Algoritmos e Estruturas de Dados 2
Trabalho Prático 1
Page Rank
Entrega: 25/09/2015

1 Descrição

A Google é, atualmente, uma das maiores empresas de tecnologia do mundo. Nos tempos iniciais, um dos diferenciais dessa empresa foi um algoritmo conhecido como PageRank. Esse algoritmo é capaz de computar a importância de páginas web de acordo com os links existentes entre as páginas. A intuição básica por trás desse algoritmo é que, se uma página é importante, então existe uma grande probabilidade de um usuário atingí-la ao navegar na Internet. Outra suposição do PageRank é que, se uma página importante possui um link para outra página, essa página também deve ser importante.

Uma das maneiras de modelar esse problema consiste em enxergar as conexões entre as páginas como um grafo direcionado $G(V, E)$, onde V consiste nas páginas (nós) e E consiste nos links entre páginas (arestas). Podemos representar tal grafo com uma matriz de adjacência A de dimensão $|V| \times |V|$. Cada célula A_{ij} assume valor 1 se existe um link da i-ésima página para a j-ésima página, ou 0 caso contrário. A figura 1 mostra um exemplo de grafo direcionado.

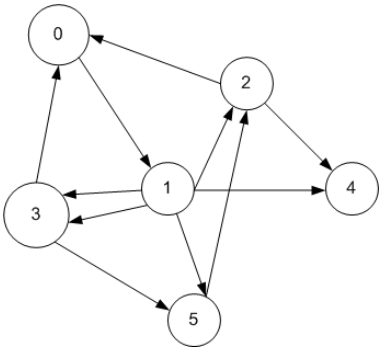

Figura 1: Exemplo de grafo para 6 páginas. A página 0, por exemplo, possui apenas um link para a página 1. No entanto, as páginas 2 e 3 possuem links para a página 0.

Para calcular as importâncias de cada página, o PageRank utiliza uma matriz S estocástica. Sem entrar em grandes detalhes, uma matriz é estocástica quando todas as suas linhas somam 1. Nesse caso, cada célula S_{ij} pode ser interpretada como a probabilidade de sairmos da i-ésima página para a j-ésima página. Uma das propriedades da matriz estocástica é que, ao obtermos $S^2 = S * S$, cada célula S^2_{ij} pode ser vista como a probabilidade de um usuário sair da i-ésima página e chegar na j-ésima página em 2 passos. De maneira geral, podemos obter a probabilidade

de sairmos da i-ésima página e chegar na j-ésima página em n passos calculando S^n. Considerando um número muito grande de passos, a probabilidade de chegarmos na j-ésima página depende somente da importância desta página, e não da página de origem. Assim, para obtermos a importância de uma página, basta calcular S^n para n suficientemente grande, até atingirmos um ponto de convergência. A partir desse ponto, multiplicações posteriores não afetam o resultado. Ou seja, $S^m \approx S^n$ para $m > n$.

Além das transições diretas por meio de links, um usuário pode acessar uma página qualquer independentemente da existência ou não de links para esta. Isso pode ser entendido como a probabilidade do usuário digitar uma URL diretamente no browser (ignorando todos os links existentes na página atual) e é modelado no PageRank por meio de um fator conhecido como *damping factor*. Podemos adicionar essa informação na matriz estocástica descrita anteriormente utilizando a fórmula

$$M_{ij} = (1 - \alpha) * S_{ij} + \alpha * \frac{1}{N} \tag{1}$$

onde α é o damping factor e N é o número de páginas.

2 O que deve ser feito

De maneira resumida, vocês devem implementar o PageRank de acordo com os seguintes passos:

1. Construir matriz de adjacência A lendo da entrada padrão. Caso uma linha de A possua apenas zeros, substituir **todos** os elementos da linha por 1[1].
2. Obter a matriz estocástica S fazendo com que cada linha de A some 1;
3. Construir a matriz M de acordo com a fórmula 1;
4. Obter M^n tal que $n > 2000$ ou $\|M^m - M^n\| \leq 10^{-12}$ para $m > n$, onde:
$$\|B\| = \sum_i \sum_j B_{ij}^2$$
5. Neste ponto, todas as linhas da matriz resultante devem ser iguais e o valor na j-ésima coluna representa a importância da j-ésima página.

2.1 Exemplo

Seja a matriz de entrada:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

[1] Isto é realizado para evitar problemas numéricos na normalização

Uma vez que a última linha possui apenas zeros, substituímos todos os elementos dessa linha por 1, obtendo

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Em seguida, obtemos a matriz estocástica:

$$S = \begin{bmatrix} 0.000000 & 0.333333 & 0.333333 & 0.333333 & 0.000000 \\ 0.500000 & 0.000000 & 0.000000 & 0.500000 & 0.000000 \\ 0.000000 & 0.500000 & 0.000000 & 0.500000 & 0.000000 \\ 0.000000 & 0.000000 & 0.000000 & 0.000000 & 1.000000 \\ 0.200000 & 0.200000 & 0.200000 & 0.200000 & 0.200000 \end{bmatrix}$$

Aplicamos então o damping factor com $\alpha = 0.1$, obtendo a matriz M:

$$M = \begin{bmatrix} 0.020000 & 0.320000 & 0.320000 & 0.320000 & 0.020000 \\ 0.470000 & 0.020000 & 0.020000 & 0.470000 & 0.020000 \\ 0.020000 & 0.470000 & 0.020000 & 0.470000 & 0.020000 \\ 0.020000 & 0.020000 & 0.020000 & 0.020000 & 0.920000 \\ 0.200000 & 0.200000 & 0.200000 & 0.200000 & 0.200000 \end{bmatrix}$$

Finalmente, realizamos multiplicações sucessivas de M até que haja convergência. Após convergência, a matriz obtida foi:

$$M^{16} = \begin{bmatrix} 0.152516 & 0.173974 & 0.119982 & 0.252263 & 0.301264 \\ 0.152516 & 0.173974 & 0.119982 & 0.252263 & 0.301264 \\ 0.152516 & 0.173975 & 0.119983 & 0.252263 & 0.301264 \\ 0.152516 & 0.173974 & 0.119982 & 0.252263 & 0.301264 \\ 0.152516 & 0.173974 & 0.119982 & 0.252263 & 0.301264 \end{bmatrix}$$

Note que todas as linhas são iguais. Nesse ponto, cada coluna mede a importância da página. Para esse exemplo, a última página foi a com maior importância.

3 Entrada e saida

A entrada será realizada através de um arquivo passado como parâmetro para seu programa. A primeira linha deste arquivo contém um inteiro $1 < D \leq 1000$ e um número decimal $\alpha \in (0, 1)$ onde: $D =$ número de paginas (dimensão da matriz quadrada) e $\alpha =$ *damping factor*. As demais linhas são compostas por dois inteiros i e j indicando que a posição i, j da matriz é igual a 1. Isto é, a i-ésima página possui um link para a j-ésima página.

Como exemplo, abaixo temos a entrada correspondente ao exemplo da seção anterior.

```
5 0.1
0 1
0 2
0 3
1 0
1 3
2 1
2 3
3 4
```

A saída utilizada será a saída padrão *stdout*. A importância de cada página i deve ser impressa em ordem (de acordo com seu identificador) em uma única linha. Esta linha deve ser composta por um inteiro i e um decimal r separados por um espaço, onde i é o identificador da página e r a importância da página. Para fins práticos, r deve ser arredondado para quatro casas decimais.

Como exemplo, abaixo temos a saída correspondente ao exemplo da seção anterior.

```
0 0.1525
1 0.1740
2 0.1200
3 0.2523
4 0.3013
```

4 Detalhes de Implementação

Por motivos de precisão, utilize tipo ***double*** para armazenar os elementos da matriz e para o cálculo da norma.

A boa prática de programação indica que o código desenvolvido deve ser legível, bem modularizado e comentado.

- Legível: Nomes intuitivos para funções e variáveis, identação, quebras de linhas adequadas, etc.
- Modularização: O código deve ser decomposto em funções e estruturas semanticamente coesas. Uma sugestão para esse trabalho é criar funções para a multiplicação de matrizes e cálculo da norma.
- Comentários: Trechos relevantes e/ou complexos do código devem ser comentados para facilitar a compreensão do código.

Todas as matrizes utilizadas devem ser alocadas dinamicamente (utilizando *malloc* ou *calloc*, por exemplo). Ao final da execução, todas as estruturas alocadas devem ser liberadas (utilizando *free*).

Uma boa ferramenta para avaliar se a alocação e liberação de memória foi realizada corretamente é o *valgrind*[2].

[2] http://cs.ecs.baylor.edu/~donahoo/tools/valgrind/

5 O que deve ser entregue

Você deve submeter uma documentação de até 10 páginas contendo uma descrição da solução proposta, detalhes relevantes da implementação, além de uma análise de complexidade de tempo dos algoritmos envolvidos e de complexidade de espaço das estruturas de dados utilizadas.

A documentação deve conter:

1. Introdução: descrição sucinta do problema a ser resolvido e visão geral sobre o funcionamento do programa.

2. Implementação: descrição sobre a implementação do programa. Devem ser detalhados as estruturas de dados utilizadas (de preferência com diagramas ilustrativos) e o funcionamento das principais funções.

3. Estudo de complexidade: estudo da complexidade do tempo de execução dos procedimentos implementados, em especial das quatro operações realizadas pelo simulador, e do programa como um todo (notação O).

4. **(Extra)** Avaliação experimental: estudo do comportamento empírico da solução proposta. Vocês deverão analisar o impacto da dimensão do problema no tempo de execução do algoritmo, relacionando-o com a análise de complexidade teórica. Além disso, realizar um estudo[3] do valor de *damping factor* no resultado final do *PageRank* . Essa seção valerá até 2 pontos extras.

5. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.

6. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet, se for o caso.

Utilizando o sistema Prático, você deverá submeter os seguintes arquivos:

- arquivo **.zip**, contendo o seu código fonte (todos os arquivos .c e .h);
- arquivo **.pdf**, contendo a documentação (máximo 10 páginas).

Seu programa será compilado pelo Prático (ambiente Linux) com o seguinte comando:

gcc -Wall -std=c99 *.c -o tp1 -lm

O programa compilado será executado pelo Prático com o seguinte comando:

```
./tp1 entrada
```

em que entrada é o nome do arquivo de entrada.

[3] Dica: Fixar a matriz e variar o *damping factor*

6 Comentários gerais

- Comece a fazer este trabalho logo, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar.
- Clareza, indentação e comentários no programa também vão valer pontos.
- Siga atentamente os formatos de entrada e saída especificados. Fique atento ao número de casas decimais.
- Essa especificação não é isenta de erros e ambiguidades. Portanto, se tiverem problemas para entender o que está escrito aqui, pergunte aos monitores.
- Você **não** precisa de utilizar uma IDE como Netbeans, Eclipse, Code::Blocks ou QtCreator para implementar esse trabalho prático. No entanto, se o fizer,

  não inclua os arquivos extras que foram gerados por essa IDE em sua submissão.
- Trabalhos copiados serão penalizados com a nota zero. Serão utilizadas ferramentas automáticas de detecção de cópias.
- Penalização por atraso: (2d - 1) pontos, em que d é o número de dias de atraso.