# Exercício programa: Como o Google ordena páginas

# 1 Instruções gerais

Os exercícios computacionais pedidos na disciplina têm por objetivo fundamental familiarizar o aluno com problemas práticos que requeiram técnicas numéricas em sua solução. Seu programa deve ser entregue no e-mail smo@ime.usp.br até o dia **31 de maio**. Não deixe de comentar seu programa, os comentários serão considerados na correção.

# 2 Introdução

Quando fazemos uma procura em um site de busca na internet, como o Google, desejamos obter as páginas que contém um determinado assunto ou palavra-chave, ordenadas em ordem decrescente de prioridade, apesar de não haver uma definição muito clara do que isso significa. O aparecimento do Google no final dos anos 90 foi uma espécie de divisor de águas no que se refere a procura de assuntos na rede. Isto porque, o Google parece sempre colocar os sites mais relevantes primeiro. Com outros sites de procura, muitas vezes era necessário olhar páginas e mais páginas até que os resultados interessantes aparecessem.

O objetivo deste exercício-programa é entender como funciona um site de procura, em particular descreveremos o algoritmo utilizado pelo Google para ordenar as páginas em ordem decrescente de importância.

Um site de procura como o Google basicamente faz 3 coisas:

i) varre a rede e localiza todas as páginas públicas;

ii) indexa os dados de i) em um banco de dados de forma que uma procura por palavra-chave possa ser feita de uma maneira eficiente;

iii) atribui uma importância a cada página do banco de dados de ii), de forma que quando um usuário faz uma procura e o subconjunto das páginas que contém um determinado termo é encontrado, elas podem ser listadas em ordem decrescente de importância;

Como foi dito, descreveremos como iii) acima é feita e implementaremos um algoritmo para realizar tal atribuição de importâncias.

A idéia básica é atribuir pesos positivos às diversas páginas, sendo as com maior peso, as mais importantes.

Para isso, é preciso armazenar a rede como um grafo orientado, onde os vértices são as diversas páginas e as arestas orientadas (setas) representam um link de uma página para outra. Como exemplo, olhemos para a rede abaixo de 4 páginas. A página 1 possui links para as páginas 2,3 e 4. A página 2 para 3 e 4, a página 3 para 1 e a página 4 para 1 e 3.

A importância da p
fbo'agina $i$, onde $i \in \{1, 2, 3, 4\}$ no exemplo anterior, será representada por um número real positivo $x_i$, o peso da página $i$.

A questão a se pensar agora é, como atribuir os $x_i's$?

Existem alguns fatores a se levar em conta, mais precisamente:

1. o fato de uma página ter muitos links apontando para ela a torna mais importante;

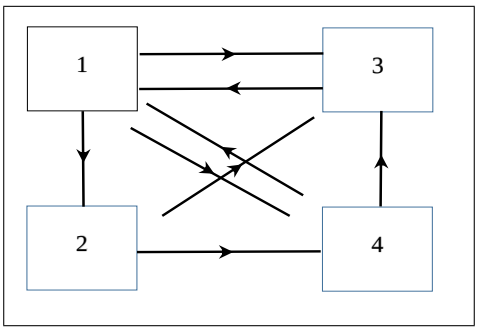


Figura 1: Exemplo 1

2. apesar do que foi dito acima, os links que apontam para uma certa página fixada $P$, em geral têm importâncias diferentes: um link vindo do Yahoo apontando diretamente para $P$ tem muito mais importância que muitos links vindos de páginas de amigos de $P$; em outras palavras, quando uma página que possui muitos links apontando pra ela aponta para $P$, isto deve pesar mais na importância de $P$ que links vindos de páginas pouco apontadas;

3. por fim, um link vindo para $P$ de uma página que aponta pra muitas outras páginas deve ter menor importância que um link vindo de páginas que apontam para poucas;

A motivação para os fatores acima vem do seguinte: uma página $P$ será importante, se um usuário clicando aleatoriamente nos links das páginas que ele encontra, tiver alta probabilidade de acessar $P$.

Vamos agora, através do exemplo 1, descrever como atribuir os pesos de acordo com os fatores qualitativos acima descritos.

$$\begin{cases} x_1 = x_3/1 + x_4/2 \\ x_2 = x_1/3 \\ x_3 = x_1/3 + x_2/2 + x_4/2 \\ x_4 = x_1/3 + x_2/2 \end{cases}$$

Observe que definimos: o peso de uma página como a soma dos pesos das páginas que apontam para ela divididos pelo número de links que saem de cada página. Obtemos um sistema linear que, em forma matricial, é dado por:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}, \tag{1}$$

ou seja transformamos o problema de ranqueamento da importância das páginas na rede num problema de encontrar uma solução para (1).

A equação que queremos resolver tem a forma:

$$Mx = x, \tag{2}$$

onde $M$ é uma matriz $n \times n$ e $x \in \mathbf{R}^n$.

É claro que tal problema só tem solução se 1 for um autovalor da matriz $M$.

Assim, no nosso exemplo, o sistema (1) terá solução se a matriz (chamada matriz de ligação)

$$A = \begin{pmatrix} 0 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{pmatrix} \text{ tiver 1 entre seus autovalores.}$$

Além disso, para que não haja ambiguidade na atribuição dos pesos relativos às diversas páginas, o auto-espaço associado ao autovalor 1 deve ser unidimensional (todos autovetores associados ao autovalor 1 devem ser múltiplos uns dos outros).

Para a rede acima, observamos que o auto-espço associado a 1 é unidimensional, dado pelos múltiplos de (12 4 9 6). Iremos normalizar o autovetor de forma que a soma das suas entradas seja 1. Isto é sempre possível, é só tomarmos um múltiplo correto do vetor encontrado.

No exemplo, obtemos os pesos normalizados:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0.387 \\ 0.129 \\ 0.290 \\ 0.194 \end{pmatrix}$$

Note que a página 3 apesar de muito apontada, não é a mais importante. Isto porque ela aponta somente para a 1, que ainda é apontada pela 4, o que a torna a mais vista por um clicador de links aleatório.

Uma outra propriedade desejada é que os pesos sejam todos positivos. Veremos adiante que a matriz de ligação pode ser construída de maneira a garantir que isto ocorra.

# 3 Um pouco de matemática

Inicialmente mostraremos que 1 sempre é autovalor de uma matriz de ligação. Para isso vamos supor que a rede não possui páginas que não apontam para lugar nenhum (essa hipótese simplificadora fara parte de todo o exercício programa), chamadas páginas mortas. Uma tal página geraria uma coluna de zeros na matriz de ligação e por diversas razões, isto não é interessante.

**Lema:** Se $A$ é uma matriz $n \times n$ com todas entradas positivas, tal que a soma dos elementos de cada coluna é igual a 1, então 1 é autovalor de $A$.

*Prova:*
Esse resultado segue do seguinte. A matriz transposta de $A$ satisfaz:

$$A^t . \begin{pmatrix} 1 \\ 1 \\ . \\ . \\ . \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ . \\ . \\ . \\ 1 \\ 1 \end{pmatrix},$$

logo o vetor com todas as componentes iguais a 1 é autovetor de $A^t$, correspondente ao autovalor 1. Por outro lado, sendo $I$ a matriz identidade,

$$\det(A - \lambda I) = \det(A - \lambda I)^t = \det\left(A^t - \lambda I\right),$$

o que implica que $A$ e $A^t$ têm os mesmos autovalores. □

Como já dissemos, dois fatos importantes nessa análise são os seguintes: o autoespaço associado ao autovalor 1 deve ser unidimensional e deve ser possível escolher o autovetor com a soma das suas entradas igual a 1, sendo todas positivas. Para uma rede qualquer isso nem sempre é verdade (ao menos a parte do auto-espaço unidimensional). Por exemplo, se a rede não for conexa, como no exemplo 2, onde a página 1 aponta para a 2, a 2 aponta para 1, a 3 aponta para a 4, a 4 aponta para 3 e a 5 aponta para 3 e 4. Neste caso o auto-espaço associado ao autovetor 1 da matriz de ligação tem dimensão 2 (verifique!). Isto tem sentido prático, pois não é tão simples comparar as importâncias de páginas em redes separadas.

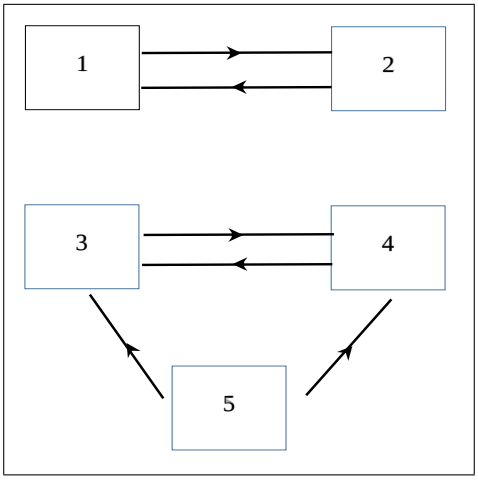


Figura 2: Exemplo 2

Vamos agora enunciar um teorema que será fundamental para que obtenhamos a distribuição de pesos para as páginas conforme desejado.

**Teorema** (Perron-Frobenius, ver [2]) Seja $A$ uma matriz $n \times n$ com todos os elementos positivos ($a_{ij} > 0, \forall i, j, \in [1, ..., n]$) e colunas com soma 1 ($\sum_{i=1}^{n} a_{ij} = 1, \forall j$). Então 1 é o autovalor de maior módulo de $A$, seu auto-espaço é unidimensional e o autovetor normalizado só tem entradas estritamente positivas.

Uma matriz de ligação $A$, em geral, não satisfaz as hipóteses do teorema de Perron-Frobenius, pois pode ter elementos nulos.

Substituímos então $A$ por

$$M = (1 - m)A + mS,$$

onde $S$ é matriz $n \times n$ com todas as entradas iguais a $1/n$ e $0 < m < 1$.

Esta é a estratégia usada pelo Google, que adota $m = 0.15$ e esse será o valor usado no exercício programa. Essa correção feita na matriz $A$ torna todas as entradas de $M$ estritamente positivas e a soma dos elementos em cada coluna continua sendo igual a 1. Uma observação importante é que isso ocorre para todo $0 < m < 1$. Assim, uma perturbação pequena de $A$, tem todas as entradas maiores que zero, com a mesma propriedade de soma 1 ao longo das colunas, satisfazendo as hipóteses do teorema de Perron-Frobenius

Assim, dada uma rede sem páginas mortas, o processo de ranquear as páginas consiste em obter a matriz $A$, depois calcular $M$ e a partir de $M$, achar $x$.

O que veremos a seguir é uma forma muito rápida (computacionalmente) de achar a distribuição de pesos $x$.

# 4 Cálculo do vetor x

Aqui apresentaremos um resultado que naturalmente induz uma forma computacionalmente boa para obtermos o vetor $x$. Antes, necessitamos de algumas definições:

1) dado um vetor $v$ com n componentes, definimos a seguinte norma:

$$\|v\|_1 = \sum_{i=1}^{n} |v_i|$$

2) Para uma matriz $M_{nxn}$ com todas suas entradas positivas, com soma dos elementos em cada coluna igual a 1, definimos

$$c = \max_{1 \leq j \leq n} \left| 1 - 2 \min_{1 \leq i \leq n} M_{ij} \right|.$$

É fácil ver que $0 < c < 1$.

Então temos o seguinte algoritmo para o cálculo do vetor de pesos $x$. Tome um vetor normalizado $x_0$ ($\|x_0\|_1 = 1$) e todo positivo (por exemplo, $x_0 = (1/n, 1/n, ..., 1/n)$), e calcule a sequência $x_k = Mx_{k-1} = M^k x_0$.

Pode-se mostrar que esta sequência de vetores obedece propriedades similares àquelas vistas para o método das aproximações sucessivas (ver [1] que é a

referência na qual este EP foi baseado). Em particular, temos que a sequência de vetores converge para $x$, obedecendo a relação

$$\left\|x - M^k x_0\right\|_1 \leq c^k . \|x - x_0\|_1 \to 0$$

e que

$$\|x - x_k\|_1 \leq \frac{c}{1-c} \|x_k - x_{k-1}\|_1 .$$

A príncipio pode parecer que calcular $M^n x_0$ será computacionalmente muito trabalhoso (pela dimensão de $M$ ser muito grande), mas mostremos que não é assim.

Note que se $y$ é um vetor com todas entradas positivas e $\|y\|_1 = 1$, então

$$z = My = (1-m)Ay + ms,$$

onde

$$s = \begin{pmatrix} 1/n \\ 1/n \\ . \\ . \\ . \\ 1/n \end{pmatrix}.$$

Mais ainda, $z$ tem todas as entradas positivas e $\|z\|_1 = 1$. Assim, de fato o que é preciso fazer repetidas vezes, é o produto de $A$ com um vetor e isto é muito mais simples, pois em geral $A$ será esparsa, terá poucas entradas não-nulas. A maneira como isto deve ser feito para economizar tempo e memória da máquina, será explicada na próxima sessão.

# 5 Implementação

**Tarefa 1**

Você deve escrever um programa para calcular o ranking de importância das páginas da rede mostrada na figura 3, com 8 páginas:

Para tanto, você deve escolher o vetor $x_0 = (\frac{1}{8}, \frac{1}{8}, ..., \frac{1}{8})$, e calcular os vetores $x_{l+1} = (1-m)Ax_l + mx_0$, onde $A$ é a matriz de ligação da rede.

Você deve parar a sua iteração quando $\|x_l - x_{l+1}\|_1 < 10^{-5}$.

A sua saída deve ser um ranking das páginas desta rede e a importância de cada uma (dada pelos pesos calculados).

**Tarefa 2**

Uma rede tem uma arquitetura do tipo **Cacique-tribo** se podemos dividir as páginas em $k$ grupos (ou tribos), onde o grupo $i$ possuiu $a_i$ páginas. As páginas de um grupo apontam para todas as outras páginas do mesmo grupo (e logo são apontadas por todas as páginas da tribo) mas apenas uma delas, o cacique do grupo, aponta para páginas fora deste grupo ou é apontada por

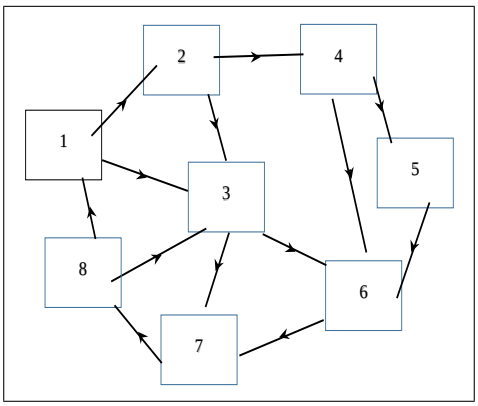


Figura 3: Tarefa 1

páginas de fora da tribo. Os caciques de cada um dos grupos apontam também para todos os outros caciques.

Um exemplo de rede com esta arquitetura é dado na figura 4, onde temos 3 grupos com 2, 2 e 3 páginas respectivamente. As páginas 1, 3, e 5 são os caciques dos grupos.

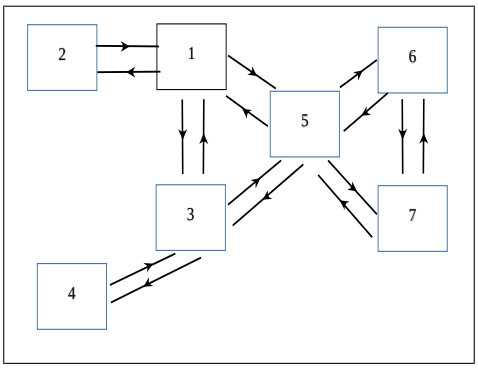


Figura 4: Tarefa 2

Podemos ver que, numa rede com esta arquitetura, cada página do grupo $i$ que não é o cacique aponta para $a_i - 1$ páginas, enquanto que o cacique do grupo $i$ aponta para $a_i - 1 + k - 1$ páginas. O total $T$ de links (ou de entradas não nulas da matriz de transição $A$) é de

$$T = \sum_{i=1}^{k} (a_i - 1)^2 + \sum_{i=1}^{k} (a_i - 1 + k - 1),$$

onde a primeira somatória descreve o total de páginas apontadas pelos índios, e a segunda descreve o total apontado pelos caciques.

Você deve escrever um programa para calcular o ranking de importância das páginas de uma rede com 20 grupos com esta arquitetura, onde o primeiro grupo tem 2 páginas (a página 1, cacique do grupo, e a página 2, índia), o segundo grupo tem 3 páginas (a página 3, cacique, e os "índios"4 e 5), o grupo terceiro grupo tem 4 páginas e assim por diante, até o vigésimo grupo que possui 21 páginas ( a página 210 é o cacique, e os "índios"são as páginas 211 a 230). O total de páginas nesta rede é $n = 230$.

O cacique do $i$-ésimo grupo é a página de número $\frac{i(i+1)}{2}$.

Na sua implementação, você deverá levar em conta que a matriz de ligação é esparsa, isto é, a maior parte das entradas são nulas. Por exemplo, é fácil ver que as entradas $A_{2,j}$ são nulas, se $j > 2$. Por isso, vocês não devem guardar todas as $(230)^2 = 52900$ entradas da matriz, e apenas guardar as entradas não nulas, $T = (\frac{20(21)(41)}{6}) + (210 + 190) = 3270$. Numa rede de dimensões maiores, o ganho na memória a ser utilizada seria ainda mais expressivo.

Uma maneira simples de fazer isto é armazenar 3 vetores, V, C e L, com $T$ elementos cada, contendo os valores das entradas não nulas, bem como as colunas e as linhas das mesmas. Assim, teríamos que $V[s] = a_{L[s],C[s]}$. Desta forma, o cálculo de $y = Ax$ pode ser feito inicializando o vetor $y$ como vetor nulo e calculando, para $1 \leq s \leq T$,

$$y_{L[s]} = y_{L[s]} + V[s]x_{C[s]}.$$

Você deve iniciar a sua iteração com o vetor $x_0 = (\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n})$ e parar o cálculo quando $\|x_l - x_{l+1}\|_1 < 10^{-5}$. Finalmente, você deverá exibir uma lista com as páginas ordenadas pelo seu ranking, bem como a importância de cada uma. Como todas as páginas do tipo índio de um mesmo grupo devem ter a mesma importância (veja a simetria do problema), basta apresentar a lista com um "índio"por grupo. Uma parte importante do trabalho será desenvolver uma rotina para calcular os vetores V,C e L para a matriz $A$.

# Referências

[1] ver o artigo "The 25,000,000,000 eigenvector: the linear algebra behind Google.", que pode ser encontrado em
https://www.rose-hulman.edu/ bryan/googleFinalVersionFixed.pdf

[2] ver http://www.mat.ufmg.br/comed/2005/e2005/perron-frobenius.pdf, página 3 em diante