CS3319 Foundations of Data Science

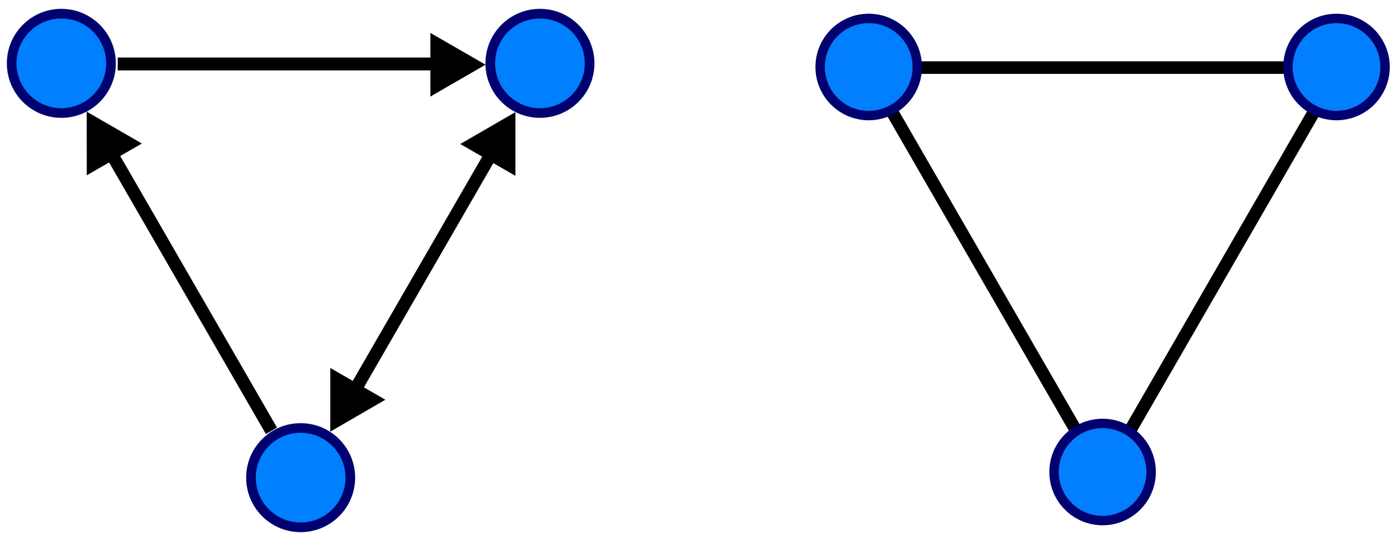
5.Graph Data

Jiaxin Ding

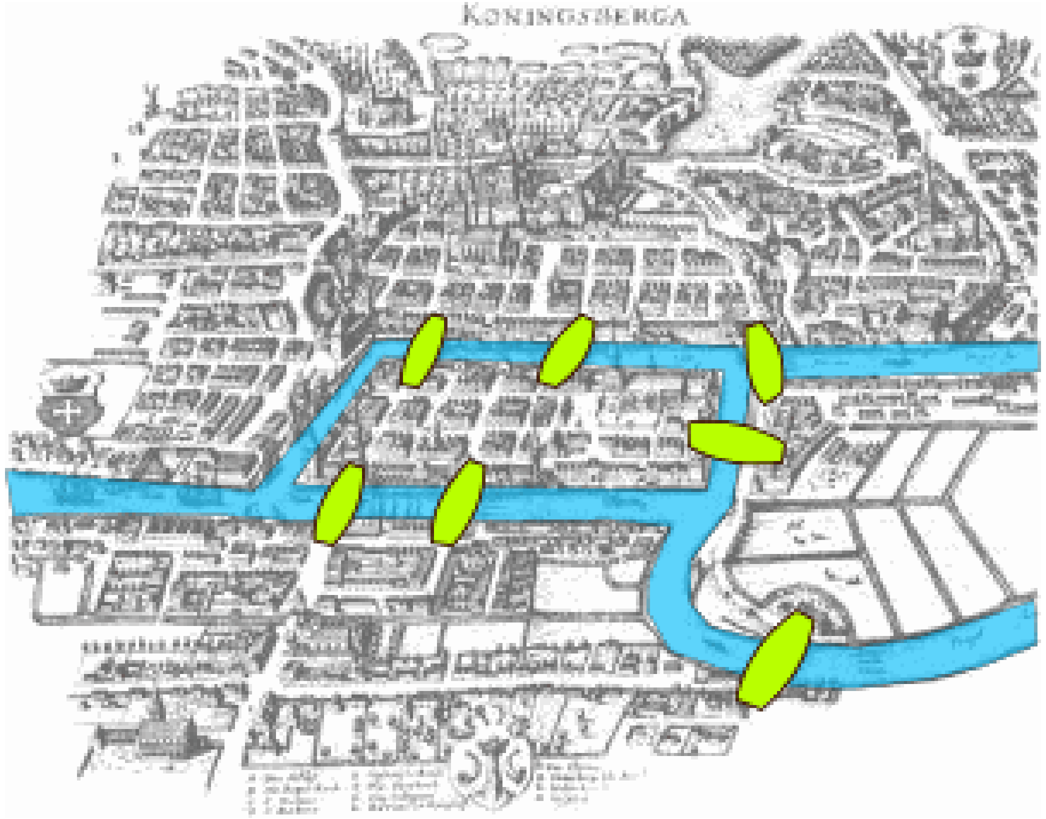
John Hopcroft Center

Graph

- **Graph**: structure of a set of objects some of which are related.
  - Vertices/Nodes (objects)
  - Edge/Links (relations, directed or undirected)



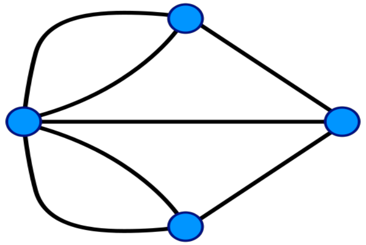
Graph Data



Seven Bridges of Königsberg

[Euler, 1735]

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



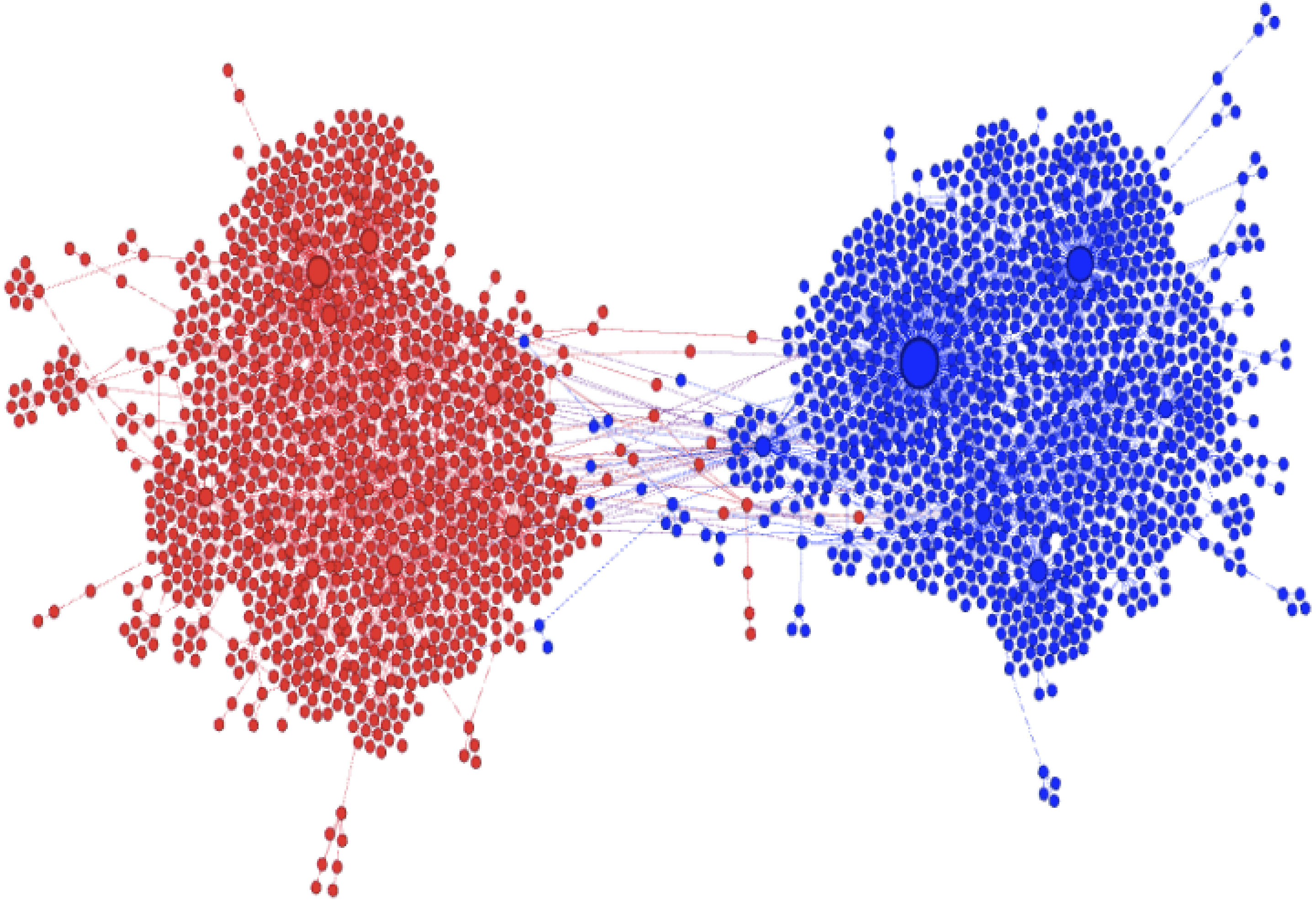
Adapted from J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Graph Data: Social Networks

Facebook social graph

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

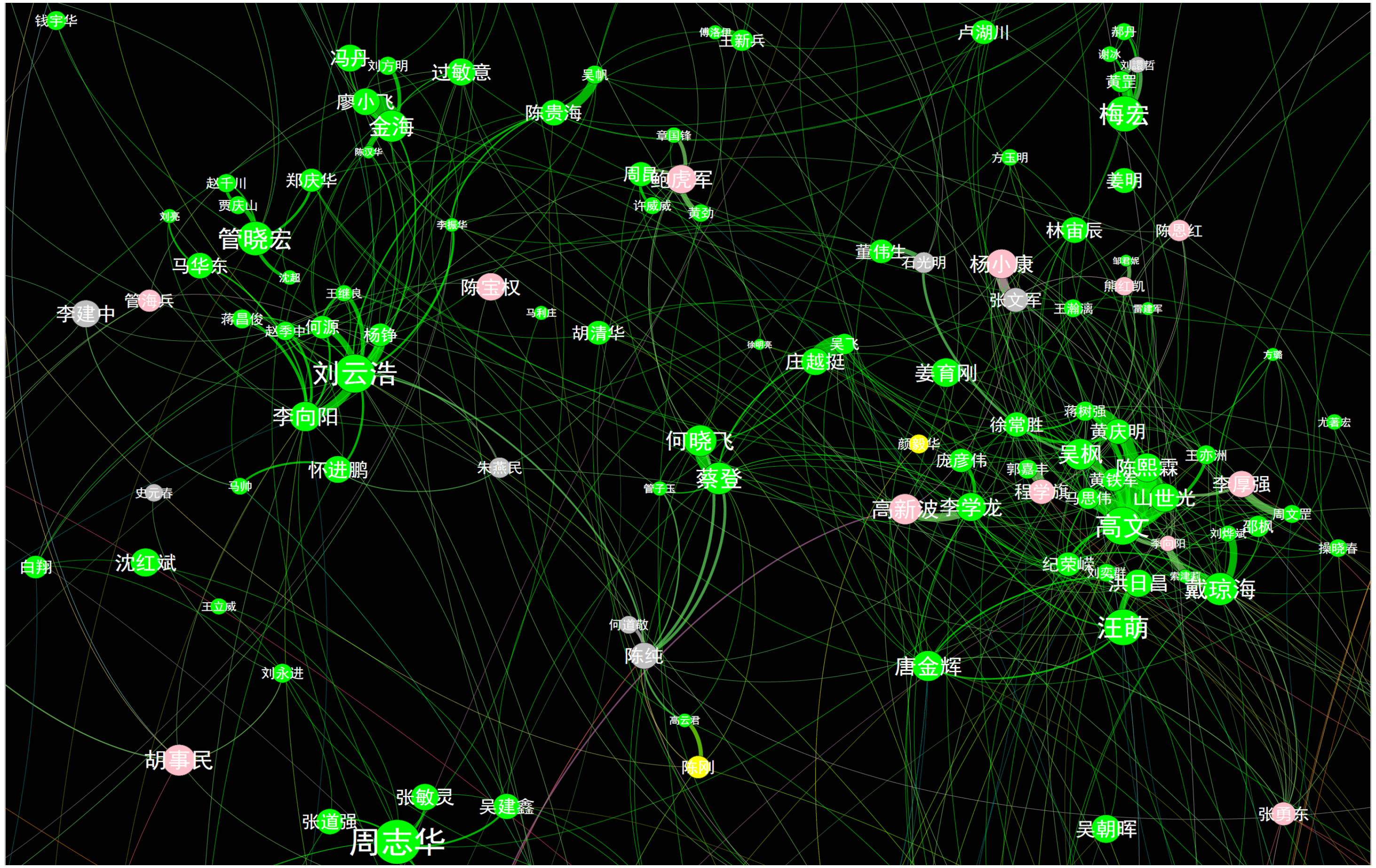
Graph Data: Media Networks



Connections between social media

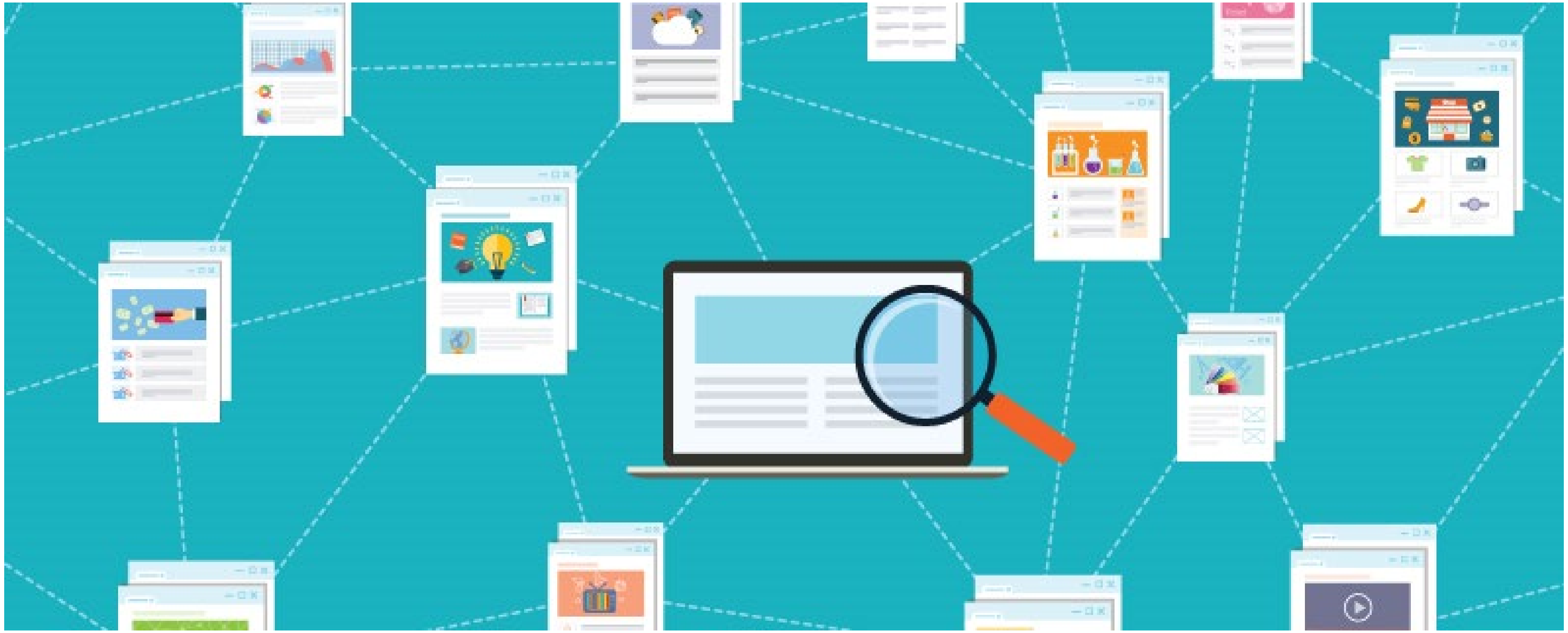
Polarization of the network

Graph Data: Academic Networks

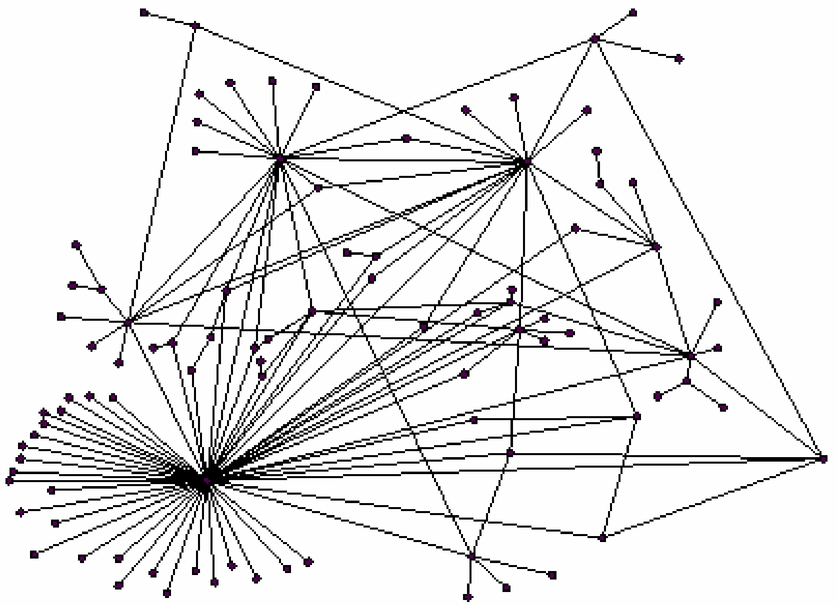


ACEMAP

Graph Data: Web Pages



Graph Algorithm



- To derive information from a graph, we ask
  - Vertex：
    - How important is a vertex? Pagerank
    - Any features? Node classification
  - Edge：
    - How important is a link? Betweenness centrality, etc.
    - Any potential links? Link prediction, recommendation
  - Structure：
    - How is the graph connected? Community detection
    - Can we represent nodes/links in vector space? Representation Learning

PageRank

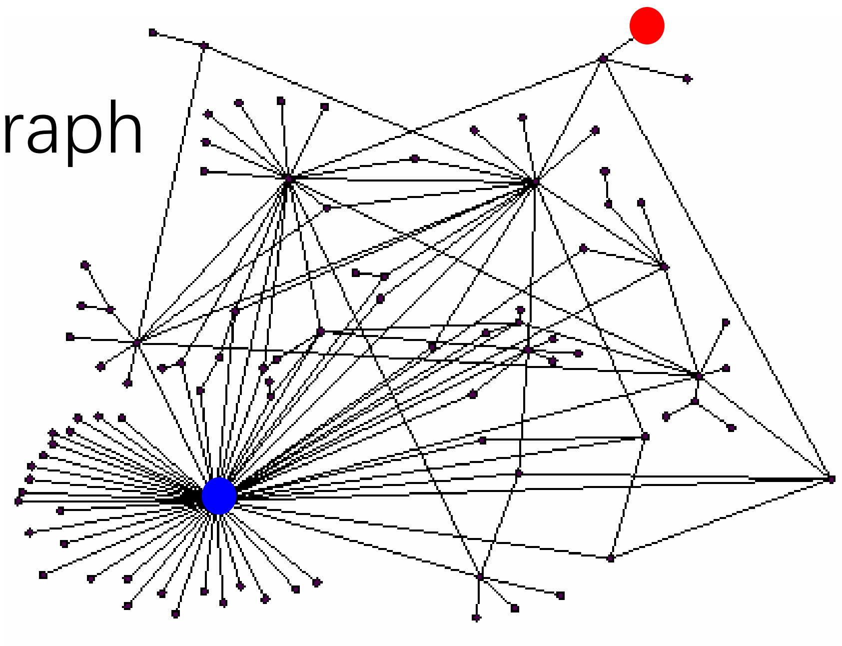
Challenges

- How to organize the Web?
  - Information Retrieval: Find **best** answer, (**relevant** docs in a small and trusted set), in **huge** number of websites, full of untrusted documents, random things, web spam, etc.
- **Meaurements:**
  - Who to **"trust"**?
    - Trustworthy pages may point to each other.
  - What is the **"best"** answer to a query?
    - Analyze the structure of the graph to get popular or high-valued answer.

Ranking Nodes on the Graph

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

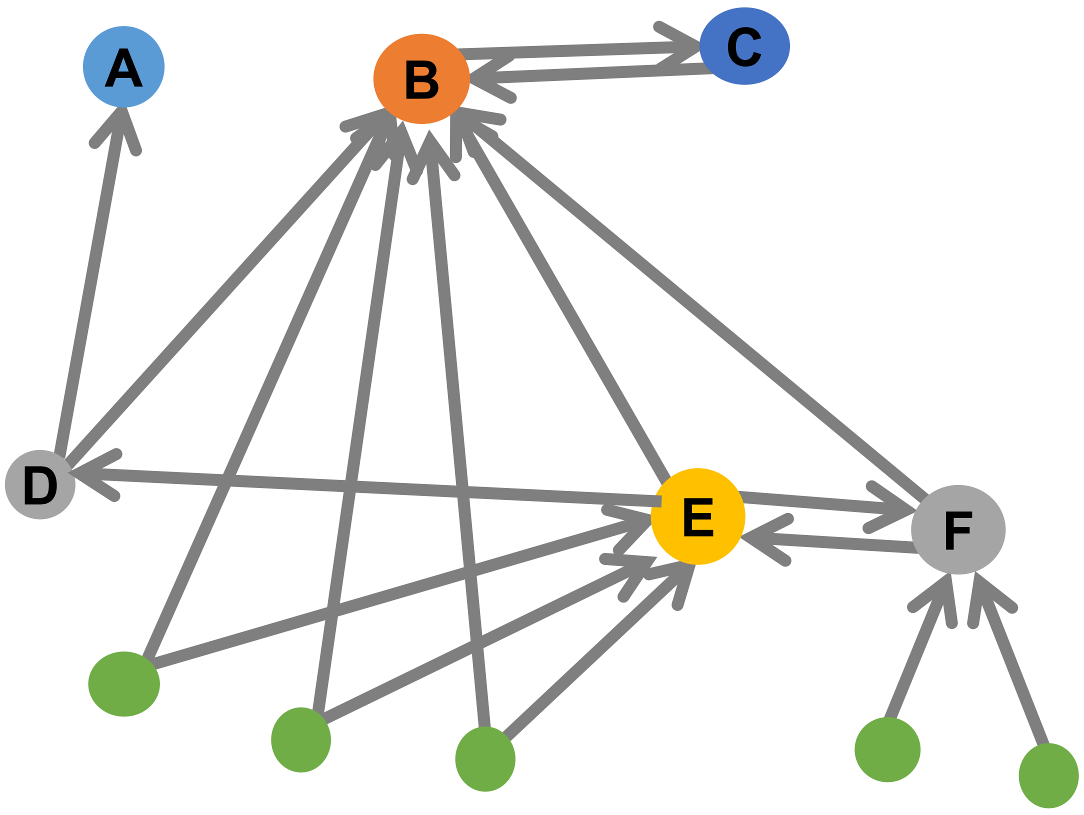
- There is large diversity in the web-graph node connectivity.
  - **rank the pages by the link structure**



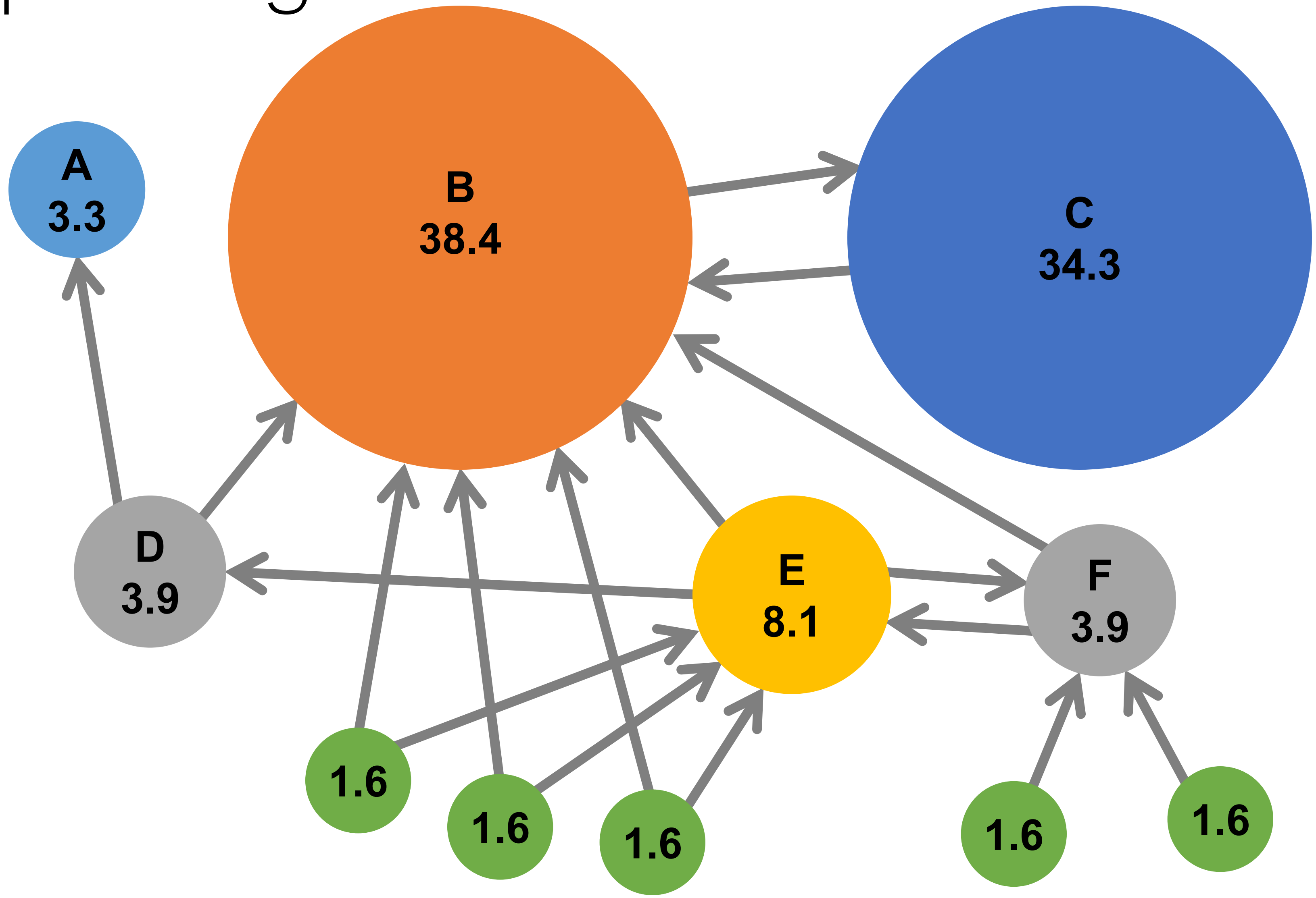
- **Page Rank**
  - Ranking the importance of a node

Links as Votes

- Idea: **Links as votes**
  - Page is more important if it has more links
    - In-coming links? Out-going links?
- Are all in-links are **equal**?
  - Links from **important** pages count more
  - **Recursive** question



Example: PageRank Scores



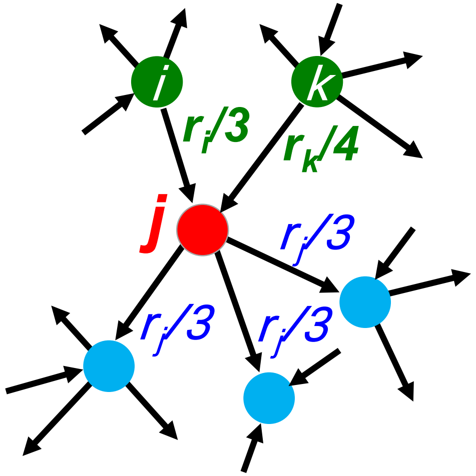
Simple Recursive Formulation

- 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

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

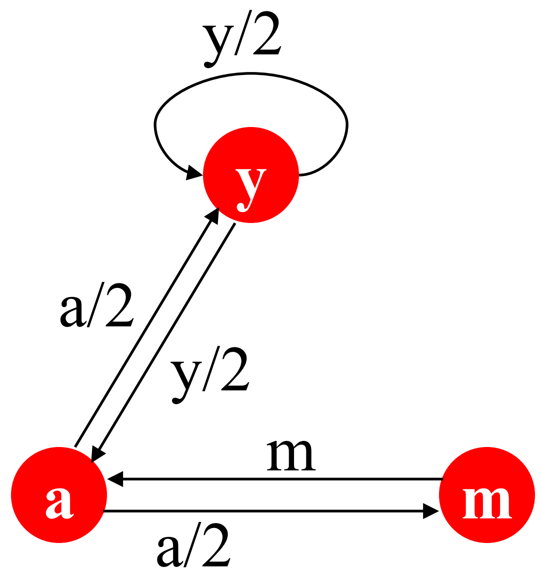


PageRank: The "Flow" Model

- 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"/"importance" r_j for page j

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

d_i* … out-degree of node *i



"Flow" equations:

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

Solving the Flow Equations

- **3 equations**

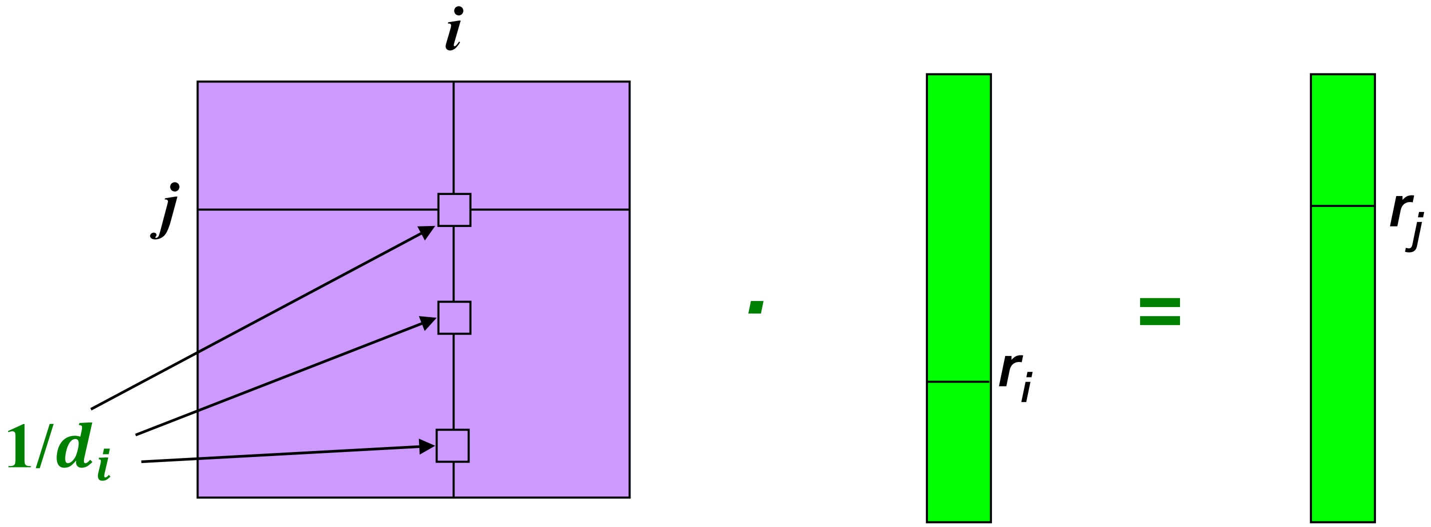
$$\mathbf{r_y = r_y/2 + r_a/2}$$
$$\mathbf{r_a = r_y/2 + r_m}$$
$$\mathbf{r_m = r_a/2}$$

- **Additional constraint forces uniqueness:**
  - $\boldsymbol{r_y + r_a + r_m = 1}$
  - Solution: $\boldsymbol{r_y = \frac{2}{5},\ r_a = \frac{2}{5},\ r_m = \frac{1}{5}}$

PageRank: Matrix Formulation

	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

- **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
  - **The flow equations can be written**



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

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

PageRank: Matrix Formulation

- **Stochastic adjacency matrix $\boldsymbol{M}$**
  - Let page i has d_i out-links
  - If $i \rightarrow j$, then $M_{ji} = \frac{1}{d_i}$ else $M_{ji} = 0$
    - $\boldsymbol{M}$ is a **column stochastic matrix**
      - Columns sum to 1
  - **The flow equations can be written**

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

- **Rank vector $\boldsymbol{r}$:** vector with an entry per page
  - r_i is the importance score of page i
  - $\sum_i r_i = 1$

Eigenvector Formulation

- **The flow equations can be written**

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

NOTE: $\boldsymbol{x}$ is an eigenvector with the corresponding eigenvalue $\boldsymbol{\lambda}$ if:

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

- So the **vector** *r* is an **eigenvector** of the stochastic web matrix *M*
  - Largest eigenvalue of *M* is **1** since *M* is column stochastic (with non-negative entries)

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

Power Iteration Method

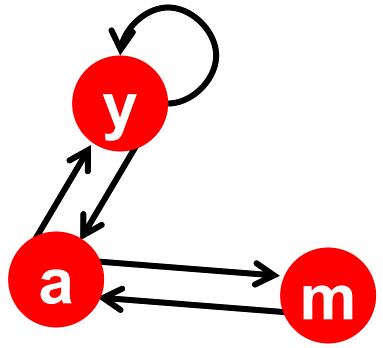
- Given a web graph with n nodes, where the nodes are pages and edges are hyperlinks
- **Power iteration:** a simple iterative scheme
  - Suppose there are N web pages
  - **Initialize**: $\mathbf{r}^{(0)} = [1/N, \cdots, 1/N]^T$
  - **Iterate**: $\mathbf{r}^{(t+1)} = \mathbf{M} \cdot \mathbf{r}^{(t)}$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

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

d_i …. out-degree of node i

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

Example: Flow Equations & M



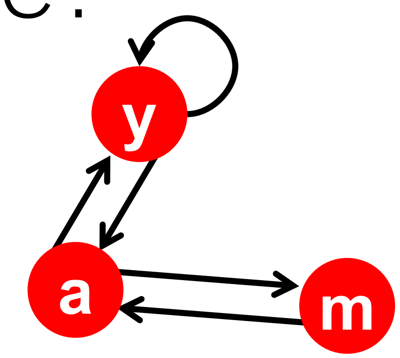
$$\begin{array}{c|ccc} & \mathbf{y} & \mathbf{a} & \mathbf{m} \\ \hline \mathbf{y} & \frac{1}{2} & \frac{1}{2} & 0 \\ \mathbf{a} & \frac{1}{2} & 0 & 1 \\ \mathbf{m} & 0 & \frac{1}{2} & 0 \end{array}$$

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

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

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

PageRank: How to solve?



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

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

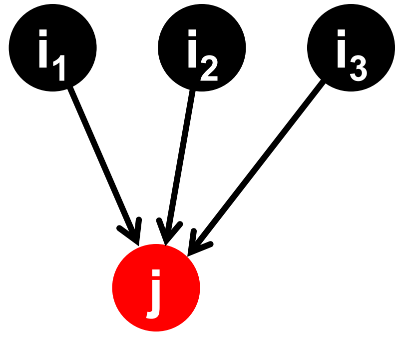
- **Power Iteration:**
  - Set $r_j = 1/N$
  - **1:** $r'_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - **2:** $r = r'$
  - Goto **1**
- **Example:**

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

Iteration 0, 1, 2, …

Random Walk Interpretation

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

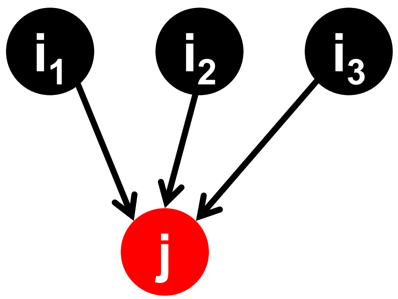


$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d_{out}(i)}}$$

The Stationary Distribution

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

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



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

- Suppose the random walk reaches a state

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

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

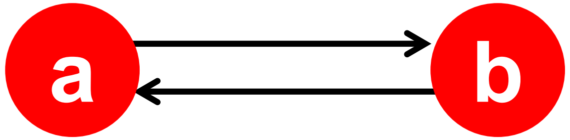
Existence and Uniqueness

- **A central result from the theory of random walks:**

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

Observation: Does this converge?

Periodic:



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

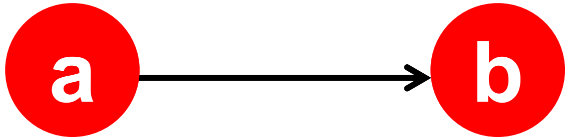
- **Example:**

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

Iteration 0, 1, 2, …

Observation: Does it converge to what we want?

Reducible:



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

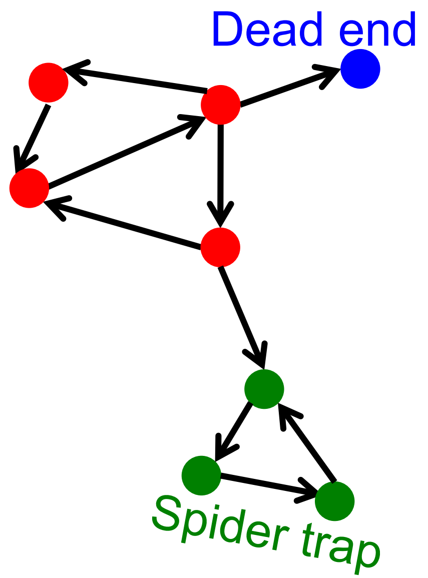
- **Example:**

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

Iteration 0, 1, 2, …

PageRank: Problems

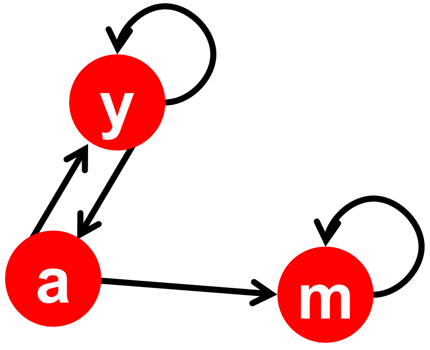
- **Spider traps** (all out-links are within the group)
  - Random walked gets “stuck” in a trap
  - Eventually spider traps absorb all importance
  - **Periodic**



- **Dead ends** (have no out-links)
  - Random walk has “nowhere” to go to
  - Such pages cause importance to “leak out”
  - **Reducible**

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

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

- **Example:**

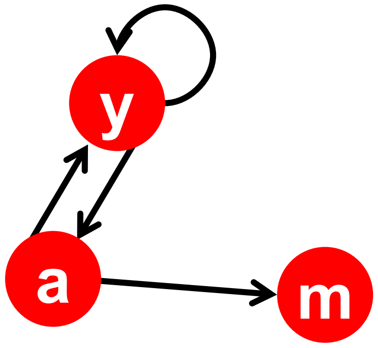
$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & \mathbf{0} \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & \mathbf{0} \\ 1/3 & 3/6 & 7/12 & 16/24 & & \mathbf{1} \end{matrix}$$

Iteration 0, 1, 2, …

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

Problem: Dead Ends

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



	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

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

$\mathbf{r_a = r_y/2}$

$\mathbf{r_m = r_a/2}$

- **Example:**

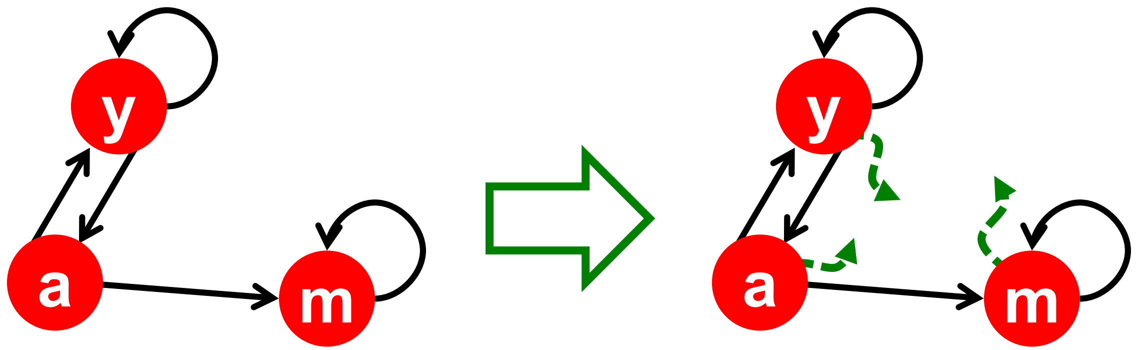
$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & \mathbf{0} \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & \mathbf{0} \\ 1/3 & 1/6 & 1/12 & 2/24 & & \mathbf{0} \end{matrix}$$

Iteration 0, 1, 2, …

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

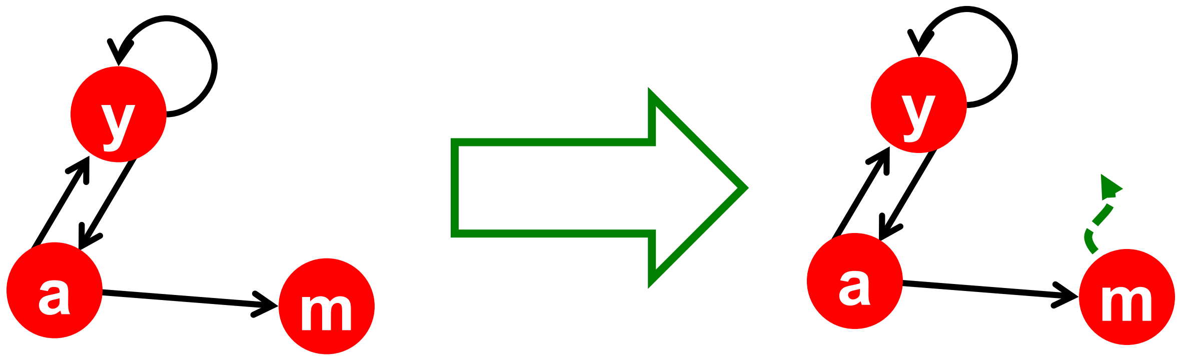
Solution: Teleports!

- The Google **solution** for **spider traps**: At each time step, the random surfer has two options
  - With prob. β, follow a neighbor 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



Solution: Teleport!

- **Teleports** also solves **dead-ends**
  - 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?

- **Spider-traps**
  - **Solution:** Never get stuck in a spider trap by teleporting out of it in a finite number of steps
- **Dead-ends**
  - 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 $\boldsymbol{\beta}$, follow a link at random
  - With probability $\boldsymbol{1\text{-}\beta}$, 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

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

The Google Matrix

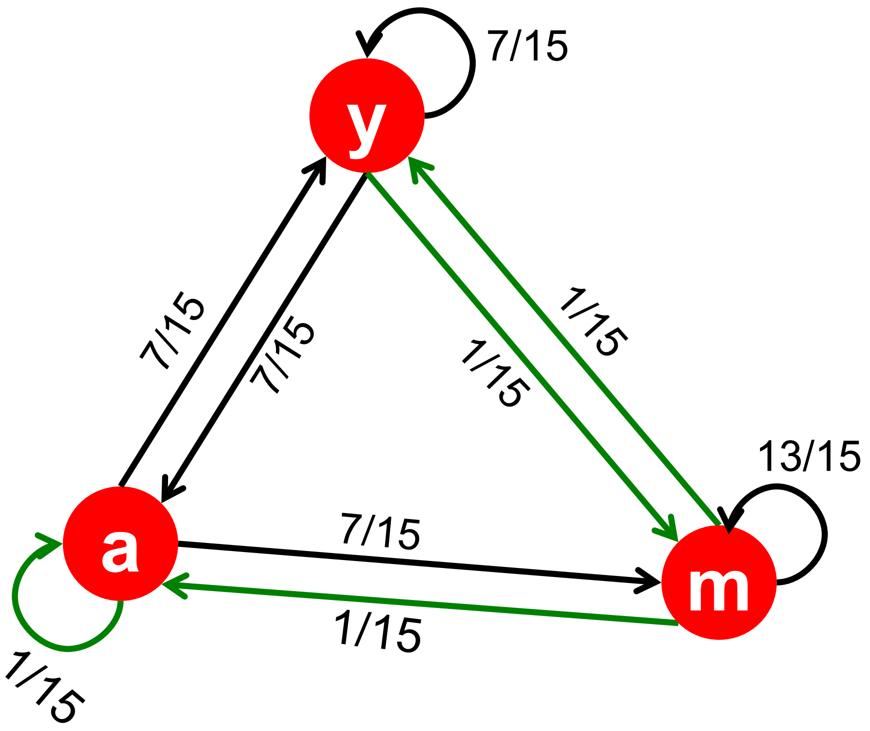
- **The Matrix *A*:**

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

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

- We have a recursive problem: $\boldsymbol{r} = \boldsymbol{A} \cdot \boldsymbol{r}$
  And the **Power Iteration method** still works!

Random Teleports ($\beta = 0.8$)



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

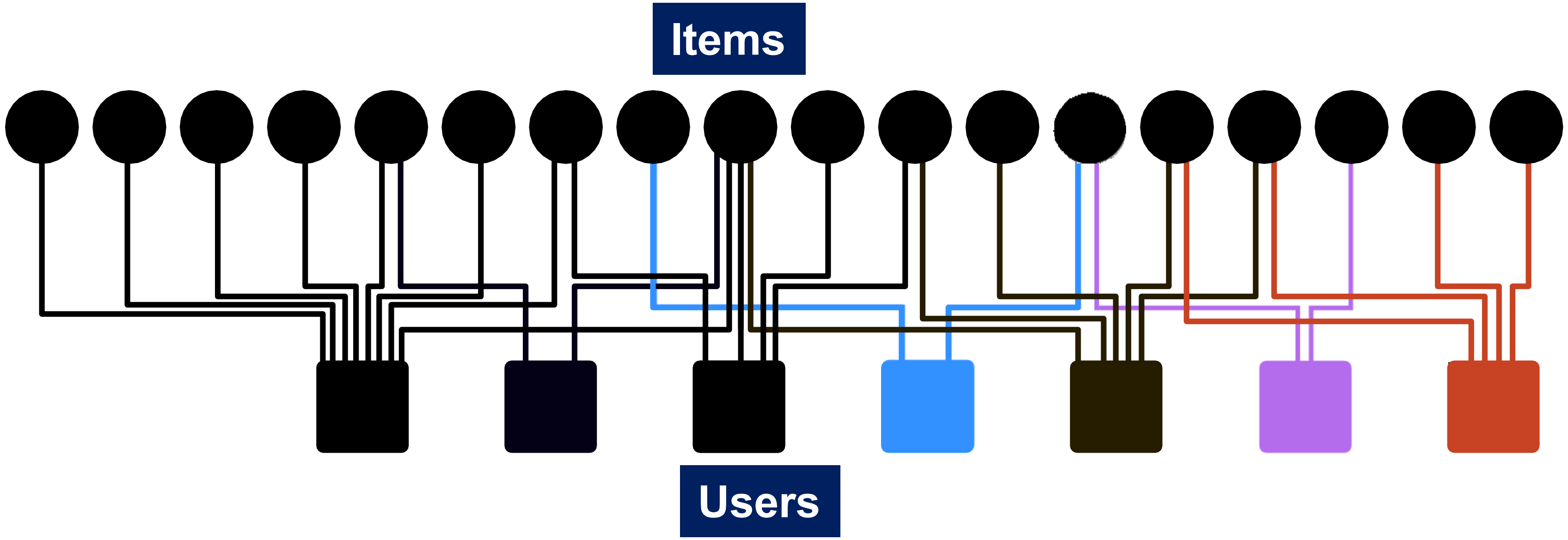
$$
\mathbf{A} = \begin{array}{c} y \\ a \\ m \end{array} \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}
$$

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

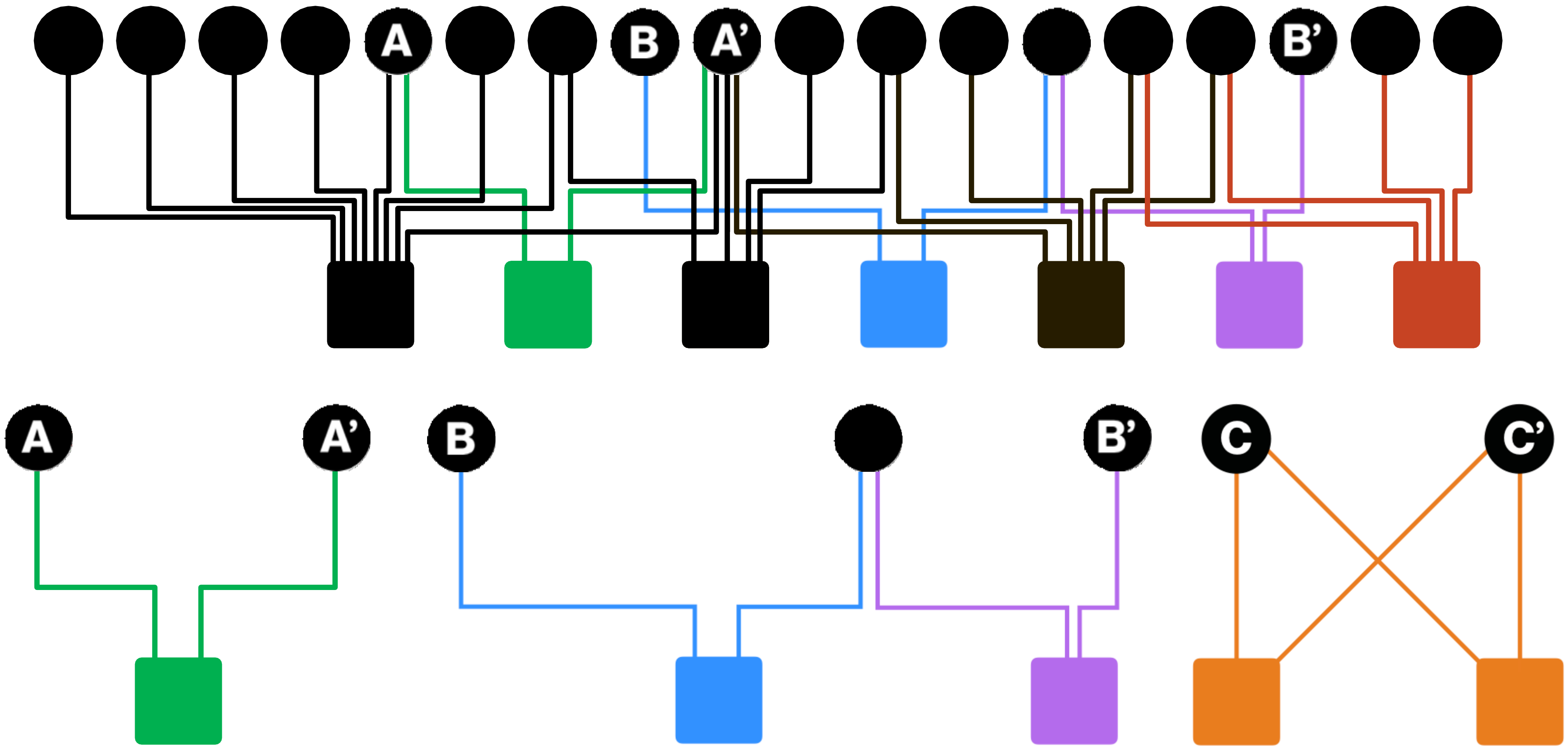
PageRank for Proximity in Graphs

Proximity on Graphs （Recommendation）

- Given: a **bipartite graph** representing user and item interactions
  - Users purchase items
- Goal: What items should we recommend to a user who interacts with a item Q?
- Intuition: find the **similar** items.



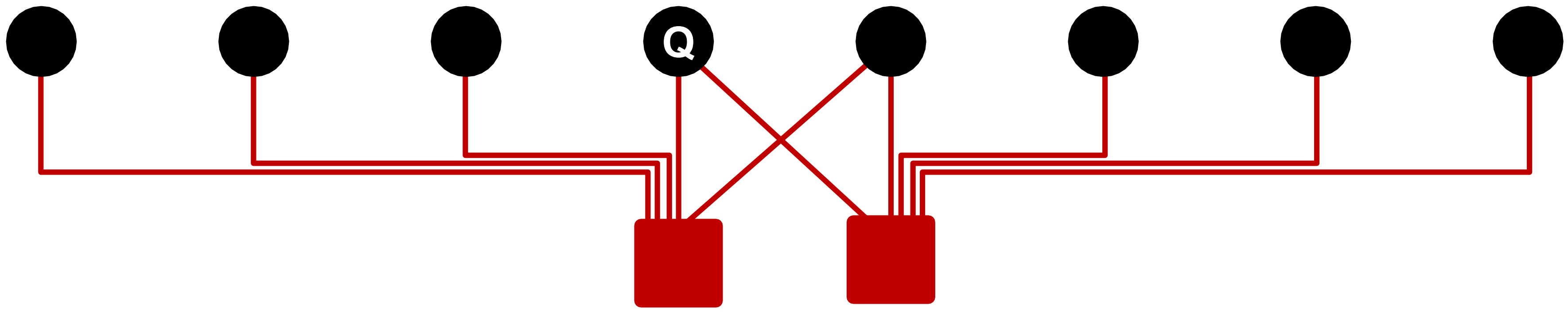
Example: which pair is more similar?



Node Proximity Measurements

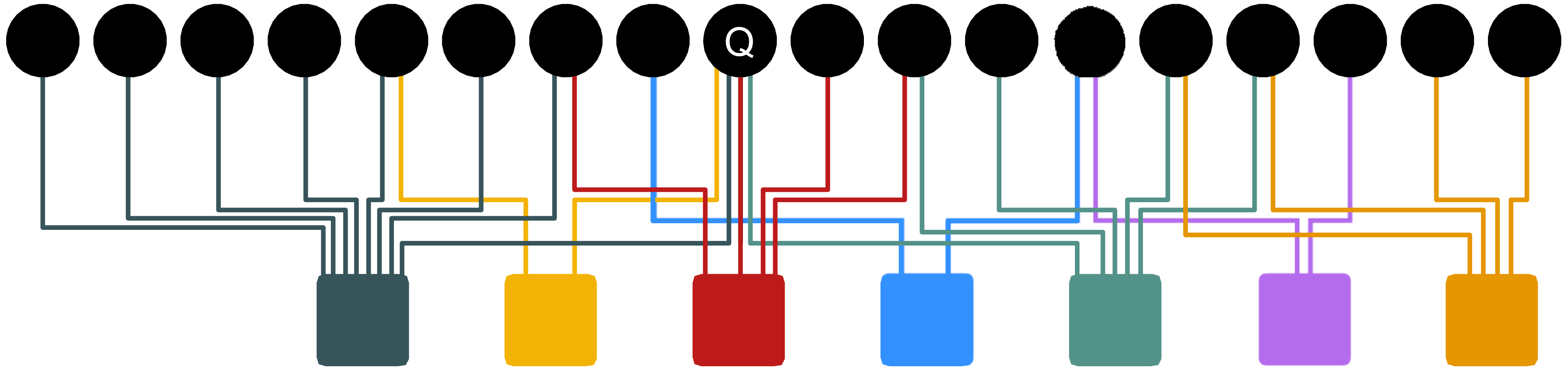
- How to measure in complicated graphs?
- **Random walk with restarters.**
  - Modified PageRank which **teleports back to the starting node**(Q).

(for each node the teleport vector S=[0, 0, 0, 0, **1**, 0, 0, 0, 0, 0, 0])



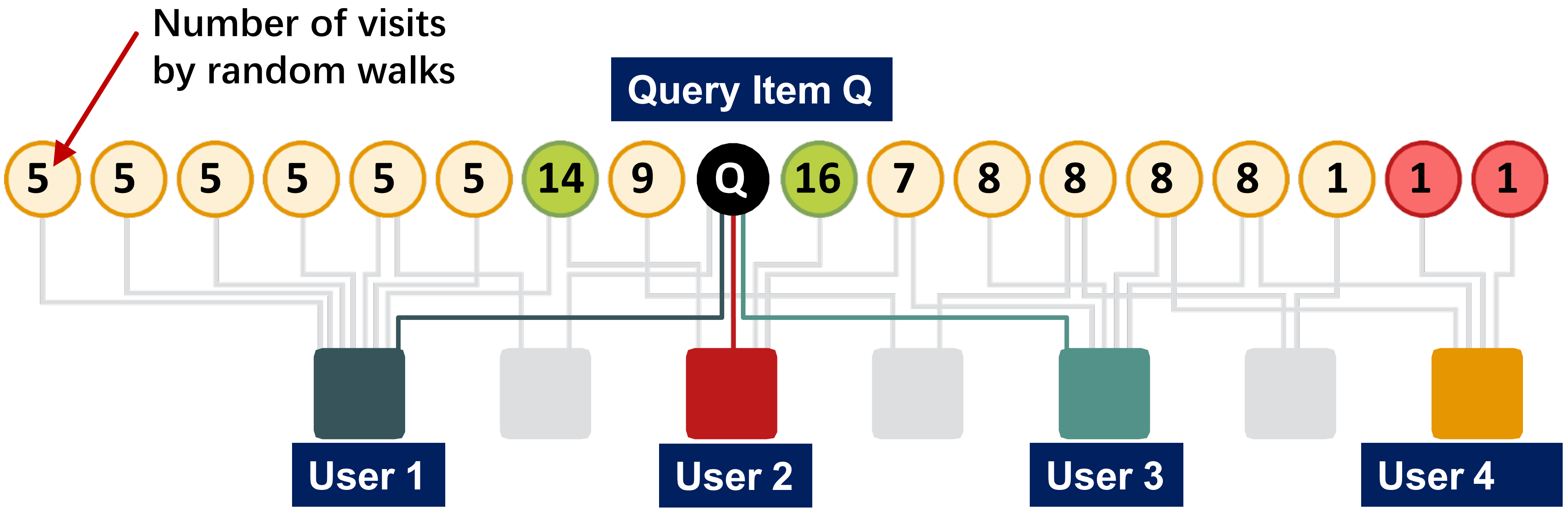
Random Walk with Restarters

- Simulate a random walk, from items to users back to items.
- With probability α, restart the random walk from Q.
- Resulting scores measures similarity to node Q.



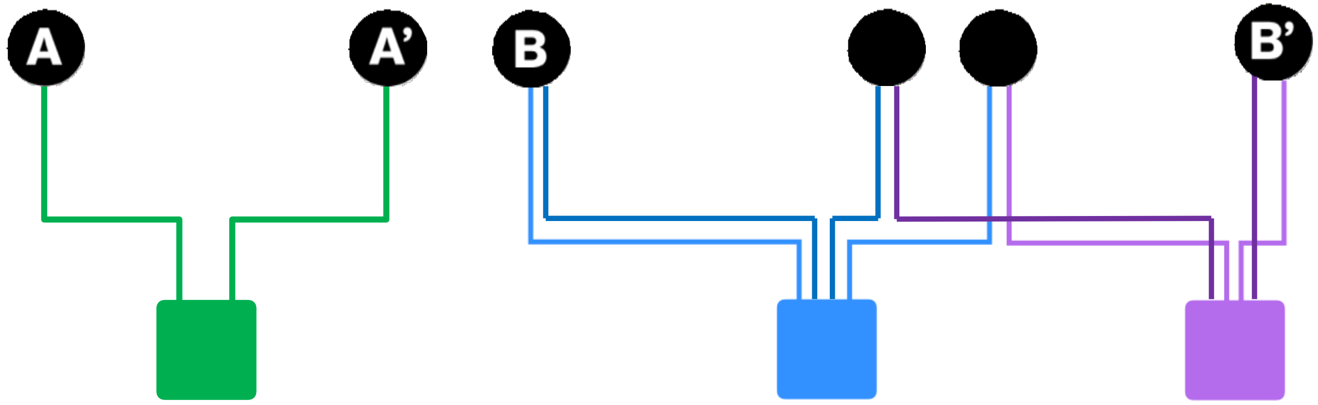
Random Walk with Restarters

- Simulate a random walk, from items to users back to items.
- With probability α, restart the random walk from Q.
- Resulting scores measures similarity to node Q.



Summary

- Random with restarters: PageRank teleporting back to the same node
- The similarity considers:
  - Multiple connections
  - Multiple paths
  - Degree of the nodes
  - Direct and indirect connections
- But we need run the algorithm for every node



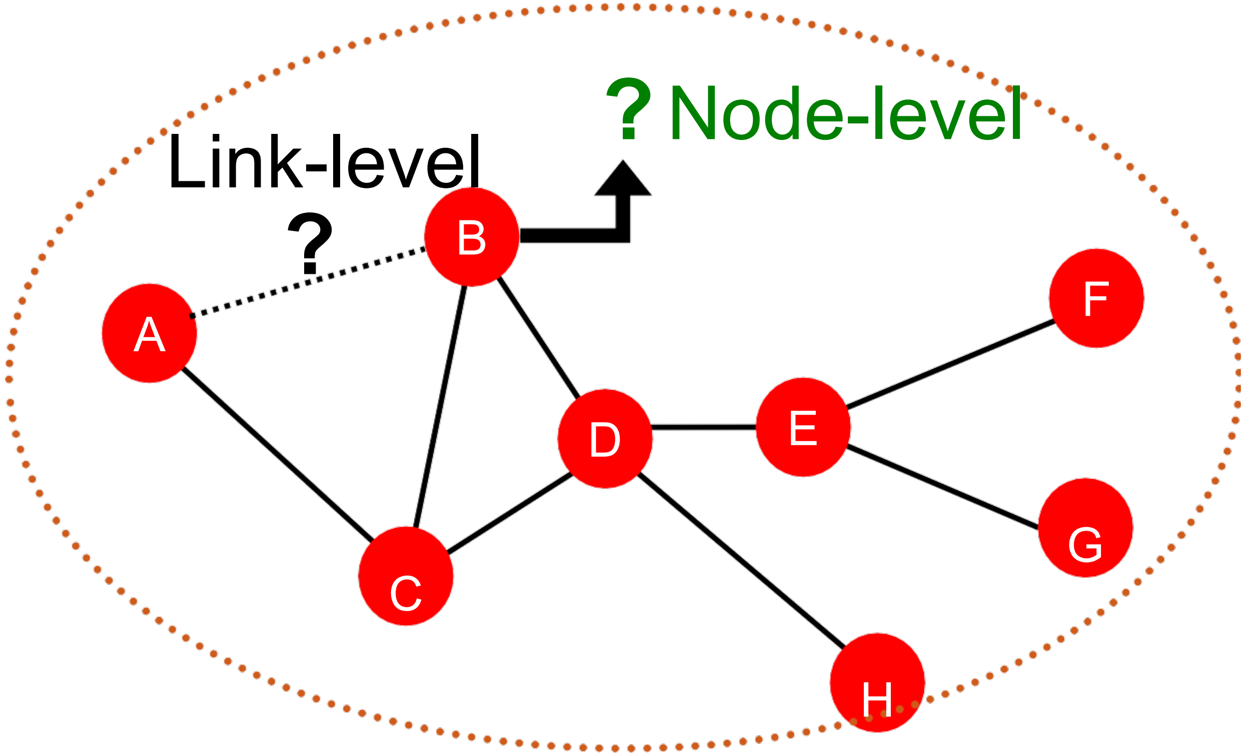
Summary of PageRank

- A **graph** is naturally represented as a **matrix**, we defined a **random walk** process over the graph
  - **Stochastic** adjacency matrix M
  - Random surfer moving across the links and with random **teleportation**
- PageRank: limiting **distribution** of the surfer location represented node **importance**
  - Corresponds to the **leading eigenvector** of transformed adjacency matrix M
  - **Power iteration**

Graph Features

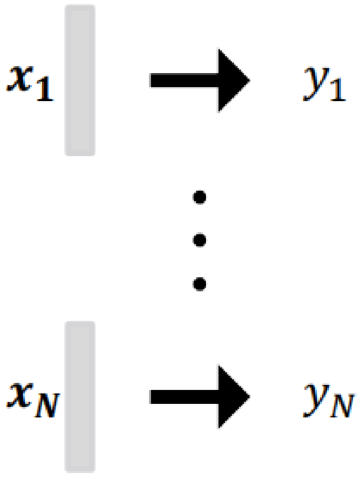
Graph Tasks

- **Node-level** prediction
- **Link-level** prediction



Machine Learning Pipeline

- **ML models:**
  - Random forest
  - SVM
  - Neural network, etc.
- **Apply the model:**
  - Given raw data, obtain its features, train models, and make prediction

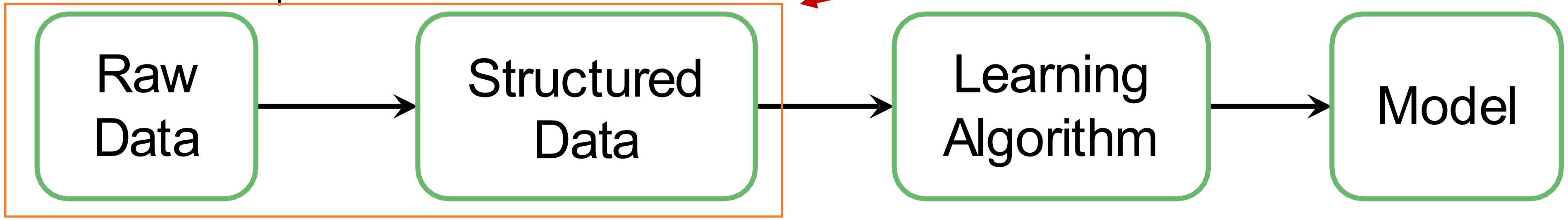


Machine learning models

$$\boxed{y} = \boxed{f}(\underline{x})$$

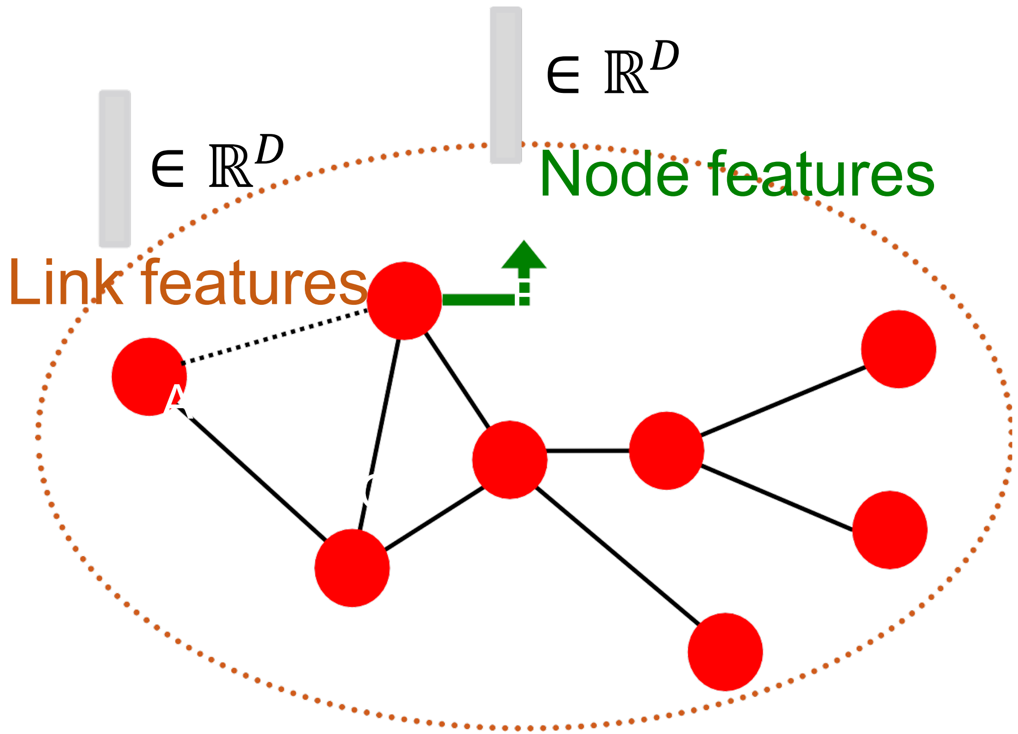
Prediction /decision

Features



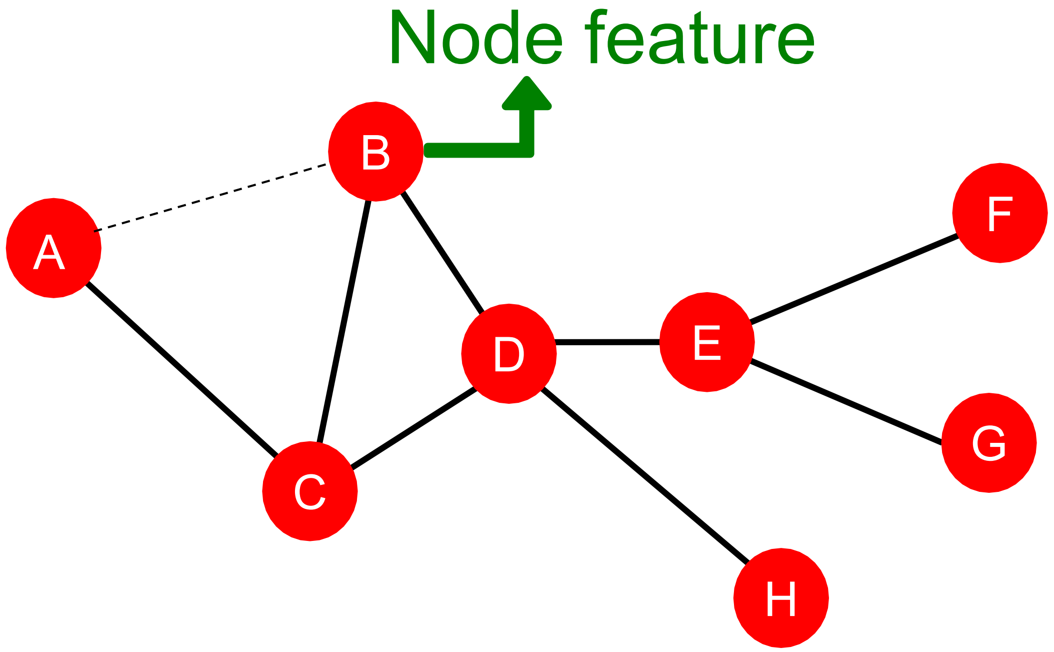
Machine Learning on Graphs

- Design **features** for nodes/links
  - **Features: $\boldsymbol{D}$**-dimensional vectors
- Apply machine learning on the feature vectors to make prediction



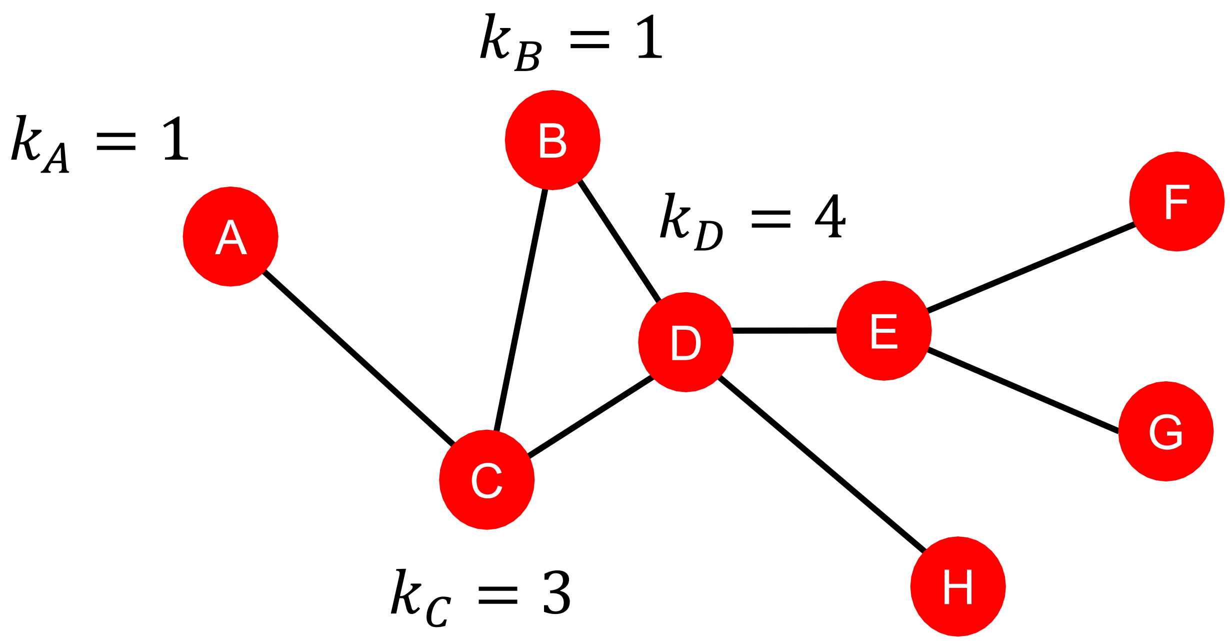
Node-Level Features: Overview

- **Goal:** Characterize the **structure** and **position** of a node in the network:
  - Node degree
  - Node centrality
  - Clustering coefficient
  - Graphlets



Node Features: Node Degree

- The degree k_v of node v is the number of **edges** (neighboring nodes) the node has.
- Treats all neighboring nodes equally.



Node Features: Node Centrality

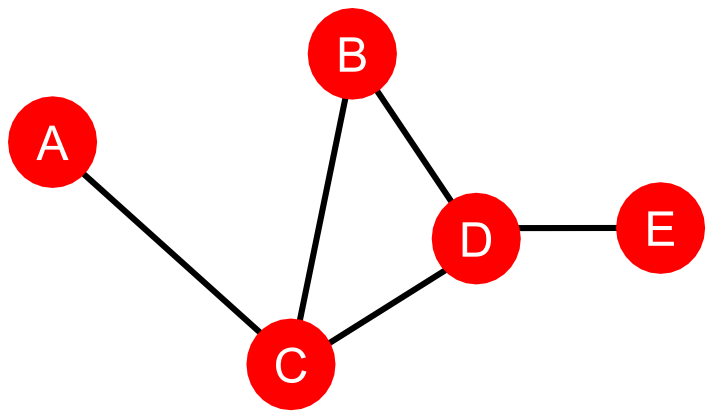
- Node degree counts the neighboring nodes without capturing their importance.
- **Node centrality** c_v takes the node importance in a graph into account
- **Different ways to model importance:**
  - PageRank
  - Betweenness centrality
  - Closeness centrality

Node Centrality

- **Betweenness centrality:**
  - A node is important if it lies on many shortest paths between other nodes.

$$c_v = \sum_{s \neq v \neq t} \frac{\#(\text{shortest paths betwen } s \text{ and } t \text{ that contain } v)}{\#(\text{shortest paths between } s \text{ and } t)}$$

  - Example:



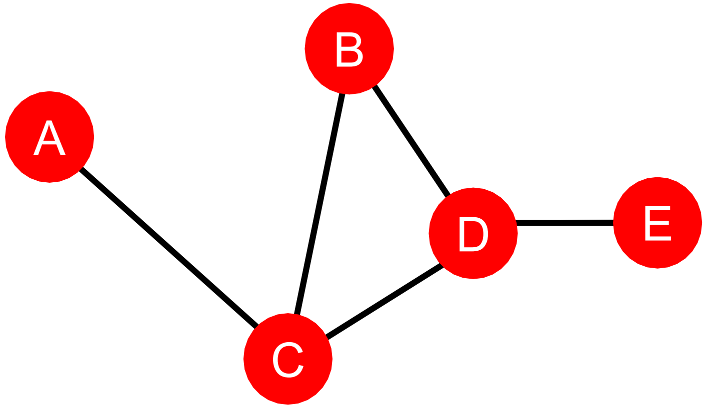
$$c_A = c_B = c_E = 0$$
$$c_C = 3$$
(A-C-B, A-C-D, A-C-D-E)
$$c_D = 3$$
(A-C-D-E, B-D-E, C-D-E)

Node Centrality

- **Closeness centrality:**
  - A node is important if it has small shortest lengths to all other nodes.

$$c_v = \frac{1}{\sum_{u \neq v} \text{length of the shortest path between } u \text{ and } v}$$

- **Example:**



$c_A = 1/(2 + 1 + 2 + 3) = 1/8$
(A-C-B, A-C, A-C-D, A-C-D-E)

$c_D = 1/(2 + 1 + 1 + 1) = 1/5$
(D-C-A, D-B, D-C, D-E)

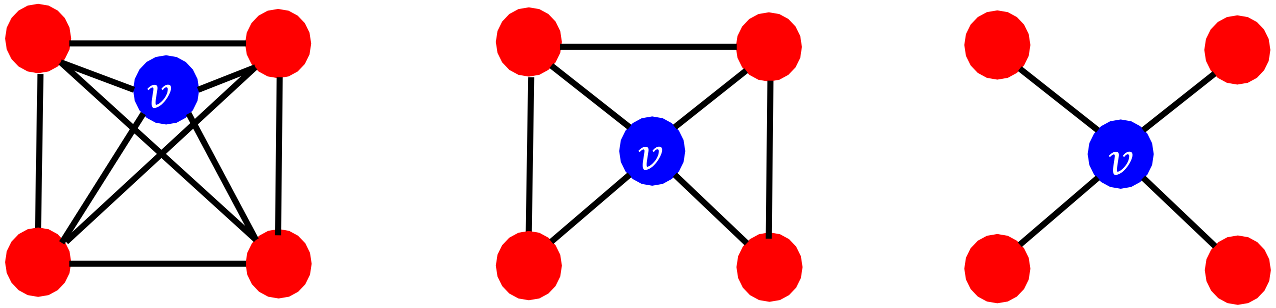
Node Features: Clustering Coefficient

- Measures how connected v's neighboring nodes are:

$$e_v = \frac{\#(\text{edges among neighboring nodes})}{\binom{k_v}{2}} \in [0,1]$$

#(node pairs among k_v neighboring nodes)

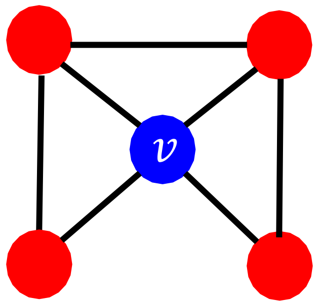
- **Examples:**



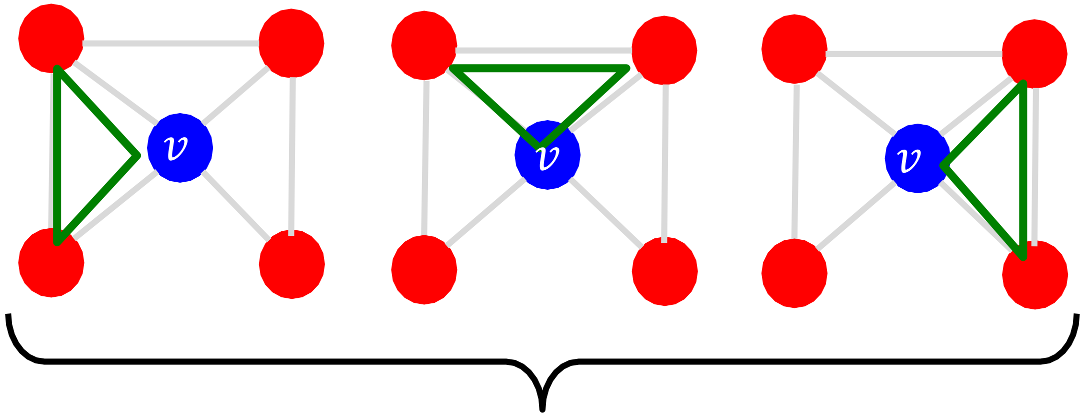
$e_v = 1$ $e_v = 0.5$ $e_v = 0$

Node Features: Graphlets

- **Observation:** Clustering coefficient counts the #(triangles) in the ego-network



$e_v = 0.5$

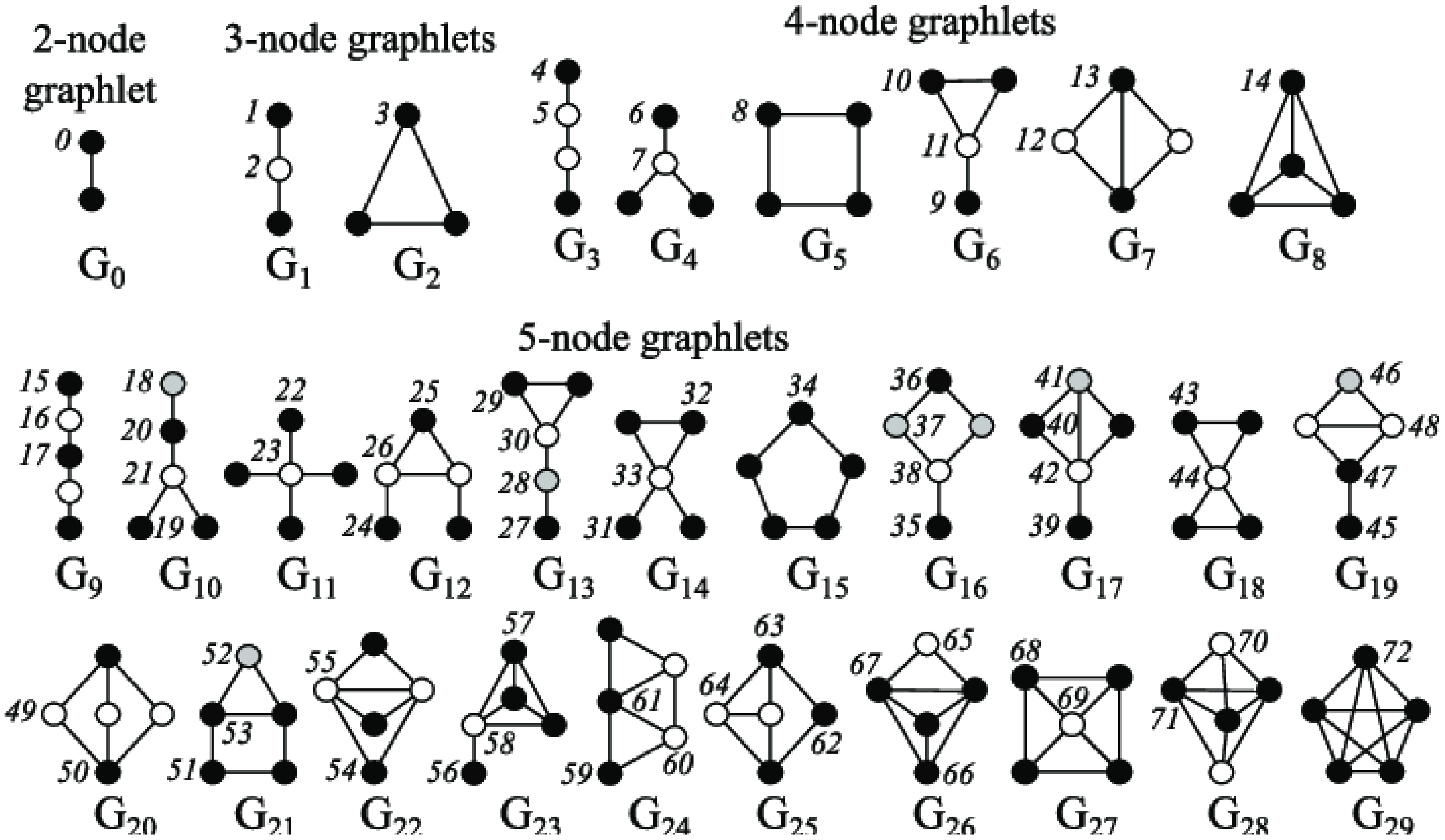


3 triangles (out of 6 node triplets)

- We can generalize the above by counting #(pre-specified subgraphs, i.e., **graphlets**).

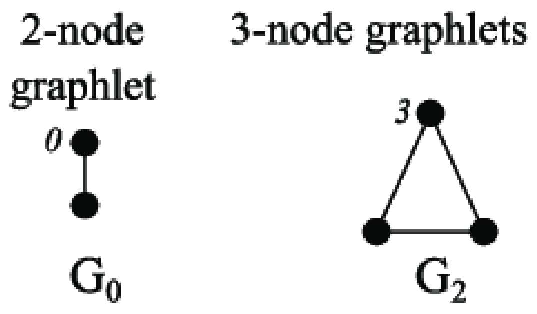
Node Features: Graphlets

- **Graphlets:** Rooted connected non-isomorphic subgraphs:
  - The indices of nodes represent all possible node types regarding topology



Node Features: Graphlets

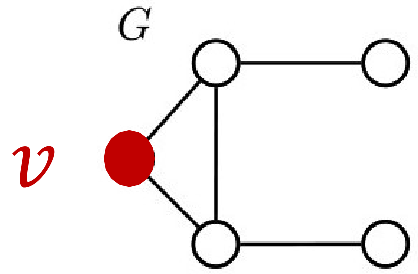
- **Graphlet Degree Vector (GDV):** Graphlet-base features for nodes
  - **Degree** counts **#(edges)** that a node touches
  - **Clustering coefficient** counts **#(triangles)** that a node touches.



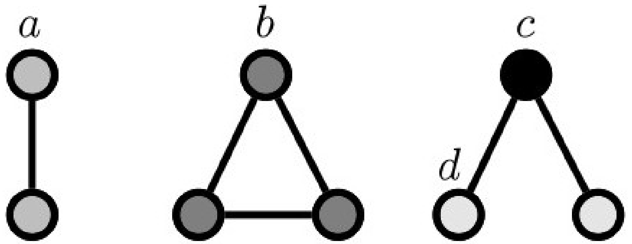
  - **GDV** counts **#(graphlets)** that a node touches

Node Features: Graphlets

- **Graphlet Degree Vector (GDV):** A count vector of graphlets rooted at a given node.
- **Example:**

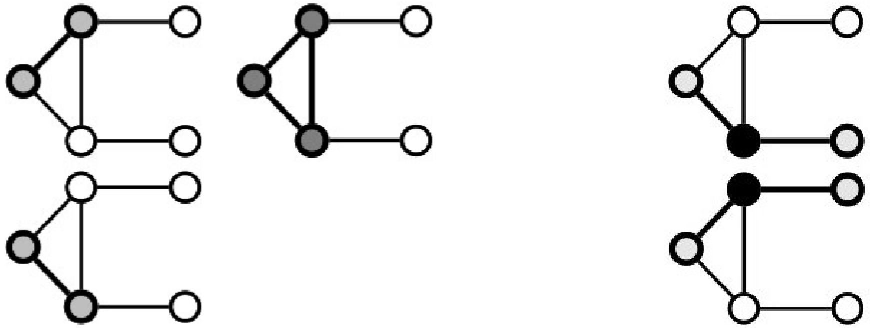


List of graphlets



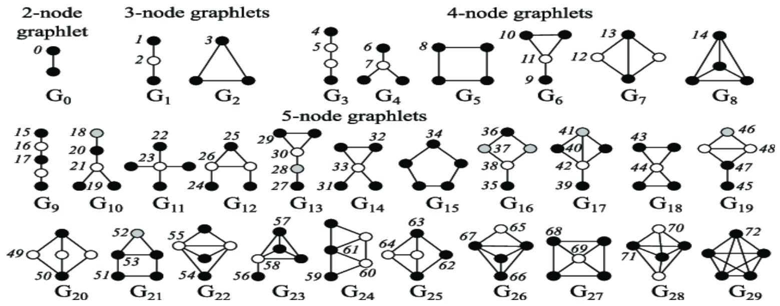
Graphlet instances:

a b c d



GDV of node v:
a, b, c, d
[2,1,0,2]

Node Features: Graphlets

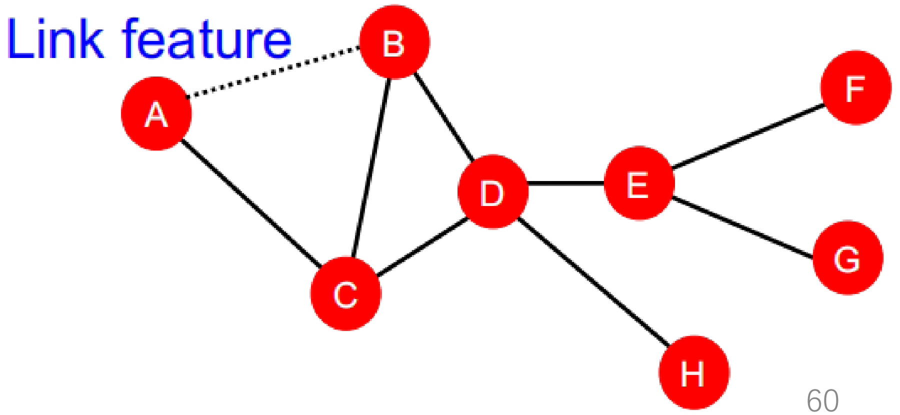


- Considering graphlets on 2 to 5 nodes we get:
  - **Vector of 73 coordinates** is a signature of a node that describes the topology of node's neighborhood
  - Captures its interconnectivities out to a **distance of 4 hops**

- Graphlet degree vector provides a measure of a **node's local network topology:**
  - Comparing vectors of two nodes provides a more detailed measure of local topological similarity than node degrees or clustering coefficient.

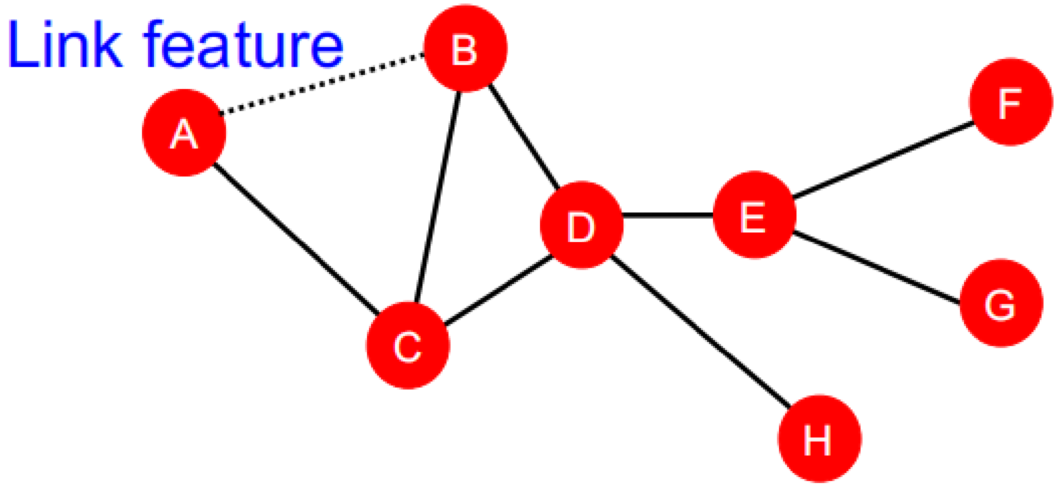
Link Prediction via Proximity

- **Methodology:**
  - For each pair of nodes (x, y) compute score $c(x, y)$
    - For example, $c(x, y)$ could be the # of common neighbors of x and y
  - Sort pairs (x, y) by the decreasing score $c(x, y)$
  - Predict **top $\boldsymbol{n}$** pairs as new links



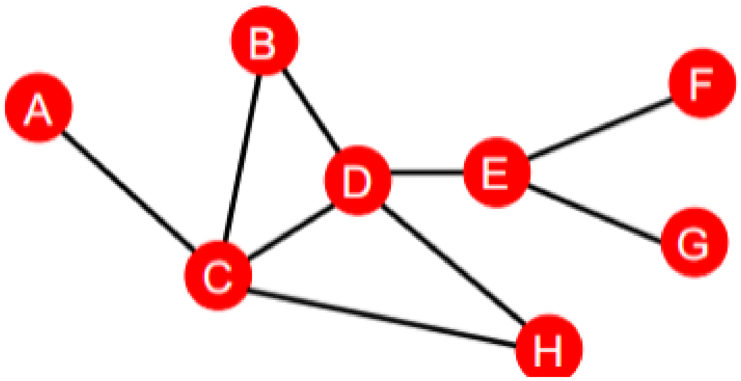
Link-Level Features: Overview

- Distance-based feature
- Local neighborhood overlap
- Global neighborhood overlap



Distance-Based Features

- **Shortest-path distance between two nodes**
- Example:



$S_{BH} = S_{BE} = S_{AB} = 2$

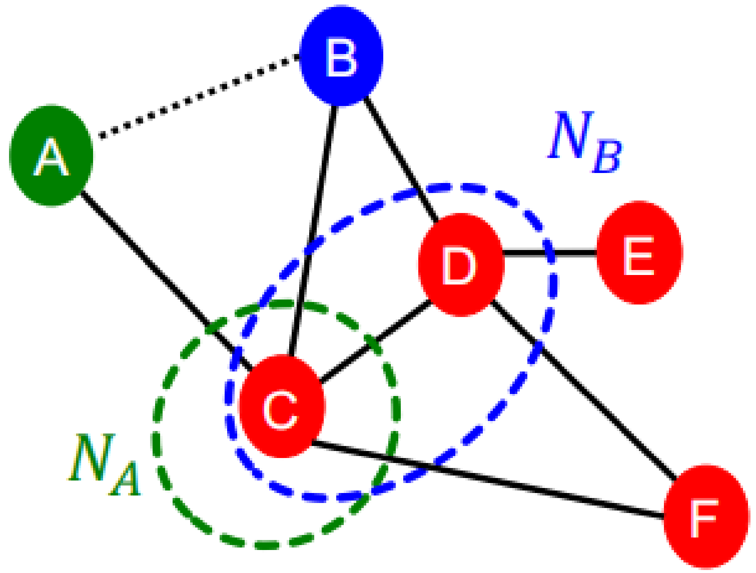
$S_{BG} = S_{BF} = 3$

BH and BE have the same shortest path distance.
If there will be a link between either BH or BE.
Which is more possible?

- However, this does not capture the degree of neighborhood overlap:
  - Node pair (B, H) has 2 shared neighboring nodes, while pairs (B, E) and (A, B) only have 1 such node.

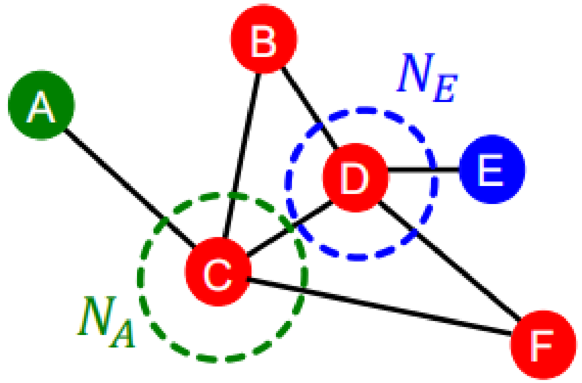
Local Neighborhood Overlap

- **Captures # neighboring nodes shared between two nodes v_1 and v_2:**
- **Common neighbors:** $|N(v_1) \cap N(v_2)|$
  - Example: $|N(A) \cap N(B)| = |\{C\}| = 1$
- **Jaccard's coefficient:** $\frac{|N(v_1) \cap N(v_2)|}{|N(v_1) \cup N(v_2)|}$
  - Example: $\frac{|N(A) \cap N(B)|}{|N(A) \cup N(B)|} = \frac{|\{C\}|}{|\{C,D\}|} = \frac{1}{2}$
- **Adamic-Adar index:** $\sum_{u \in N(v_1) \cap N(v_2)} \frac{1}{log(k_u)}$
  - k_u is degree of u
  - Example: $\frac{1}{log(k_C)} = \frac{1}{log4}$



Global Neighborhood Overlap

- **Limitation of local neighborhood features:**
  - Metric is always zero if the two nodes **do not have any neighbors** in common.



$$N_A \cap N_E = \phi$$
$$|N_A \cap N_E| = 0$$

  - However, the two nodes may still potentially be connected in the future.
- **Global neighborhood overlap** metrics resolve the limitation by considering the entire graph.

Global Neighborhood Overlap

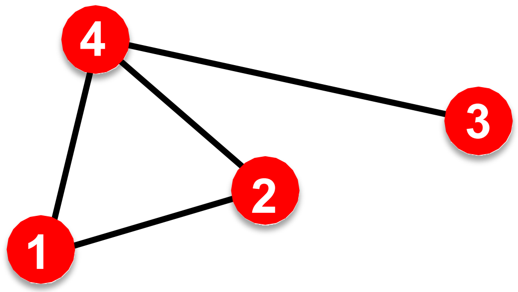
- **Katz index:** count the number of paths of all lengths between a given pair of nodes.

- **Q:** How to compute #paths between two nodes?
- Use **powers of the graph adjacency matrix!**

Intuition: Power of Adj Matrices

- **Computing #paths between two nodes**
  - Recall: $\boldsymbol{A_{uv}} = 1$ if $u \in N(v)$
  - Let $\boldsymbol{P_{uv}^{(K)}}$ = #paths of length $\boldsymbol{K}$ between $\boldsymbol{u}$ and $\boldsymbol{v}$
  - We will show $\boldsymbol{P^{(K)} = A^{k}}$
  - $\boldsymbol{P_{uv}^{(1)}}$ = #paths of length 1 (direct neighborhood) between u and $v = \boldsymbol{A_{uv}}$

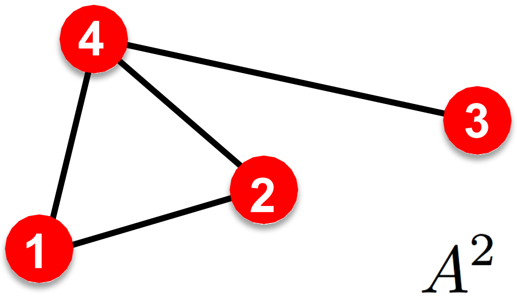
$$\boldsymbol{P_{12}^{(1)} = A_{12}}$$



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

Intuition: Power of Adj Matrices

- **How to compute $P_{uv}^{(2)}$?**
  - **Step 1:** Compute **#paths** of length 1 **between each of u's neighbor and v**
  - **Step 2:** **Sum up** these #paths across u's neighbors
  - $P_{uv}^{(2)} = \sum_i A_{ui} * P_{iv}^{(1)} = \sum_i A_{ui} * A_{iv} = A_{uv}^{2}$



Node 1's neighbors

#paths of length 1 between Node 1's neighbors and Node 2

$P_{12}^{(2)} = A_{12}^{2}$

$$A^2 = \begin{pmatrix} 0 & \boxed{1} & 0 & \boxed{1} \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & \boxed{0} & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & \boxed{1} & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & \boxed{1} & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 3 \end{pmatrix}$$

Power of adjacency

Global Neighborhood Overlap

- How to compute #paths between two nodes?
- Use **adjacency matrix powers**!
  - $\boldsymbol{A}_{uv}$ specifies #paths of length 1 (direct neighborhood) between u and v.
  - $\boldsymbol{A}_{uv}^{\boldsymbol{2}}$ specifies #paths of **length 2** (neighbor of neighbor) between u and v.
  - And, $\boldsymbol{A}_{uv}^{\boldsymbol{l}}$ specifies #paths of **length $\boldsymbol{l}$**.

Katz index:

Count the number of paths of all lengths between a pair of nodes.

- **Katz index** between v_1 and v_2 is calculated as

Sum over all path lengths

$$S_{v_1 v_2} = \sum_{l=1}^{\infty} \boxed{\beta^l} \boxed{\boldsymbol{A}^l_{v_1 v_2}}$$

#paths of length l between v_1 and v_2

$0 < \beta < 1$: discount factor

- Katz index matrix is computed in closed-form:

$$\boldsymbol{S} = \sum_{i=1}^{\infty} \beta^i \boldsymbol{A}^i = \underbrace{(\boldsymbol{I} - \beta \boldsymbol{A})^{-1}}_{= \sum_{i=0}^{\infty} \beta^i \boldsymbol{A}^i} - \boldsymbol{I}$$

by geometric series of matrices

Graph Feature Summary

- **Traditional ML Pipeline**
  - Hand-crafted feature + ML model
- **Hand-crafted features for graph data**
  - **Node-level:**
    - Node degree, centrality, clustering coefficient, graphlets
  - **Link-level:**
    - Distance-based feature
    - local/global neighborhood overlap