Google's PageRank

O objetivo deste trabalho é implementar o primeiro algoritmo utilizado pelo Google para ordenar resultados de buscas na Web, o PageRank (https://goo.gl/YRu7FX). A metodologia atual utilizada pelo Google é muito mais complexa que a versão original. No entanto, o PageRank é a base de vários algoritmos utilizados atualmente. Dessa forma, implementar o PageRank de forma eficiente é uma tarefa relevante para o aprendizado de programação, além de desafiadora!

O problema

Quando um usuário digita um termo em uma máquina de busca (e.g., www.google.com), há interesse não apenas que os resultados obtidos sejam relevantes, mas também que os mais relevantes sejam apresentados primeiro. Em outras palavras, dadas as páginas que contêm o termo buscado (ou conteúdo relevante aos termos buscados), em que ordem essas páginas devem ser apresentadas pela máquina de busca?

Nos anos 90, havia um interesse muito grande em resolver esse problema, sendo que várias abordagens foram experimentadas. Por volta de 1997/1998, os criadores do Google propuseram o PageRank, método que serve como base para novos algoritmos até os dias de hoje.

O princípio básico do PageRank consiste no fato de que a importância de uma página Web não depende de seu conteúdo, autor, quem a possui, onde está hospedada, etc., mas sim de quão influente essa página é sobre outras páginas da Web.
Nesse contexto, a influência de uma página sobre outra é dada pela existência de um link da segunda para a primeira.

Por exemplo, suponha que a página principal do jornal The New York Times (www.nytimes.com) possua um link para a página principal da UFV (www.ufv.br), de forma que um visitante de www.nytimes.com possa clicar nesse link e possa então ir para www.ufv.br. Assim, é dito que www.ufv.br influencia www.nytimes.com, ou de forma recíproca, que www.nytimes.com passa parte de sua "autoridade" para www.ufv.br.

Dessa forma, as únicas coisas que o PageRank leva em consideração são um conjunto de páginas da Web (as quais se deseja ordenar) e a estrutura de links entre elas. A Figura 1 ilustra uma situação com 5 páginas. Na figura, uma seta de i para j indica que a página i contém um link para a página j. Alguns comentários:

- A página 0 não influencia nenhuma outra página;
- A página 1 influencia as páginas 0, 3 e 4, sendo desta forma uma página importante na rede;
- A página 2 influencia apenas a página 1. Inicialmente, pode-se pensar que esta é uma página pouco relevante, no entanto, quando uma página i é influente sobre uma outra página influente j, tem-se que i também deve ser influente. Desta forma, a página 2 também é influente na rede.

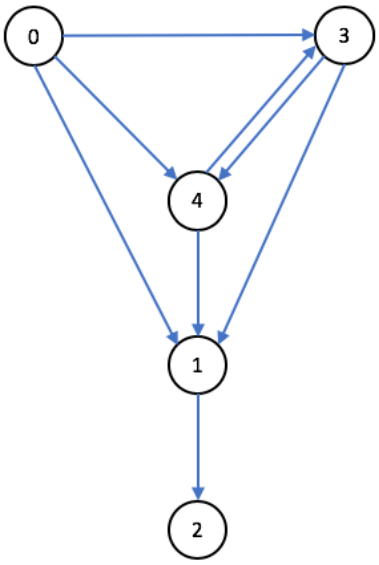

Figura 1: Ilustração de uma rede com cinco páginas

Com a intuição formada com o exemplo acima, no caso de uma rede qualquer, o PageRank da página i é definido pela seguinte fórmula:

$$PR(i) = \begin{cases} \dfrac{1-d}{n} + d \sum\limits_{j\ In(i)} \dfrac{PR(j)}{|Out(j)|}, & se\ |Out(i)| \neq 0 \\ \dfrac{1-d}{n} + d\ PR(i) + d \sum\limits_{j\ In(i)} \dfrac{PR(j)}{|Out(j)|}, & se\ |Out(i)| = 0. \end{cases}$$

Na definição acima:

- $PR(i)$ denota o PageRank da página i;
- n é o número total de páginas;
- d é um parâmetro, entre 0 e 1. Normalmente, d vale 0.85, sendo este o valor a ser usado neste trabalho;
- $In(k)$ é o conjunto de todas as páginas que tem um link para a página k;
- $Out(k)$ é o conjunto de todas as páginas com um link saindo da página k;
- $|Y|$ denota o número de elementos de um conjunto qualquer Y.

Aplicando a definição acima para a rede da Figura 1, tem-se que:

- $PR(0) = \frac{0.15}{5}$;
- $PR(1) = \frac{0.15}{5} + 0.85\left(\frac{PR(0)}{3} + \frac{PR(3)}{2} + \frac{PR(4)}{2}\right)$;

- $PR(2) = \frac{0.15}{5} + 0.85PR(2) + 0.85\left(\frac{PR(1)}{1}\right)$;
- $PR(3) = \frac{0.15}{5} + 0.85\left(\frac{PR(0)}{3} + \frac{PR(4)}{2}\right)$;
- $PR(4) = \frac{0.15}{5} + 0.85\left(\frac{PR(0)}{3} + \frac{PR(3)}{2}\right)$.

Note que nesse caso, e na verdade no caso geral, a definição do valor do PageRank de cada página é cíclica. Ou seja, não há uma fórmula fechada para calculá-la.

Descrição do algoritmo

Para calcular o PageRank, vocês deverão implementar um método iterativo conhecido como *Power Method*. O algoritmo tem um passo de inicialização dos valores do PageRank e então entra em um laço para atualizar tais valores até que convergência seja obtida. Em detalhes:

- Inicialização. No passo zero, o PageRank de cada página é inicializado de maneira uniforme:
$$PR^{(0)}(i) = \frac{1}{n}, i = 0, \dots, n-1.$$
- Laço de atualização. Na iteração k, o valor do PageRank é computado com base nos valores da iteração anterior:
$$PR^{(k)}(i) = \begin{cases} \frac{1-d}{n} + d \sum_{j\ In(i)} \frac{PR^{(k-1)}(j)}{|Out(j)|}, & se\ |Out(i)| \neq 0 \\ \frac{1-d}{n} + d\ PR^{(k-1)}(i) + d \sum_{j\ In(i)} \frac{PR^{(k-1)}(j)}{|Out(j)|}, & se\ |Out(i)| = 0, \end{cases}$$
para $i = 0, \dots, n-1$ e $k > 0$.
Repare que essa equação é similar à definição do PageRank. Mas agora, os valores de PageRank da iteração $k-1$ são utilizados para computar os valores do PageRank da iteração k.
- Término. O processo iterativo de atualização deve ser encerrado quando os valores do PageRank pararem de mudar significativamente. É possível mostrar que isso sempre ocorrerá. De forma prática, o critério de parada será baseado na quantidade a seguir:
$$E(k) = \sqrt{\sum_{i=0}^{n-1} \left(PR^{(k)}(i) - PR^{(k-1)}(i)\right)^2}.$$
Dado $E(k)$, o algoritmo para se $E(k) < \epsilon$, onde ϵ é uma constante pequena, e.g., 10^{-6}. Quando tal fato ocorrer, o valor de $PR^{(k)}(i)$ será tomado como o valor do PageRank da página $i, i = 0, \dots, n-1$.

Considerando os valores de parâmetros descritos acima e o exemplo usando nas seções anteriores (Figura 1), tem-se o exemplo de computação dado no quadro a seguir:

k	$PR^{(k)}$	$E(k)$
0	[0.2 0.2 0.2 0.2 0.2]	-
1	[0.03 0.25666667 0.37 0.17166667 0.17166667]	0.250233224546
2	[0.03 0.18441667 0.56266667 0.11145833 0.11145833]	0.222689455857
...	...	...
17	[0.03 0.09541328 0.74067344 0.06695664 0.06695664]	7.52838631399e-07

Entrada

A entrada será lida da Entrada Padrão. A primeira linha conterá n, o número de páginas na rede. Cada página será rotulada com num número entre 0 e $n-1$. Tem-se que $1 \leq n \leq 10^6$.

As demais linhas indicarão as vizinhanças direcionadas da rede. Cada linha terá o seguinte formato:

a: $b_1, b_2, \ldots, b_l$

indicando que há um link da página a para cada uma das páginas $b_1, b_2, \ldots, b_l$.
É garantido que um computador convencional (8GB de RAM) terá memória suficiente para armazenar a rede toda (se você fizer as escolhas certas) para os exemplos que serão testados.

Exemplo de entrada (referente à Figura 1).

```
5
0:1,3,4
1:2
3:1,4
4:1,3
```

Observação: as funções `getline` e `strtok` do C podem ser úteis na leitura dos dados.

Saída

A saída deve conter n linhas, cada uma com o identificador de uma página. As linhas devem estar ordenadas, em ordem decrescente, pelo valor do PageRank de cada página. Em caso de empate, a página com menor identificador deve aparecer primeiro.

Exemplo de saída para a Figura 1:

```
2
1
3
4
0
```

Regras:

- O trabalho deve ser implementado em C (compilado com *gcc*)
- O trabalho pode ser feito em dupla (duplas desse trabalho não poderão estar no mesmo grupo nos trabalhos 2 e 3)
- A tarefa de ordenação deve utilizar a função `qsort` do C
- **Plágio não será tolerado. O regimento acadêmico será seguido à risca em caso de suspeita**

Critérios de correção

Os seguintes critérios serão considerados:

- Se usar variáveis globais, nota zero;
- Se usar goto, nota zero;
- Se seu trabalho não compilar, nota zero. Se por algum motivo seu código compilar (ou funcionar) no seu computador, mas não no meu, o critério de "desempate" será se o seu trabalho compila (funciona) nos computadores do laboratório CCE 416, considerando o Sistema Operacional Ubuntu. Muita atenção aos usuários de **Windows**!
- Os demais critérios de correção são os seguintes:
    - Fração de respostas corretas;
    - Organização e modularização do código;
    - Legibilidade, i.e., código comentado e nomes intuitivos para as variáveis;
    - Presença de vazamento de memória e acesso a posições inválidas de memória. Use **Valgrind** desde os primeiros testes! Em casos extremos, a não observância desse quesito implicará em nota zero.

Entrega

Cada dupla deve enviar apenas um trabalho para o Email `gcom.tp.sub@gmail.com`, até às 23:59 do dia **14 de Setembro de 2018**. O trabalho deve ser enviado por um e-mail de domínio ufv.br (e-mails de qualquer outro domínio não serão considerados).

O título (*subject*) do e-mail deve conter o número de matrícula dos integrantes do grupo (sem o ES), separados por uma vírgula. Se você optar for fazer o trabalho sozinho, esse campo terá apenas seu número de matrícula.

Se a mesma dupla enviar mais de um trabalho, apenas o e-mail mais recente será considerado.
O e-mail deve ter em anexo apenas um arquivo comprimido, no formato `.tar.gz`, com nome `trab1.tar.gz`, contendo seu código fonte e um `Makefile`.

Após baixar o arquivo, o professor digitará os comandos:

```
tar -xzvf trab1.tar.gz
make
```

Após esses dois comandos, deve ser gerado um executável de nome `trab1`, o qual será utilizado para testes.

Atenção: a conformidade com os critérios aqui estabelecidos faz parte da avaliação. Se você não entregar o trabalho no prazo, ou se o executável não for gerado da forma indicada, você receberá nota zero.