

# MPEI 2024-2025

PageRank

# Como descobrir informação na web?

- Primeiras tentativas:

- Directórios da Web

- Listas criadas por humanos

- Exemplos:

- Yahoo, DMOZ, LookSmart



# Como descobrir informação na web?

- Segunda geração: **Web Search**
  - **Information Retrieval** :  
Procura de documentos relevantes num conjunto pequeno e de confiança
    - Artigos de jornais, Patentes, etc.
- **MAS:** A Web é gigantesca, cheia de documentos sobre os quais não temos garantias de ser de confiança, cheia de SPAM, etc.

# Os primeiros motores de procura

- Baseavam-se em percorrer (crawl) a web e **listar os termos** (palavras ou outras sequências de caracteres excluindo os espaços) de cada página, num **índice invertido**



**SAPO**

- Um índice invertido/inverso (**inverted index**) é uma estrutura de dados que torna simples, **dado um termo, descobrir** (apontar para) **todas as páginas em que o termo ocorre**.
  - Assunto da área de Information Retrieval

# Ataques de spam

- Rapidamente estes primeiros [motores de procura](#) foram atacados.



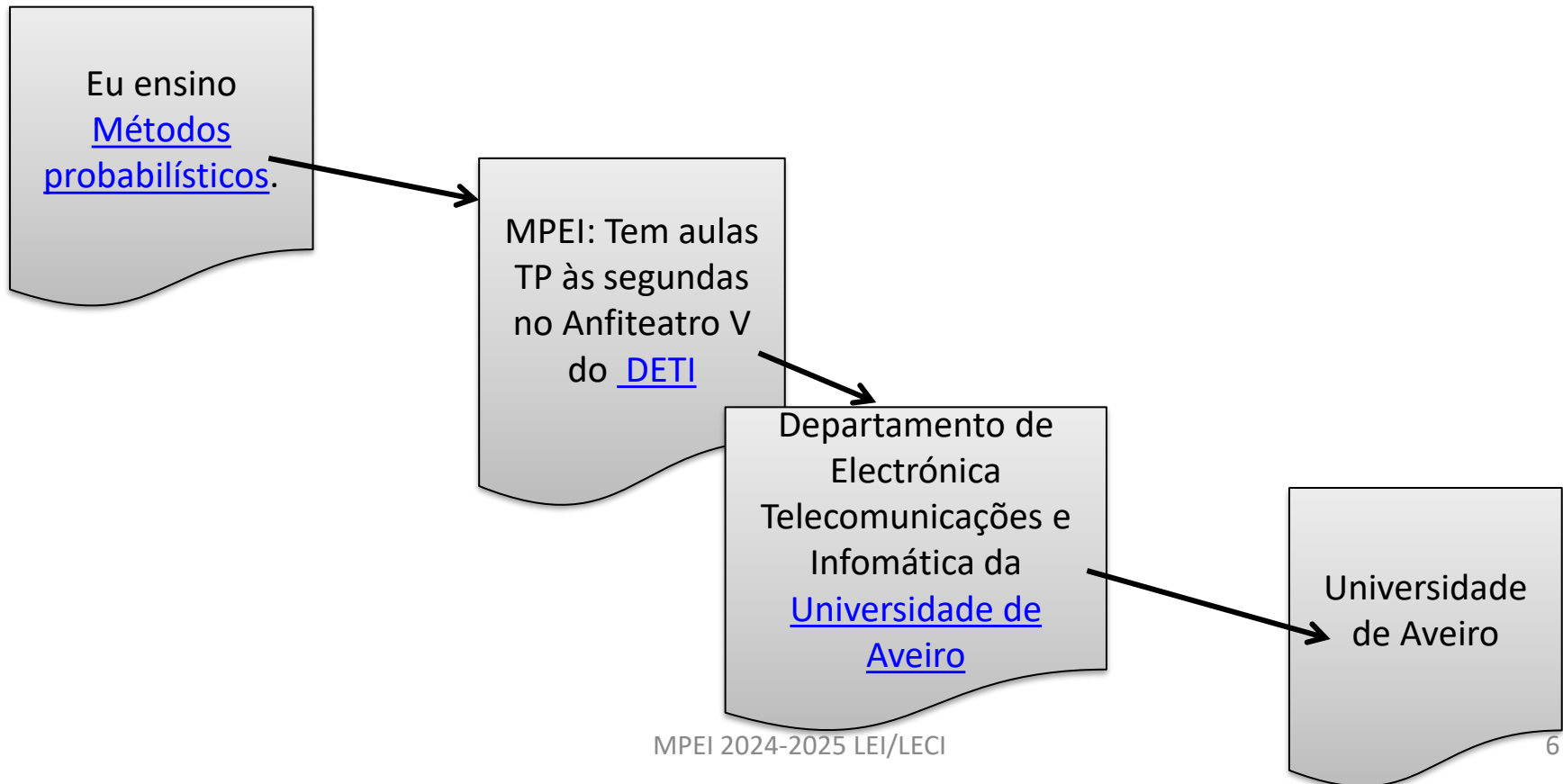
SAPO



- Sendo **sensíveis às palavras nas páginas**, facilmente os detentores de páginas com menos escrúpulos podiam **inflacionar a importância das suas páginas**:
  - **Adicionando muitas cópias de uma ou várias palavras** ao conteúdo da página
    - e tornando essa parte invisível quando mostrada num browser
  - Usando o motor de procura para saber a página mais importante segundo o seu algoritmo e copiando o seu conteúdo para as suas páginas
    - mantendo esta parte invisível no browser

# A Web como um Grafo

- Nós /nodos/vértices : Páginas Web
- Ligações/arcos: Hyperlinks



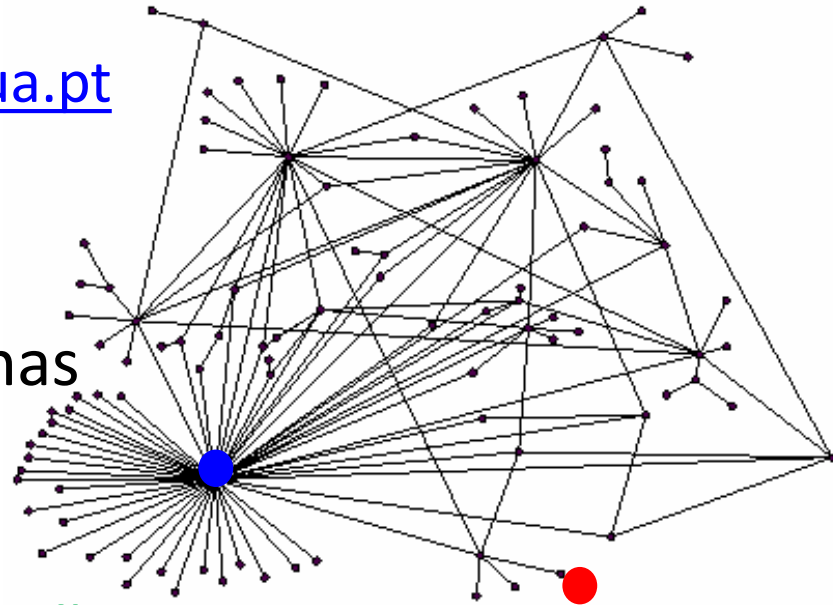
# Estrutura do grafo

- As páginas da web Não são todas igualmente “importantes”

[www.ze-ninguem.com](http://www.ze-ninguem.com) vs. [www.ua.pt](http://www.ua.pt)

- Existe um grande diversidade nas ligações

- Ideia: Usar a estrutura das ligações para saber quais as páginas “importantes”



# Contribuição da Google

- Não calcular a relevância das páginas apenas com base no conteúdo  
usar também informação sobre as ligações a essa página
  - Desenvolvendo e patenteando o Pagerank
  - Que começou como um projecto de investigação



# Ideia Base - Passeios aleatórios

- Simular onde passeios aleatórios pelas páginas de surfistas aleatórios (*random surfers*) tendem a passar mais se se escolherem aleatoriamente os links de saída de uma página (em que se encontram)
  - Começando numa página aleatória
  - E permitindo que o processo se repita muitas vezes
- As páginas visitadas por muitos passeios (ou surfers) serão “mais importantes” do que páginas raramente visitadas

# Ideia Base - Passeios aleatórios

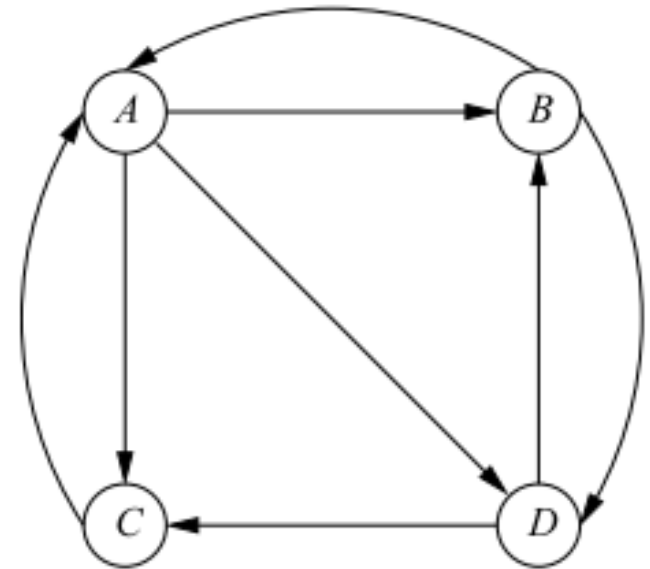
- O Google dá preferência a páginas mais importante ao decidir quais as páginas a mostrar primeiro em resposta a um *query*
- Mas obviamente as páginas têm de conter os termos ...

# Versão base 'ideal'

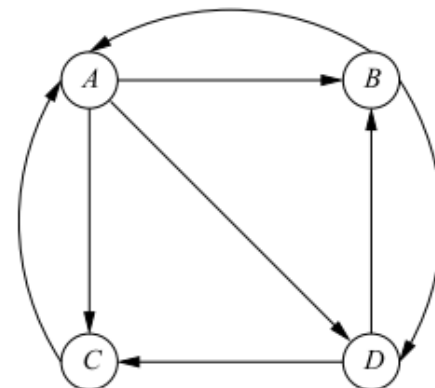
- Consideremos a web como um **grafo orientado** (*directed*) em que:
  - as **páginas são os nodos** (ou vértices)
  - existe um arco (ou **ligação**) da página  $P_1$  para a página  $P_2$  se existe um ou mais links de  $P_1$  para  $P_2$

## Exemplo:

- Rede muito pequena: apenas 4 páginas
- A página A tem links para as outras 3
- A página B tem ligações apenas para a A e a D;
- A página C tem apenas um link, para a A
- E a página D tem links apenas para B e C



# Versão base (cont.)



- Suponhamos que o surfista aleatório começa na página A:
- Existem links para B, C e D, logo o surfista estará de seguida numa dessas 3 páginas, com probabilidade  $1/3$  [ 1 a dividir pelo nº de links de saída]
  - E probabilidade zero de estar em A
- O surfista aleatório B terá, no próximo passo, probabilidade  $1/2$  de estar em A,  $1/2$  de estar em D e 0 de estar em C

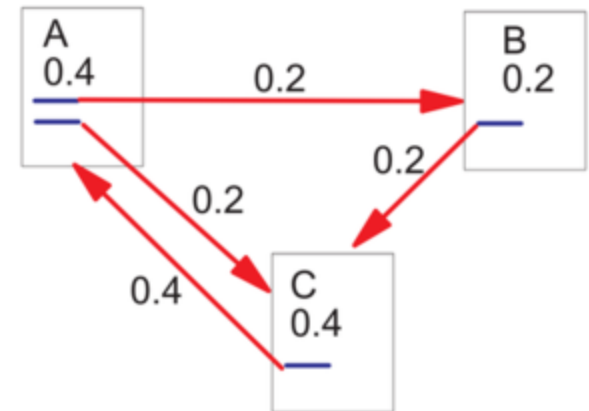
# Definição de pagerank

- O PageRank é uma função/algoritmo que atribui um número real a cada página da Web  
(ou a porção dela que foi processada e as ligações obtidas)
  - Designamos esse número por pagerank
  - Quanto maior é o valor mais “importante” é a página.
- Baseia-se na ideia dos random surfers
- Não existe propriamente um algoritmo fixo, havendo variações do algoritmo
  - Que podem dar valores diferentes de pagerank

# pagerank

- O pagerank ( $r$ ) de uma página  $P_j$  é, por **definição**:

$$r(P_j) = \sum_i \frac{r(P_i)}{d_i}$$



- sendo:
  - $i$  o índice das **páginas que apontam para  $P_j$**
  - $d_i$  o número de páginas para as quais  $P_i$  aponta ou seja, número de links de saída

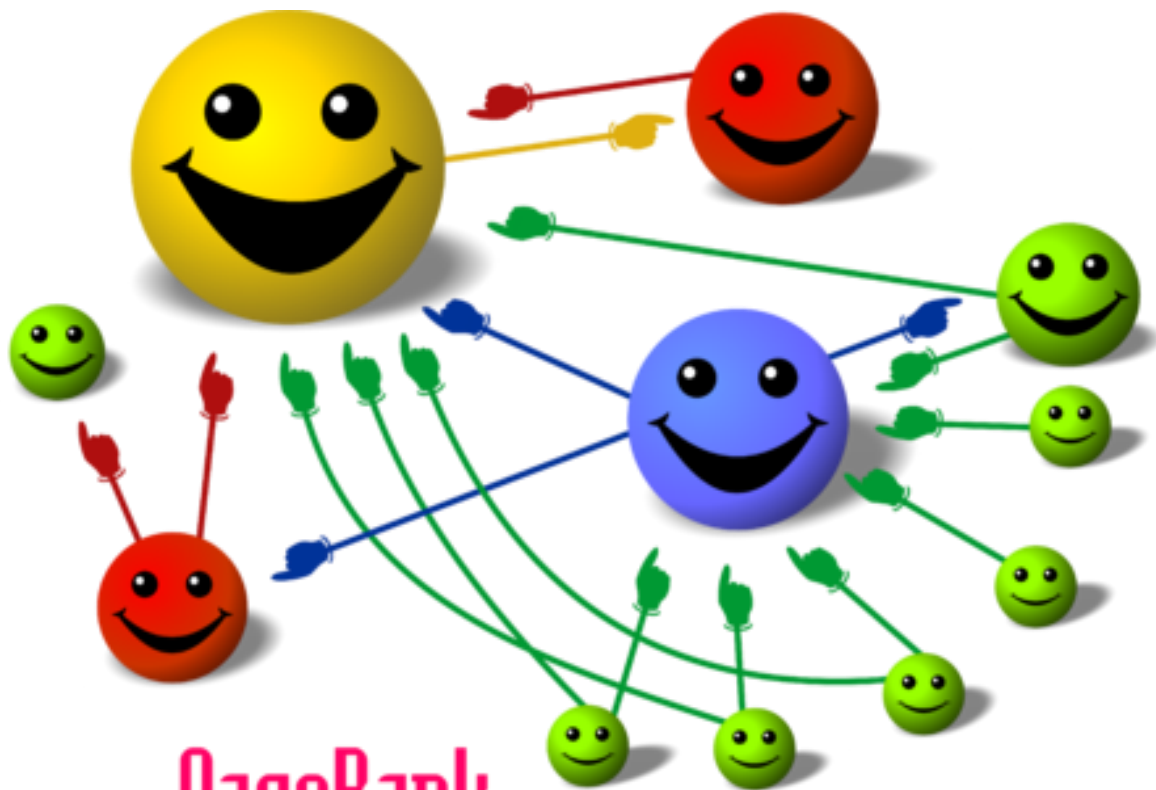
# Cálculo do pagerank

- O pagerank de uma página depende do pagerank das páginas que têm links para ela
  - identificadas pelo índice  $i$

- O que sugere um cálculo iterativo

$$r_{k+1}(P_j) = \sum_i \frac{r_k(P_i)}{d_i}$$

- A condição inicial é  $r_0(P_i) = 1/n$ , com  $n$  igual ao número de páginas



## PageRank

- Uma simplificação do sistema do PageRank,
- Cada bola representa uma página e o tamanho de cada uma a sua importância (PageRank).
- Quanto maior a bola, mais valor tem o seu voto:
- Repare que a bola superior vermelha é grande mesmo recebendo só um voto, pois o voto que ela recebe, da bola maior amarela, tem mais valor

DE: <https://pt.wikipedia.org/wiki/PageRank>.



# Forma matricial

- Definindo a **matriz de hyperlinks**  $H$  como

- $$H_{ji} = \begin{cases} \frac{1}{d_i} & , \text{se existir link de } i \text{ para } j \\ 0 & , \text{caso contrário} \end{cases}$$

- Teremos  $r^{(k+1)} = H r^{(k)}$

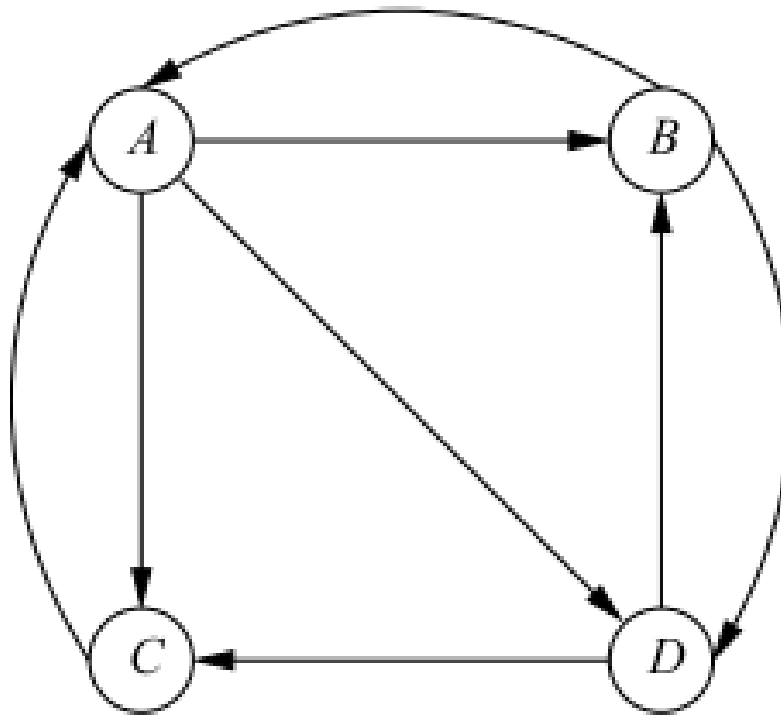
– Sendo  $r^{(k)}$  o vector com *pageranks* na iteração  $k$

# Forma matricial

- A matriz  $H$  pode ser interpretada como contendo as probabilidades de transição entre páginas (os estados)
- Em consequência pode aplicar-se o que aprendemos sobre Cadeias de Markov para:
  - calcular probabilidades após múltiplas transições
  - estudar o comportamento quando o número de transições (iterações) tende para infinito
  - etc .

# Matriz para o nosso exemplo ?

- Qual será então a matriz  $H$  para a nossa mini web ?



# Solução

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

- Acertaram ?

# Situação Limite

- Sabemos que a distribuição dos pageranks atingirá um **estado estacionário**, em que  $r = Hr$ 
  - Pelo menos em certas condições:
    - O grafo ser fortemente ligado, sendo possível ir de qualquer página para qualquer página
    - Não existirem becos sem saída (*dead ends*)
      - páginas que não têm links de saída
- O limite é atingido quando multiplicando os pageranks por  $H$  mais uma vez a distribuição de pageranks não se altera

# Aplicando ao nosso exemplo

- Aplicando  $r^{(k+1)} = H r^{(k)}$  sucessivamente  
– e iniciando com  $1/n$

$$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}, \begin{bmatrix} 9/24 \\ 5/24 \\ 5/24 \\ 5/24 \end{bmatrix}, \begin{bmatrix} 15/48 \\ 11/48 \\ 11/48 \\ 11/48 \end{bmatrix}, \begin{bmatrix} 11/32 \\ 7/32 \\ 7/32 \\ 7/32 \end{bmatrix}, \dots, \begin{bmatrix} 3/9 \\ 2/9 \\ 2/9 \\ 2/9 \end{bmatrix}$$

# Comentário

- Esta diferença em probabilidade é pequena
- Mas na web real, com bilhões de páginas the grande variedade de importância, a verdadeira probabilidade de uma página como [www.amazon.com](http://www.amazon.com) é ordens de magnitude superior à probabilidade de outras páginas, como uma página pessoal



# Questões

- É mesmo assim tão simples ?
- Converge sempre?
- Converge para o que queremos?
- Os resultados são razoáveis?

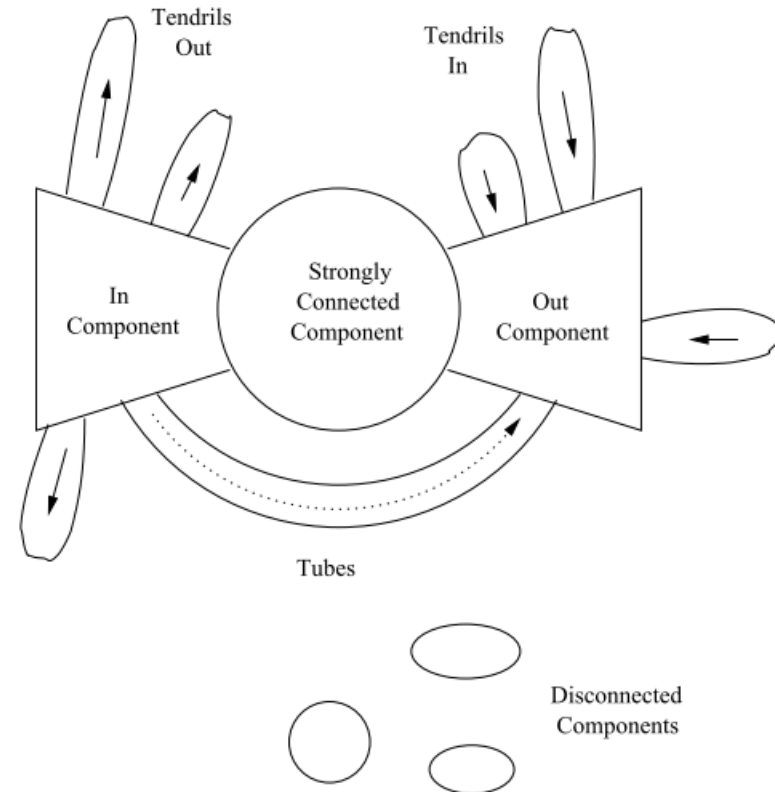


# Estrutura da web

- Será a **web tão fortemente ligada** como o nosso exemplo ?
- Seria bom que fosse...
- Mas, na prática, não é
  - Pelo menos não na sua totalidade

# Estrutura da web

- Um estudo antigo da web revelou a estrutura à direita
- Existe uma parte fortemente ligada
  - o Strongly Connected Component-SCC
- Mas também muitas páginas com:
  - Ligações ao SCC mas às quais não é possível chegar a partir do SCC
    - in-component
  - Ligações a partir do SCC mas sem forma de chegar ao SCC
    - out-component
  - Ligações do in-component mas incapazes de aceder a esse componente
  - Etc ..



# Isto traz problemas

Temos **dois tipos de problemas** que têm de ser resolvidos

1. Becos sem saída (*dead ends*)

2. Grupos de páginas que têm links de saída mas apenas para esse grupo, impedindo a ida para outras páginas

— Estas estruturas são chamadas de *spider traps*

- Porquê ?

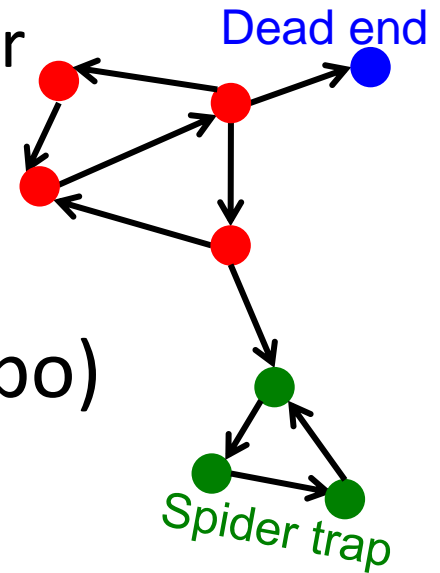
A spider is a program run by a [search engine](#) to build a summary of a website's content (*content index*). Spiders create a text-based summary of content and an address ([URL](#)) for each webpage.



# Problemas (continuação)

## 1. **Dead ends** ( sem links de saída)

- Passeio aleatório não tem para onde ir



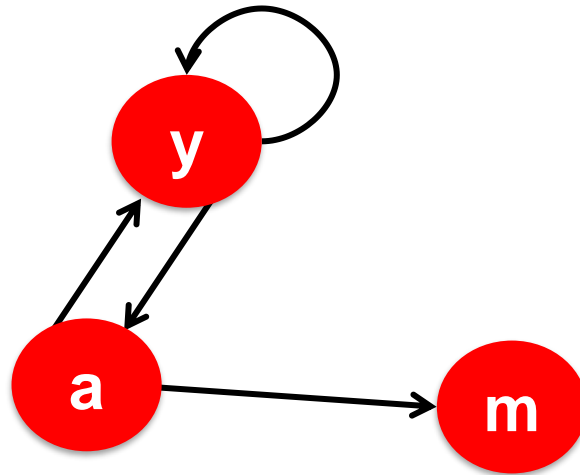
## 2. Spider traps:

(todos os links de saída para o grupo)

- O passeio aleatório fica “preso” na armadilha (trap)

- Qual o efeito nos pageranks ?

# Dead Ends – Exemplo



	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	0
a	$\frac{1}{2}$	0	0
m	0	$\frac{1}{2}$	0

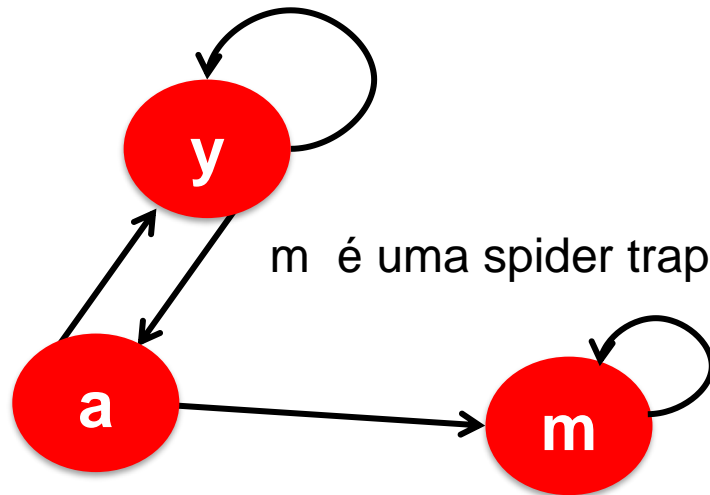
## • Exemplo:

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

Iteração 0, 1, 2, ...

O PageRank “desaparece” pois a matriz não é estocástica.

# Spider Traps – Exemplo



	y	a	m
y	$\frac{1}{2}$	$\frac{1}{2}$	0
a	$\frac{1}{2}$	0	0
m	0	$\frac{1}{2}$	1

## • Exemplo:

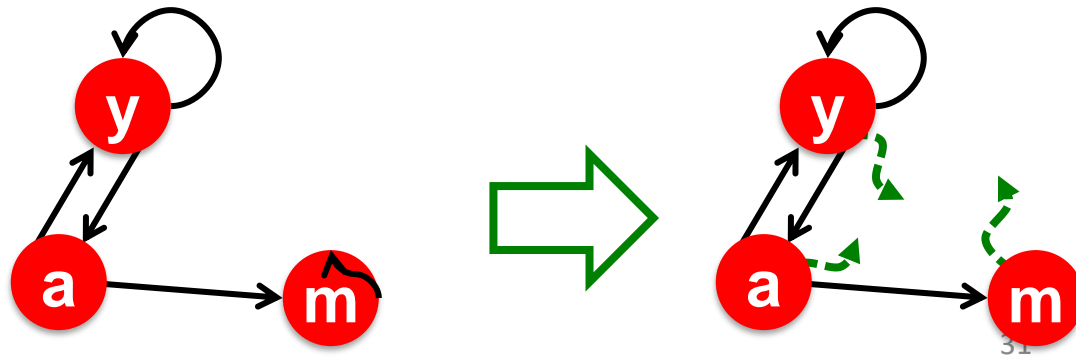
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{pmatrix} 1/3 & 2/6 & 3/12 & 5/24 & & 0 \\ 1/3 & 1/6 & 2/12 & 3/24 & \dots & 0 \\ 1/3 & 3/6 & 7/12 & 16/24 & & 1 \end{pmatrix}$$

Iteração 0, 1, 2, ...

O PageRank é todo “apanhado” pelo nó m.

# Solução para *spider traps*

- Em cada passo, **o surfista aleatório ter duas opções**
  - Com probabilidade  $\beta$ :
    - **Seguir um link** aleatoriamente
  - Com probabilidade  $1-\beta$ ,
    - **Saltar aleatoriamente para uma página qualquer**

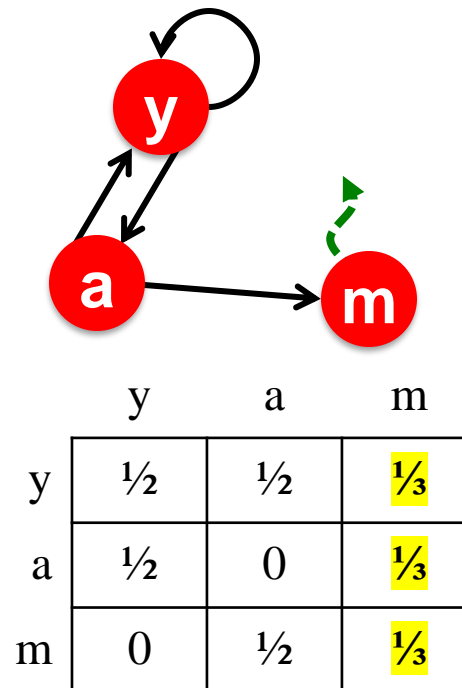
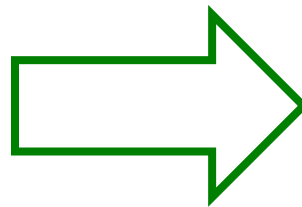
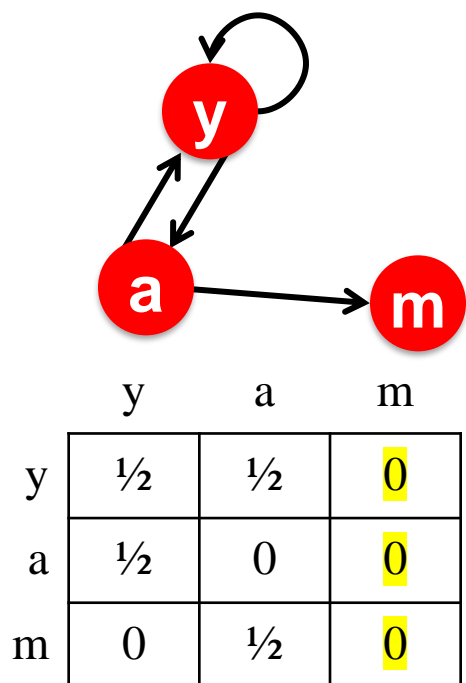


- **O surfista teletransporta-se para fora da *spider trap* ao fim de alguns passos**
- Valores usuais para  $\beta$ : intervalo 0.8 to 0.9



# Solução para *dead ends*

- Teletransportar (teleport) sempre
- **Implica ajustar a matriz** por forma a:
  - seguir um link com probabilidade  $1/n$





# Os teletransportes resolvem os dead-ends ?

- **SIM**
- **Deixa de haver dead-ends**
  - Existe sempre para onde ir
  - Matriz volta a ser estocástica



# Solução da Google

- Em cada passo, **o surfista aleatório tem duas opções**
  - Com probabilidade  $\beta$ :
    - **Seguir um link** aleatoriamente
  - Com probabilidade  $1-\beta$ :
    - **Saltar aleatoriamente para uma página qualquer**
- **PageRank equation** [Brin-Page, 98]

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

# A Matriz da Google

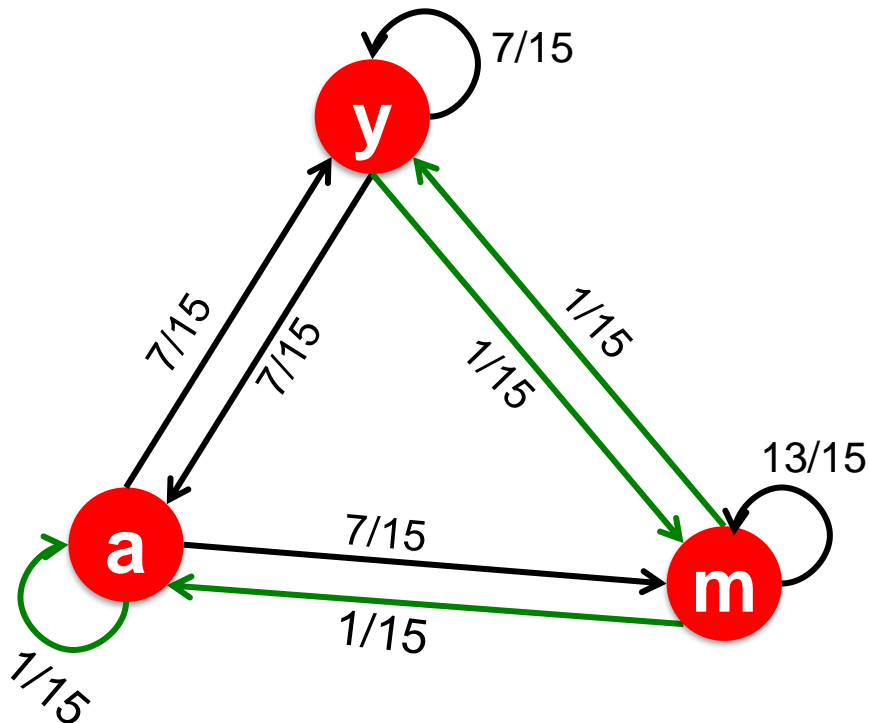
- **Matriz da Google:**

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

$[1/N]_{N \times N}$ ...matriz N por N com todas entradas iguais a  $1/N$

- Temos um **problema recursivo**:  $r = A \cdot r$
- A que se pode aplicar o método das potências (Power)
- $\beta$ ?
  - Na prática  $\beta = 0.8, 0.9$  ( 5 passos em média para saltar)

# Exemplo: ( $\beta = 0.8$ )



$$0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

y	7/15	7/15	1/15
a	7/15	1/15	1/15
m	1/15	7/15	13/15

**A**

y	1/3	0.33	0.24	0.26	7/33
a	1/3	0.20	0.20	0.18	5/33
m	1/3	0.46	0.52	0.56	21/33



# PageRank in Matlab

# Aplicação a uma “pequena rede”

- Consideremos apenas um número reduzido de páginas
  - exemplo:  $N=20$  páginas
- Principais passos:
  - Obter informação das páginas e suas ligações
  - Com base nos links obter  $H$
  - Criar a matriz  $A$  aplicando o método da Google para evitar dead ends e spider traps
  - Aplicar power method
  - Apresentar resultados

# Informação das páginas e suas ligações

- Em casos reais é obtida usando um crawler

- Pode ser simulada facilmente:

$N = 5$

$L = \text{randi}([0,1], N, N)$  % matriz com ligações (1 = ligação)

for  $i=1:\text{length}(L)$

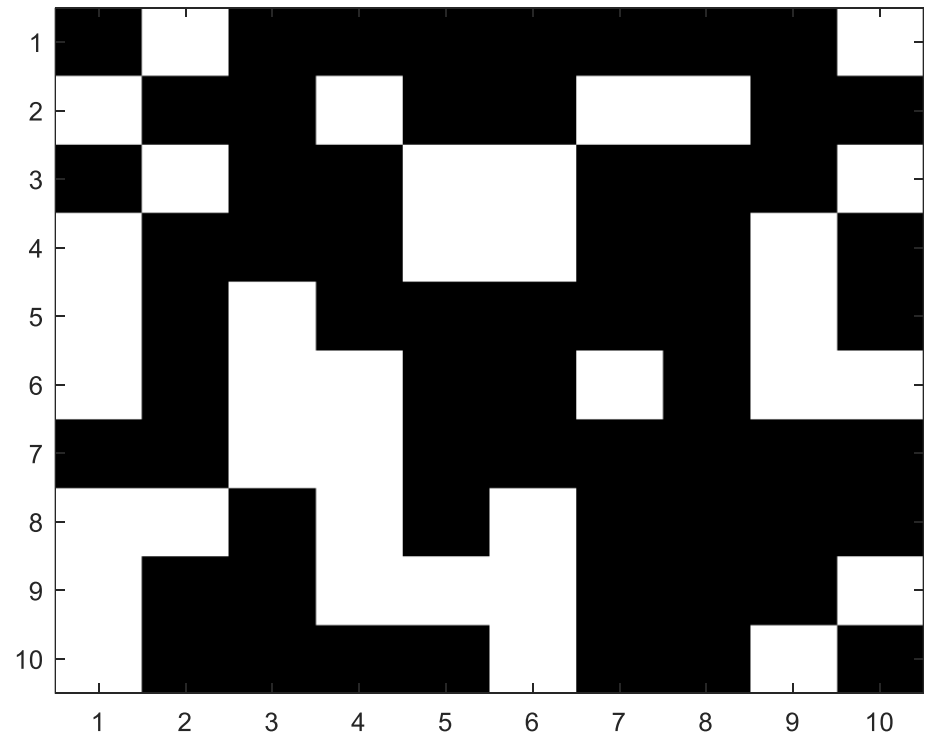
$L(i,i) = 0$  % não contam ligações para a própria página

end

# Resultados exemplificativos

`imagesc(L)`

`colormap(gray)`





# Obter H e A

```
d=sum(full(L)); % n de ligações (d)
```

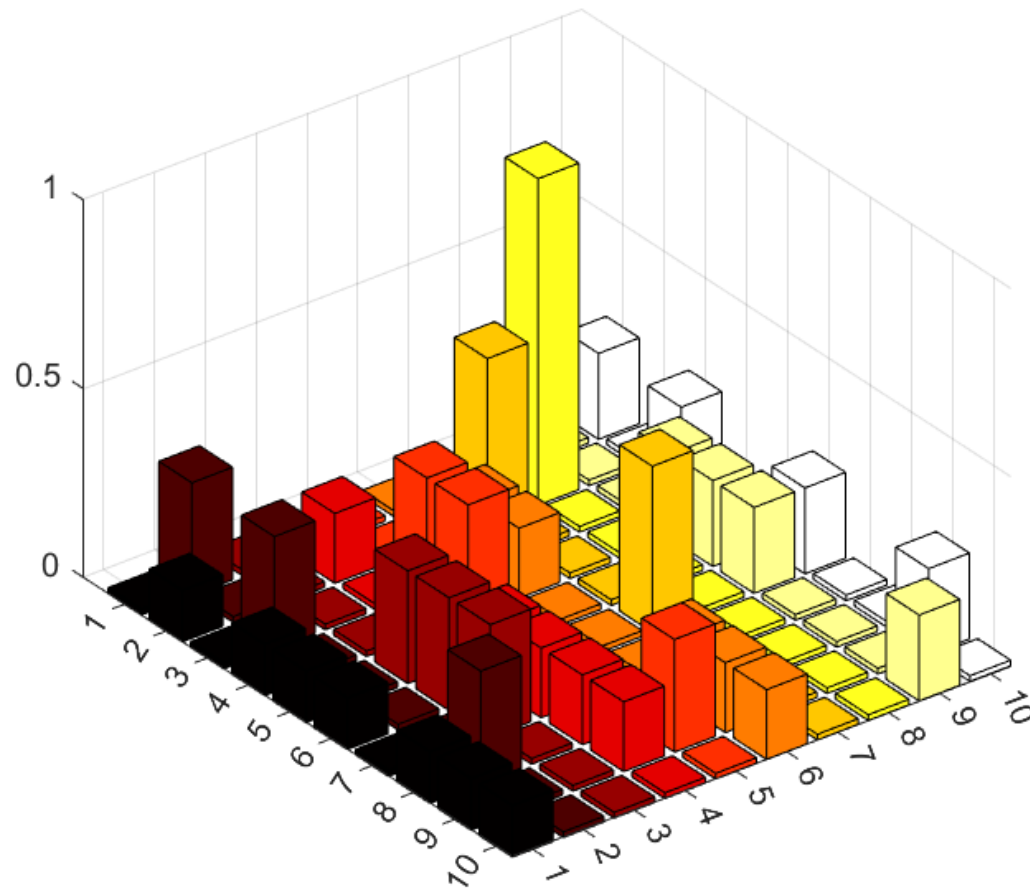
```
H=H./d
```

```
p=0.85
```

```
A=p*H+(1-p)*ones(N)/N % matriz da Google
```

```
A(isnan(A))=1/N % resolver dead ends
```

# A



# Aplicar “power method”

```
pr0=ones(N,1)/N;
```

```
% -----
```

```
iter=1;
```

```
pr=pr0;
```

```
epsilon=1e-3;
```

```
while 1
```

```
    fprintf(1,'iteração %d\n',iter);
```

```
    prold=pr;
```

```
    pr=A*pr;
```

```
    if max(abs(pr-prold))<epsilon break ; end
```

```
    iter=iter+1;
```

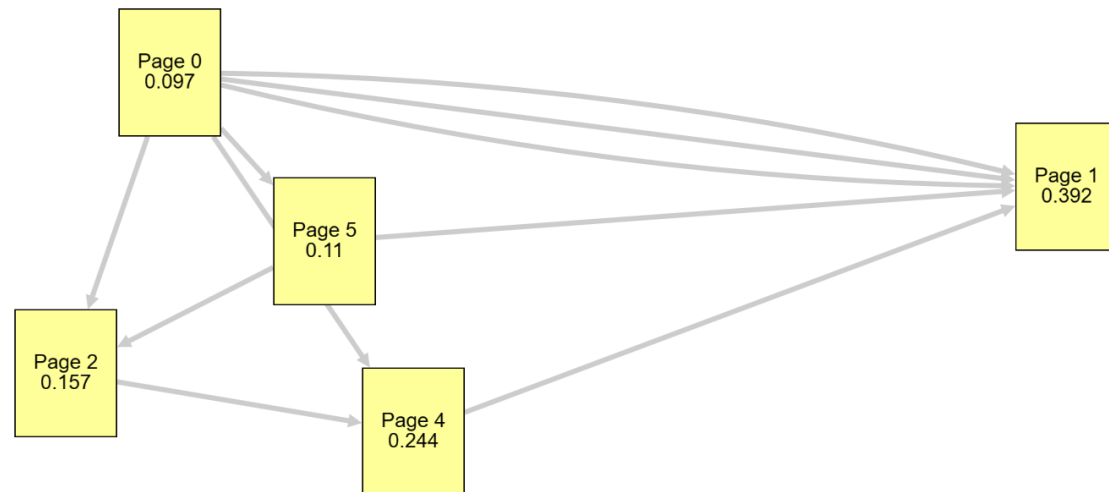
```
end
```

```
pr
```

# “Simulador”

## Page Rank Simulator

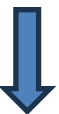
- Click Add Page to add a new page.
- Click and drag a page to move it.
- Click a page and then click another page to add a link.
- Click a page or link and then Delete Selected (or press Delete) to remove something.
- Click Run Page Rank to display rankings.



- <https://computerscience.chemeketa.edu/cs160Reader/static/pageRankApp/index.html>

# E na realidade ?

Apenas 2 ou 3 slides para terem uma ideia, pois começa a sair do âmbito de MPEI



# Calcular para toda a web ...

- O passo mais importante é a multiplicação

$$r^{k+1} = A \cdot r^k$$

- Fácil se desse para ter tudo em memória:  $A$ ,  $r^{k+1}$ ,  $r^k$

- Se  $N = 10^9$  páginas

- 1 bilhão na América, 1000 milhões na Europa

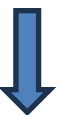
- E considerarmos 4 bytes por entrada

- Temos:

- $2 \times 10^9$  posições para os 2 vectores (aprox. 8GB)

- Uma matriz  $A$  com  $N^2$  elementos

- Ou seja  $10^{18}$

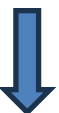


# Partes da solução ...

- **Aproveitar o facto da matriz ser esparsa**
- **Codificá-la com base apenas nas entradas não-nulas**

origem	grau	Nós destino
0	3	1, 5, 7
1	5	17, 64, 113, 117, 245
2	2	13, 23

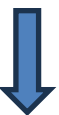
- Espaço proporcional aproximadamente ao número de links
- Por exemplo: 10N, or  $4 \cdot 10^9 \cong 40\text{GB}$
- **Contínua a “não caber” em memória mas cabe em disco**



# Algoritmo básico: Passo de actualização

- Assumindo que  $r^{k+1}$  cabe em memória
  - $r^k$  e a matriz no disco
- 1 passo da iteração do método das potências :

**Inicializar** todas as entradas de  $r^{k+1}$  com  $(1-\beta) / N$   
Para cada página  $i$  (com grau  $d_i$ ):  
Ler para memória:  $i, d_i, dest_1, \dots, dest_{d_i}, r^k(i)$   
Para  $j = 1 \dots d_i$   
 $r^{k+1}(dest_j) += \beta r^k(i) / d_i$





# Exemplo



50

**Inicializar** todas as entradas de  $r^{k+1}$  com  $(1-\beta) / N$

Para cada página ORIGEM  $i$  (com grau  $d_i$ ):

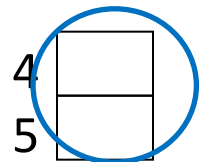
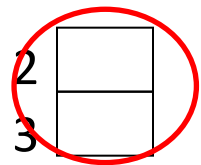
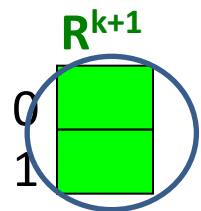
Ler para memória:  $i$ ,  $d_i$ ,  $dest_1, \dots, dest_{d_i}$ ,  $r^k(i)$

Para  $j = 1 \dots d_i$

$$r^{k+1}(dest_j) += \beta r^k(i) / d_i$$

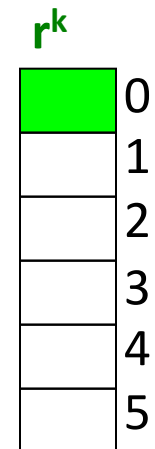
# E se nem $r^{k+1}$ cabe em memória ?

- Dividir  $r^{k+1}$  em  $k$  blocos que caibam em memória
- Processar a Matriz e  $r^k$  uma vez para cada bloco



origem	grau	destino
0	4	0, 1, 3, 5
1	2	0, 5
2	2	3, 4

$H$



# Para saber mais

- [Use PageRank Algorithm to Rank Websites - MATLAB & Simulink Example \(mathworks.com\)](#)
- Capítulo *Link Analysis* do livro **Mining of Massive Datasets**
  - Disponível em <http://infolab.stanford.edu/~ullman/mmds/book.pdf>
- Capítulo *PageRank* do livro **Experiments with MATLAB** de C. Moler
  - e respectivo software
  - [pagerank.pdf \(mathworks.com\)](#)
- Notas do Prof. Paulo Jorge Ferreira “MPEI - pagerank”
  - Disponíveis no elearning
- Artigos dos autores do PageRank e fundadores da Google
  - É uma questão de usar o Google 😊

Nesta apresentação foram usados e/ou adaptados slides da seguinte apresentação:

# Analysis of Large Graphs: Link Analysis, PageRank

Mining of Massive Datasets

Jure Leskovec, Anand Rajaraman, Jeff Ullman Stanford  
University

<http://www.mmds.org>



**Note to other teachers and users of these slides:** We would be delighted if you found this our material useful in giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. If you make use of a significant portion of these slides in your own lecture, please include this message, or a link to our web site: <http://www.mmds.org>