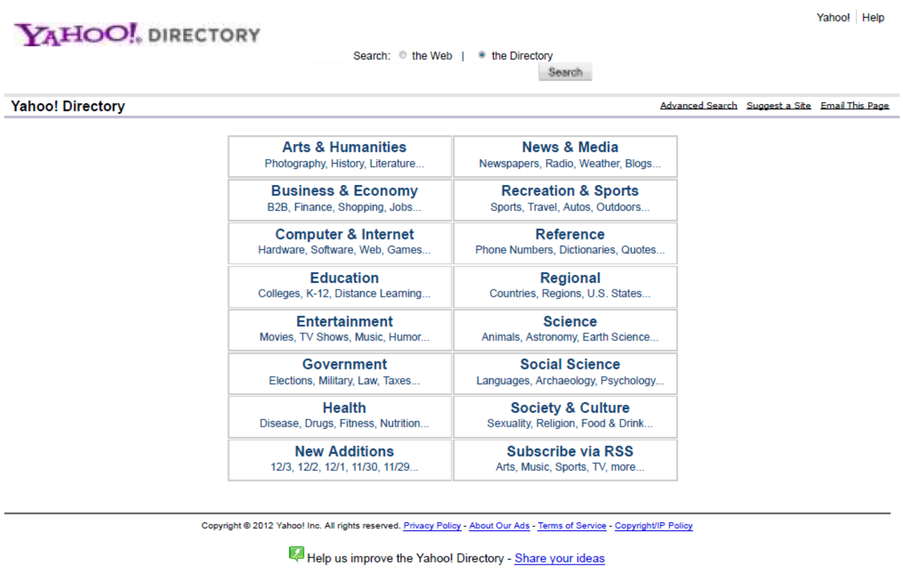
Online Social Networks and Media

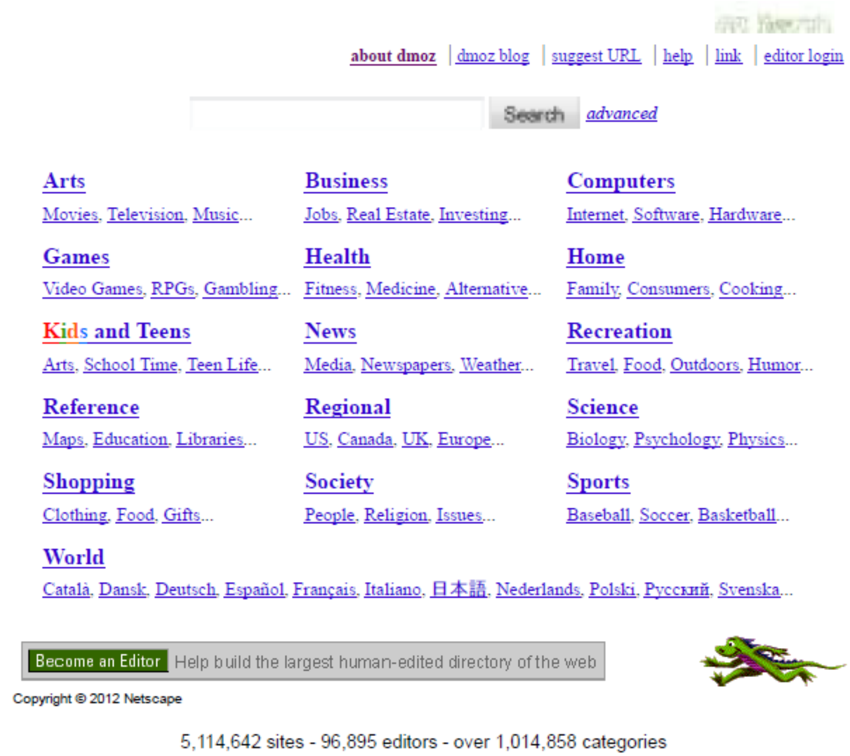
Link Analysis

How to Organize the Web

First try: Human curated Web directories

Yahoo, DMOZ, LookSmart





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

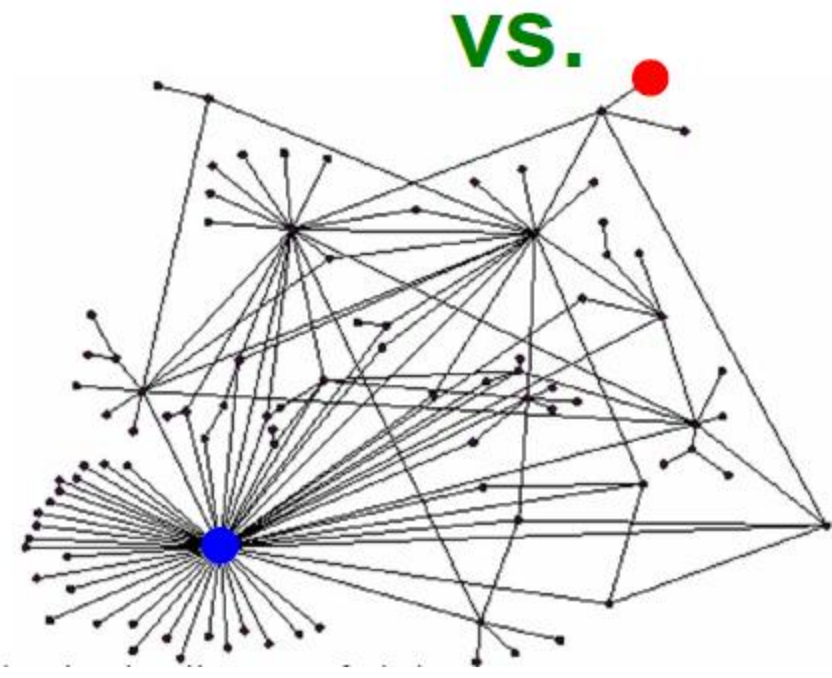
How to organize the web

- Third try (the Google era): using the web graph
  - Shift 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 “covid19”?

Link Analysis

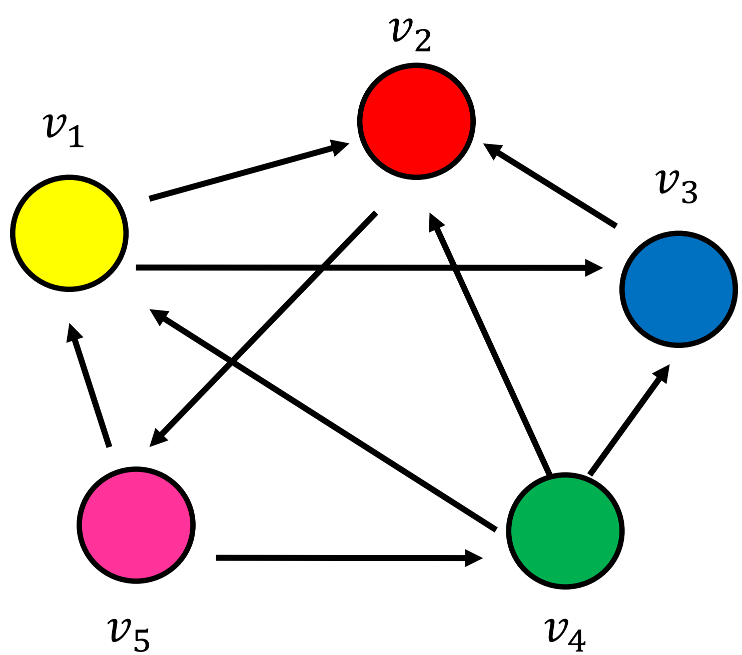
- Not all web pages are created equal on the web
- The links act as endorsements:
  - When page *p* links to *q* it endorses the content of the content of *q*

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



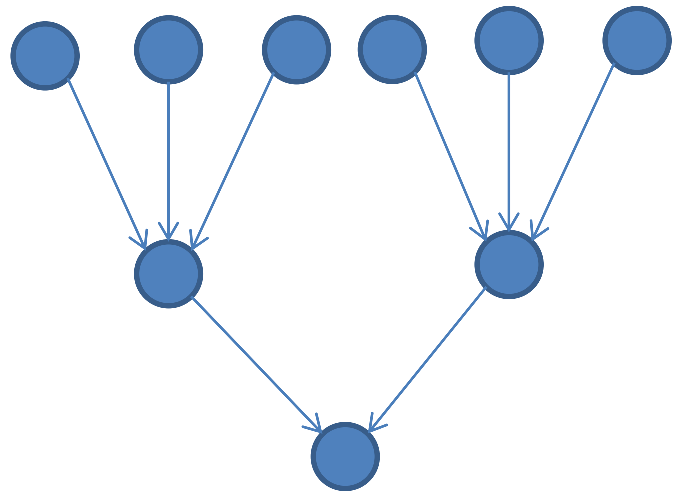
Rank by Popularity

- 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.
  - Node i gets a fraction w_i of that authority weight
  - 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.

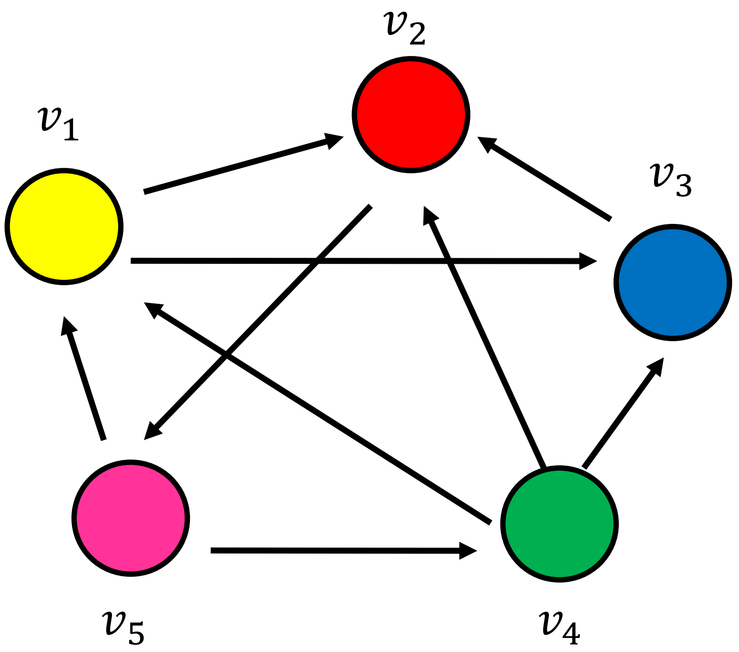
$$w_i = \sum_{j \to i} \frac{1}{|N_{out}(j)|} w_j$$

w_v: the PageRank value of node v

Recursive definition

An example

$$w_i = \sum_{j \to i} \frac{1}{|N_{out}(j)|} w_j$$



$w_1 = 1/3\ w_4 + 1/2\ w_5$

$w_2 = 1/2\ w_1 + w_3 + 1/3\ w_4$

$w_3 = 1/2\ w_1 + 1/3\ w_4$

$w_4 = 1/2\ w_5$

$w_5 = w_2$

$w_1 + w_2 + w_3 + w_4 + w_5 = 1$

We can obtain the weights by solving this system of equations

Computing PageRank weights

- A simpler way to compute the weights is by iteratively updating the weights using the equations
- PageRank Algorithm

Initialize all PageRank weights to $w_i^0 = \frac{1}{n}$

Repeat:

$$w_i^t = \sum_{j \to i} \frac{1}{|N_{out}(j)|} w_j^{t-1}$$

Until the weights do not change

- This process converges

Example

$$w_i = \sum_{j \to i} \frac{1}{|N_{out}(j)|} w_j$$

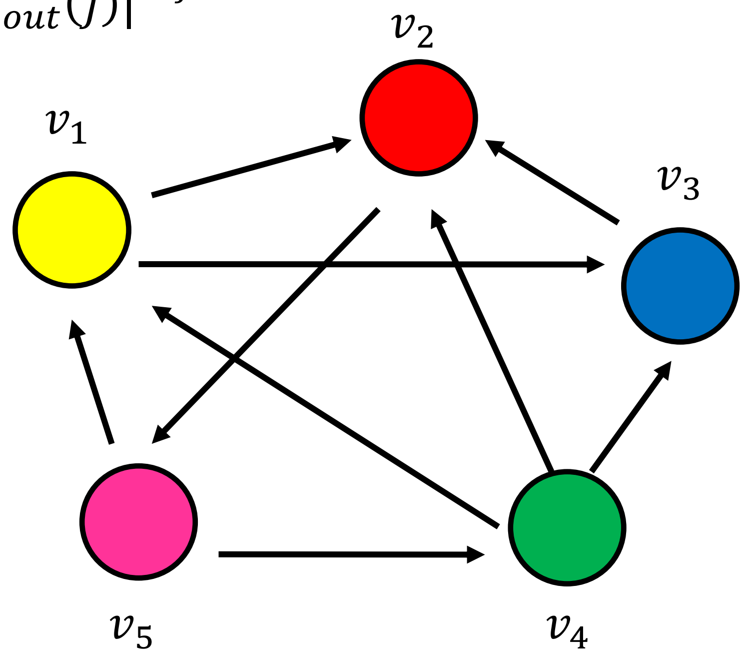
$w_1 = 1/3\ w_4 + 1/2\ w_5$

$w_2 = 1/2\ w_1 + w_3 + 1/3\ w_4$

$w_3 = 1/2\ w_1 + 1/3\ w_4$

$w_4 = 1/2\ w_5$

$w_5 = w_2$



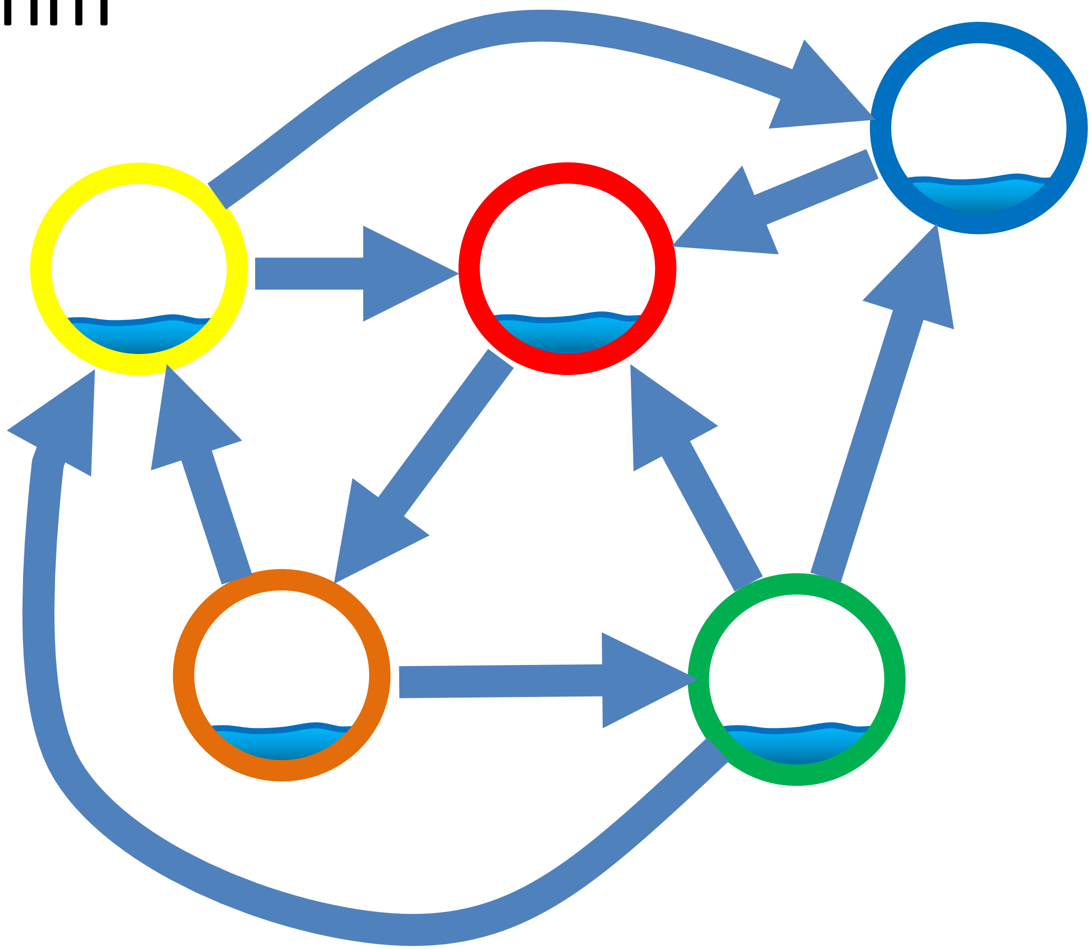
	w_1	w_2	w_3	w_4	w_5
t=0	0.2	0.2	0.2	0.2	0.2
t=1	0.16	0.36	0.16	0.1	0.2
t=2	0.13	0.28	0.11	0.1	0.36
t=3	0.22	0.22	0.1	0.18	0.28
t=4	0.2	0.27	0.17	0.14	0.22

Think of the weight as a fluid: there is constant amount of it in the graph, but it moves around until it stabilizes

The PageRank algorithm

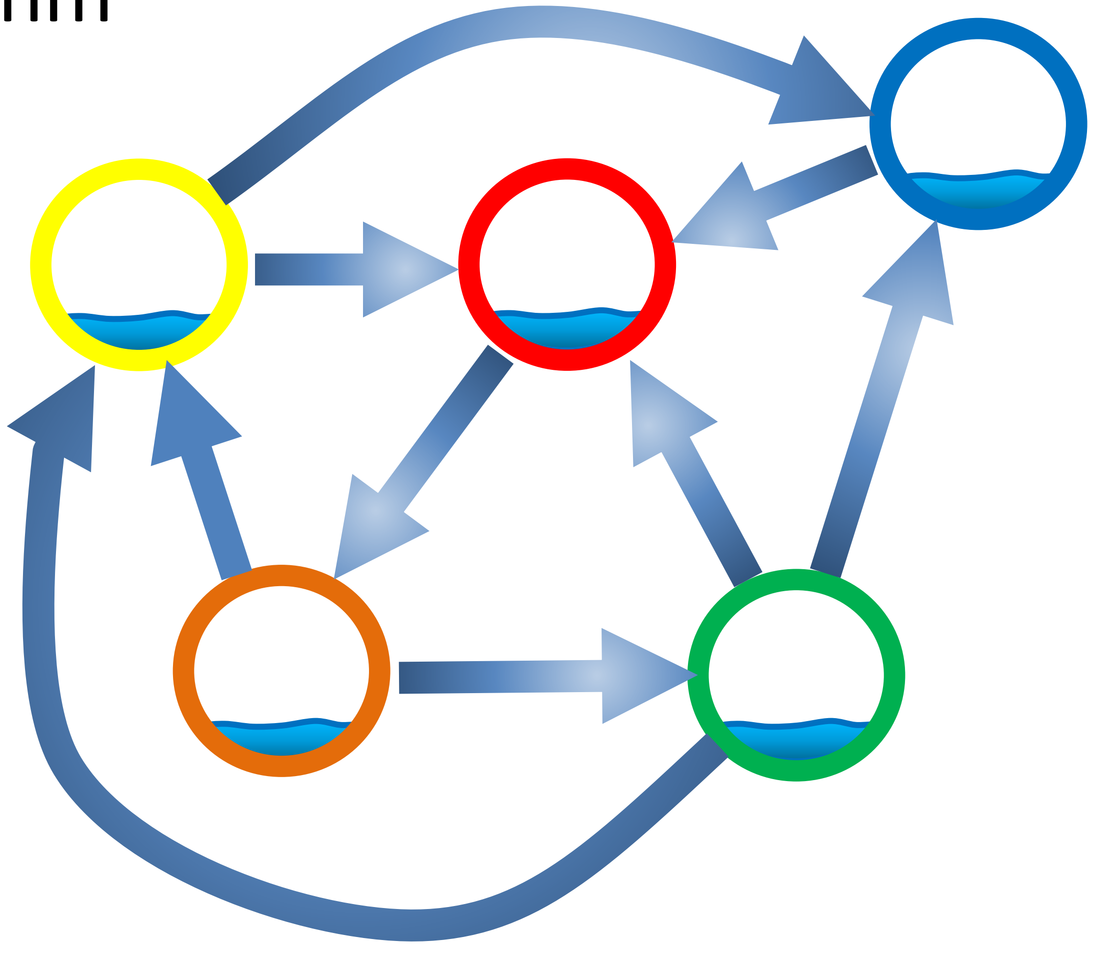
Think of the nodes in the graph as containers of capacity of 1 liter.

We distribute a liter of liquid equally to all containers



The PageRank algorithm

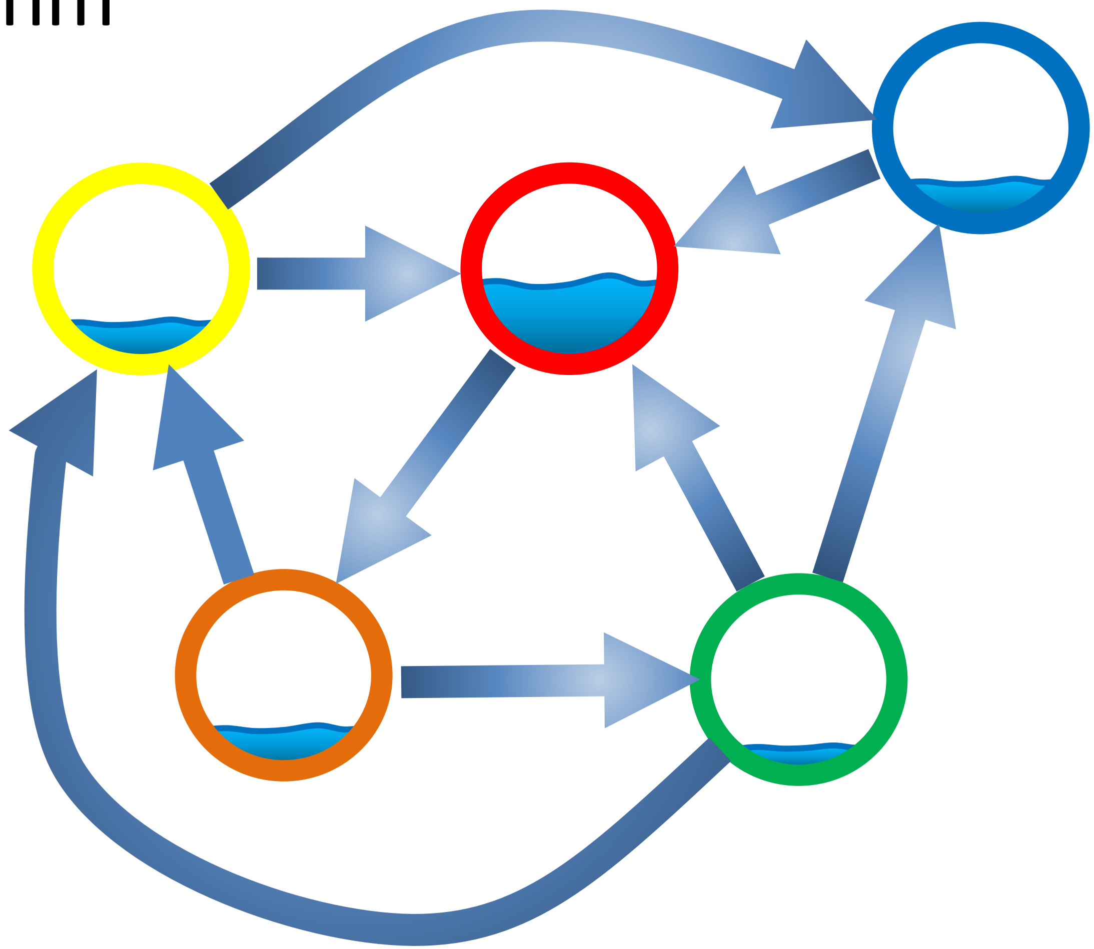
The edges act like pipes that transfer liquid between nodes.



The PageRank algorithm

The edges act like pipes that transfer liquid between nodes.

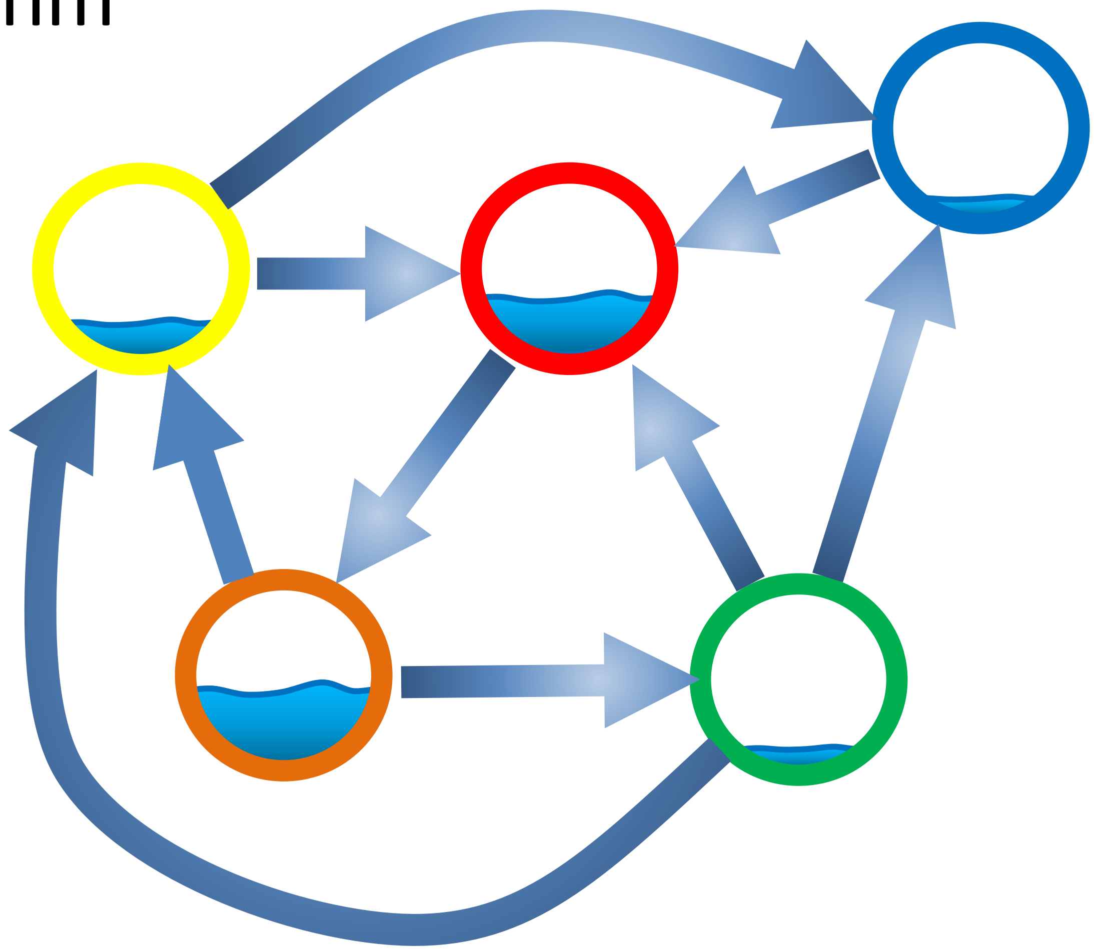
The contents of each node are distributed to its neighbors.



The PageRank algorithm

The edges act like pipes that transfer liquid between nodes.

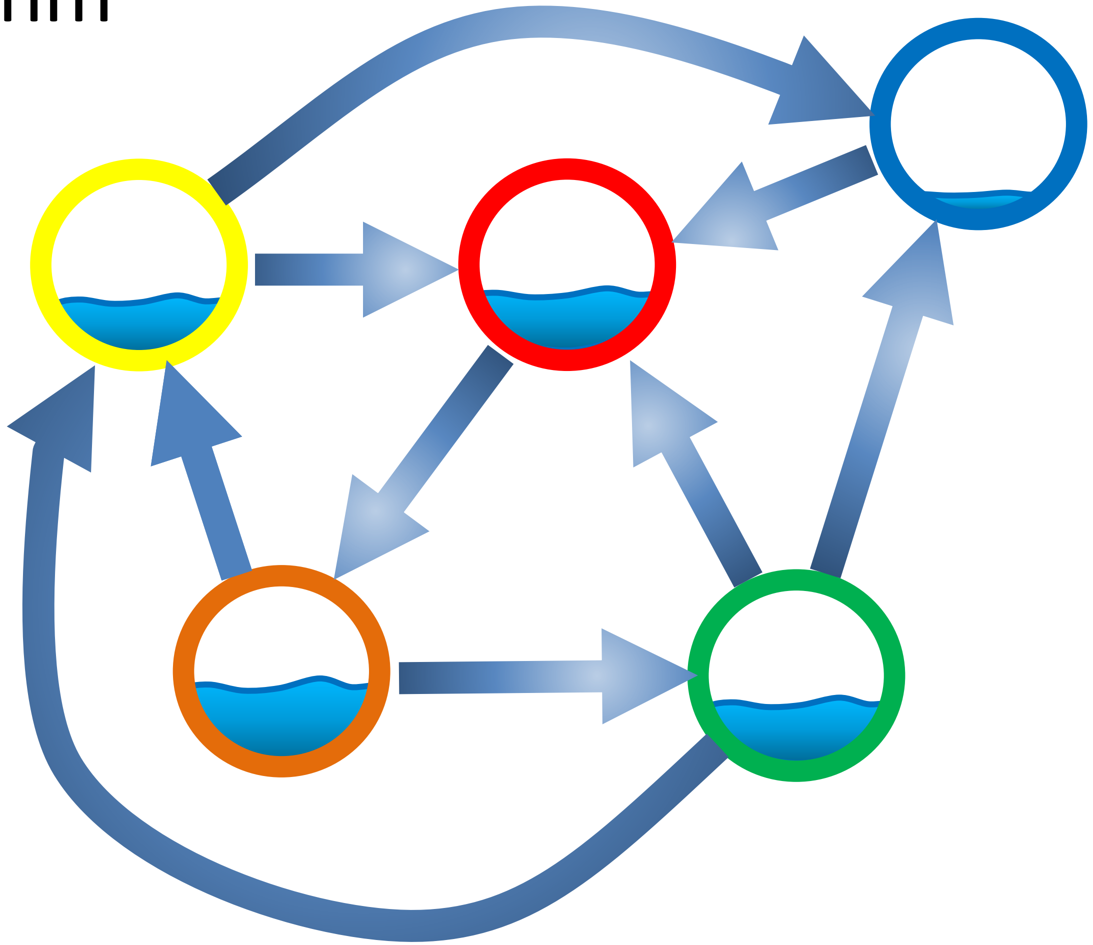
The contents of each node are distributed to its neighbors.



The PageRank algorithm

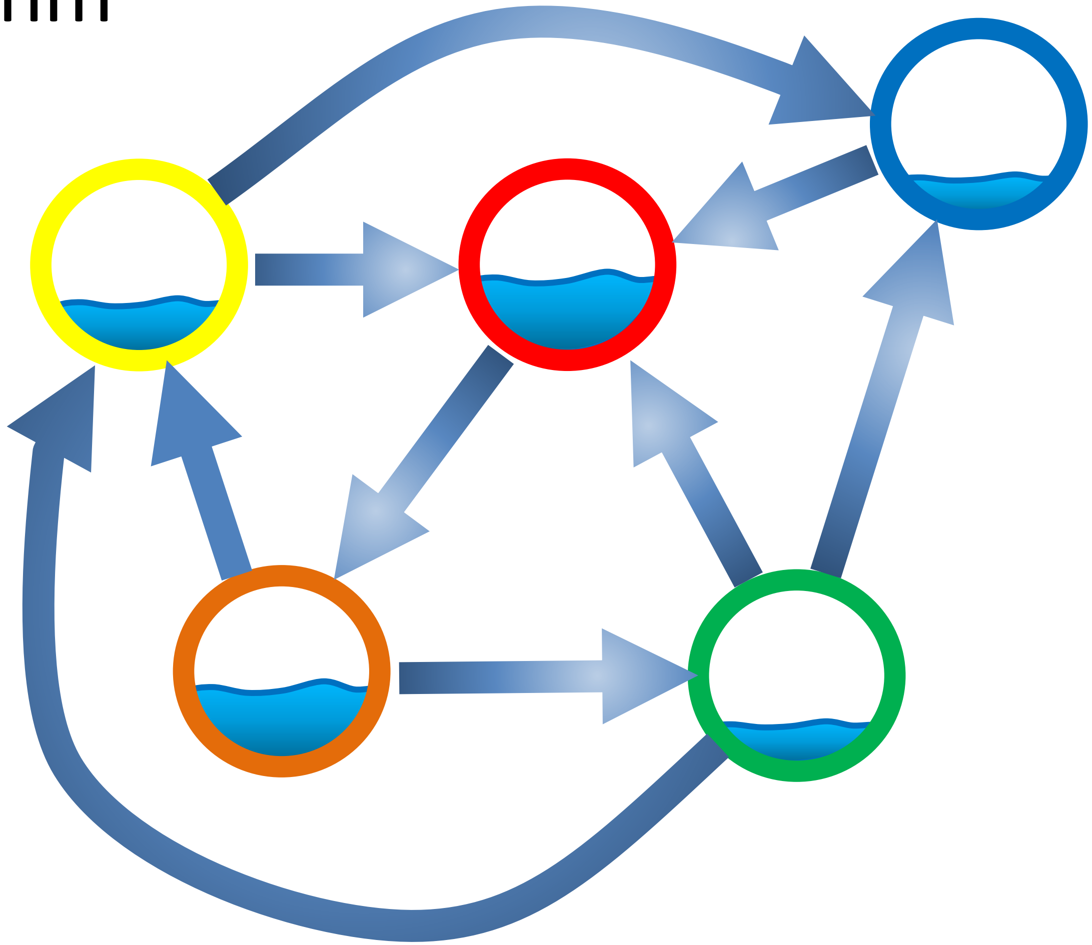
The edges act like pipes that transfer liquid between nodes.

The contents of each node are distributed to its neighbors.



The PageRank algorithm

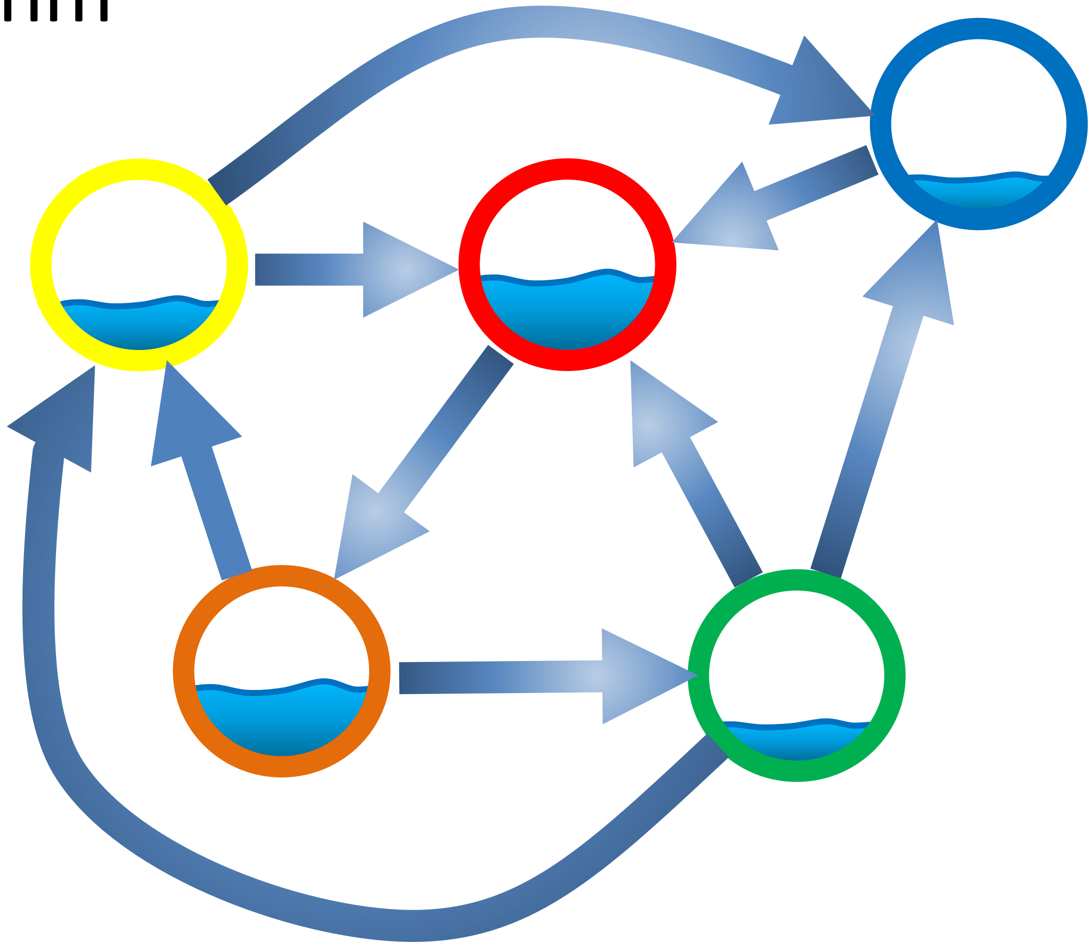
The system will reach an equilibrium state where the amount of liquid in each node remains constant.



The PageRank algorithm

The amount of liquid in each node determines the importance of the node.

Large quantity means large incoming flow from nodes with large quantity of liquid.



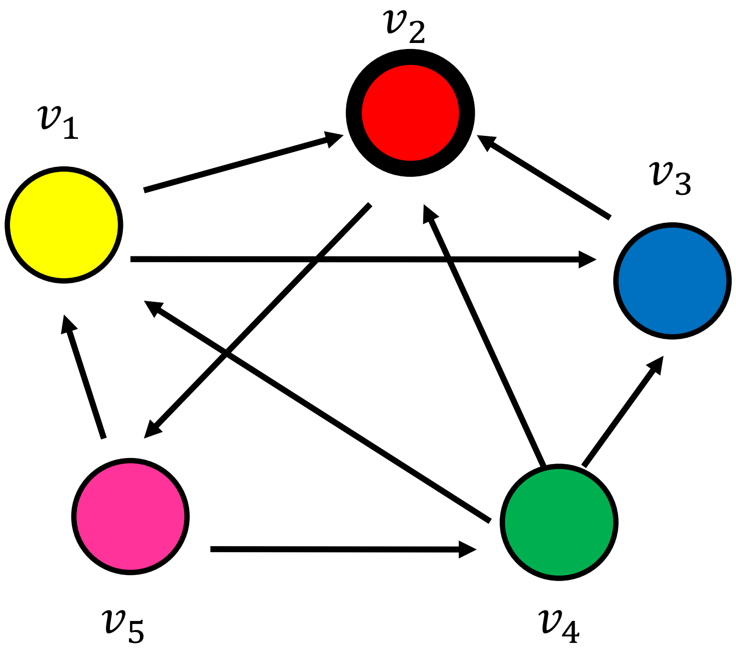
Random Walks on Graphs

- 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.

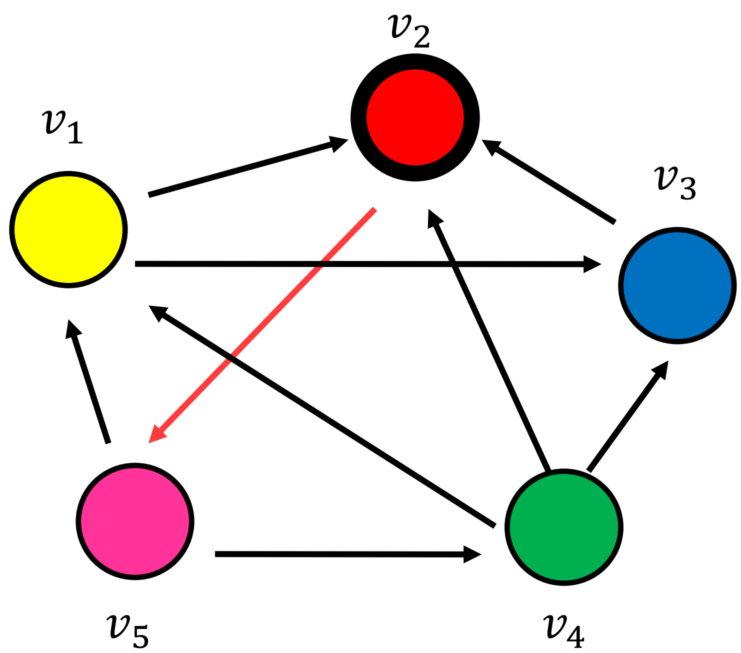
Example

- Step 0



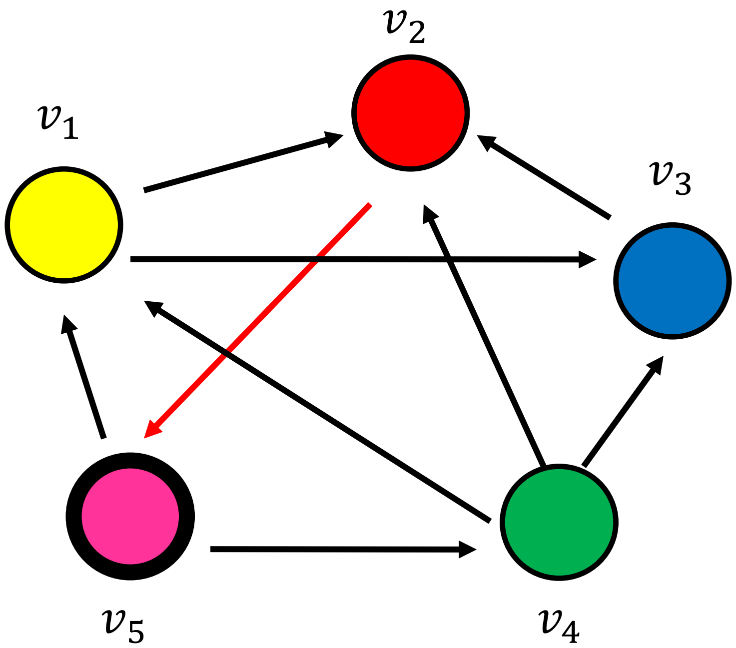
Example

- Step 0



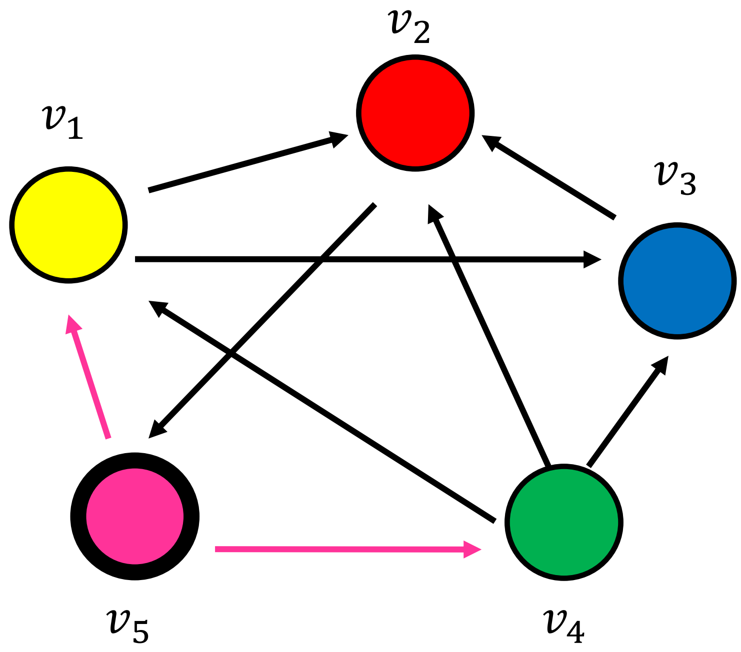
Example

- Step 1



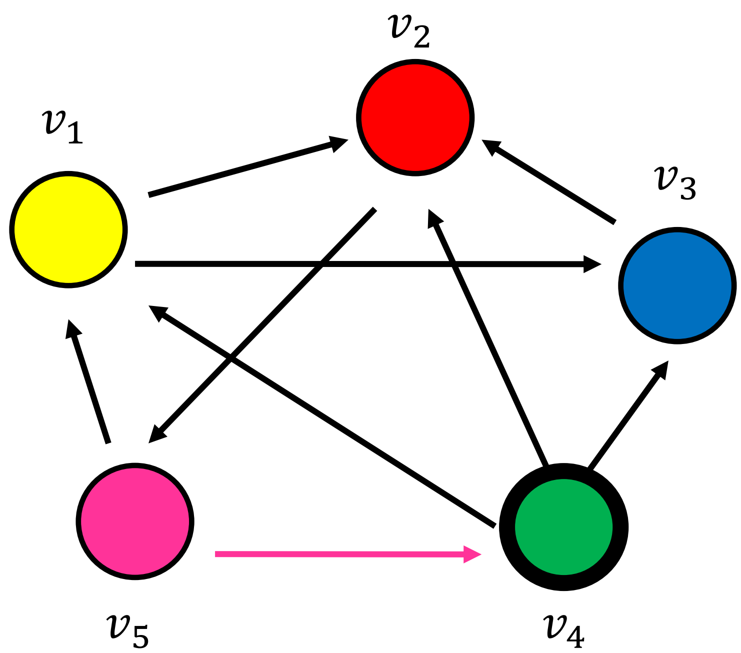
Example

- Step 1



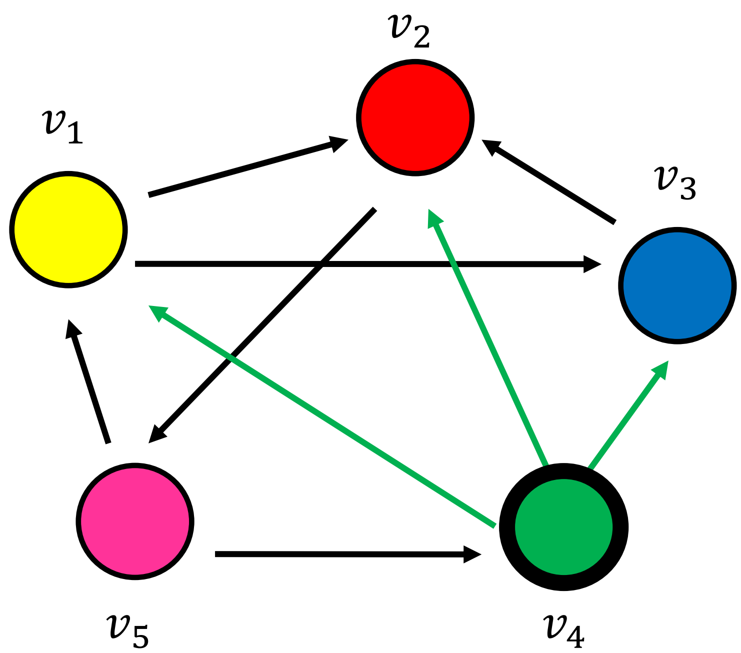
Example

- Step 2



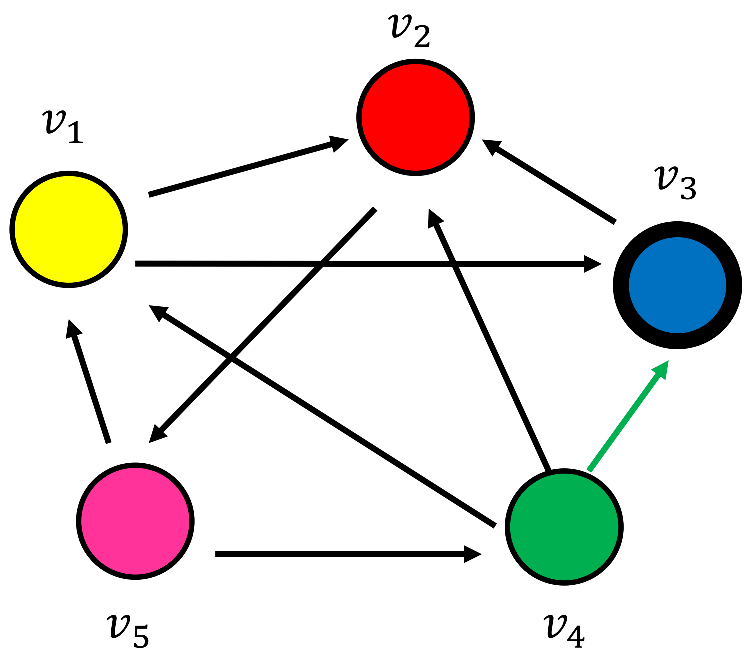
Example

- Step 2



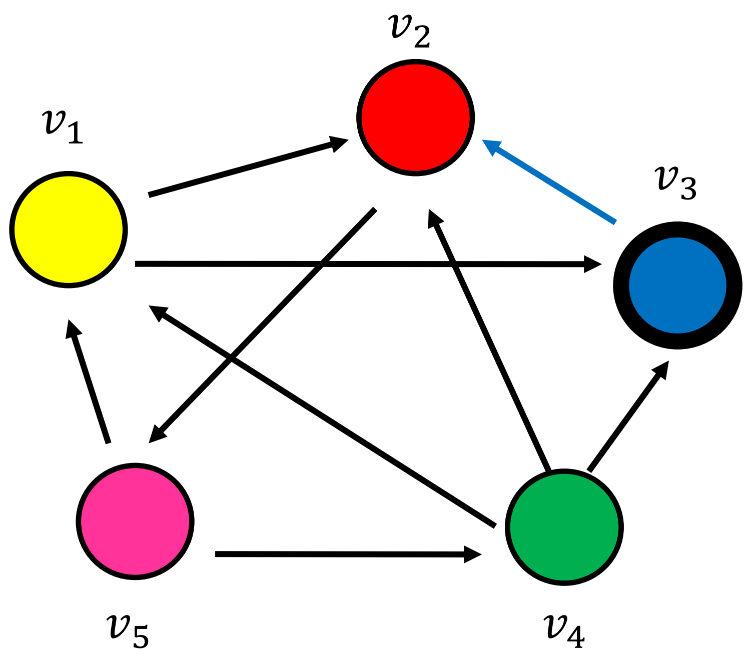
Example

- Step 3



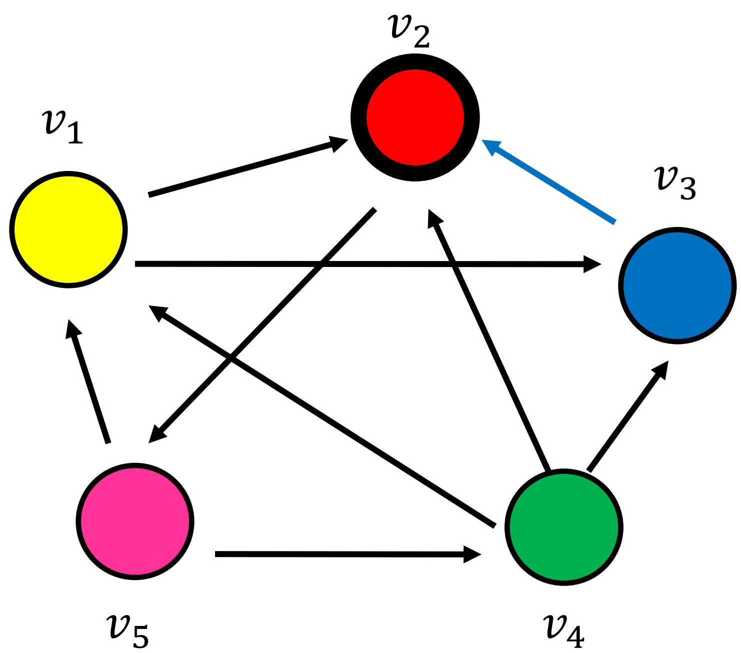
Example

- Step 3



Example

- Step 4…



Random walk

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

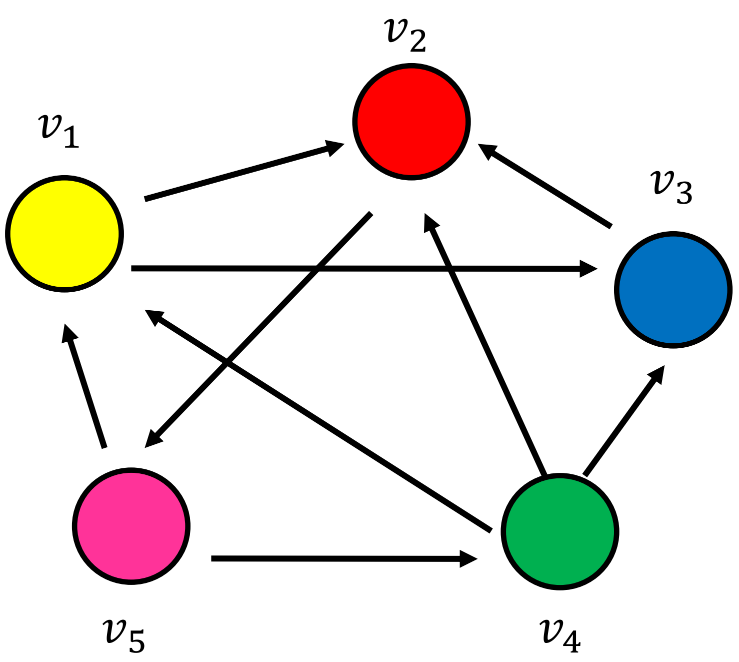
$$p_1^0 = \frac{1}{5} \qquad p_1^t = \frac{1}{3} p_4^{t-1} + \frac{1}{2} p_5^{t-1}$$

$$p_2^0 = \frac{1}{5} \qquad p_2^t = \frac{1}{2} p_1^{t-1} + p_3^{t-1} + \frac{1}{3} p_4^{t-1}$$

$$p_3^0 = \frac{1}{5} \qquad p_3^t = \frac{1}{2} p_1^{t-1} + \frac{1}{3} p_4^{t-1}$$

$$p_4^0 = \frac{1}{5} \qquad p_4^t = \frac{1}{2} p_5^{t-1}$$

$$p_5^0 = \frac{1}{5} \qquad p_5^t = p_2^{t-1}$$



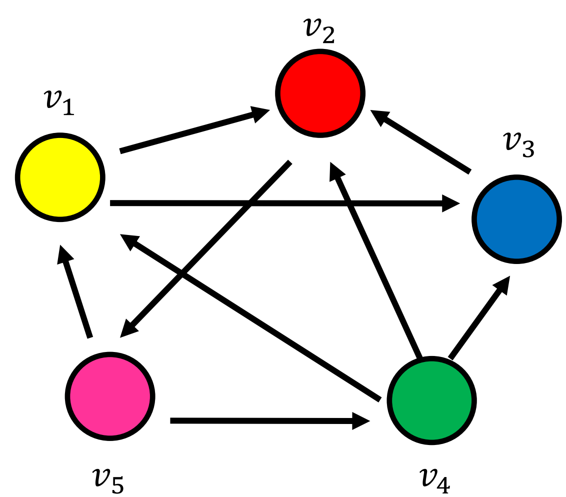
$$p_i^t = \sum_{j \to i} \frac{1}{|N_{out}(j)|} p_j^{t-1}$$

The equations are the same as those for the PageRank iterative computation

Random walk

- At convergence:

$$p_i = \sum_{j \to i} \frac{1}{|N_{out}(j)|} p_j$$



We get the same equation as for PageRank

The PageRank of node i is the probability that the random walk is at node i after a very large (infinite) number of steps

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 from state i to state j

- 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

Markov chains

- The stochastic process proceeds in steps and moves between the states:
  - State probability distribution: The vector $p^t = (p_1^t, p_2^t, \dots, p_n^t)$ that stores the probability distribution 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
- We can compute the vector p^t at step t using a vector-matrix multiplication

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

Random walks

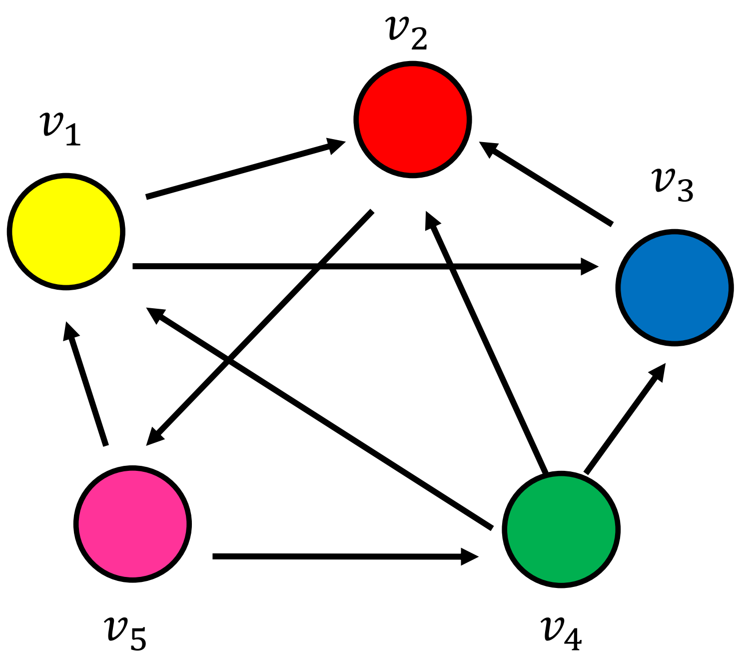
- Random walks on graphs correspond to Markov Chains
  - The set of states S is the set of nodes of the graph G
  - The transition probability matrix is the probability that we follow an edge from one node to another

$$P[i,j] = \frac{1}{\mathrm{d}_{out}(i)}$$

An example

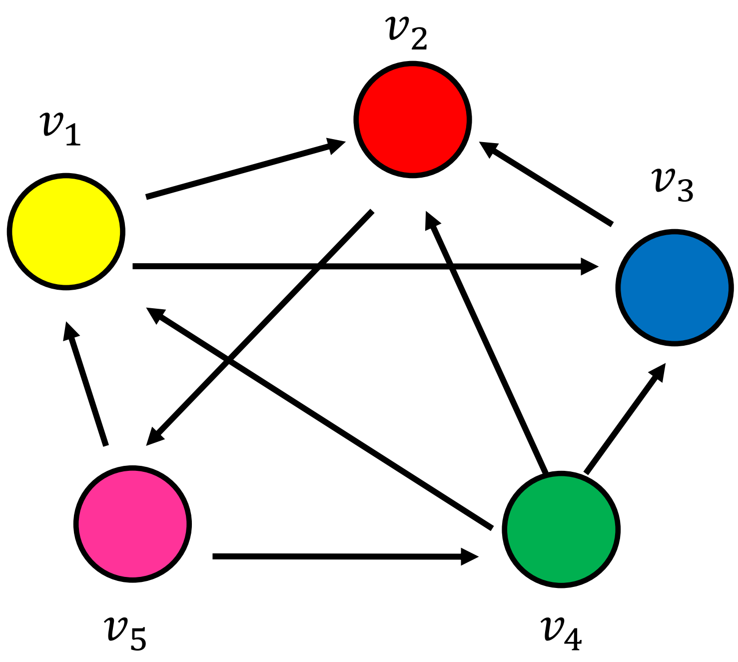
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

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



An example

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



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

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

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

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

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

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

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

- 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.
  - In our case these are the PageRank values.

Computing the stationary distribution

- The Power Method

> Initialize p^0 to some distribution
> Repeat
> $\quad p^t = p^{t-1}P$
> Until convergence

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

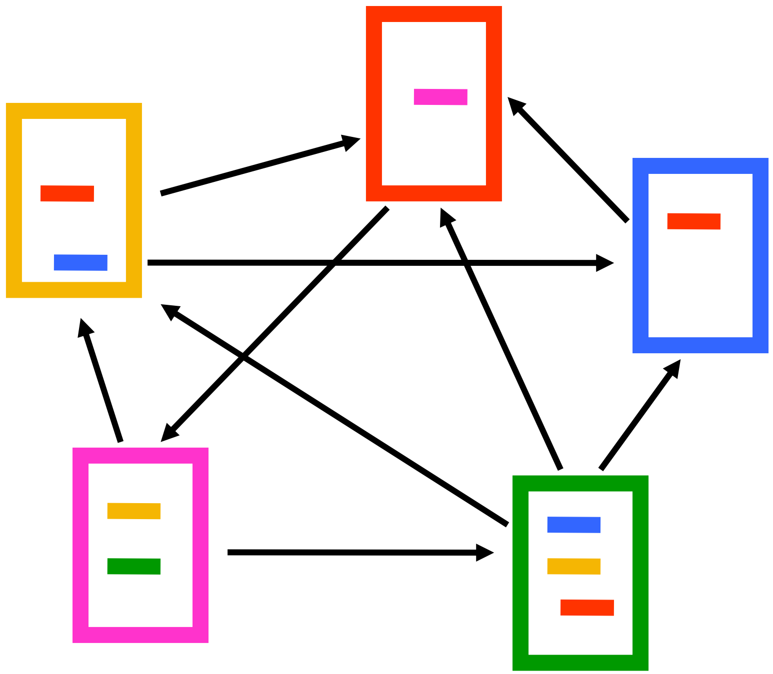
The stationary distribution

- π is the left eigenvector of transition matrix P
- $\pi(i)$: the probability of being at node i after very large (infinite) number of steps
- $\pi(i)$: the fraction of times that the random walk visited state i as $t \rightarrow \infty$
- $\pi = p_0 P^\infty$, 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 – same for all i, starting point does not matter.

The PageRank random walk

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

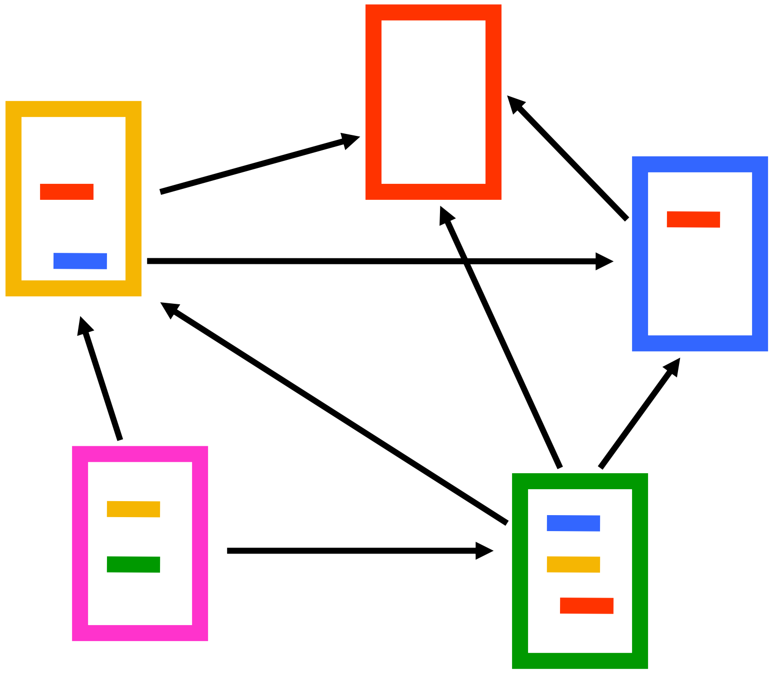
$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$



The PageRank random walk

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

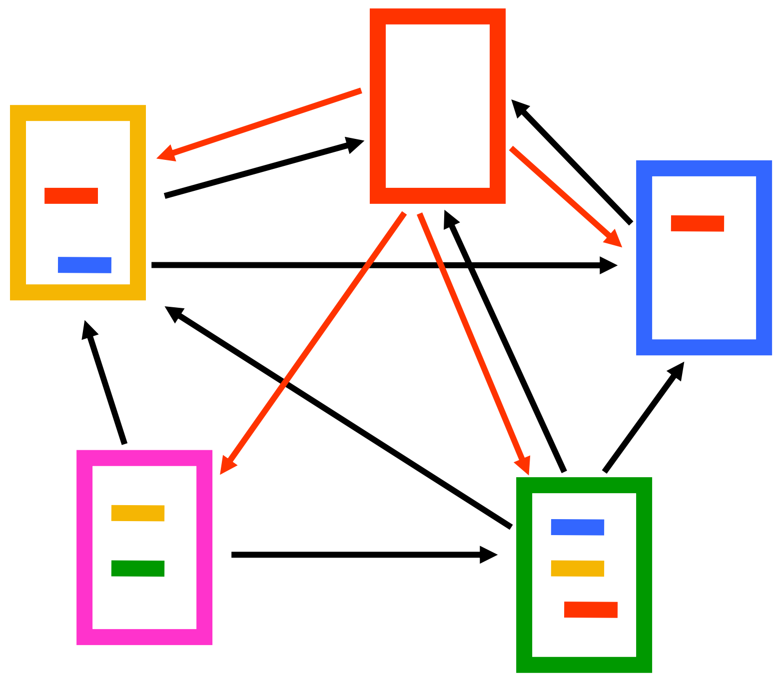
$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$



The PageRank random walk

- Replace these row vectors with a vector v
  - typically, the uniform vector

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



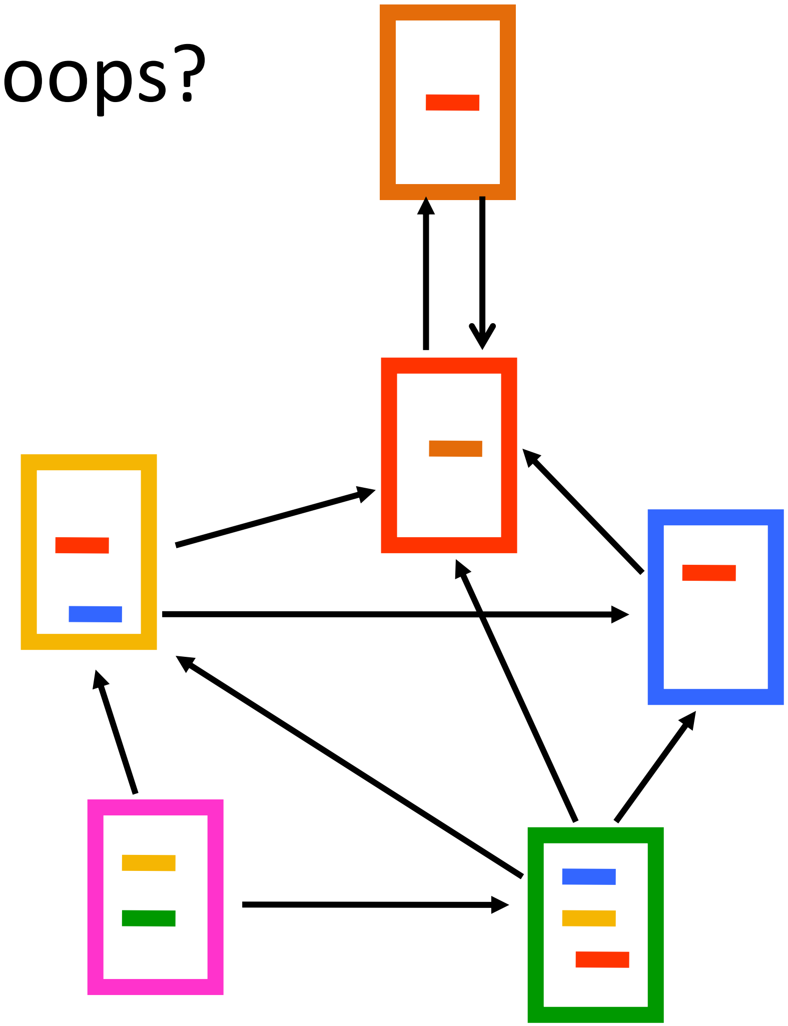
$P' = P + dv^T$

Outer product

$$d = \begin{cases} 1 & \text{if i is sink} \\ 0 & \text{otherwise} \end{cases}$$

The PageRank random walk

- What about loops?
  - Spider traps



The PageRank random walk

- Add a random jump to a node chosen according to the vector v with prob $1 - \alpha$
  - typically, to v is a uniform probability vector

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

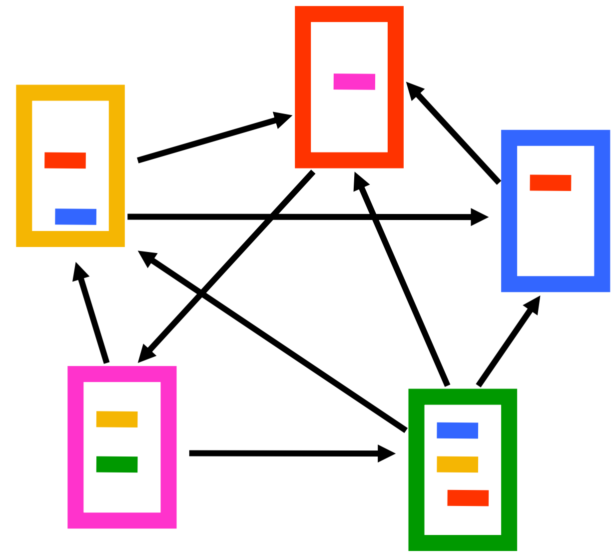
$P'' = \alpha P' + (1-\alpha) u v^T$, where u is the vector of all 1s

Random walk with restarts

PageRank algorithm [BP98]

- The Random Surfer model
  - pick a page at random
  - with probability α follow a random outgoing link
  - with probability $1 - \alpha$ jump to a random page
- Rank according to the stationary distribution
- $w_i = \alpha \sum_{j \to i} \frac{1}{|N_{out}(i)|} w_j + (1-\alpha) \frac{1}{n}$

  $\alpha = 0.85$ in most cases
- We repeat this computation until convergence



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

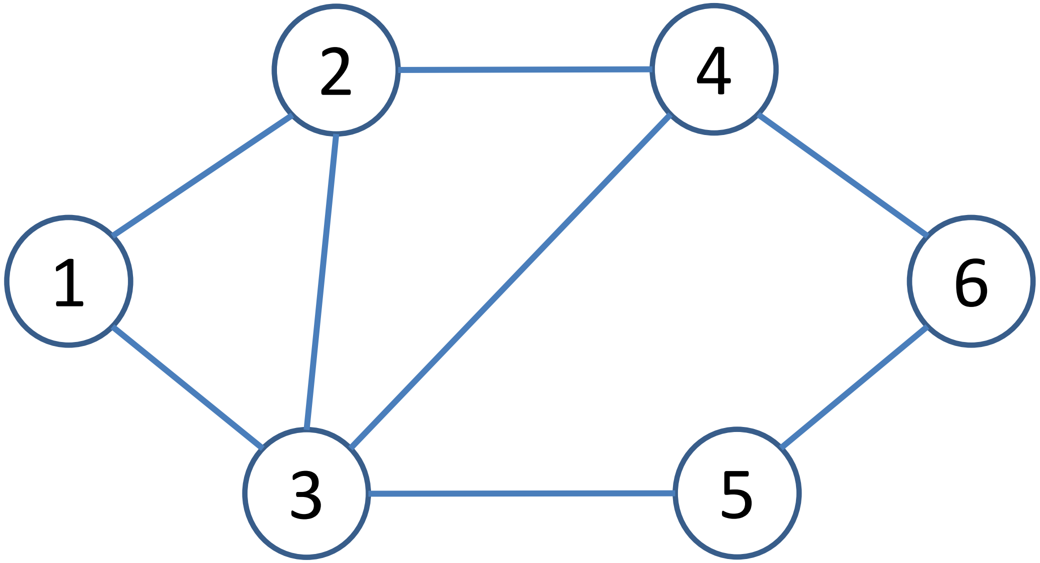
Stationary distribution with random jump

- If v is the jump vector
  - $p^0 = v$
  - $p^1 = \alpha p^0 P' + (1-\alpha)v = \alpha v P' + (1-\alpha)v$
  - $p^2 = \alpha p^1 P' + (1-\alpha)v = \alpha^2 v P'^2 + (1-\alpha)v\alpha P' + (1-\alpha)v$
  - $\vdots$
  - $p^\infty = (1-\alpha)v + (1-\alpha)v\alpha P' + (1-\alpha)v\alpha^2 P'^2 + \cdots = (1-\alpha)v(I-\alpha P')^{-1}$
- Explanation: From the last step trace the last restart :
  - With probability $1-\alpha$ you just restarted in the last step
  - With probability $\alpha(1-\alpha)$ you restarted one step before and then did a random walk step
  - With probability $\alpha^2(1-\alpha)$ you restarted two steps before and then did two random walk steps
  - Etc…
- Conclusion: you will not walk very far
- With the random jump the shorter paths are more important, since the weight decreases exponentially
  - makes sense when thought of as a restart

Random walks with restarts

- If v is not uniform, we can bias the random walk towards the nodes that are close to v
- Personalized PageRank:
  - Restart the random walk from a specific node x
  - All nodes are ranked according to their closeness to x
- Topic-Specific Pagerank.
  - Restart the random walk from a specific set of nodes (e.g., nodes about a topic)
  - All nodes are ranked according to their closeness to the topic.
- Random Walks with restarts is a general technique for measuring closeness on graphs.

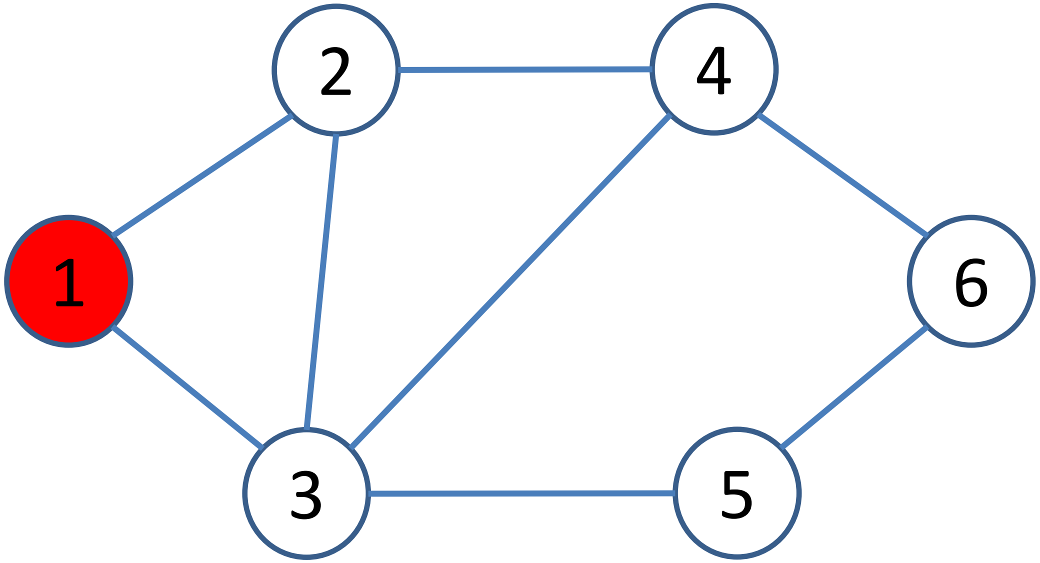
Personalized Pagerank Example



- Global Pagerank vector (jump vector $[\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}]$)

  [0.13, 0.18, 0.24, 0.18, 0.13, 0.13]

Personalized Pagerank Example



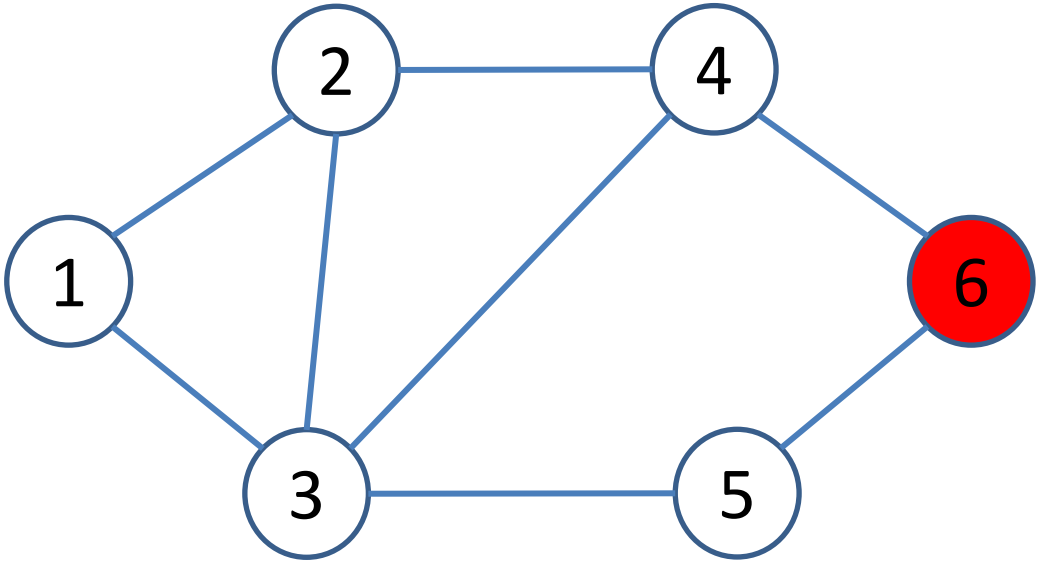
- Global Pagerank vector (jump vector $[\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6}]$):

  [0.13, 0.18, 0.24, 0.18, 0.13, 0.13]

- Personalized Pagerank for node 1 (jump vector [1,0,0,0,0,0]):

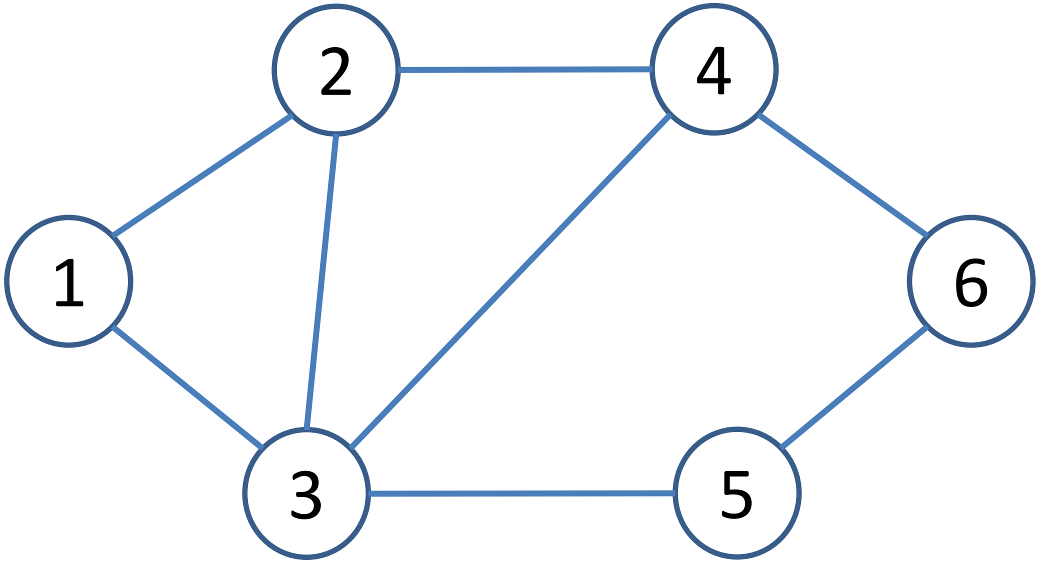
  [0.26, 0.20, 0.24, 0.14, 0.08, 0.07]

Personalized Pagerank Example



- Global Pagerank vector (jump vector $[\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6},\frac{1}{6}]$):
  [0.13, 0.18, 0.24, 0.18, 0.13, 0.13]
- Personalized Pagerank from node 1 (jump vector [1,0,0,0,0,0]):
  [0.26, 0.20, 0.24, 0.14, 0.08, 0.07]
- Personalized Pagerank from node 6 (jump vector [0,0,0,0,0,1]):
  [0.07, 0.13, 0.19, 0.19, 0.15, 0.27]

Personalized Pagerank Example



With $a = 0.5$

- Global Pagerank vector (jump vector $[\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}]$):
  [0.14, 0.17, 0.21, 0.18, 0.15, 0.15]
- Personalized Pagerank from node 1 (jump vector [1,0,0,0,0,0]):
  [0.55, 0.17, 0.18, 0.05, 0.03, 0.02]
- Personalized Pagerank from node 6 (jump vector [0,0,0,0,0,1]):
  [0.02, 0.04, 0.07, 0.16, 0.15, 0.56]

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 of a random walk 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.
  - Random walks with restarts to a single node are commonly used on undirected graphs for measuring similarity between nodes

PageRank implementation

- Store the graph as a 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 between the pagerank vectors (L_1 or L_∞ difference) is below some small value ε.

A (Matlab/Numpy-friendly) PageRank algorithm

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

$q^0 = v$

$t = 1$

repeat

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

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

$\quad t = t + 1$

until $\delta < \varepsilon$

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.

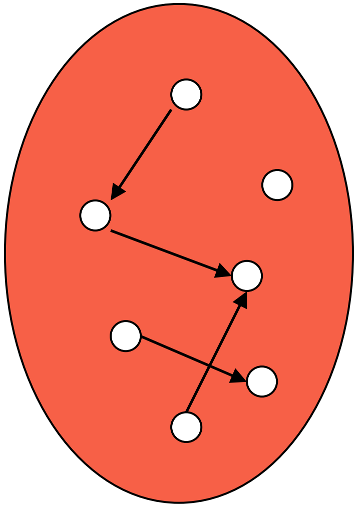
THE HITS ALGORITHM

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

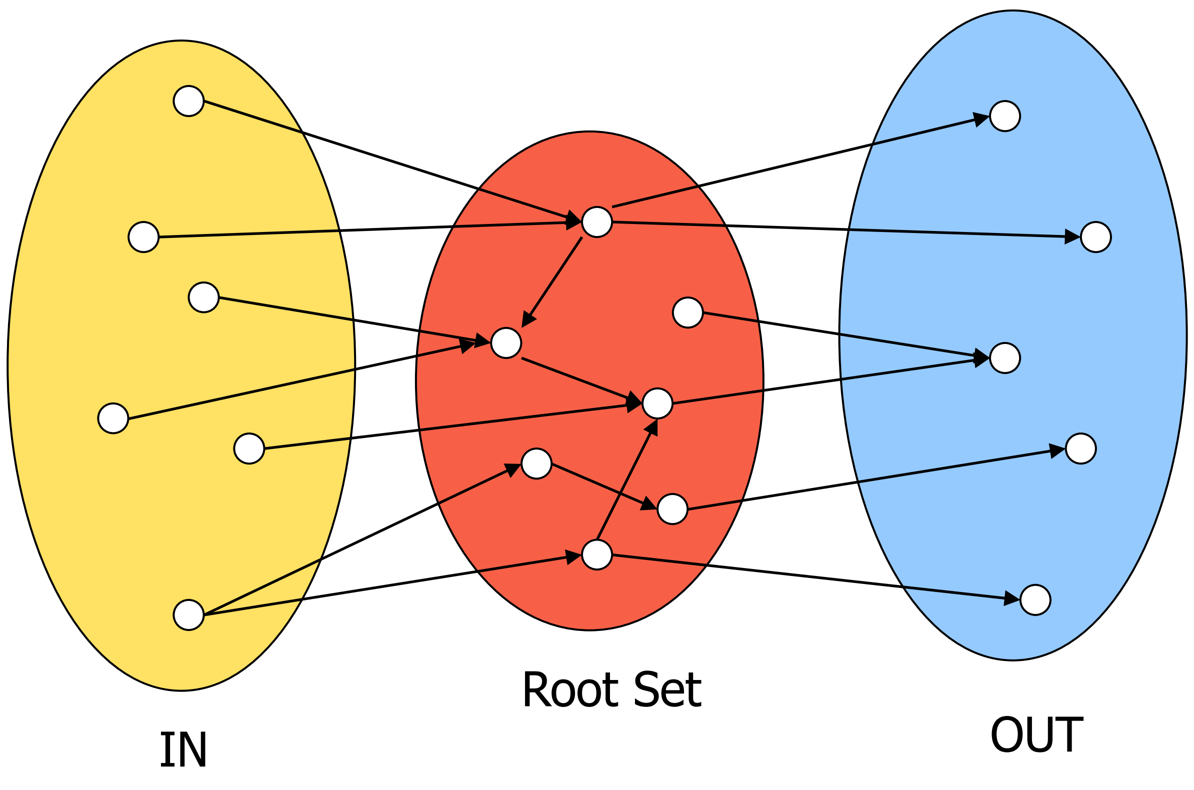
Query dependent input

Root set obtained from a text-only search engine

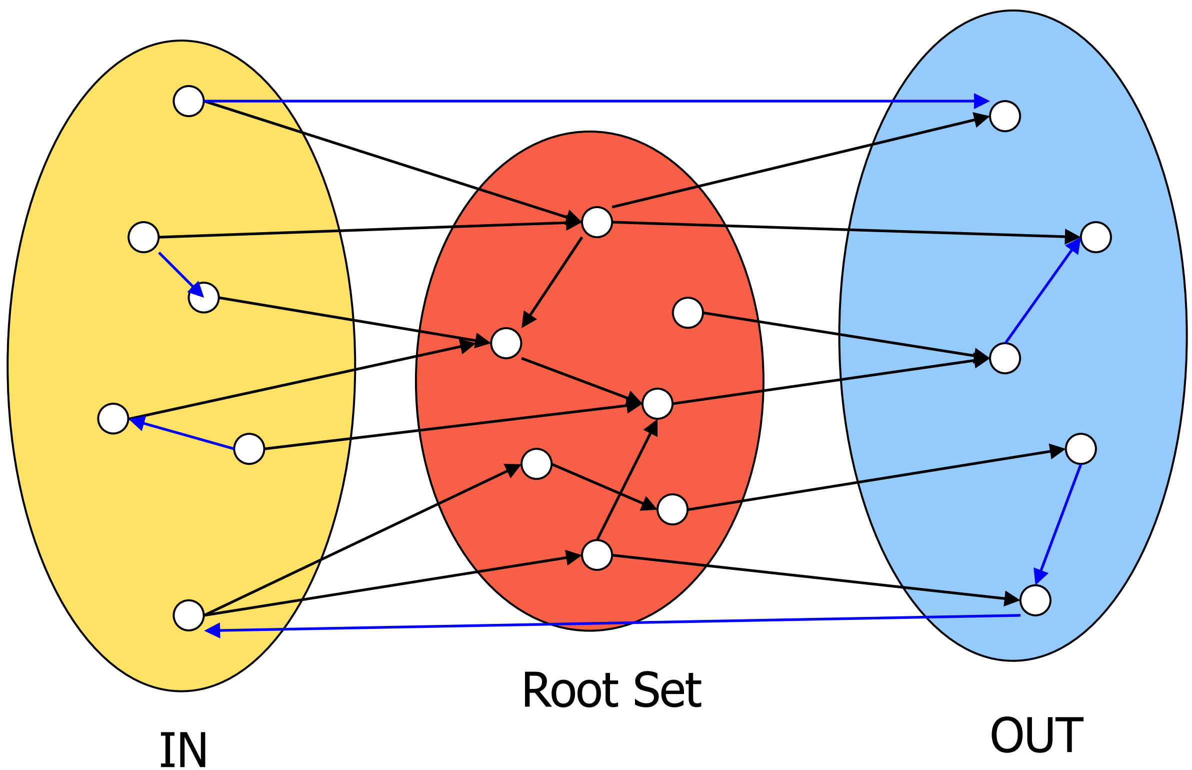


Root Set

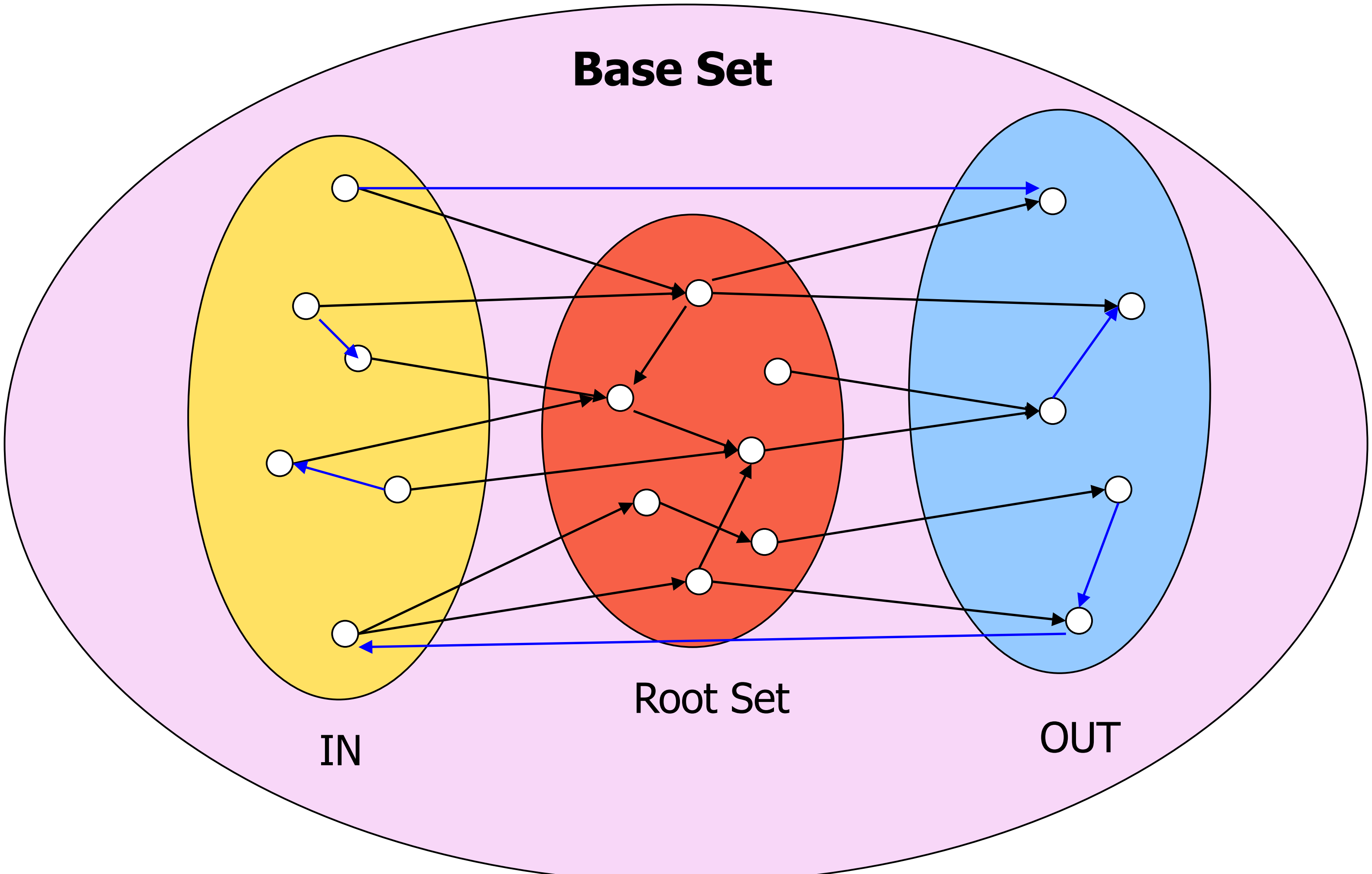
Query dependent input



Query dependent input

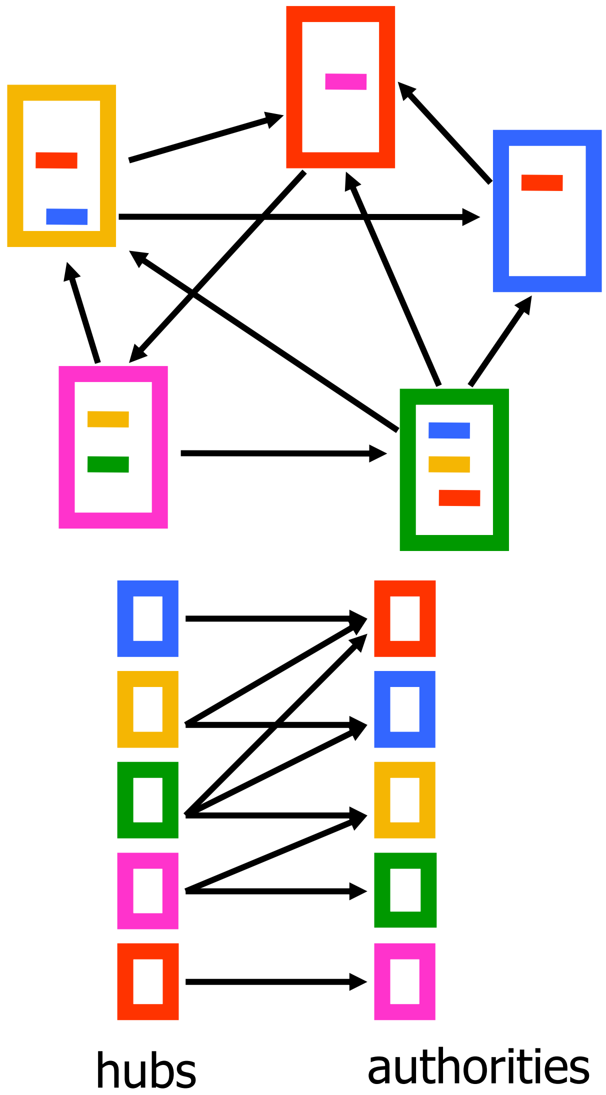


Query dependent input



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

$$h_i^t = \sum_{j : i \to j} a_i^{t-1}$$

  - I operation: authorities collect the weight of the hubs

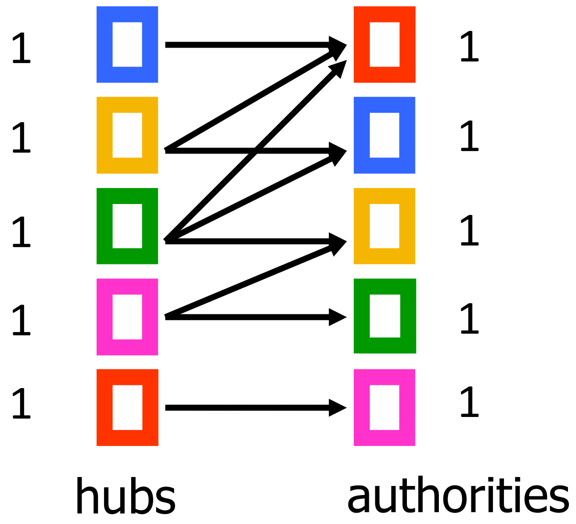
$$a_i^t = \sum_{j : j \to i} h_j^{t-1}$$

  - Normalize weights under some norm

Note: The order of the operations is not important. You could do them in parallel or sequentially, the result will still be the same.

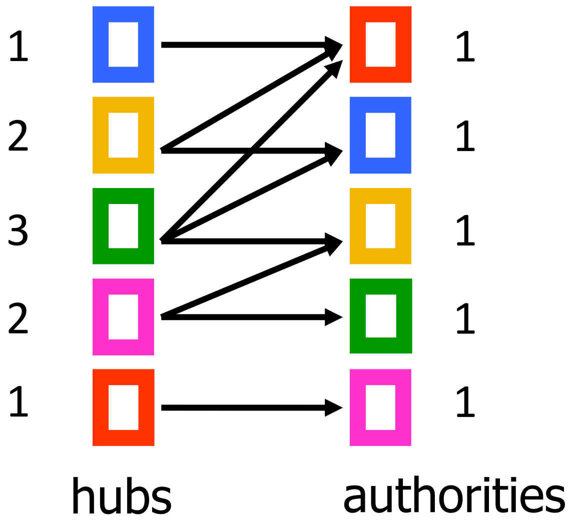
Example

Initialize



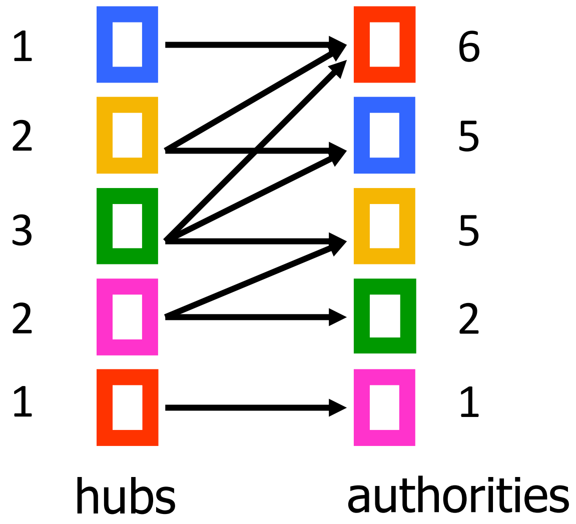
Example

Step 1: O operation



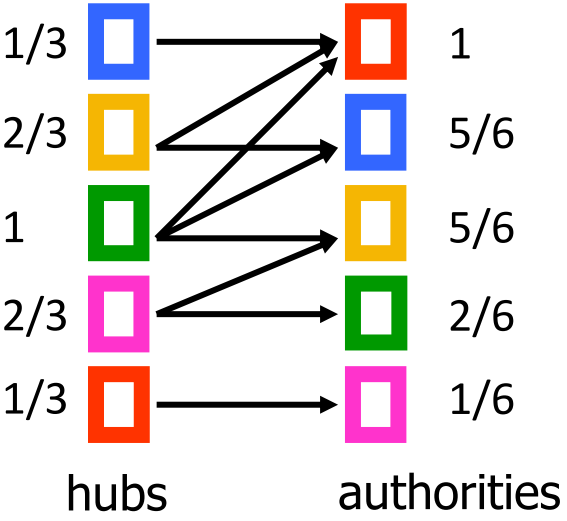
Example

Step 1: I operation



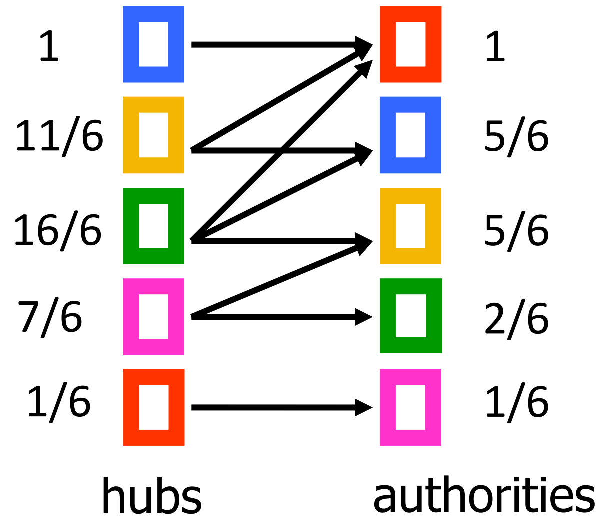
Example

Step 1: Normalization (Max norm)



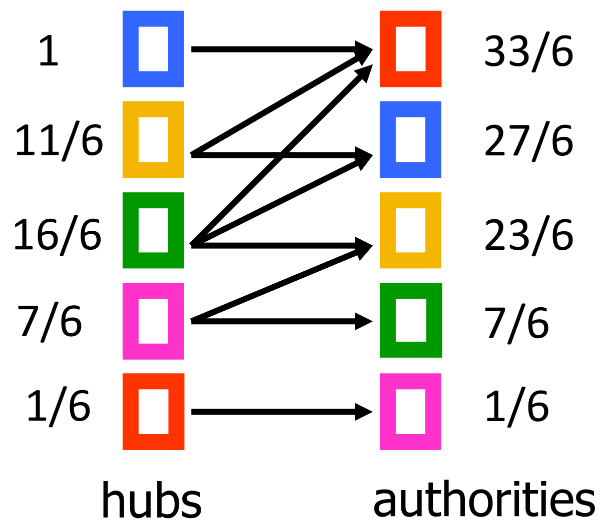
Example

Step 2: O step



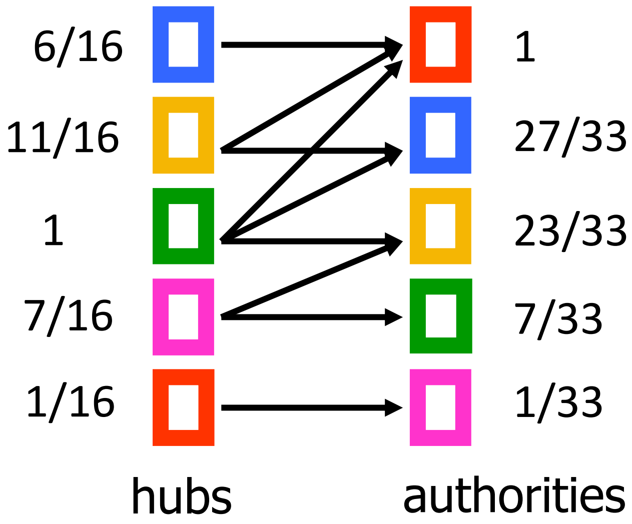
Example

Step 2: I step



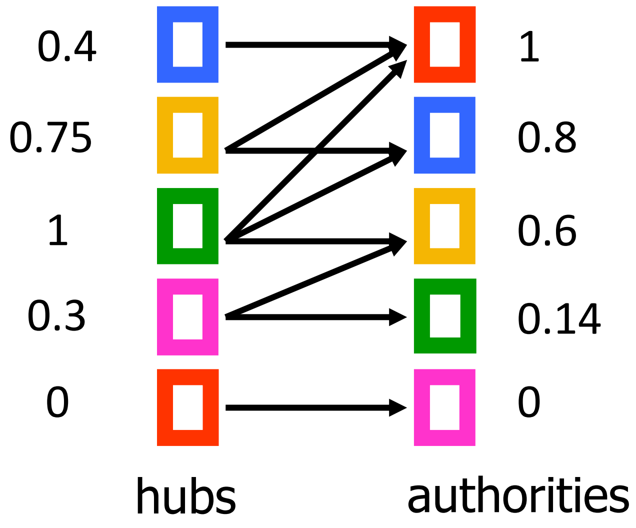
Example

Step 2: Normalization



Example

Convergence



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

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

$$[n \times r]\,[r \times r]\,[r \times n]$$

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

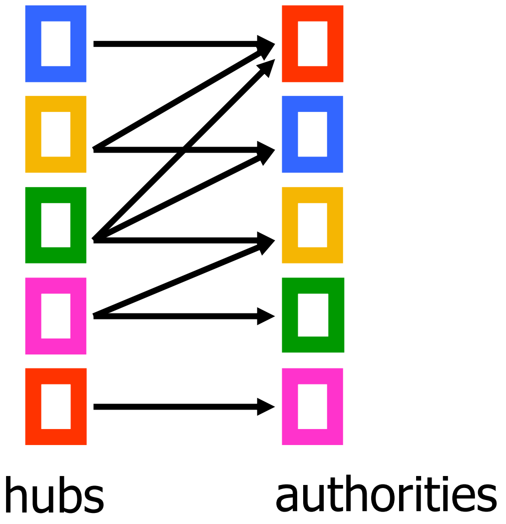
$$A = \sigma_1 \vec{u}_1 \vec{v}_1^T + \sigma_2 \vec{u}_2 \vec{v}_2^T + \cdots + \sigma_r \vec{u}_r \vec{v}_r^T$$

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
  - $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_r|$
- 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, $Rx = \alpha_1 w_1 + a_2 w_2 + \cdots + a_r w_r$
- After t multiplications we have:
  - $R^t x = \lambda_1^{t-1} \alpha_1 w_1 + \lambda_2^{t-1} a_2 w_2 + \cdots + \lambda_r^{t-1} a_r w_r$
- Normalizing (divide by λ_1^{t-1}) leaves only the term w_1.

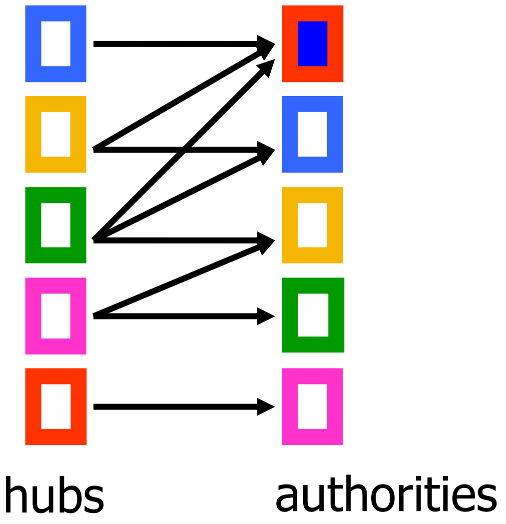
The SALSA algorithm

- Perform a random walk on the bipartite graph of hubs and authorities alternating between the two



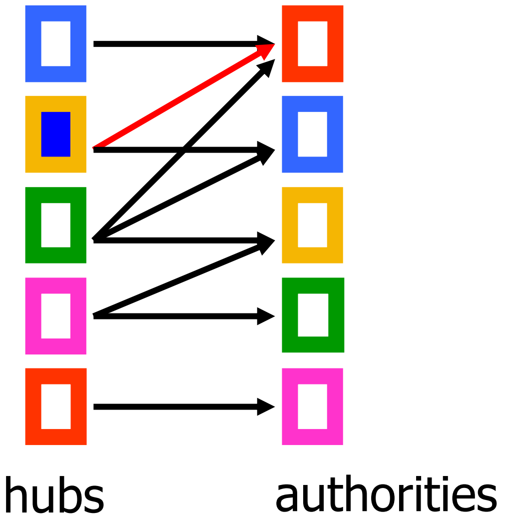
The SALSA algorithm

- Start from an authority chosen uniformly at random
  - e.g. the red authority



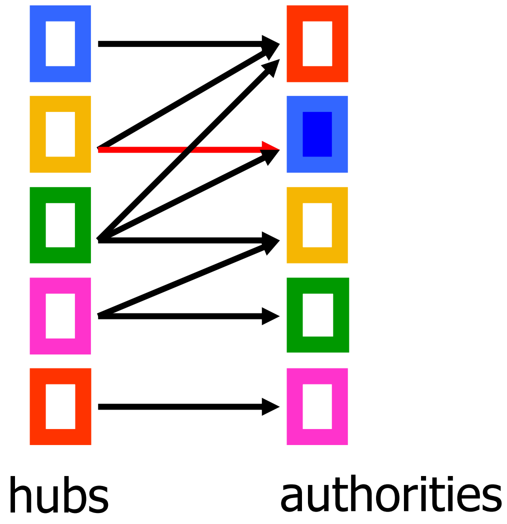
The SALSA algorithm

- Start from an authority chosen uniformly at random
  - e.g. the red authority
- Choose one of the in-coming links uniformly at random and move to a hub
  - e.g. move to the yellow authority with probability 1/3



The SALSA algorithm

- Start from an authority chosen uniformly at random
  - e.g. the red authority
- Choose one of the in-coming links uniformly at random and move to a hub
  - e.g. move to the yellow authority with probability 1/3
- Choose one of the out-going links uniformly at random and move to an authority
  - e.g. move to the blue authority with probability 1/2



The SALSA algorithm

- Formally we have probabilities:
  - a_i: probability of being at authority i
  - h_j: probability of being at hub j
- The probability of being at authority i is computed as:

$$a_i = \sum_{j \in N_{in}(i)} \frac{1}{d_{out}(j)} h_j$$

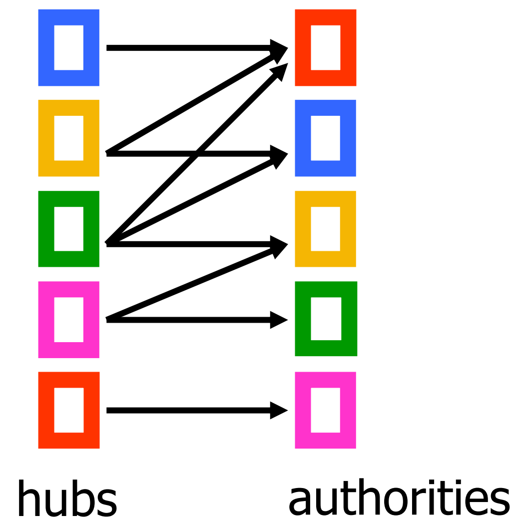
- The probability of being at hub j is computed as

$$h_j = \sum_{i \in N_{out}(j)} \frac{1}{d_{in}(i)} a_i$$

- Repeated computation converges

The SALSA algorithm [LM00]

- In matrix terms
  - A_c = the matrix A where columns are normalized to sum to 1
  - A_r = the matrix A where rows are normalized to sum to 1
- The hub computation
  - $h = A_c a$
- The authority computation
  - $a = A_r^T h = A_r^T A_c a$
- In MC terms the transition matrix
  - $P = A_r A_c^T$



$$\boldsymbol{h_2} = \mathbf{1/3}\, \boldsymbol{a_1} + \mathbf{1/2}\, \boldsymbol{a_2}$$

$$\boldsymbol{a_1} = \boldsymbol{h_1} + \mathbf{1/2}\, \boldsymbol{h_2} + \mathbf{1/3}\, \boldsymbol{h_3}$$