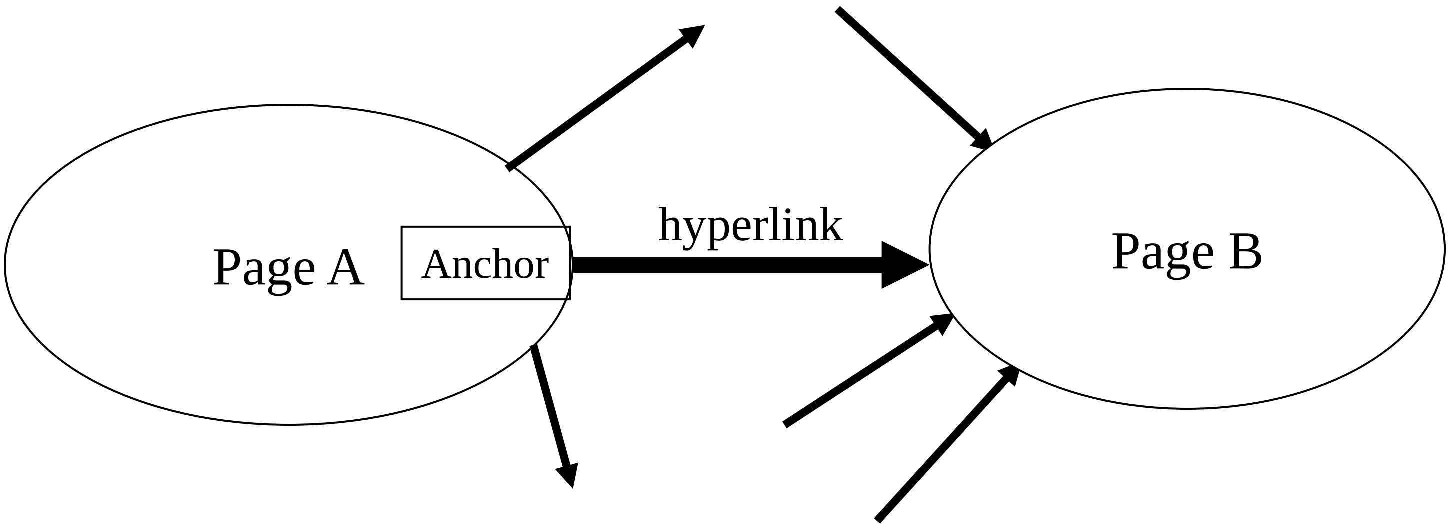
Introduction to

Information Retrieval

Link analysis

The Web as a Directed Graph

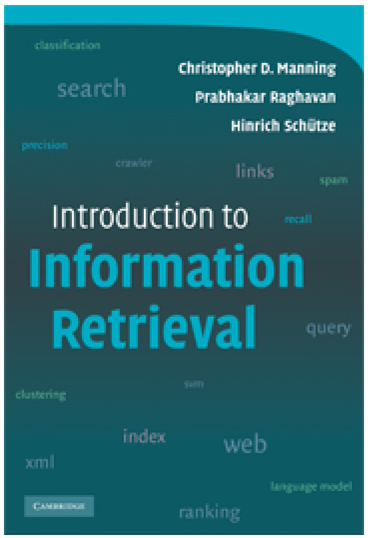


Our study of link analysis builds on two intuitions:

1. The anchor text pointing to page B is a good description of page B.
2. The hyperlink from A to B represents an endorsement of page B, by the creator of page A

Assumption 1: reputed sites

Introduction to Information Retrieval



This is the companion website for the following book.

Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Informat*

You can order this book at CUP, at your local bookstore or on the internet. The best search

The book aims to provide a modern approach to information retrieval from a computer scie
University and at the University of Stuttgart

We'd be pleased to get feedback about how this book works out as a textbook, what is m
comments to: informationretrieval (at) yahoogroups (dot) com

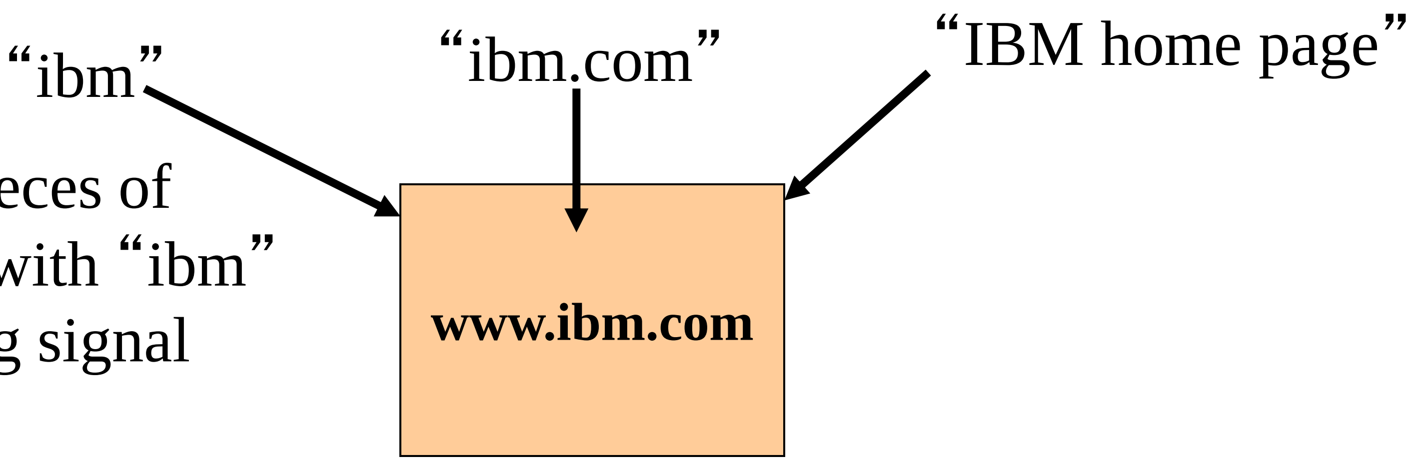
Assumption 2: annotation of target

Anchor Text

WWW Worm - McBryan [Mcbr94]

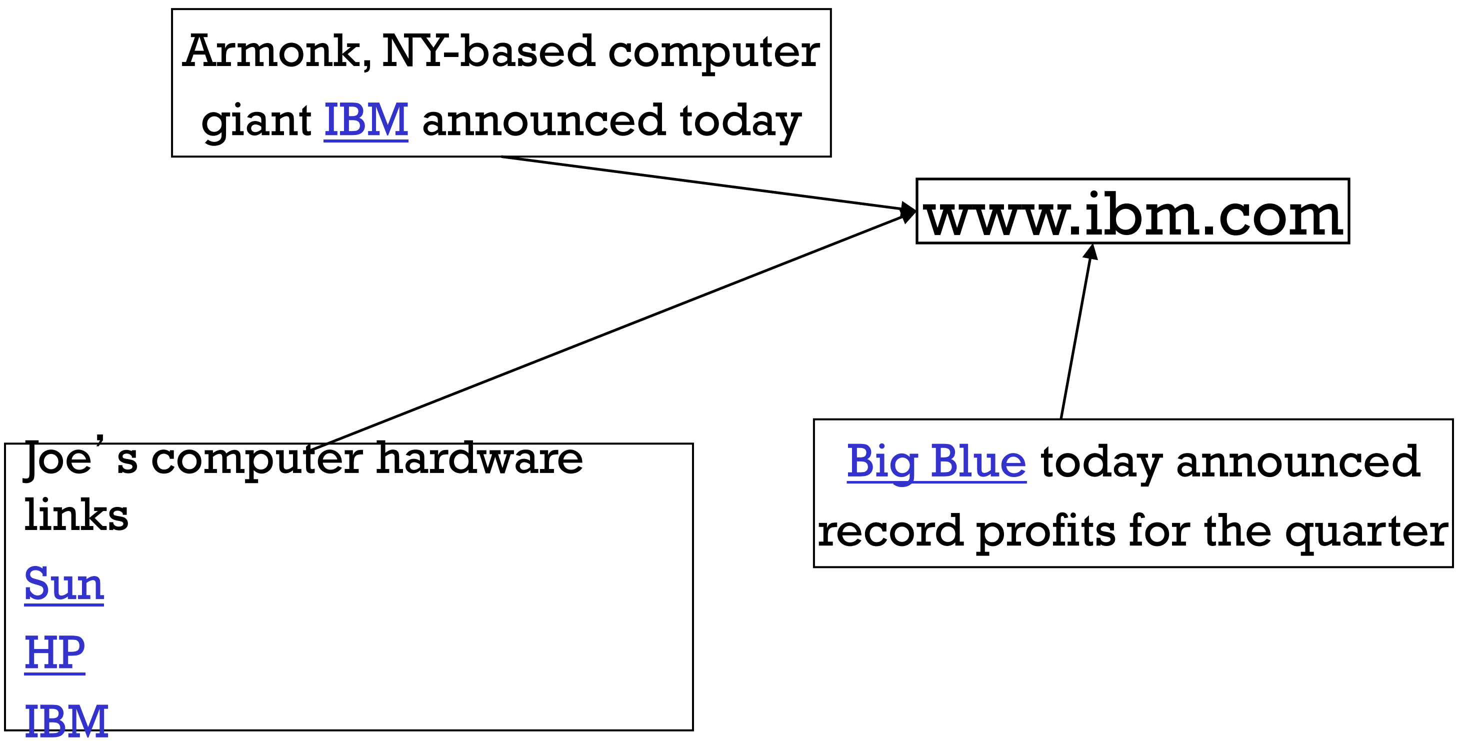
- For ***ibm*** how to distinguish between:
  - IBM’s home page (mostly graphical)
  - IBM’s copyright page (high term freq. for ‘ibm’)
  - Rival’s spam page (arbitrarily high term freq.)

A million pieces of anchor text with “ibm” send a strong signal



Indexing anchor text

- When indexing a document *D*, include (with some weight) anchor text from links pointing to *D*.



Indexing anchor text

- Can sometimes have unexpected effects, e.g., spam, **miserable failure**
- Can score anchor text with weight depending on the authority of the anchor page's website
  - E.g., if we were to assume that content from cnn.com or yahoo.com is authoritative, then trust (more) the anchor text from them
  - Increase the weight of off-site anchors (non-nepotistic scoring)

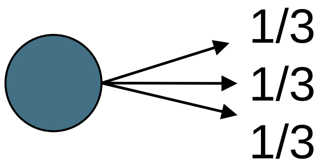
Link analysis: Pagerank

PageRank

- Every node in the web graph is assigned a numerical score between 0 and 1, known as its *PageRank*.
- The PageRank of a node will depend on the link structure of the web graph.
- Given a query, a web search engine computes a composite score for each web page that combines hundreds of features such as cosine similarity and term proximity , together with the PageRank score.

Pagerank scoring – Random walk

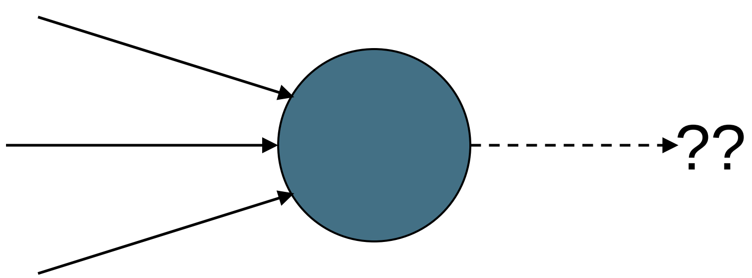
- Imagine a user doing a random walk on web pages:
  - Start at a random page
  - At each step, go out of the current page along one of the links on that page, equiprobably



- "In the long run" each page has a long-term visit rate - use this as the page's score.

Not quite enough

- The web is full of dead-ends.
  - Random walk can get stuck in dead-ends.
  - Makes no sense to talk about long-term visit rates.



Teleporting

- What if the current location of the surfer, the node A, has no out-links?
- To address this we introduce an additional operation for our random surfer: the *teleport* operation.
- In the teleport operation the surfer jumps from a node to any other node in the web graph.
- If N is the total number of nodes in the web graph1, the teleport operation takes the surfer to each node with probability $1/N$.
- The surfer would also teleport to his present position with probability $1/N$.

- In assigning a PageRank score to each node of the web graph, we use the teleport operation in two ways:

(1) When at a node with no out-links, the surfer invokes the teleport operation.

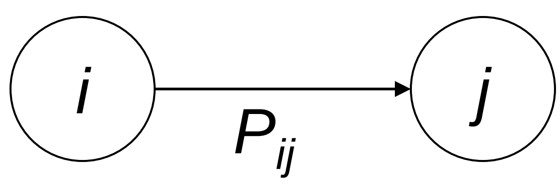
(2)At any node that has outgoing links, the surfer invokes the teleport operation with probability $0 < \alpha < 1$ and the standard random walk with probability $1 - \alpha$, where α is a fixed parameter chosen in advance.

Typically, α might be 0.1.

Markov chains

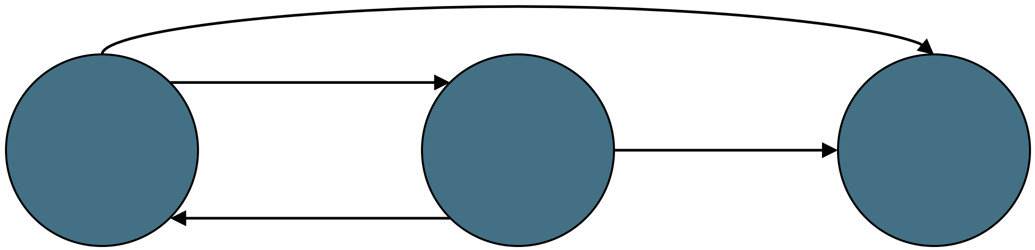
- A Markov chain consists of n states, plus an $n \times n$ transition probability matrix **P**.
- At each step, we are in one of the states.
- For $1 \leq i,j \leq n$, the matrix entry P_{ij} tells us the probability of j being the next state, given we are currently in state i.

$P_{ii}>0$ is OK.



Markov chains

- Clearly, for all *i*, $\sum_{j=1}^{n} P_{ij} = 1.$
- Markov chains are abstractions of random walks.
- *Exercise*: represent the teleporting random walk from 3 slides ago as a Markov chain, for this case:



Ergodic Markov chains

- For any *ergodic* Markov chain, there is a unique long-term visit rate for each state.
  - *Steady-state probability distribution*.
- Over a long time-period, we visit each state in proportion to this rate.
- It doesn't matter where we start.

- We can readily derive the transition probability matrix P for our Markov chain from the $N \times N$ matrix A:

  1. If a row of A has no 1's, then replace each element by 1/N. For all other rows proceed as follows.
  2. Divide each 1 in A by the number of 1's in its row. Thus, if there is a row with three 1's, then each of them is replaced by 1/3.
  3. Multiply the resulting matrix by $1 - \alpha$.
  4. Add α / N to every entry of the resulting matrix, to obtain P.

Probability vectors

- A probability (row) vector **x** = $(x_1, \dots x_n)$ tells us where the walk is at any point.
- E.g., (000…1…000) means we're in state *i*.

    1 *i* *n*

More generally, the vector **x** = $(x_1, \dots x_n)$ means the walk is in state *i* with probability x_i.

$$\sum_{i=1}^{n} x_i = 1.$$

Change in probability vector

- If the probability vector is $\mathbf{x} = (x_1, \ldots x_n)$ at this step, what is it at the next step?
- Recall that row *i* of the transition prob. Matrix **P** tells us where we go next from state *i*.
- So from **x**, our next state is distributed as **xP**
  - The one after that is $\mathbf{xP}^2$, then $\mathbf{xP}^3$, etc.
  - (Where) Does this converge?

How do we compute this vector?

- Let **a** = $(a_1, \ldots a_n)$ denote the row vector of steady-state probabilities.
- If our current position is described by **a**, then the next step is distributed as **aP**.
- But **a** is the steady state, so **a**=**aP**.
- Solving this matrix equation gives us **a**.
  - So **a** is the (left) eigenvector for **P**.
  - (Corresponds to the "principal" eigenvector of **P** with the largest eigenvalue.)
  - Transition probability matrices always have largest eigenvalue 1.

Link analysis: HITS

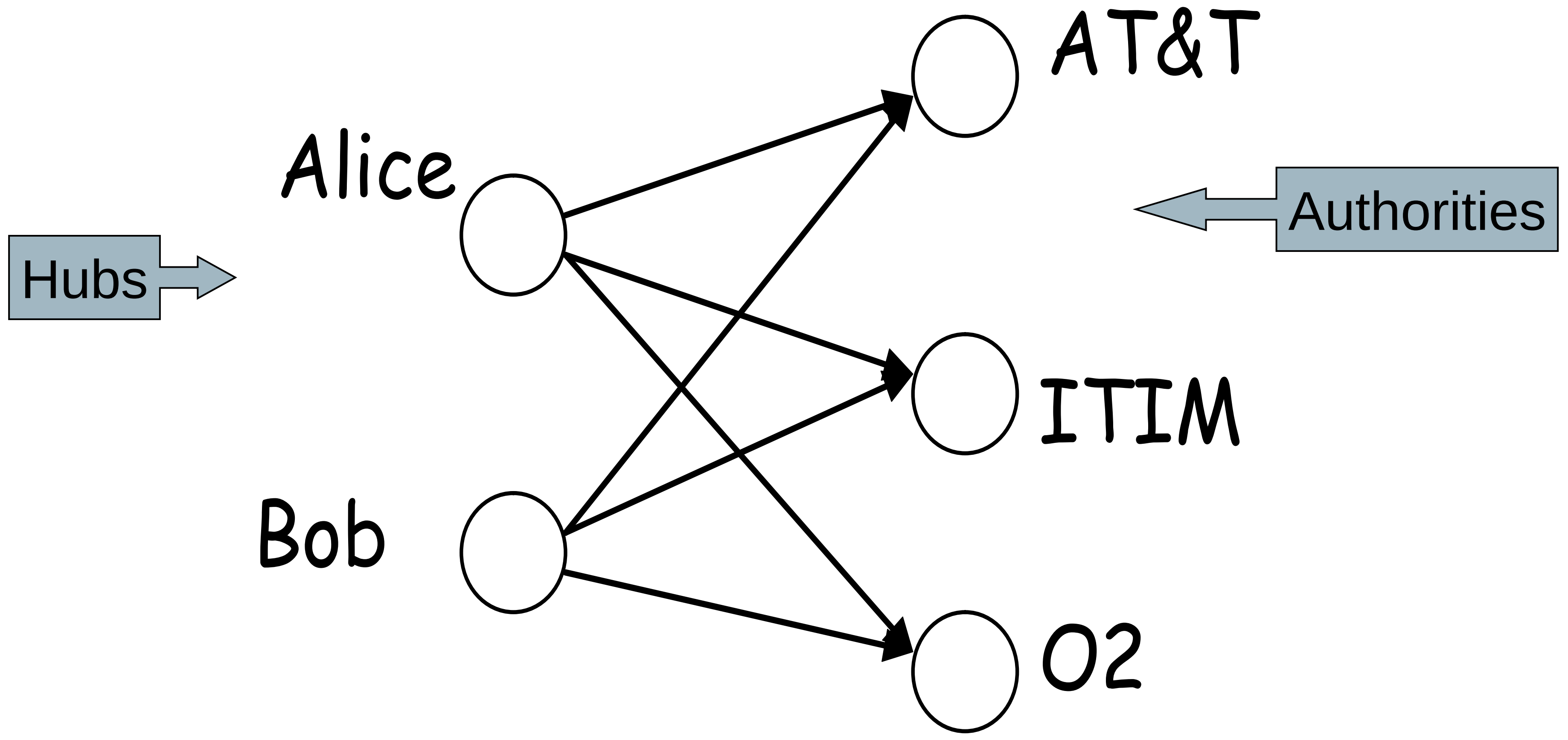
Hyperlink-Induced Topic Search (HITS)

- In response to a query, instead of an ordered list of pages each meeting the query, find two sets of inter-related pages:
  - *Hub pages* are good lists of links on a subject.
    - e.g., "Bob's list of cancer-related links."
  - *Authority pages* occur recurrently on good hubs for the subject.
- Best suited for "broad topic" queries rather than for page-finding queries.
- Gets at a broader slice of common *opinion*.

Hubs and Authorities

- Thus, a good hub page for a topic *points* to many authoritative pages for that topic.
- A good authority page for a topic is *pointed* to by many good hubs for that topic.
- Circular definition - will turn this into an iterative computation.

The hope



Mobile telecom companies

High-level scheme

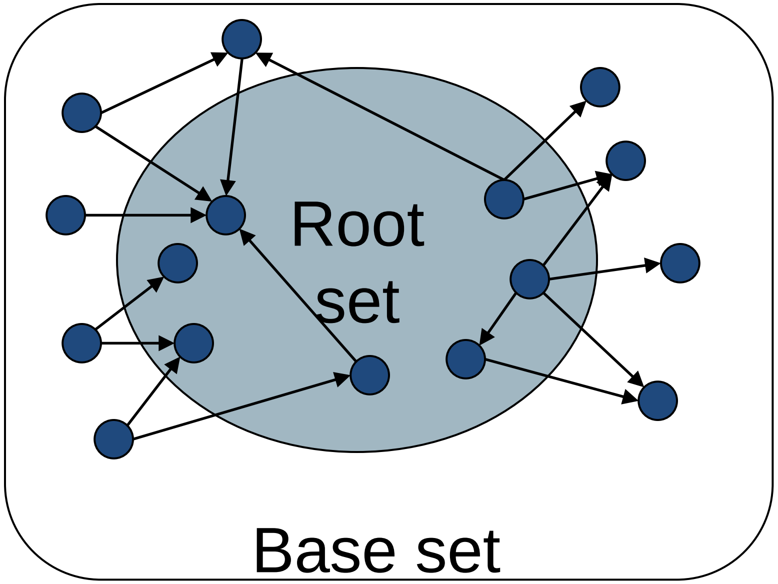
- Extract from the web a base set of pages that *could* be good hubs or authorities.
- From these, identify a small set of top hub and authority pages;

  →iterative algorithm.

Base set

- Given text query (say ***browser***), use a text index to get all pages containing ***browser.***
  - Call this the root set of pages.
- Add in any page that either
  - points to a page in the root set, or
  - is pointed to by a page in the root set.
- Call this the base set.

Visualization



Get in-links (and out-links) from a *connectivity server*

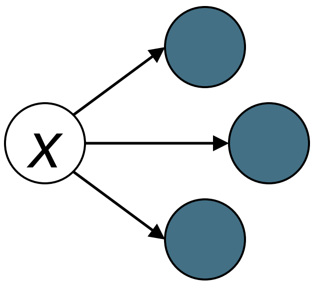
Distilling hubs and authorities

- Compute, for each page *x* in the base set, a hub score *h(x)* and an authority score *a(x).*
- Initialize: for all *x*, $h(x) \leftarrow 1$; $a(x) \leftarrow 1$;
- Iteratively update all *h(x), a(x)*; ⇐ Key
- After iterations
  - output pages with highest *h()* scores as top hubs
  - highest *a()* scores as top authorities.

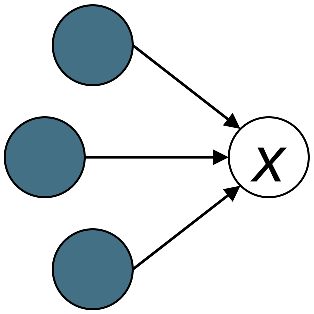
Iterative update

- Repeat the following updates, for all *x*:

$$h(x) \leftarrow \sum_{x \mapsto y} a(y)$$



$$a(x) \leftarrow \sum_{y \mapsto x} h(y)$$



Scaling

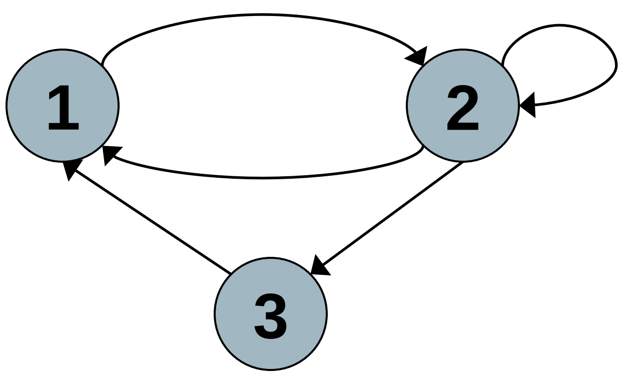
- To prevent the *h()* and *a()* values from getting too big, can scale down after each iteration.
- Scaling factor doesn't really matter:
  - we only care about the *relative* values of the scores.

How many iterations?

- Claim: relative values of scores will converge after a few iterations:
  - in fact, suitably scaled, *h()* and *a()* scores settle into a steady state!
  - proof of this comes later.
- In practice, ~5 iterations get you close to stability.

Proof of convergence

- $n \times n$ adjacency matrix **A**:
  - each of the n pages in the base set has a row and column in the matrix.
  - Entry $A_{ij} = 1$ if page i links to page j, else = 0.



$$
\begin{array}{c|ccc|}
 & 1 & 2 & 3 \\
\hline
1 & 0 & 1 & 0 \\
2 & 1 & 1 & 1 \\
3 & 1 & 0 & 0 \\
\hline
\end{array}
$$

Hub/authority vectors

- View the hub scores *h()* and the authority scores *a()* as vectors with *n* components.
- Recall the iterative updates

$$h(x) \leftarrow \sum_{x \mapsto y} a(y)$$

$$a(x) \leftarrow \sum_{y \mapsto x} h(y)$$

Rewrite in matrix form

- **h=Aa.**
- **a=A^{t}h.**

Recall A^t is the transpose of A.

Substituting, **h=AA^{t}h** and **a=A^{t}Aa**.

Thus, **h** is an eigenvector of **AA^t** and **a** is an eigenvector of **A^{t}A**.

Further, our algorithm is a particular, known algorithm for computing eigenvectors: the *power iteration* method.

Guaranteed to converge.