1.2.1 Solving the Flow Equations

Flow equations:

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

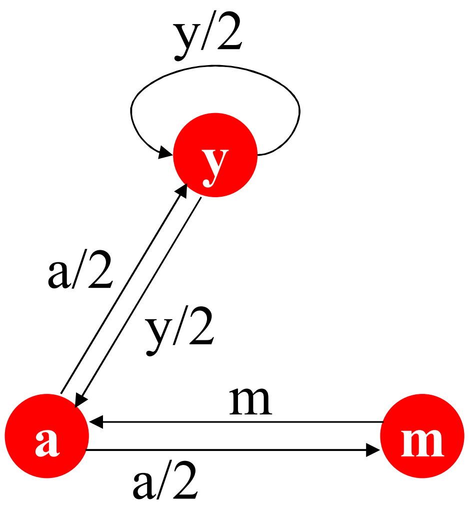
- **3 equations, 3 unknowns, no constants**
  - No unique solution
  - All solutions equivalent modulo the scale factor
- **Additional constraint forces uniqueness:**
  - $\boldsymbol{r_y + r_a + r_m = 1}$
  - **Solution:** $\boldsymbol{r_y = \frac{2}{5},\ r_a = \frac{2}{5},\ r_m = \frac{1}{5}}$
- **Gaussian elimination method (高斯消元法/高斯消去法) works for small examples, but we need a better method for large web-size graphs** ⇨ **We need a new formulation!**

1.2.2 PageRank: Matrix Formulation

华中科技大学 计算机科学与技术学院 School of Computer Science &Technology, HUST

❑1、Stochastic adjacency matrix(邻接矩阵) M

- Let page i has d_i out-links
- If $i \to j$, then $M_{ji} = 1/d_i$ else $M_{ji} = 0$
- $\boldsymbol{M}$ is a **column stochastic matrix,** columns sum to 1
- Note: $\boldsymbol{M}$ also is called Web transition matrix(Web转移矩阵) in some literature

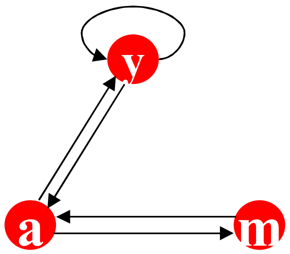


$$\boldsymbol{M} = \begin{array}{c c} & \begin{array}{ccc} y & a & m \end{array} \\ \begin{array}{c} y \\ a \\ m \end{array} & \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \end{array}$$

【备注】:
随机矩阵是这样一类方阵, 其元素为非负实数, 且行和或列和为1.
如果列和为1, 则称为**列随机矩阵**; 如果行和为1, 则称为行随机矩阵.
如果行和和列和都为1, 则称为双随机矩阵

1.2.2 PageRank: Matrix Formulation

- **2、Rank vector(秩向量)r:** vector with an entry per page
  - r_i is the importance score of page i
  - Initial, each page has $1/n$ importance score, when total n pages.
  - $\sum_i r_i = 1$



$$r = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

- **Now, the flow equations can be written**

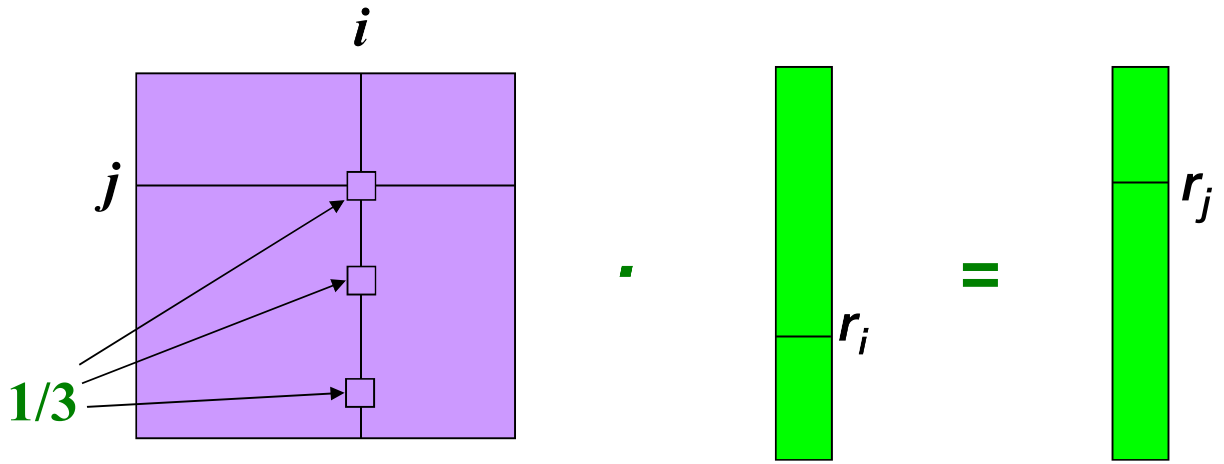
$$\boldsymbol{r} = \boldsymbol{M} \cdot \boldsymbol{r}$$

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

- **M fixed. How to calculate *r*?**

1.2.2 Example

- Remember the flow equation: $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
- Flow equation in the matrix form: $M \cdot r = r$
  - Suppose page *i* links to 3 pages, including *j*

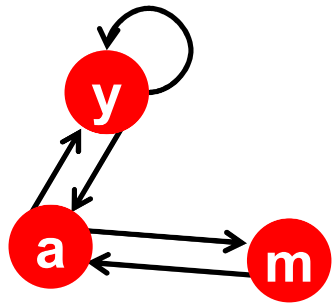


$M \cdot r = r$

Both two forms mean the same thing!

1.2.2 Example

- **With the given example, we can give the "flow" model, and the matrix formulation:**



(2) the matrix formulation

$$M = \begin{array}{c|c|c|c|} & y & a & m \\ \hline y & \frac{1}{2} & \frac{1}{2} & 0 \\ \hline a & \frac{1}{2} & 0 & 1 \\ \hline m & 0 & \frac{1}{2} & 0 \\ \hline \end{array}$$

(1) the "flow" model

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

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

$$r = M \cdot r$$

$$\begin{bmatrix} y \\ a \\ m \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 1 \\ 0 & \frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} y \\ a \\ m \end{bmatrix}$$

1.2.2 Eigenvector Formulation

华中科技大学 计算机科学与技术学院 School of Computer Science &Technology, HUST

- **The flow equations can be written**

$$r = M \cdot r$$

【备注线性代数相关知识】: $\boldsymbol{x}$ is an eigenvector (特征向量) with the corresponding eigenvalue (特征值) λ if: $\boldsymbol{Ax} = \boldsymbol{\lambda x}$

- So the **rank vector *r*** is an **eigenvector** (特征向量) of the stochastic web matrix ***M***
  - In fact, its first or principal eigenvector (主特征向量, 最大特征值对应的特征向量) with corresponding eigenvalue **1** (最大特征值为1)
    - Largest eigenvalue of ***M*** is 1, since ***M*** is column stochastic (columns sum to 1)

- **We can now efficiently solve for r! The method is called Power iteration** (幂迭代法)

1.2.2 Power Iteration Method

- **Given a web graph with *N* nodes, where the nodes are pages and edges are hyperlinks**
- **Power iteration:** a simple iterative scheme
  - Suppose there are *N* web pages
  - Initialize: $\mathbf{r}^{(0)} = [1/N, \ldots., 1/N]^T$
  - Iterate: $\mathbf{r}^{(t+1)} = \boldsymbol{M} \cdot \mathbf{r}^{(t)}$
  - Stop when $|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}|_1 < \varepsilon$

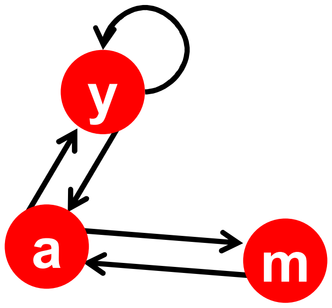
    $|\mathbf{x}|_1 = \sum_{1 \le i \le N} |x_i|$ is the $\mathbf{L_1}$ norm
    Can use any other vector norm, e.g., Euclidean

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

d_i …. out-degree of node i

1.2.2 PageRank: How to solve?

❑Example:



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$

❑Power Iteration:

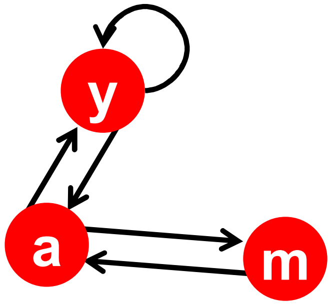
- Set $r_j = 1/N$
- **1:** $r'_j = \sum_{i \to j} \frac{r_i}{d_i}$
- **2:** $r = r'$
- Go to **1**

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

Iteration 0, 1, 2, …

1.2.2 PageRank: How to solve?

Example:



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$

Power Iteration:

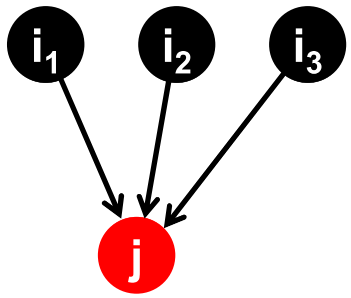
- Set $r_j = 1/N$
- **1:** $r'_j = \sum_{i \to j} \frac{r_i}{d_i}$
- **2:** $r = r'$
- Go to **1**

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

Iteration 0, 1, 2, …

1.2.3 Random Walk Interpretation

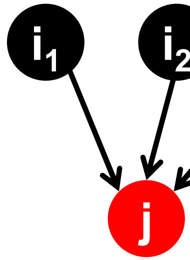
- **Imagine a random web surfer(随机冲浪者):**
  - At **any time** $\boldsymbol{t}$, surfer is on some page $\boldsymbol{i}$
  - At **time** $\boldsymbol{t+1}$, the surfer follows an out-link from $\boldsymbol{i}$ uniformly at random
  - Ends up on some page $\boldsymbol{j}$ linked from $\boldsymbol{i}$
  - Process repeats indefinitely



$$r_j = \sum_{i \to j} \frac{r_i}{d_{out}(i)}$$

- **Let:**
  - $\boldsymbol{p(t)}$ … vector whose $\boldsymbol{i}^{th}$ coordinate is the prob. that the surfer is at page $\boldsymbol{i}$ at time $\boldsymbol{t}$
  - So, $\boldsymbol{p(t)}$ is a probability distribution over pages(概率分布向量)

1.2.3 The Stationary Distribution



- **Where is the surfer at time *t+1*?**
  - Follows a link uniformly at random

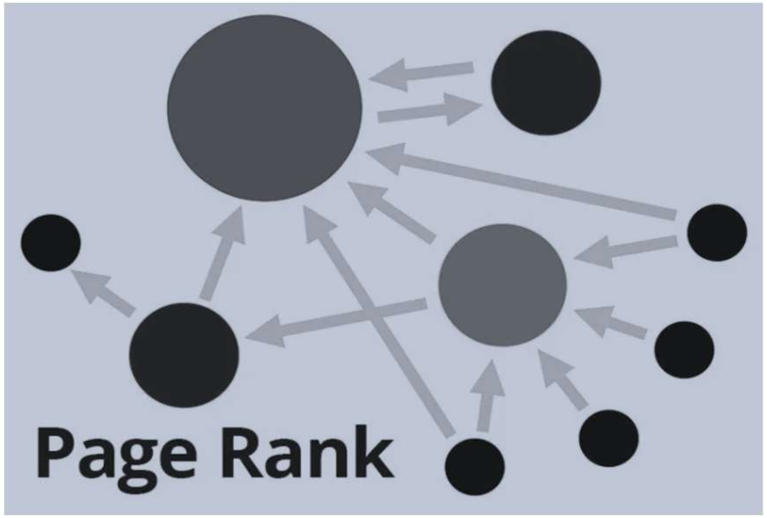
$$\boldsymbol{p}(\boldsymbol{t}+\mathbf{1}) = \boldsymbol{M} \cdot \boldsymbol{p}(\boldsymbol{t})$$

- Suppose the random walk reaches a state $\boldsymbol{p}(\boldsymbol{t}+\mathbf{1}) = \boldsymbol{M} \cdot \boldsymbol{p}(\boldsymbol{t}) = \boldsymbol{p}(\boldsymbol{t})$ then $\boldsymbol{p}(\boldsymbol{t})$ is **stationary distribution (稳态分布)** of a random walk
- **Our original rank vector** $\boldsymbol{r}$ satisfies $\boldsymbol{r} = \boldsymbol{M} \cdot \boldsymbol{r}$
  - **So, r is a stationary distribution for the random walk**

1.2.3 Existence and Uniqueness

- **A central result from the theory of random walks (a.k.a. Markov processes, 马尔科夫过程):**

For graphs that satisfy **certain conditions**, the **stationary distribution is unique** and eventually will be reached no matter what the initial probability distribution at time **t = 0**



Section 1.3: PageRank,

The Google Formulation

Content

1. PageRank Problems
2. Spider Traps → Teleports
3. Dead Ends → Teleports
4. Google Matrix

1.3.1 PageRank: Three Questions

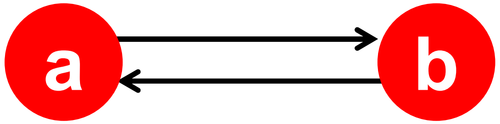
华中科技大学
计算机科学与技术学院
School of Computer Science &Technology, HUST

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

- ❑Does this converge(收敛)?
- ❑Does it converge to what we want?
- ❑Are results reasonable?

1.3.1 Does this converge?

❑Example:



	a	b
a	0	1
b	1	0

$$r = Mr$$

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

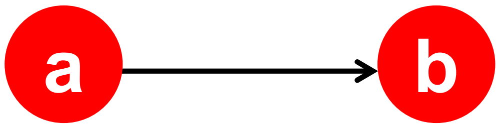
$$\begin{array}{c} r_a \\ r_b \end{array} = \begin{array}{cccc} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array}$$

iteration 0, 1, 2, …

无法收敛！这是一个典型的蜘蛛陷阱问题(后续马上会讲)

1.3.1 Does it converge to what we want?

- **Example:**



	a	b
a	0	0
b	1	0

$$r = Mr$$

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

$$\begin{array}{cccccc} r_a & = & 1 & 0 & 0 & 0 \\ r_b & & 0 & 1 & 0 & 0 \end{array}$$

iteration 0, 1, 2, …

收敛了但不是我们想要的结果！这是一个典型的死角问题(后续马上会讲)

1.3.1 Are results reasonable?

华中科技大学 计算机科学与技术学院 School of Computer Science &Technology, HUST

- 以下为同一个网页的部分内容截图示例:

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

每个链接被跳转的概率是一样么?

NO!

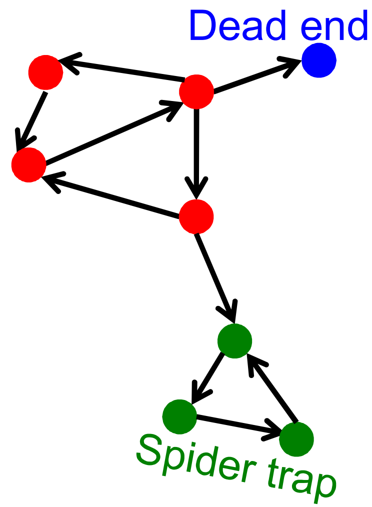
Random Surfer Model vs **Reasonable Surfer Model**

【备注】在我们后续的学习中还是采用基础的Random Surfer Model

1.3.1 PageRank: Problems

<u>**There maybe 2 problems:**</u>

- **(1)** Some pages are **dead ends** (死角问题, 死胡同问题, 或称终止点 have no out-links)
  - Random walk has "nowhere" to go to
  - Such pages cause importance to "leak out"

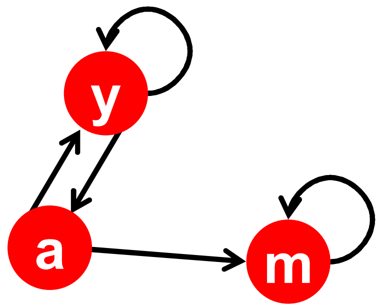


- **(2) Spider traps:** (蜘蛛陷阱问题，或称采集器陷阱, all out-links are within the group)
  - Random walked gets "stuck" in a trap
  - And eventually spider traps absorb all importance

1.3.2 Problem: Spider Traps

Power Iteration:

- Set $r_j = 1/N$
- $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - And iterate



m is a spider trap

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	1

$$\mathbf{r_y} = \mathbf{r_y}/2 + \mathbf{r_a}/2$$
$$\mathbf{r_a} = \mathbf{r_y}/2$$
$$\mathbf{r_m} = \mathbf{r_a}/2 + \mathbf{r_m}$$

Example:

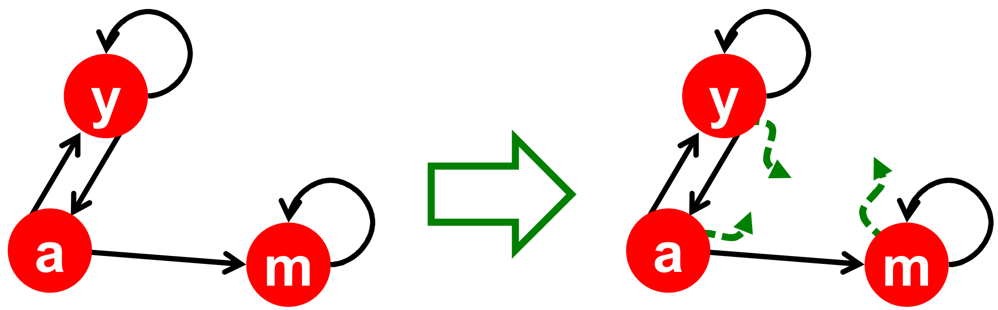
$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{array}{cccccc} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \dots & 0 \\ 1/3 & 3/6 & 7/12 & 16/24 & & 1 \end{array}$$

Iteration 0, 1, 2, …

All the PageRank score gets "trapped" in node m.

1.3.2 Solution: Teleports(随机跳转)!

- **The Google solution (called taxation method, 抽税法) for spider traps: At each time step, the random surfer has two options**
  - With prob. β (也叫阻尼系数), follow a link at random. Note, common values for β are in the range 0.8 to 0.9
  - With prob. **1-**β, jump to some random page

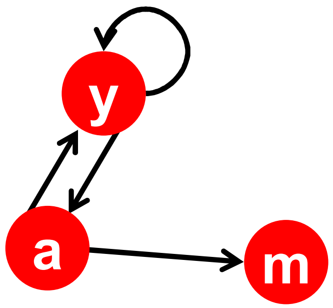


- **Surfer will teleport out of spider trap within a few time steps**

1.3.3 Problem: Dead Ends

❑Power Iteration:

- Set $r_j = 1/N$
- $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
  - And iterate



m is a dead end

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

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

❑Example:

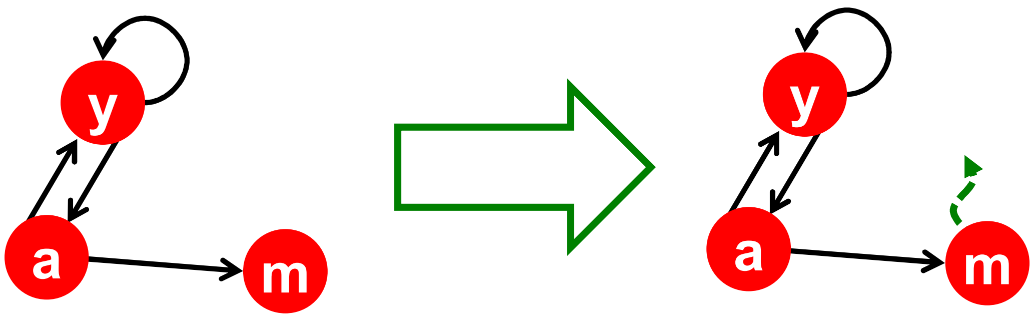
$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\ 1/3 & 1/6 & 1/12 & 2/24 & & 0 \end{matrix}$$

Iteration 0, 1, 2, …

Here the PageRank “leaks” out since the matrix is not stochastic.

1.3.3 Solution: Always Teleport!

- Taxation (抽税法, or Teleports): Follow random teleport links with probability 1.0 from dead-ends



- Adjust matrix accordingly

	y	a	m
y	½	½	0
a	½	0	0
m	0	½	0

⇒

	y	a	m
y	½	½	⅓
a	½	0	⅓
m	0	½	⅓

1.3.3 Why Teleports Solve the Problem?

- **Why are dead-ends and spider traps a problem and why do teleports solve the problem?**
- **Spider-traps** are not a problem, but with traps PageRank scores are **not** what we want
  - **Solution:** Never get stuck in a spider trap by teleporting out of it in a finite number of steps’
- **Dead-ends** are a problem
  - The matrix is not column stochastic so our initial assumptions are not met
  - **Solution:** Make matrix column stochastic by always teleporting when there is nowhere else to go

1.3.4 Solution: Random Teleports

- **Google's solution (Taxation, 抽税法) that does it all:** At each step, random surfer has two options:
  - With probability β, follow a link at random
  - With probability $1\text{-}\beta$, jump to some random page

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

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

d_i … out-degree of node i

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

1.3.4 The Google Matrix

华中科技大学 计算机科学与技术学院 School of Computer Science &Technology, HUST

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

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

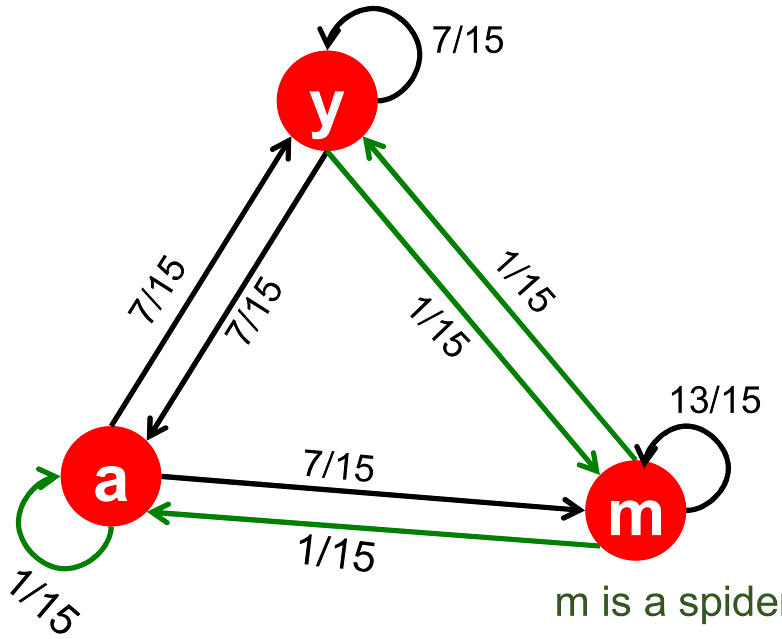
- **The Google Matrix *A*:**

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

$[1/N]_{NxN}$…N by N matrix where all entries are 1/N

- **What is β?** In practice $\beta = 0.8, 0.9$ (make 5 steps on avg., jump)
- **We have a recursive problem: $\boldsymbol{r} = \boldsymbol{A} \cdot \boldsymbol{r}$, and the Power iteration method still works!**

1.3.4 Random Teleports ($\beta = 0.8$)



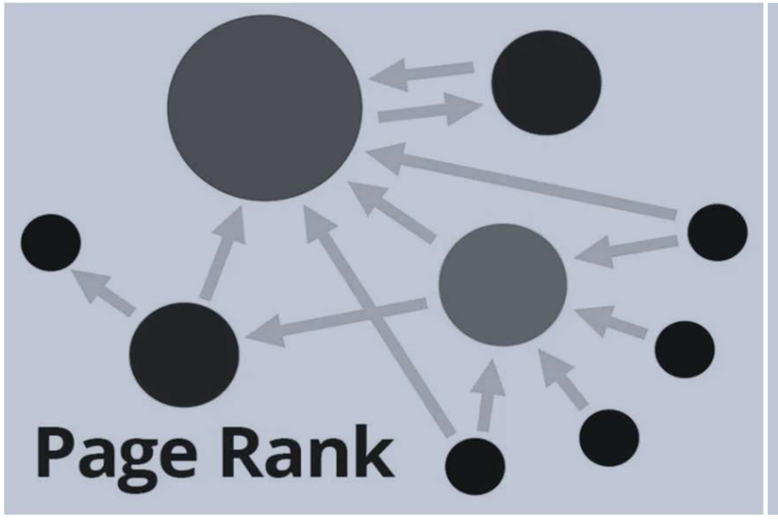
m is a spider trap

$$0.8 \underbrace{\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix}}_{\mathbf{M}} + 0.2 \underbrace{\begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}}_{[1/N]_{NxN}}$$

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

$$\begin{bmatrix} r_y \\ r_a \\ r_m \end{bmatrix} = \begin{matrix} 1/3 & 0.33 & 0.24 & 0.26 & & 7/33 \\ 1/3 & 0.20 & 0.20 & 0.18 & \dots & 5/33 \\ 1/3 & 0.46 & 0.52 & 0.56 & & 21/33 \end{matrix}$$

Previously, all the PageRank score gets “trapped” in node m.
But now taxation solves it!



Section 1.4: Computing PageRank

Content

1. The Complete PageRank Algorithm
2. Basic Update Step
3. Block-based Update Step
4. Block-Stripe Update Step

1.4.1 Computing PageRank

- **Key step is matrix-vector multiplication**
  - $r^{new} = A \cdot r^{old}$

$$A = \beta \bullet M + (1-\beta)\,[1/N]_{N \times N}$$

- **Easy if we have enough main memory to hold A, r^{old}, r^{new}**

$$A = 0.8 \begin{bmatrix} ½ & ½ & 0 \\ ½ & 0 & 0 \\ 0 & ½ & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

$$= \begin{bmatrix} 7/15 & 7/15 & 1/15 \\ 7/15 & 1/15 & 1/15 \\ 1/15 & 7/15 & 13/15 \end{bmatrix}$$

- **Say N = 1 billion (十亿) pages**
  - We need 4 bytes for each entry (say)
  - r^{old}, r^{new} : 2 billion entries for vectors, approx 8GB
  - A: Matrix **A** has N^2 entries
    - 10^{18} is a large number!

【备注】观察矩阵A和M可知，
Matrix **A** dense matrix!
Matrix **M** sparse matrix