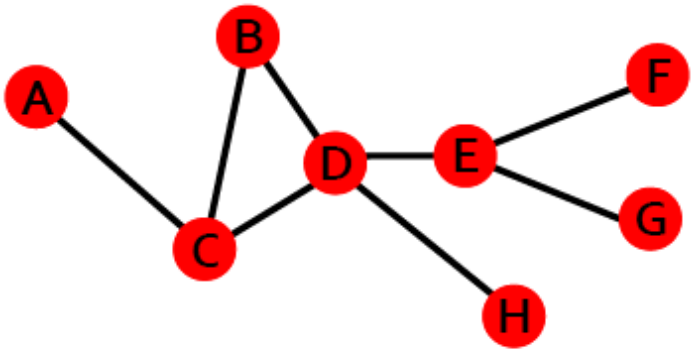


Paths in a Graph

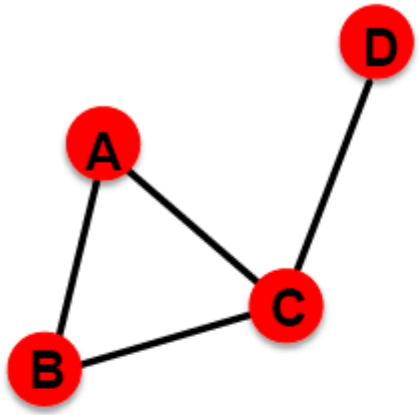
- A ***path*** is a sequence of nodes in which each node is adjacent to the next one

$$P_n = \{i_0, i_1, i_2, \ldots, i_n\} \qquad P_n = \{(i_0, i_1), (i_1, i_2), (i_2, i_3), \ldots, (i_{n-1}, i_n)\}$$

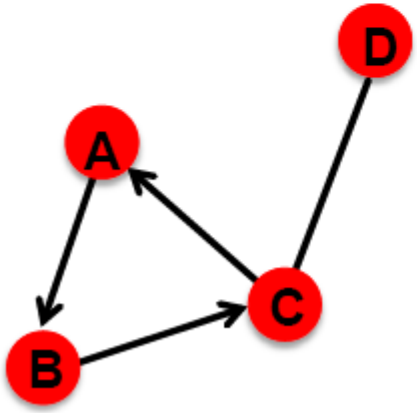
- Path can intersect itself and pass through the same edge multiple times
  - E.g.: ACBDCDEG
  - In a directed graph a path can only follow the direction of the "arrow"



Distance in a Graph

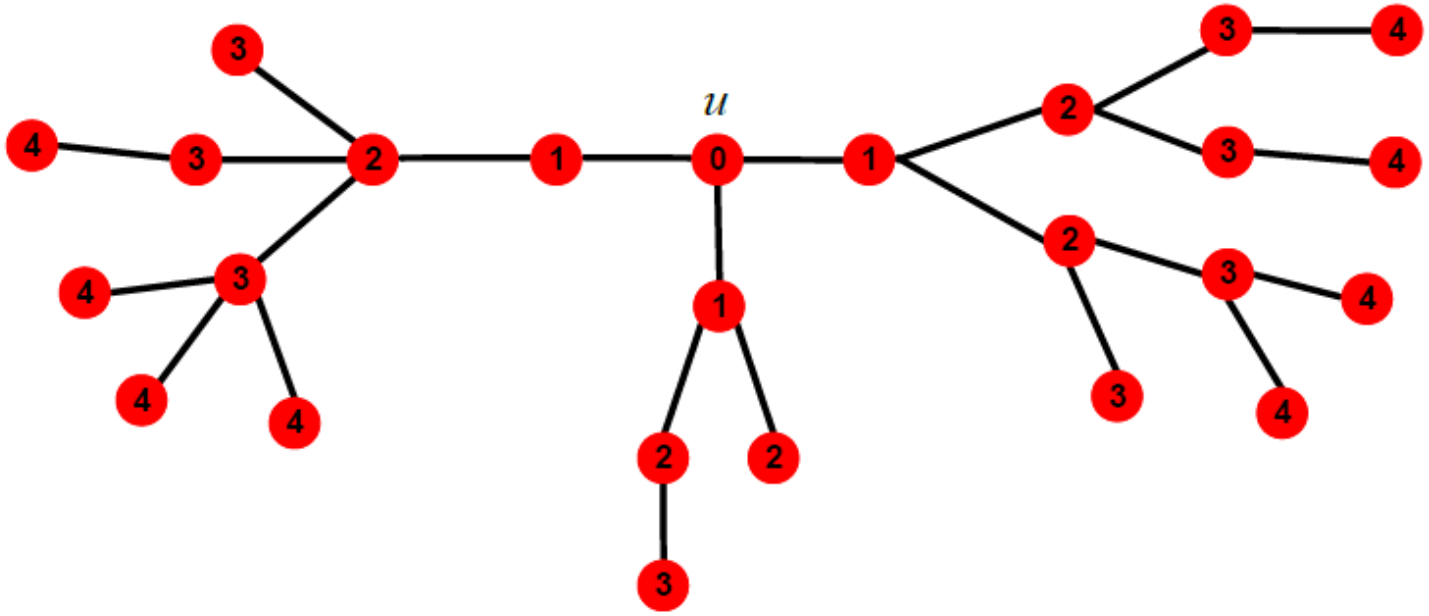


- **Distance (shortest path, geodesic)** between a pair of nodes is defined as the number of edges along the shortest path connecting the nodes.
  - *If the two nodes are disconnected, the distance is defined as infinite



- In **directed graphs** paths need to follow the direction of the arrows.
  - Consequence: Distance is not symmetric: $h(A,C) \neq h(C,A)$

- **Breath-First Search:**
  - Start with node u, mark it to be at distance $h_u(u)=0$, add u to the queue
  - While queue not empty:
    - Take node v off the queue, put it's unmarked neighbor w into the queue and mark $h_u(w)=h_u(v)+1$



Network Diameter

- **Diameter:** the maximum (shortest path) distance between any pair of nodes in the graph
- **Average path length/distance** for a connected graph (component) or a strongly connected (component of a) directed graph

$$\bar{h} = \frac{1}{2E_{\max}} \sum_{i,j \neq i} h_{ij}$$

where h_{ij} is the distance from node i to node j

  - Many times we compute the average only over the connected pairs of nodes (*i.e.*, we ignore "infinite" paths)

Six Degrees of Separation

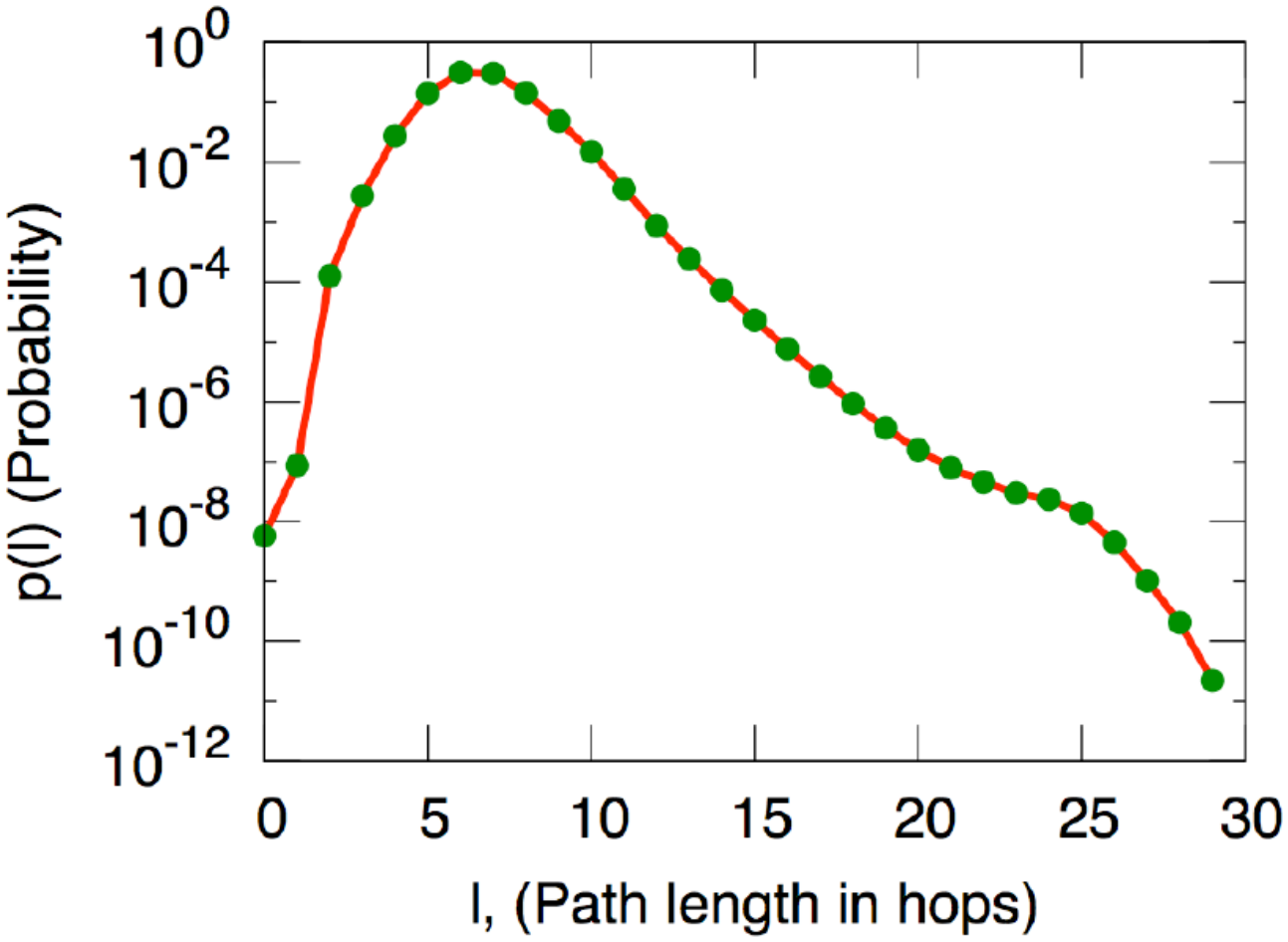
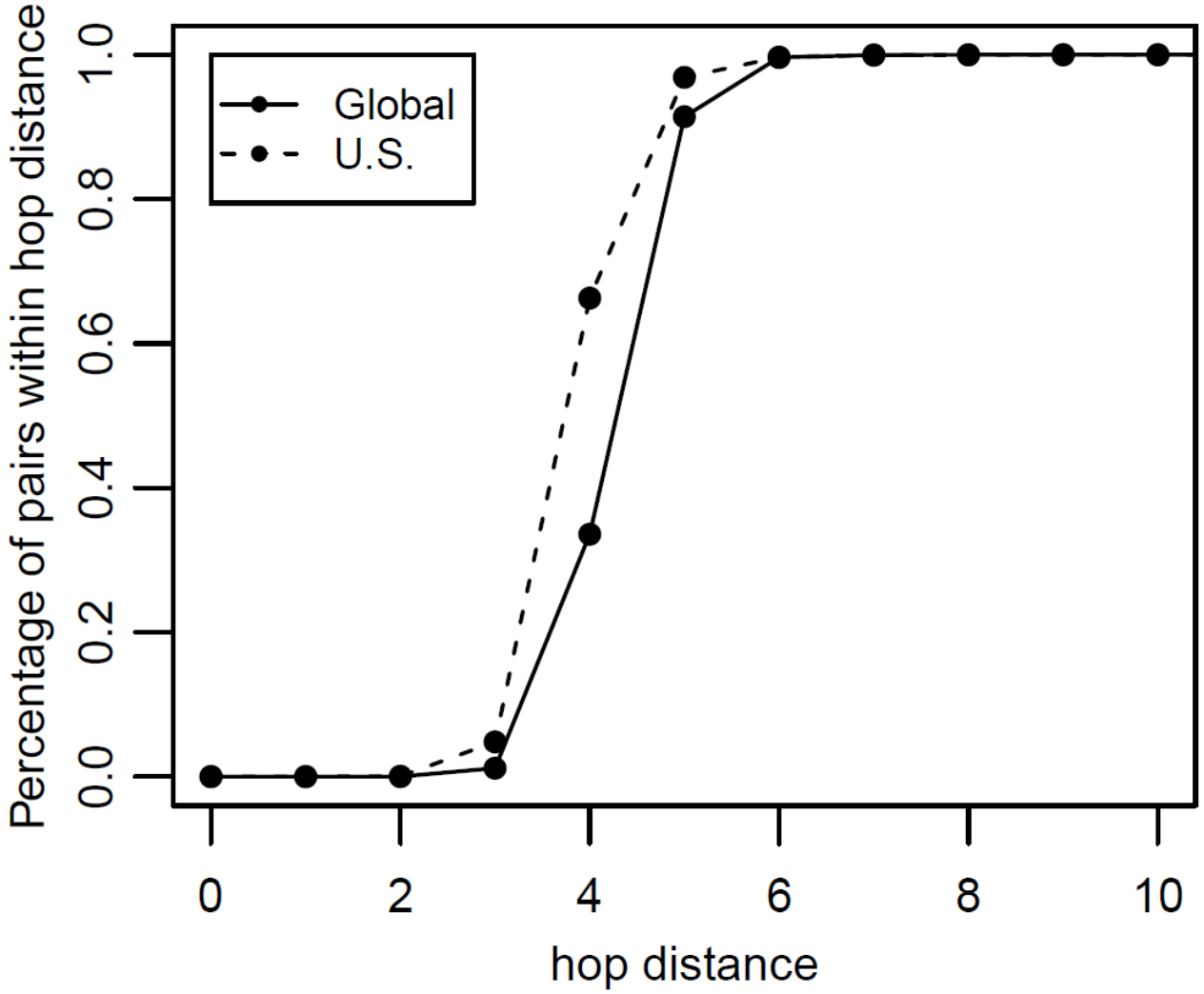


Figure 2.11: The distribution of distances in the graph of all active Microsoft Instant Messenger user accounts, with an edge joining two users if they communicated at least once during a month-long observation period [273].

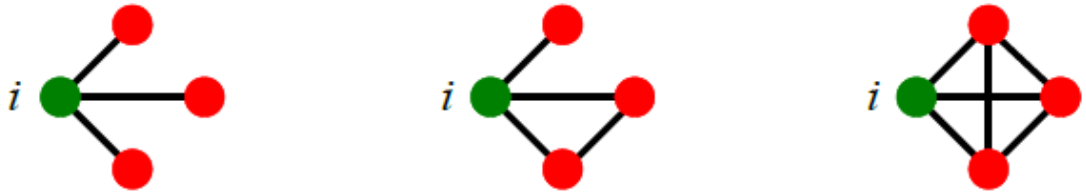
Diameter of the Facebook Social Graph



Clustering Coefficient

- **Clustering coefficient:**
  - What portion of i's neighbors are connected?
  - Node i with degree k_i
  - $C_i \in [0,1]$
  - $C_i = \dfrac{2e_i}{k_i(k_i-1)}$

where e_i is the number of edges between the neighbors of node i

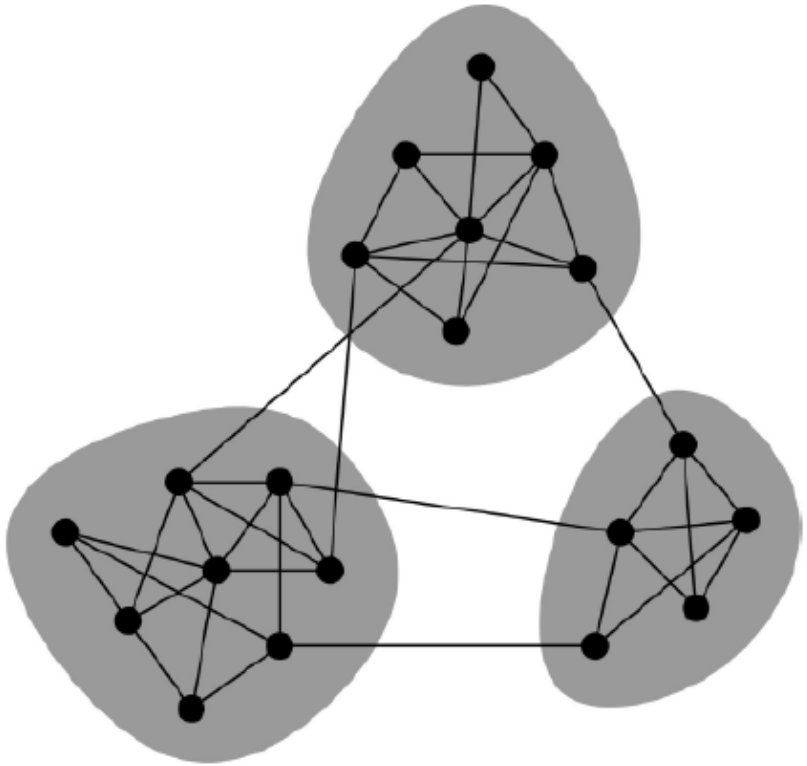


C_i=0 C_i=1/3 C_i=1

- Average Clustering Coefficient: $C = \dfrac{1}{N}\sum_{i}^{N} C_i$

Network Communities

- Networks of **tightly connected groups**
- **Network communities:**
  - Sets of nodes with **lots** of connections **inside** and **few** to **outside** (the rest of the network)



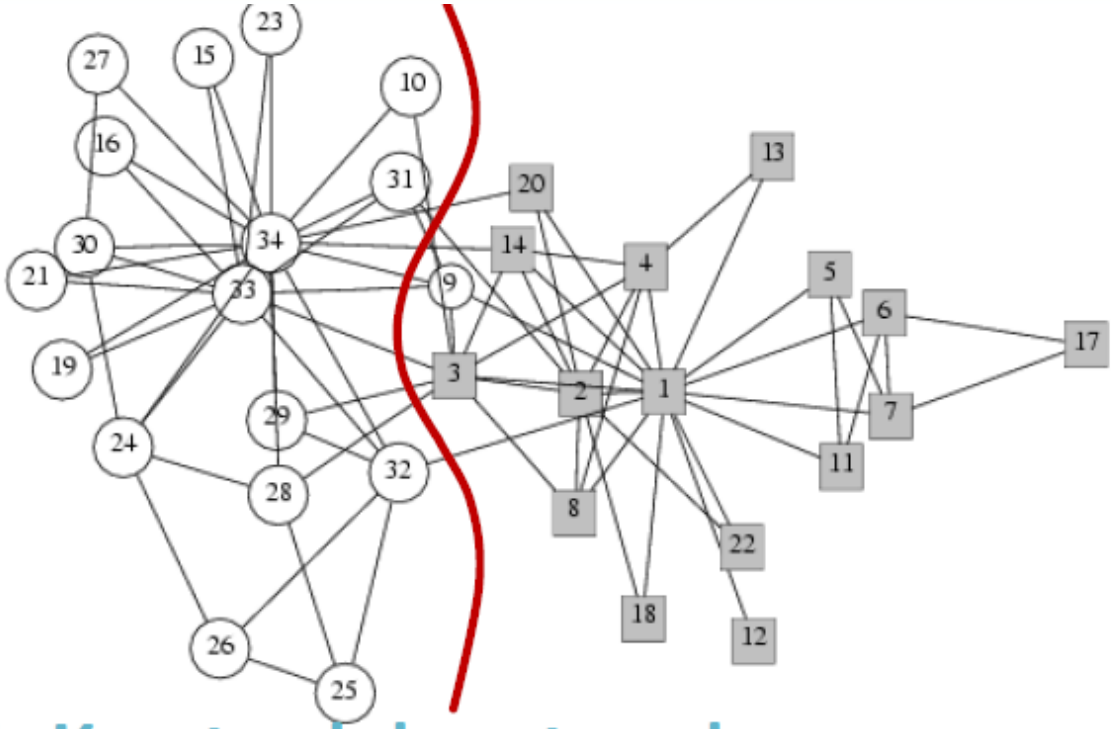
Communities, clusters, groups, modules

Finding Network Communities

- How to automatically find such densely connected groups of nodes?
- Ideally such clusters then correspond to real groups
- For example:

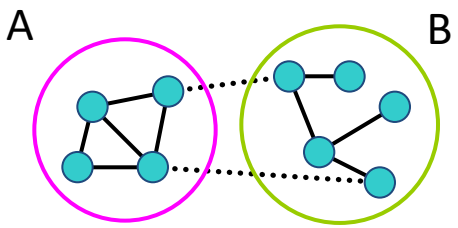
Communities, clusters, groups, modules

Social Network Data



- **Zachary's Karate club network:**
  - Observe social ties and rivalries in a university karate club
  - During his observation, conflicts led the group to split
  - Split could be explained by a minimum cut in the network

Girvan-Newman (GN) algorithm



The few edges that lie between communities can be thought of as forming “bottlenecks” between the communities.

Betweenness and edge betweenness :

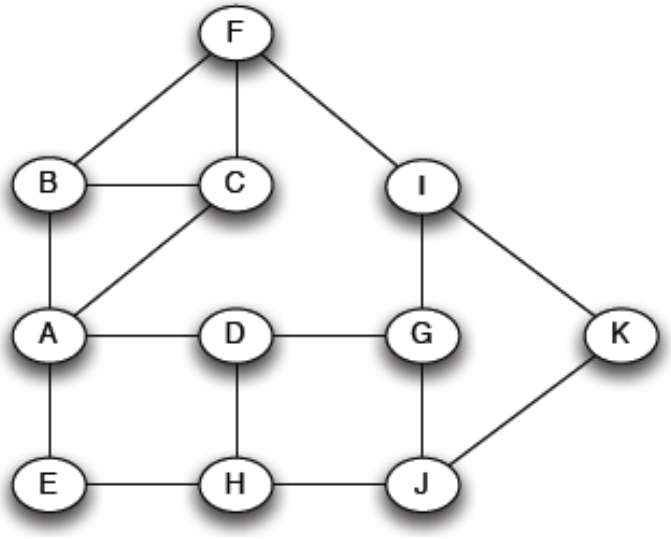
The number of shortest paths between vertex pairs that run along the edge in question, summed over all vertex pairs.

Edge removal :

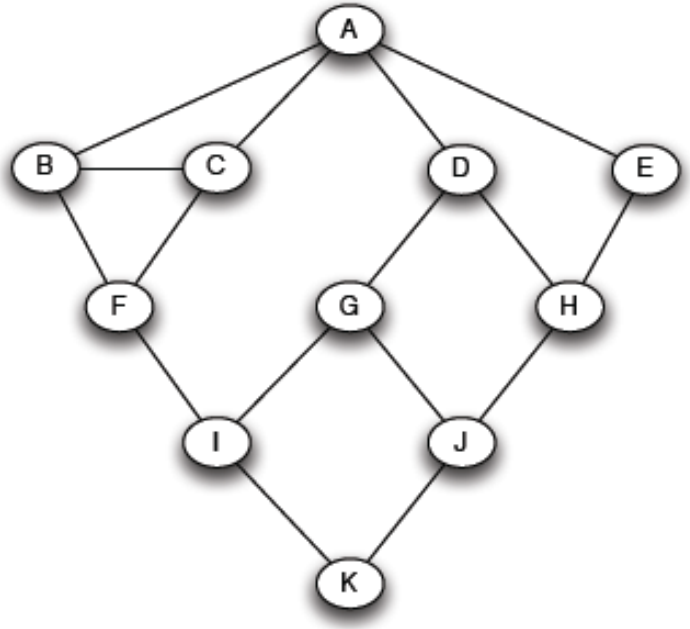
1. Calculate the betweenness for all edges in the network.
2. Remove the edge with the highest betweenness.
3. Recalculate betweennesses for all edges affected by the removal.
4. Repeat from step 2 until no edges remain.

How to Compute Betweenness?

Want to compute betweenness of paths starting at node A

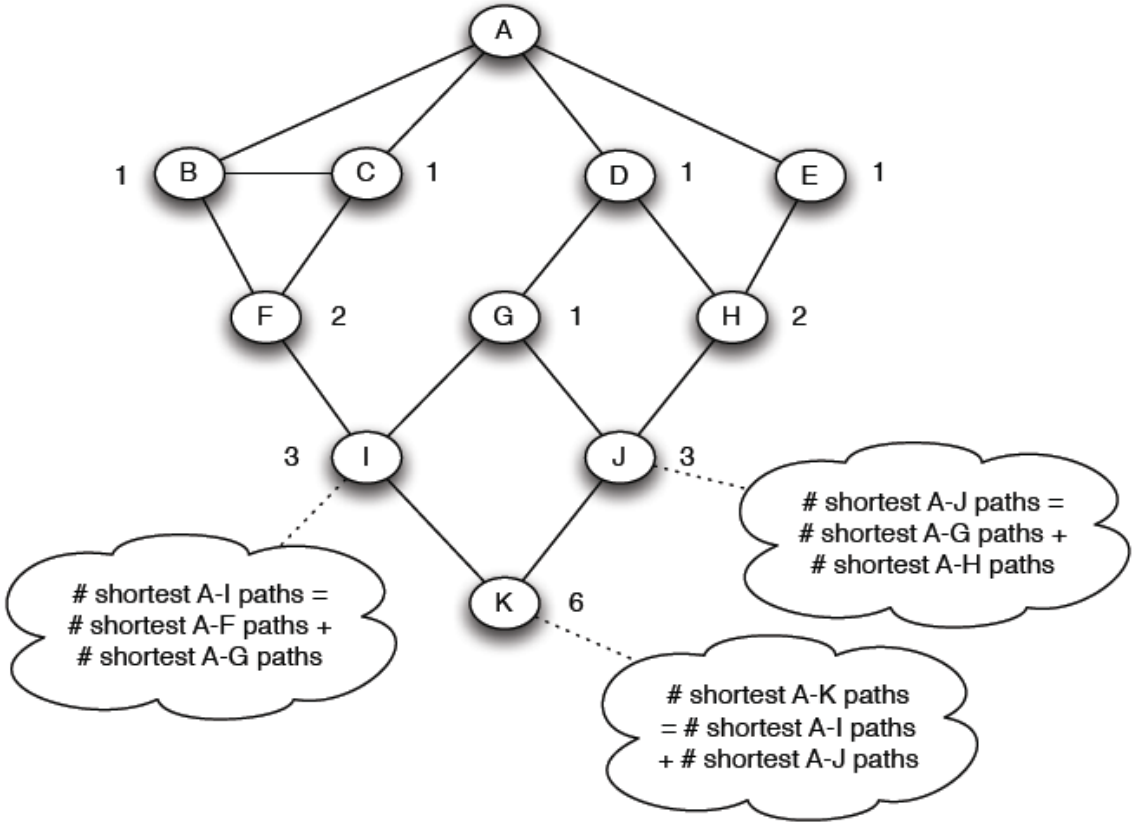


Breath first search starting from A:



How to Compute Betweenness?

- Count the number of shortest paths from A to all other nodes of the network:



How to Compute Betweenness?

- **Compute betweenness by working up the tree:** If there are multiple paths count them fractionally

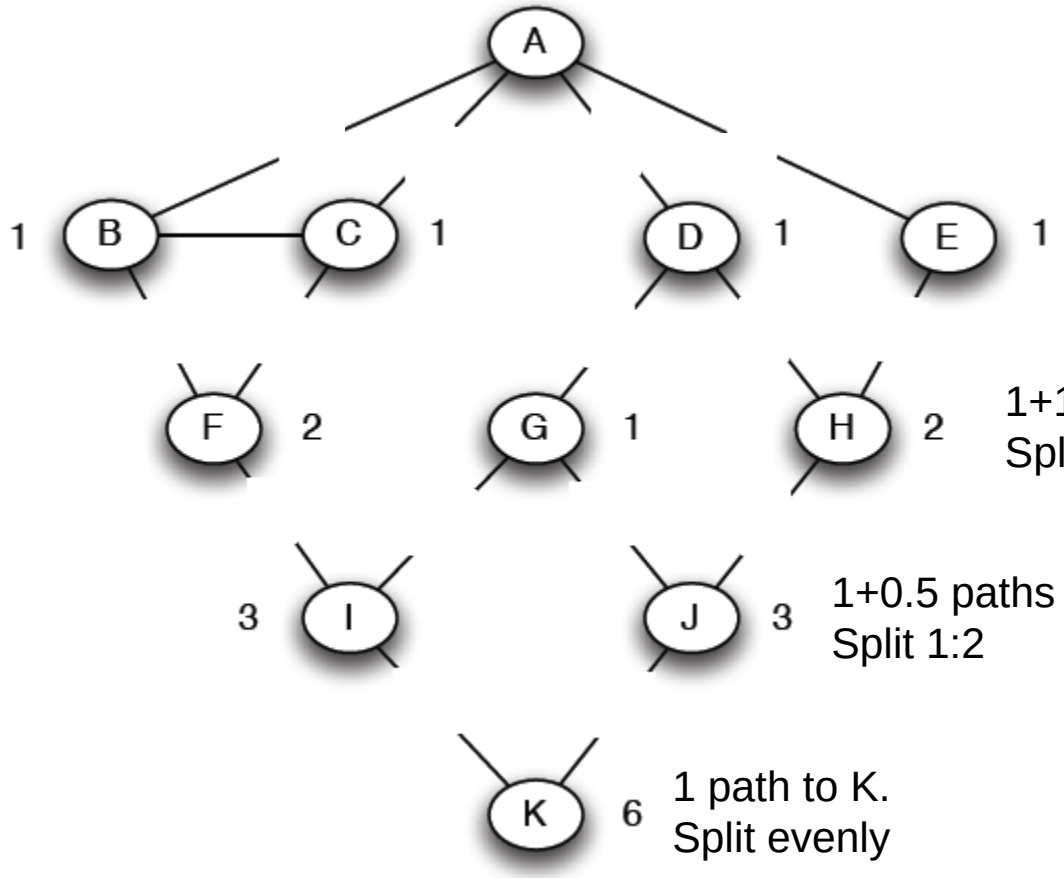
The algorithm:

•Add edge **flows**:

-- node flow = 1+∑child edges

-- split the flow up based on the parent value

• Repeat the BFS procedure for each starting node U



How to Compute Betweenness?

- **Compute betweenness by working up the tree:** If there are multiple paths count them fractionally

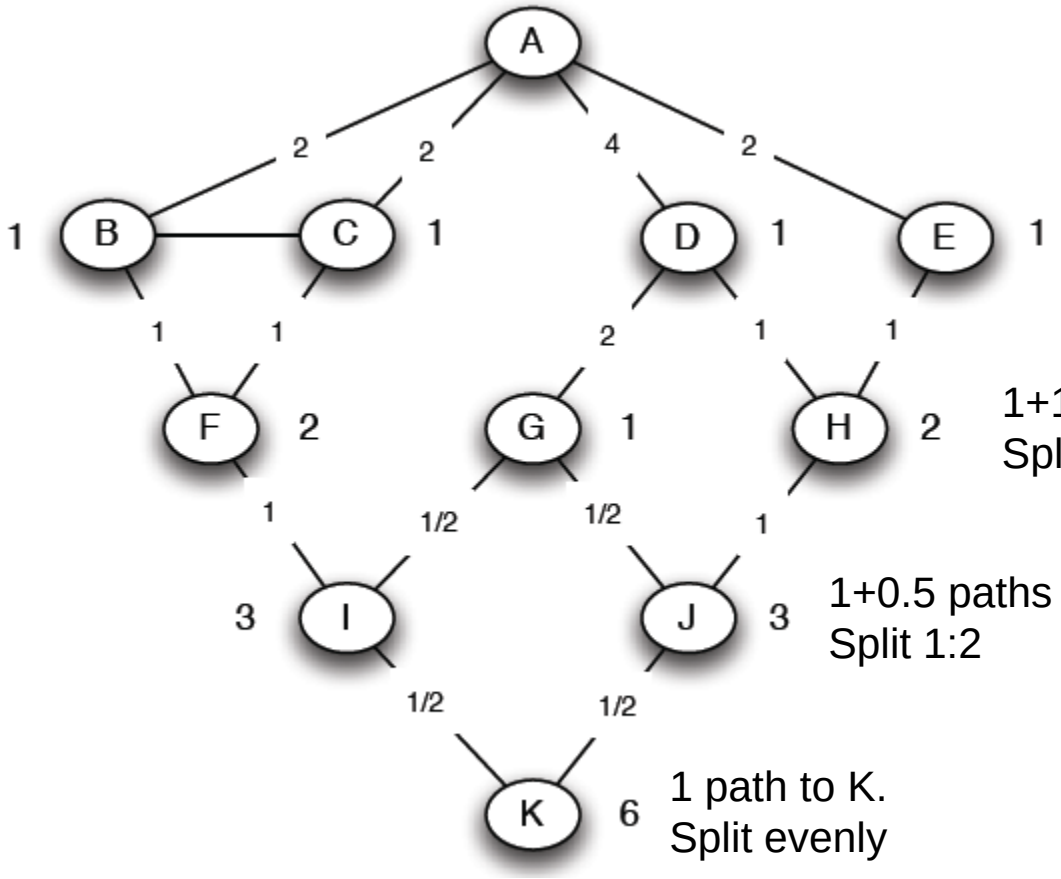
The algorithm:

•Add edge **flows**:

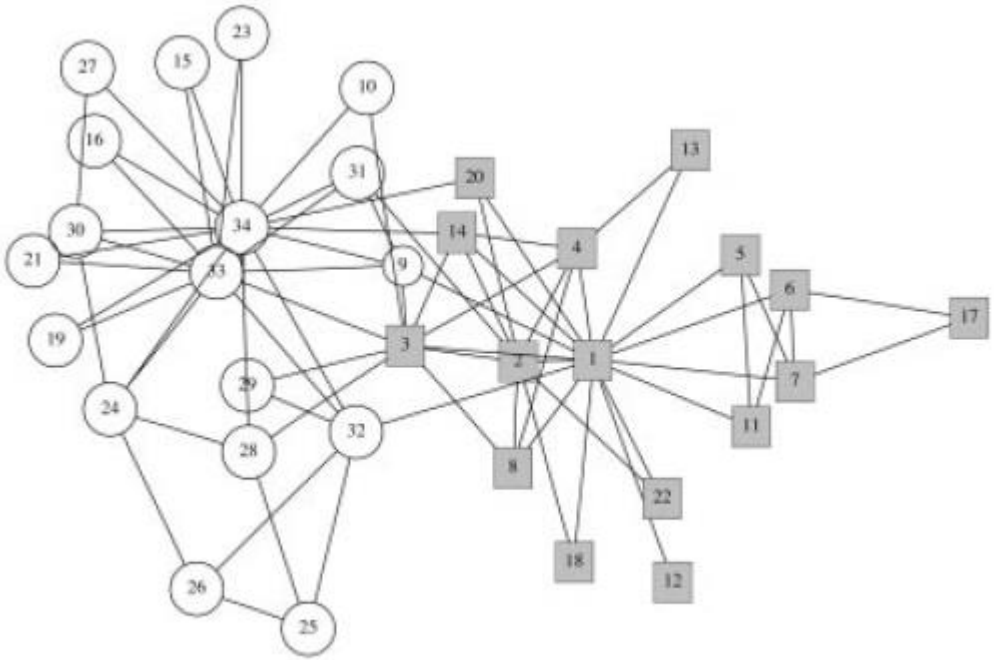
-- node flow = 1+∑child edges

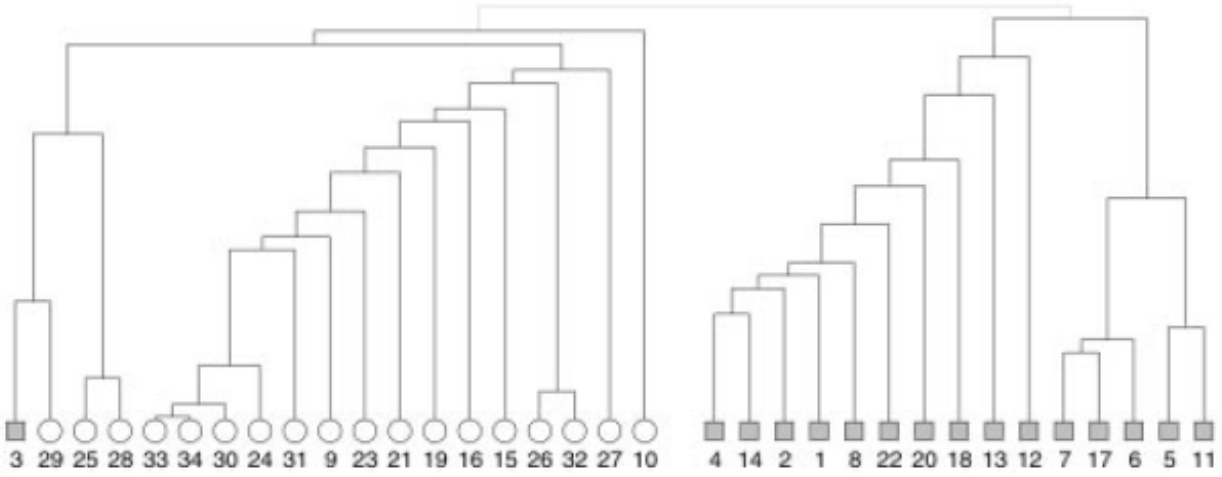
-- split the flow up based on the parent value

• Repeat the BFS procedure for each starting node U



Zachary's karate club





M. Girvan and M. E. J. Newman, PNAS 99, 7821 (2002)

The structure of the web graph

* Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. 2000. Graph structure in the Web. *Comput. Netw.* 33, 1-6 (June 2000), 309-320.

Power Law Degree Distributions

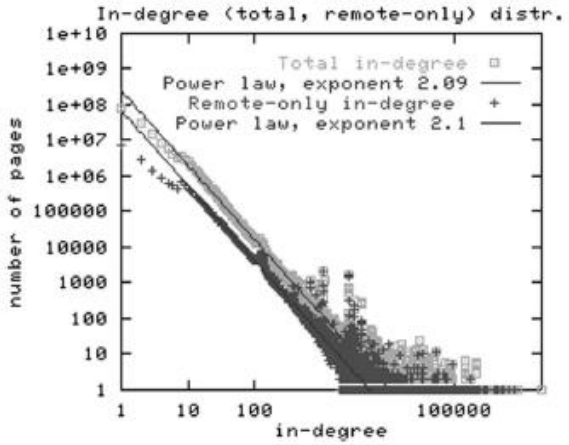


Fig. 1. In-degree distributions subscribe to the power law. The law also holds if only off-site (or 'remote-only') edges are considered.

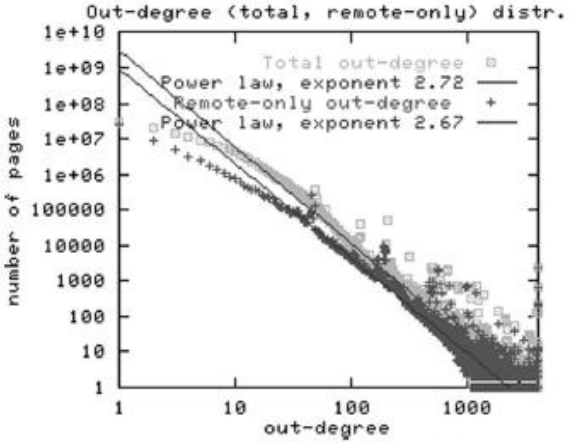


Fig. 2. Out-degree distributions subscribe to the power law. The law also holds if only off-site (or 'remote-only') edges are considered.

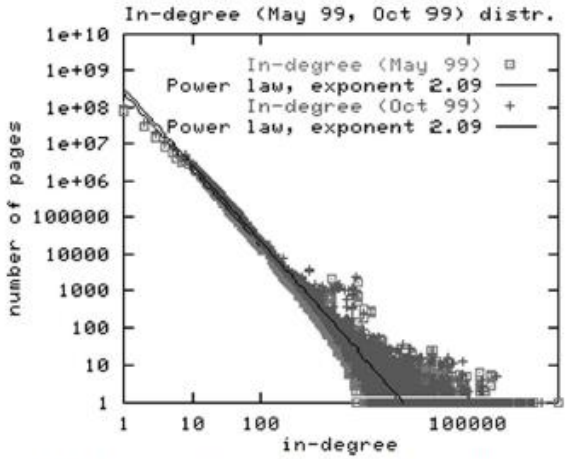


Fig. 3. In-degree distributions show a remarkable similarity over two crawls, run in May and October 1999. Each crawl counts well over 1 billion distinct edges of the Web graph.

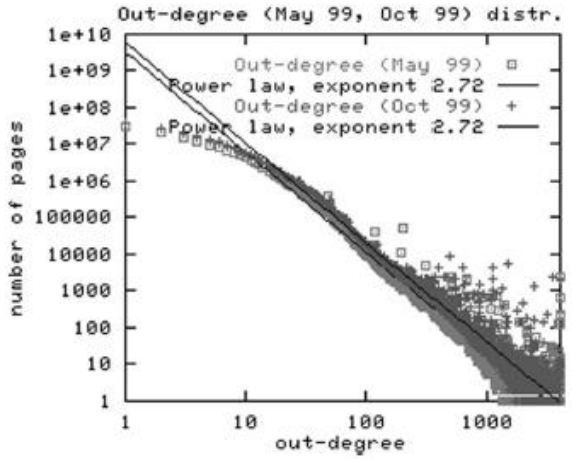
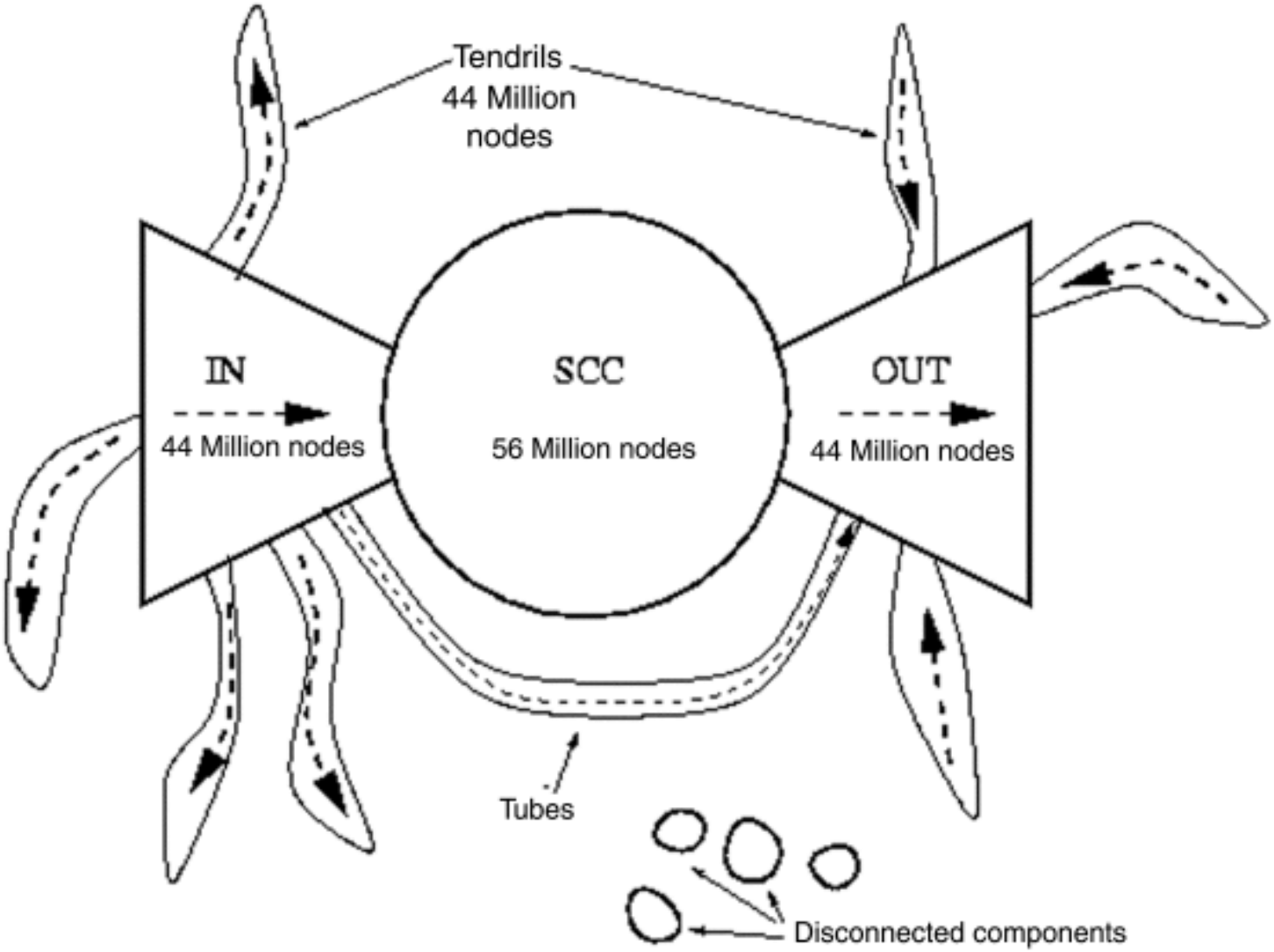


Fig. 4. Out-degree distributions show a remarkable similarity over two crawls, run in May and October 1999. Each crawl counts well over 1 billion distinct edges of the Web graph.

The „bow tie“ structure of the Web



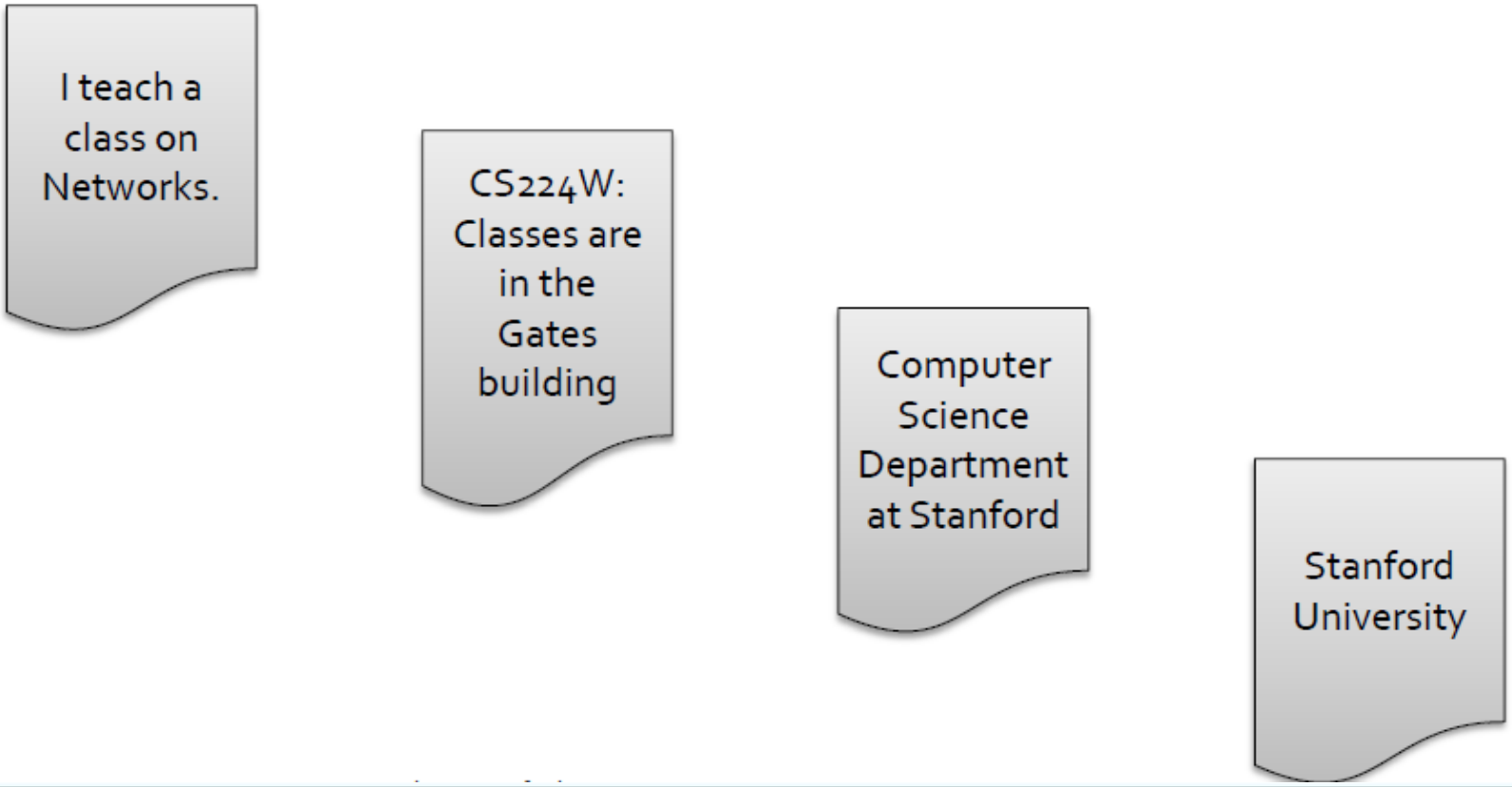
(Social) Networks

- Degree Distribution
  - many networks possess right-skewed degree distributions
  - there are typically many vertices in a network with low degree and a small number with high degree, the precise distribution often following a power-law or exponential form
- Small world phenomaena (N degrees of separation)
  - the average distance between vertices in a network is short usually scaling logarithmically with the total number *n* of vertices
- One Giant Component
- Network Transitivity (Triadic Closure)
  - two vertices that are both neighbors of the same third vertex have a heightened probability of also being neighbors of one another
  - In the language of social networks, two of your friends will have a greater probability of knowing one another than will two people chosen at random from the population, on account of their common acquaintance with you.
  - This effect is quantified by the clustering coefficient *C*

PageRank

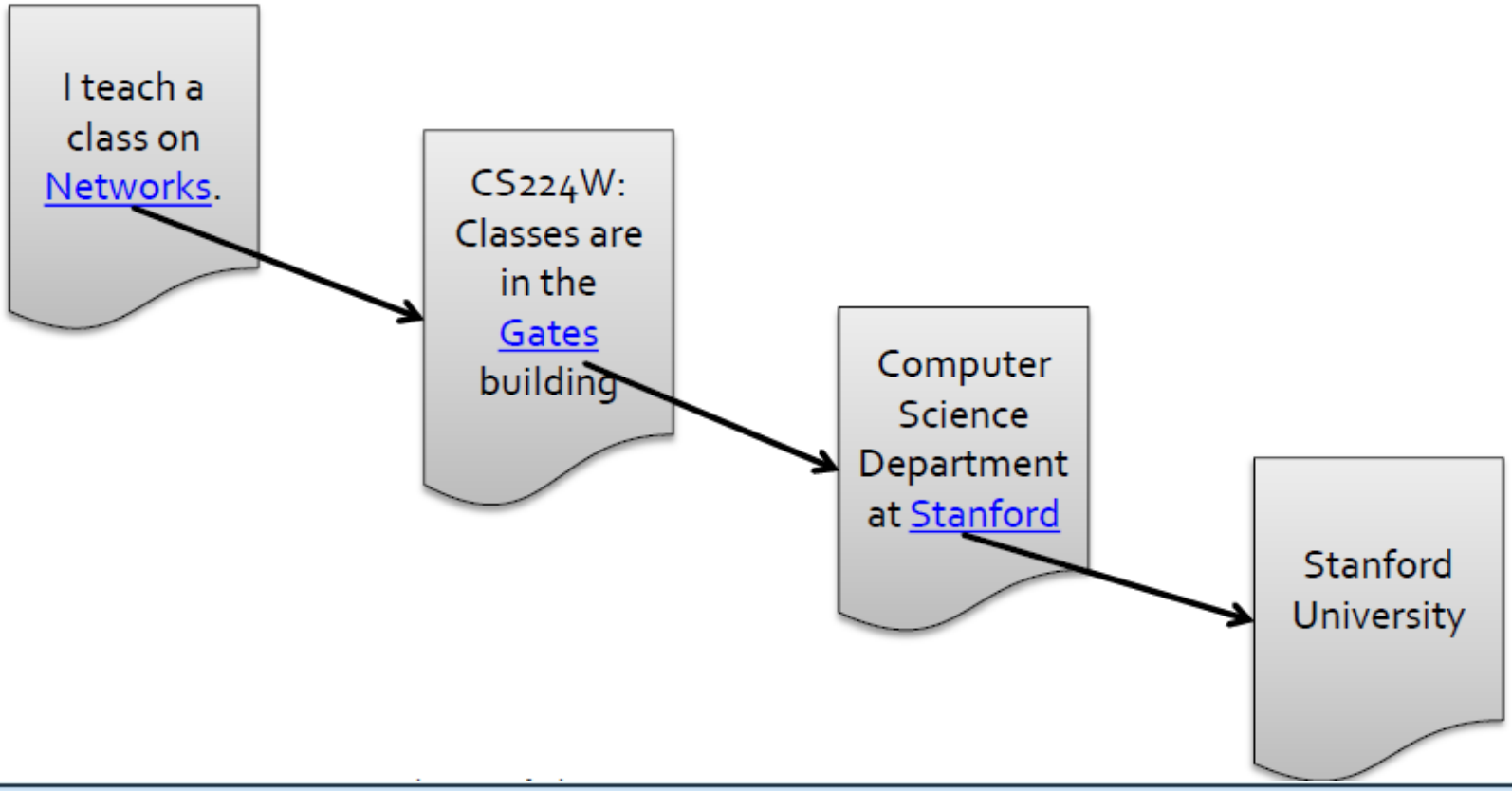
Web as a Graph

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



Web as a Graph

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



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:

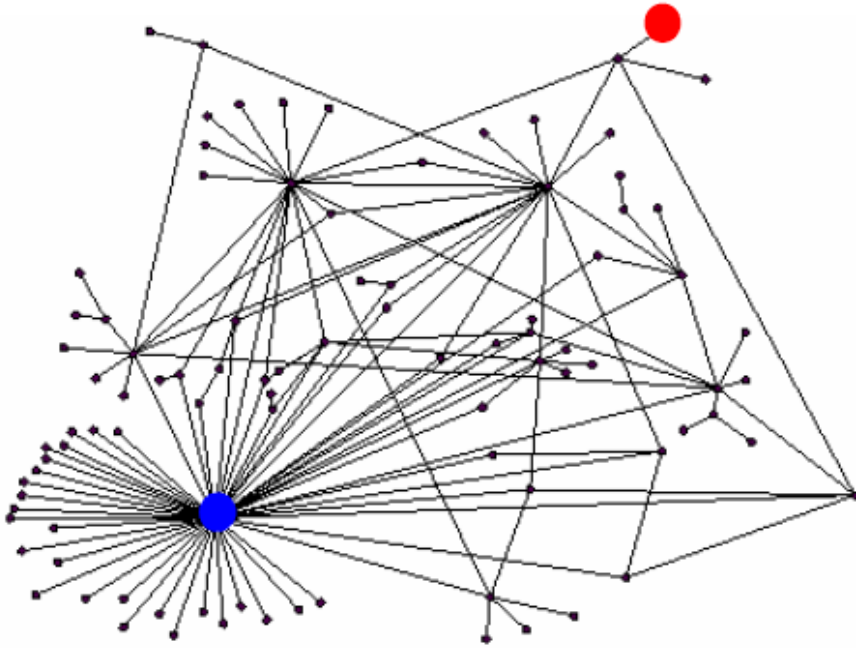
- **(1) Web contains many sources of information Who to "trust"?**
  - **Trick:** Trustworthy pages may point to each other!
- **(2) What is the "best" answer to query "newspaper"?**
  - No single right answer
  - **Trick:** Pages that actually know about newspapers might all be pointing to many newspapers

- **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.

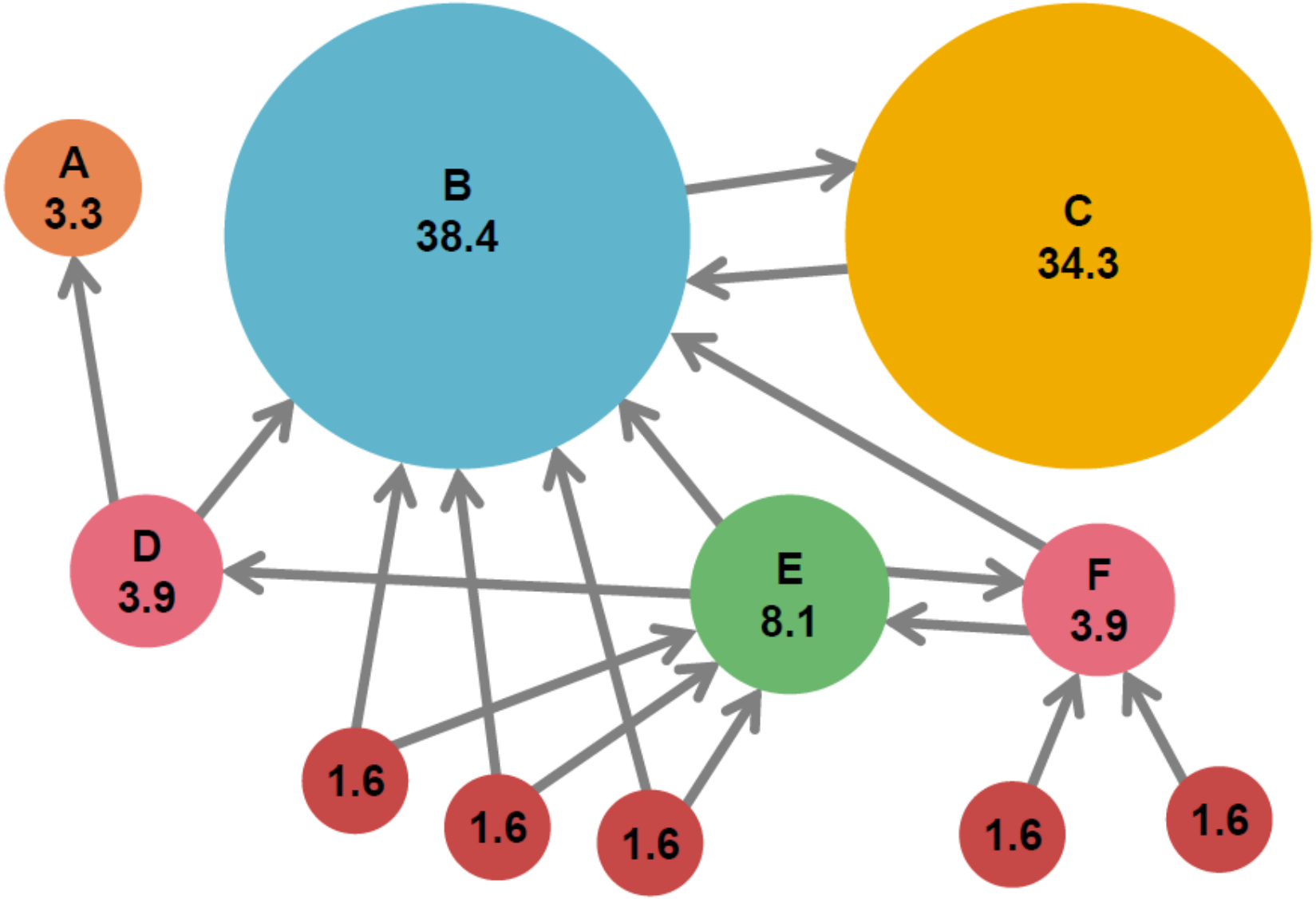
  Let's rank the pages by the link structure!



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!

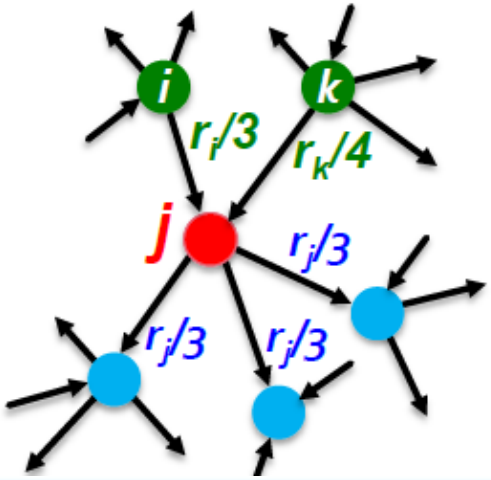
Example: PageRank Scores



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

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



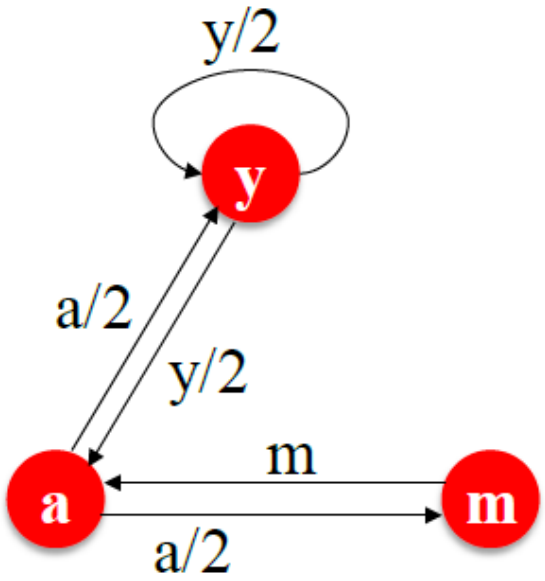
PageRank: The „Flow“ Model

- 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

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

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

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

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

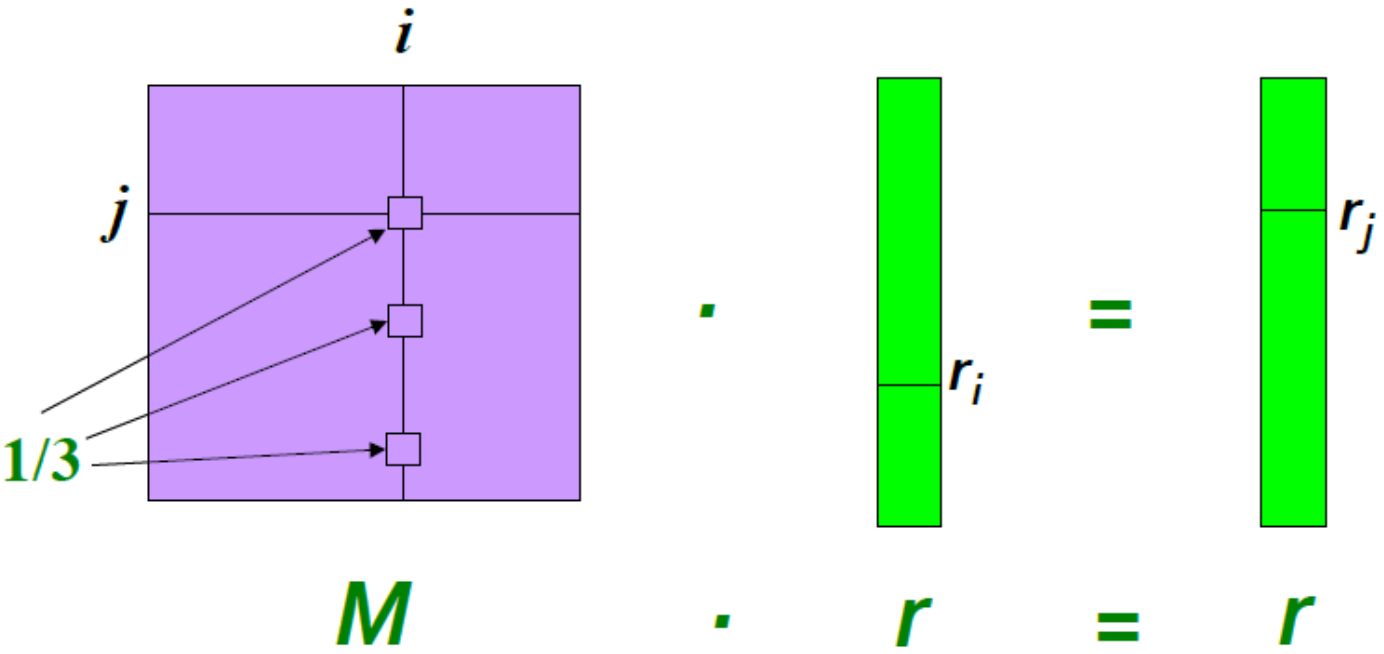
$$r = M \cdot r$$

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***



Eigenvector Formulation

- **The flow equations can be written**

$$r = M \cdot r$$

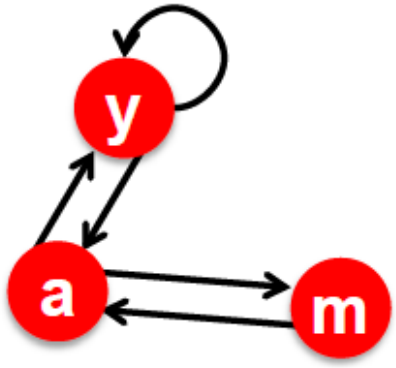
- So the **rank vector** ***r*** is an **eigenvector** of the stochastic web matrix ***M***
  - In fact, its first or principal eigenvector, with corresponding eigenvalue ***1***
    - Largest eigenvalue of ***M*** is **1** since ***M*** is column stochastic (with non-negative entries)
      - *We know*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

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

$$Ax = \lambda x$$

- **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 $|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}|_1 < \varepsilon$

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

d_i …. out-degree of node i

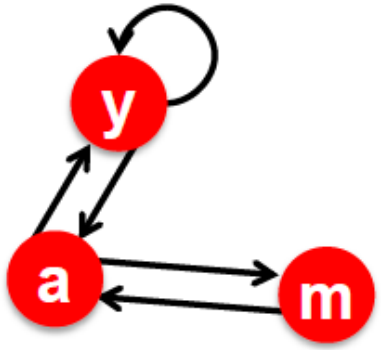
$|\mathbf{x}|_1 = \sum_{1 \le i \le N} |x_i|$ is the L_1 norm
Can use any other vector norm, e.g., Euclidean

PageRank: How to solve?

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

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

Iteration 0, 1, 2, …



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

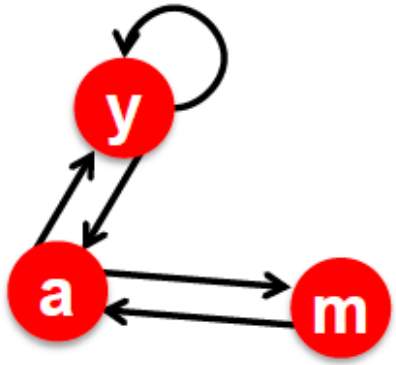
$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$

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**
- **Example:**



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

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

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

Iteration 0, 1, 2, …

Why Power Iteration Works

- **Power iteration:**

  A method for finding dominant eigenvector (the vector corresponding to the largest eigenvalue)

  - $\boldsymbol{r^{(1)} = M \cdot r^{(0)}}$
  - $\boldsymbol{r^{(2)} = M \cdot r^{(1)} = M\left(M r^{(1)}\right) = M^2 \cdot r^{(0)}}$
  - $\boldsymbol{r^{(3)} = M \cdot r^{(2)} = M\left(M^2 r^{(0)}\right) = M^3 \cdot r^{(0)}}$

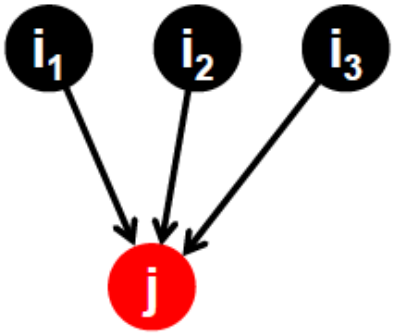
- **Claim:**

  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 Sketch in Appendix of Slides!

Random Walk Interpretation

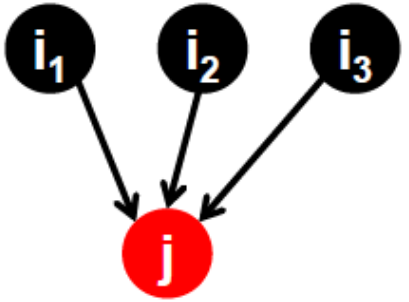
- **Imagine a random web surfer:**
  - At any time t, surfer is on some page i
  - At time $t+1$, the surfer follows an out-link from i uniformly at random
  - Ends up on some page j linked from i
  - Process repeats indefinitely



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

- **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

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

- Suppose the random walk reaches a state

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

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

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

Existance and Uniqueness

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

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

PageRank: Three Questions

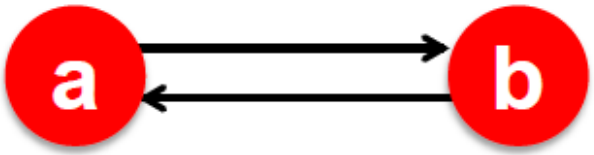
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{\mathrm{d_i}}$$

or equivalently

$$r = Mr$$

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

Does this converge?



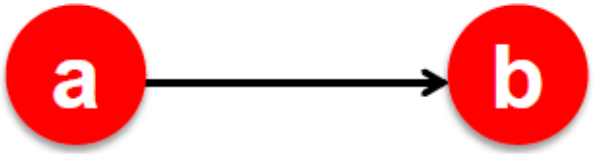
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i}$$

- **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?



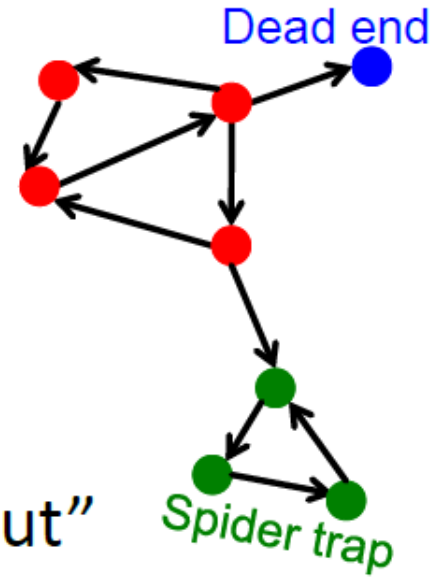
$$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i}$$

- **Example:**

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

Iteration 0, 1, 2, …

PageRank: Problems

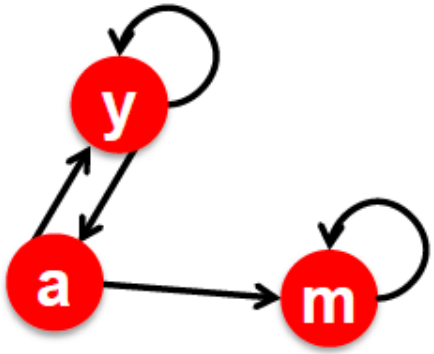


2 problems:

- **(1)** Some pages are **dead ends** (have no out-links)
  - Random walk has "nowhere" to go to
  - Such pages cause importance to "leak out"
- **(2) Spider traps:** (all out-links are within the group)
  - Random walked gets "stuck" in a trap
  - And eventually spider traps absorb all importance

Problem: Spidertraps

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



m is a spider trap

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

$r_y = r_y/2 + r_a/2$

$r_a = r_y/2$

$r_m = r_a/2 + r_m$

- **Example:**

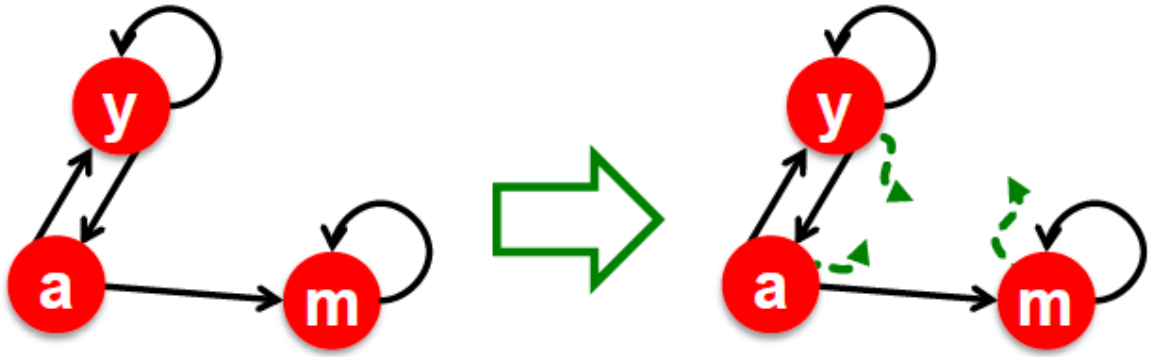
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \dots & 0 \\ 1/3 & 3/6 & 7/12 & 16/24 & & 1 \end{matrix}$$

Iteration 0, 1, 2, …

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

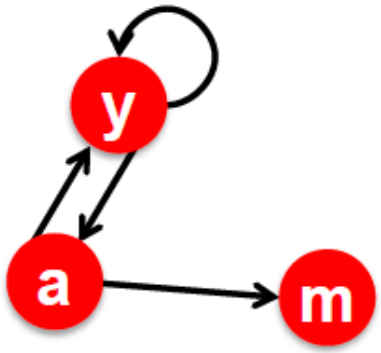
Solution: Teleports

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



Problem: Dead Ends

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



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

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

- **Example:**

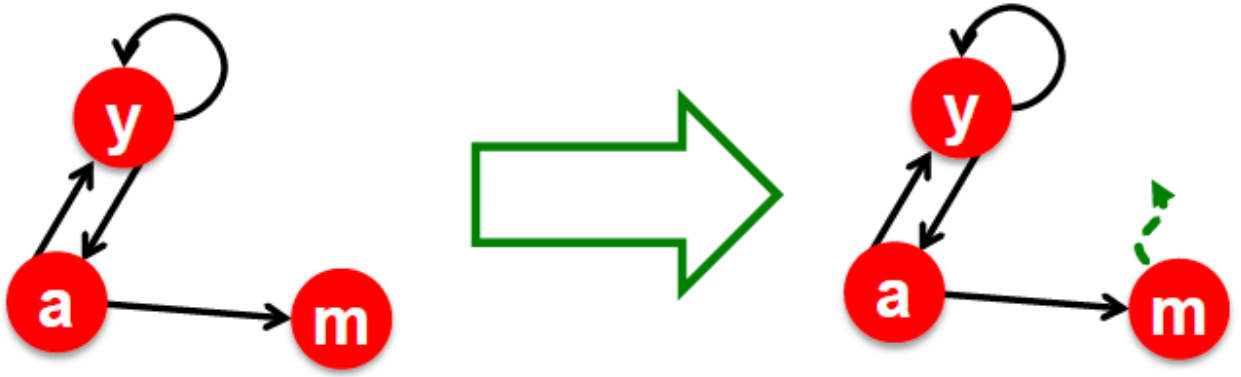
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\ 1/3 & 1/6 & 1/12 & 2/24 & & 0 \end{matrix}$$

Iteration 0, 1, 2, …

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

Solution: Always Teleport

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



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

	y	a	m
y	½	½	⅓
a	½	0	⅓
m	0	½	⅓

Why Teleports solve the Problem

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

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

Solution: Random Teleports

- **Google's solution that does it all:**

  At each step, random surfer has two options:
  - With probability β, follow a link at random
  - With probability $1-\beta$, 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

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

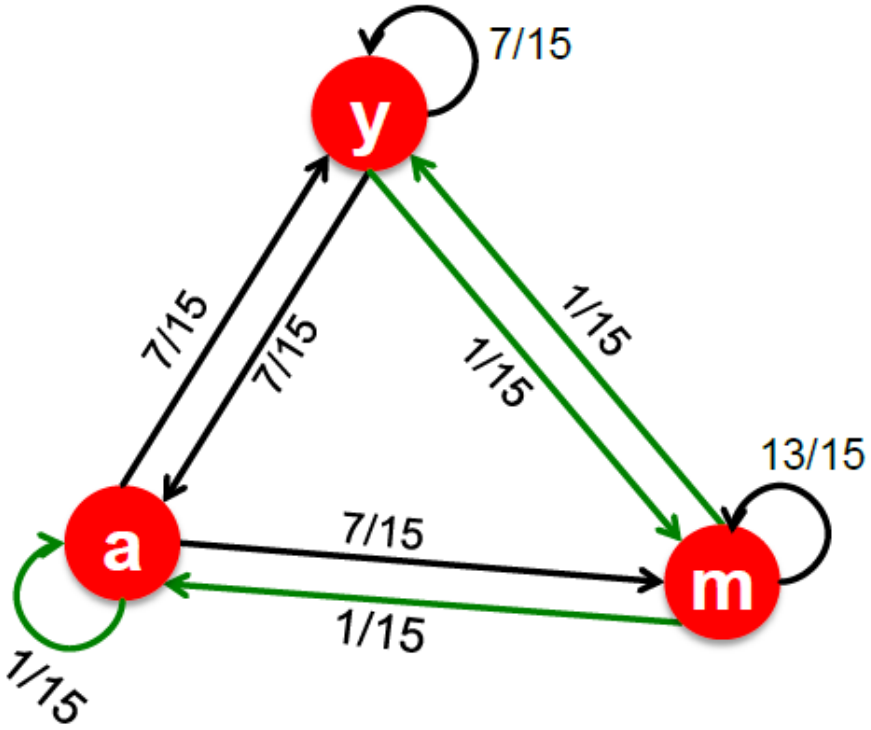
$$A = \beta\, M + (1 - \beta) \left[\frac{1}{N}\right]_{N \times N}$$

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

  And the Power method still works!

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

Random Teleports (β=0.8)



M

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

$[1/N]_{NxN}$

A

y	7/15	7/15	1/15
a	7/15	1/15	1/15
m	1/15	7/15	13/15

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

Computing PageRank

- **Key step is matrix-vector multiplication**
  - $r^{new} = A \cdot r^{old}$
- 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!

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

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

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

Matrix Formulation

- Suppose there are N pages
- Consider page i, with d_i out-links
- We have $M_{ji} = 1/|d_i|$ when $i \rightarrow j$
  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 $A_{ji} = \beta M_{ji} + \frac{1-\beta}{N}$
- $r_j = \sum_{i=1}^{N} A_{ji} \cdot r_i$
- $r_j = \sum_{i=1}^{N} \left[\beta M_{ji} + \frac{1-\beta}{N}\right] \cdot r_i$

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

$$= \sum_{i=1}^{N} \beta M_{ji} \cdot r_i + \frac{1-\beta}{N} \quad \text{since } \sum r_i = 1$$

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

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**

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

  - where $\boldsymbol{[(1-\beta)/N]_N}$ is a vector with all $\boldsymbol{N}$ entries $\boldsymbol{(1-\beta)/N}$

- $\boldsymbol{M}$ is a **sparse matrix!** (with no dead-ends)
  - 10 links per node, approx 10N entries
- So in each iteration, we need to:
  - Compute $\boldsymbol{r}^{new} = \boldsymbol{\beta M} \cdot \boldsymbol{r}^{old}$
  - Add a constant value $\boldsymbol{(1-\beta)/N}$ to each entry in $\boldsymbol{r}^{new}$
    - **Note if M contains dead-ends then $\sum_j r_j^{new} < 1$ and we also have to renormalize $\boldsymbol{r}^{new}$ 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:** $\sum_j \left| r_j^{new} - r_j^{old} \right| > \varepsilon$
    - $\forall j$: $r'^{new}_j = \sum_{i \to j} \beta \frac{r_i^{old}}{d_i}$
      $r'^{new}_j = 0$ if in-degree of j is 0
    - **Now re-insert the leaked PageRank:**
      $\forall j$: $r_j^{new} = r'^{new}_j + \frac{1-S}{N}$ **where:** $S = \sum_j r'^{new}_j$
    - $r^{old} = r^{new}$

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

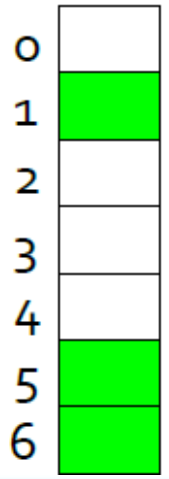
- **Encode sparse matrix using only nonzero entries**
  - Space proportional roughly to number of links
  - Say 10N, or 4*10*1 billion = 40GB
  - **Still won't fit in memory, but will fit on disk**

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

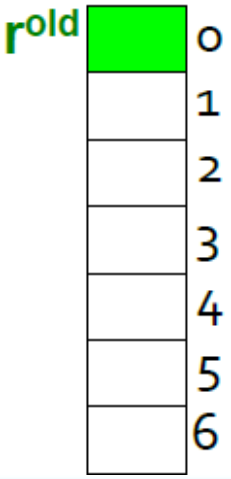
Basic Algorithm: Update Step

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



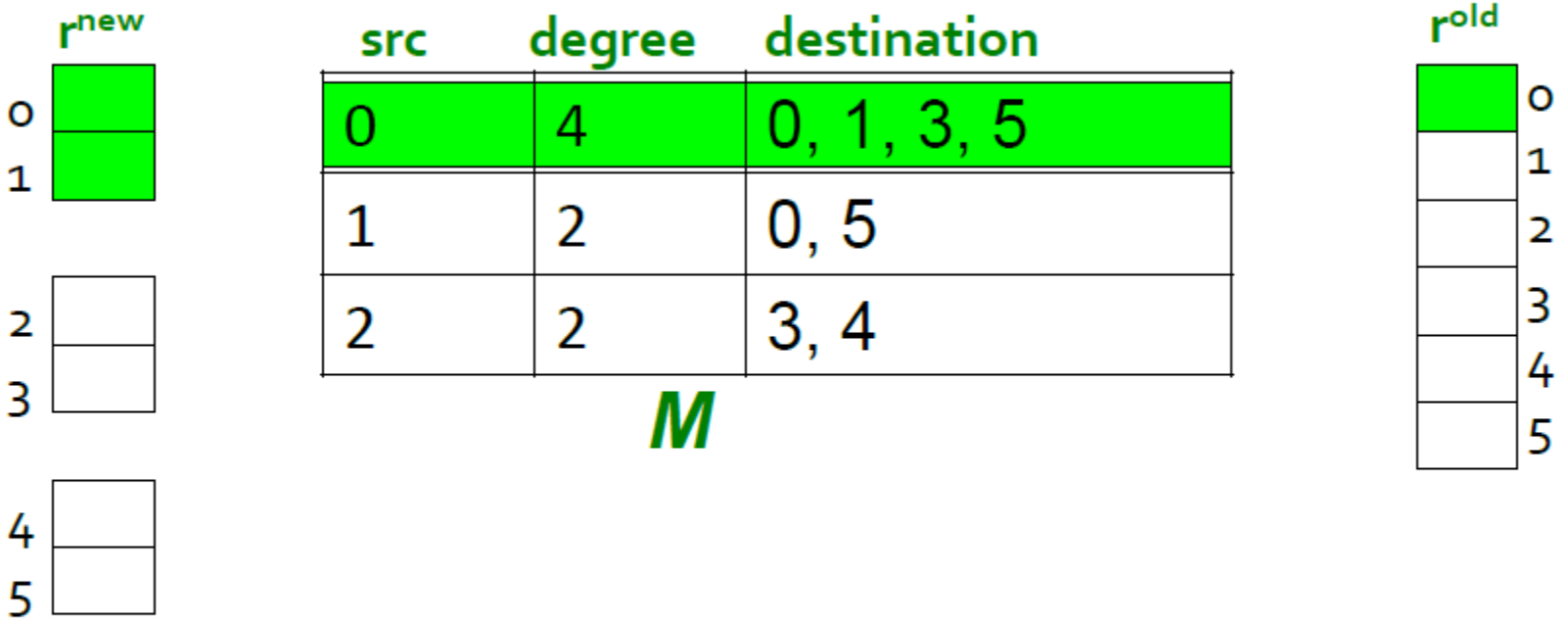
source	degree	destination
0	3	1, 5, 6
1	4	17, 64, 113, 117
2	2	13, 23



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

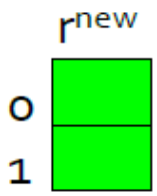


- Break r^{new} into k blocks that fit in memory
- Scan M and r^{old} once for each block

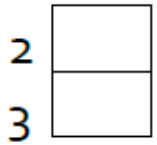
Analysis of Block Update

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

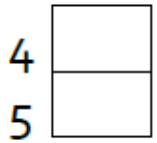
Block-Stripe Update Algorithm



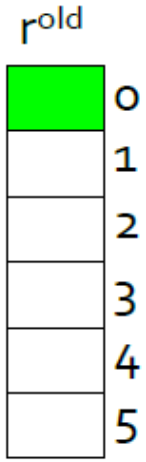
src	degree	destination
0	4	0, 1
1	3	0
2	2	1



0	4	3
2	2	3



0	4	5
1	3	5
2	2	4



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

Block-Stripe Analysis

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

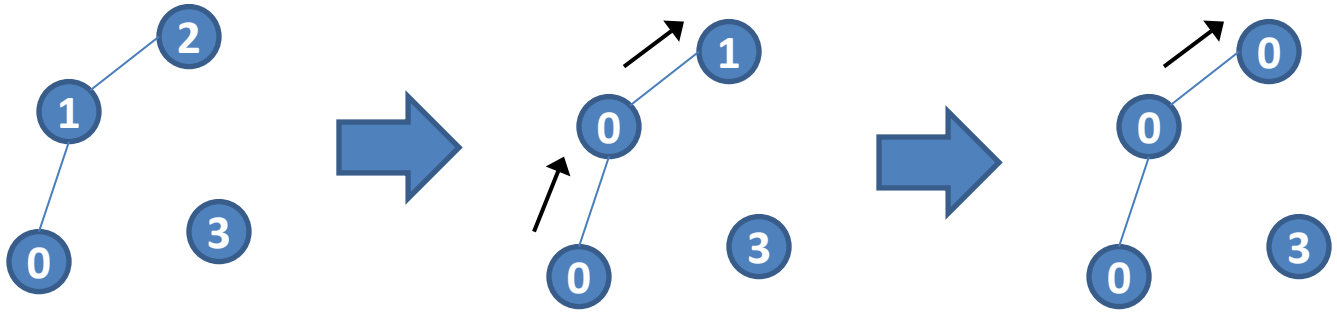
Some Problems with PageRank

- **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

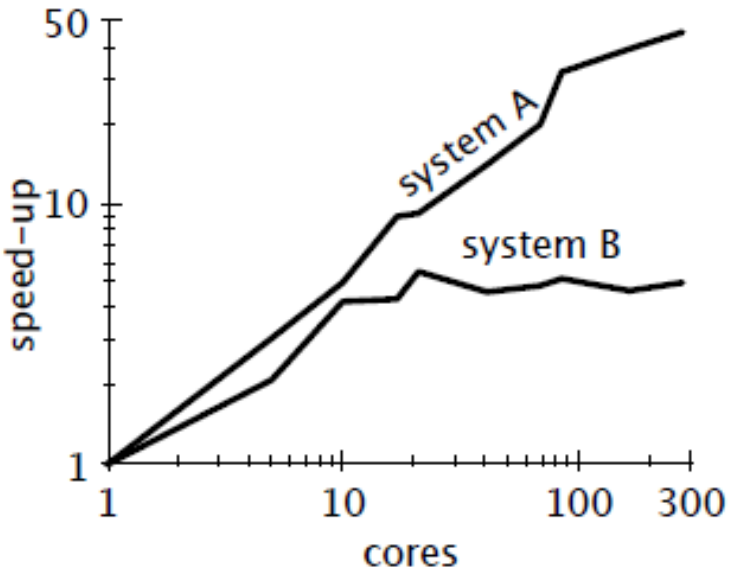
Example: finding the connected components of an undirected graph

- algorithm: propagate smallest vertex label to neighbors until convergence

- in the end, all vertices of a component will have the same label



Scalability



- *Scalability! But at what COST?* Frank McSherry, Michael Isard, Derek G. Murray

20 PageRank Iterations

name	twitter_rv [11]	uk-2007-05 [4]
nodes	41,652,230	105,896,555
edges	1,468,365,182	3,738,733,648
size	5.76GB	14.72GB

scalable system	cores	twitter	uk-2007-05
GraphChi [10]	2	3160s	6972s
Stratosphere [6]	16	2250s	-
X-Stream [17]	16	1488s	-
Spark [8]	128	857s	1759s
Giraph [8]	128	596s	1235s
GraphLab [8]	128	249s	833s
GraphX [8]	128	419s	462s

Label propagation (Connected Components)

scalable system	cores	twitter	uk-2007-05
Stratosphere [6]	16	950s	-
X-Stream [17]	16	1159s	-
Spark [8]	128	1784s	$\geq$ 8000s
Giraph [8]	128	200s	$\geq$ 8000s
GraphLab [8]	128	242s	714s
GraphX [8]	128	251s	800s
Single thread (SSD)	1	153s	417s

Appendix

Why Power Iteration Works (1)

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

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 $\boldsymbol{M}$ has $\boldsymbol{n}$ 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: $r^{(0)} = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$
  - $$\boldsymbol{Mr^{(0)}} = \boldsymbol{M(c_1 x_1 + c_2 x_2 + \cdots + c_n x_n)}$$
    $$= c_1(Mx_1) + c_2(Mx_2) + \cdots + c_n(Mx_n)$$
    $$= c_1(\lambda_1 x_1) + c_2(\lambda_2 x_2) + \cdots + c_n(\lambda_n x_n)$$
  - **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 $\boldsymbol{M \cdot r^{(0)}, M^2 \cdot r^{(0)}, \ldots M^k \cdot r^{(0)}, \ldots}$ 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:** $\boldsymbol{M^k r^{(0)} \approx c_1(\lambda_1^k x_1)}$
    - Note if $c_1 = 0$ then the method won't converge