

# Stanford CS224W: Graph as Matrix: PageRank, Random Walks and Embeddings

CS224W: Machine Learning with Graphs  
Jure Leskovec, Stanford University  
<http://cs224w.stanford.edu>



# ANNOUNCEMENTS

- **Homework 1 will be released after class**
- **Next Thursday (10/07):** Colab 1 due, Colab 2 out
  - **Do Colab 0!** It has almost everything you need to complete Colab 1.
- **Office hours:** we've added Zoom links to our OH calendar.
  - See <http://web.stanford.edu/class/cs224w/oh.html> for OH calendar, Zoom links, and QueueStatus link.

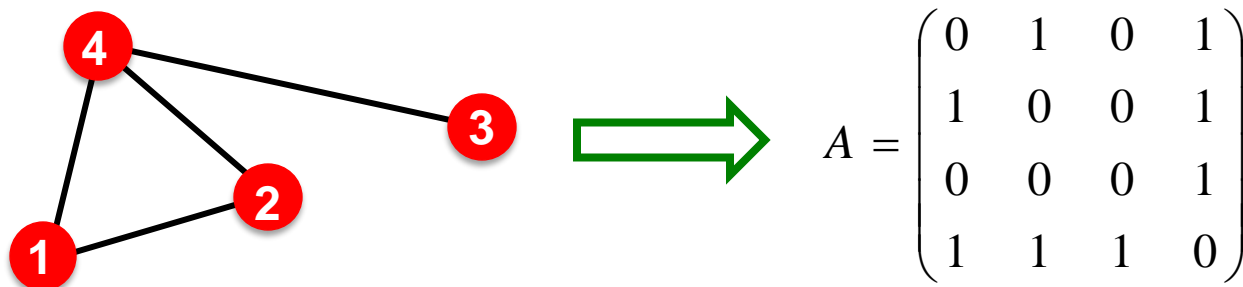
CS224W: Machine Learning with Graphs  
Jure Leskovec, Stanford University  
<http://cs224w.stanford.edu>



# Graph as Matrix

In this lecture, we investigate graph analysis and learning from a matrix perspective.

- Treating a graph as a matrix allows us to:
  - Determine node importance via **random walk** (PageRank)
  - Obtain node embeddings via **matrix factorization (MF)**
  - View other **node embeddings** (e.g. Node2Vec) as MF
- **Random walk, matrix factorization and node embeddings are closely related!**



# Stanford CS224W: PageRank (aka the Google Algorithm)

CS224W: Machine Learning with Graphs  
Jure Leskovec, Stanford University  
<http://cs224w.stanford.edu>



# Example: The Web as a Graph

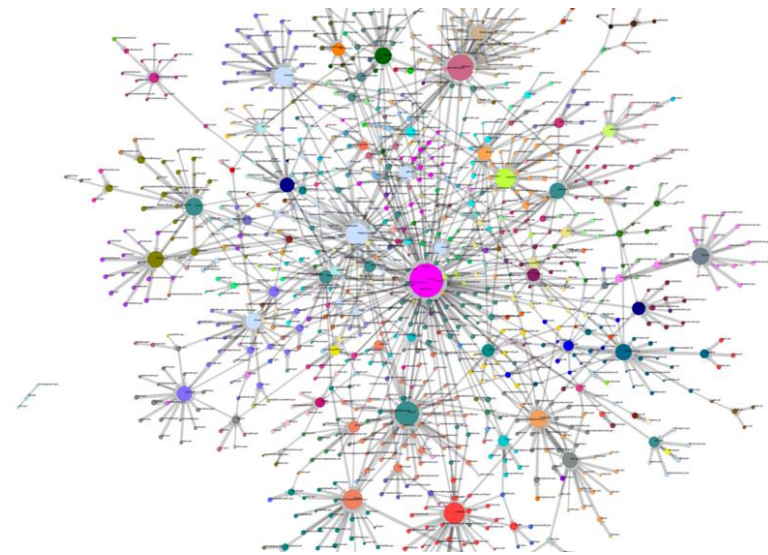
Q: What does the Web “look like” at a global level?

- **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

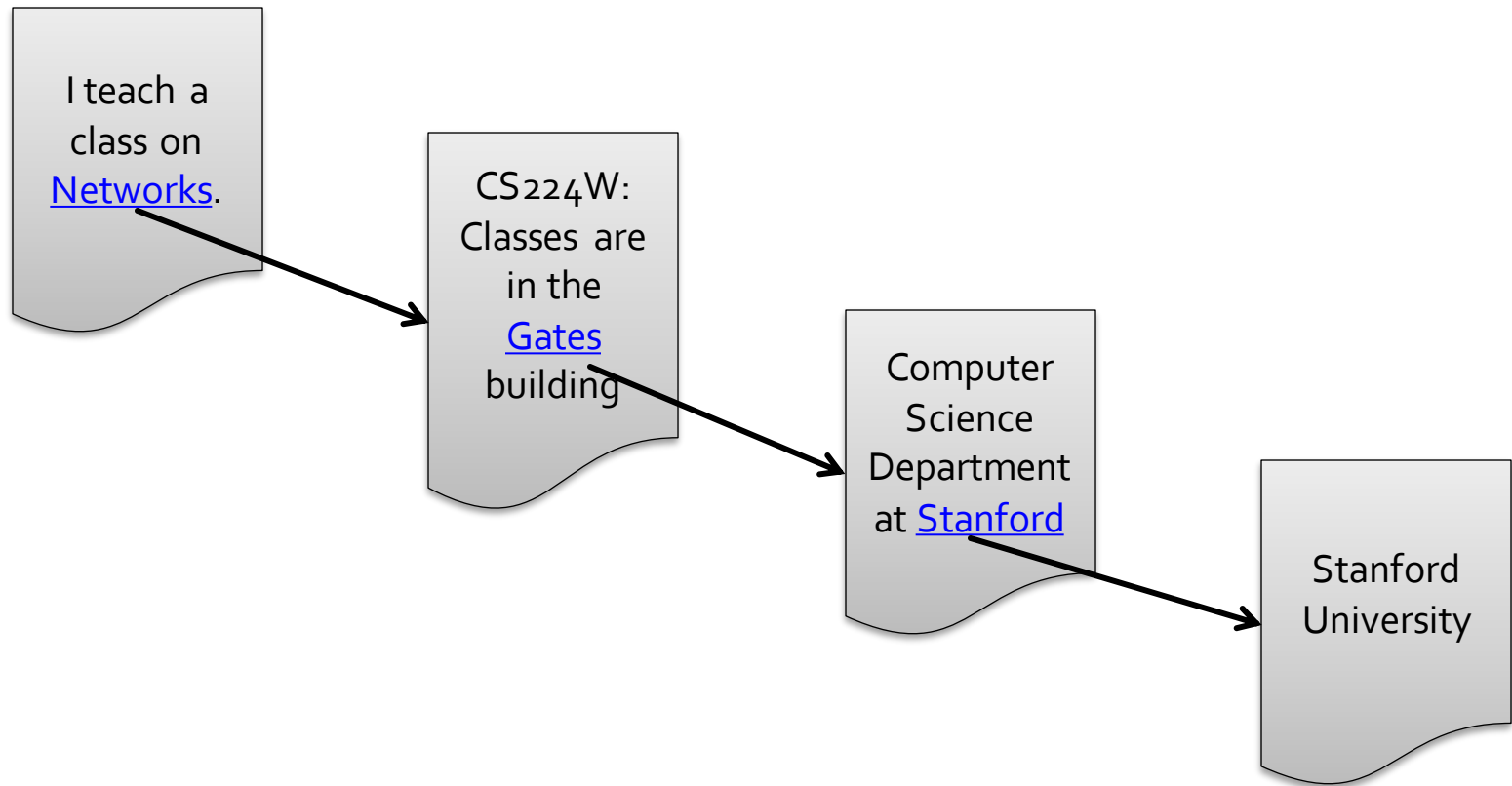
I teach a  
class on  
Networks.

CS224W:  
Classes are  
in the  
Gates  
building

Computer  
Science  
Department  
at Stanford

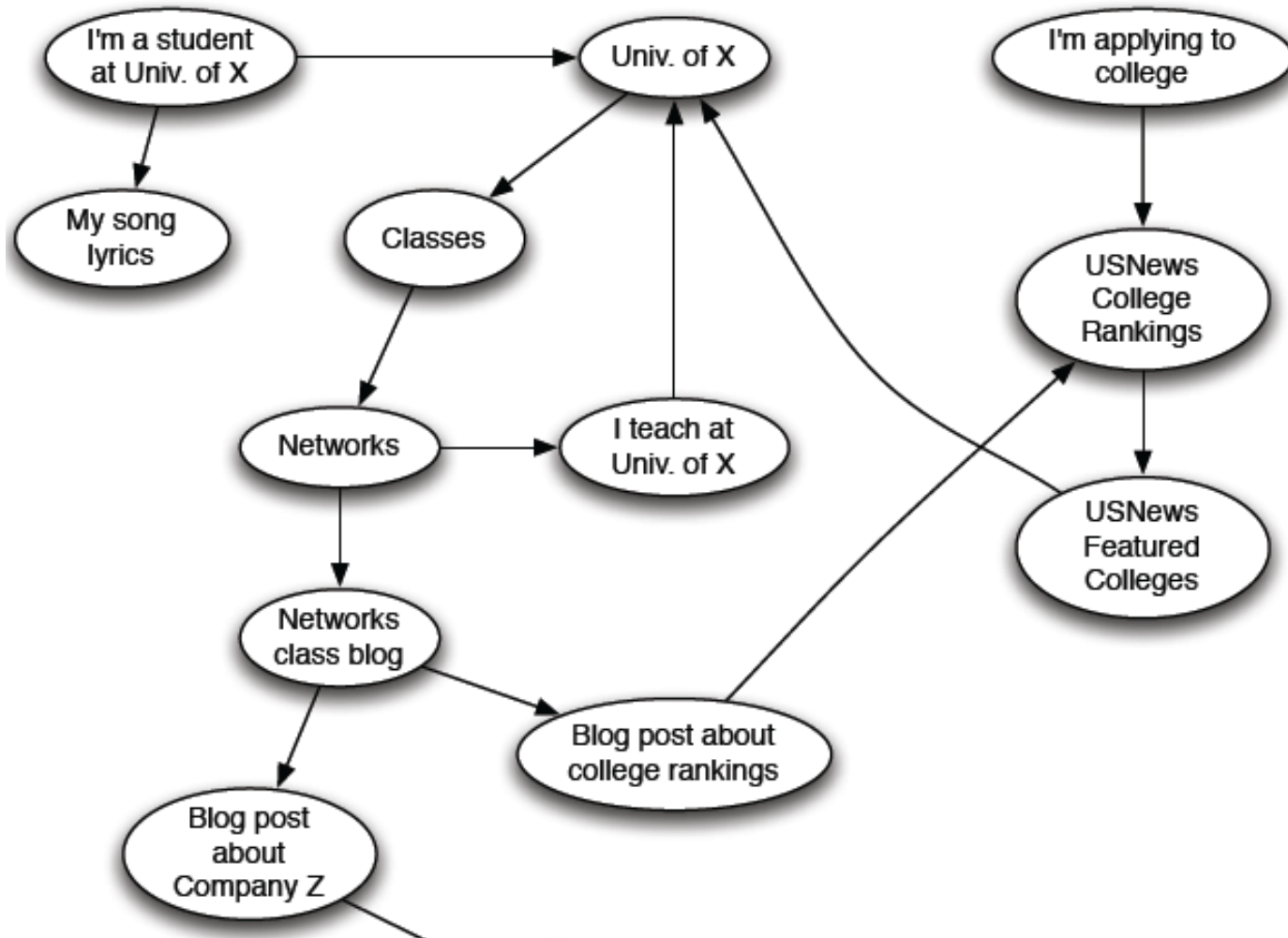
Stanford  
University

# The Web as a Graph



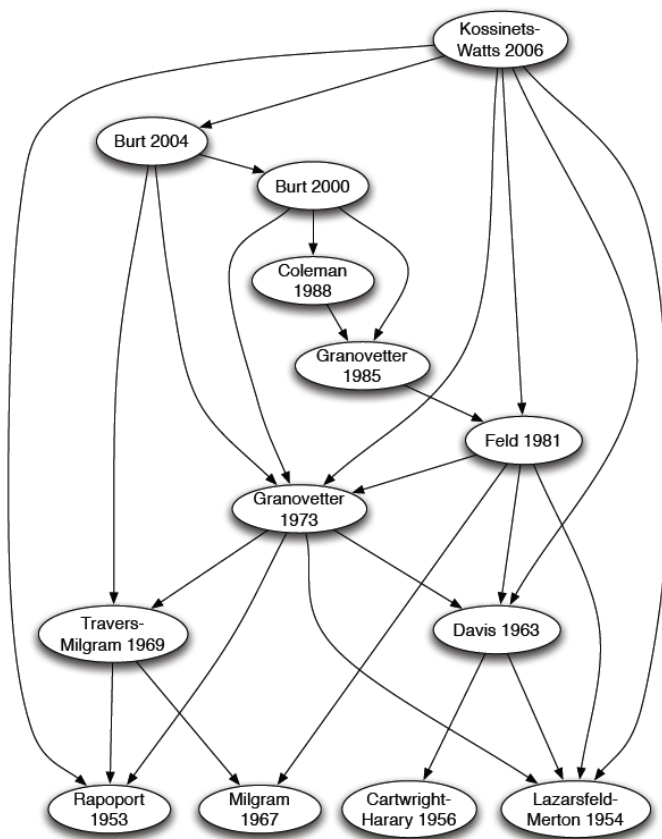
- In early days of the Web links were **navigational**
- Today many links are **transactional** (used not to navigate from page to page, but to post, comment, like, buy, ...)

# The Web as a Directed Graph

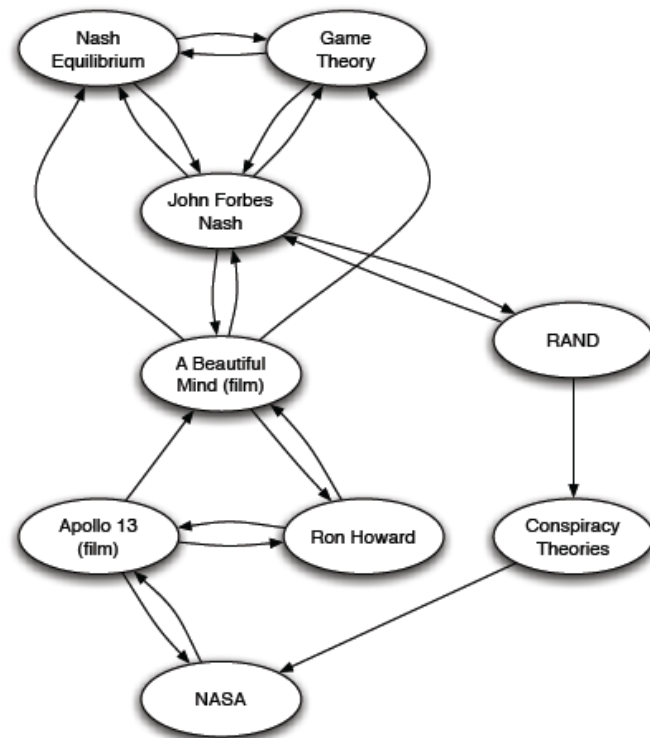




# Other Information Networks



Citations



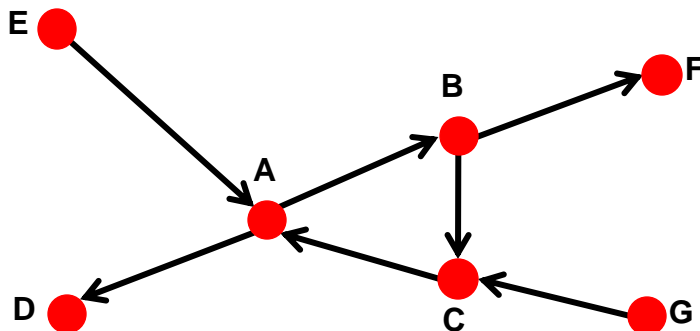
References in an Encyclopedia

# What Does the Web Look Like?

- How is the Web linked?
- What is the “map” of the Web?

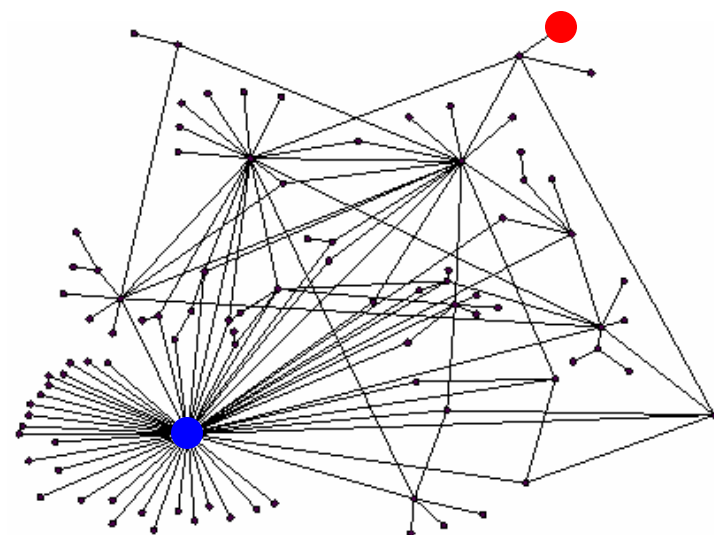
**Web as a directed graph** [Broder et al. 2000]:

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



# Ranking Nodes on the Graph

- All web pages are not equally “important”  
[thispersondoesnotexist.com](http://thispersondoesnotexist.com) vs. [www.stanford.edu](http://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

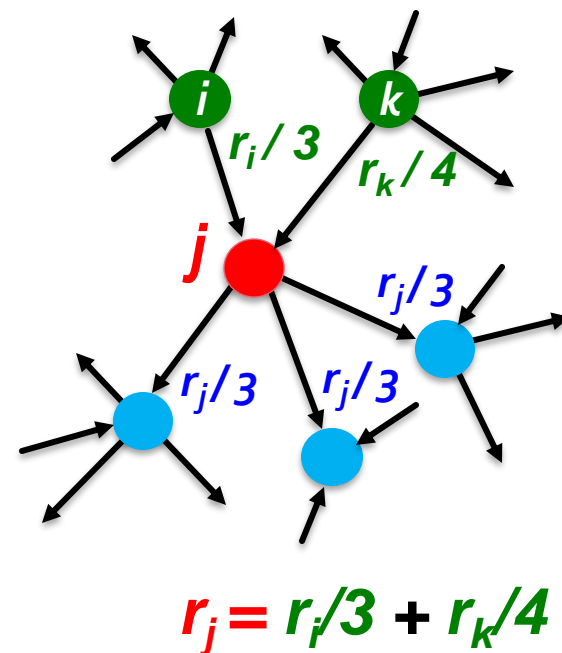
- We will cover the following **Link Analysis approaches** to compute the **importance** of nodes in a graph:
  - PageRank
  - Personalized PageRank (PPR)
  - Random Walk with Restarts

# 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](http://www.stanford.edu) has 23,400 in-links
  - [thispersondoesnotexist.com](http://thispersondoesnotexist.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



# PageRank: The “Flow” Model

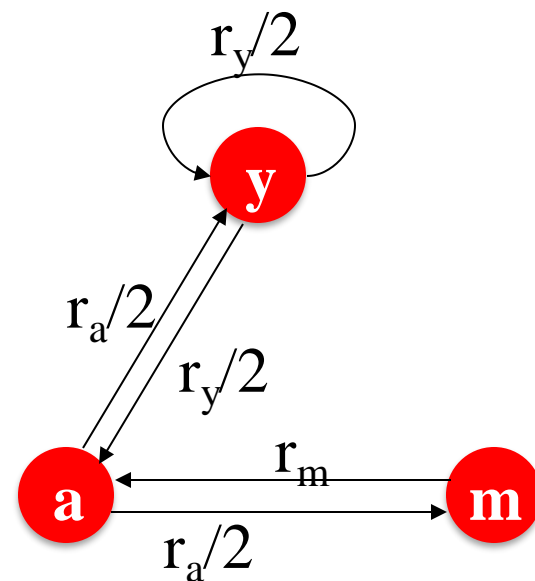
- A page is important if it is pointed to by other important pages
- Define “rank”  $r_j$  for node  $j$

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

$d_i$  ... out-degree of node  $i$

You might wonder: Let’s just use Gaussian elimination to solve this system of linear equations. Bad idea!

The web in 1839



“Flow” equations:

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

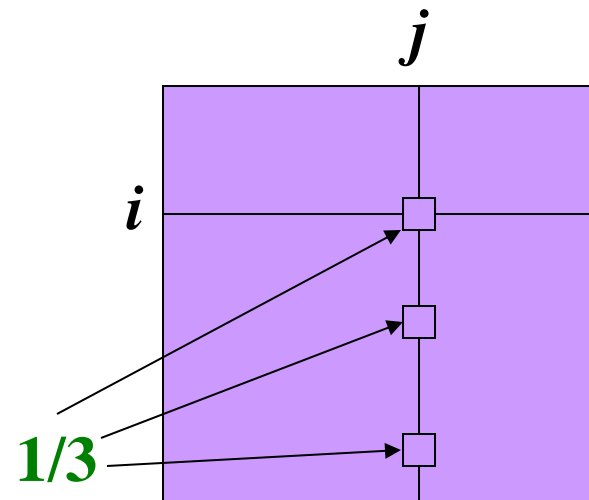
$$r_a = r_y/2 + r_m$$

$$r_m = r_a/2$$

# 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**

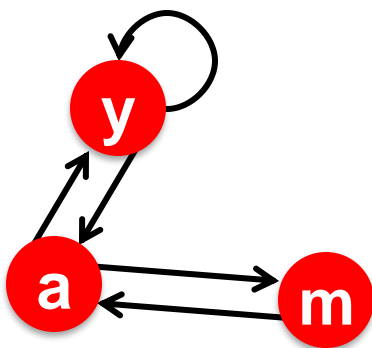
$$r = M \cdot r$$

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

$M$



# Example: Flow Equations & M



	$r_y$	$r_a$	$r_m$
$r_y$	$1/2$	$1/2$	$0$
$r_a$	$1/2$	$0$	$1$
$r_m$	$0$	$1/2$	$0$

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

$$r_a = r_y / 2 + r_m$$

$$r_m = r_a / 2$$

$$\begin{matrix} r_y \\ r_a \\ r_m \end{matrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{matrix} r_y \\ r_a \\ r_m \end{matrix}$$

$\mathbf{r} \qquad \qquad \mathbf{M} \qquad \qquad \mathbf{r}$

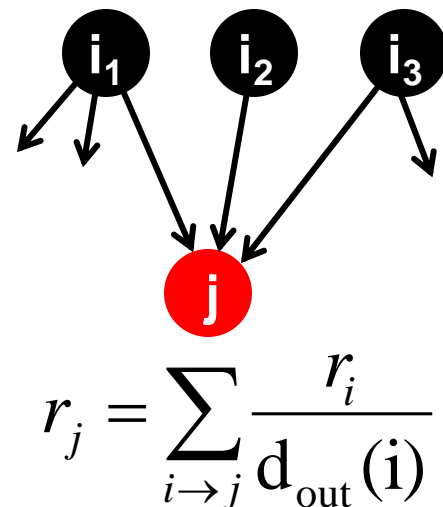
# Connection to Random Walk

- **Imagine a random web surfer:**

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

- **Let:**

- $\mathbf{p}(t)$  ... vector whose  $i^{\text{th}}$  coordinate is the prob. that the surfer is at page  $i$  at time  $t$
- So,  $\mathbf{p}(t)$  is a probability distribution over pages

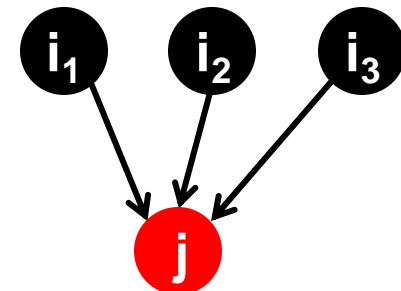


# The Stationary Distribution

- Where is the surfer at time  $t+1$ ?

- Follow a link uniformly at random

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



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

- Suppose the random walk reaches a state

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

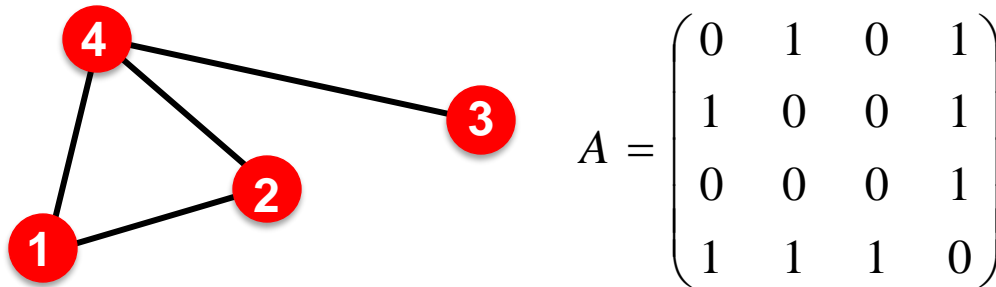
then  $\mathbf{p}(t)$  is **stationary distribution** of a random walk

- Our original rank vector  $\mathbf{r}$  satisfies  $\mathbf{r} = \mathbf{M} \cdot \mathbf{r}$

- So,  $\mathbf{r}$  is a stationary distribution for the random walk

# Recall Eigenvector of A Matrix

- Recall from lecture 2 (eigenvector centrality), let  $A \in \{0, 1\}^{n \times n}$  be an adj. matrix of undir. graph:



- Eigenvector of adjacency matrix:  
vectors satisfying  $\lambda \mathbf{c} = A \mathbf{c}$
- $\mathbf{c}$ : eigenvector;  $\lambda$ : eigenvalue
- Note:
  - This is the definition of eigenvector centrality (for undirected graphs).
  - PageRank is defined for directed graphs

# Eigenvector Formulation

- The flow equation:

$$1 \cdot \mathbf{r} = \mathbf{M} \cdot \mathbf{r}$$

$$\begin{array}{|c|} \hline r_y \\ \hline r_a \\ \hline r_m \\ \hline \end{array} = \begin{array}{|ccc|} \hline 1/2 & 1/2 & 0 \\ \hline 1/2 & 0 & 1 \\ \hline 0 & 1/2 & 0 \\ \hline \end{array} \begin{array}{|c|} \hline r_y \\ \hline r_a \\ \hline r_m \\ \hline \end{array}$$

$\mathbf{r} \qquad \mathbf{M} \qquad \mathbf{r}$

- So the rank vector  $\mathbf{r}$  is an **eigenvector** of the stochastic adj. matrix  $\mathbf{M}$  (with eigenvalue 1)
  - Starting from any vector  $\mathbf{u}$ , the limit  $\mathbf{M}(\mathbf{M}(\dots \mathbf{M}(\mathbf{M} \mathbf{u})))$  is the **long-term distribution** of the surfers.
    - **PageRank** = Limiting distribution = **principal eigenvector** of  $\mathbf{M}$
    - **Note**: If  $\mathbf{r}$  is the limit of the product  $\mathbf{M}\mathbf{M} \dots \mathbf{M}\mathbf{u}$ , then  $\mathbf{r}$  satisfies the **flow equation**  $1 \cdot \mathbf{r} = \mathbf{M}\mathbf{r}$
    - So  $\mathbf{r}$  is the **principal eigenvector** of  $\mathbf{M}$  with eigenvalue 1
- We can now efficiently solve for  $\mathbf{r}$ !
  - The method is called **Power iteration**

# PageRank: Summary

- **PageRank:**
  - Measures importance of nodes in a graph using the link structure of the web
  - Models a random web surfer using the **stochastic adjacency matrix  $M$**
  - PageRank solves  $\mathbf{r} = M\mathbf{r}$  where  $\mathbf{r}$  can be viewed as both the **principle eigenvector of  $M$**  and as **the stationary distribution of a random walk** over the graph

# Stanford CS224W: PageRank: How to solve?

CS224W: Machine Learning with Graphs  
Jure Leskovec, Stanford University  
<http://cs224w.stanford.edu>



# PageRank: How to solve?

Given a graph with  $n$  nodes, we use an iterative procedure:

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

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

$d_i$  .... out-degree of node  $i$



# 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

- Initialize:  $\mathbf{r}^{(0)} = [1/N, \dots, 1/N]^T$

- Iterate:  $\mathbf{r}^{(t+1)} = \mathbf{M} \cdot \mathbf{r}^{(t)}$

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

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

$d_i$  .... out-degree of node  $i$

$\|\mathbf{x}\|_1 = \sum_1^N |x_i|$  is the **L1** norm

Can use any other vector norm, e.g., Euclidean

About 50 iterations is sufficient to estimate the limiting solution.

# PageRank: How to solve?

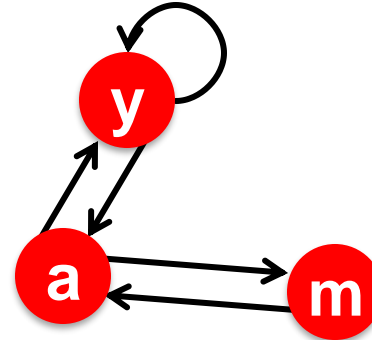
## ■ Power Iteration:

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

## ■ Example:

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

Iteration 0, 1, 2, ...



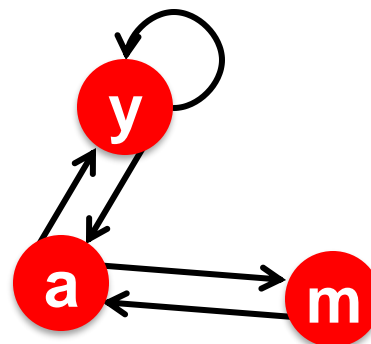
	y	a	m
y	1/2	1/2	0
a	1/2	0	1
m	0	1/2	0

$$\begin{aligned} r_y &= r_y/2 + r_a/2 \\ r_a &= r_y/2 + r_m \\ r_m &= r_a/2 \end{aligned}$$

# PageRank: How to solve?

## ■ Power Iteration:

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



	y	a	m
y	1/2	1/2	0
a	1/2	0	1
m	0	1/2	0

$$\begin{aligned}
 r_y &= r_y/2 + r_a/2 \\
 r_a &= r_y/2 + r_m \\
 r_m &= r_a/2
 \end{aligned}$$

## ■ Example:

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

Iteration 0, 1, 2, ...

# PageRank: Three Questions

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

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

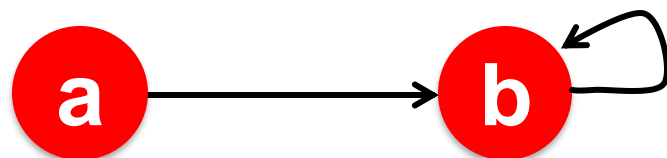
# PageRank: Problems

## 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?

- The “Spider trap” problem:



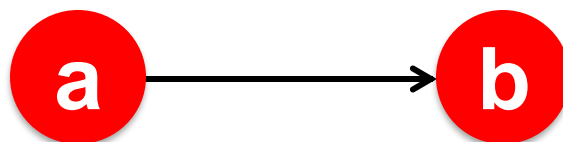
$$r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i}$$

- Example:

Iteration:		0,	1,	2,	3...
$r_a$	=	1	0	0	0
$r_b$		0	1	1	1

# Does it converge to what we want?

- The “Dead end” problem:



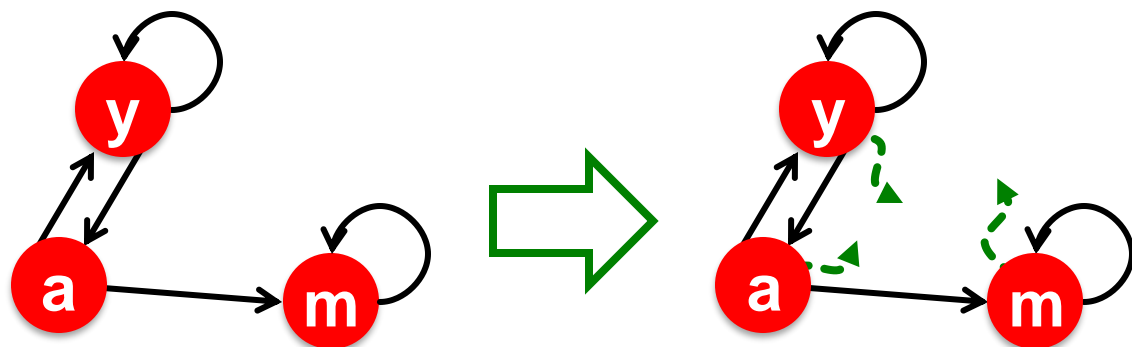
$$r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i}$$

- Example:

Iteration:		0,	1,	2,	3...
$r_a$	=	1	0	0	0
$r_b$		0	1	0	0

# Solution to Spider Traps

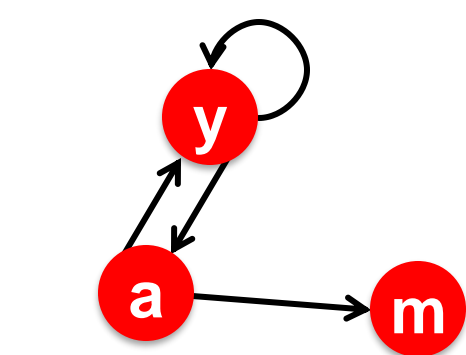
- **Solution for spider traps:** At each time step, the random surfer has two options
  - With prob.  $\beta$ , follow a link at random
  - With prob.  $1-\beta$ , jump to a random page
  - Common values for  $\beta$  are in the range 0.8 to 0.9
- **Surfer will teleport out of spider trap within a few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



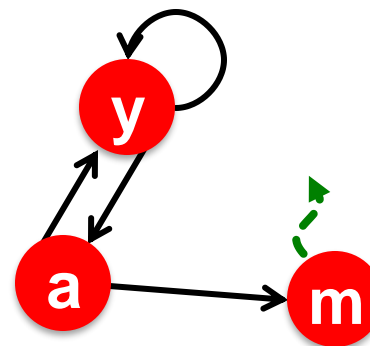
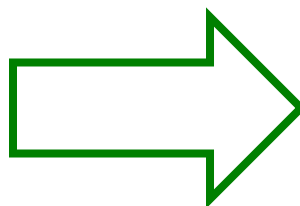


# Solution to Dead Ends

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



	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	0
a	$\frac{1}{2}$	0	0
m	0	$\frac{1}{2}$	0



	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$
a	$\frac{1}{2}$	0	$\frac{1}{3}$
m	0	$\frac{1}{2}$	$\frac{1}{3}$

# 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  $\beta$ , follow a link at random
- With probability  $1-\beta$ , jump to some random page

- **PageRank equation** [Brin-Page, 98]

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

$d_i$  ... out-degree of node  $i$

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

# The Google Matrix

- **PageRank equation** [Brin-Page, '98]

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

- **The Google Matrix  $G$ :**

$[1/N]_{N \times N} \dots N$  by  $N$  matrix  
where all entries are  $1/N$

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

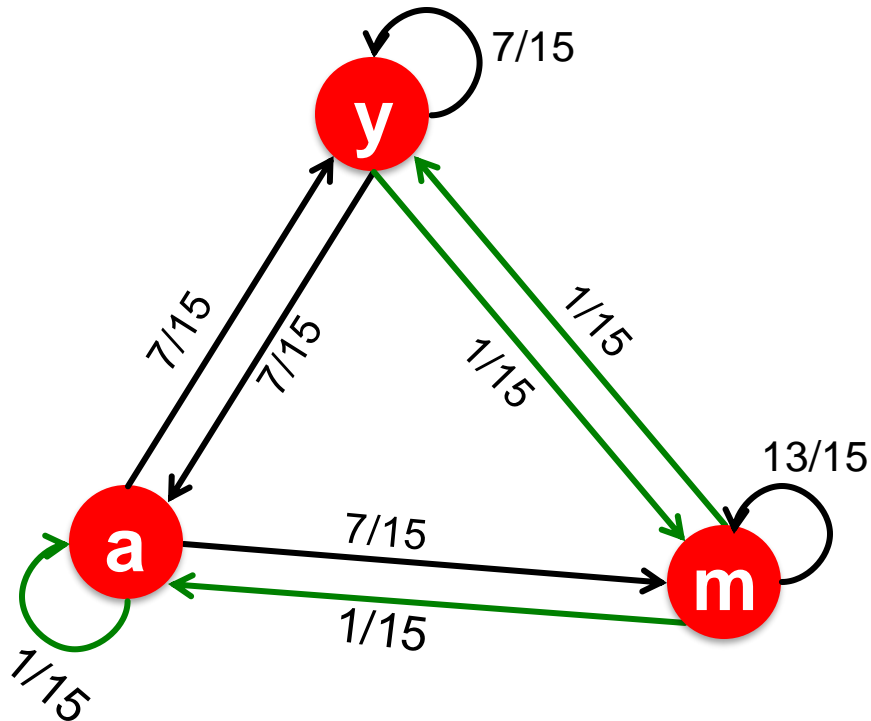
- **We have a recursive problem:  $\mathbf{r} = G \cdot \mathbf{r}$**

**And the Power method still works!**

- **What is  $\beta$ ?**

- In practice  $\beta = 0.8, 0.9$  (make 5 steps on avg., jump)

# Random Teleports ( $\beta = 0.8$ )



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

$$\mathbf{G} = \begin{matrix} & \begin{matrix} y & a & m \end{matrix} \\ \begin{matrix} y \\ a \\ m \end{matrix} & \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix} \end{matrix}$$

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

# PageRank Example

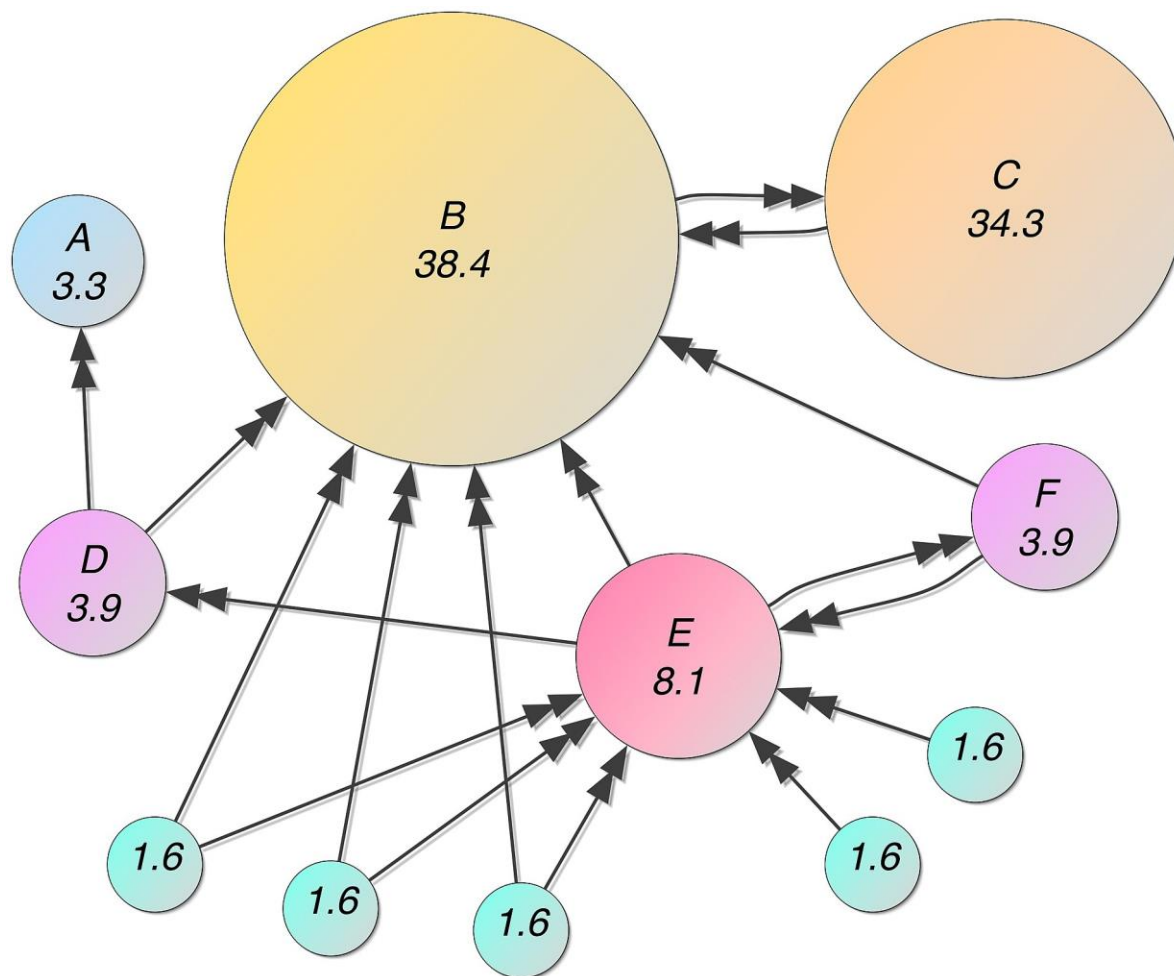


Image credit: [Wikipedia](https://en.wikipedia.org/wiki/File:PageRank_algorithm_diagram.svg)

# Solving PageRank: Summary

- PageRank solves for  $\mathbf{r} = \mathbf{G}\mathbf{r}$  and can be efficiently computed by power iteration of the stochastic adjacency matrix ( $\mathbf{G}$ )
- Adding random uniform teleportation solves issues of dead-ends and spider-traps

# Stanford CS224W: Random Walk with Restarts and Personalized PageRank

CS224W: Machine Learning with Graphs  
Jure Leskovec, Stanford University  
<http://cs224w.stanford.edu>

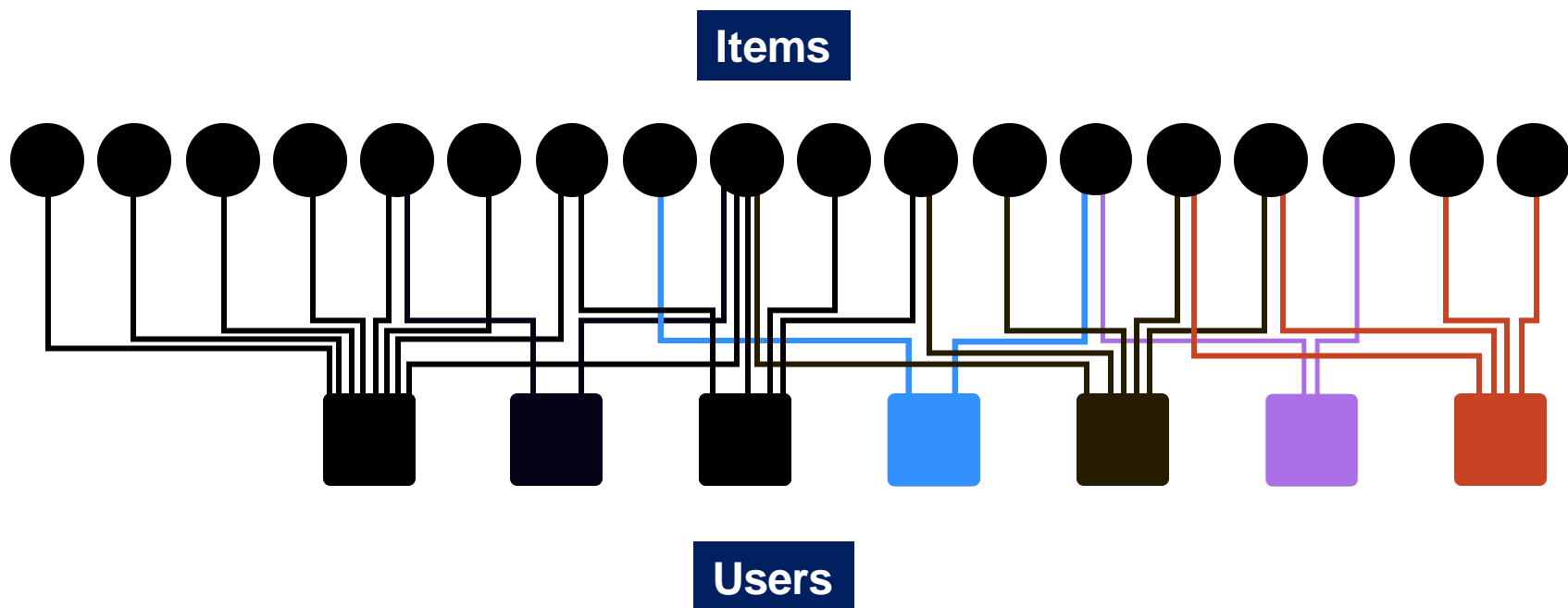




# Example: Recommendation

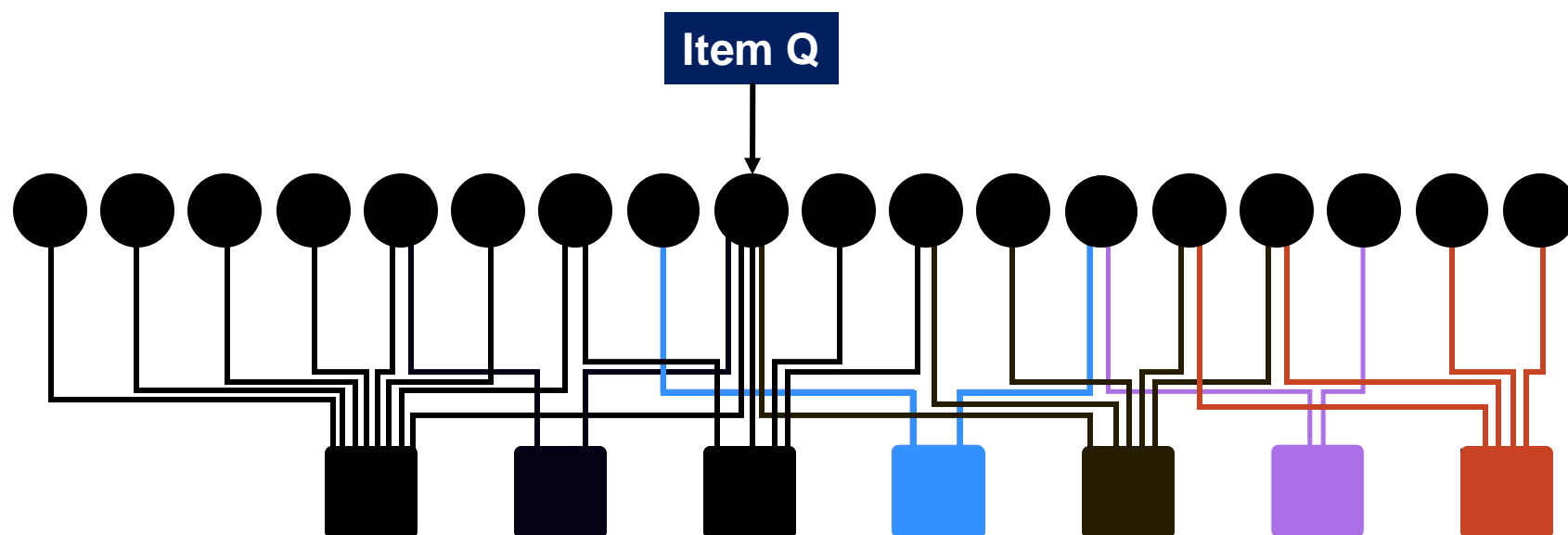
- **Given:**

A bipartite graph representing user and item interactions (e.g. purchase)



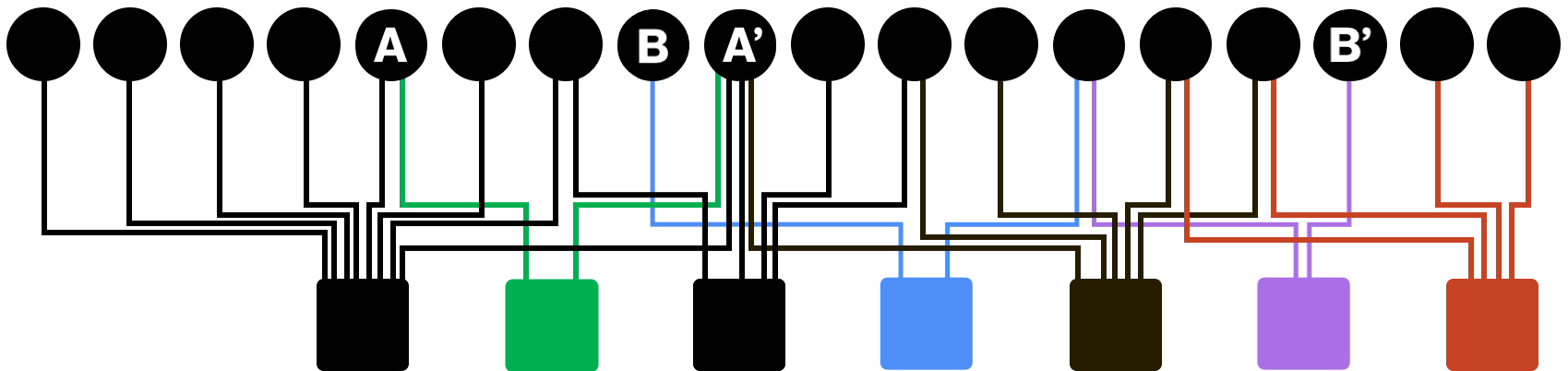
# Bipartite User-Item Graph

- **Goal:** Proximity on graphs
  - What items should we recommend to a user who interacts with item Q?
  - **Intuition:** if items Q and P are interacted by similar users, recommend P when user interacts with Q



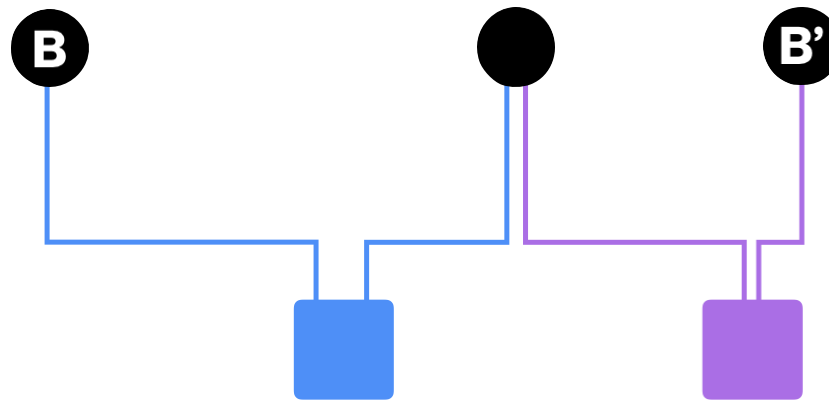
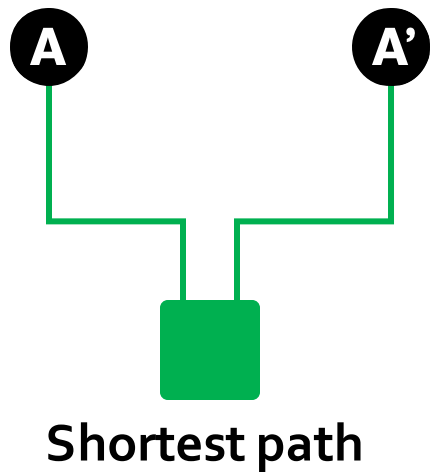
# Bipartite User-to-Item Graph

- Which is more related A,A' or B,B'?



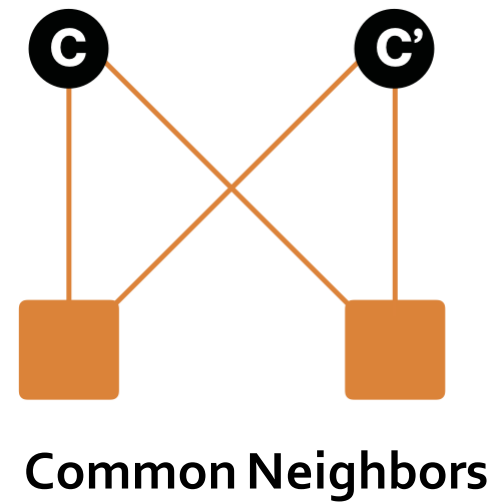
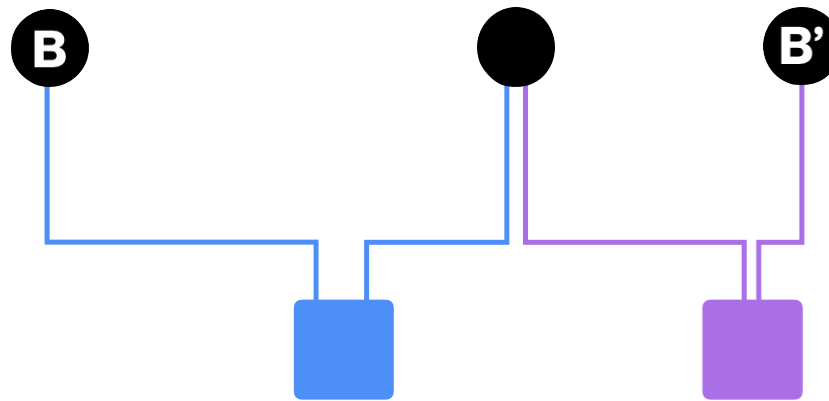
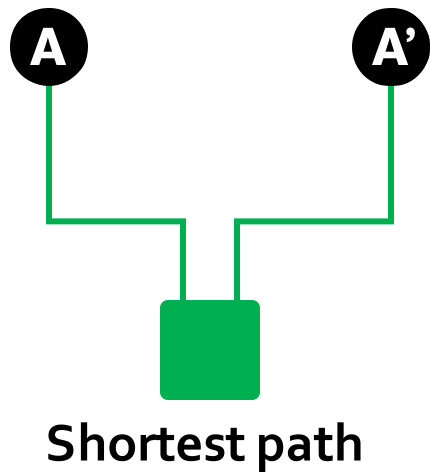
# Node proximity Measurements

- Which is more related A,A', B,B' or C,C'?



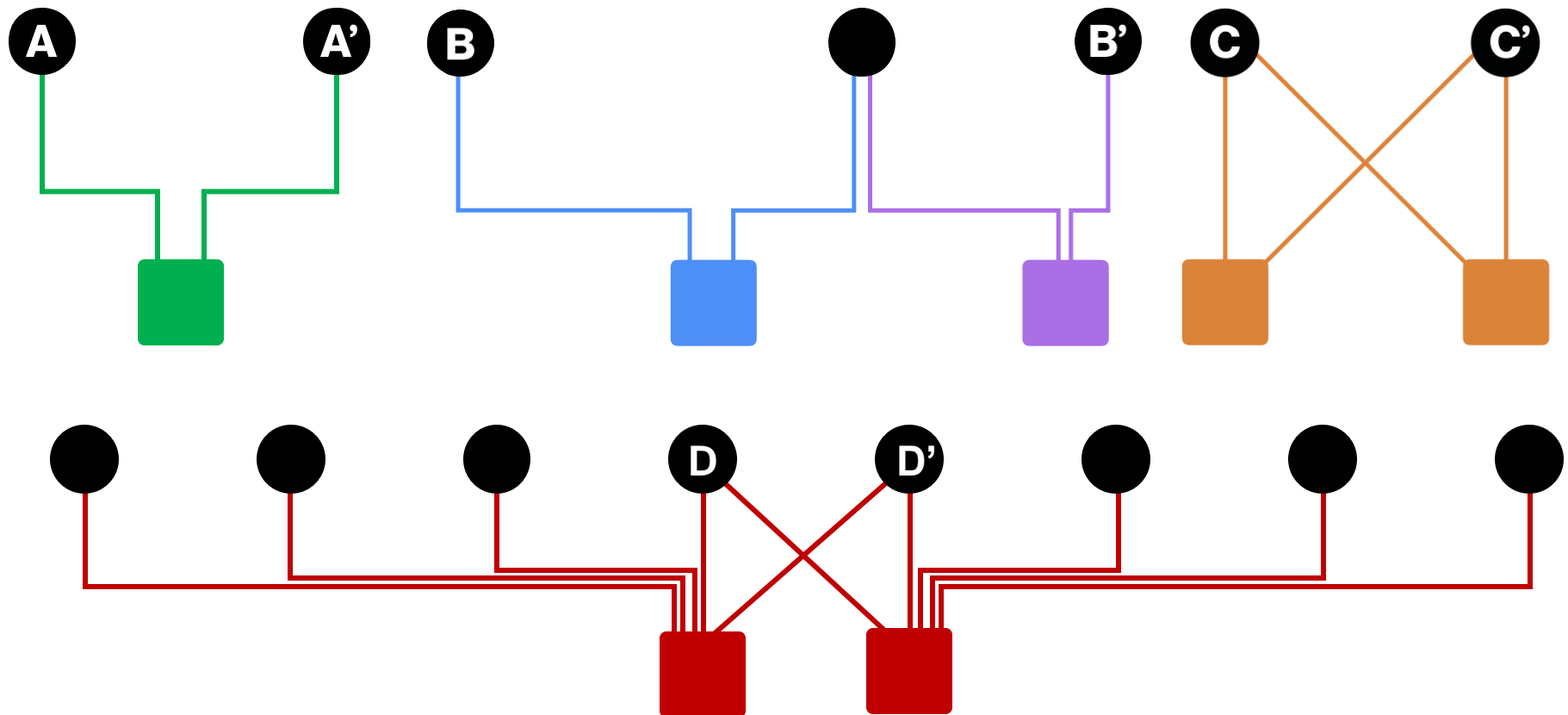
# Node proximity Measurements

- Which is more related A,A', B,B' or C,C'?



# Node proximity Measurements

- Which is more related A,A', B,B' or C,C'?



Personalized Page Rank/Random Walk with Restarts

# Proximity on Graphs

- **PageRank:**
  - Ranks nodes by “importance”
  - Teleports with uniform probability to any node in the network
- **Personalized PageRank:**
  - Ranks proximity of nodes to the teleport nodes  $S$
- **Proximity on graphs:**
  - **Q:** What is most related item to **Item Q**?
  - **Random Walks with Restarts**
    - Teleport back to the starting node:  $S = \{Q\}$

# Idea: Random Walks

## ■ Idea

- Every node has some importance
- Importance gets evenly split among all edges and pushed to the neighbors:

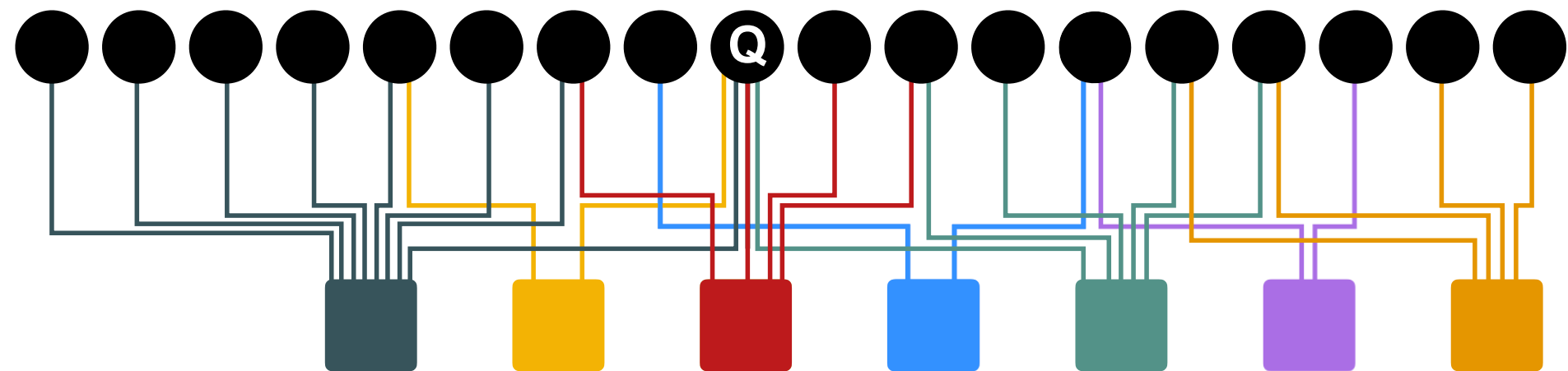
## ■ Given a set of QUERY\_NODES, we simulate a random walk:

- Make a step to a random neighbor and record the visit (visit count)
- With probability ALPHA, restart the walk at one of the QUERY\_NODES
- The nodes with the highest visit count have highest proximity to the QUERY\_NODES



# Random Walks

- **Idea:**
  - Every node has some importance
  - Importance gets evenly split among all edges and pushed to the neighbors
- Given a set of **QUERY NODES Q**, simulate a random walk:



# Random Walk Algorithm

## ■ Proximity to query node(s) $Q$ :

$\text{ALPHA} = 0.5$

$\text{QUERY\_NODES} = \{ \text{Q} \}$

```
item = QUERY_NODES.sample_by_weight()
```

```
for i in range( N_STEPS ):
```

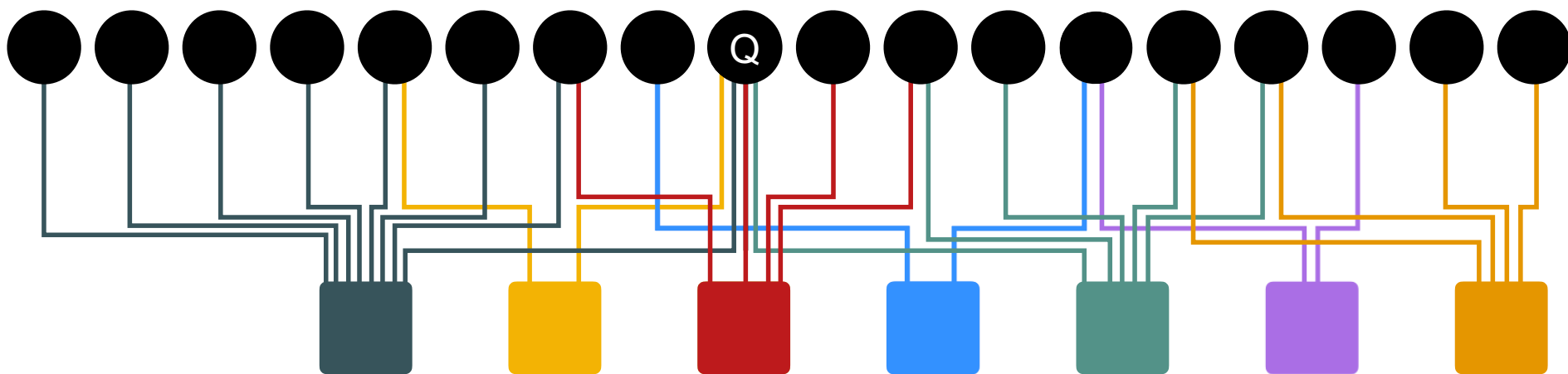
```
    user = item.get_random_neighbor()
```

```
    item = user.get_random_neighbor()
```

```
    item.visit_count += 1
```

```
    if random( ) < ALPHA:
```

```
        item = QUERY_NODES.sample.by_weight( )
```



# Random Walk Algorithm

## ■ Proximity to query node(s) $Q$ :

ALPHA = 0.5

QUERY\_NODES =



```
item = QUERY_NODES.sample_by_weight()  
for i in range( N_STEPS ):
```

```
    user = item.get_random_neighbor()
```

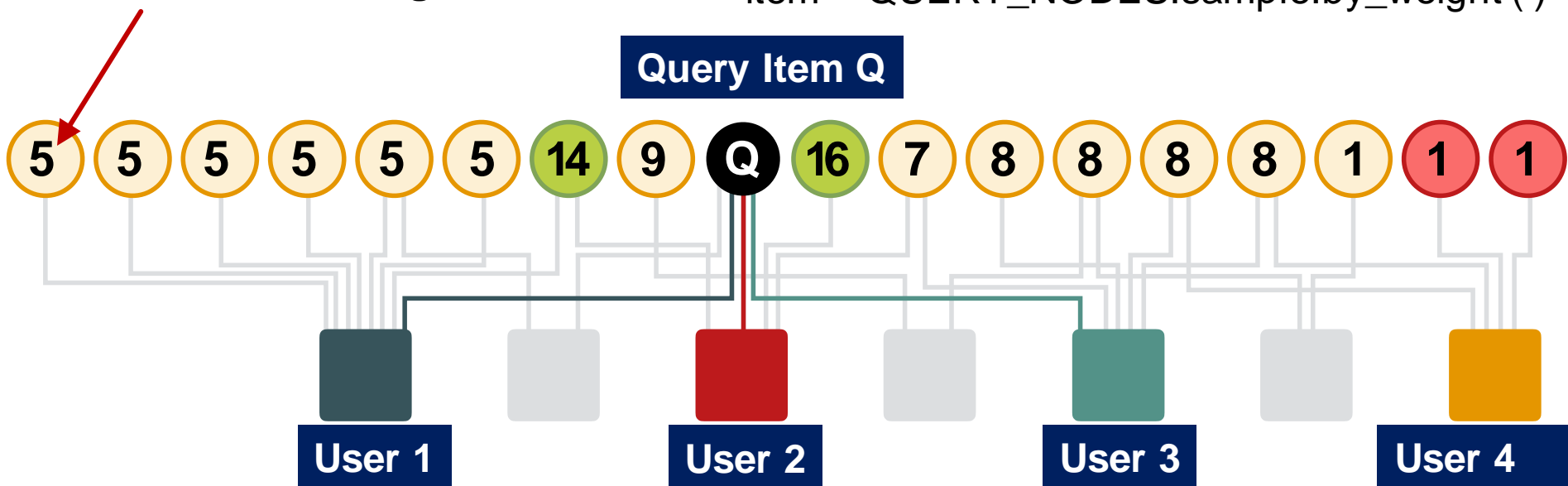
```
    item = user.get_random_neighbor()
```

```
    item.visit_count += 1
```

```
    if random( ) < ALPHA:
```

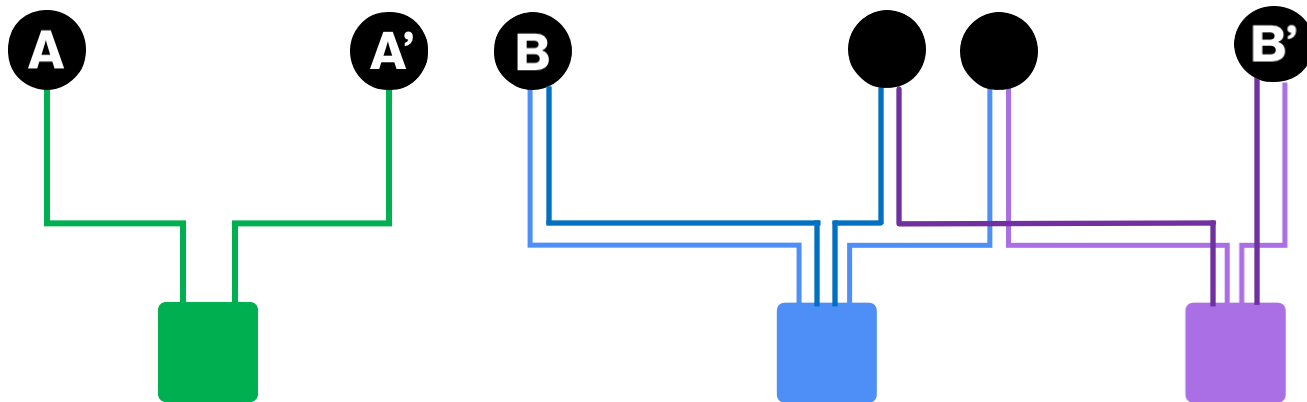
```
        item = QUERY_NODES.sample.by_weight ( )
```

Number of visits by  
random walks starting at  $Q$



# Benefits

- Why is this a good solution?
- Because the “similarity” considers:
  - Multiple connections
  - Multiple paths
  - Direct and indirect connections
  - Degree of the node



# Summary: Page Rank Variants

## ■ PageRank:

- Teleports to any node
- Nodes can have the same probability of the surfer landing:  
 $S = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]$

## ■ Topic-Specific PageRank aka Personalized PageRank:

- Teleports to a specific set of nodes
- Nodes can have different probabilities of the surfer landing there:

$$S = [0.1, 0, 0, 0.2, 0, 0, 0.5, 0, 0, 0.2]$$

## ■ Random Walk with Restarts:

- Topic-Specific PageRank where teleport is always to the same node:

$$S = [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]$$

# Summary

- A graph is naturally represented as a matrix
- We defined a random walk process over the graph
  - Random surfer moving across the links and with random teleportation
  - Stochastic adjacency matrix  $M$
- PageRank = Limiting distribution of the surfer location represented node importance
  - Corresponds to the leading eigenvector of transformed adjacency matrix  $M$ .

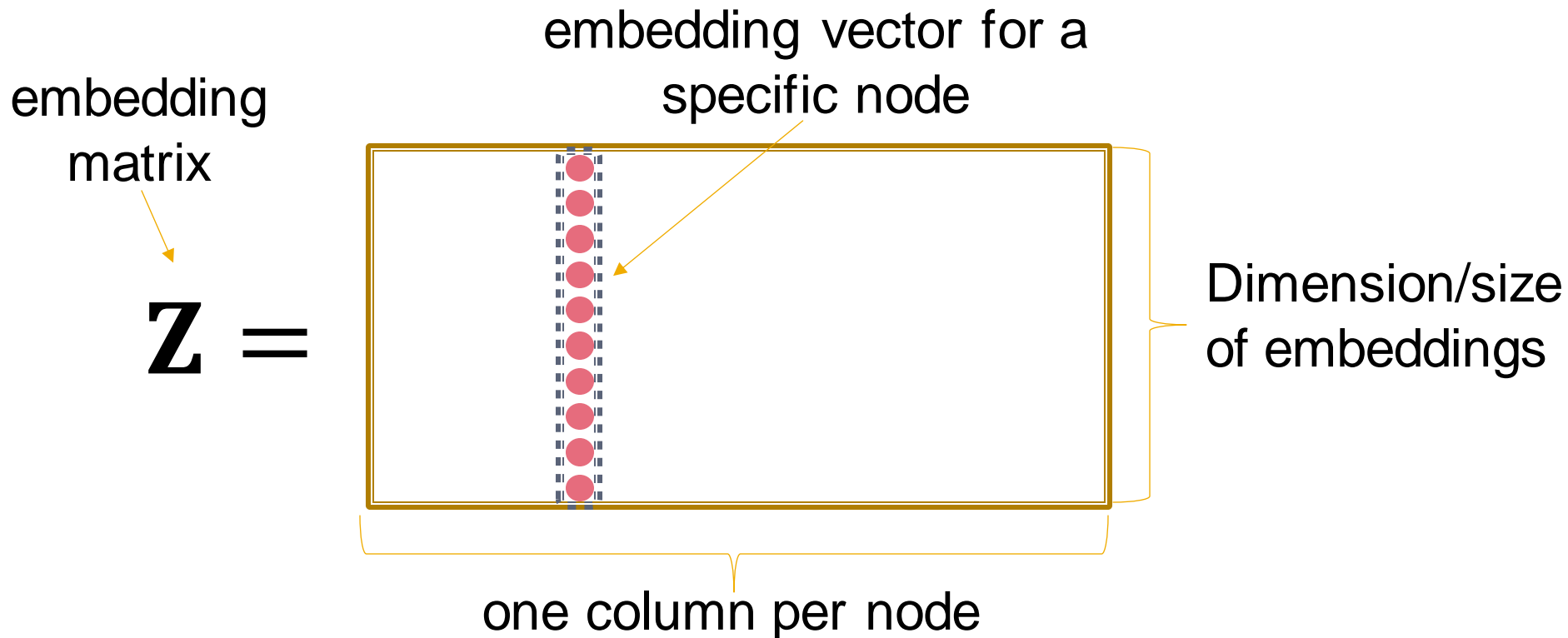
# Stanford CS224W: Matrix Factorization and Node Embeddings

CS224W: Machine Learning with Graphs  
Jure Leskovec, Stanford University  
<http://cs224w.stanford.edu>



# Embeddings & Matrix Factorization

- **Recall:** encoder as an embedding lookup

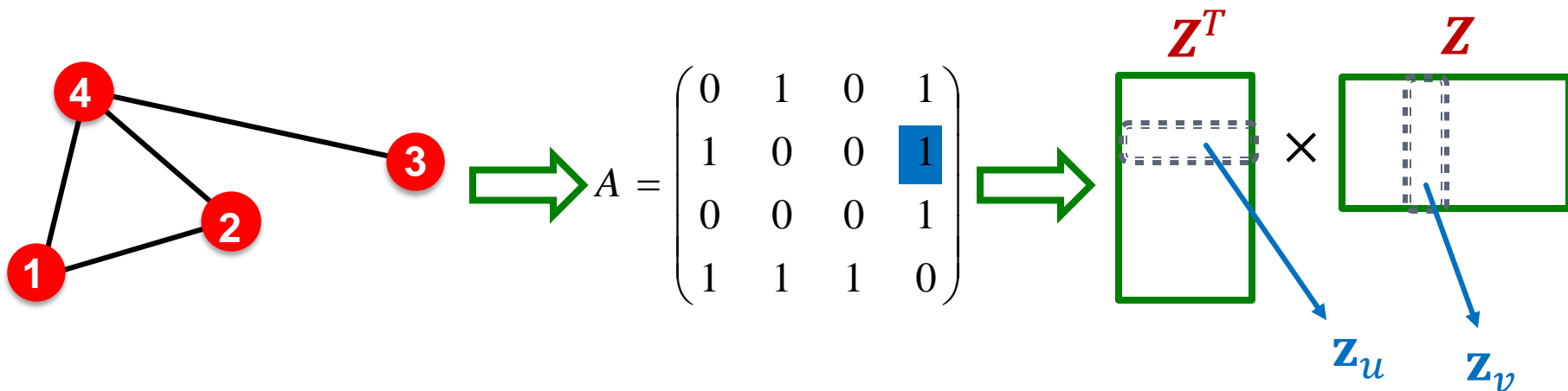


**Objective:** maximize  $\mathbf{z}_v^T \mathbf{z}_u$  for node pairs  $(u, v)$  that are **similar**



# Connection to Matrix Factorization

- Simplest **node similarity**: Nodes  $u, v$  are similar if they are connected by an edge
- This means:  $\mathbf{z}_v^T \mathbf{z}_u = A_{u,v}$   
which is the  $(u, v)$  entry of the graph adjacency matrix  $A$
- Therefore,  **$\mathbf{Z}^T \mathbf{Z} = A$**



# Matrix Factorization

- The embedding dimension  $d$  (number of rows in  $\mathbf{Z}$ ) is much smaller than number of nodes  $n$ .
- Exact factorization  $\mathbf{A} = \mathbf{Z}^T \mathbf{Z}$  is generally not possible
- However, we can learn  $\mathbf{Z}$  approximately
- **Objective:**  $\min_{\mathbf{Z}} \|\mathbf{A} - \mathbf{Z}^T \mathbf{Z}\|_2$ 
  - We optimize  $\mathbf{Z}$  such that it minimizes the L2 norm (Frobenius norm) of  $\mathbf{A} - \mathbf{Z}^T \mathbf{Z}$
  - Note in Lecture 3 we used softmax instead of L2. But the goal to approximate  $\mathbf{A}$  with  $\mathbf{Z}^T \mathbf{Z}$  is the same.
- Conclusion: **Inner product decoder with node similarity defined by edge connectivity is equivalent to matrix factorization of  $\mathbf{A}$ .**

# Random Walk-based Similarity

- **DeepWalk** and **node2vec** have a more complex **node similarity** definition based on random walks
- **DeepWalk** is equivalent to matrix factorization of the following complex matrix expression:

$$\log \left( \text{vol}(G) \left( \frac{1}{T} \sum_{r=1}^T (D^{-1}A)^r \right) D^{-1} \right) - \log b$$

- Explanation of this equation is on the next slide.

[Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec](#), WSDM 18

# Random Walk-based Similarity

**Volume of graph**

$$vol(G) = \sum_i \sum_j A_{i,j}$$

**Diagonal matrix  $D$**

$$D_{u,u} = \deg(u)$$

$$\log \left( vol(G) \left( \frac{1}{T} \sum_{r=1}^T (D^{-1}A)^r \right) D^{-1} \right) - \log b$$

**context window size**

See Lec 3 slide 30:

$$T = |N_R(u)|$$

**Power of normalized  
adjacency matrix**

**Number of  
negative samples**

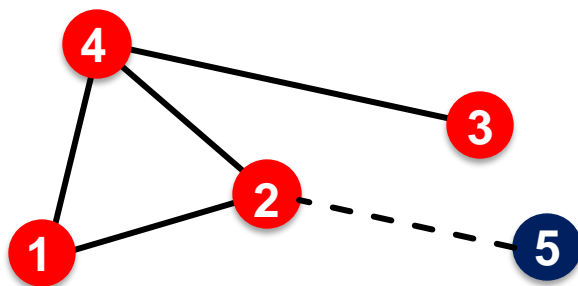
- **Node2vec** can also be formulated as a matrix factorization (albeit a more complex matrix)
- Refer to the paper for more details:

Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec, WSDM 18

# Limitations (1)

## Limitations of node embeddings via matrix factorization and random walks

- Cannot obtain embeddings for nodes not in the training set



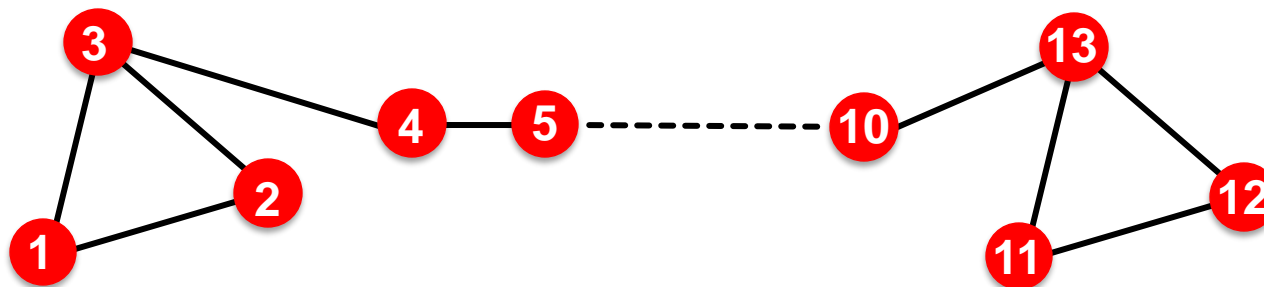
Training set

A newly added node 5 at test time  
(e.g., new user in a social network)

**Cannot compute its embedding  
with DeepWalk / node2vec. Need to  
recompute all node embeddings.**

# Limitation (2)

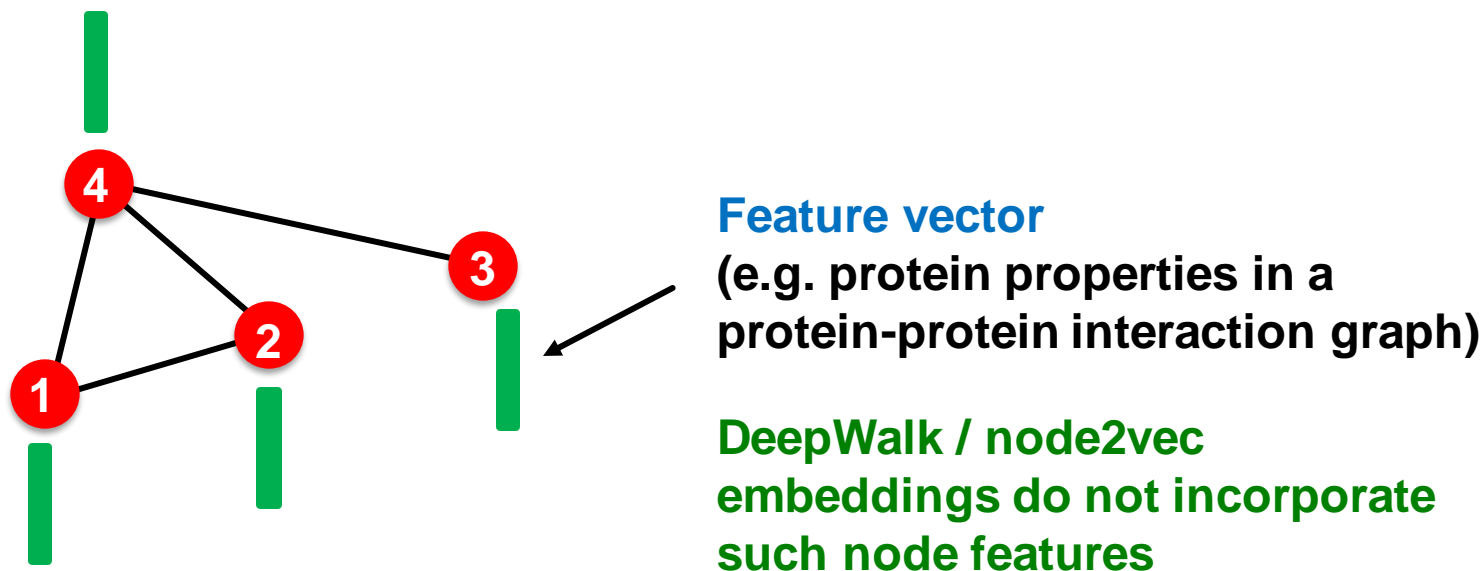
- Cannot capture **structural similarity**:



- Node 1 and 11 are **structurally similar** – part of one triangle, degree 2, ...
- However, they have very **different** embeddings.
  - It's unlikely that a random walk will reach node 11 from node 1.
- **DeepWalk and node2vec do not capture structural similarity.**

# Limitations (3)

- Cannot utilize node, edge and graph features



**Solution to these limitations: Deep Representation Learning and Graph Neural Networks**  
(To be covered in depth next week)

# Summary

- **PageRank**
  - Measures importance of nodes in graph
  - Can be efficiently computed by **power iteration of adjacency matrix**
- **Personalized PageRank (PPR)**
  - Measures importance of nodes with respect to a particular node or set of nodes
  - Can be efficiently computed by **random walk**
- **Node embeddings** based on random walks can be expressed as **matrix factorization**
- **Viewing graphs as matrices plays a key role in all above algorithms!**