Link Analysis: PageRank

Lin Guosheng
School of Computer Science and Engineering
Nanyang Technological University

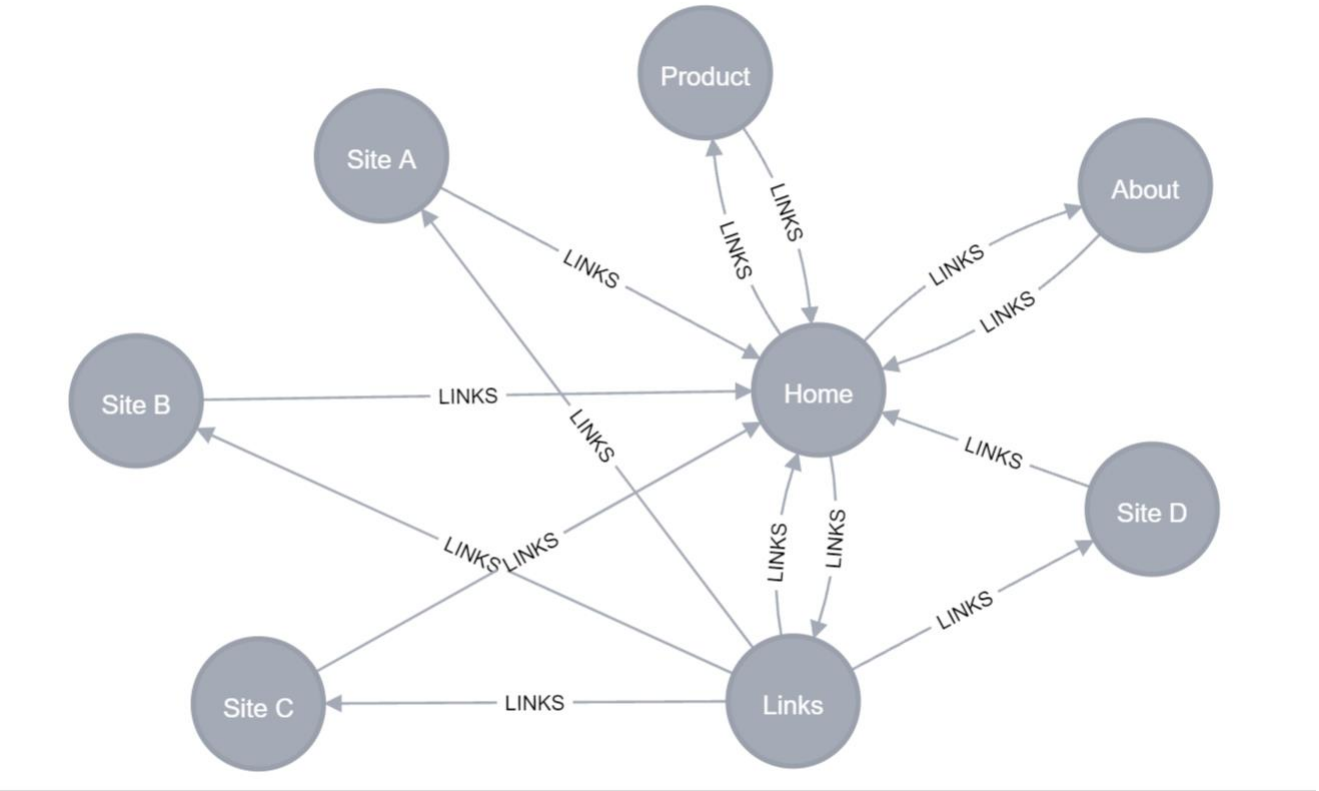
Outline

- PageRank Algorithm
  - Page rank
  - Page rank extensions
    - Potential problems
    - Teleportation
    - Page rank – Google Matrix

Many slides are from:
http://web.stanford.edu/class/cs224w/slides/04-pagerank.pdf

Web as a directed graph:

- Nodes = web pages
- Edges with directions = hyperlinks (in-links and out-links)

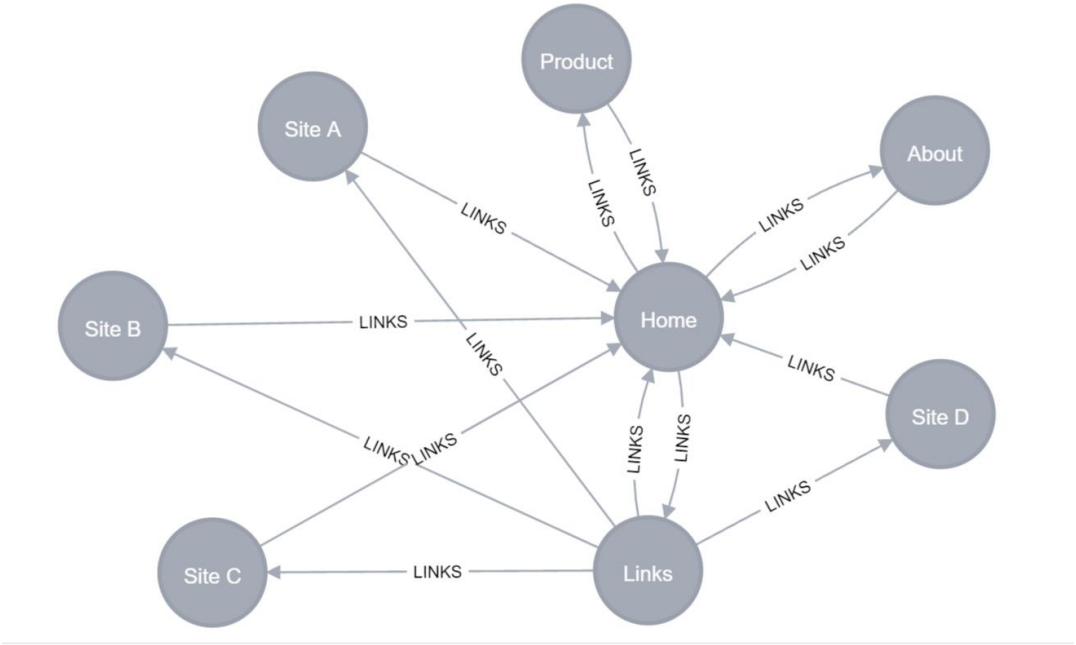


Represent the web as directed graph

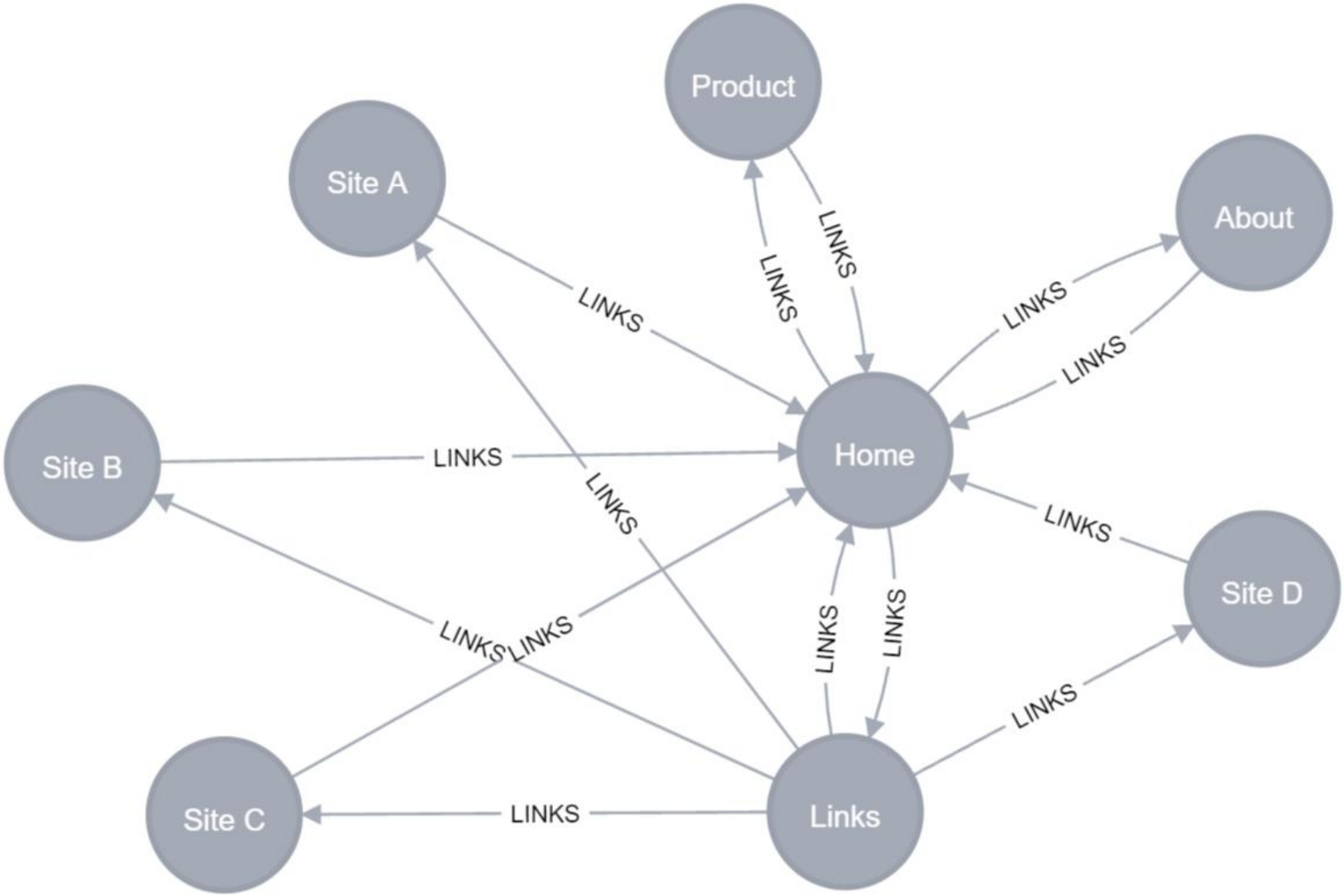
- Link analysis is to discover useful information from relationships/connections between data objects. (mining the graph data)
- Use graphs to describe connections.
  - Vertices represent objects/data points
  - Links describe relationships among objects.
- PageRank is an algorithm to rank web pages.
  - An algorithm for link analysis and web search.
  - proposed by Sergey Brin and Larry Page,
    - students at Stanford University and the founders of Google.

- All web pages are not equally “important”
  - https://guosheng.github.io/
  - https://www.ntu.edu.sg/ (More important)

- How to measure the importance of web pages?



- How to measure the importance of web pages?
  - We can look at the links.
    - Q1: In-coming links vs out-going links?
    - Q2: Are all links equally important?


Product
Site A
About
LINKS
LINKS
LINKS
LINKS
LINKS
Site B
Home
LINKS
LINKS
LINKS
Site D
LINKS
LINKS
LINKS
LINKS
LINKS
Site C
LINKS
Links

Idea: Links as votes

- Page is more important if it has more links
  - In-coming links vs Out-going links
    - In-coming links are more important
  - Examples:
    - www.stanford.edu has 23,400 in-links
    - thispersondoesnotexist.com has 1 in-link

Idea: Links as votes

- Are all in-links equal?
  - In-links from important pages count more
- Summary:
  - Two key factors affecting the rank score:
  - 1. The number of in-links.
  - 2. The source nodes of the in-links

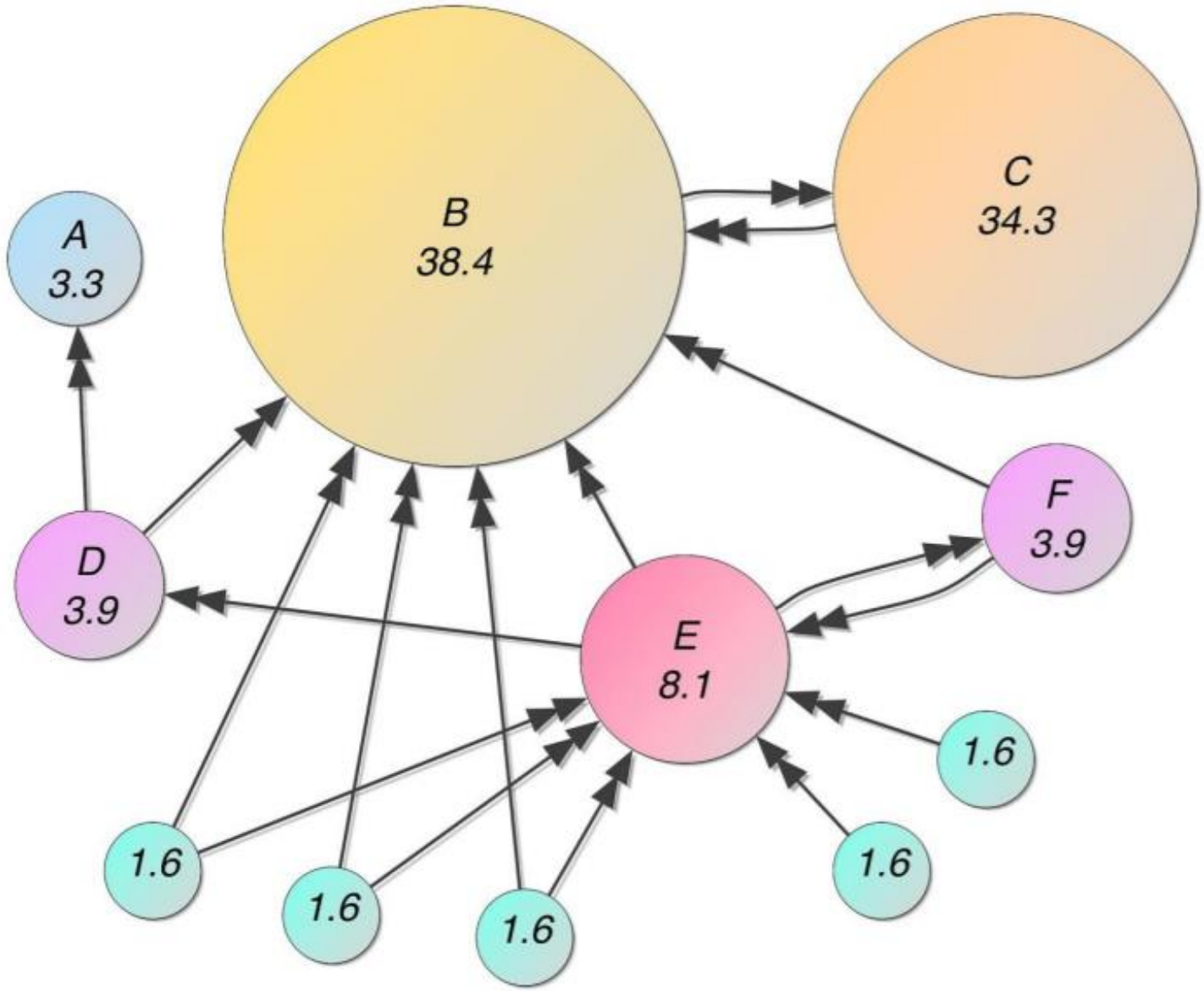


Image credit: Wikipedia

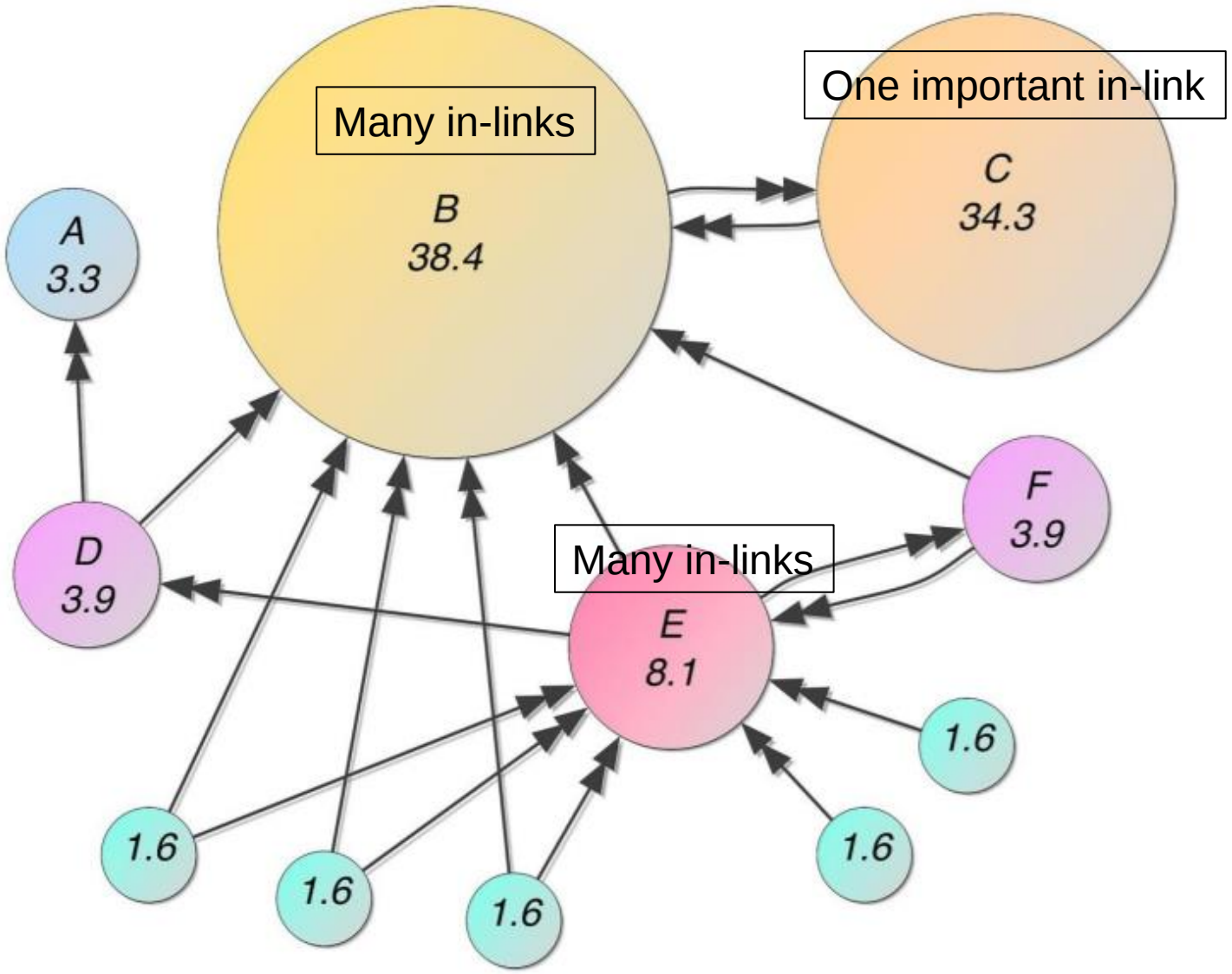


Image credit: Wikipedia

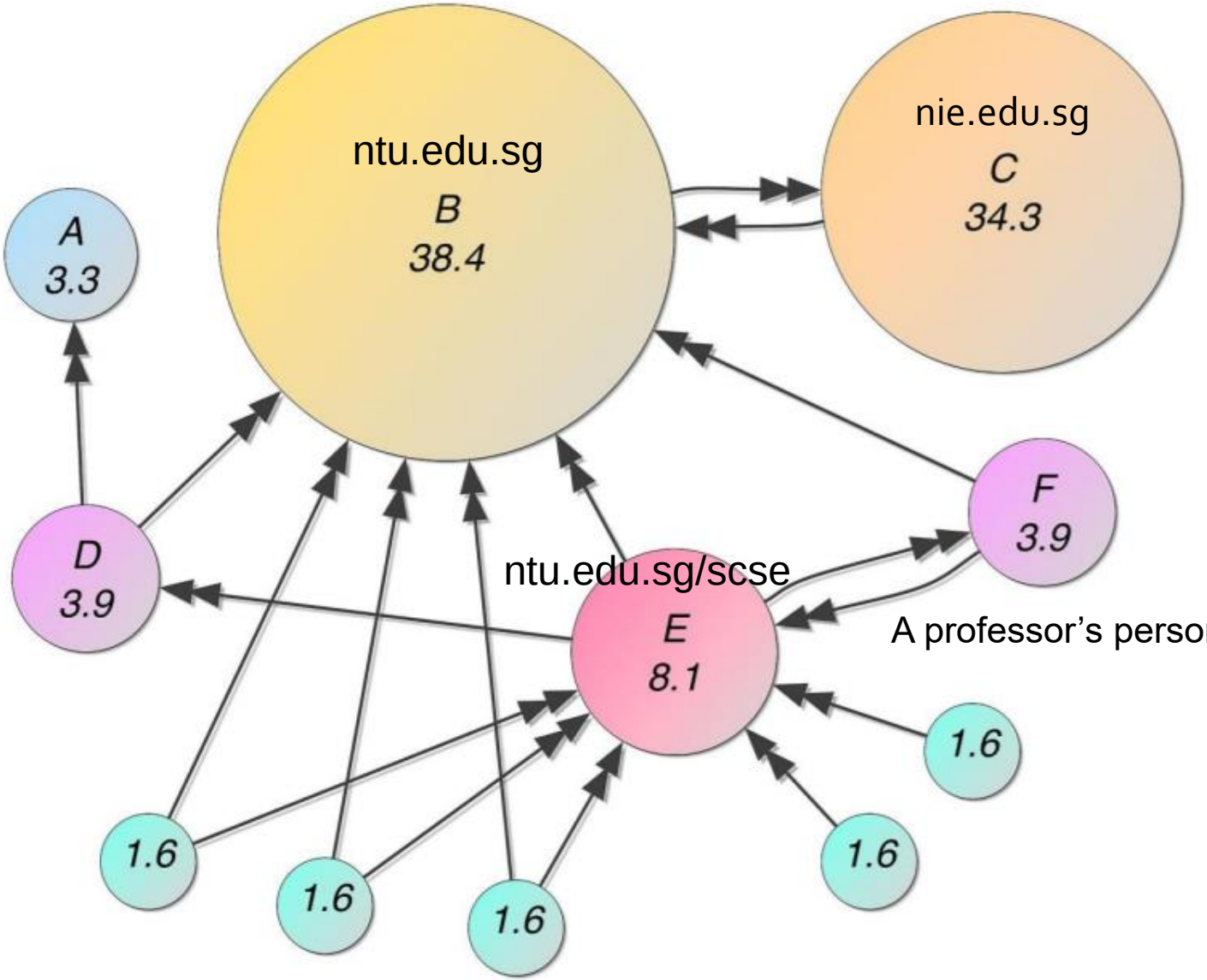
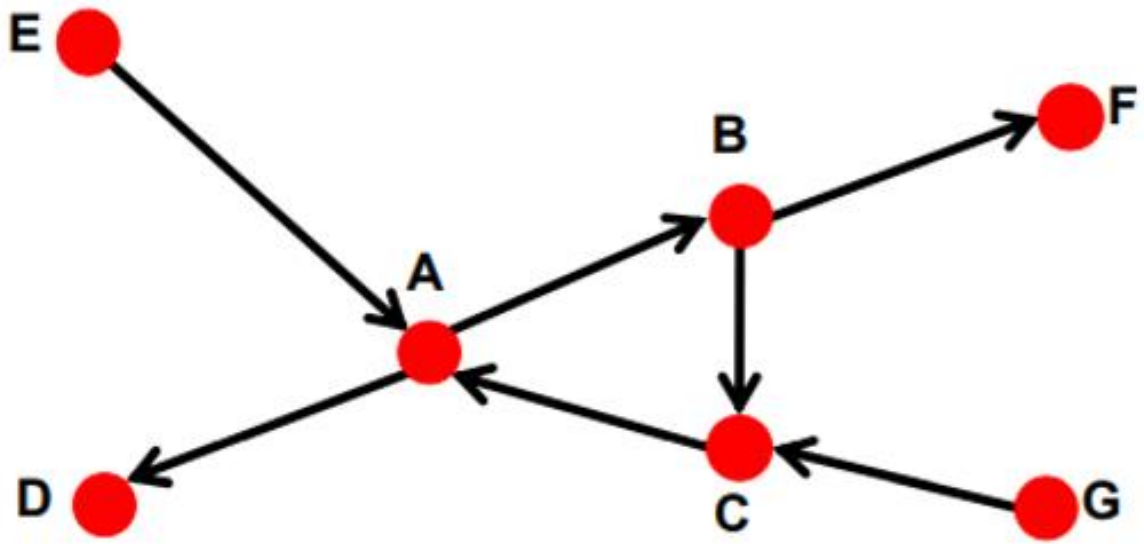


Image credit: Wikipedia

- In-links of a node (or backlinks)
  - the links pointing in (from other nodes), incoming links
  - in-degree: the number of in-links
- Out-links of a node
  - The links pointing out (to other nodes), outgoing links
  - out-degree: the number of out-links



Examples:

In-degree of node A: 2
(from E and C)

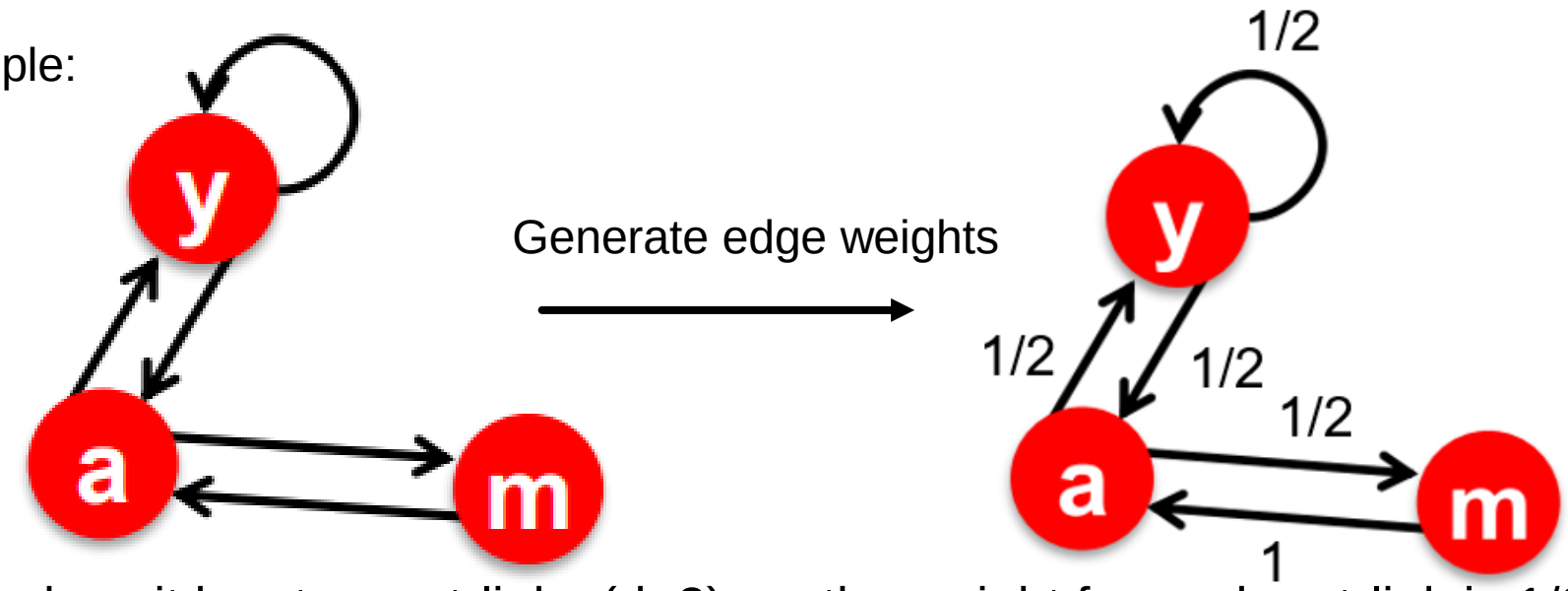
Out-degree of node A: 2
(to D and B)

PageRank

- Define the edge (link) weights as $w_{ij} = \frac{1}{d_i}$

d_i: out-degree of the source node i (the number of out-links)
use the source node of the link to define the weight.

Example:



Node y: it has two out-links (d=2), so the weight for each out-link is 1/2

Node a: it has two out-links (d=2), so the weight for each out-link is 1/2

Node m: it has one out-link (d=1), so the weight for each out-link is 1/1

PageRank

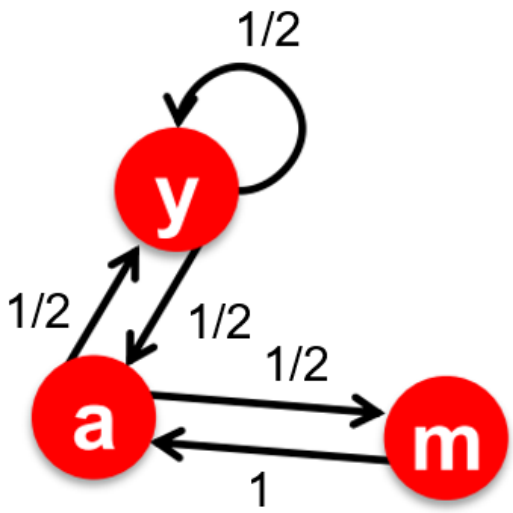
- Define the edge (link) weights as $w_{ij} = \frac{1}{d_i}$

d_i: out-degree of the source node i (the number of out-links)
use the source node of the link to define the weight.

The edge weights will be used to calculate the node importance score (page rank score)

The edge weights indicate the "votes" from the source node to the target node.

Example:



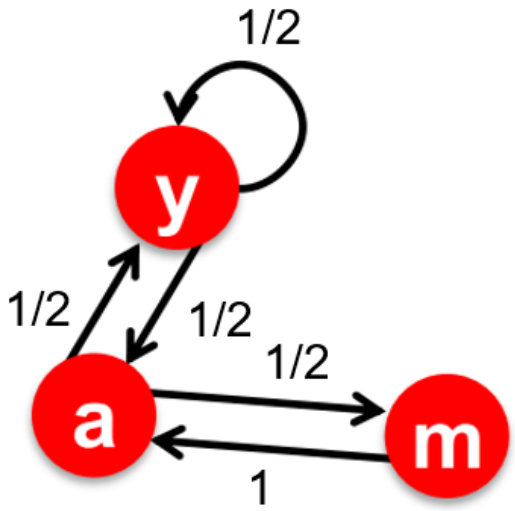
- Define the edge weights as $w_{ij} = \frac{1}{d_i}$

All the out-links of one node share the same weight

Node y: it has two out-links (d=2), so the weight for each out-link is 1/2

Node a: it has two out-links (d=2), so the weight for each out-link is 1/2

Node m: it has one out-link (d=1), so the weight for each out-link is 1/1



- PageRank: assign a rank score to each node

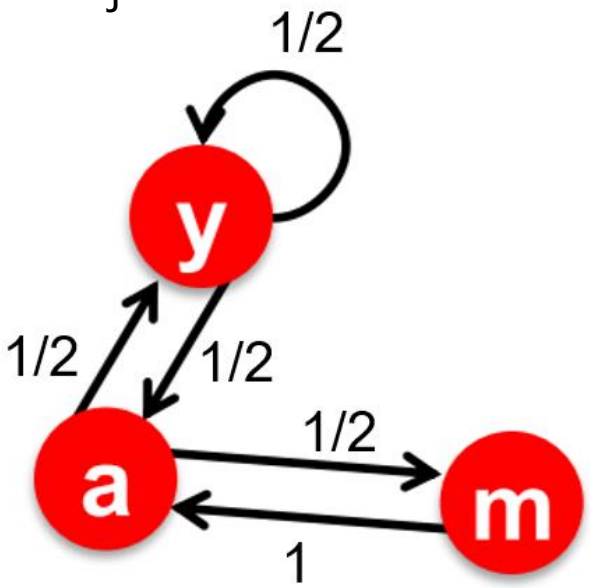
For a node j, we define the rank score r_j as (flow equations):

$$r_j = \sum_{i \to j} w_{ij}\, r_i = \sum_{i \to j} \frac{1}{d_i}\, r_i$$

the edge weight:
(the link from i to j) $w_{ij} = \frac{1}{d_i}$

One node rank score
= weighted sum of the
in-linked node scores;

Example:



$r_y = r_y/2 + r_a/2$

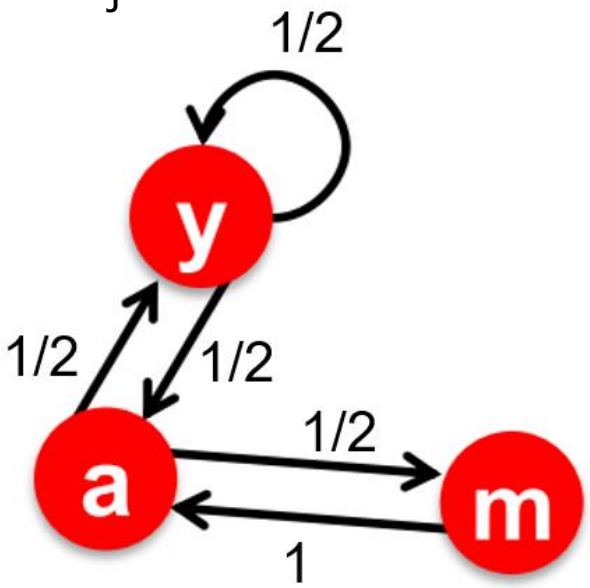
y has two in-links:
a->y and y->y

- PageRank: assign a rank score to each node

For a node j, we define the rank score r_j as (flow equations):

$$r_j = \sum_{i \to j} w_{ij}\, r_i = \sum_{i \to j} \frac{1}{d_i}\, r_i$$

Example:



$r_y = r_y/2 + r_a/2$

y has two in-links: a->y and y->y

We also require:
(The sum of all node scores equals 1):

$$\sum_i r_i = 1$$

Flow equation:
flow-out value = flow-in value

One node score = weighted sum of the in-linked node scores;

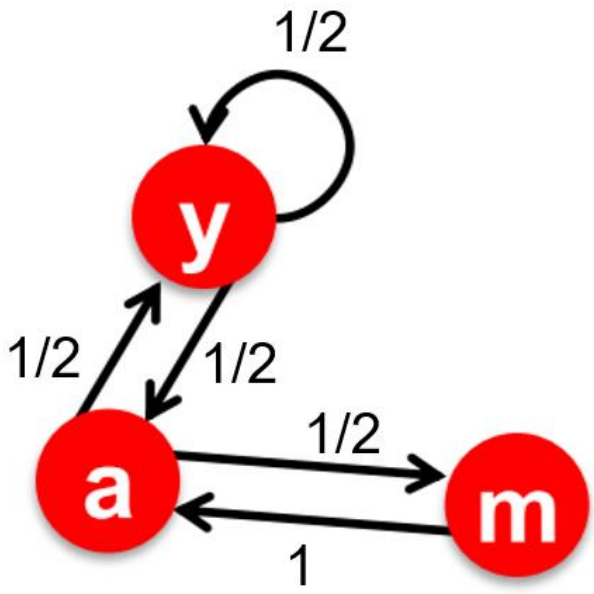
$$r_j = \sum_{i \to j} w_{ij}\, r_i = \sum_{i \to j} \frac{1}{d_i}\, r_i$$

"Flow" equations:

$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$



1. y has two in-links: y->y and a->y; two in-linked nodes: y and a.
2. a has two in-links: y->a and m->a; two in-linked nodes: y and m.
3. m has one in-link: a->m; one in-linked node: a.

- How to solve for the rank scores?
  - Solution 1: directly solve the linear equations
  - Solution 2: power iteration method

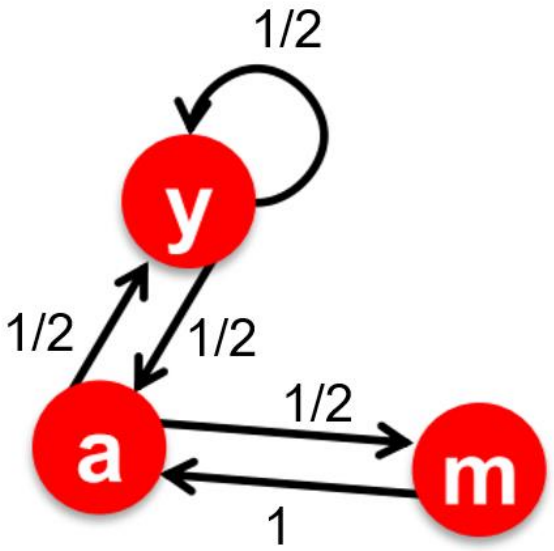
- **Solution 1:**
  Directly solve the linear equations

"Flow" equations:

$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$



Adding this equation:

$$\sum_i r_i = 1$$

(The sum of all rank scores equals 1)

We can solve these 4 linear equations to get the solutions:

$r_y = 0.4$; $r_a = 0.4$; $r_m = 0.2$

Example using elimination methods to solve linear systems:

- EQ1: y= y/2 + a/2
- EQ2: a=y/2 + m
- EQ3: m=a/2
- EQ4: a + y + m =1

- With EQ2, EQ3 -> a=y/2 + a/2 (eliminate y)
  -> EQ5: a=y

- With EQ4, EQ3, EQ5 -> a + a + a/2 =1 (eliminate y, m)

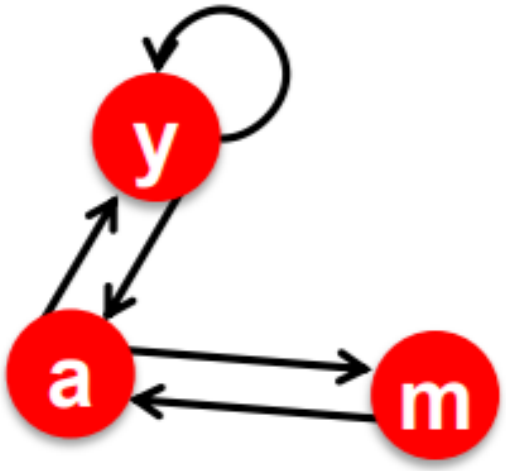
- -> a= 2/5 = 0.4
- -> y= 0.4, m-> 0.2

- **Solution 2:**
  power iteration method
  - Construct transition matrix: M
    - Matrix size: n x n (n: the number of nodes)
    - M is constructed by the edge weights.

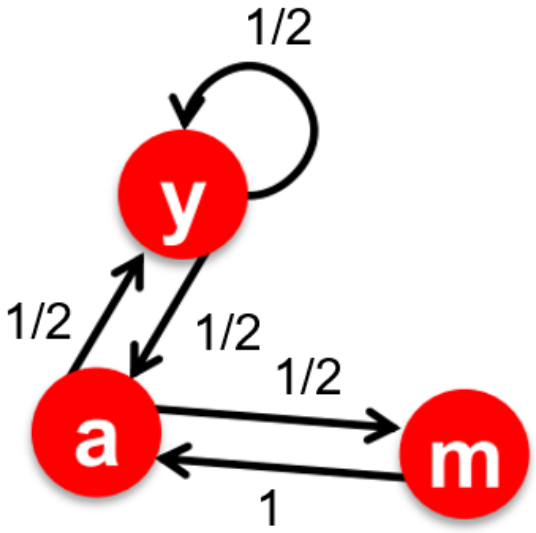
If there is a link from i to j (i->j),

$$M_{j,i} = \frac{1}{d_i}$$

d_i is the out-degree of node i



Generate edge weights



If there is a link from i to j (i->j), $M_{j,i} = \frac{1}{d_i}$

d_i is the out-degree of node i

Construct M column by column or row by row:
1. column by column:
Each column in M indicates out-links for one node

2. row by row:
Each row in M indicates the in-links for one node

	r_y	r_a	r_m
r_y	½	½	0
r_a	½	0	1
r_m	0	½	0

Transition matrix M

- Convert flow equations into matrix multiplication

The flow equation of one node (j=1, 2,…, n):

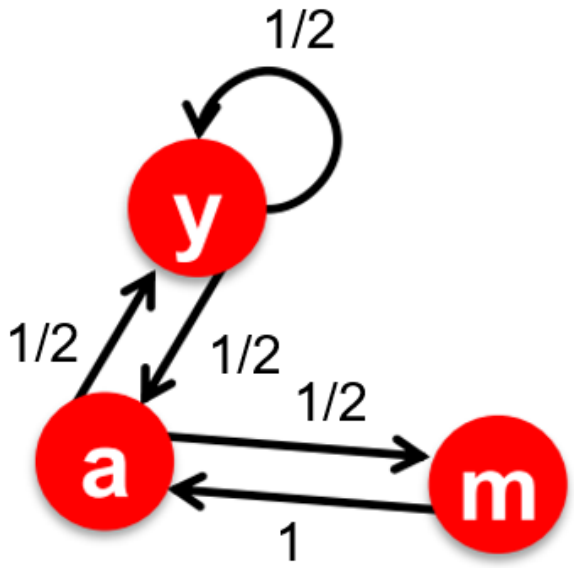
$$r_j = \sum_{i \to j} \frac{r_i}{d_i}$$

Matrix expression of all flow equations:

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

Rank vector **r :** a column vector of all rank scores.

r = $[r_1, r_2, r_3, \ldots, r_n]^T$**.** **n**: the total number of nodes.



	r_y	r_a	r_m
r_y	½	½	0
r_a	½	0	1
r_m	0	½	0

Transition matrix M

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

Matrix expression

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} ½ & ½ & 0 \\ ½ & 0 & 1 \\ 0 & ½ & 0 \end{bmatrix} \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix}$$

r M r

- Solution 2: Power iteration method
  - 1. assign initial values to rank scores : $r_i = 1/N$;
    - N is the total number of nodes
    - Uniform distribution
  - 2. repeat the following util converge:

Calculate the page rank: $$r^{(t+1)} = M \cdot r^{(t)}$$

Or equivalently update each node using the flow equation: $$r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i}$$

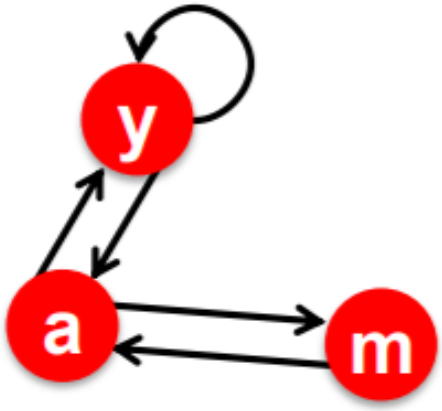
Converge criteria: $\left(\sum_i \left|r_i^{t+1} - r_i^t\right| < \epsilon\right)$

ϵ : is a pre-defined small value, e.g., 0.0001

Or converge when reach the maximum number of iterations

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

$\boldsymbol{r'}$ is the updated score vector $\boldsymbol{r}^{(t+1)}$



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

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

Initialization (uniform distribution): r_y=1/3, r_a=1/3, r_m=1/3 ;

Iteration 1: (Flow equation update)

$r'_y = r_y/2 + r_a/2 = 1/3 * 1/2 + 1/3 * 1/2 = 1/3;$

$r'_a = r_y/2 + r_m = 1/3 * 1/2 + 1/3 = 3/6;$

$r'_m = r_a/2 = 1/3 * 1/2 = 1/6;$

Update r: r_y=1/3, r_a=3/6, r_m=1/6 ;

Or we can use matrix update:

$$r^{(t+1)} = M \cdot r^{(t)}$$

	r_y	r_a	r_m
r_y	½	½	0
r_a	½	0	1
r_m	0	½	0

Transition matrix M

Iteration 1:

$$\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} = \begin{bmatrix} 1/3 \\ 3/6 \\ 1/6 \end{bmatrix}$$

M **r_0** **r_1**

Matrix update:

$$r^{(t+1)} = M \cdot r^{(t)}$$

Iteration 1:

$$\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} = \begin{bmatrix} 1/3 \\ 3/6 \\ 1/6 \end{bmatrix}$$

M **r_0** **r_1**

Flow equation update:

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

Initialization: r_y=1/3, r_a=1/3, r_m=1/3 ;

Iteration 1:

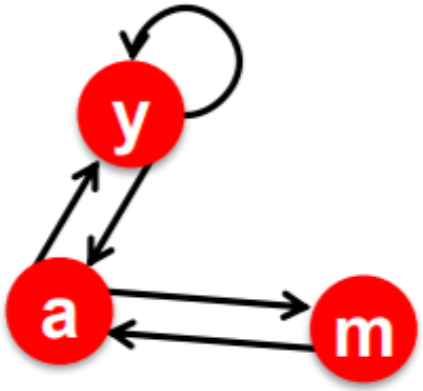
$r'_y = r_y/2 + r_a/2 = 1/3 * 1/2 + 1/3 * 1/2 = 1/3;$

$r'_a = r_y/2 + r_m = 1/3 * 1/2 + 1/3 = 3/6;$

$r'_m = r_a/2 = 1/3 * 1/2 = 1/6;$

Update r: r_y=1/3, r_a=3/6, r_m=1/6 ;

They give the same result



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

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

The node scores after iteration 1: r_y=1/3, r_a=3/6, r_m=1/6 ;

Iteration 2:

$r'_y = r_y/2 + r_a/2 = 1/3 * 1/2 + 3/6 * 1/2 = 5/12;$

$r'_a = r_y/2 + r_m = 1/3 * 1/2 + 1/6 = 1/3;$

$r'_m = r_a/2 = 3/6 * 1/2 = 3/12;$

Update r: r_y=5/12, r_a=1/3, r_m=3/12 ;

The node scores after iteration 2: r_y=5/12, r_a=1/3, r_m=3/12 ;

Iteration 3:

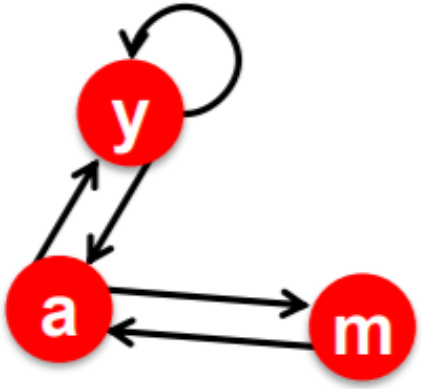
$r'_y = r_y/2 + r_a/2 = 5/12 * 1/2 + 1/3 * 1/2 = 9/24;$

$r'_a = r_y/2 + r_m = 5/12 * 1/2 + 3/12 = 11/24;$

$r'_m = r_a/2 = 1/3 * 1/2 = 1/6;$

Update r: r_y=9/24, r_a=11/24, r_m=1/6 ;

Iteration 4, 5, 7, …., until converged



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

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

Summarized the results for solution2:

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

Initial value: 1/N (N=3)

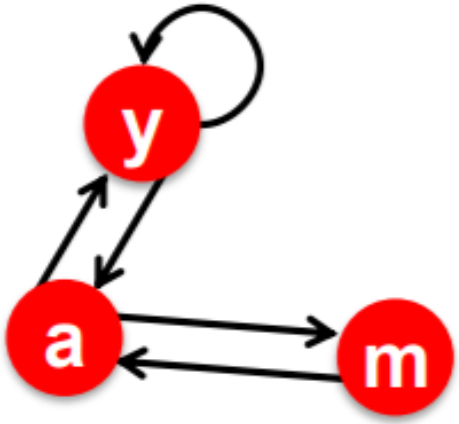
iter1

iter2

iter3

Converged

- Verify your result:



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{pmatrix} 6/15 \\ 6/15 \\ 3/15 \end{pmatrix}$$

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

Substitute your result into the flow equations to verify whether the equations hold or not

- Compare solution 1 an 2 (same results):

Solution 1: directly solve the linear equations

"Flow" equations:

$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$

$\sum_i r_i = 1$

We can solve these 4 linear equations to get the solutions:

$r_y = 0.4;\ r_a = 0.4;\ r_m = 0.2$

Solution 2: power iteration method

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \begin{bmatrix} 1/3 \\ 3/6 \\ 1/6 \end{bmatrix} \begin{bmatrix} 5/12 \\ 1/3 \\ 3/12 \end{bmatrix} \begin{bmatrix} 9/24 \\ 11/24 \\ 1/6 \end{bmatrix} \dots \begin{bmatrix} 6/15 \\ 6/15 \\ 3/15 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.4 \\ 0.2 \end{bmatrix}$$

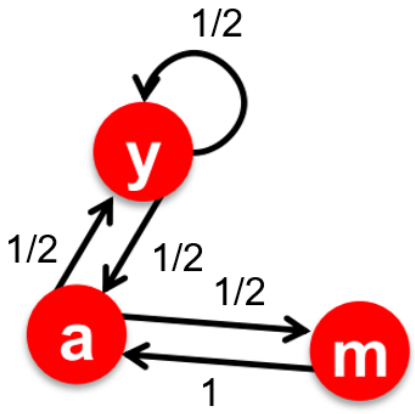
Initial value: 1/N (N=3) | iter1 | iter2 | iter3 | Converged

- Discussion
  - Power iteration method may need many iterations to converge.
  - Why we still need power iteration method?

- Discussion
  - Why we still need power iteration method?
  - It can handle large scale problems.
    - In practice, we have a huge transition matrix. Directly solving a large number of linear equations is computational expensive or maybe even intractable.
    - In power iteration method, we can control the number of iterations to get an approximate solution. Usually, the approximated solution will be good enough.

- Properties of the transition matrix: M
  - Also call stochastic adjacency matrix
  - M should be a column stochastic matrix:
    - each column sums to 1; all elements >=0

Each column in M indicates the weights of out-links for one node



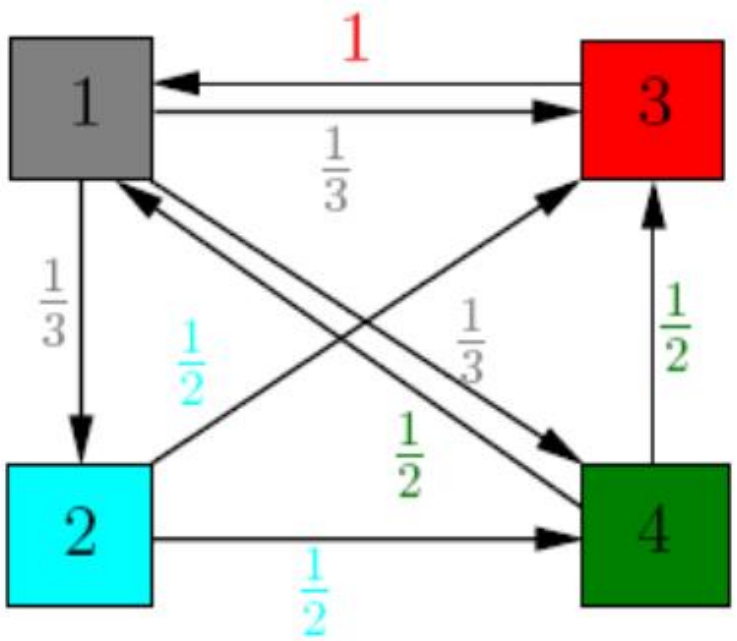
	r_y	r_a	r_m
r_y	½	½	0
r_a	½	0	1
r_m	0	½	0

Transition matrix M

The edge weight is defined by the number of out-links (d) $\frac{1}{d_i}$

All the out-links of one node share the same weight
-> non-zero cell will take the same value for each column
-> the sum of a column is: d* (1/d) = 1.

- More examples will be discussed in the tutorial class



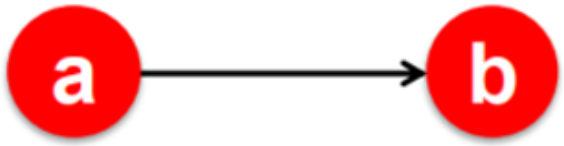
Page rank score for node 1,2,3,4:

$$\begin{pmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{pmatrix}$$

- PageRank extensions
  - “Dead-end” problem
  - “Spider-trap” problem
  - The Google matrix

- Two problems:
  - 1) the dead-end problem
    - Dead-end nodes
  - 2) the spider trap problem
    - Spider trap groups

1. The dead-end problem:
some nodes are dead ends (have no out-links)



Node b is a dead-end node

Transition Matrix: M

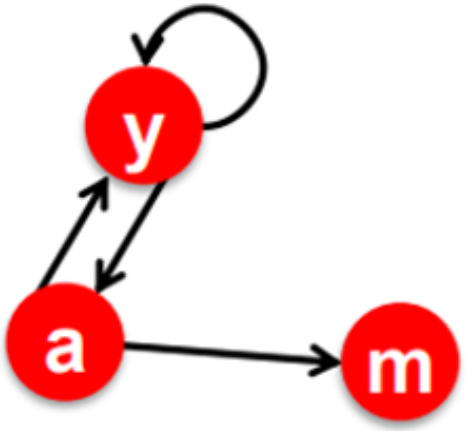
	a	b
a	0	0
b	1	0

Use the power iteration method:

	init	**Iter 1**	**Iter 2**
Node a	0.5	0	0
Node b	0.5	0.5	0

$$r^{(t+1)} = M \cdot r^{(t)}$$

The sum of the all node scores for each iteration is reducing (the sum should be 1 if there is no score leaking)



Transition Matrix: M

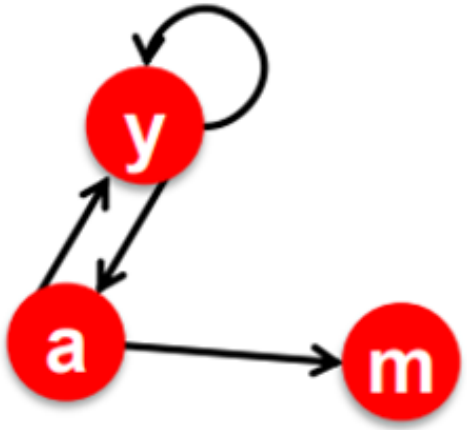
	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

Node m is a dead-end node

$$r^{(t+1)} = M \cdot r^{(t)}$$

	init	Iter 1	Iter 2	Iter 3	Iter 4	Iter 5	…	
Node y	1/3	2/6	3/12	5/24	8/48	13/96	…	0
Node a	1/3	1/6	2/12	3/24	5/48	8/96	…	0
Node m	1/3	1/6	1/12	1/24	3/48	5/96	…	0

Score leaking: the sum of each column is reducing.



Transition Matrix: M

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

Node m is a dead-end node

$$r^{(t+1)} = M \cdot r^{(t)}$$

Flow equations:

$$r_y = \frac{1}{2} r_y + \frac{1}{2} r_a$$

$$r_a = \frac{1}{2} r_y$$

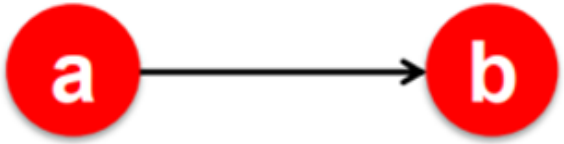
$$r_m = \frac{1}{2} r_a$$

$$r_y + r_a + r_m = 1$$

There is no solution

If only consider the top 3 equations, the solution is 0 for each node.

The dead-end problem:



	a	b
a	0	0
b	1	0

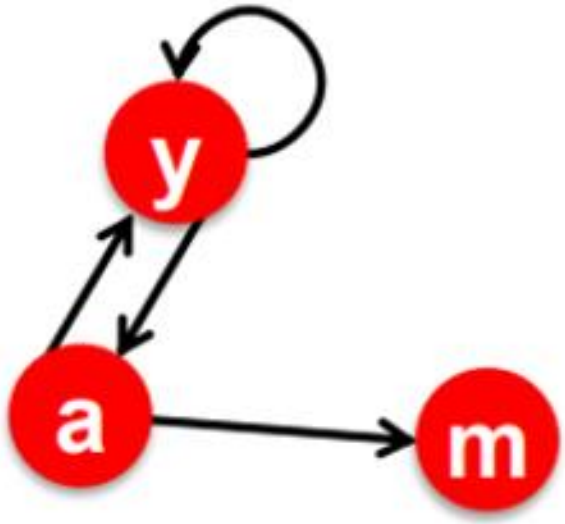
Transition Matrix: M

The problem of M:
It's not a column stochastic matrix, so it cannot meet the assumption of the PageRank algorithm.

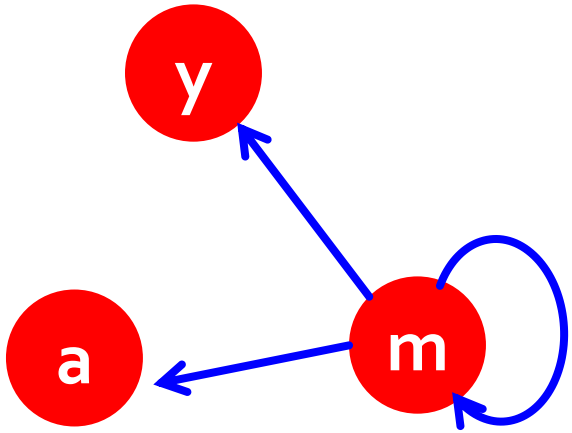
A column stochastic matrix should satisfy: each column sums to 1; all elements >=0 (stochastic here means probability)

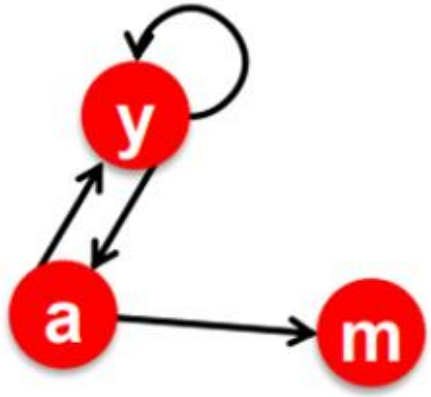
- Solution to Dead End:
  - Add teleport links for the dead-end nodes
    - teleport links:
      the links from the dead-end node to all other nodes
  - Fix the transition matrix by adding the teleport links

original links



Teleport links for node m



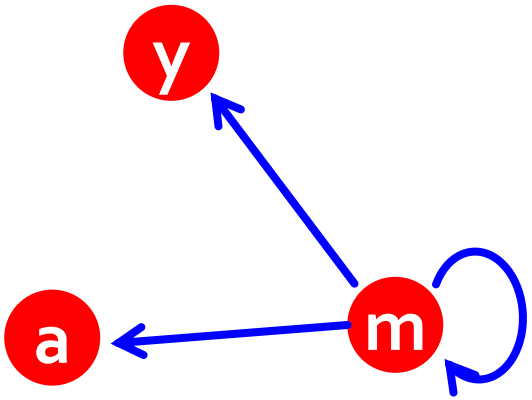


original links

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

Transition matrix M
(original links)

Teleport links for node m



	y	a	m
y	0	0	**1/3**
a	0	0	**1/3**
m	0	0	**1/3**

Transition matrix Q
(Teleport links)

teleport links: the links from the dead-end node to all other nodes

- Adding the teleport links to the original graph
  - Update the transition matrix

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

Transition matrix M (original links)

	y	a	m
y	0	0	**1/3**
a	0	0	**1/3**
m	0	0	**1/3**

Transition matrix Q (Teleport links)

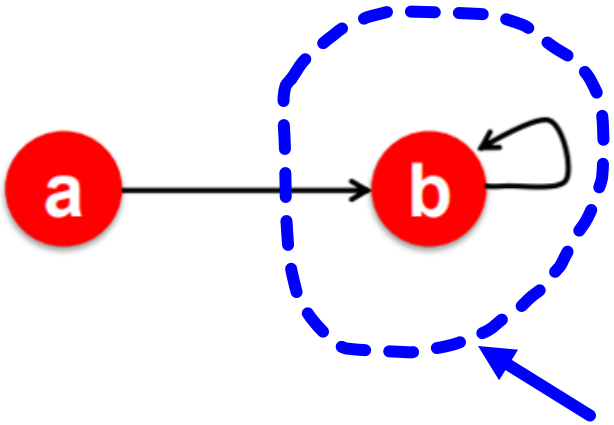
Update column m in the original matrix

New transition matrix

	y	a	m
y	½	½	⅓
a	½	0	⅓
m	0	½	⅓

Now the sum of column m is 1

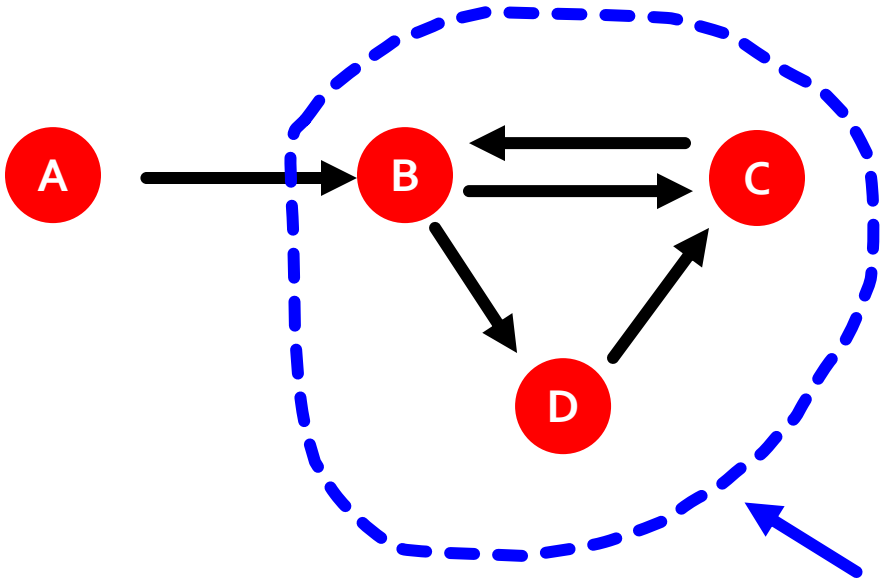
2. The spider trap problem:



A spider trap group: (a group of nodes)
all out-links are within the group

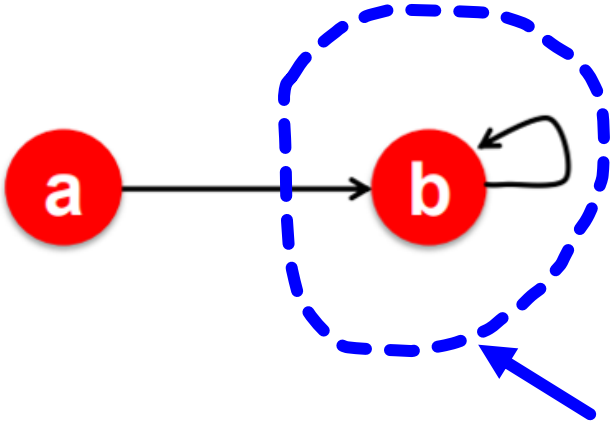
After entering the spider-trap group, the web user cannot go out of the group by navigating through the links

Another spider trap example:
(all out-links are within the group)



the group forms a spider trap problem

After entering the spider-trap group, the web user cannot go out of the group by navigating through the links



a spider trap

	a	b
a	0	0
b	1	1

Transition Matrix: M

Spider trap issue: eventually, the spider traps absorb all the importance (high rank scores)

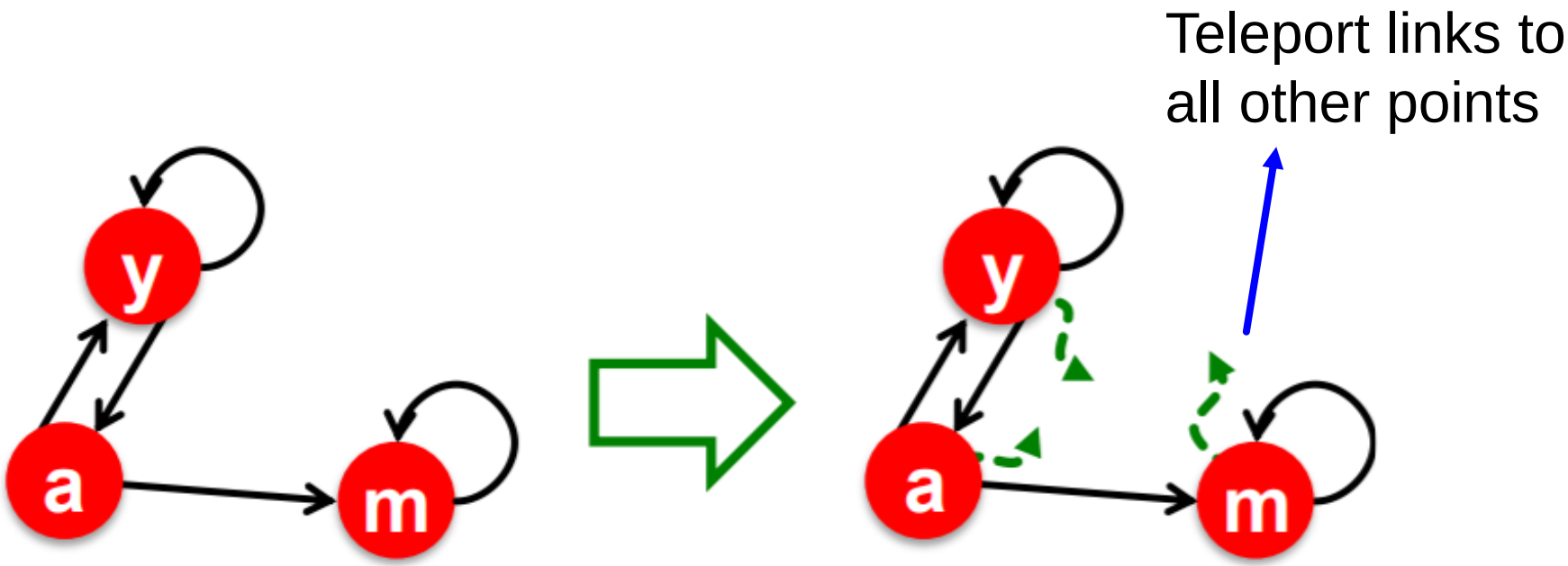
Use the power iteration method:

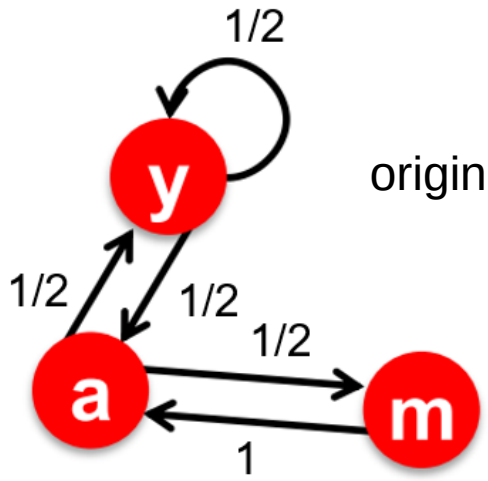
	init	**Iter 1**	**Iter 2**
Node a	0.5	0	0
Node b	0.5	1	1

$$r^{(t+1)} = M \cdot r^{(t)}$$

Solution:

Add special teleport links for every node
The teleport links will be separately handled

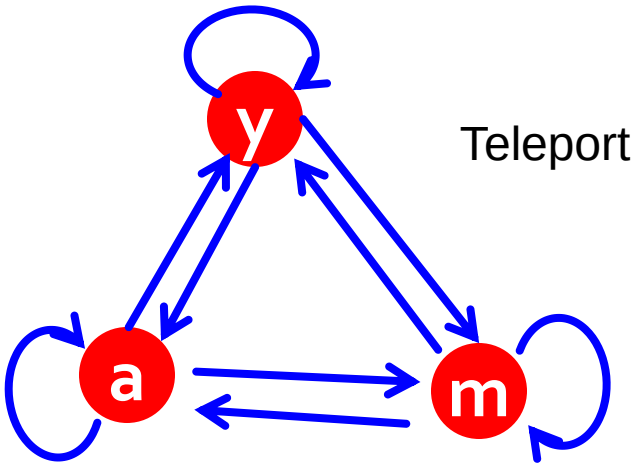




original links

	r_y	r_a	r_m
r_y	½	½	0
r_a	½	0	1
r_m	0	½	0

Transition matrix M
(original links)



Teleport links

	r_y	r_a	r_m
r_y	1/3	1/3	1/3
r_a	1/3	1/3	1/3
r_m	1/3	1/3	1/3

Transition matrix Q
(Teleport links)

PageRank – Google Matrix

Using the Google matrix as the new transition matrix

The Google matrix is constructed by two transition matrixes:

$$G = \beta\, M + (1 - \beta)\, Q$$

1. the original matrix 2. teleport matrix

β is a weighting parameter, in practice, $\beta = 0.8$ or 0.9

Q is an N by N matrix, where all elements are 1/N
N is the number of nodes in the graph

Use the Google matrix in the power iteration method:

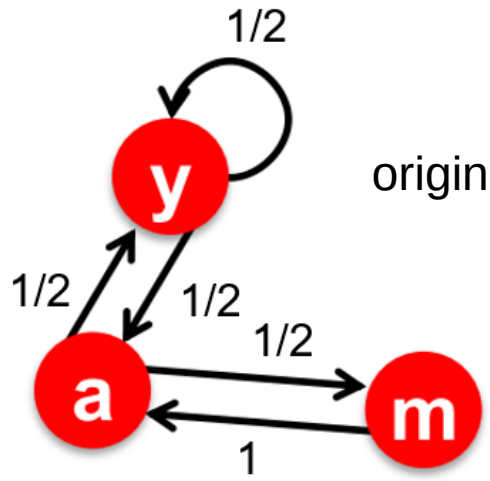
$$r^{t+1} = G \cdot r^{t}$$

PageRank – Google Matrix

PageRank – Google Matrix

The flow equation:

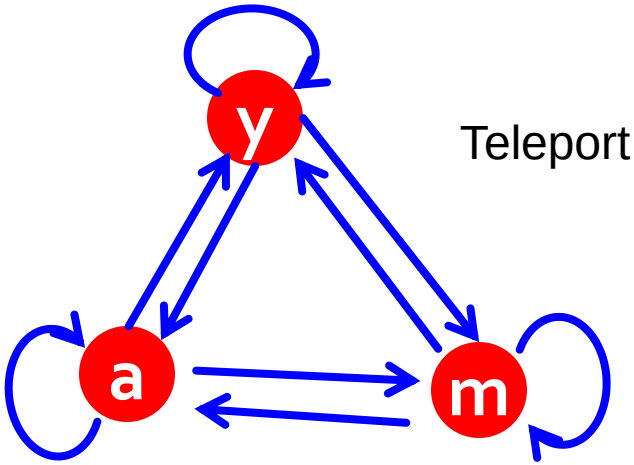
$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{N}$$



original links

	r_y	r_a	r_m
r_y	½	½	0
r_a	½	0	1
r_m	0	½	0

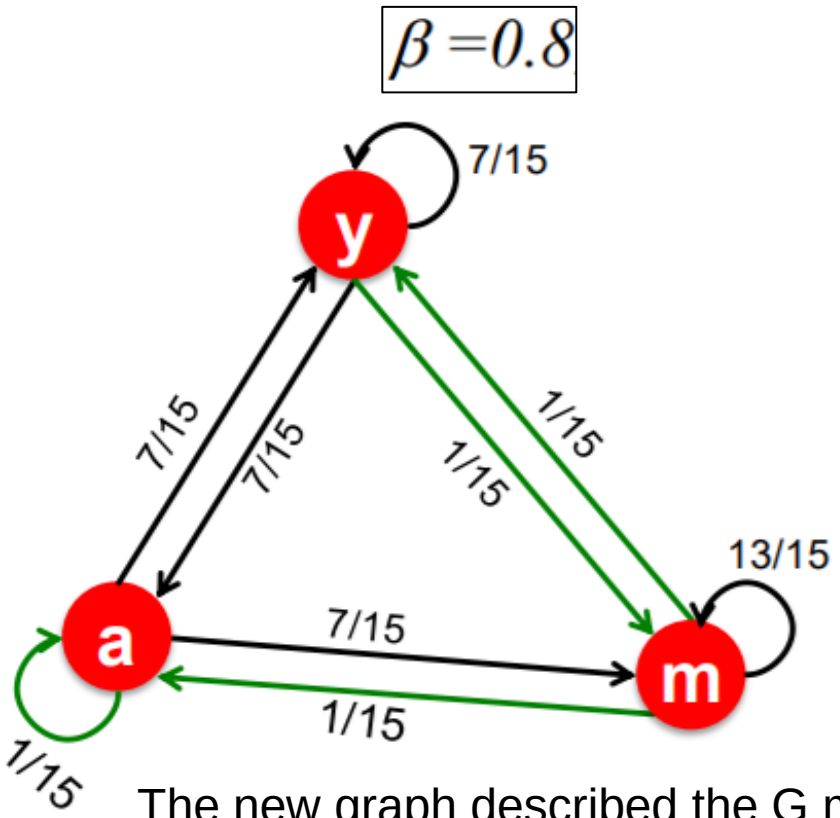
Transition matrix M
(original links)



Teleport links

	r_y	r_a	r_m
r_y	1/3	1/3	1/3
r_a	1/3	1/3	1/3
r_m	1/3	1/3	1/3

Transition matrix Q
(Teleport links)



Original links: **M**

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

Teleport links: Q

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

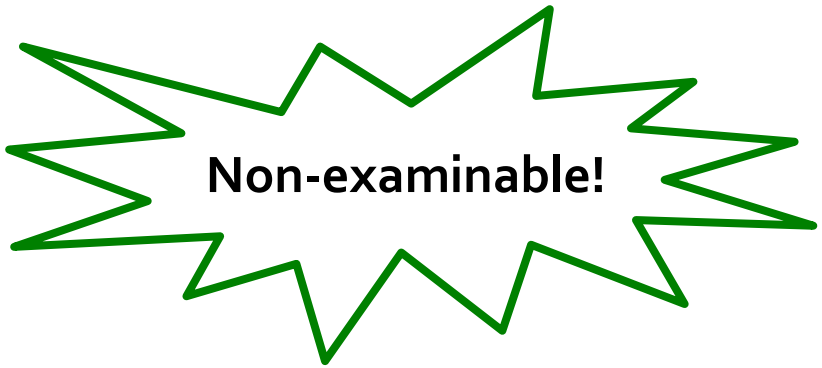
$$\begin{array}{c} \\ y \\ a \\ m \end{array} \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}$$

G

The new graph described the G matrix

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

Further discussions

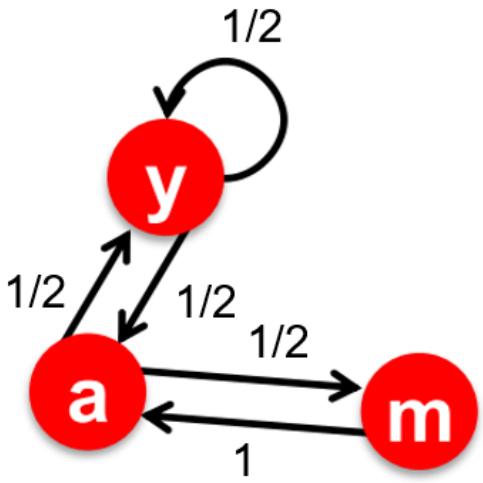


(The following topics are not examinable!)

- PageRank and random walk algorithm
- PageRank and eigenvector

Random walk

One random walk step:

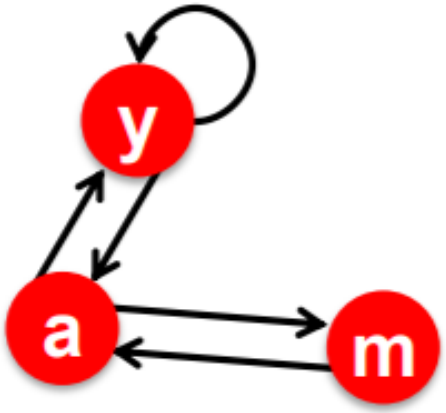


- A random walker on the graph:
  - At time t, the walker is at node i
  - At time $t + 1$, the walker follows the probabilities in the transition matrix to jump to another node.

	r_y	r_a	r_m
r_y	½	½	0
r_a	½	0	1
r_m	0	½	0

Transition matrix M

Random walk

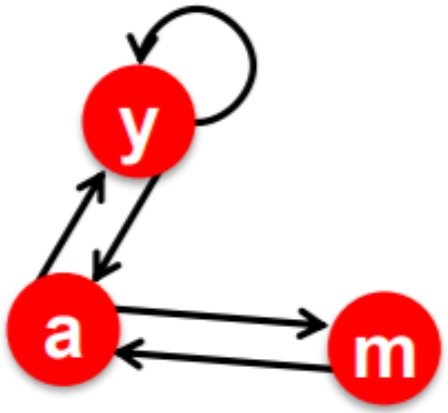


	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

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

A random walk problem:
Q: if a walker start at node **m**, after 3 random-walk steps, what is the location of the walker?
(calculate the probability of landing for each node)

Random walk



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

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

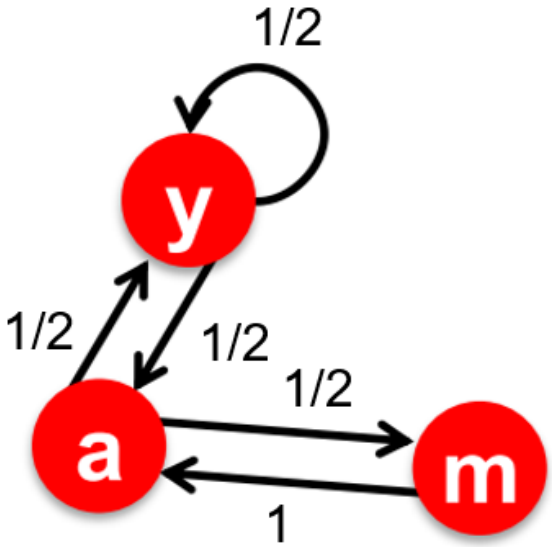
Solution:
We can change the initialization and use the power iteration method.

Starting from node m:

1st step:
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

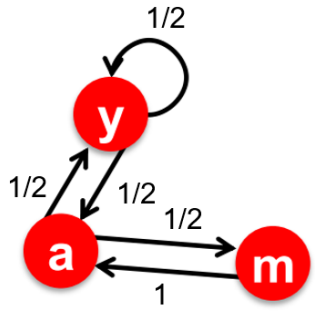
2nd step:
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 0 \\ 1/2 \end{bmatrix}$$

3rd step:
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \begin{bmatrix} 1/2 \\ 0 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 1/4 \\ 3/4 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.25 \\ 0.75 \\ 0 \end{bmatrix}$$



The **r** vector gives the probability of each node that the walker will visit after 3 random walk steps

Random walk and PageRank



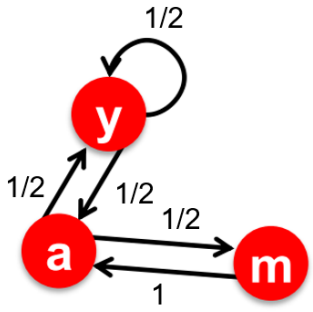
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix} \begin{pmatrix} 1/3 \\ 3/6 \\ 1/6 \end{pmatrix} \begin{pmatrix} 5/12 \\ 1/3 \\ 3/12 \end{pmatrix} \begin{pmatrix} 9/24 \\ 11/24 \\ 1/6 \end{pmatrix} \cdots \begin{pmatrix} 6/15 \\ 6/15 \\ 3/15 \end{pmatrix} = \begin{bmatrix} 0.4 \\ 0.4 \\ 0.2 \end{bmatrix}$$

Initial value: 1/N (N=3) iter1 iter2 iter3 Converged

PageRank is a random walk algorithm on a graph:

A walker randomly start at one node (equal probability for each node);
in each time step, the walker follow the probabilities in the transition matrix to jump to the next node.

Random walk and PageRank



$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \begin{bmatrix} 1/3 \\ 3/6 \\ 1/6 \end{bmatrix} \begin{bmatrix} 5/12 \\ 1/3 \\ 3/12 \end{bmatrix} \begin{bmatrix} 9/24 \\ 11/24 \\ 1/6 \end{bmatrix} \cdots \begin{bmatrix} 6/15 \\ 6/15 \\ 3/15 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.4 \\ 0.2 \end{bmatrix}$$

Initial value: 1/N (N=3) | iter1 | iter2 | iter3 | Converged

PageRank is a random walk algorithm on a graph:

Q: What is the probability of the walker's location after ***m*** time steps?

Perform the power iteration method with random initialization (1/n) for ***m*** steps and the PageRank score vector gives the probability for each node.

PageRank and Eigenvector

- **The flow equation:**

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

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{bmatrix} ½ & ½ & 0 \\ ½ & 0 & 1 \\ 0 & ½ & 0 \end{bmatrix} \begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix}$$

$\boldsymbol{r}$ $\quad$ $\boldsymbol{M}$ $\quad$ $\boldsymbol{r}$

the rank vector $\boldsymbol{r}$ is an eigenvector of the transition matrix $\boldsymbol{M}$ (with eigenvalue 1)
With the requirement: all rank score sum equals to 1
The converged rank vector **r** is called the stationary distribution

Definition of eigenvector:

$\lambda \boldsymbol{c} = \boldsymbol{A}\boldsymbol{c}$ ⟶ $\boldsymbol{c}$: eigenvector; λ: eigenvalue

- Further readings:
  - Stanford course CS224w
    - http://web.stanford.edu/class/cs224w/slides/04-pagerank.pdf
  - Lecture from Cornell
    - http://pi.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture3/lecture3.html