

Online Social Networks and Media

Link Analysis and Web Search

How to Organize the Web

First try: Human curated Web directories

Yahoo, DMOZ, LookSmart

YAHOO! DIRECTORY Yahoo! | Help

Search: the Web | the Directory

Yahoo! Directory Advanced Search | Suggest a Site | Email This Page

Arts & Humanities Photography, History, Literature...	News & Media Newspapers, Radio, Weather, Blogs...
Business & Economy B2B, Finance, Shopping, Jobs...	Recreation & Sports Sports, Travel, Autos, Outdoors...
Computer & Internet Hardware, Software, Web, Games...	Reference Phone Numbers, Dictionaries, Quotes...
Education Colleges, K-12, Distance Learning...	Regional Countries, Regions, U.S. States...
Entertainment Movies, TV Shows, Music, Humor...	Science Animals, Astronomy, Earth Science...
Government Elections, Military, Law, Taxes...	Social Science Languages, Archaeology, Psychology...
Health Disease, Drugs, Fitness, Nutrition...	Society & Culture Sexuality, Religion, Food & Drink...
New Additions 12/3, 12/2, 12/1, 11/30, 11/29...	Subscribe via RSS Arts, Music, Sports, TV, more...

Copyright © 2012 Yahoo! Inc. All rights reserved. [Privacy Policy](#) - [About Our Ads](#) - [Terms of Service](#) - [Copyright/IP Policy](#)

Help us improve the Yahoo! Directory - [Share your ideas](#)

about dmoz | dmoz blog | suggest URL | help | link | editor login

[advanced](#)

Arts Movies , Television , Music ...	Business Jobs , Real Estate , Investing ...	Computers Internet , Software , Hardware ...
Games Video Games , RPGs , Gambling ...	Health Fitness , Medicine , Alternative ...	Home Family , Consumers , Cooking ...
Kids and Teens Arts , School Time , Teen Life ...	News Media , Newspapers , Weather ...	Recreation Travel , Food , Outdoors , Humor ...
Reference Maps , Education , Libraries ...	Regional US , Canada , UK , Europe ...	Science Biology , Psychology , Physics ...
Shopping Clothing , Food , Gifts ...	Society People , Religion , Issues ...	Sports Baseball , Soccer , Basketball ...
World Català , Dansk , Deutsch , Español , Français , Italiano , 日本語 , Nederlands , Polski , Русский , Svenska ...		

Become an Editor Help build the largest human-edited directory of the web



Copyright © 2012 Netscape

5,114,642 sites - 96,895 editors - over 1,014,858 categories

How to organize the web

- **Second try: Web Search**

- Information Retrieval investigates:

- Find relevant docs in a small and trusted set e.g., Newspaper articles, Patents, etc. (“needle-in-a-haystack”)
- Limitation of keywords (synonyms, polysemy, etc)

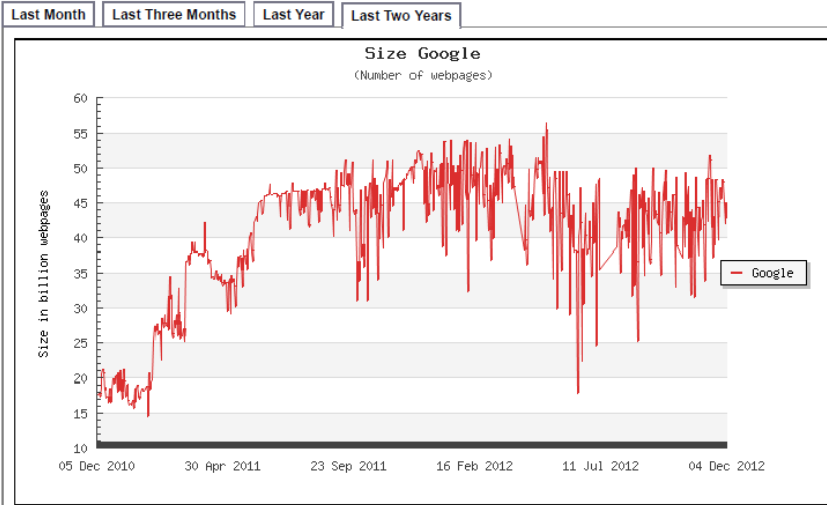
But: Web is huge, full of untrusted documents, random things, web spam, etc.

- Everyone can create a web page of high production value
- Rich diversity of people issuing queries
- Dynamic and constantly-changing nature of web content

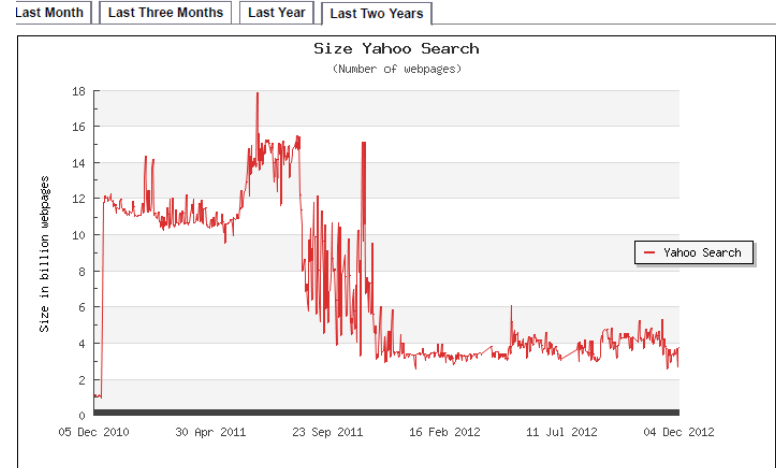
Size of the Search Index



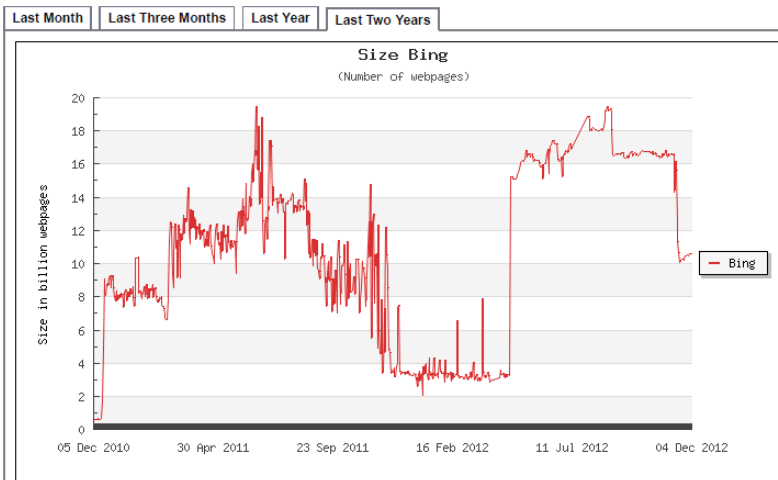
The size of the World Wide Web:
Estimated size of Google's index



The size of the World Wide Web:
Estimated size of Yahoo Search index



The size of the World Wide Web:
Estimated size of Bing index



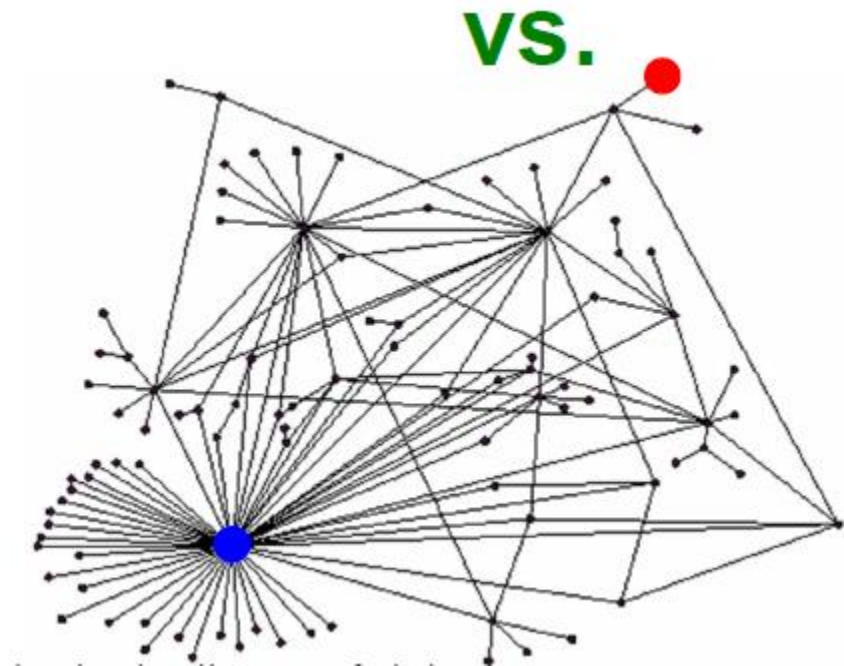
How to organize the web

- Third try (the Google era): using the web graph
 - Swift from relevance to *authoritativeness*
 - It is not only important that a page is relevant, but that it is also important on the web
- For example, what kind of results would we like to get for the query “greek newspapers”?

Link Analysis

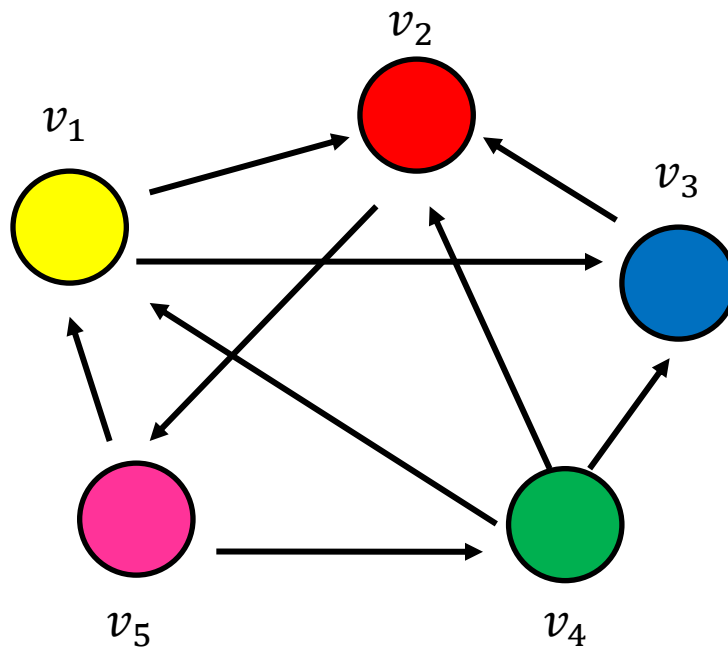
- Not all web pages are equal on the web
- The links act as endorsements:
 - When page p links to q it endorses the content of the content of q

What is the simplest way to measure importance of a page on the web?



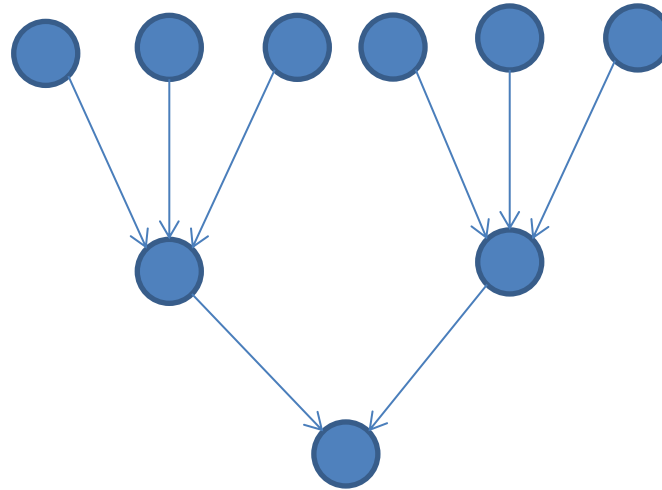
Rank by Popularity

- Rank pages according to the number of incoming edges (**in-degree**, **degree centrality**)



- 1. Red Page**
- 2. Yellow Page**
- 3. Blue Page**
- 4. Purple Page**
- 5. Green Page**

Popularity



- It is not important only **how many** link to you, but also **how important** are the people that link to you.
- **Good** authorities are pointed by **good** authorities
 - Recursive definition of importance

THE PAGERANK ALGORITHM

PageRank

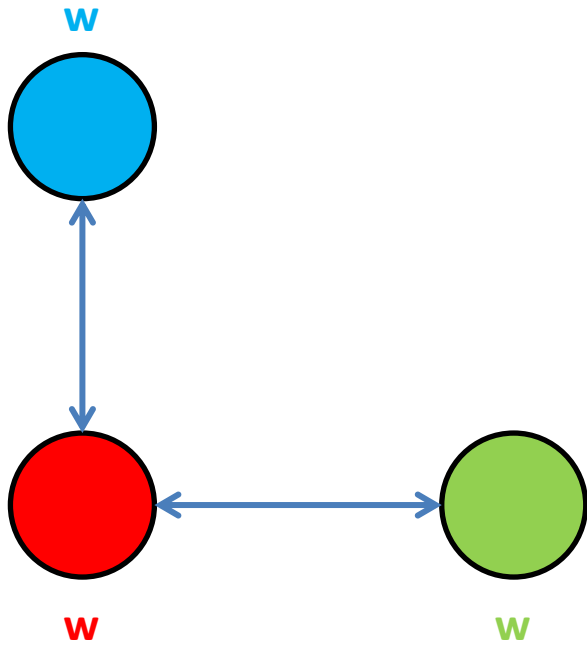
- **Good** authorities should be pointed by **good** authorities
 - The value of a node is the value of the nodes that point to it.
- How do we implement that?
 - Assume that we have **a unit of authority** to distribute to all nodes.
 - Initially each node gets $\frac{1}{n}$ amount of authority
 - Each node **distributes** the authority value they have **to their neighbors**
 - The authority value of each node is the sum of the **authority fractions** it collects from its neighbors.

$$w_v = \sum_{u \rightarrow v} \frac{1}{d_{out}(u)} w_u$$

w_v : the **PageRank value** of node v

Recursive definition

A simple example



$$w + w + w = 1$$

$$w = w + w$$

$$w = \frac{1}{2} w$$

$$w = \frac{1}{2} w$$

- Solving the system of equations we get the authority values for the nodes

$$- w = \frac{1}{2} \quad w = \frac{1}{4} \quad w = \frac{1}{4}$$

A more complex example

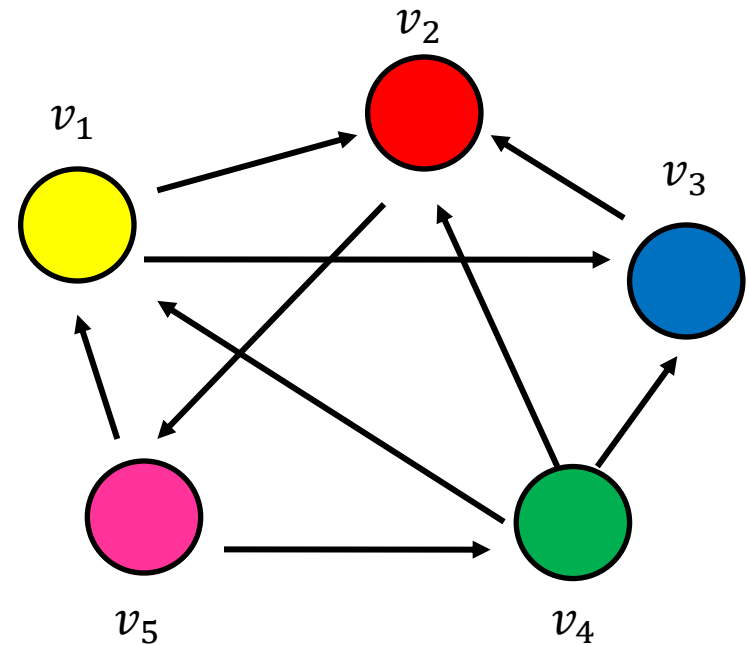
$$w_1 = \frac{1}{3} w_4 + \frac{1}{2} w_5$$

$$w_2 = \frac{1}{2} w_1 + w_3 + \frac{1}{3} w_4$$

$$w_3 = \frac{1}{2} w_1 + \frac{1}{3} w_4$$

$$w_4 = \frac{1}{2} w_5$$

$$w_5 = w_2$$



$$w_v = \sum_{u \rightarrow v} \frac{1}{d_{out}(u)} w_u$$

Computing PageRank weights

- A simple way to compute the weights is by iteratively updating the weights
- PageRank Algorithm

Initialize all PageRank weights to $\frac{1}{n}$

Repeat:

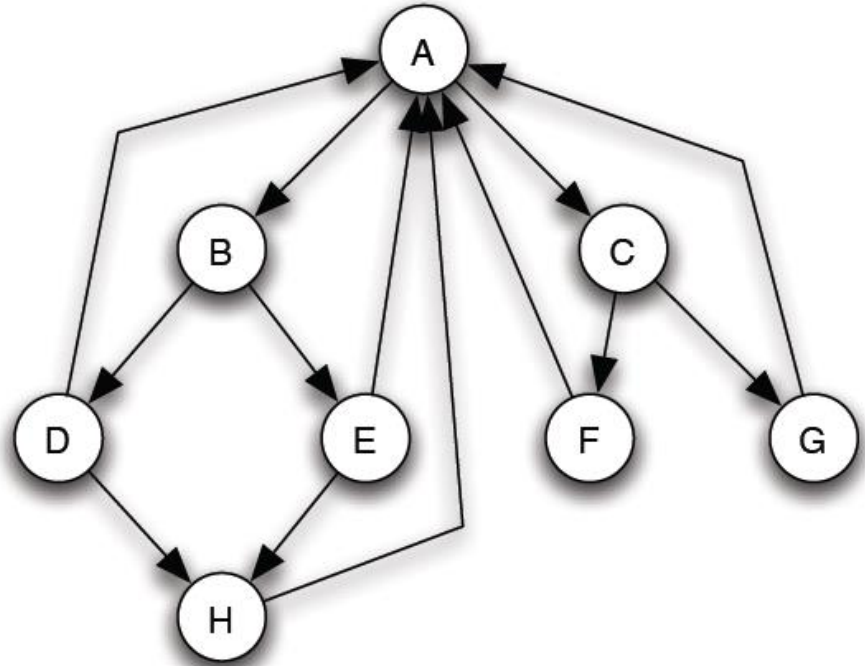
$$w_v = \sum_{u \rightarrow v} \frac{1}{d_{out}(u)} w_u$$

Until the weights do not change

- This process converges

PageRank

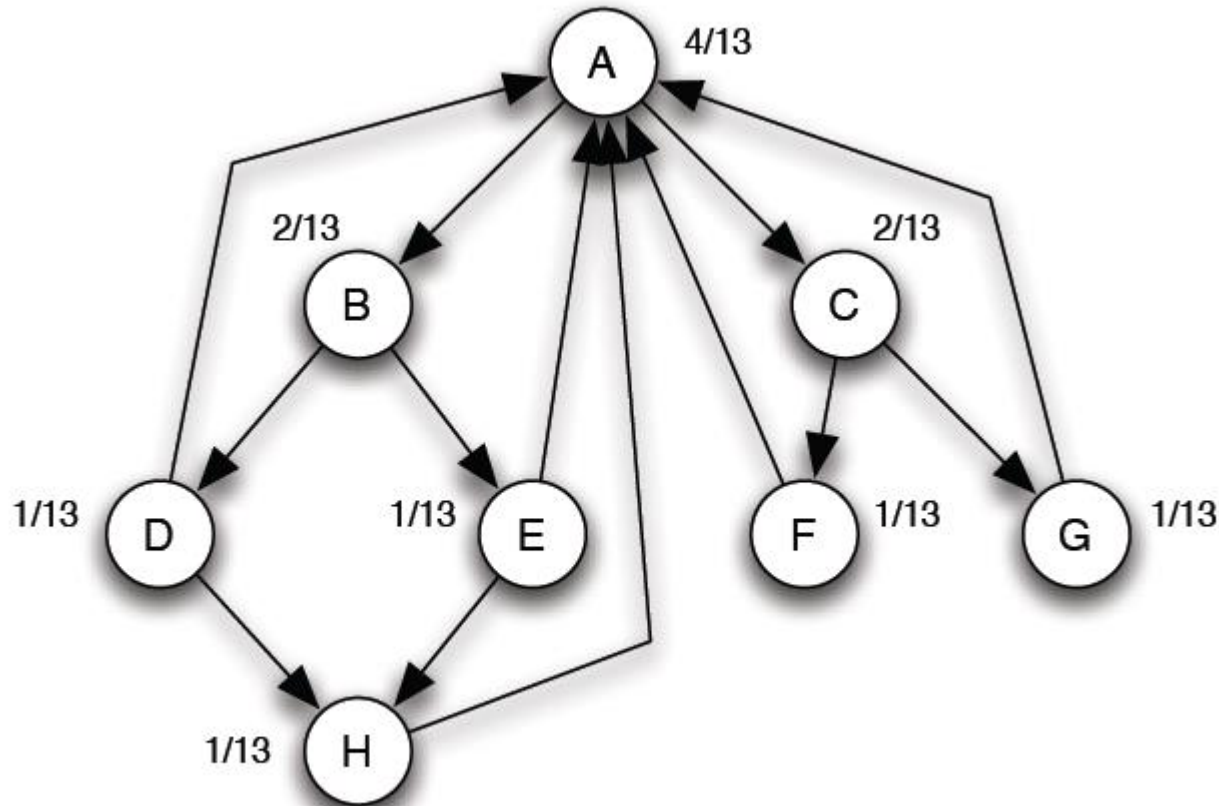
Initially, all nodes PageRank $1/8$



Step	A	B	C	D	E	F	G	H
1	$1/2$	$1/16$	$1/16$	$1/16$	$1/16$	$1/16$	$1/16$	$1/8$
2	$3/16$	$1/4$	$1/4$	$1/32$	$1/32$	$1/32$	$1/32$	$1/16$

- ✓ As a kind of “fluid” that circulates through the network
- ✓ The total PageRank in the network remains constant (no need to normalize)

PageRank: equilibrium



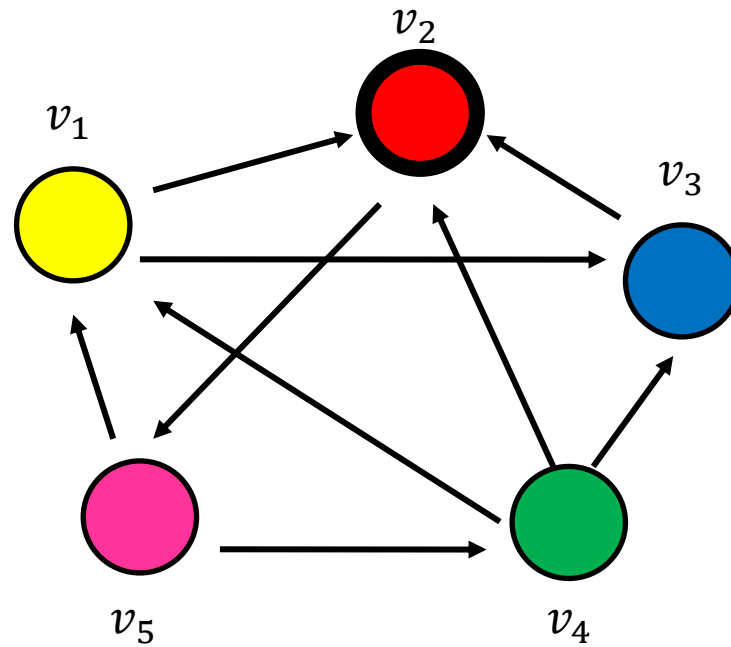
- A simple way to *check* whether an assignment of numbers forms an equilibrium set of PageRank values: check that they sum to 1, and that when apply the Basic PageRank Update Rule, we get the same values back.
- If the network is *strongly connected*, then there is a unique set of equilibrium values.

Random Walks on Graphs

- The algorithm defines a **random walk** on the graph
- Random walk:
 - **Start** from a node chosen **uniformly at random** with probability $\frac{1}{n}$.
 - **Pick** one of the **outgoing edges** **uniformly at random**
 - **Move** to the destination of the edge
 - Repeat.
- The **Random Surfer** model
 - Users wander on the web, following links.

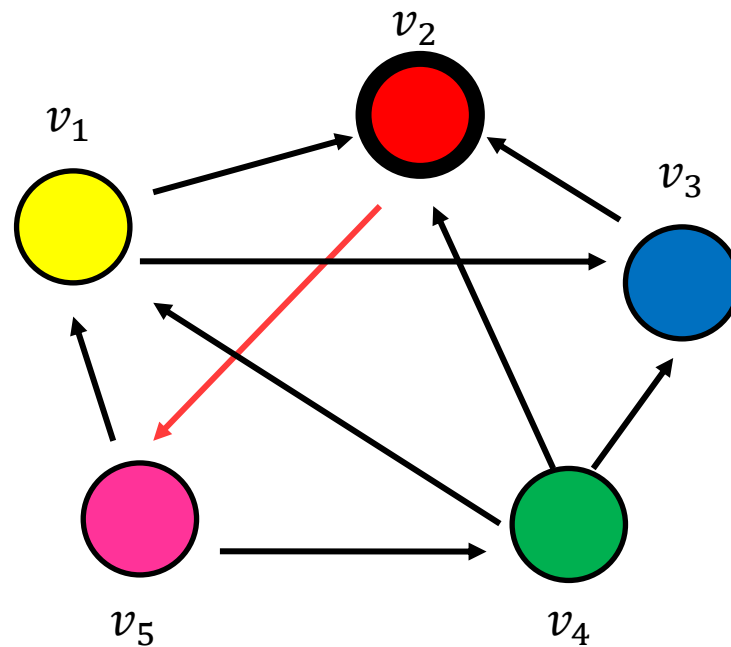
Example

- Step 0



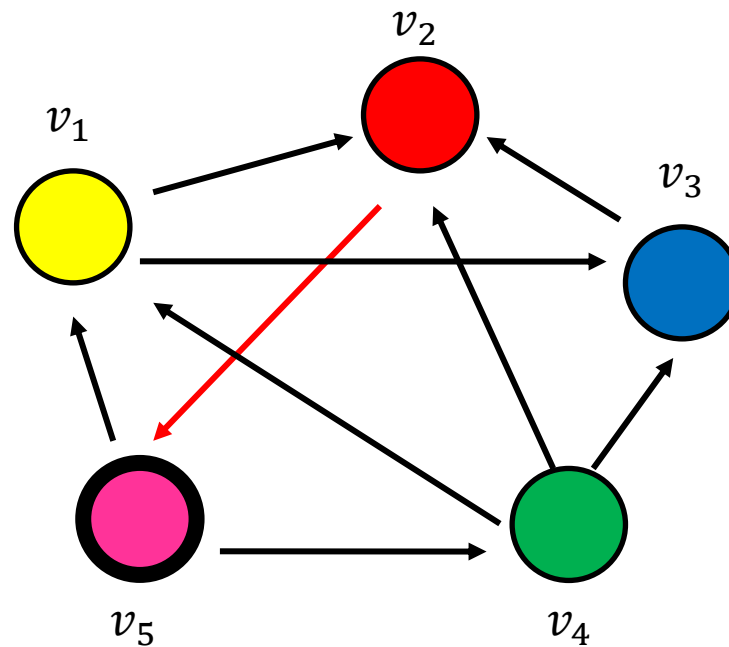
Example

- Step 0



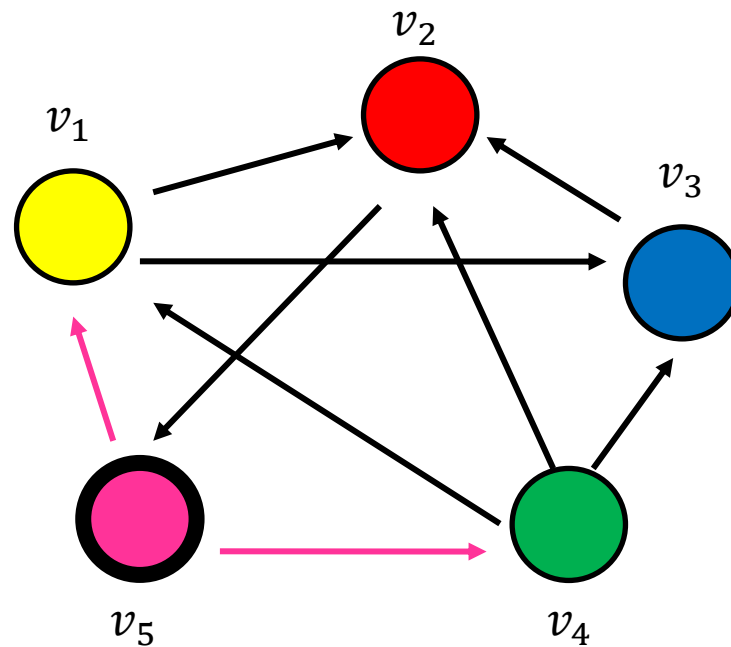
Example

- Step 1



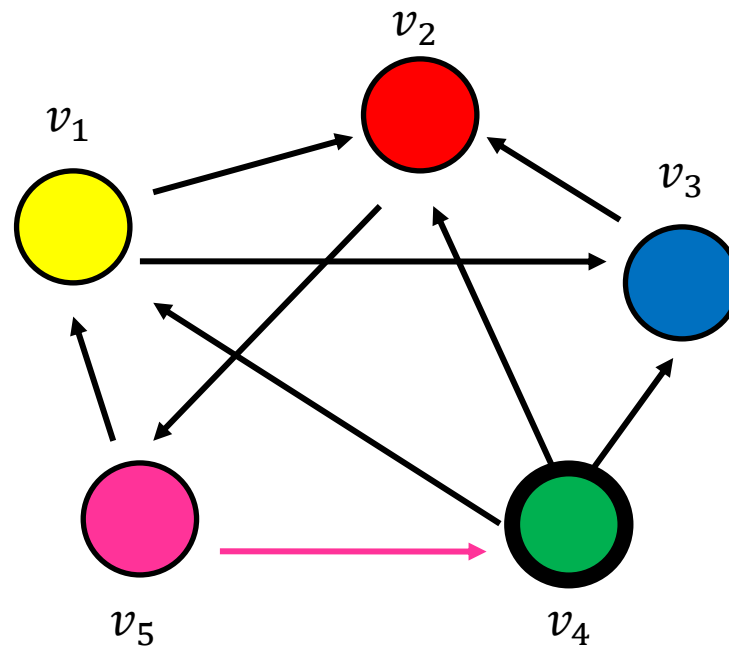
Example

- Step 1



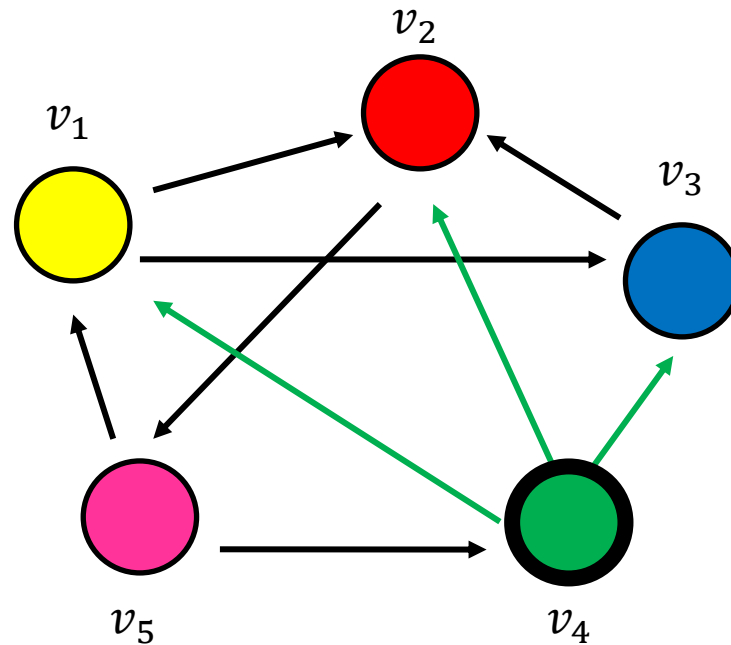
Example

- Step 2



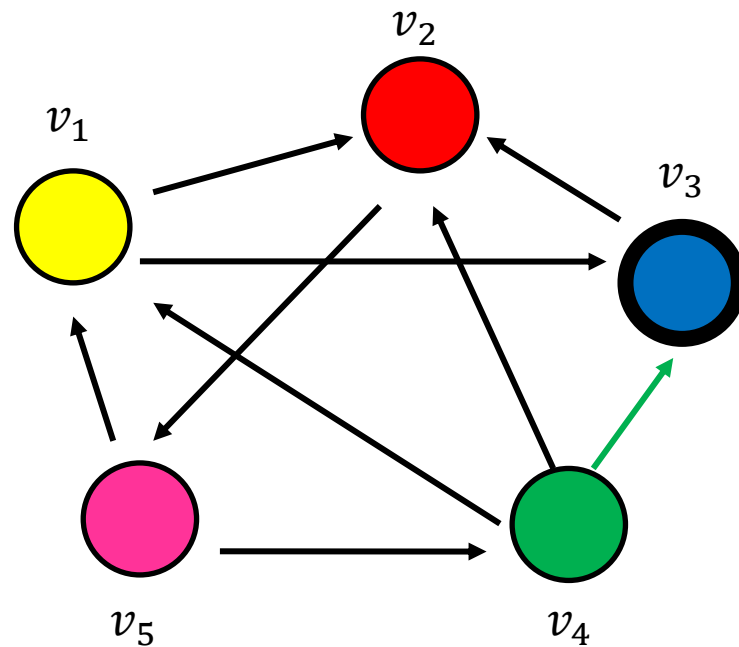
Example

- Step 2



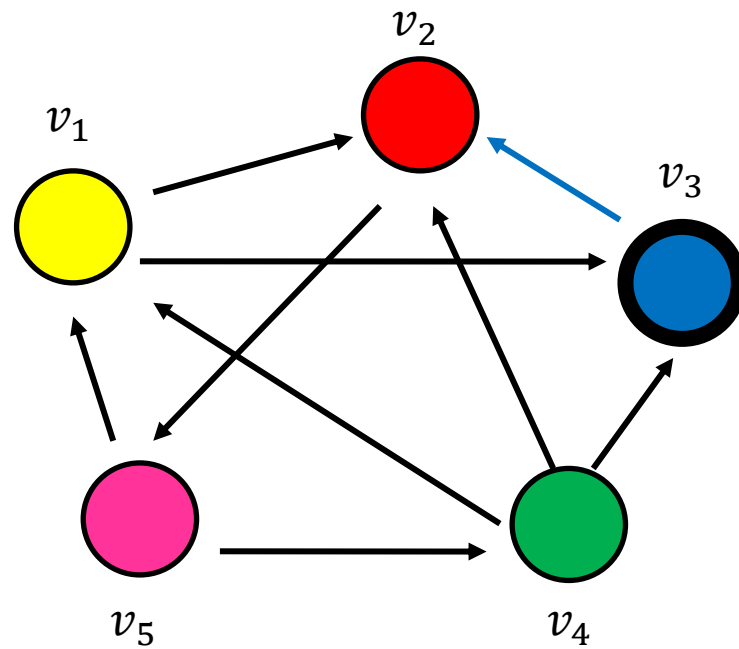
Example

- Step 3



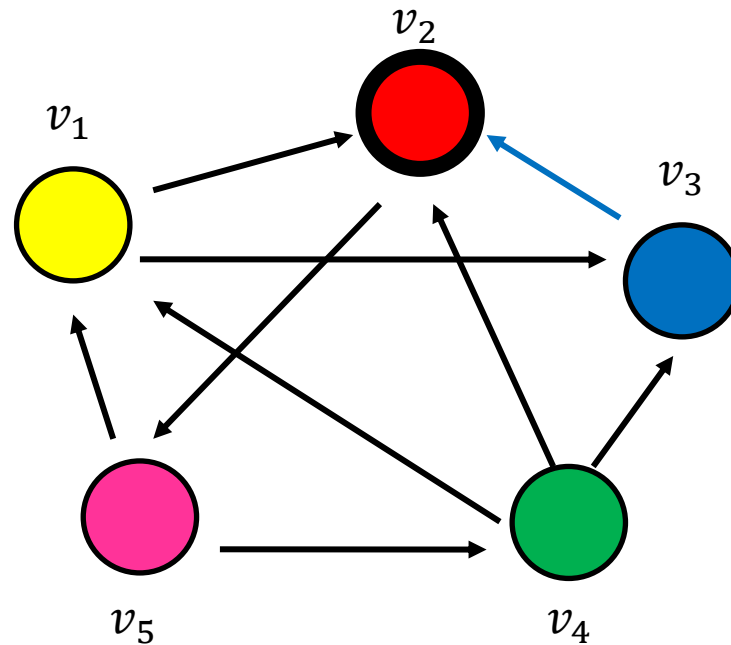
Example

- Step 3



Example

- Step 4...



Random walk

- Question: what is the probability p_i^t of being at node i after t steps?

$$p_1^0 = \frac{1}{5}$$

$$p_2^0 = \frac{1}{5}$$

$$p_3^0 = \frac{1}{5}$$

$$p_4^0 = \frac{1}{5}$$

$$p_5^0 = \frac{1}{5}$$

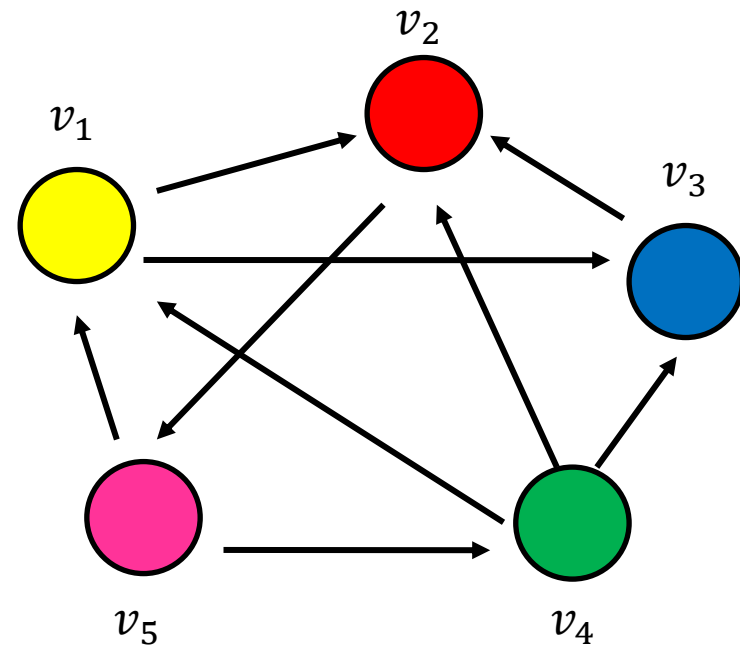
$$p_1^t = \frac{1}{3}p_4^{t-1} + \frac{1}{2}p_5^{t-1}$$

$$p_2^t = \frac{1}{2}p_1^{t-1} + p_3^{t-1} + \frac{1}{3}p_4^{t-1}$$

$$p_3^t = \frac{1}{2}p_1^{t-1} + \frac{1}{3}p_4^{t-1}$$

$$p_4^t = \frac{1}{2}p_5^{t-1}$$

$$p_5^t = p_2^{t-1}$$



Markov chains

- A Markov chain describes a **discrete time stochastic process** over a set of states

$$S = \{s_1, s_2, \dots, s_n\}$$

according to a transition probability matrix $P = \{P_{ij}\}$

- P_{ij} = probability of moving to state j when at state i

- Matrix P has the property that the entries of all **rows sum to 1**

$$\sum_j P[i, j] = 1$$

A matrix with this property is called **stochastic**

- **State probability distribution**: The vector $p^t = (p_1^t, p_2^t, \dots, p_n^t)$ that stores the probability of being at state s_i after t steps
- **Memorylessness property**: The **next state** of the chain **depends only at the current state** and not on the past of the process (**first order MC**)
 - **Higher order** MCs are also possible
- **Markov Chain Theory**: After infinite steps the **state probability vector converges** to a **unique** distribution if the chain is **irreducible** (possible to get from any state to any other state) and **aperiodic**

Random walks

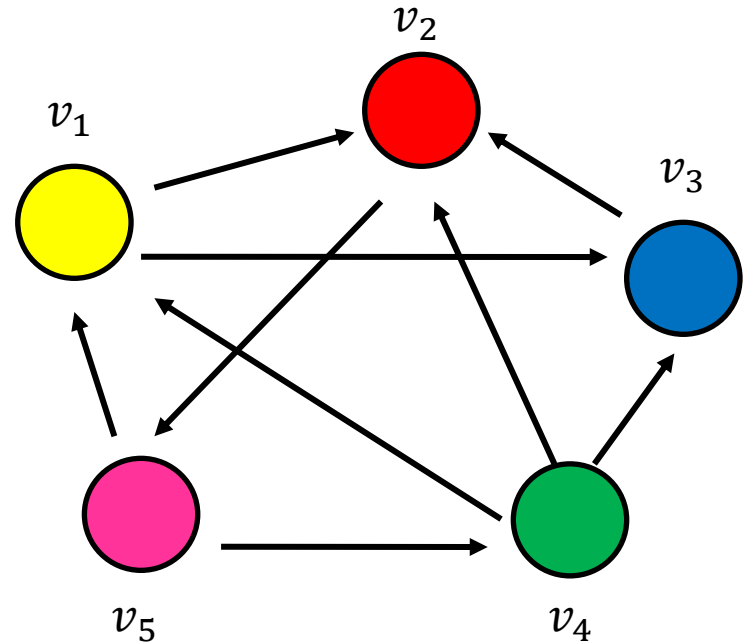
- Random walks on graphs correspond to Markov Chains
 - The set of states S is the set of nodes of the graph G
 - The **transition probability matrix** is the probability that we follow an edge from one node to another

$$P[i, j] = 1 / \text{deg}_{out}(i)$$

An example

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

$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$



Node Probability vector

- The vector $p^t = (p_1^t, p_2^t, \dots, p_n^t)$ that stores the probability of being at node v_i at step t
- p_i^0 = the probability of starting from state i (usually) set to **uniform**
- We can compute the vector p^t at step t using a vector-matrix multiplication

$$p^t = p^{t-1} P$$

An example

$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$

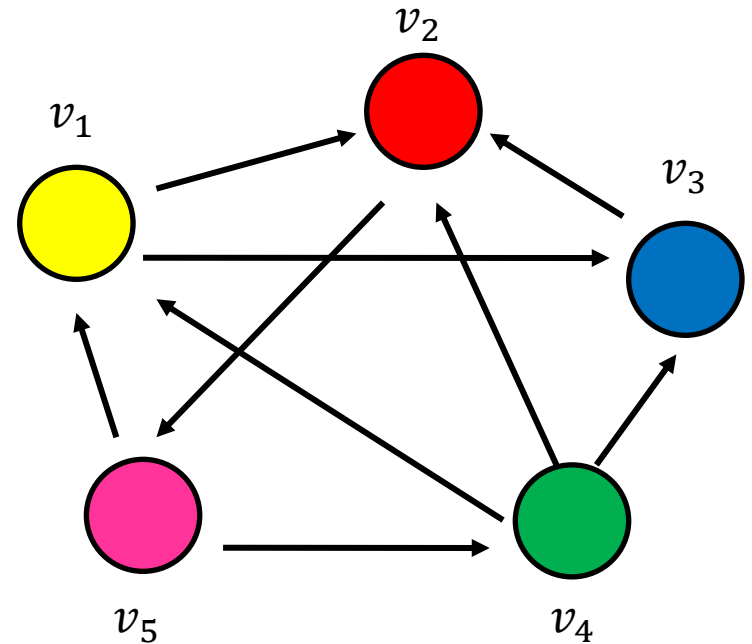
$$p_1^t = \frac{1}{3}p_4^{t-1} + \frac{1}{2}p_5^{t-1}$$

$$p_2^t = \frac{1}{2}p_1^{t-1} + p_3^{t-1} + \frac{1}{3}p_4^{t-1}$$

$$p_3^t = \frac{1}{2}p_1^{t-1} + \frac{1}{3}p_4^{t-1}$$

$$p_4^t = \frac{1}{2}p_5^{t-1}$$

$$p_5^t = p_2^{t-1}$$



Stationary distribution

- The **stationary distribution** of a random walk with transition matrix P , is a probability distribution π , such that $\pi = \pi P$
- The stationary distribution is an **eigenvector** of matrix P
 - the **principal left eigenvector** of P – stochastic matrices have maximum eigenvalue 1
- The probability π_i is the fraction of times that we visited state i as $t \rightarrow \infty$
- **Markov Chain Theory**: The random walk converges to a **unique stationary distribution independent of the initial vector** if the graph is **strongly connected**, and **not bipartite**.

Computing the stationary distribution

- The Power Method

Initialize q^0 to some distribution

Repeat

$$q^t = q^{t-1}P$$

Until convergence

- After many iterations $q^t \rightarrow \pi$ regardless of the initial vector q^0
- Power method because it computes $q^t = q^0 P^t$
- Rate of convergence
 - determined by the second eigenvalue $\frac{|\lambda_2|}{|\lambda_1|}$

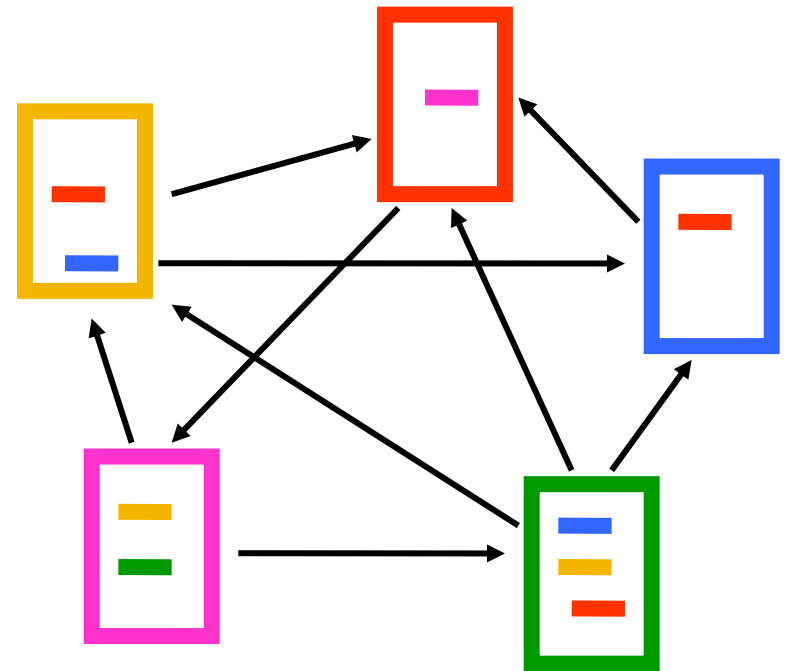
The stationary distribution

- What is the meaning of the stationary distribution π of a random walk?
- $\pi(i)$: the probability of being at node i after very large (infinite) number of steps
- $\pi = p_0 P^\infty$, where P is the transition matrix, p_0 the original vector
 - $P(i, j)$: probability of going from i to j in one step
 - $P^2(i, j)$: probability of going from i to j in two steps (probability of all paths of length 2)
 - $P^\infty(i, j) = \pi(j)$: probability of going from i to j in infinite steps – starting point does not matter.

The PageRank random walk

- Vanilla random walk
 - make the adjacency matrix stochastic and run a random walk

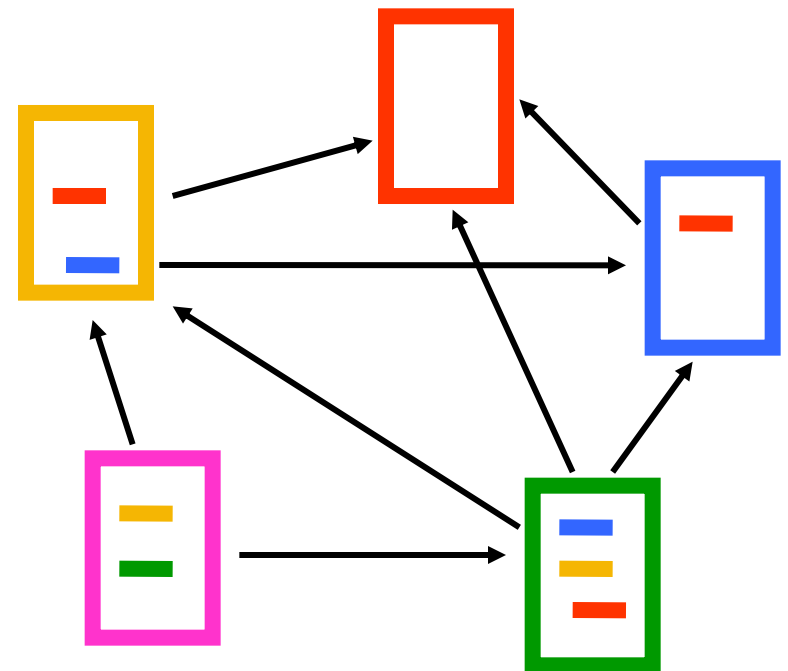
$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$



The PageRank random walk

- What about **sink** nodes?
 - what happens when the random walk moves to a node without any outgoing links?

$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$

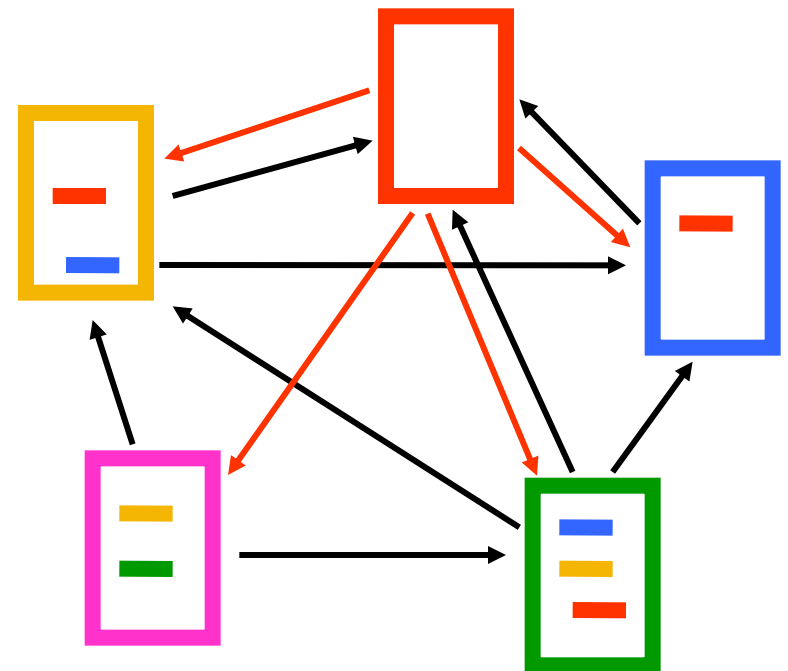


The PageRank random walk

- Replace these row vectors with a vector \mathbf{v}
 - typically, the uniform vector

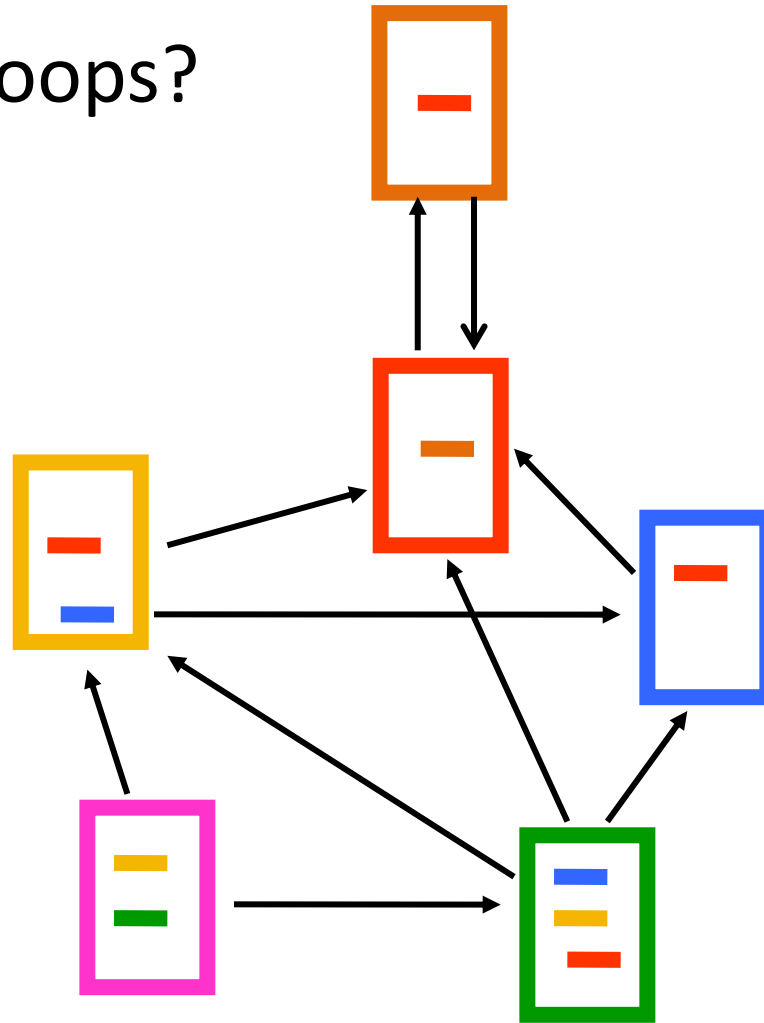
$$P' = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$

$$P' = P + d\mathbf{v}^T \quad d = \begin{cases} 1 & \text{if } i \text{ is sink} \\ 0 & \text{otherwise} \end{cases}$$



The PageRank random walk

- What about loops?
 - Spider traps



The PageRank random walk

- Add a **random jump** to vector \mathbf{v} with prob $1-\alpha$
 - typically, to a uniform vector
- Restarts after $1/(1-\alpha)$ steps in expectation
 - Guarantees irreducibility, convergence

$$P'' = \alpha \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \end{bmatrix} + (1-\alpha) \begin{bmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{bmatrix}$$

$P'' = \alpha P' + (1-\alpha)uv^T$, where u is the vector of all 1s

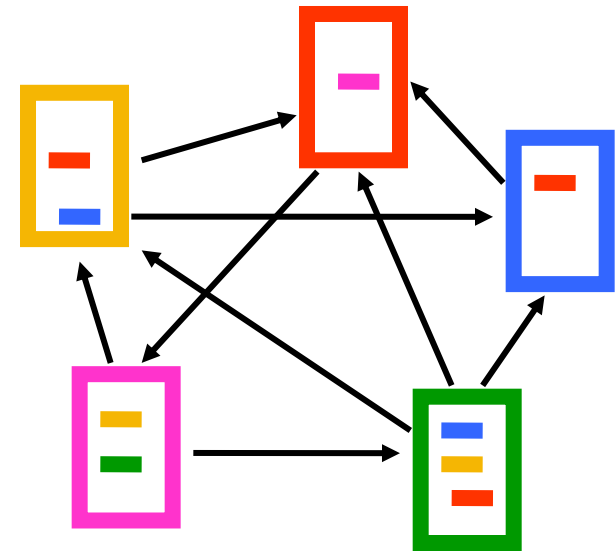
Random walk with restarts

PageRank algorithm [BP98]

- The Random Surfer model
 - pick a page at random
 - with probability $1 - \alpha$ jump to a random page
 - with probability α follow a random outgoing link
- Rank according to the stationary distribution

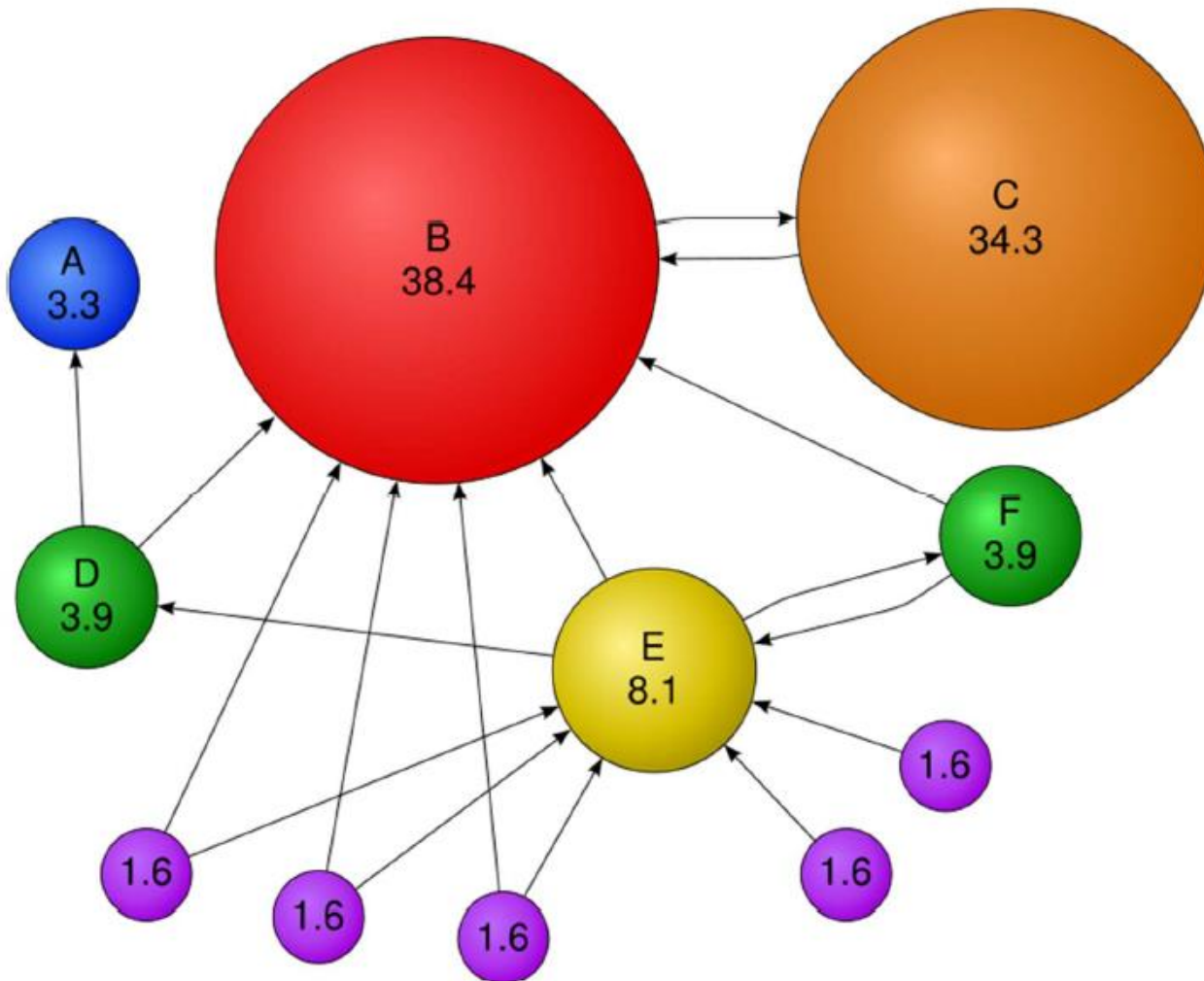
$$PR(p) = \alpha \sum_{q \rightarrow p} \frac{PR(q)}{|Out(q)|} + (1 - \alpha) \frac{1}{n}$$

$\alpha = 0.85$ in most cases



1. Red Page
2. Purple Page
3. Yellow Page
4. Blue Page
5. Green Page

PageRank: Example



Stationary distribution with random jump

- If v is the jump vector

$$\begin{aligned}p^0 &= v \\p^1 &= \alpha p^0 P' + (1 - \alpha)v = \alpha v P' + (1 - \alpha)v \\p^2 &= \alpha p^1 P' + (1 - \alpha)v = \alpha^2 v P'^2 + (1 - \alpha)v \alpha P' + (1 - \alpha)v \\&\vdots \\p^\infty &= (1 - \alpha)v + (1 - \alpha)v \alpha P' + (1 - \alpha)v \alpha^2 P'^2 + \dots \\&= (1 - \alpha)(I - \alpha P')^{-1}\end{aligned}$$

- With the random jump the **shorter paths** are more important, since the weight decreases **exponentially**
 - makes sense when thought of as a restart
- If v is **not uniform**, we can bias the random walk towards the nodes that are **close** to v
 - **Personalized** and **Topic-Specific** Pagerank.

Effects of random jump

- Guarantees **convergence** to unique distribution
- Motivated by the concept of **random surfer**
- Offers additional flexibility
 - **personalization**
 - **anti-spam**
- Controls the **rate of convergence**
 - the second eigenvalue of matrix P'' is **α**

Random walks on undirected graphs

- For undirected graphs, the stationary distribution is proportional to the degrees of the nodes
 - Thus in this case a random walk is the same as degree popularity
- This is not longer true if we do random jumps
 - Now the short paths play a greater role, and the previous distribution does not hold.

PageRank implementation

- Store the graph in adjacency list, or list of edges
- Keep current pagerank values and new pagerank values
- Go through edges and update the values of the destination nodes.
- Repeat until the difference (L_1 or L_∞ difference) is below some small value ϵ .

A (Matlab-friendly) PageRank algorithm

- Performing vanilla power method is now too expensive – the matrix is not sparse

$$q^0 = v$$

$$t = 1$$

repeat

$$q^t = (P'')^T q^{t-1}$$

$$\delta = \|q^t - q^{t-1}\|$$

$$t = t + 1$$

until $\delta < \epsilon$

Efficient computation of $y = (P'')^T x$

$$y = \alpha P^T x$$

$$\beta = \|x\|_1 - \|y\|_1$$

$$y = y + \beta v$$

P = normalized adjacency matrix

$P' = P + dv^T$, where d_i is 1 if i is sink and 0 o.w.

$P'' = \alpha P' + (1-\alpha)uv^T$, where u is the vector of all 1s

PageRank history

- Huge advantage for Google in the early days
 - It gave a way to get an idea for the **value of a page**, which was useful in many different ways
 - Put an **order to the web**.
 - After a while it became clear that the **anchor text** was probably more important for ranking
 - Also, **link spam** became a new (dark) art
- Flood of research
 - Numerical analysis got rejuvenated
 - Huge number of variations
 - **Efficiency** became a great issue.
 - Huge number of applications in different fields
 - Random walk is often referred to as PageRank.

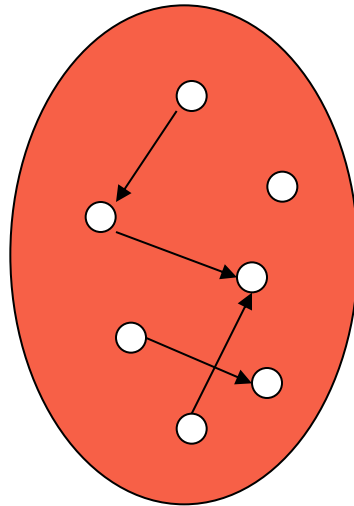
THE HITS ALGORITHM

The HITS algorithm

- Another algorithm proposed around the same time as PageRank for using the hyperlinks to rank pages
 - Kleinberg: then an intern at IBM Almaden
 - IBM never made anything out of it

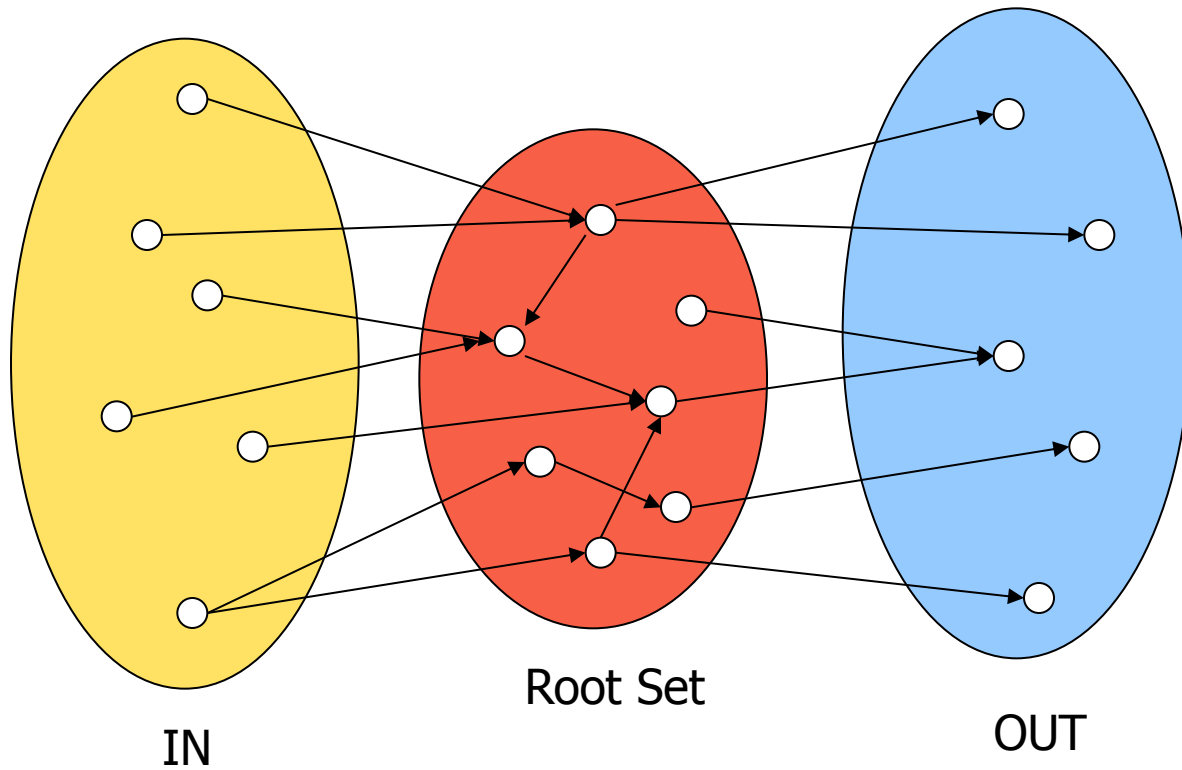
Query dependent input

Root set obtained from a text-only search engine

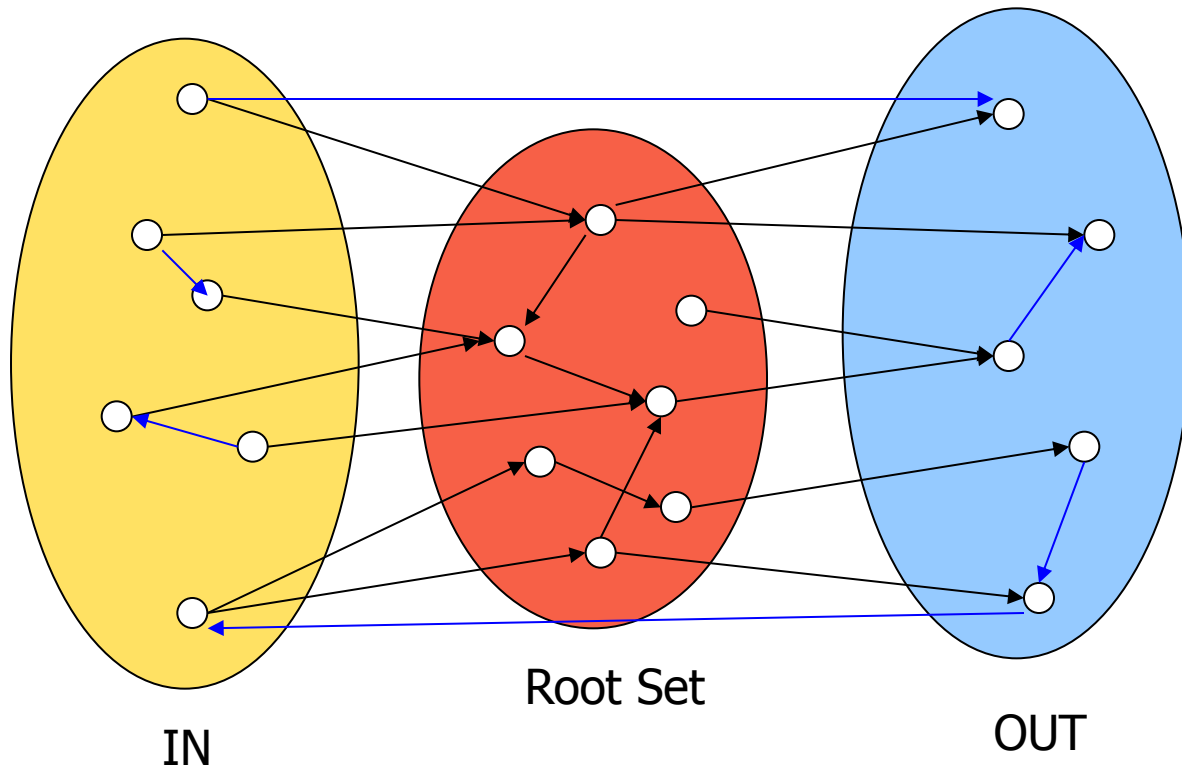


Root Set

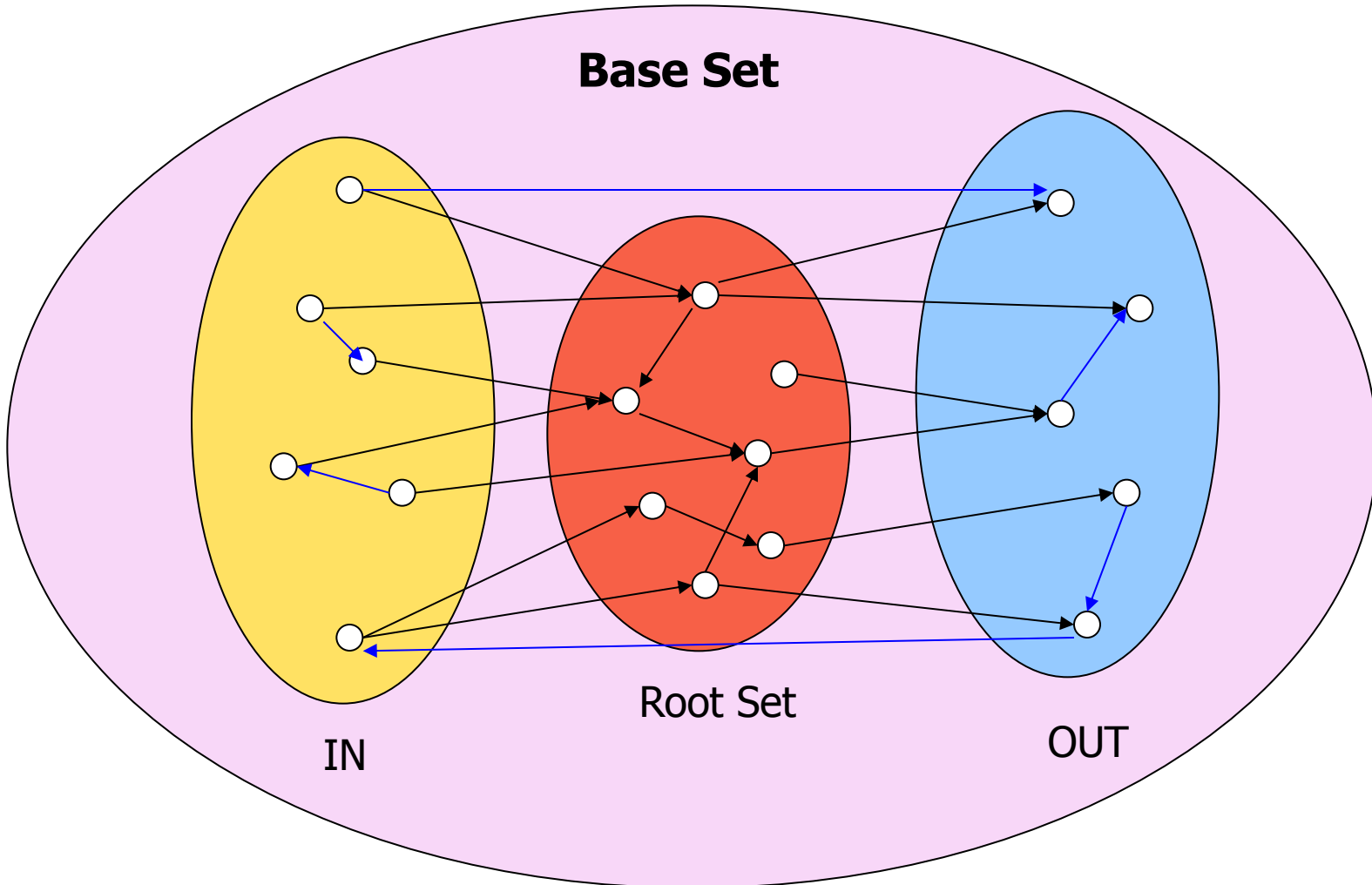
Query dependent input



Query dependent input

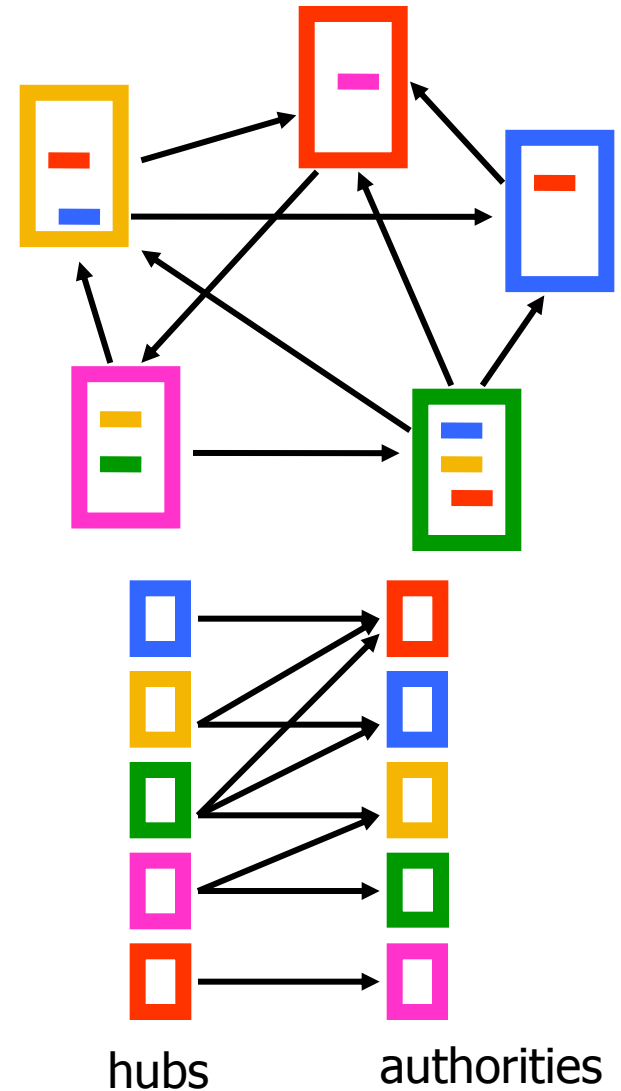


Query dependent input



Hubs and Authorities [K98]

- Authority is not necessarily transferred directly between authorities
- Pages have double identity
 - hub identity
 - authority identity
- Good hubs point to good authorities
- Good authorities are pointed by good hubs



Hubs and Authorities

- Two kind of weights:
 - Hub weight
 - Authority weight
- The hub weight is the sum of the authority weights of the authorities pointed to by the hub
- The authority weight is the sum of the hub weights that point to this authority.

HITS Algorithm

- Initialize all weights to 1.
- Repeat until convergence
 - *O* operation : hubs collect the weight of the authorities

$$h_i = \sum_{j:i \rightarrow j} a_j$$

- *I* operation: authorities collect the weight of the hubs

$$a_i = \sum_{j:j \rightarrow i} h_j$$

- Normalize weights under some norm

HITS and eigenvectors

- The HITS algorithm is a **power-method** eigenvector computation
- In vector terms
 - $a^t = A^T h^{t-1}$ and $h^t = A a^{t-1}$
 - $a^t = A^T A a^{t-1}$ and $h^t = A A^T h^{t-1}$
 - Repeated iterations will converge to the eigenvectors
- The **authority** weight vector a is the **eigenvector** of $A^T A$ and the **hub** weight vector h is the **eigenvector** of $A A^T$
- The vectors a and h are called the **singular vectors** of the matrix A

Singular Value Decomposition

$$A = U \Sigma V^T = \begin{bmatrix} \vec{u}_1 & \vec{u}_2 & \cdots & \vec{u}_r \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_r \end{bmatrix} \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_r \end{bmatrix}$$

$[n \times r] \quad [r \times r] \quad [r \times n]$

- r : rank of matrix A
- $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r$: singular values (square roots of eig-vals $AA^T, A^T A$)
- $\vec{u}_1, \vec{u}_2, \dots, \vec{u}_r$: left singular vectors (eig-vectors of AA^T)
- $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_r$: right singular vectors (eig-vectors of $A^T A$)

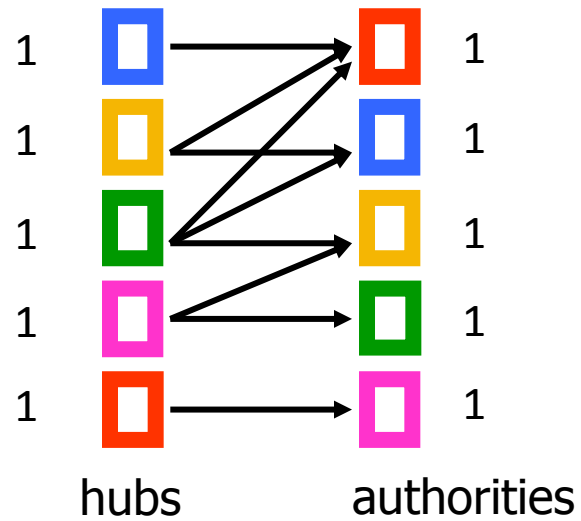
$$A = \sigma_1 \vec{u}_1 \vec{v}_1^T + \sigma_2 \vec{u}_2 \vec{v}_2^T + \cdots + \sigma_r \vec{u}_r \vec{v}_r^T$$

Why does the Power Method work?

- If a matrix R is real and symmetric, it has real eigenvalues and eigenvectors: $(\lambda_1, w_1), (\lambda_2, w_2), \dots, (\lambda_r, w_r)$
 - r is the rank of the matrix
 - $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_r|$
- For any matrix R , the eigenvectors w_1, w_2, \dots, w_r of R define a basis of the vector space
 - For any vector x , $Rx = \alpha_1 w_1 + \alpha_2 w_2 + \dots + \alpha_r w_r$
- After t multiplications we have:
 - $R^t x = \lambda_1^{t-1} \alpha_1 w_1 + \lambda_2^{t-1} \alpha_2 w_2 + \dots + \lambda_r^{t-1} \alpha_r w_r$
- Normalizing (divide by λ_1^{t-1}) leaves only the term w_1 .

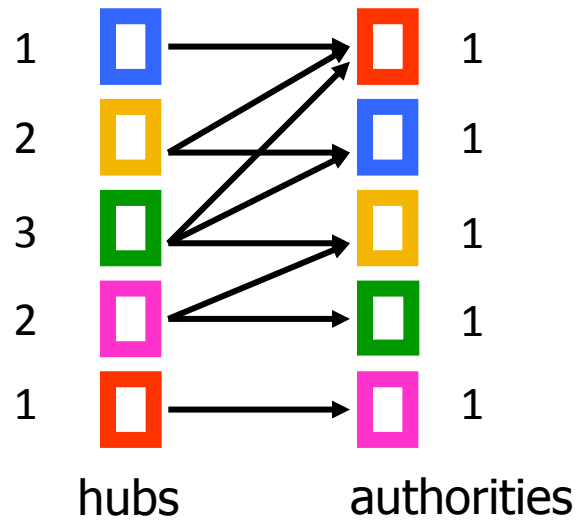
Example

Initialize



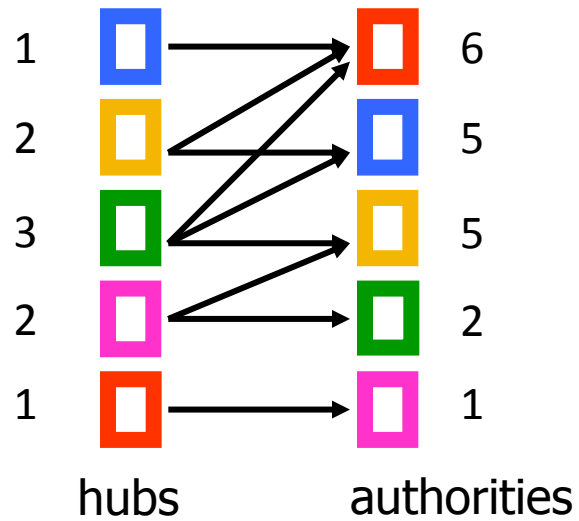
Example

Step 1: O operation



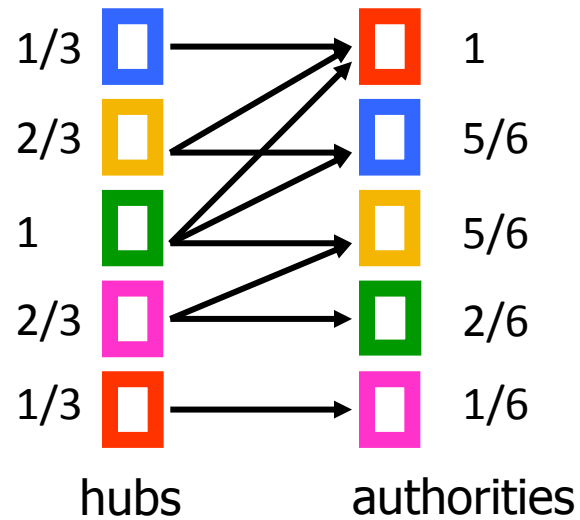
Example

Step 1: I operation



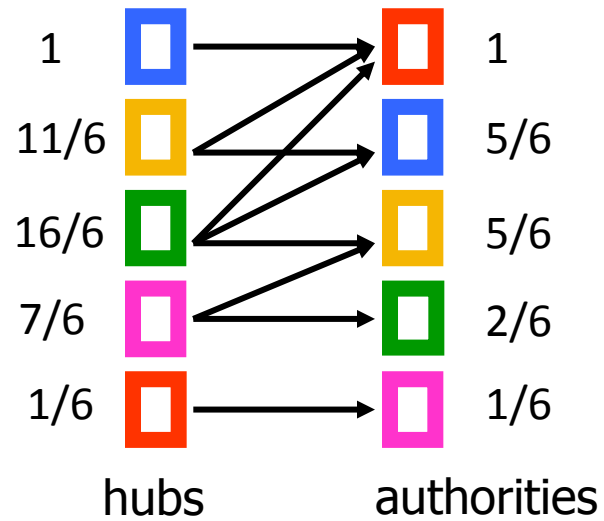
Example

Step 1: Normalization (Max norm)



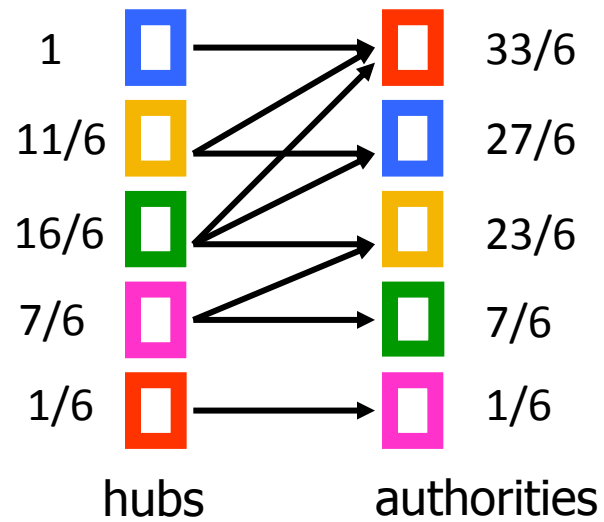
Example

Step 2: 0 step



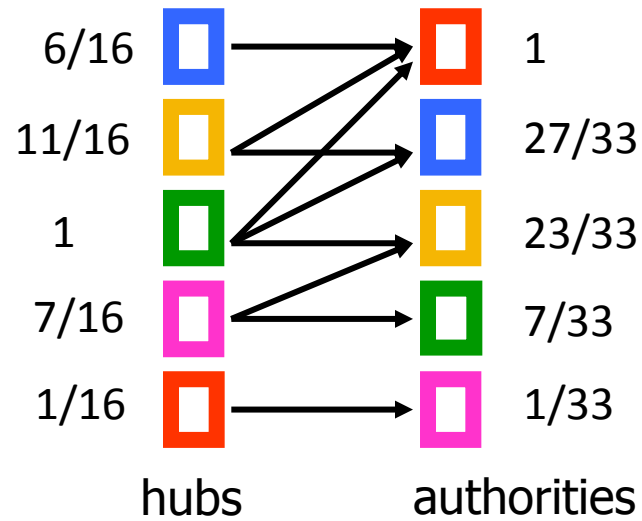
Example

Step 2: 1 step



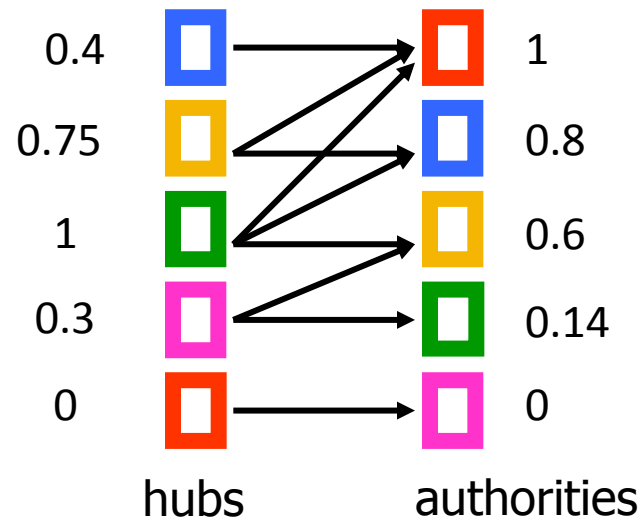
Example

Step 2: Normalization



Example

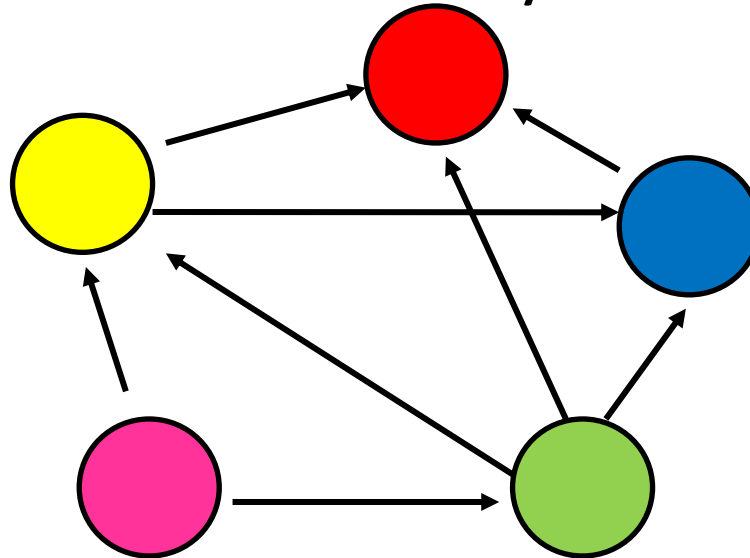
Convergence



**ABSORBING RANDOM WALKS
LABEL PROPAGATION
OPINION FORMATION ON SOCIAL
NETWORKS**

Random walk with absorbing nodes

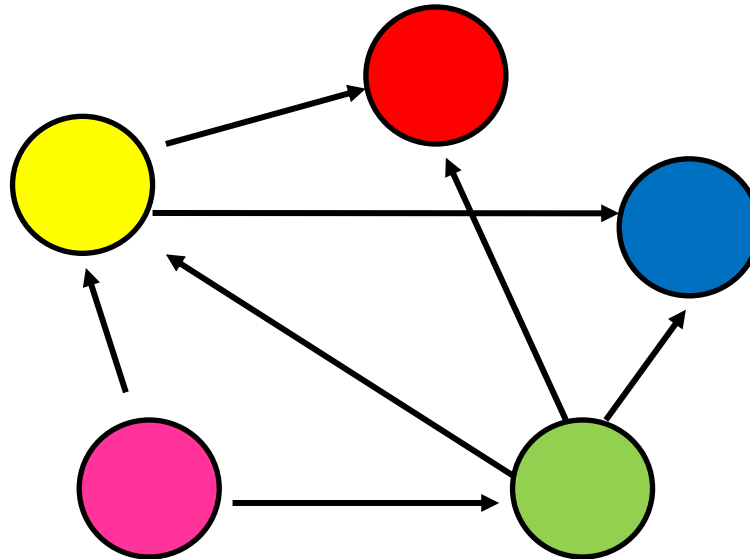
- What happens if we do a random walk on this graph? What is the stationary distribution?



- All the probability mass on the red **sink** node:
 - The red node is an **absorbing node**

Random walk with absorbing nodes

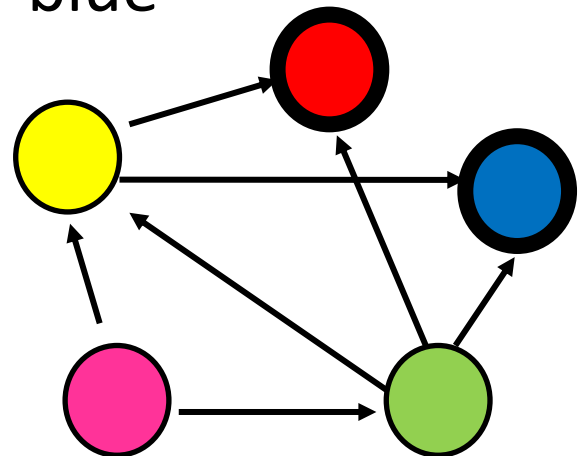
- What happens if we do a random walk on this graph? What is the stationary distribution?



- There are two absorbing nodes: the red and the blue.
- The probability mass will be **divided** between the two

Absorption probability

- If there are more than one **absorbing nodes** in the graph a random walk that starts from a **non-absorbing** node will be absorbed in one of them with some probability
 - The **probability of absorption** gives an estimate of how **close** the node is to red or blue



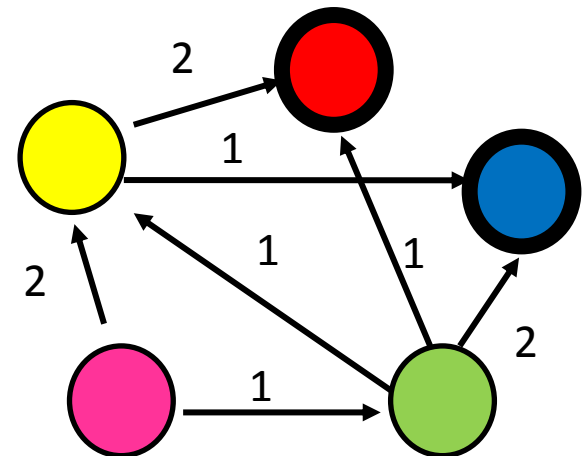
Absorption probability

- Computing the probability of being absorbed:
 - The **absorbing nodes** have probability 1 of being absorbed in themselves and zero of being absorbed in another node.
 - For the **non-absorbing nodes**, take the (weighted) average of the absorption probabilities of your neighbors
 - if one of the neighbors is the absorbing node, it has probability 1
 - Repeat until convergence (= very small change in probs)

$$P(\text{Red}|\text{Pink}) = \frac{2}{3}P(\text{Red}|\text{Yellow}) + \frac{1}{3}P(\text{Red}|\text{Green})$$

$$P(\text{Red}|\text{Green}) = \frac{1}{4}P(\text{Red}|\text{Yellow}) + \frac{1}{4}$$

$$P(\text{Red}|\text{Yellow}) = \frac{2}{3}$$



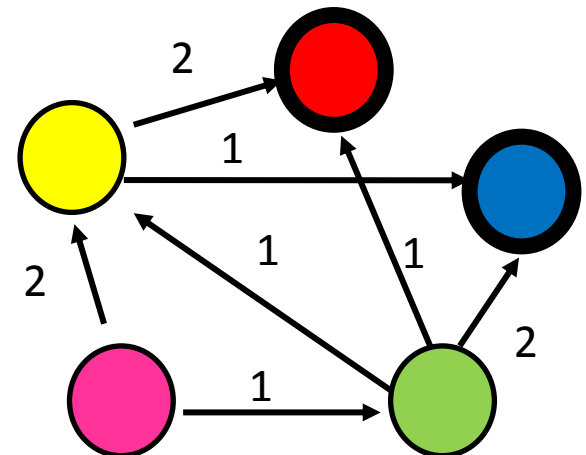
Absorption probability

- Computing the probability of being absorbed:
 - The **absorbing nodes** have probability 1 of being absorbed in themselves and zero of being absorbed in another node.
 - For the **non-absorbing nodes**, take the (weighted) average of the absorption probabilities of your neighbors
 - if one of the neighbors is the absorbing node, it has probability 1
 - Repeat until convergence (= very small change in probs)

$$P(\text{Blue}|\text{Pink}) = \frac{2}{3}P(\text{Blue}|\text{Yellow}) + \frac{1}{3}P(\text{Blue}|\text{Green})$$

$$P(\text{Blue}|\text{Green}) = \frac{1}{4}P(\text{Blue}|\text{Yellow}) + \frac{1}{2}$$

$$P(\text{Blue}|\text{Yellow}) = \frac{1}{3}$$

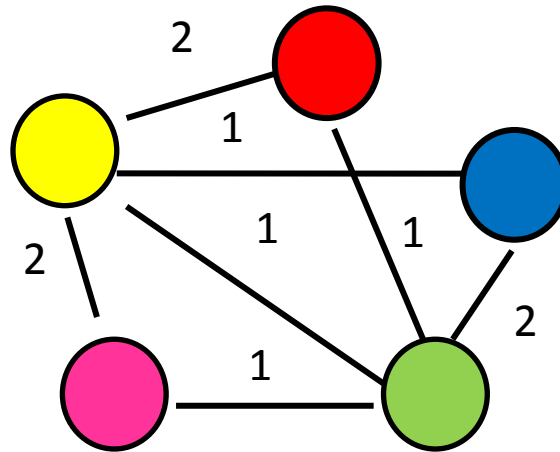


Why do we care?

- Why do we care to compute the absorption probability to sink nodes?
- Given a graph (**directed** or **undirected**) we can choose to **make** some nodes **absorbing**.
 - Simply **direct** all edges incident on the chosen nodes towards them.
- The absorbing random walk provides a measure of **proximity** of non-absorbing nodes to the chosen nodes.
 - Useful for **understanding** proximity in graphs
 - Useful for **propagation** in the graph
 - E.g, on a social network some nodes have **high income**, some have **low income**, to which income class is a non-absorbing node closer?

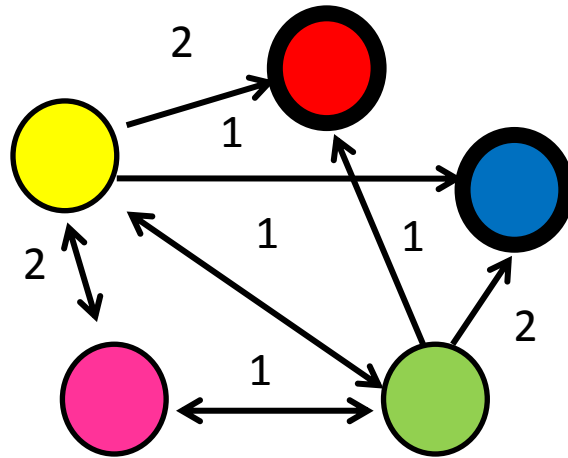
Example

- In this **undirected** graph we want to learn the proximity of nodes to the **red** and **blue** nodes



Example

- Make the nodes absorbing



Absorption probability

- Compute the absorption probabilities for red and blue

$$P(\text{Red}|\text{Pink}) = \frac{2}{3}P(\text{Red}|\text{Yellow}) + \frac{1}{3}P(\text{Red}|\text{Green})$$

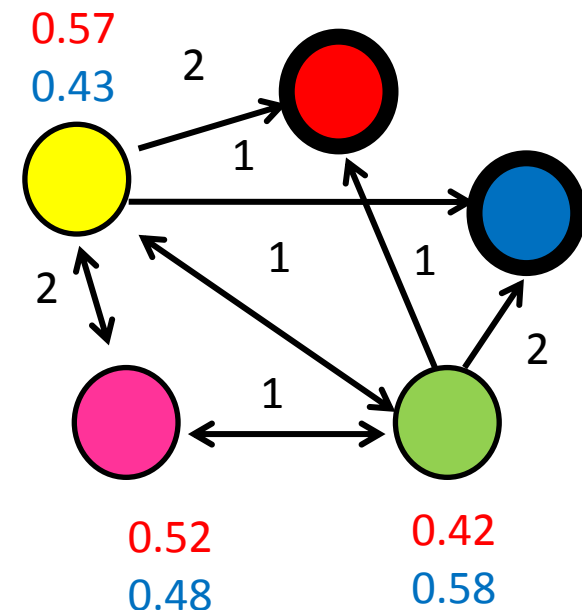
$$P(\text{Red}|\text{Green}) = \frac{1}{5}P(\text{Red}|\text{Yellow}) + \frac{1}{5}P(\text{Red}|\text{Pink}) + \frac{1}{5}$$

$$P(\text{Red}|\text{Yellow}) = \frac{1}{6}P(\text{Red}|\text{Green}) + \frac{1}{3}P(\text{Red}|\text{Pink}) + \frac{1}{3}$$

$$P(\text{Blue}|\text{Pink}) = 1 - P(\text{Red}|\text{Pink})$$

$$P(\text{Blue}|\text{Green}) = 1 - P(\text{Red}|\text{Green})$$

$$P(\text{Blue}|\text{Yellow}) = 1 - P(\text{Red}|\text{Yellow})$$

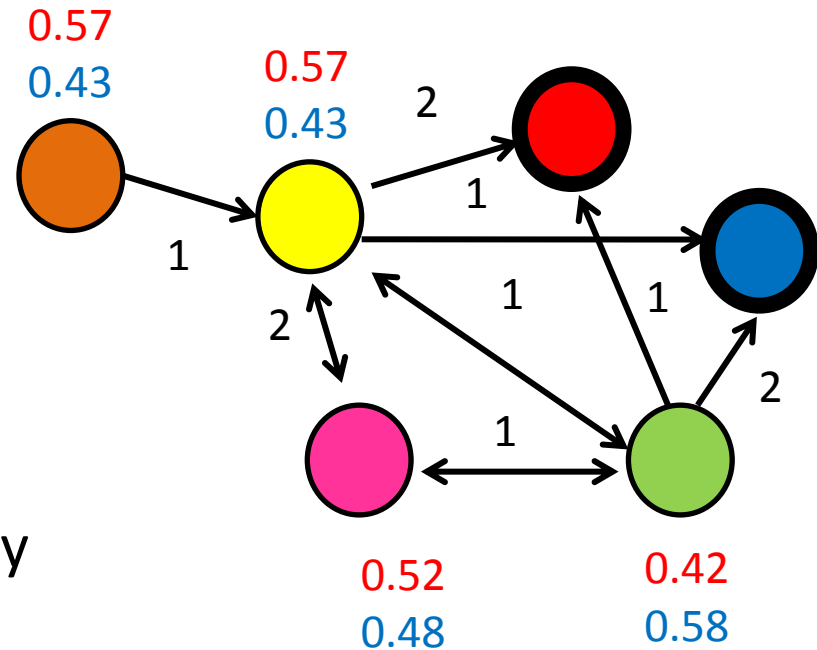


Penalizing long paths

- The orange node has the same probability of reaching red and blue as the yellow one

$$P(\text{Red}|\text{Orange}) = P(\text{Red}|\text{Yellow})$$

$$P(\text{Blue}|\text{Orange}) = P(\text{Blue}|\text{Yellow})$$



- Intuitively though it is further away

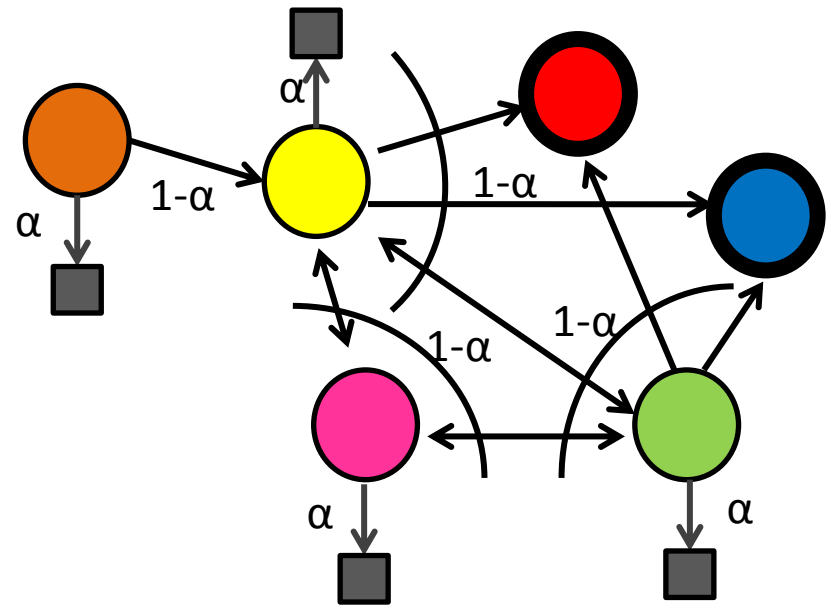
Penalizing long paths

- Add an **universal absorbing node** to which each node gets absorbed with probability α .

With probability α the random walk **dies**

With probability $(1-\alpha)$ the random walk continues as before

The longer the path from a node to an absorbing node the more likely the random walk dies along the way, the lower the absorption probability



$$P(\text{Red}|\text{Green}) = (1 - \alpha) \left(\frac{1}{5} P(\text{Red}|\text{Yellow}) + \frac{1}{5} P(\text{Red}|\text{Pink}) + \frac{1}{5} \right)$$

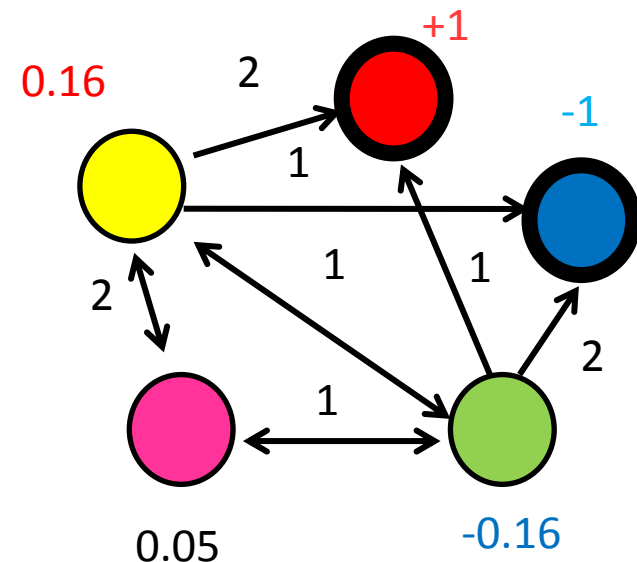
Propagating values

- Assume that **Red** has a positive value and **Blue** a negative value
 - Positive/Negative **class**, Positive/Negative **opinion**
- We can compute a value for all the other nodes in the same way
 - This is the **expected** value for the node

$$V(\text{Pink}) = \frac{2}{3}V(\text{Yellow}) + \frac{1}{3}V(\text{Green})$$

$$V(\text{Green}) = \frac{1}{5}V(\text{Yellow}) + \frac{1}{5}V(\text{Pink}) + \frac{1}{5} - \frac{2}{5}$$

$$V(\text{Yellow}) = \frac{1}{6}V(\text{Green}) + \frac{1}{3}V(\text{Pink}) + \frac{1}{3} - \frac{1}{6}$$



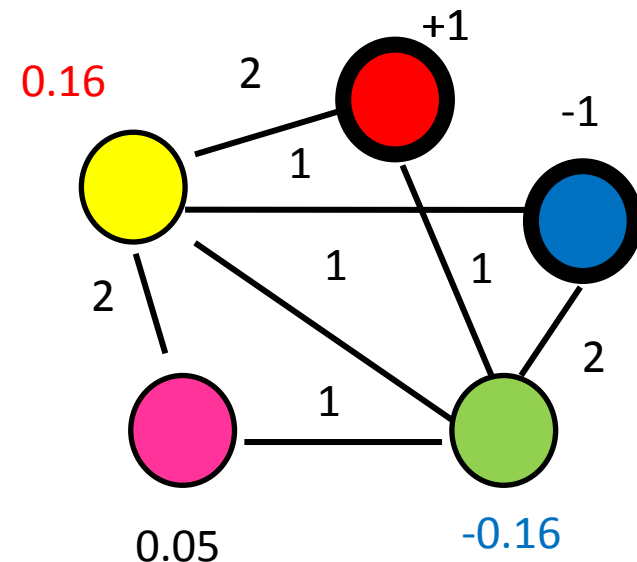
Electrical networks and random walks

- Our graph corresponds to an **electrical network**
- There is a positive **voltage** of **+1** at the Red node, and a negative voltage **-1** at the Blue node
- There are **resistances** on the edges **inversely proportional** to the weights (or **conductance proportional** to the weights)
- The computed values are the **voltages** at the nodes

$$V(\text{Pink}) = \frac{2}{3}V(\text{Yellow}) + \frac{1}{3}V(\text{Green})$$

$$V(\text{Green}) = \frac{1}{5}V(\text{Yellow}) + \frac{1}{5}V(\text{Pink}) + \frac{1}{5} - \frac{2}{5}$$

$$V(\text{Yellow}) = \frac{1}{6}V(\text{Green}) + \frac{1}{3}V(\text{Pink}) + \frac{1}{3} - \frac{1}{6}$$



Opinion formation

- The value propagation can be used as a model of opinion formation.
- Model:
 - Opinions are **values** in $[-1,1]$
 - Every user u has an **internal opinion** s_u , and **expressed opinion** z_u .
 - The expressed opinion **minimizes** the **personal cost** of user u :

$$c(z_u) = (s_u - z_u)^2 + \sum_{v:v \text{ is a friend of } u} w_u (z_u - z_v)^2$$

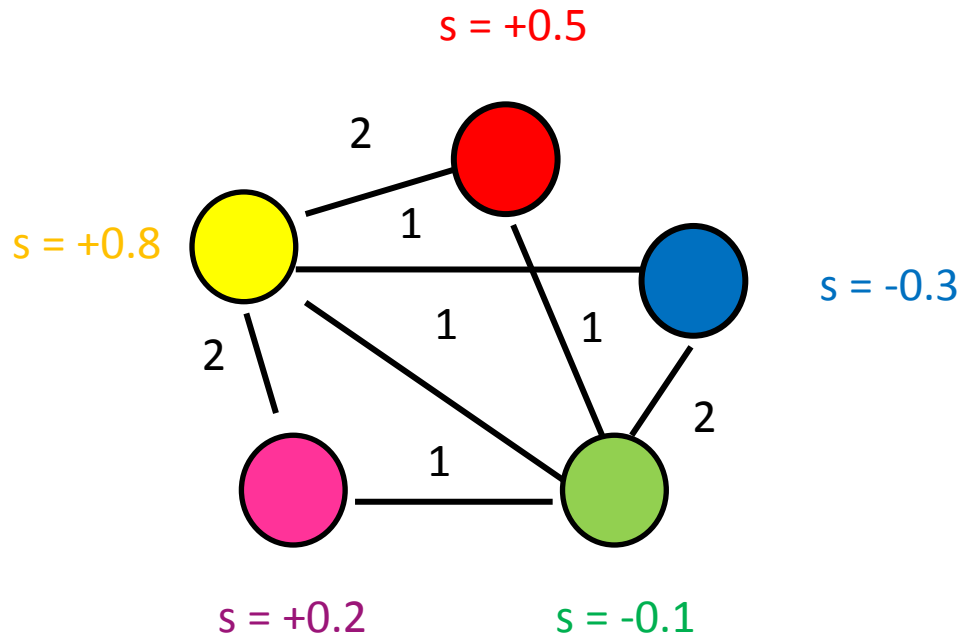
- Minimize deviation from your beliefs and conflicts with the society
- If every user tries **independently (selfishly)** to minimize their personal cost then the best thing to do is to set z_u to the **average** of all opinions:

$$z_u = \frac{s_u + \sum_{v:v \text{ is a friend of } u} w_u z_v}{1 + \sum_{v:v \text{ is a friend of } u} w_u}$$

- This is the same as the value propagation we described before!

Example

- Social network with **internal opinions**



Example

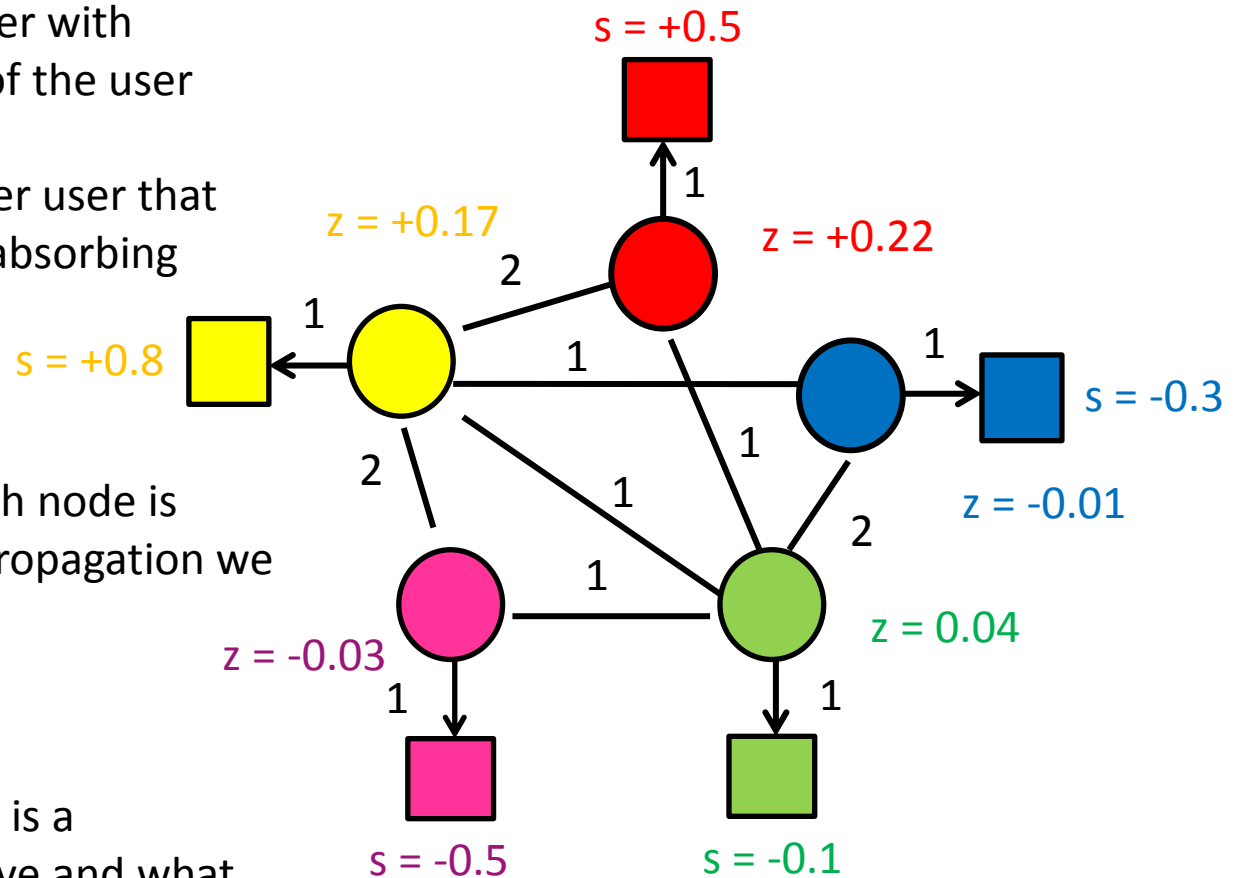
One absorbing node per user with value the **internal opinion** of the user

One non-absorbing node per user that links to the corresponding absorbing node

The **external opinion** for each node is computed using the value propagation we described before

- Repeated averaging

Intuitive model: my opinion is a combination of what I believe and what my social network believes.



Transductive learning

- If we have a graph of relationships and some **labels** on some nodes we can **propagate** them to the remaining nodes
 - Make the labeled nodes to be absorbing and compute the probability for the rest of the graph
 - E.g., a social network where some people are tagged as spammers
 - E.g., the movie-actor graph where some movies are tagged as action or comedy.
- This is a form of **semi-supervised learning**
 - We make use of the unlabeled data, and the relationships
- It is also called **transductive learning** because it does not produce a model, but just labels the unlabeled data that is at hand.
 - Contrast to **inductive learning** that learns a model and can label any new example

Implementation details

- Implementation is in many ways similar to the PageRank implementation
 - For an edge (u, v) instead of updating the value of v we update the value of u .
 - The value of a node is the average of its neighbors
 - We need to check for the case that a node u is absorbing, in which case the value of the node is not updated.
 - Repeat the updates until the change in values is very small.